

GESELLSCHAFT
FÜR INFORMATIK



Gesellschaft für Informatik (Hrsg.)

SKILL 2022
Studierendenkonferenz Informatik

29. und 30. September 2022
in Hamburg

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Seminars

Series of the Gesellschaft für Informatik (GI)

Volume S 18

ISSN 1614-3213

ISBN 978-3-88579-752-4

Volume Editors

Gesellschaft für Informatik e.V.

Ahrstraße 45

53175 Bonn

E-Mail: bonn@gi.de

Redaktion: Michael Becker

E-Mail: michael.becker@uni-leipzig.de

Series Editorial Board

Andreas Oberweis, KIT Karlsruhe,

(Chairman, andreas.oberweis@kit.edu)

Torsten Brinda, Universität Duisburg-Essen, Germany

Dieter Fellner, Technische Universität Darmstadt, Germany

Ulrich Flegel, Infineon, Germany

Ulrich Frank, Universität Duisburg-Essen, Germany

Michael Goedicke, Universität Duisburg-Essen, Germany

Ralf Hofestädt, Universität Bielefeld, Germany

Wolfgang Karl, KIT Karlsruhe, Germany

Michael Koch, Universität der Bundeswehr München, Germany

Peter Sanders, Karlsruher Institut für Technologie (KIT), Germany

Andreas Thor, HFT Leipzig, Germany

Ingo Timm, Universität Trier, Germany

Karin Vosseberg, Hochschule Bremerhaven, Germany

Maria Wimmer, Universität Koblenz-Landau, Germany

Dissertations

Steffen Hölldobler, Technische Universität Dresden, Germany

Thematics

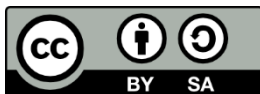
Agnes Koschmider, Universität Kiel, Germany

Seminars

Judith Michael, RWTH Aachen, Germany

© Gesellschaft für Informatik, Bonn 2022

printed by Köllen Druck+Verlag GmbH, Bonn



This book is licensed under a Creative Commons BY-SA 4.0 licence.

Vorwort

Die Studierendenkonferenz Informatik (SKILL) der Gesellschaft für Informatik e.V. ist die jährliche stattfindende Konferenz für Studentinnen und Studenten der Informatik, sowie angrenzender Disziplinen, im deutschsprachigen Raum (D-A-CH). Die Intention des SKILL-Konferenzformats ist es, sehr guten studentischen Arbeiten eine öffentliche Plattform zur Diskussion zu bieten. Die Studierenden können Erfahrungen zum wissenschaftlichen Publizieren sammeln und ihre Ergebnisse vor einem breit gefächerten Publikum vorstellen.

Nach zwei Corona-bedingten Online-Veranstaltungen findet die SKILL 2022 zusammen mit der GI Jahrestagung INFORMATIK 2022 wieder in Präsenz in Hamburg statt. Die Erfahrungen der letzten zehn Jahre zeigen, dass ein Format wie die SKILL wichtig ist, um Studierende für eine wissenschaftliche Diskussion und in letzter Konsequenz für eine wissenschaftliche Arbeit zu begeistern. Die SKILL wird daher auch in Zukunft offen für alle Themen der Informatik bleiben und die gesamte Vielfalt unseres Faches widerspiegeln.

In diesem Jahr wurden insgesamt 44 Beiträge als Full- oder Short-Paper eingereicht und wissenschaftlich begutachtet. In diesem Band erscheinen 13 Beiträge, die auf zwei Konferenztagen durch die Studierenden präsentiert werden. Wie in den Vorjahren ist der Trend zu Beiträgen mit Bezug zu den Themen maschinelles Lernen und KI groß. Aber auch fundierte theoretische Arbeiten sind dieses Jahr mehrfach vertreten.

Die Mitglieder des Organisationskomitees der SKILL 2022 bedanken sich bei den Autorinnen und Autoren, ohne deren hochwertige Beiträge die Konferenz nicht möglich wäre. Wir freuen uns darüber hinaus, dass wir auch in diesem Jahr namhafte Gutachterinnen und Gutachter gewinnen konnten, die den Studierenden mit hilfreichen und ausführlichen Kommentaren zu ihren Arbeiten zur Seite standen.

Leipzig & Berlin, 19. September 2022

Organisationskomitee der SKILL 2022

- Michael Becker, Institut für Angewandte Informatik e.V.
- Thomas Riechert, Hochschule für Technik, Wirtschaft und Kultur Leipzig
- Ludger Porada, Gesellschaft für Informatik e.V.

Inhaltsverzeichnis

Routenberechnung

Lukas Berner	
<i>Generierung und Abdeckung repräsentativer Pfadmengen in Straßennetzwerken</i>	11
Jurek Sander	
<i>Berechnung optimaler Wege im öffentlichen Verkehr</i>	23
Bjarne Valentin Rentz	
<i>Simulation von Microservices</i>	35

Maschinelles Lernen und Informatik in der Anwendung

Moritz Schmidt	
<i>Implementierung und Analyse von Gradientenberechnung in Quantenalgorithmen</i>	49
Robin Kremer	
<i>Ray-Set Classification to Guide Adaptive Light Field Processing</i>	61
Philipp Rall, Nicolas Bender	
<i>Vorhersage von Handbewegungen</i>	73

Short Papers

Osama Hanoun	
<i>TD-Browser</i>	87
Clara Voß	
<i>Identifying Alternatives and Deciding Factors for a Data Mesh Architecture</i>	93

Theoretische Informatik

Jona Dirks, Enna Gerhard
An improved P_3 packing heuristic 103

Florentina Voboril
Computing Treewidth with Constraint Programming 115

Hannes Dröse
Bisecting K -Prototypes 127

Natural Language Processing

Marcel Franzen
Multilinguale Spracheingaben zu Datenbankabfragen 141

Julian Dörenberg
*Comparing Link Grammars and Dependency Grammars for parsing
histological reports* 153

Autor:innenverzeichnis

Routenberechnung

Generierung und Abdeckung repräsentativer Pfadmengen in Straßennetzwerken

Lukas Berner¹

Abstract: Für die Suche nach kürzesten Pfaden in sehr großen Graphen wurden verschiedene Beschleunigungstechniken, wie z.B. Contraction Hierarchies, Hub-Labels oder Transit Node Routing, entwickelt. Um optimale Anfragezeiten und Speicherverbrauch zu erreichen, benötigen viele Beschleunigungstechniken eine Menge *wichtiger* Knoten. In dieser Arbeit wird eine Methode zur Berechnung wichtiger Knoten eines Graphen vorgestellt. Um diese Knoten zu finden, wird auf einer *repräsentativen Pfadmenge* ein Hitting Set Problem mit einem Greedy-Algorithmus gelöst. Die repräsentative Pfadmenge, die möglichst unterschiedliche kürzeste Pfade des Graphen enthalten soll, wird mit einer well-separated pair decomposition und einem Quadtree berechnet. Das Verfahren wurde mit dem deutschen Straßennetzwerk (25M Knoten) getestet und liefert hier einige tausend wichtige Knoten, mit denen bereits etwa 99.9 % aller kürzesten Pfade im Graph abgedeckt sind.

Keywords: Kürzeste Wege; Straßennetzwerke; Well-Separated Pair Decomposition; Greedy Hitting Set

1 Einleitung

Die Suche nach kürzesten Pfaden in Graphen ist ein grundlegendes Problem mit vielen Anwendungen. Der klassische Algorithmus zur Lösung dieses Problems ist der Dijkstra-Algorithmus [Di59], der das Problem in $O(|E| + |V| \log |V|)$ löst. In großen Graphen kann die Suche mit dem Dijkstra-Algorithmus jedoch mehrere Sekunden dauern und ist somit nicht für Echtzeitanwendungen geeignet. Um die Suche nach kürzesten Wegen auch in sehr großen Graphen, wie z.B. Straßennetzen, in wenigen Millisekunden ausführen zu können, wurden verschiedene Beschleunigungstechniken entwickelt. Diese bestehen meist aus einem Vorbereitungsschritt, in dem zusätzliche Informationen berechnet und gespeichert werden und einem modifizierten Suchalgorithmus, der diese Informationen verwendet und damit schneller Lösungen findet. Eine Übersicht findet sich in [Ba16].

Im Vorbereitungsschritt dieser Methoden wird oft eine Ordnung der Knoten im Graph nach ihrer *Wichtigkeit* benötigt. Diese Ordnung hat großen Einfluss auf Speicherbedarf und Suchzeiten dieser Beschleunigungstechnik. Dabei wird die Wichtigkeit eines Knotens üblicherweise über sein Vorkommen in kürzesten Pfaden des Graphen definiert. Zur Berechnung dieser Ordnung gibt es verschiedene Heuristiken. Man kann diese insbesondere

¹ Universität Stuttgart, Institut für Formale Methoden der Informatik, Universitätsstraße 38, 70569 Stuttgart
Lukas.Berner@rub.de

danach unterscheiden, ob die Ordnung lokal während der Berechnungen in der Vorbereitung festgelegt wird, oder (global) im Voraus vorgegeben ist. Ein Beispiel für eine lokale Heuristik wird etwa in [Ge08] für Contraction Hierarchies (CH) vorgestellt. Eine globale Heuristik wird z.B. in Transit Node Routing [Ba07] verwendet.

In dieser Arbeit wird eine weitere Methode zur Berechnung wichtiger Knoten vorgestellt. Diese wichtigen Knoten können dann beispielsweise als globale Heuristik für die Ordnung der Knoten verwendet werden.

Grundsätzlich besteht die hier vorgestellte Methode aus zwei Schritten: zuerst wird eine *repräsentative Pfadmenge* P erzeugt. Diese repräsentative Pfadmenge ist eine breit aufgestellte Sammlung unterschiedlicher kürzester Pfade. Die Berechnung von P verwendet die Einbettung der Knoten in \mathbb{R}^2 zur Berechnung eines Quadrees und eine well-separated pair decomposition (WSPD) [CK95].

Im zweiten Schritt werden aus P wichtige Knoten berechnet. Die Frage nach wichtigen Knoten kann als Hitting Set Problem formuliert werden: finde eine minimale Menge $H \subset V$, sodass jeder Pfad mindestens einen Knoten aus H enthält: $\forall \pi \in P: \pi \cap H \neq \emptyset$. Dieses Problem ist NP-hart; daher wird die Lösung nur mit einem Greedy-Algorithmus approximiert. Um auch sehr große Pfadmengen verarbeiten zu können, wird der Graph einer Contraction Hierarchy verwendet, da Pfade hier mit den Abkürzungskanten der CH beschrieben werden können und somit weniger Speicherplatz benötigen.

Diese Arbeit ist wie folgt aufgebaut: Zunächst werden in Abschnitt 2 relevante Eigenschaften von Contraction Hierarchies und WSPD kurz zusammengefasst. Anschließend folgt die Berechnung der repräsentativen Pfadmengen (Abschnitt 3) und die Berechnung des Hitting Sets (Abschnitt 4). Zuletzt werden einige Versuchsergebnisse am deutschen Straßennetzwerk vorgestellt (Abschnitt 5).

2 Grundlagen

2.1 Contraction Hierarchies

Contraction Hierarchies [Ge08] sind eine Beschleunigungstechnik für die Berechnung kürzester Wege. CHs haben eine Vorbereitungsphase, in der aus einem Graph $G = (V, E_B)$ ein erweiterter CH-Graph $G_{CH} = (V, E_B \cup E_{CH})$ erzeugt wird. In der Vorbereitung wird eine Ordnung der Knoten benötigt. Die Suche erfolgt dann bidirektional auf einem eingeschränkten Graph, der nur Kanten von kleineren zu größeren Knoten bzgl. der Ordnung betrachtet.

Vorbereitungsphase Der CH-Graph G_{CH} wird iterativ erzeugt. Kern der Vorbereitungsphase ist die *Kontraktion* von Knoten. Die Idee ist dabei, Knoten nacheinander (temporär) aus G zu entfernen und bei Bedarf Abkürzungskanten einzufügen, um kürzeste Wege in G

zu erhalten. Alle Knoten werden nacheinander in der Reihenfolge der Ordnung kontrahiert. Abkürzungskanten sind also Kanten, die kürzeste Wege im Graph nicht verändern, aber mehrere Kanten zu einer Kante zusammenfassen und so eine schnellere Suche ermöglichen. Jede Abkürzungskante ersetzt genau zwei andere Kanten.

CH-Pfade entpacken Der Pfad, der in der CH-Suche gefunden wird, enthält möglicherweise Abkürzungskanten. Um aus diesem CH-Pfad alle Knoten des Pfades zu erhalten, können die Abkürzungskanten rekursiv entpackt werden, bis nur noch Basiskanten übrig sind. Daraus ergeben sich dann die Knoten des Pfades.

Pfadrepräsentationen Mit dem CH-Graph können Pfade effizienter gespeichert werden. Verwendet man die Abkürzungskanten in der Beschreibung eines Pfades, wird potenziell weniger Speicher benötigt als bei der Beschreibung nur mit Kanten aus G . Verarbeitet man sehr viele Pfade, ist der gesamte Speicherbedarf aller CH-Pfade und dem CH-Graphen (je nach Implementierung) kleiner als mit einfachen Kanten- oder Knotenlisten und ohne CH-Graph. Allerdings sind in einem solchen CH-Pfad nicht mehr alle Knoten explizit gespeichert und bei Bedarf muss der Pfad entpackt werden, um alle Knoten zu betrachten.

2.2 Well-separated pair decomposition

Die well-separated pair decomposition (WSPD) [CK95] ist eine Methode, mit der ähnliche Abstände zwischen Punkten effizient beschrieben werden können. Dazu sollen Punktpaare, die sich ähnlich sind, zusammengefasst werden.

Die Definition und der Algorithmus zur Berechnung einer WSPD sind aus [Hal1] übernommen.

Definition Sei $A \otimes B = \{\{x, y\} \mid x \in A, y \in B, x \neq y\}$ die Menge aller Punktpaare der Mengen A und B . Diese Menge wird *Paar* von A und B genannt.

Definition 2.1 (Well-separated Pair Decomposition) $W = \{\{A_1, B_1\}, \dots, \{A_s, B_s\}\}$ ist eine well-separated pair decomposition einer Punktmenge $P \in \mathbb{R}^d$ mit Parameter ϵ , wenn gilt:

1. $\forall i : A_i, B_i \subset P$
 2. $\forall i : A_i \cap B_i = \emptyset$
 3. $\cup_{i=1}^s A_i \otimes B_i = P \otimes P$ (die Paare decken alle Punktpaare aus P ab)
 4. $\max(\text{diam}(A_i), \text{diam}(B_i)) \leq \epsilon * \mathbf{d}(A_i, B_i)$ (die Paare sind ϵ -separiert)
- mit $\mathbf{d}(A, B) = \min_{a \in A, b \in B} |a - b|$

Eine well-separated pair decomposition ist eine Zerlegung von P in ϵ -separierte Paare von Teilmengen, sodass jeder Punkt in mindestens einem Paar enthalten ist. Ein Paar ist ϵ -separiert, wenn die Punkte der Mengen jeweils *nah zusammen* und die Mengen voneinander *weit entfernt* sind. Ein Beispiel ist in Abb. 1a zu sehen.

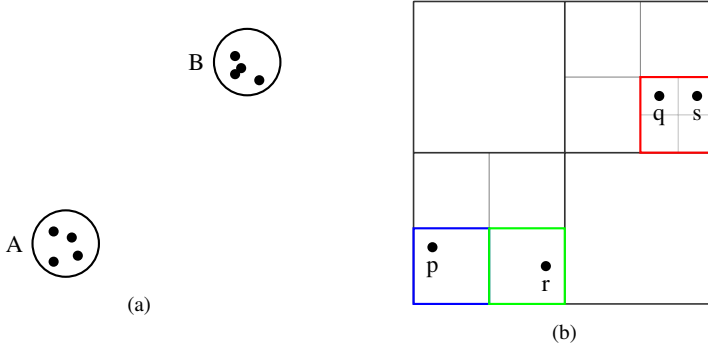


Abb. 1: (a): Zwei Punktmengen A und B , die well-separated sind: Die Punkte der Mengen liegen jeweils nah zusammen und der Abstand zwischen A und B ist im Verhältnis zum Durchmesser groß. (b): Ein Quadtree (mit Tiefe 3). Hier wären z.B. die blaue und rote Zelle oder die grüne und rote Zelle well-separated. Grafik modifiziert nach [Hal1, Abb. 2.1 und Abb. 3.2]

Berechnung Eine WSPD kann mithilfe eines Quadrees berechnet werden, wie am Beispiel Abb. 1b zu sehen ist. Die Knoten des Quadrees (inklusive aller Punkte, die im Teilbaum unter dem Knoten gespeichert sind) werden als Teilmengen für die WSPD verwendet. Eine solche Punktmenge nennen wir Zelle und diese entsprechen einer Zerlegung des Einheitsquadrats in kleinere Quadrate.

Die WSPD wird rekursiv berechnet: Beginnend mit dem Paar (*Wurzel*, *Wurzel*) wird der Baum rekursiv durchlaufen. In jedem Schritt wird geprüft, ob das aktuelle Paar (u, v) schon ϵ -separiert ist. Falls ja, wird dieses Paar zur WSPD hinzugefügt und die Rekursion beendet. Falls nein, wird der größere Knoten in seine vier Kinder geteilt und der Algorithmus rekursiv auf den vier neuen Paaren ausgeführt.

Da jedes Blatt eines Quadrees nur einen Punkt enthält und somit per Definition Durchmesser 0 hat, wird in jeder Rekursion spätestens mit den Blättern ein Paar gefunden. Somit erzeugt der Algorithmus eine Pair Decomposition und jedes Paar ist ϵ -separiert. Also berechnet der Algorithmus eine WSPD. Dies gilt jedoch nur für vollständige Quadrees: Falls ein Blatt mehr als einen Punkt enthält (wie im Fall eines Quadrees, der eine feste maximale Tiefe hat), könnten einige Punktpaare nicht im Ergebnis enthalten sein. Insbesondere sind dann alle Paare aus Punkten, die im gleichen Blatt gespeichert sind, nicht im Ergebnis enthalten. In dieser Arbeit werden jedoch auch solche unvollständigen Ergebnisse des Algorithmus WSPD genannt.

3 Berechnung repräsentativer Pfadmengen

Wie in der Einleitung beschrieben, wird zur Berechnung *wichtiger* Knoten in einem Graph zuerst eine repräsentative Pfadmenge erzeugt. Dazu betrachten wir zunächst, wie Pfade in einem Straßennetzwerk verteilt sind, wie ähnliche Pfade aussehen und wie sich daraus eine repräsentative Pfadmenge ergibt. Anschließend folgt die Berechnung mit der WSPD.

3.1 Pfadmengen und Stichproben

Da es in großen Graphen zu viele Punktpaare und damit zu viele kürzeste Pfade gibt, um alle Pfade betrachten zu können, müssen wir uns mit einer Stichprobe zufriedengeben. Es stellt sich die Frage, wie eine *repräsentative* Stichprobe aussehen soll.

Betrachten wir dazu zuerst eine einfache Methode, eine Stichprobe zu erzeugen und überlegen, welche Probleme sich daraus ergeben: Um diese Stichprobe zu erzeugen, wählt man zufällige Punktpaare und berechnet jeweils den kürzesten Weg zwischen ihnen. Aus der Verteilung der Punkte auf der Karte ergibt sich, dass die so erzeugten Pfade sich oft recht ähnlich sind: Man kann etwa damit rechnen, dass es mehrere Punktpaare geben wird, die jeweils einen Pfad von Stuttgart nach Berlin erzeugen, da es in beiden Städten sehr viele Punkte gibt. Dagegen wird es (wenn die Stichprobe nicht sehr groß ist) wahrscheinlich überhaupt kein Punktpaar geben, das die Verbindung zwischen zwei kleinen Dörfern im Schwarzwald beschreibt, da beide Dörfer jeweils aus nur wenigen Punkten bestehen.

Nun kann man sich die Frage stellen, ob eine solche Pfadmenge *repräsentativ* für alle Pfade im Graph ist. Bezogen auf die Häufigkeit der Pfade ist die Stichprobe sicherlich korrekt. Allerdings könnte man die Häufigkeiten, mit denen *ähnliche* Pfade (also z.B. alle Pfade, die zwischen Stuttgart und Berlin verlaufen) auftreten, auch effizienter beschreiben, indem man jedem Pfad in der Stichprobe ein Gewicht gibt und dieses dann in der weiteren Verwendung berücksichtigt. Speichert man nun also nur noch einen solchen Pfad von Stuttgart nach Berlin mit großem Gewicht, verbleibt deutlich mehr (Speicher-) Platz in der Stichprobe, um auch die weniger häufigen lokalen Verbindungen mit aufzunehmen. Diesen Gedanken wollen wir weiter verfolgen. Es stellt sich die Frage, wann Pfade *ähnlich* sind, wie man möglichst viele nicht-*ähnliche* Pfade findet, und wie man für diese effizient ein Gewicht berechnen kann.

Ähnliche Pfade Eine Beobachtung zu kürzesten Wegen in Straßennetzen ist, dass kürzeste Wege zwischen Punktpaaren, die nah zusammen liegen (die Startpunkte und Zielpunkte beider Paare liegen jeweils nah zueinander), oft einen ähnlichen Verlauf haben: abgesehen von einem Teilpfad am Anfang und Ende verlaufen beide über die gleichen großen Straßen oder Autobahnen. Besonders für Paare mit großem Abstand gilt, dass sich der Weg kaum verändert, wenn man einen Punkt lokal verschiebt - der Großteil des Weges bleibt gleich.

Wir nennen solche Pfade, die sich durch lokale Verschiebungen kaum verändern, *ähnlich*. Aus dieser Definition ergeben sich Pfadmengen, für die gilt, dass sich alle Pfade in der Menge ähnlich sind. Für die repräsentative Pfadmenge reicht es, aus einer solchen Pfadmenge nur einen Pfad auszuwählen und die Anzahl der ähnlichen Pfade als Gewicht zu verwenden.

3.2 Pfadmengen mit WSPD

Aus der Definition der WSPD wissen wir, dass Zellpaare der WSPD im Verhältnis zu ihrer Größe weit voneinander entfernt sind. Daraus ergibt sich, dass kürzeste Pfade zwischen Punktpaaren aus beiden Zellen zueinander ähnlich sein sollten, da die Start- und Zielpunkte nur lokal (im Verhältnis zur Länge des Pfades) verschoben werden. Gleichzeitig sollten sich die Pfade, die sich aus verschiedenen Zellpaaren ergeben, nicht ähnlich sein, da die Start- und Zielpunkte dann eben nicht nur lokal, sondern zumindest bis in eine andere Zelle verschoben werden. Da diese Zerlegung nur geometrisch definiert ist, wird es besonders an den Rändern benachbarter Zellen natürlich auch Punktpaare geben, die sich ähnlich sind.

Wählt man nun für jedes Zellpaar der WSPD einen Pfad aus und gewichtet ihn mit der Anzahl der Punktpaare zwischen den Zellen, erhält man eine gewichtete Pfadmenge, die eine breite Auswahl an verschiedenen Pfaden enthält. Wir nennen diese so erzeugte Pfadmenge eine repräsentative Pfadmenge.

4 Hitting Set

Aus der repräsentativen Pfadmenge wollen wir nun *wichtige* Knoten berechnen. Wie bereits in der Einleitung beschrieben definieren wir *Wichtigkeit* als *Häufigkeit eines Knotens in der Pfadmenge*, basierend auf der Intuition, dass Knoten, die in vielen Pfaden enthalten sind, wichtig sind. Damit ergibt sich dann das Hitting Set Problem:

Definition 4.1 *Hitting Set Problem* Gegeben ein Graph $G = (V, E)$ und eine Pfadmenge P von Pfaden aus G .

Wähle die kleinste Menge von Knoten $H \subset V$, sodass jeder Pfad mindestens einen Knoten aus H enthält: $\forall \pi \in P : \pi \cap H \neq \emptyset$.

Das Problem ist NP-hart und da wir das Problem auf sehr großen Eingaben (etwa 10^7 Pfade) lösen wollen, werden wir nur eine Approximation mit einem Greedy-Algorithmus berechnen. Wir gehen hier davon aus, dass die Pfadmenge gewichtet ist und in CH-Kantendarstellung vorliegt, wie von der Berechnung im letzten Kapitel erzeugt.

Die Idee des Greedy-Algorithmus ist leicht beschrieben: Zähle die Häufigkeit aller Knoten in allen Pfaden. Wählen dann den häufigsten Knoten, füge ihn zum Hitting Set hinzu, und

entferne die von diesem Knoten getroffenen Pfade aus der Eingabe. Wiederhole, bis kein Pfad mehr übrig ist.

Grundsätzlich müssen hier zwei Operationen ausgeführt werden: Im ersten Schritt wird ein Histogramm der Knotenhäufigkeiten berechnet, im zweiten Schritt müssen für einen Knoten alle Pfade gefunden werden, die ihn enthalten. Beide haben ineffiziente naive Lösungen (CH-Pfade nacheinander entpacken und einzeln betrachten); bessere Lösungen werden nun vorgestellt. Um die Diskussion zu vereinfachen, definieren wir zuvor noch einen Graph und eine Operation darauf.

4.1 Abdeckung von Kanten und Knoten im Metagraph

In der Konstruktion des CH-Graphen fällt eine Eigenschaft auf: Eine Abkürzungskante ersetzt immer genau zwei andere Kanten und die ersetzten Kanten kommen in den folgenden Konstruktionsschritten nicht mehr vor. Wir nennen die Abkürzungskante die Elternkante und die ersetzten Kanten die Kindkanten. Aus den Eltern-Kind-Beziehungen aller Kanten ergibt sich ein DAG, den wir den Metagraph G_M nennen. Zusätzlich fügen wir für alle Basiskanten E_B die beiden zugehörigen Knoten als "Kinder" zu G_M hinzu.

Aus G_M lassen sich dann Abdeckungen von Kanten und Knoten ablesen. Betrachtet man den Pfad $\pi = e$, der von einer einzigen Kante $e \in E$ beschrieben wird und schreibt ihn in Knotendarstellung auf, erhält man eine Liste aus mindestens zwei, potenziell aber deutlich mehr Knoten (falls e eine CH-Kante mit vielen rekursiven Kindern ist). Diese Knoten sind die von e *abgedeckten* Knoten. Für eine Kante $e \in E$ können aus G_M alle Knoten abgelesen werden, die von e abgedeckt werden: Die Blätter V' , die von e in G_M erreichbar sind, sind genau die Knoten, die durch e abgedeckt werden.

Anders herum können auch alle Kanten angegeben werden, die einen Knoten $v \in V$ abdecken: Dies sind die Kanten E' , die im inversen Metagraph G_M^{-1} von v aus erreichbar sind. V' und E' können jeweils mit einer einfachen Breitensuche o.Ä. gefunden werden.

Wenn man die Abdeckung mehrerer Kanten (bzw. Knoten) finden möchte, kann die Suche auch von mehreren Kanten (Knoten) aus gleichzeitig gestartet werden. Dann erhält man die Knoten (Kanten), die von mindestens einer Kante (Knoten) abgedeckt werden. In diesem Fall sollte die Suche nach der topologischen Sortierung des Metagraphen priorisiert werden, damit jeder Knoten des Metagraphs maximal ein mal betrachtet werden muss.

4.2 Histogramm berechnen

Ziel dieser Operation ist, die Häufigkeiten aller Knoten zu zählen. Dabei wollen wir die Pfade nicht naiv nacheinander entpacken, sondern strategisch gleichzeitig. Dazu werden zuerst die Häufigkeiten der Kanten in den CH-Pfaden gezählt. Das Gewicht der repräsentativen

Pfade wird dabei mit berücksichtigt. Anschließend werden die CH-Kanten entpackt und ihre Häufigkeiten additiv auf die Kinderkanten übertragen. Diese Operation ist also eine Suche nach den abgedeckten Knoten in G_M , allerdings wird beim Durchlaufen von G_M immer auch die Häufigkeit der Elternkante auf die Kinder übertragen. Wichtig ist die Reihenfolge, in der die Häufigkeiten der Kanten auf ihre Kindkanten übertragen werden: Ist sichergestellt, dass die Werte der Eltern bereits vollständig sind, wenn sie auf die Kinder übertragen werden, dann muss jede Kante insgesamt nur einmal bearbeitet werden. Da G_M ein DAG ist, existiert eine topologische Sortierung und somit eine Reihenfolge, die diese Bedingung erfüllt.

Speedup mit Update statt Neuberechnung In einem Schritt i des Algorithmus wird die Pfadmenge geteilt in die Pfade R_i , die den häufigsten Knoten enthalten, und die Pfade P_i , die ihn nicht enthalten. Gesucht ist nun das Histogramm $H(P_i)$. Dieses kann direkt berechnet werden, oder indirekt über das Histogramm für R_i : $H(P_i) = H(P_{i-1}) - H(R_i)$. Die Differenz lässt sich beim Kantenzählen sehr einfach implementieren - das alte Histogramm wird beibehalten und in dem Schritt, in dem die Häufigkeiten einer Basiskante auf den zugehörigen Knoten übertragen werden, werden diese Häufigkeiten vom alten Wert abgezogen. Je nachdem, welche Pfadmenge größer ist, ist die Berechnung des entsprechenden Histogramms schneller und wird verwendet.

4.3 Getroffene Pfade finden

Das Problem, das hier gelöst werden muss, lässt sich wie folgt formulieren: Gegeben ein Knoten v (üblicherweise der Häufigste aus dem Histogramm). Welche Pfade enthalten diesen Knoten? Dabei liegen die Pfade noch immer in CH-Kantendarstellung vor. Um dieses Problem platzsparend zu lösen, verwenden wir wieder die Abdeckungen in G_M .

In Abschnitt 4.1 wurde bereits gezeigt, dass alle Kanten, die einen Knoten abdecken (also als Pfad diesen Knoten enthalten), in G_M^{-1} von diesem Knoten aus erreichbar sind. Damit können also alle Pfade gefunden werden, die einen Knoten enthalten. Um alle Pfade zu finden, die eine dieser abdeckenden Kanten enthalten, wird in einem Vorbereitungsschritt zunächst an jeder Kante gespeichert, welche Pfade sie enthalten. Der benötigte Speicherplatz entspricht der Anzahl der Kanten in der CH-Kantendarstellung der Pfade, denn nur an diesen Kanten wird die Information zum entsprechenden Pfad gespeichert. Diese Vorbereitung hat lineare Laufzeit in der Größe der CH-Kantendarstellung.

Um nun alle Pfade für einen Knoten v zu finden, werden alle von v aus in G_M^{-1} erreichbaren Kanten betrachtet und alle Pfade, die an einer dieser Kanten gespeichert sind, ausgegeben.

5 Ergebnisse und Auswertung

Die Methoden der beiden vorherigen Kapitel wurden am Beispiel des deutschen Straßennetzes getestet.

Datengrundlage der Auswertung ist das deutsche Straßennetzwerk aus OpenStreetMap [Op17]. Das Straßennetzwerk wird in einen Graph umgewandelt und aus dem Graph anschließend mit dem CHConstructor [NB17] ein CH-Graph berechnet. Dieser CH-Graph wird dann für die Berechnung der repräsentativen Pfadmenge und des Hitting Sets verwendet. Einige Auswertungen in diesem Kapitel beinhalten auch Messungen zu Laufzeiten; diese Ergebnisse wurden mit einem Intel i5-9500 und 64 GB Arbeitsspeicher gemessen.

Zur Berechnung der repräsentativen Pfadmenge müssen zwei Parameter gewählt werden: Die maximale Tiefe des Quadrees, im Folgenden immer d genannt, und die Separationskonstante $\epsilon \in (0, 1]$ der WSPD. Ein tieferer Baum zerlegt die Karte in kleinere Zellen und erlaubt damit auch kürzere, lokalere Pfade in der repräsentativen Pfadmenge. Für das deutsche Straßennetzwerk wählen wir $d \in [8, 12]$; die kleinsten Zellen haben dann Kantenlängen von ca. 150 m bis 3 km.

Die Ergebnisse und Laufzeiten ausgewählter Parameter sind in Tab. 1 aufgezählt. Erste Versuche haben ergeben, dass das Testsystem mit 64GB Speicher etwa 60M Pfade verarbeiten kann. Daher sind hier Parameterpaare gewählt worden, bei denen ungefähr 60M Pfade erzeugt werden. Die in der Tabelle angegebene Laufzeit ist nur die Zeit, die zur Berechnung der WSPD und des Hitting Sets benötigt wurde. Die Zeit zur Berechnung der kürzesten Pfade zwischen den WSPD-Paaren (mit einer CH-Suche) ist nicht angegeben und liegt bei etwa 2ms pro Pfad (parallel berechnet etwa 5.5h für 60M Pfade).

Parameter		P	Covering Error	HS	Lower Bound	Laufzeit	Pfadabdeckung		
d	ϵ						99 %	99.9 %	99.99 %
8	0.06	$63 * 10^6$	2.838 %	2505	957	11 min	379	837	1315
9	0.15	$49 * 10^6$	0.185 %	22978	10747	12 min	829	3444	8062
10	0.25	$59 * 10^6$	0.024 %	104322	60479	42 min	962	5595	18926
11	0.45	$60 * 10^6$	0.004 %	382127	248725	106 min	1051	6719	29394
12	0.9	$59 * 10^6$	0.001 %	1190699	844573	320 min	1136	7572	37624

Tab. 1: Für eine Auswahl von Parameterpaaren: Größe der repräsentativen Pfadmenge P und Anteil der Punktpaare, die nicht repräsentiert sind (Covering Error), Größe des Hitting Sets (HS) und Abschätzung der minimalen Größe des Hitting Sets (Lower Bound), Berechnungszeit (ohne kürzeste Wege), und Größe eines Hitting Sets, das 99 %, 99.9 %, bzw. 99.99 % aller gewichteten Pfade in P abdeckt (Pfadabdeckung).

Repräsentative Pfadmenge In Abb. 2 sehen wir, dass die Größe der Pfadmenge mit einem tieferen Quadtree und mit kleinerem ϵ steigt, da die Punkte in mehr kleinere Zellen aufgeteilt werden und die WSPD durch die stärkere Separationsbedingung aus Paaren kleinerer Zellen besteht. Für sehr kleine ϵ sinkt die Größe der WSPD dann wieder, da immer

weniger Zellpaare die Separationsbedingung erfüllen können.

Außerdem steigt auch der Abdeckungsfehler, also der Anteil der Punktpaare, die nicht in der WSPD enthalten sind, mit kleinerem ϵ , da immer mehr Zellpaare selbst auf der tiefsten Ebene des Quadrtrees die Separationsbedingung nicht mehr erfüllen können. Selbst für $\epsilon = 1$ ist ein kleiner Abdeckungsfehler vorhanden, da die Punktpaare, die in einem gemeinsamen Blatt des Baumes liegen, nicht abgedeckt werden. Bei flachen Bäumen ist der Abdeckungsfehler kleiner als bei tiefen Bäumen, da hier mehr Punkte in einem Blatt gespeichert sind.

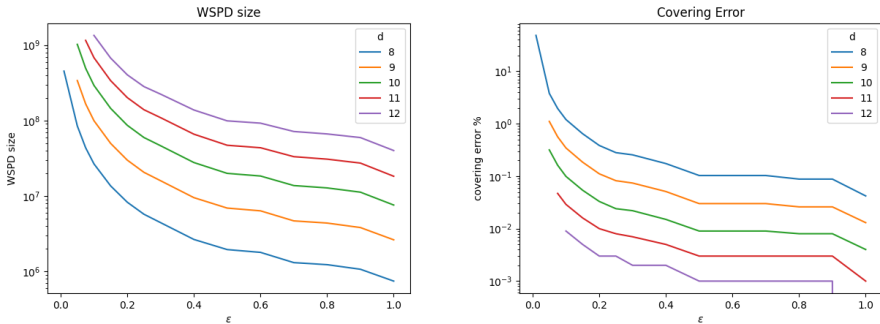


Abb. 2: Links: Größe der WSPD (und damit der Pfadmenge) für verschiedene Parameter d und ϵ . Rechts: Abdeckungsfehler für verschiedene Parameter.

Hitting Set Im Verlauf der Berechnung steigt mit jeder Iteration der gewichtete Anteil der Pfade, die vom aktuellen (unvollständigen) Hitting Set abgedeckt werden. In Abb. 3 ist der Verlauf zu sehen: Anfangs werden sehr schnell die meisten Pfade abgedeckt; anschließend werden sehr viele weitere Knoten gefunden, die jeweils nur eine kleine Anzahl neuer Pfade abdeckt. In Tab. 1 sehen wir genauer, dass nur einige hundert bis zehntausend Knoten benötigt werden, um bereits 99 % - 99.99 % der Pfade im Graphen abzudecken.

Die Größen der vollständigen Hitting Sets unterscheiden sich zwar je nach Parameterwahl deutlich voneinander, aber trotzdem sind die meisten Pfade bei allen Parametern schon früh, und ungefähr gleichzeitig, abgedeckt. Der Unterschied liegt hauptsächlich darin, wie viele disjunkte Pfade (bzw. Pfade mit sehr wenigen Überschneidungen) gegen Ende der Berechnung übrig sind, die dann jeweils einen eigenen Knoten im Hitting Set erhalten. Das Verhalten lässt sich mit der Wahl von ϵ erklären. Wie bereits beschrieben, werden die Zellen, die die Separationsbedingung erfüllen, mit kleinerem ϵ kleiner. Der Abstand zwischen den Zellen und damit auch die Länge der Pfade wächst daher mit kleinerem ϵ . Längere Pfade haben eine größere Wahrscheinlichkeit, andere lange Pfade zu schneiden (besonders in Straßennetzen, da hier zusätzlich Fernstraßen hinzukommen) und somit wird das Hitting Set kleiner.

In Tab. 1 ist neben der Größe des Hitting Sets auch noch eine Abschätzung der unteren

Schranke für die Größe angegeben. Da es keine allgemeine untere Schranke für das Hitting Set Problem gibt, wird hier eine Abschätzung für die spezifische Problem Instanz angegeben. Diese Abschätzung wird durch iteratives entfernen von einem Pfad mit allen Pfaden, die ihn schneiden, berechnet. Wir sehen, dass die Lösung des Greedy-Algorithmus besonders für große Lösungen recht nah an der unteren Schranke liegt.

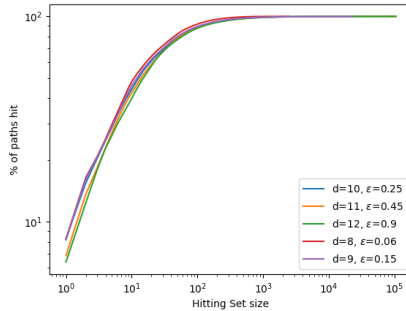


Abb. 3: Anteil der Pfade, die im Verlauf der Hitting Set Berechnung abgedeckt werden.

6 Zusammenfassung und Ausblick

Zusammenfassend lässt sich feststellen, dass die hier vorgestellten Methoden zur Berechnung einer repräsentativen Pfadmenge und wichtiger Knoten qualitativ gute Ergebnisse liefern. Der Algorithmus zur Berechnung des Hitting Sets kann Probleme im Bereich von 10^7 Pfaden schnell berechnen und ist hier nur durch den Speicherbedarf limitiert. Die Berechnung der repräsentativen Pfadmenge mit einer WSPD scheint vielversprechend; in der recht einfachen Version mit einem Quadtree erhalten wir bereits Pfadmengen, die unsere Vorgaben weitgehend erfüllen. Diese Berechnung lässt sich gut parallelisieren und kann somit auch auf deutlich größeren Problemen noch Ergebnisse liefern. Am Beispiel des deutschen Straßennetzes erhalten wir eine sehr kleine Menge von etwa eintausend bis zehntausend Knoten, die bereits 99.9 % aller kürzesten Pfade im Graph abdecken.

Aus der geometrischen Zerlegung der Punktmenge mit dem Quadtree ergeben sich jedoch auch noch offene Fragen: Die Zerlegung geht implizit davon aus, dass Knoten, die geometrisch nah zusammen liegen, immer auch ähnliche kürzeste Pfade zu anderen Knoten haben. Am Beispiel einer Autobahnbrücke sieht man jedoch, dass diese Annahme nicht immer gilt. Wie gut ist diese Zerlegung, wenn man bedenkt, dass in einem Straßennetz nicht alle nah zusammen liegenden Punkte auch fast gleiche kürzeste Wege haben? Der Frage, wie groß dieser Unterschied zwischen geometrischer Zerlegung und echter Graphstruktur ist, gehe ich in meiner Masterarbeit [Be22] unter dem Stichwort *geometrische Abweichung* nach, komme dort jedoch zu keinem eindeutigen Ergebnis.

Außerdem stellt sich natürlich die Frage nach der Qualität der Lösung in der weiteren Verwendung. Offensichtlich ist hier die Verwendung in Beschleunigungstechniken, die eine Menge wichtiger Knoten verwenden. Besonders beim Transit Node Routing würde sich eine Verwendung anbieten. Versuchsreihen zum Vergleich der hier vorgestellten Methode mit anderen Heuristiken sind ein naheliegendes nächstes Projekt.

Möchte man Varianten der hier vorgestellten Methode betrachten, dann stellt sich die Frage, ob sich durch die Verwendung einer anderen Baumstruktur statt des Quadrees zur Berechnung der WSPD bessere repräsentative Pfadmengen ergeben. Gibt es eine Baumstruktur, die die Struktur des Graphen besser beschreibt oder keinen Umweg über die geometrische Zerlegung machen muss? Auch wäre ein Baum, der nicht auf der Einbettung des Graphen in \mathbb{R}^2 basiert, für die Anwendung auf andere Graphtypen nützlich.

Literatur

- [Ba07] Bast, H.; Funke, S.; Matijevic, D.; Sanders, P.; Schultes, D.: In Transit to Constant Time Shortest-Path Queries in Road Networks. In: 2007 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX). S. 46–59, 2007.
- [Ba16] Bast, H.; Delling, D.; Goldberg, A.; Müller-Hannemann, M.; Pajor, T.; Sanders, P.; Wagner, D.; Werneck, R.F.: Route planning in transportation networks. In: Algorithm engineering. Springer, S. 19–80, 2016.
- [Be22] Berner, L.: Generierung und Abdeckung repräsentativer Pfadmengen in Straßennetzen, Masterarbeit, Universität Stuttgart, 2022.
- [CK95] Callahan, P. B.; Kosaraju, S. R.: A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. Journal of the ACM (JACM) 42/1, S. 67–90, 1995.
- [Di59] Dijkstra, E. W. et al.: A note on two problems in connexion with graphs. Numerische mathematik 1/1, S. 269–271, 1959.
- [Ge08] Geisberger, R.; Sanders, P.; Schultes, D.; Delling, D.: Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In: International Workshop on Experimental and Efficient Algorithms. Springer, S. 319–333, 2008.
- [Ha11] Har-Peled, S.: Geometric approximation algorithms. American Mathematical Soc., 2011.
- [NB17] Nusser, A.; Bühler, S.: chconstructor, <https://theogit.fmi.uni-stuttgart.de/nusserae/chconstructor>, 2017.
- [Op17] OpenStreetMap contributors: Planet dump retrieved from <https://planet.osm.org>, <https://www.openstreetmap.org>, 2017.

Berechnung optimaler Wege im öffentlichen Verkehr

Jurek Sander¹

Abstract: In dieser Arbeit stellen wir einen neuen Algorithmus zur Berechnung optimaler Wege in öffentlichen Verkehrsnetzen vor, der auf dem Round-Based Public Transit Routing (RAPTOR) Algorithmus von Delling et al. aus [Delling et al., Transportation Science, 2015] basiert. Im Gegensatz zu den meisten bestehenden Arbeiten wählen wir als Optimalitätskriterium nicht die planmäßige, sondern die erwartete Ankunftszeit. Wir berücksichtigen somit mögliche Verspätungen durch die Definition eines geeigneten Wahrscheinlichkeitsmodells und erreichen dadurch eine deutlich höhere Planungssicherheit. Unser Algorithmus ist darüber hinaus in der Lage, weitere Kriterien wie beispielsweise die maximale Anzahl der Umstiege in der Berechnung optimaler Routen zu berücksichtigen und ist deshalb flexibler einsetzbar als der einzige bereits bekannte Algorithmus dieser Art, der in [Dibbelt et al., ACM J Exp. Alg. 23, 2018] vorgestellt wurde.

Keywords: Fahrplan; Öffentlicher Verkehr; Erwartete Ankunftszeit; Verspätung; Entscheidungsgraph

1 Einleitung

Bei der Benutzung des öffentlichen Regional- und Fernverkehrs spielt die Routenplanung eine entscheidende Rolle. In der klassischen Routenberechnung werden Reisen beispielsweise mit dem Connection Scan Algorithm (CSA) oder RAPTOR-Algorithmus auf der Basis der geplanten Verbindungen hinsichtlich ihrer Fahrtzeiten optimiert. In der Realität kann es jedoch zu Verspätungen der Verkehrsmittel kommen, die sich auf Reisen der Passagiere auswirken. Daher wollen wir sie unter Berücksichtigung möglicher Verspätungen optimieren. In diesem Bereich gibt es in der Forschung bisher wenige Ansätze, da lediglich für die Art der CSAs durch Dibbelt et al. (2018) in [Di18] bereits eine Optimierung der erwarteten Ankunftszeiten durchgeführt wurde. Für die RAPTOR-Algorithmen gibt es noch keine Lösung für diese Problemstellung und es wurde noch nicht untersucht, inwiefern weitere Optimalitätskriterien im Zusammenhang mit den erwarteten Ankunftszeiten berücksichtigt werden können.

Im Bereich der Routenplanung innerhalb von öffentlichen Verkehrsnetzen wurden bereits unterschiedliche Herangehensweisen erforscht und für diese Algorithmen entwickelt. Verkehrsnetze können als Graphen dargestellt werden, wodurch es möglich ist, die Berechnungen mit einem modifizierten Dijkstra-Algorithmus aus [Di59] durchzuführen. Dieser ist jedoch deutlich unflexibler und ineffizienter als Algorithmen, die konkret für öffentliche Verkehrsnetze entwickelt werden. In [BGM10] verbessern Berger et al. (2010) mit ihrer

¹ Universität Stuttgart, Institut für Formale Methoden der Informatik, Universitätsstraße 38, 70569 Stuttgart, Deutschland, st162270@stud.uni-stuttgart.de

Methode die Verwendung des Dijkstra-Algorithmus, indem sie untere Grenzen verwenden. Delling et al. (2012) haben in [DKP12] ihren Self-Pruning Connection-Setting (SPCS) Algorithmus eingeführt. Auch dieser basiert auf der Darstellung der öffentlichen Verkehrsnetze als Graphen, aber er sucht nicht ausschließlich nach der schnellstmöglichen Route für Passagiere, sondern gibt unterschiedliche Möglichkeiten innerhalb eines gegebenen Zeitintervalls an. Der Algorithmus verwendet Vereinfachungsmethoden und besitzt zudem eine parallelisierbare Variante. Um Reisen innerhalb eines öffentlichen Verkehrsnetzes nach mehr als nur einem Kriterium zu optimieren, wurde der Layered-Dijkstra-Algorithmus von Brodal et al. (2004) in [BJ04] eingeführt. Dieser ermöglicht die Verwendung eines zweiten Kriteriums auf der Basis des Dijkstra-Algorithmus. Der Multi-Label-Correcting (MLC) Algorithmus von Pygra et al. (2008), der in [Py08] und [DW09] vorgestellt wurde, basiert ebenfalls auf dem Dijkstra-Algorithmus, aber ermöglicht die Optimierung von mehr als zwei Kriterien. Er wurde von Disser et al. (2008) in [DMS08] verbessert. Die CSAs, die von Dibbelt et al. (2018) in [Di18] vorgestellt wurden, basieren nicht auf dem Dijkstra-Algorithmus. Für sie werden aus den Fahrplandaten jeweils für zwei aufeinanderfolgend angefahrenen Haltestellen der Verkehrsmittel Verbindungen erstellt. Die Algorithmen haben gemeinsam, dass es während ihrer Ausführung ausreicht, einmalig über die Menge der Verbindungen zu iterieren. Mit ihnen können Problemstellungen der Routenberechnungen innerhalb von öffentlichen Verkehrsnetzen, die auf den geplanten Ankunftszeiten basieren, gelöst werden. Zudem wurde für die CSAs eine Variante entwickelt, mit der mögliche Verspätungen über erwartete Ankunftszeiten bei der Optimierung der Routen berücksichtigt werden können. Deren Ergebnisse werden über Entscheidungsgraphen visualisiert, die dem Passagier alternative Reisen anzeigen, falls er durch Verspätungen von seiner ursprünglichen Route abweichen muss. Auch die RAPTOR-Algorithmen, die von Delling et al. (2015) in [DPW15] entwickelt wurden, basieren nicht auf dem Konzept des Dijkstra-Algorithmus. Im Gegensatz zu den CSAs sind die Ausführungszeiten der RAPTOR-Algorithmen etwas langsamer, aber mit ihnen ist es deutlich einfacher umsetzbar zusätzliche Optimalitätskriterien zu verwenden. Zudem sind diese im Gegensatz zu den CSAs leicht parallelisierbar.

Uns ist noch kein Algorithmus der RAPTOR-Familie bekannt, der mögliche Verspätungen der Verkehrsmittel berücksichtigt. Um die Vorteile dieser Art an Algorithmen mit diesem Optimalitätskriterium zu verbinden, führt unser neu entwickelter Algorithmus eine Optimierung der erwarteten Ankunftszeiten durch. Mit ihm ist es möglich weitere Kriterien, wie beispielsweise die maximale Anzahl an Umstiegen zu berücksichtigen, die ein Passagier auf seiner Reise zwischen Start- und Zielpunkt vornehmen muss. Dies liefert alternative Ergebnisse, die komfortablere Reisen darstellen können, aber trotzdem mögliche Verspätungen über die erwarteten Ankunftszeiten berücksichtigen.

2 Grundlagen

Als Grundlage für unseren neuen Algorithmus benötigen wir einige Definitionen, die Problembeschreibungen und den grundlegenden Aufbau der RAPTOR-Algorithmen.

2.1 Definitionen

Die nachfolgenden Definitionen verwenden wir analog zu Delling et al. (2015) in [DPW15]. Innerhalb eines öffentlichen Verkehrsnetzes stellen Haltestellen Knotenpunkte dar. An diesen Stopps können Passagiere in Verkehrsmittel einsteigen, aussteigen oder zwischen ihnen umsteigen. Verkehrsmittel fahren auf fest definierten Routen. Eine Route stellt dabei eine Teilmenge der Menge der Stopps dar. Für diese ist innerhalb der Route eine feste Reihenfolge definiert, in der sie abgefahren werden. Für jede Route existieren Trips. Diese definieren die Ankunfts- und Abfahrtszeiten von Verkehrsmitteln an den Stopps der Route. Somit besitzt jeder Trip t' einer Route r an jedem Stopp p von r eine Ankunftszeit $\tau_{\text{arr}}(t', p)$ und Abfahrtszeit $\tau_{\text{dep}}(t', p)$. Für diese gilt $\tau_{\text{arr}}(t', p) \leq \tau_{\text{dep}}(t', p)$. Die Ankunftszeit des ersten Stopps einer Route und die Abfahrtszeit des letzten Stopps sind undefiniert. Für die Trips einer Route gilt zudem, dass sie sich gegenseitig nicht überholen dürfen. Um Wege von Passagieren im Verkehrsnetz beschreiben zu können, benötigen wir Streckenabschnitte. Jeder Streckenabschnitt l ist einem Trip l_{trip} zugeordnet und definiert sich über den Einstiegspunkt $l_{\text{dep_stop}}$ und Ausstiegspunkt $l_{\text{arr_stop}}$ des Passagiers und deren Zeiten innerhalb des Trips. Die Reise j eines Passagiers zwischen dem Startstopp s und dem Zielstopp t innerhalb des Verkehrsnetzes kann nun über die Sequenz an Streckenabschnitten $j = l^0, l^1, \dots, l^k$ mit $l^0_{\text{dep_stop}} = s$ und $l^k_{\text{arr_stop}} = t$ definiert werden. Sie besitzt die Abfahrtszeit $j_{\text{dep_time}}$ an s und Ankunftszeit $j_{\text{arr_time}}$ an t . Dabei muss $l^i_{\text{arr_stop}} = l^{i+1}_{\text{dep_stop}}$ und $l^i_{\text{arr_time}} \leq l^{i+1}_{\text{dep_time}}$ für alle $i = 0, \dots, k - 1$ gelten, um einen Umstieg zwischen den Trips zweier aufeinanderfolgender Streckenabschnitte zu ermöglichen. Die Reise j besteht dabei aus $k + 1$ Trips, zwischen denen der Passagier k -Mal umsteigen muss. Für Reisen können Mengen an Optimierungskriterien definiert werden. Dabei dominiert eine Reise j_1 die Reise j_2 , falls j_1 in keinem Kriterium schlechter als j_2 ist. Die Dominanz wird mit $j_1 \leq j_2$ angegeben. Die Pareto-Optimalität der Reise j in einer Menge J an Reisen lässt sich so definieren, dass in J keine andere Reise existiert, die j dominiert.

2.1.1 Verspätungen

Für die erwarteten Ankunftszeiten müssen zusätzlich Verspätungen definiert werden. Da wir für unsere Datensätze keine historischen Daten zur Verfügung haben, benötigen wir ein Modell, das uns die Wahrscheinlichkeiten von Verspätungen angibt. Dieses führen wir analog zu Dijkstra et al. in [Di18] ein. Dabei definieren wir für jeden Trip t' an jedem seiner Stopps p eine Zufallsvariable $D(t', p)$, die die Verspätung des Trips an diesem Stopp angibt. Für diese Zufallsvariable definiert die Verteilungsfunktion $f(x)$ die Wahrscheinlichkeit $P[D(t', p) \leq x]$ mit x als Verspätung in Minuten. Der zugehörige Erwartungswert, der die erwartete Verspätung des Trips t' an p darstellt, sei $E[D(t', p)]$. Damit wir erwartete Ankunftszeiten berechnen können, müssen wir zusätzlich zu dem Verspätungsmodell Annahmen treffen. Die erste besagt, dass jede Zufallsvariable $D(t', p)$ eine maximale Verspätung $\max D(t', p)$ besitzt. Zudem nehmen wir an, dass alle Zufallsvariablen unabhängig voneinander sind

und dass die Verkehrsmittel aller Trips pünktlich abfahren. Verspätungen werden somit nur bei ihren Ankünften betrachtet. Diese Annahmen werden analog zu [Di18] getroffen. Hier wurde gezeigt, dass das Modell realistisch ist, solange keine massiven Verspätungen auftreten.

Mit dem Verspätungsmodell können sichere Reisen definiert werden. Eine sichere Reise garantiert, dass der Passagier trotz möglicher Verspätungen der genutzten Trips keinen Trip der Reise verpasst. Somit muss die Umsteigezeit an jedem Umstiegspunkt der Reise mindestens so groß wie die maximale Verspätung des eingehenden Trips an diesem Stopp sein.

2.1.2 Entscheidungsgraphen

Erwartete Ankunftszeiten definieren wir analog zu [Di18] über (s, t, τ_s) -Entscheidungsgraphen. Diese umfassen Reisen zwischen dem Startstopp s und dem Zielstopp t mit einer frühesten Abfahrtszeit τ_s . Ein Entscheidungsgraph $G = (V, E)$ besteht aus einer Menge an Stopps als Knoten V und einer Menge an Streckenabschnitten als Kanten E . Die Kanten verlaufen dabei jeweils zwischen $l_{\text{dep_stop}}$ und $l_{\text{arr_stop}}$ der zugehörigen Streckenabschnitte l . Die erwartete Ankunftszeit des Graphen G kann über die erwarteten Ankunftszeiten seiner Streckenabschnitte bestimmt werden. Die erwartete Ankunftszeit eines Streckenabschnitts $l \in E$ wird rekursiv per Fallunterscheidung definiert. Falls l zum Zielstopp t führt, setzt sie sich aus der geplanten Ankunftszeit des zugehörigen Trips und seiner erwarteten Verspätung zusammen. Andernfalls wird die erwartete Ankunftszeit über die ausgehenden Streckenabschnitte des Knotens $l_{\text{arr_stop}}$ definiert. Sie stellt dabei die gewichtete Aufsummierung der erwarteten Ankunftszeiten der ausgehenden Trips dar, in die der Passagier umsteigen kann. Die Gewichtung der Streckenabschnitte erfolgt dabei jeweils über die Wahrscheinlichkeit, zu der der Passagier abhängig von der Verspätung seines eingehenden Verkehrsmittels in den zugehörigen Trip umsteigt. Die erwartete Ankunftszeit des Entscheidungsgraphen G entspricht der erwarteten Ankunftszeit des ersten Streckenabschnitts am Startstopp s . Die späteste Ankunftszeit des Graphen wird mit $G_{\text{max_arr}}$ notiert. Für den Nutzer ergibt sich die Reise, die er dem Graph zufolge nehmen sollte, jeweils über den ersten Streckenabschnitt, den er an einem Umstiegspunkt nach seiner Ankunft erreicht. Dabei bezeichnen wir als Alternativreisen alle Reisen, die er nur deswegen nimmt, da er durch eine Verspätung des eingehenden Trips die erste Möglichkeit des Graphen verpasst. In Abb. 1 ist das Beispiel eines Entscheidungsgraphen dargestellt.

2.2 Problembeschreibungen

Wir beschäftigen uns mit unterschiedlichen Optimalitätskriterien für Reisen zwischen zwei Stopps und definieren hierfür die Probleme analog zu [Di18] und [DPW15]. Das Problem der frühesten Ankunftszeit (Earliest Arrival Time (EAT)) gibt auf der Eingabe

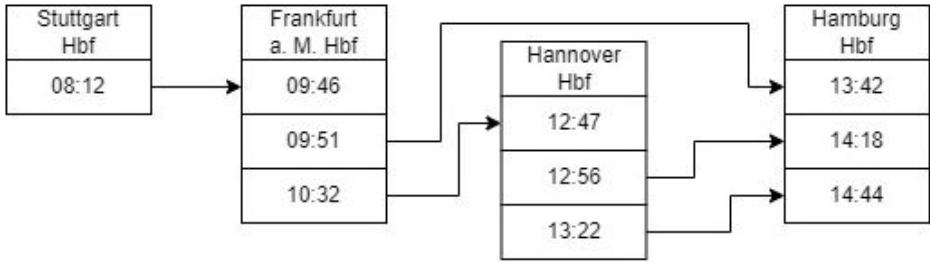


Abb. 1: Ein Entscheidungsgraph zwischen den Stopps Stuttgart Hbf und Hamburg Hbf.

eines Startstopps s , Zielstopps t und einer Startzeit τ_s die minimale Ankunftszeit aller Reisen an, die an Stopp s nach τ_s abfahren und an Stopp t ankommen. Eine Abwandlung des EAT-Problems ist das Problem der frühesten sicheren Ankunftszeit (Earliest Safe Arrival Time (ESAT)), das lediglich sichere Reisen berücksichtigt. Das EAT-Problem kann auf das Problem der frühesten Ankunftszeit im Intervall (Earliest Arrival Time Profile (EATP)) erweitert werden. Dabei ist als Eingabe ein Startstopp s , Zielstopp t und ein Intervall an Abfahrtszeiten Δ gegeben. Gesucht sind alle Pareto-optimalen Reisen zwischen s und t , die innerhalb des Intervalls an s abfahren. Die Pareto-Optimalität wird bezüglich der Abfahrts- und Ankunftszeit der Reisen definiert. Das Problem der minimalen erwarteten Ankunftszeit (Minimum Expected Arrival Time (MEAT)) besitzt als Eingabe einen Startstopp s , Zielstopp t und eine minimale Abfahrtszeit τ_s . Wie in [Di18] verwenden wir die α -beschränkte Variante des Problems, wobei die maximale Ankunftszeit auf $G_{\max_arr} \leq \tau_s + \alpha \cdot (\text{ESAT}(s, t, \tau_s) - \tau_s)$ mit $\alpha \geq 1$ gesetzt und daraufhin der Entscheidungsgraph G mit der minimalen erwarteten Ankunftszeit gesucht wird.

2.3 RAPTOR-Algorithmen

Die nachfolgenden RAPTOR-Algorithmen wurden von Delling et al. (2015) in [DPW15] vorgestellt. RAPTOR-Algorithmen berechnen die frühesten Ankunftszeiten jeweils rundenbasiert. So erhält man nach k Runden die frühesten Ankunftszeiten der Reisen, die maximal k Streckenabschnitte und somit $k - 1$ Umstiege umfassen.

In der Standardvariante des RAPTOR-Algorithmus zur Lösung des EAT-Problems wird für jeden Stopp p in $\tau^*(p)$ die bisherige frühestmögliche Ankunftszeit und in $\tau_k(p)$ die minimale Ankunftszeit nach k Runden gespeichert. In einer Runde werden jeweils alle Stopps p markiert, für die ein neuer Wert in $\tau^*(p)$ gesetzt wurde. In jeder Runde des Algorithmus betrachten wir alle Routen, die einen markierten Stopp der letzten Runde enthalten und ermitteln ihren frühesten Trip, den der Passagier erreichen kann. Dessen Ankunftszeiten können nun verwendet werden, um die frühesten Ankunftszeiten der Stopps der Route in ihrer definierten Reihenfolge zu aktualisieren. An jedem Stopp wird zusätzlich überprüft, ob ausgehend von der minimalen Ankunftszeit der letzten Runde ein früherer

Trip erreicht werden kann. Der Algorithmus terminiert, sobald keine Stopps in einer Runde markiert werden. Die Reise der frühesten Ankunftszeit am Ziel t , die in $\tau^*(t)$ gespeichert ist, können wir über zusätzliche Zeiger ermitteln.

Der davon abgeleitete McRAPTOR-Algorithmus kann dazu genutzt werden, das EATP-Problem zu lösen. Dazu speichern wir Labels einer Runde k für Stopps p in Beuteln $B_k(p)$. Jedes Label ist einem Trip zugeordnet und enthält die Abfahrtszeit am Startstopp s und Ankunftszeit am Stopp p . Beutel sind jeweils hinsichtlich dieser Parameter Pareto-optimal. Der Routenbeutel B_r enthält die Labels der aktuellen Route r . Während des Algorithmus betrachten wir in jeder Runde k jede Route r . Dabei führen wir für jeden ihrer Stopps p in ihrer definierten Reihenfolge drei Schritte durch. Zunächst aktualisieren wir die Ankunftszeiten aller Labels des B_r mit den Ankunftszeiten der zugehörigen Trips an p . Dann fügen wir unter Berücksichtigung der Dominanzregeln die Labels aus B_r in $B_k(p)$ ein und wiederholen dies für $B_{k-1}(p)$ und B_r . Bei dem letzten Schritt weisen wir den neuen Labels in B_r jeweils den ersten erreichbaren Trip von r zu. Zusätzliche Beutel $B^*(p)$ für Stopps p können verwendet werden, um alle nicht dominierten Labels zu speichern und somit den Algorithmus zu optimieren.

3 RAPTOR-MEAT

Nachdem das MEAT-Problem bereits mit dem Ansatz der CSAs gelöst wurde, entwickeln wir nun einen Algorithmus, bei dem dies auch über die rundenbasierte Herangehensweise des McRAPTOR-Algorithmus möglich ist. Der RAPTOR-MEAT-Algorithmus kann so erweitert werden, dass zusätzliche Optimalitätskriterien berücksichtigt werden.

Um minimale erwartete Ankunftszeiten berechnen zu können, verändern wir das Konzept des McRAPTOR-Algorithmus in der Hinsicht, dass wir ausgehend vom Zielstopp t die erwarteten Ankunftszeiten bis hin zum Startstopp s berechnen. Dazu benötigen wir ein Intervall an Ankunftszeiten als Eingabe. Über einen modifizierten CSA erhalten wir die früheste sichere Ankunftszeit $\text{ESAT}(s, t, \tau_s)$ mit der Eingabe des Startstopps s , der minimalen Abfahrtszeit τ_s und dem Zielstopp t . Als obere Grenze des Intervalls ergibt sich daraus die maximale Ankunftszeit τ_t : $\tau_t = \tau_s + \alpha \cdot (\text{ESAT}(s, t, \tau_s) - \tau_s)$. Zudem benötigen wir für unseren Algorithmus die frühesten Ankunftszeiten an jedem Stopp ausgehend von dem Startstopp s und der minimalen Abfahrtszeit τ_s . Dazu führen wir einen One-To-All-CSA zur Lösung des $\text{EAT}(s, \cdot, \tau_s)$ -Problems durch. Aus dessen Ergebnis erhalten wir auch die früheste Ankunftszeit an dem Zielstopp t . Diese ist die untere Grenze des Intervalls für unseren Algorithmus.

Im Gegensatz zu der Standardvariante des McRAPTOR-Algorithmus optimiert unsere Variante die minimalen erwarteten Ankunftszeiten. Die Labels eines Stopps p speichern somit Tupel bestehend aus der Abfahrtszeit an p und der erwarteten Ankunftszeit am Zielstopp t . Labels desselben Stopps werden wie bei dem McRAPTOR-Algorithmus in Beuteln abgespeichert. Sie sind so aufgebaut, dass sie ausschließlich Pareto-optimale Labels

hinsichtlich der Abfahrts- und erwarteten Ankunftszeit enthalten und ihre Einträge zudem nach den Abfahrtszeiten sortiert sind. Der RAPTOR-MEAT-Algorithmus speichert für jeden Stopp p einen Beutel $B^*(p)$ mit allen nicht dominierten Labels der bisher ausgeführten Runden. Unser Algorithmus arbeitet wie der McRAPTOR-Algorithmus rundenbasiert. In jeder Runde benötigen wir drei Arrays, die danach wieder zurückgesetzt werden können. Im ersten Array LD speichern wir für jeden Stopp die späteste Abfahrtszeit der Labels, die in der vorherigen Runde für diesen Stopp hinzugefügt wurden. Vor dem Start der ersten Runde fügen wir nur für den Zielstopp t die maximale Ankunftszeit τ_t hinzu. In dem Array Q speichern wir alle Routen-Stopp-Paare, die wir in der aktuellen Runde betrachten. Im letzten Array sichern wir für jeden Stopp p einen Beutel $B_c(p)$ mit den Labels, die in der aktuellen Runde hinzugefügt werden. Initial wird der Stopp t markiert und der Rundenzähler k auf 0 gesetzt.

Nach der Initialisierung führen wir Folgendes in jeder Runde des Algorithmus durch:

Zu Beginn jeder Runde inkrementieren wir den Rundenzähler k und bestimmen alle Routen, die mindestens einen markierten Stopp der letzten Runde enthalten und somit in der aktuellen Runde betrachtet werden müssen. Die zugehörigen Routen-Stopp-Paare (r, p) werden in Q gespeichert. Dabei entspricht p dem letzten markierten Stopp der Route r .

Die darauffolgende Schleife führen wir für jeden Eintrag (r, p) von Q durch. Zunächst initialisieren wir den Routenbeutel B_r , der alle nicht dominierten Label von r speichert, mit einem leeren Array. Während der Behandlung einer Route r führen wir für jeden Stopp p' ab dem Stopp p drei Schritte analog zu dem McRAPTOR durch. Dabei werden die Stopps innerhalb der Route von hinten nach vorne abgearbeitet. Zunächst wird der Routenbeutel mit den Abfahrtszeiten am aktuellen Stopp aktualisiert, dann wird er mit dem Beutel der Labels von p' der aktuellen Runde zusammengefügt und abschließend werden neue Labels in den Routenbeutel hinzugefügt.

An Stopp p' innerhalb der Route r erfolgt das Anpassen der Labels im ersten Schritt über die Abfahrtszeiten an diesem Stopp. Jedem Label des Routenbeutels ist ein Trip zugeordnet. Von diesem können wir die Abfahrtszeit an Stopp p' ermitteln und die bisherige Abfahrtszeit des Labels durch sie ersetzen. Im nächsten Schritt werden die Labels des Routenbeutels dem Beutel $B_c(p')$ des Stopps p' hinzugefügt. Wir verwenden für den Vorgang des Zusammenfügens, dass sowohl der Beutel B_r als auch der Beutel $B_c(p')$ nach den Abfahrtszeiten ihrer Labels sortiert sind. Daher reicht es aus, in einer Iteration über beide Beutel von der spätesten zur frühesten Abfahrtszeit zu gehen und alle nicht dominierten Labels in einen neuen Beutel einzufügen. Dieser wird danach als Beutel $B_c(p')$ für den Stopp p' verwendet. Im dritten Schritt werden neue Labels in den Routenbeutel hinzugefügt, falls es möglich ist, an diesem Stopp umzusteigen. Da wir in diesem Schritt zusätzlich die erwarteten Ankunftszeiten der neuen Labels berechnen müssen, wird hier das Verfahren des McRAPTOR angepasst. Dabei wählen wir zunächst alle Trips der Route r aus, deren Ankunftszeit zwischen der frühestmöglichen Ankunftszeit und der spätesten Abfahrtszeit der letzten Runde an diesem Stopp liegt. Die frühestmögliche Ankunftszeit erhalten wir

über das Ergebnis des One-To-All-CSA vom Startstopp s ausgehend, den wir während der Initialisierung durchgeführt haben. Die späteste Abfahrtszeit befindet sich in dem Array LD , das uns diesen Wert für jeden Stopp speichert. Ist dieser Wert nicht definiert, wurde der Stopp in der letzten Runde nicht markiert. Dann kann der Schritt übersprungen werden, da die dadurch erzeugten Labels keine Veränderungen der letzten Runde beinhalten würden. Für alle Trips, die innerhalb des definierten Intervalls zwischen frühester Ankunftszeit und spätester Abfahrtszeit an p' ankommen, erzeugen wir ein neues Label. Bei der Berechnung der erwarteten Ankunftszeiten der neuen Labels führen wir analog zu ihrer Definition eine Fallunterscheidung durch. Falls p' der Zielstopp t ist, wird die erwartete Ankunftszeit über die Summe aus der Ankunftszeit des Trips an Stopp p' und seiner erwarteten Verspätung berechnet. Im anderen Fall ergibt sich die erwartete Ankunftszeit über die gewichtete Addition der erwarteten Ankunftszeiten der Streckenabschnitte, in die der Umstieg an dem Stopp p' erfolgen kann. Die Gewichtung entspricht dabei jeweils der Wahrscheinlichkeit, mit der der Trip, der dem Streckenabschnitt zugeordnet ist, von dem Nutzer genommen wird. Wir verwenden für diese Berechnung die Labels, die in dem Beutel $B^*(p')$ für p' gespeichert sind. Dieser umfasst alle nicht dominierten Labels der vorangegangenen Runden. Falls die erwartete Ankunftszeit berechnet werden kann, setzen wir die Abfahrtszeit des neuen Labels auf die Abfahrtszeit des Trips an dem Stopp p' . Die neuen Labels fügen wir in den Routenbeutel B_r ein.

Bei der klassischen Variante des McRAPTOR werden die Beutel B^* während der vorherigen Schleife um die neuen Labels ergänzt. Da diese Beutel bei uns zur Berechnung der erwarteten Ankunftszeiten verwendet werden, erfolgt dies beim RAPTOR-MEAT erst, nachdem alle Routen einer Runde behandelt wurden. Dabei wird zunächst das Array LD der spätesten Abfahrtszeiten geleert. Anschließend fügen wir für alle Stopps p die Beutel $B_c(p)$ und $B^*(p)$ zusammen und verwenden das Ergebnis als neuen Beutel $B^*(p)$. Dazu werden nur nicht dominierte Labels berücksichtigt, und falls mindestens eines der Labels aus der aktuellen Runde darunter ist, markieren wir den Stopp p . Zusätzlich wird die späteste Abfahrtszeit der neuen Labels, die in den Beutel $B^*(p)$ eingefügt wurden, am Stopp p in dem Array der spätesten Abfahrtszeiten gespeichert.

Bevor wir die nächste Runde des Algorithmus durchführen, überprüfen wir, ob in der aktuellen Runde Stopps markiert wurden. Ist dies nicht der Fall, terminiert der Algorithmus. Sobald der Algorithmus terminiert ist, können die Labels analog zu dem Vorgehen beim CSA-MEAT in [Di18] verwendet werden, um den Entscheidungsgraphen der minimalen erwarteten Ankunftszeit zu erstellen.

Die Behandlung der einzelnen Routen und das abschließende Aktualisieren der Beutel B^* kann innerhalb einer Runde jeweils parallelisiert werden, da es unabhängig für die Routen beziehungsweise Stopps erfolgt. Analog zu [DPW15] können dabei Konfliktgraphen verwendet werden, um Speicherungs-Konflikte zu vermeiden. Die Korrektheit des Algorithmus basiert auf der Invariante, dass zu Beginn der Runde k die minimalen erwarteten Ankunftszeiten der Reisen, die aus maximal $k - 1$ Streckenabschnitten bestehen, in B^* gespeichert sind. Diese können in der Runde k dazu verwendet werden, gemäß der Definition

der erwarteten Ankunftszeiten diese für die Reisen bestehend aus k Streckenabschnitten zu berechnen. Somit werden die bisher betrachteten Reisen rundenbasiert jeweils um einen Umstieg bzw. Streckenabschnitt verlängert. Die Invariante kann dazu verwendet werden, den RAPTOR-MEAT-Algorithmus in der Hinsicht zu erweitern, dass dem Nutzer alternative Ergebnisse vorgeschlagen werden, die Vorteile hinsichtlich der maximalen Anzahl an Umstiegen besitzen. Dabei sollen kleinere Verschlechterungen der erwarteten Ankunftszeit durch Verbesserungen der maximalen Anzahl an Umstiegen aufgewogen werden. Dies kann für Passagiere den Komfort einer Reise verbessern, da sie seltener zwischen Trips umsteigen müssen, aber eine annähernd gleich gute Ankunftszeit erwarten können. Die entstehende Variante des Algorithmus bezeichnen wir mit RAPTOR-MEAT-TO (Transfer Optimisation). Dazu müssen wir die Standardvariante des RAPTOR-MEAT-Algorithmus anpassen. Da wir nach der Ausführung des Algorithmus potenziell Entscheidungsgraphen basierend auf den Labels aus bestimmten vorherigen Runden erstellen müssen, sichern wir in jeder Runde den Zustand der B^* Beutel. Abgesehen von dieser Anpassung führen wir die Standardvariante des RAPTOR-MEAT-Algorithmus durch. Nach seiner Ausführung müssen wir die erste Runde bestimmen, in der das Verhältnis aus der maximalen Anzahl an Umstiegen zu der erwarteten Ankunftszeit passend ist. Als Parameter definieren wir hierfür die Zeit pro verringerter Anzahl an Umstiegen, um die sich die erwartete Ankunftszeit von der minimalen erwarteten Ankunftszeit unterscheiden darf. Über die Ergebnisse, die in den Beuteln B^* gespeichert wurden, können wir die erste Runde bestimmen, in der das Kriterium erfüllt ist. Der zugehörige Entscheidungsgraph stellt das Resultat des RAPTOR-MEAT-TO dar. Eine ähnlich einfache Umsetzung der Variante ist aus unserer Sicht mit dem bestehenden CSA-MEAT aus [Di18] nicht möglich.

4 Experimente

Wir führen nun Benchmark-Tests und weitere Experimente für den neuen RAPTOR-MEAT-Algorithmus und seine Variante aus dem Kapitel 3 durch. Als Grundlage für die Tests nutzen wir die Datensätze des Schienenregional- und Schienenfernverkehrs in Deutschland. Diese umfassen alle S-Bahnen, Regionalbahnen, ICs und ICEs. Wir führen alle Tests auf denselben 1000 zufällig ausgewählten Anfragen aus. Als Verspätungsmodell nutzen wir das Modell von Disser et al. (2008) in [DMS08]. Es basiert auf der Funktion $\text{rel} : \tau \mapsto s - e^{\ln(1-a) - \frac{\tau}{b}}$, die wir als Verteilungsfunktion f mit $s = 1$ verwenden. Für die Trips des Schienenfernverkehrs wählen wir dabei andere Parameter aus als bei den Trips des Schienenregionalverkehrs, da wir annehmen, dass es durch die deutlich längeren Fahrtzeiten des Fernverkehrs wahrscheinlicher zu längeren Verspätungen kommt. Wir wählen als maximale Verspätung des Fernverkehrs 30 Minuten und 15 Minuten für den Regionalverkehr. Als Parameter setzen wir dementsprechend $a = 0.5$, $b = 7$ im Fernverkehr und $a = 0.65$, $b = 3.5$ im Regionalverkehr. Die für die Tests verwendete Hardware besitzt einen Intel Core i5 mit 6 Kernen, eine 500 GB NVMe Festplatte und 64 GB RAM. Auf ihr läuft Ubuntu 20.04. Bei dem RAPTOR-MEAT-Algorithmus verwenden wir die Single-Core Variante ohne Parallelisierung.

4.1 Benchmark-Tests

Bei den Benchmark-Tests vergleichen wir die Ausführungszeiten des neuen RAPTOR-MEAT-Algorithmus mit denen des bestehenden CSA-MEAT. Dabei führen wir die Tests mit den Beschränkungsparametern von $\alpha = 1$, $\alpha = 2$ und $\alpha = 3$ durch, um mögliche Einflüsse der Größe des initialen Intervalls an Verbindungen und Trips, die genutzt werden können, zu erkennen. Wie man in Abb. 2 sehen kann, ist der RAPTOR-MEAT-Algorithmus für $\alpha = 1$ mit einer Zeit von 182 ms schneller als der CSA-MEAT, der 214 ms benötigt. Dies ändert sich für größere Werte von α , da hier die Ausführung des RAPTOR-MEAT-Algorithmus länger dauert. Das lässt sich damit erklären, dass einerseits dann insgesamt mehr Trips berücksichtigt werden, die wir durch das rundenbasierte Vorgehen potenziell mehrfach betrachten und zudem längere Reisen möglich sind. Daraus resultieren mehr Runden, die während des Algorithmus ausgeführt werden müssen. Im Gegensatz dazu steigt bei den CSAs lediglich die Anzahl der betrachteten Verbindungen linear, da bei ihnen gesichert ist, dass jede Verbindung maximal einmal betrachtet wird. Dennoch lohnt es sich den RAPTOR-MEAT-Algorithmus auszuführen, da nur mit ihm eine einfache Optimierung der maximalen Anzahl an Umstiegen und eine Parallelisierung möglich ist. Außerdem verbessern sich in unseren Experimenten die erwarteten Ankunftszeiten durchschnittlich nur um sechs Minuten, wenn man $\alpha = 2$ statt $\alpha = 1$ wählt. Die Differenz von $\alpha = 2$ zu $\alpha = 3$ liegt im Durchschnitt sogar nur bei 0,03 s.

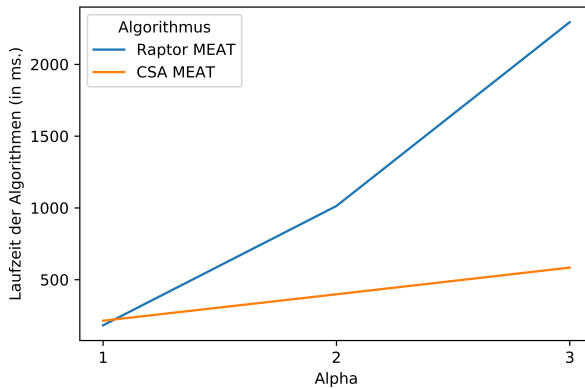


Abb. 2: Die Gesamtlaufrzeiten des CSA-MEAT und RAPTOR-MEAT-Algorithmus für unterschiedliche Werte von α .

4.2 Relevanztests

Um bestimmen zu können, welche Relevanz die Berechnung der minimalen erwarteten Ankunftszeiten hat, vergleichen wir sie mit den erwarteten Ankunftszeiten der Reisen, deren Optimierung ausschließlich hinsichtlich der frühesten geplanten Ankunftszeit erfolgt. Um diese Werte zu bestimmen, haben wir eine Variante der CSAs entwickelt, die auch

als Alternativreisen jeweils die Reisen mit den minimalen Ankunftszeiten auswählt und daraus die erwartete Ankunftszeit für den entstehenden Entscheidungsgraphen berechnet. Dieser Wert stellt die Ankunftszeit dar, die ein Passagier unter Berücksichtigung unseres Verspätungsmodells erwarten muss, wenn er sich bei jeder Entscheidung lediglich nach der minimalen Ankunftszeit richtet. Im Gegensatz dazu liefert der RAPTOR-MEAT-Algorithmus eine mindestens genauso gute erwartete Ankunftszeit, da er sie minimiert. Somit sind zwar hier potenziell die schnellsten Reisen in dem Entscheidungsgraphen nicht enthalten, aber der Passagier kann unter Annahme unseres Verspätungsmodells erwarten, dass er mindestens genauso früh ankommt wie ein Passagier, der sich nach den frühesten Ankunftszeiten der klassischen Routenberechnung in öffentlichen Verkehrsnetzen richtet. Im Durchschnitt beträgt die Differenz der Ergebnisse der beiden Algorithmen für $\alpha = 2$ etwa 1200 s bzw. 20 Minuten. Ein Passagier muss somit annehmen, um diese Zeitspanne später am Ziel anzukommen, wenn er sich nicht nach den minimalen erwarteten, sondern frühesten Ankunftszeiten richtet. Im Extremfall beträgt die Differenz absolut sogar über 8 Stunden und relativ gesehen 62% zur minimalen erwarteten Fahrtzeit.

Um herauszufinden, welche Vorteile die RAPTOR-MEAT-TO-Variante liefern kann, vergleichen wir ihre erwarteten Ankunftszeiten und maximale Anzahl an Umstiegen mit den zugehörigen Werten des RAPTOR-MEAT-Algorithmus. Dabei ist die erwartete Ankunftszeit des RAPTOR-MEAT-TO-Algorithmus mindestens so groß wie die der klassischen Variante. Im Gegensatz dazu ist die Anzahl an Umstiegen, die der Nutzer auf seiner Reise zwischen Start- und Zielstopp maximal vornehmen muss, kleiner oder gleich groß wie die des RAPTOR-MEAT-Algorithmus. Für $\alpha = 2$ und fünf Minuten als Parameter der Variante ist die erwartete Ankunftszeit der Transferoptimierungsvariante im Durchschnitt ca. 100 s später als die minimale erwartete Ankunftszeit. Dafür ist die maximale Anzahl an Umstiegen des Passagiers durchschnittlich um einen Umstieg geringer. Dies kann für Passagiere ein Vorteil sein, wenn sie auf Kosten einer etwas späteren erwarteten Ankunftszeit die Variante mit weniger Umstiegen wählen, da diese für sie eine einfachere Reise darstellt. Bei den Maximalwerten lohnt es sich besonders das Ergebnis der RAPTOR-MEAT-TO-Variante zu verwenden, da hier die Anzahl an Umstiegen um 10 reduziert werden konnte. Jedoch muss man beachten, dass die erwartete Ankunftszeit um bis zu 30 Minuten erhöht wurde.

5 Fazit

Wir haben basierend auf den bestehenden RAPTOR-Algorithmen einen neuen Algorithmus zur Lösung des MEAT-Problems entwickelt. Mit dem RAPTOR-MEAT-Algorithmus haben wir eine parallelisierbare Variante zur Berechnung der minimalen erwarteten Ankunftszeiten geschaffen. Für kleine Werte von α resultiert er dabei in besseren Ausführungszeiten als der CSA-MEAT. Zudem liefert er den Vorteil, dass wir über das rundenbasierte Vorgehen die maximale Anzahl an Umstiegen innerhalb der Reisen einfach optimieren können. Dies ist mit dem CSA-MEAT nur schwer möglich, kann aber einem Passagier Vorteile bieten. Aus unseren Relevanztests ergibt sich, dass das Konzept der minimalen erwarteten

Ankunftszeiten im Vergleich zu der Auswahl der Reisen ausschließlich über die frühesten Ankunftszeiten in den meisten Fällen hinsichtlich der erwarteten Ankunftszeit deutliche Vorteile liefert und somit für mehr Planungssicherheit sorgt. Ein weiteres Ergebnis der Tests ist, dass über die Transferoptimierungsvariante des RAPTOR-MEAT-Algorithmus die maximale Anzahl der Umstiege verringert werden kann, ohne die minimale erwartete Ankunftszeit erheblich zu verschlechtern. Dies resultiert in alternativen Ergebnissen, die für Passagiere potenziell einfachere Reisen ermöglichen und trotzdem mögliche Verspätungen berücksichtigen.

Literatur

- [BGM10] Berger, A.; Grimmer, M.; Müller-Hannemann, M.: Fully Dynamic Speed-Up Techniques for Multi-criteria Shortest Path Searches in Time-Dependent Networks. In: SEA. 2010.
- [BJ04] Brodal, G.; Jakob, R.: Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries. Proceedings of the 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'03), Electronic Notes in Theoretical Computer Science/, 2004.
- [Di18] Dibbelt, J.; Pajor, T.; Strasser, B.; Wagner, D.: Connection Scan Algorithm. ACM J. Exp. Algorithmics 23/, Okt. 2018, ISSN: 1084-6654, URL: <https://doi.org/10.1145/3274661>.
- [Di59] Dijkstra, E. W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1/, S. 269–271, 1959.
- [DKP12] Delling, D.; Katz, B.; Pajor, T.: Parallel Computation of Best Connections in Public Transportation Networks. ACM J. Exp. Algorithmics 17/, Okt. 2012, ISSN: 1084-6654, URL: <https://doi.org/10.1145/2133803.2345678>.
- [DMS08] Disser, Y.; Müller-Hannemann, M.; Schnee, M.: Multi-Criteria Shortest Path in Time Dependent Train Networks. Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08), Lecture Notes in Computer Science/, S. 347–361, 2008.
- [DPW15] Delling, D.; Pajor, T.; Werneck, R. F.: Round-based public transit routing. Transportation Science/, S. 591–604, 2015.
- [DW09] Delling, D.; Wagner, D.: Time-Dependent Route Planning. In (Ahuja, R. K.; Möhring, R. H.; Zaroliagis, C. D., Hrsg.): Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 207–230, 2009, URL: https://doi.org/10.1007/978-3-642-05465-5_8.
- [Py08] Pyrga, E.; Schulz, F.; Wagner, D.; Zaroliagis, C.: Efficient models for timetable information in public transportation systems. ACM Journal of Experimental Algorithmics/, 2008.

MICROLATION - EDSL zum Simulieren von komplexen Microservice-Anwendungen zur Bewertung ihrer Resilienz

Bjarne Valentin Rentz¹

Abstract: Microservice-Anwendungen müssen aufgrund der Kommunikation über das Netzwerk mit neuen Fehlerquellen im Vergleich zu klassischen Anwendungen umgehen können. Dafür gibt es verschiedene Entwurfsmuster (Retry, Timeout etc.), die unterschiedliche Einsatz- und Konfigurationsmöglichkeiten bieten. Es ist jedoch schwer für konkrete Anwendungen die optimale Verwendung und Konfiguration dieser herauszufinden. Deshalb stellt diese Arbeit eine Embedded Domain Specific Language (EDSL) vor, durch welche Microservice-Anwendungen definiert und simuliert werden können, um das eben genannte Problem zu lösen. MICROLATION ermöglicht damit die Erforschung der Muster und ihrer Konfigurationen im Rahmen von Microservices.

Keywords: Microservices; Simulation; Resilienz; EDSL

1 Einleitung

Für Microservice-Architekturen besteht ein aktuelles Interesse in der Industrie [Zi17] und Forschung [Ja18]. Als verteilte Systeme bringen Microservices neue Herausforderungen mit sich. Eine davon ist die Kommunikation über das Netzwerk und die damit verbundene Fehlerbehandlung. Einzelne Microservices können ausfallen oder nicht erreichbar sein. Auch können Aufrufe verloren gehen [Ja18; Ne15]. Im Sinne der *Antifragilen Organisation* [Ts13], sollten Organisationen daher von Anfang an mit Fehlern rechnen und ihr System dementsprechend planen und realisieren. Eine Antwort auf die Fehlerquellen sind Entwurfsmuster zur Erhöhung der Resilienz im Netzwerkverkehr.

Bei Betrachtung einer Microservice-Anwendung, die verschiedene Muster zur Erhöhung der Resilienz in der Kommunikation verwendet, ergibt sich folgende Fragestellung: Wie lässt sich die optimale Konfiguration für das System und insbesondere die der verwendeten Muster finden, um die Ausfallsicherheit respektive Resilienz zu maximieren? Ein Beispiel hierfür ist eine Anwendung mit zwei Microservices, wobei Microservice A Microservice B mit einem Retry-Muster aufruft, um eine Geschäftsfunktionalität zu implementieren. Hierbei stellt sich die Frage, wie die Anzahl an erneuten Versuchen und Wartezeit konfiguriert werden sollte, um eine hohe Resilienz zu erhalten.

Diese Arbeit zeigt einen simulationsbasierten Ansatz zum Bewerten verschiedener Entwurfsmuster und ihrer Konfiguration, welche die Resilienz im Netzwerkverkehr verbessern.

¹ NORDAKADEMIE, Angewandte Informatik, Kölner Chaussee, 25337 Elmshorn, Deutschland bjarnarentz@live.de

Die Simulation basiert auf einer EDSL in C#, durch welche die Systeme programmatisch definiert und simuliert werden können. In Listing 1 ist eine beispielhafte Definition zu sehen.

```
1 var ms1 = new Microservice();
2 var ms2 = new Microservice{
3     Routes = {
4         new Route{ Url = "Users", Faults = new TimeoutFault() }
5     }
6 };
7 var policy = new Retry();
8 var call = ms1.call(ms2,
9     new CallConfig{ Route = "Users", Policies = policy, Interval = TimeSpan.FromSeconds(1)});
```

Listing 1: Beispielhafte Definition von zwei Microservices durch *MICROLATION*

Hierbei werden folgende Themen behandelt:

- Welche Probleme bringen verbreitete Muster mit sich und welcher Lösungsansatz resultiert daraus (Abschnitt 2)?
- Wie können mithilfe von *MICROLATION* Microservices, ihre Aufrufe und Rückgabetypen und -werte definiert und simuliert werden können (Abschnitt 3)?
- Wie funktioniert *MICROLATION* und wie kann es erweitert und genutzt werden (Abschnitt 4)?
- Welche Ergebnisse können mit *MICROLATION* erzielt werden (Abschnitt 5) und wie steht dieser Ansatz im Vergleich zu anderen Arbeiten (Abschnitt 6)?

2 Problemstellung

Für die Erhöhung der Resilienz im Netzwerkverkehr existieren verschiedene Muster. In der Literatur wird dabei häufig auf die Muster aus *Release It* [Ny07] wie Retry, Timeout oder Circuit Breaker Bezug genommen. Wie in der Einleitung gezeigt wurde, bringen Muster für eine erhöhte Resilienz in der Netzwerkkommunikation mehrere Unbekannte mit sich. Jedes hat eigene Konfigurationsmöglichkeiten und unterschiedliche Aufgabenbereiche. Insbesondere komplexere Muster wie Circuit Breaker können oftmals in der Anzahl an Fehlern bis zum Auslösen, der Zeit, bis sie in den nächsten Zustand fallen, oder auch die Fehler, auf welche sie reagieren, konfiguriert werden. Zusätzlich können die Entwurfsmuster kombiniert werden, sodass beispielsweise Retry und Circuit Breaker verschachtelt werden. Dadurch ergibt sich eine hohe Anzahl an möglichen Kombinationen zwischen den Mustern selbst und entsprechenden ihren Konfigurationen, aus denen die möglichst optimale gefunden werden soll. Fachliche Aspekte und Vorgaben wie maximales Alter der angezeigten Daten können die Komplexität zusätzlich erhöhen. Forschung und Industrie stehen beide vor dem Problem, wie effektiv eine optimale bzw. gut passende Kombination aus Mustern und

Konfigurationen für den konkreten Einsatz gefunden werden kann. Aufgrund der daraus resultierenden hohen Komplexität verfolgt diese Arbeit einen simulationsbasierten Ansatz.

3 MICROLATION

MICROLATION ermöglicht es, mithilfe einer EDSL zuerst Microservice-Anwendungen mit verschiedenen Entwurfsmustern sowie Fehlerquellen zu definieren und anschließend das Gesamtsystem zu simulieren. Der Fokus der Bewertung liegt auf der Resilienz des Systems. Als EDSL besteht MICROLATION aus Klassen, die bereitgestellt werden und durch Interaktionen miteinander die benötigten Funktionalitäten implementieren. Die Details der Implementierung und Erstellung der EDSL werden in Abschnitt 4 behandelt. Die Funktionsweise wird im Folgenden anhand eines Beispiels erläutert.

Eine Microservice-Anwendung bestehend aus zwei Microservices: Der *Aufrufer* Microservice ruft unter Verwendung des Retry-Musters den *Ziel* Microservice auf. Mithilfe von MICROLATION soll nun eine gut geeignete Anzahl an erneuten Versuchen herausgefunden werden. Um den Retry auszulösen, werfen 10 % der Aufrufe des Ziels einen Fehler. Die zu durchsuchende Lösungsmenge ist die Anzahl an erneuten Versuchen von 0 bis 10.

```

1  int[] retryCounts = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
2
3  var caller = new Microservice("Caller");
4  var target = new Microservice("Target")
5  {
6      Routes =
7      {
8          new Route<int> { Url = "Target", Value = () => 1,
9              Faults = MonkeyPolicy.InjectException(with => with.Fault(new Exception())
10                  .InjectionRate(0.1)
11                  .Enabled()
12                  .AsPolicy<int>() }
13      }
14  };
15  var call = caller.Call(target, new CallOptions<int>
16  {
17      Route = "Target",
18      Interval = _ => TimeSpan.FromMilliseconds(500)
19  });
20  var simulation = new Simulation()
21  {
22      Microservices = { caller, target }
23  };

```

Listing 2: Definition einer Microservice-Anwendung in *MICROLATION*

In Listing 2 ist die Definition besagter Microservice-Anwendung zu sehen. Zuerst wird die Lösungsmenge (Zeile 1) und der aufrufende Microservice (Zeile 3) definiert. In den

Zeilen 4–14 wird der *Ziel* Microservice definiert. Dieser hat eine Route unter der URL *Target*, welche immer den Wert 1 zurückgibt (Zeile 8). Mithilfe der Zeilen 9–12 wird dafür gesorgt, dass 10 % der Aufrufe fehlschlagen, wodurch das Retry-Muster ausgelöst wird. Anschließend wird der Aufruf zwischen den beiden Microservice erstellt (Zeilen 15–19). Dafür stellt die Microservice-Klasse die *Call*-Methode bereit, welche als Parameter den aufzurufenden Microservice und *CallOptions* entgegennimmt. Die *CallOptions* definieren die aufzurufende Route (*Target*) und das Intervall wie oft der Aufruf ausgeführt werden soll, in diesem Fall beträgt es 500 Millisekunden. Die Definition der Anwendung wird durch die Erstellung einer Simulation (Zeilen 20–23) abgeschlossen. Dieser werden die zu simulierenden Microservices übergeben.

```

1  var results = new Dictionary<int, List<CallResult>>();
2  foreach (var retryCount in retryCounts)
3  {
4      call.CallOptions.Policies = Policy<int>.Handle<Exception>().Retry(retryCount);
5      var result = await simulation.Run(TimeSpan.FromSeconds(60));
6      results.Add(retryCount, result.First().Value);
7  }

```

Listing 3: Sequenzielle Ausführung der Simulationen durch *MICROLATION*

Die anschließende Ausführung der Simulation ist in Listing 3 zu sehen. Zuerst wird in Zeile 1 ein Dictionary zum Speichern der Ergebnisse erstellt. Als Schlüssel wird die jeweils verwendete Anzahl an erneuten Versuchen verwendet. Anschließend wird für jede zuvor definierte Anzahl an erneuten Versuchen eine Richtlinie – in diesem Fall das Retry-Muster – erstellt und dem Aufruf zugewiesen (Zeile 4). Daraufhin wird die Simulation für 60 Sekunden ausgeführt (Zeile 5) und das Ergebnis dem Dictionary hinzugefügt (Zeile 6). *MICROLATION* ermöglicht auch eine parallele Ausführung verschiedener Simulationen. Aufgrund der Einfachheit ist dieses Beispiel sequenziell.

Eine exemplarische Auswertung der Simulation ist in Tabelle 1 zu sehen. Dafür wurde für jede Anzahl an erneuten Versuchen die durchschnittliche Aufrufzeit und die Anzahl an Fehlern, die der aufrufende Microservice erhalten hat, bestimmt. Dabei ist deutlich zu erkennen, wie die Anzahl an erhaltenen Fehlern stark abnimmt und ab zwei erneuten Versuchen keine Fehler mehr erhalten wurden. Die durchschnittliche Verbindungszeit nimmt ebenfalls zuerst stark ab und erreicht im Bereich von 0,04ms ein Plateau. Die Aussagekraft der Ergebnisse sind jedoch fraglich, da die durchgeführte Simulation und Auswertung nicht für eine Bewertung ausreicht.

# Erneute Versuche	Durchschnittliche Aufrufzeit	# Fehler
0	0,128ms	13
1	0,0533ms	1
2	0,0457ms	0
3	0,0451ms	0
4	0,0428ms	0
5	0,0398ms	0
6	0,0412ms	0
7	0,0471ms	0
8	0,0455ms	0
9	0,0443ms	0
10	0,0375ms	0

Tab. 1: Ergebnisse des Beispiels

Der primäre Verwendungszweck ist die Simulation der Netzwerkaufrufe von Microservice-Anwendungen. Diese können flexibel definiert werden, wobei Entwurfsmuster zur Erhöhung der Resilienz eingesetzt werden können. Muster können frei konfiguriert werden sodass einer der Zwecke die gezielte Bewertung von unterschiedlichen Konfigurationen ist, wie im Beispiel gezeigt wurde. Ebenso kann der Einsatz von verschiedenen Mustern bzw. die Kombination dieser bewertet werden. Da die Microservices flexibel definiert werden können, ist es ebenfalls möglich verschiedene Architekturen zu vergleichen. Zusammengefasst soll durch MICROLATION die Bewertung verschiedener Microservice bezogener Aspekte wie der Einsatz von Entwurfsmustern, Konfiguration und Architektur ermöglichen werden. Somit kann MICROLATION auch in der Forschung und Industrie für verschiedene Einsatzmöglichkeiten verwendet werden. MICROLATION ist keine eigenständige Sprache. Als eingebettet Sprache bietet sie explizit die Möglichkeit weitere Auswertungen sowie Automatisierungen in C# zu implementieren.

4 Details

Dieser Abschnitt behandelt die technischen Details. Die Implementierung von MICROLATION orientiert sich an den Richtlinien für die Erstellung von DSLs von *Karsai et al.* [Ka14]. Nach der Implementierung wird zudem auf die Erweiterbarkeit eingegangen. Der Sourcecode ist auf Github² zu finden.

² <https://github.com/BjarneRantz/Microlation>

4.1 Implementierung

Zuerst wurden anhand des Verwendungszwecks die Schlüsselwörter der Sprache abgeleitet und als Klassen implementiert. Die Funktionalitäten von MICROLATION sind folglich durch die Interaktionen der Klassen untereinander realisiert. Bei den Schlüsselwörtern handelt es sich um `Simulation`, `Microservice`, `Route`, `CallConfig`, `Call` und `CallResult`. Ihre genaue Bedeutung sowie die Interaktionsmöglichkeiten werden im Folgenden behandelt.

Die `Simulation` dient als zentrale Klasse dem zentralen Ausführen einer Simulation über die `Run`-Methode. Die Methode gibt ein Dictionary zurück, mit dem jeweiligen `Call` als Schlüssel und einer Liste vom Typ `CallResult` als Wert. Somit kann jeder Aufruf eigenständig ausgewertet werden. Dafür werden der `Simulation` alle zu simulierenden `Microservices` übergeben.

Die `Microservice`-Klasse ist ein zentrales Element von MICROLATION. Fachlich repräsentiert sie einen zu simulierenden `Microservice`. Die Klasse kann zudem Routen und Calls definieren. Über die `Simulate`-Methode werden alle Calls des `Microservices` für die angegebene Dauer parallel ausgeführt, sodass Calls sich nicht untereinander beeinflussen. Aufrufe bzw. Calls zwischen zwei `Microservices` können über die `Call`-Methode der `Microservice`-Klasse erstellt werden. Diese nimmt als Parameter den aufzurufenden `Microservice`, sowie die `CallOptions` entgegen und erstellt anhand dieser einen Call welcher einer internen Liste hinzugefügt und ebenfalls als Rückgabewert zurückgegeben wird. Die Rückgabe ermöglicht es Calls manuell aufzurufen, sie als Schlüssel für die Ergebnisse der Simulation zu verwenden und das Erweitern des Calls um Validatoren. Wie in Listing 4 zu sehen können diese einem Call über Fluent-Ausdrücke hinzugefügt werden (Zeile 8). Dies ermöglicht die Verkettung mehrere `Validate`-Ausdrücke. In dem Beispiel ist der erhaltene Wert des Calls valide, wenn er größer als null ist.

```
1 var callWithoutPolicies = msl
2   .Call(ms2,
3     new CallOptions<int>
4     {
5       Route = "Id",
6       Interval = _ => TimeSpan.FromMilliseconds(Random.Shared.Next(100, 700))
7     })
8   .Validate(v => v > 0);
```

Listing 4: Beispielhafte Verwendung von Validatoren

Ein weiterer elementarer Bestandteil ist die Klasse `Route`. Wie bereits erwähnt repräsentiert sie einen Endpunkt bzw. eine Route eines `Microservices`, die aufgerufen werden kann. Routen haben ein Rückgabewert und sind typisiert, damit jede Route einen unterschiedlichen Rückgabebetypen haben kann. Damit sie trotzdem innerhalb einer Liste verwaltet werden können, implementieren sie das Interface `IRoute`. Routen verfügen zudem über die ebenfalls typisierten Attribute `Value` und `Faults`. Ersteres ist eine Funktion, welche den Rückgabewert

der Route zurückgibt. Damit ist es beispielsweise möglich zufällige Werte für jeden Aufruf der Route zu definieren. *Faults* sind optional und ermöglichen das injizieren von Fehlern bei Aufrufen der Route. Auf die genaue Funktionsweise wird im Rahmen von *Calls* detaillierter eingegangen.

CallOptions dienen lediglich der Sammlung von Parametern für den *Call*. Sie beinhalten die URL der aufzurufenden Route, eine Funktion *Interval*, welche abhängig von der Iteration die Zeit zwischen zwei Aufrufen zurückgibt und die *Policies* (Richtlinien), welche der Aufruf nutzt. Wie die *Faults* werden sie im Folgendem detaillierter erläutert.

CallResults repräsentieren das Ergebnis eines Aufrufs. Sie beinhalten die Dauer des Aufrufs (*ConnectionTime*), ob der Aufruf ein valides Ergebnis geliefert hat (*Valid*) und – falls aufgetreten – eine Ausnahme (*Exception*). Die Verbindungszeit ist eine gängige Metrik in Verschiedenen Arbeiten [Me20; ZGD19] und ermöglicht zudem eine effektive Bewertung des Fail-Fast-Ansatzes. Durch die Ausnahmen und Validierung wird zudem eine inhaltliche Bewertung ermöglicht.

Calls sind das Gegenstück von Routen und repräsentieren einen Aufruf dieser. Zudem beinhalten sie den simulativen Part von *MICROLATION* und führen die Aufrufe aus, die sie repräsentieren. Deshalb sind sie wie die Routen ebenfalls typisiert. Zudem existiert analog zu den Routen das Interface *ICall*, um sie trotz Typisierung als Schlüssel für das Dictionary zu verwenden, welches von der *Simulate*-Methode der *Microservice*-Klasse als Ergebnis zurückgeben wird. Über die *Execute*-Methode lässt sich der *Call* ausführen. Diese Methode baut zunächst die Aufrufkette über die *Policies* und *Faults* auf. *Policies* stehen für die eingesetzten Entwurfsmuster wie beispielsweise *Retry* oder *Timeout*. *Faults* hingegen verfolgen den diametralen Zweck und erzeugen künstliche Fehler wie Latenz, Ausnahmen oder anderer Rückgabewerte. *MICROLATION* nutzt dafür die Bibliothek *Polly*³, welche verschiedene der Muster bereits implementiert sowie über Interfaces verfügt, um neue Muster zu erstellen. Für die Fehler existiert die auf *Polly* basierende Bibliothek *Simmy*⁴, welche ebenfalls verschiedene Muster bereitstellt. Dadurch sind zum einen bereits gängige Muster wie *Retry*, *Timeout* oder *Circuit Breaker* implementiert. Zum anderen kann *MICROLATION* durch die Interfaces mit eigenen Mustern erweitert werden.

In Listing 5 ist die erste Hälfte der *Execute*-Methode zu sehen. Zuerst wird in Zeile 1 der bereits erwähnte Intervall abgewartet, bevor mit der Ausführung fortgeschritten wird. In den Zeilen 5 bis 9 wird der Aufruf aufgebaut. Da sowohl *Policies* als auch *Faults* optional sind, ergeben sich verschiedene Konstellationen für den Aufbau der Aufrufkette. Die in Zeile 9 verwendete Methode *Policy.Wrap* ermöglicht das verschachteln mehrerer Instanzen von *ISyncPolicy*. Dadurch ist dafür gesorgt, dass die Fehler innerhalb der Richtlinien ausgeführt werden und somit entsprechend abgefangen werden können.

³ <https://github.com/App-vNext/Polly>

⁴ <https://github.com/Polly-Contrib/Simmy>

```
1  await Task.Delay(CallOptions.Interval(iteration), token);
2  watch.Reset();
3
4  // Build the callChain based on the provided policies and faults.
5  ISyncPolicy<T>? callChain = null;
6  if (CallOptions.Policies != null)
7      callChain = CallOptions.Policies;
8  if (TypedRoute.Faults != null)
9      callChain = callChain == null ? TypedRoute.Faults : Policy.Wrap(callChain,
    ↪ TypedRoute.Faults);
```

Listing 5: Aufbau der Aufrufkette in der Execute-Methode

Die Durchführung des Aufrufs ist in Listing 6 abgebildet. Innerhalb eines Try-Catch-Blocks wird die Stoppuhr gestartet (Zeile 4) und anschließend der Aufruf ausgeführt. Dafür wird zunächst überprüft, ob die Aufrufkette existiert. Falls ja wird über diese die Value-Methode der Route aufgerufen. Andernfalls wird die Methode direkt aufgerufen, um das Ergebnis zu erhalten (Zeile 5). Daraufhin wird die Stoppuhr gestoppt und das Ergebnis validiert, falls Validatoren vorhanden sind. Sind keine vorhanden wird das Valid-Feld von CallResult auf false gesetzt. Tritt eine Ausnahme während des Aufrufs ein, wird diese gefangen (Zeilen 10–14). In diesem Fall wird ebenfalls die Stoppuhr gestoppt. Anstelle des Valid-Felds wird die Ausnahme dem Ergebnis hinzugefügt. Abschließend wird das Ergebnis zurückgegeben. Durch die Ausführung der Simulation auf der Aufruf-Ebene ist garantiert, dass Calls sich nicht untereinander beeinflussen. Zudem ist es dadurch auch möglich, auf jeder Ebene manuell Simulationen sowie Änderungen durchzuführen, wie bereits in dem Beispiel gezeigt werden konnte.

```
1  var result = new CallResult();
2  try
3  {
4      watch.Start();
5      var value = callChain != null ? callChain.Execute() => TypedRoute.Value() :
    ↪ TypedRoute.Value();
6      watch.Stop();
7      result.CallDuration = watch.Elapsed;
8      result.Valid = Validators.Any() && Validators.Aggregate(true, (curr, next) => curr &&
    ↪ next(value));
9  }
10 catch (Exception e)
11 {
12     watch.Stop();
13     result.Exception = e;
14     result.CallDuration = watch.Elapsed;
15 }
16 return result;
```

Listing 6: Durchführung des Aufrufs in der Execute-Methode

4.2 Erweiterbarkeit

Insbesondere im Bereich der Policies sowie Faults ist MICROLATION durch die Verwendung von Polly respektive Simmy und deren Interfaces erweiterbar. Dies wird dadurch bestärkt, dass bereits verschiedene Erweiterungen für Polly existieren⁵. Für die Auswertung lassen sich zudem verschiedene Bibliotheken wie LINQ von Microsoft verwenden, da es sich bei CallResult um ein Plain Old Class Object (POCO) handelt.

5 Evaluation

Für eine abschließende Evaluation soll mithilfe von MICROLATION folgende Problemstellung gelöst werden: Ein Aufruf eines anderen Microservices verwendet die Muster Retry und Timeout, um die Resilienz zu erhöhen. Welche Konfigurationen erreichen abhängig zur Erreichbarkeit des aufzurufenden Microservices die besten Ergebnisse? Ähnlich wie in der Arbeit [Me20] sollen die Muster bzw. ihre Konfiguration erforscht werden.

Für die Beantwortung werden zwei Microservices definiert, wobei einer eine entsprechende Route zur Verfügung stellt. Dieser wird von dem anderen Microservice unter Verwendung von Retry und Timeout aufgerufen. Dabei ist Timeout in Retry verschachtelt. Dadurch reagiert der Retry auf Fehler, die der Timeout wirft. Der Aufruf wird alle 500 Millisekunden ausgeführt. Für die Simulation der Erreichbarkeit, wird dem aufgerufenen Microservice eine Latenz von 4 Sekunden injiziert. Die Wahrscheinlichkeit, mit welcher die Latenz injiziert wird, ergibt sich durch folgende Berechnung: $1 - \text{Erreichbarkeit}$, wobei *Erreichbarkeit* ein Wert zwischen 0 und 1 ist und die Erreichbarkeit in Prozent angibt.

Dieses System wird mit allen Kombinationen aus Timeout in Millisekunden (1000, 1500, 2000, 2500, 3000), Anzahl an erneuten Versuchen (0–4) und Erreichbarkeiten (50 %, 60 %, 70 %, 80 %, 90 %, 100 %) für zehn Minuten simuliert. Die Erhebung der Metriken beschränkt sich aufgrund des begrenzten Umfangs auf die gesamte Dauer der Aufrufe und wie viele Fehler der Aufrufer erhalten hat. Der Sourcode für die Evaluation ist ebenfalls über Github bereitgestellt⁶.

In Abbildung 1 sind für fünf der 150 simulierten Konfigurationen die gesamte Aufrufdauer (Abbildung 1a) und die Anzahl an erhaltenen Fehlern (Abbildung 1b) zu sehen. In der Legende gibt der erste Wert den konfigurierten Timeout und der zweite Wert die Anzahl an erneuten Versuchen an. Abgebildet sind fünf Konfigurationen mit gleichem Timeout (1000 ms) aber unterschiedlicher Anzahl an erneuten Versuchen (0–4). In Abbildung 1a ist zu sehen, wie sich eine erhöhte Anzahl an erneuten Versuchen negativ auf die Gesamtdauer aller Aufrufe auswirkt. Ebenfalls ist zu erkennen, dass bei sinkender Erreichbarkeit die Differenz steigt. Bei der Durchführung mit zwei erneuten Versuchen ist auffällig, dass bei

⁵ <https://github.com/Polly-Contrib>

⁶ <https://github.com/BjarneRentz/Microlation/blob/main/Examples/SampleEvaluation.cs>

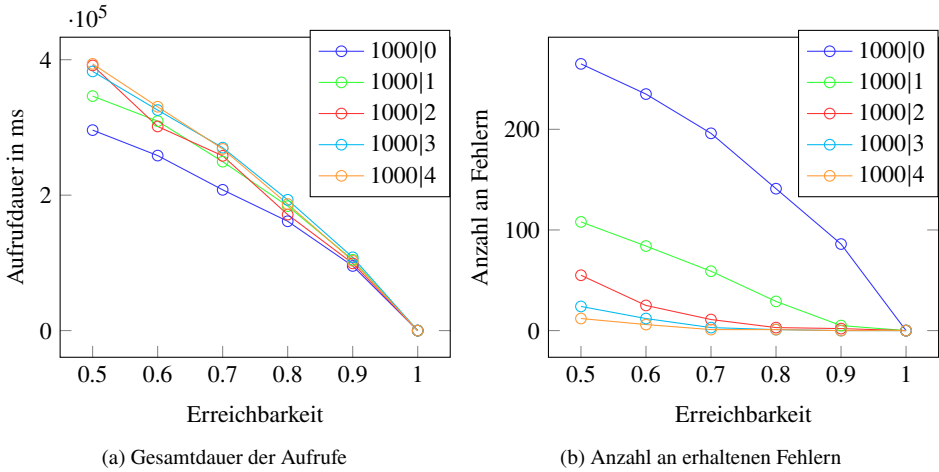


Abb. 1: Ausschnitt der Ergebnisse von fünf Konfigurationen

einer Erreichbarkeit von 60 % die Aufrufdauer geringer ist als bei einem erneuten Versuch. Im Bereich der erhaltenen Fehler (Abbildung 1b) fällt auf, dass die Graphen – im Gegensatz zur Aufrufdauer – unterschiedliche Formen je nach Konfiguration annehmen. Während der Graph bei der Durchführung ohne erneuten Versuch bei steigender Erreichbarkeit zunehmend stärker fällt, ist bei einem erneuten Versuch eine lineare Entwicklung zu sehen. Bei zunehmenden erneuten Versuchen wird der Effekt bei besserer Erreichbarkeit stetig weniger. Ebenfalls sind die Differenzen zwischen den Konfigurationen stärker ausgeprägt, nimmt jedoch bei zunehmender Anzahl an erneuten Versuchen ab.

Ein Vergleich der Ergebnisse ist aufgrund mangelnder Arbeiten in diesem Bereich schwierig. Die Arbeit von [Me20] bestätigt trotz unterschiedlicher Herangehensweise und Aufbau der Microservice-Anwendung die Tendenzen: Der Einsatz von Mustern erhöht die Dauer der Aufrufe und verringert die Anzahl an auftretenden Fehlern. Daher sollte prinzipiell eine Konfiguration verwendet werden, welche auch die fachlichen Anforderungen erfüllt. Beispielsweise hat ein erneuter Aufruf einen geringeren Einfluss auf die Zeit, aber einen vergleichsweise hohen Einfluss auf die erhaltenen Fehler.

Durch die Evaluation wird deutlich, dass sich durch MICROLATION auch Kombinationen von unterschiedlichen Mustern und Ausgangssituationen simulieren lassen. Durch den programmatischen Ansatz können die Ergebnisse zudem automatisiert gesammelt und verarbeitet werden. Dies ist insbesondere bei größeren Datenmengen hilfreich, die durch die Kombinatorik entstehen. Im Rahmen der Evaluation sind zudem Limitationen deutlich geworden. Derzeit existiert keine Möglichkeit, verkettete Aufrufe abzubilden. Auch die Möglichkeit, die Ergebnisse zu validieren, ist in ihrer Auswirkung begrenzt. Da die Validierung erst nach dem Ausführen des Aufrufs durchgeführt wird, können die eingesetzten Muster nicht darauf reagieren.

6 Verwandte Arbeiten

Die bereits erwähnte Arbeit von *Mendonça et al.* [Me20] behandelt die Bewertung verschiedener Konfiguration unterschiedlicher Entwurfsmuster anhand eines stochastischem Modells und unterscheidet sich damit in der Herangehensweise. Zudem verfolgt diese Arbeit das Ziel, ein Framework zur Untersuchung von Mustern zu erarbeiten und nicht die konkrete Bewertung dieser. Die Simulation von Microservice-Anwendungen wird ebenfalls in [GIM17] behandelt. Die Arbeit hat jedoch das Ziel, den Einfluss der Anwendung auf die Performance und daraus resultierenden Hardwareabhängigkeiten zu untersuchen. Einen ähnlichen Ansatz wie MICROLATION verfolgt *μqSim* [ZGD19]. Dies erlaubt es, komplexe Microservice-Anwendungen in ihrem kompletten Verhalten in Bezug auf Antwortzeiten zu simulieren. Jedoch werden die Microservices dafür über eine JSON-Datei definiert und anschließend durch *μqSim* simuliert. MICROLATION bietet im Vergleich einen programmiertechnischeren Ansatz mit Fokus auf die Bewertung der Resilienz und nicht dem Latenzverhalten der gesamten Anwendung.

7 Fazit

Die Simulation von Microservice-Anwendungen ist ein breites Gebiet mit verschiedenen Ansätzen und Zielen. In dieser Arbeit wurde als Ansatz eine EDSL vorgestellt, die eine programmatische Definition und Simulation der Microservice-Anwendung erlaubt. Der Fokus liegt dabei auf dem Netzwerkverkehr zwischen den Microservices, wobei verschiedene Entwurfsmuster mit unterschiedlichen Konfigurationen zur Erhöhung der Resilienz eingesetzt und simuliert werden können. MICROLATION ermöglicht somit eine Erforschung bestehender und neuer Entwurfsmuster, die die Resilienz des Netzwerkverkehrs von Microservice-Anwendungen erhöhen sollen. Dazu zählt insbesondere auch, welchen Einfluss verschiedene Konfigurationen für bestehende Muster haben und welchen Einfluss die Kombination von unterschiedlichen Mustern hat.

Die zukünftige Arbeit sollte sich darauf konzentrieren, die aktuellen Limitation aufzuheben. Die Validierung kann durch eine injizierte Funktion in den Call, welche Ausnahmen wirft und damit die Muster auslöst, erweitert werden. Eine Implementierung von verketteten Aufrufen, würde eine realitätsnahe Simulation von Microservice-Anwendungen ermöglichen, ist jedoch in der Umsetzung aufgrund der aktuellen Realisierung der Calls komplex. Der Bereich der Auswertung sollte durch eine standardmäßige Evaluation erweitert werden. Diese sollte die Ergebnisse der einzelnen Aufrufe während der Simulation bereits aggregieren, um die Anzahl der verwalteten Objekte zu reduzieren und verschiedene Möglichkeiten bieten, die Daten zu speichern (z. B. Datei oder Datenbank).

Danksagung

Ich möchte mich bei den Reviewern für das sehr ausführliche und hilfreiche Feedback bedanken. Die Vorschläge haben entscheidende Beiträge zur Qualität dieser Arbeit geleistet.

Literatur

- [GIM17] Gribaudo, M.; Iacono, M.; Manini, D.: Performance Evaluation of Massively Distributed Microservices Based Applications. In: 31st European Conference on Modelling and Simulation, ECMS 2017. European Council for Modelling und Simulation, S. 598–604, 2017.
- [Ja18] Jamshidi, P.; Pahl, C.; Mendonça, N. C.; Lewis, J.; Tilkov, S.: Microservices: The Journey so Far and Challenges Ahead. IEEE Software 35/3, Publisher: IEEE, S. 24–35, 2018.
- [Ka14] Karsai, G.; Krahn, H.; Pinkernell, C.; Rumpe, B.; Schindler, M.; Völkel, S.: Design Guidelines for Domain Specific Languages. arXiv:1409.2378 [cs]/, 8. Sep. 2014, arXiv: 1409.2378, URL: <http://arxiv.org/abs/1409.2378>, Stand: 08.05.2022.
- [Me20] Mendonça, N. C.; Aderaldo, C. M.; Cámara, J.; Garlan, D.: Model-Based Analysis of Microservice Resiliency Patterns. In: 2020 IEEE International Conference on Software Architecture (ICSA). IEEE, S. 114–124, 2020.
- [Ne15] Newman, S.: Building Microservices: Designing Fine-Grained Systems. O’Reilly Media, Beijing Sebastopol, CA, 2015, ISBN: 978-1-4919-5035-7.
- [Ny07] Nygard, M. T.: Release It! Design and Deploy Production-Ready Software. Pragmatic Bookshelf, Raleigh, N.C, 2007, ISBN: 978-0-9787392-1-8.
- [Ts13] Tseitlin, A.: The Antifragile Organization. Communications of the ACM 56/8, S. 40–44, Aug. 2013, ISSN: 0001-0782, 1557-7317, URL: <https://dl.acm.org/doi/10.1145/2492007.2492022>, Stand: 07.05.2022.
- [ZGD19] Zhang, Y.; Gan, Y.; Delimitrou, C.: μ qSim: Enabling Accurate and Scalable Simulation for Interactive Microservices. In: 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, S. 212–222, 2019.
- [Zi17] Zimmermann, O.: Microservices Tenets. Computer Science - Research and Development 32/3, S. 301–310, 1. Juli 2017, ISSN: 1865-2042, URL: <https://doi.org/10.1007/s00450-016-0337-0>.

Maschinelles Lernen und Informatik in der Anwendung

Implementierung und Analyse von Gradientenberechnung in Quantenalgorithmen

Moritz Schmidt¹

Abstract: Quantencomputer bieten die theoretische Möglichkeit, verschiedenste Probleme präziser und schneller zu lösen als klassische Computer. Auch im Gebiet des maschinellen Lernens, welches in den letzten Jahren in einem immer größer werdenden Spektrum an Disziplinen Anwendung findet, hofft man, das Potential des Quantencomputers zu entfalten. Viele Algorithmen des maschinellen Lernens sind im Kern Optimierungsprobleme. Um eine möglichst genaue Lösung für diese Probleme zu finden, werden oft gradientenbasierte Verfahren als Kompromiss zwischen Rechenaufwand und Qualität der Lösung verwendet. In dieser Arbeit werden verschiedene Methoden zur Bestimmung von Gradienten von Funktionen, die durch Quantenschaltkreise implementiert werden, analysiert und verglichen. Die Ergebnisse zeigen, wie die inhärente Varianz von Messungen auf Quantencomputern zu einem Dilemma bei der Wahl von Hyperparametern von numerischen Verfahren führt, warum das analytische Parameter-Shift Verfahren einzelne Gradienten nicht nur exakt, sondern auch effizient berechnet und warum das SPSA Verfahren vor allem zur Gradientenberechnung auf großen Schaltkreisen mit vielen Parametern eine gute numerische Alternative sein kann. Dies kann als Entscheidungsgrundlage zur Gradientenberechnung für zukünftige Implementierungen von Algorithmen des maschinellen Lernens auf Quantencomputern dienen.

Keywords: Quanteninformatik; Quantum Machine Learning; Variationelle Quantenalgorithmen; Gradientenverfahren

1 Einleitung

Variationelle Quantenalgorithmen (VQAs) sind eine Klasse von Quantenalgorithmen, die Schaltkreise aus parametrisierten Gattern verwenden, sogenannte *Ansätze* oder Variational Forms, deren Parameter mit klassischen Methoden optimiert werden. Als Verlustfunktion fungiert dabei die Messung in einer problemspezifischen Messmatrix, der sogenannten *Observable*, beziehungsweise eine Funktion über mehrere solcher Messungen.

Ein als gut angesehener Ansatz kann mit wenig Gattern eine große Menge unitärer Operationen generieren [Du20]. Man muss sich also zu jedem Problem nicht spezifisch einen konkreten Schaltkreis überlegen, sondern hofft, dass die gewünschte unitäre Operation durch den Ansatz erzeugt und die dazugehörige Parameterkonfiguration durch Optimierung

¹ Universität Stuttgart, Institut für Architektur von Anwendungssystemen, Universitätsstraße 38, 70569 Stuttgart, Deutschland st165607@stud.uni-stuttgart.de

gefunden werden kann. Auf das maschinelle Lernen bezogen lassen sich Parallelen zum klassischen Feature bzw. Representation Learning erkennen: Beispielsweise soll ein neuronales Netzwerk bei einem Klassifizierungsproblem eine Funktion darstellen, die Eingabedaten verschiedenen Klassen zuordnet, wobei man hofft, dass die gewünschte Funktion durch die Netzwerkarchitektur möglichst akkurat dargestellt werden kann.

Bekannte VQAs sind der Quantum Approximate Optimization Algorithmus (QAOA), der beispielsweise angewandt auf das Max-Cut Problem in Clusteringalgorithmen Verwendung findet [FGG14] oder der Variational Quantum Eigensolver (VQE) [Pe14], der in modifizierter Form [Ce20] für PCA verwendet werden kann.

VQAs sind in der heutigen Noisy Intermediate-Scale Quantum (NISQ) Ära [Pr18], in der Gatter von Quantencomputern fehlerbehaftet sind und ungenauer werden, je länger die Berechnung andauert, besonders prominent, da in einer Iteration ein einzelner Schaltkreis meist kurz ist. Außerdem bestehen die Ansätze meist aus Gattern, die von vielen Quantencomputern als Basisgatter verwendet werden.

Da VQAs iterative Optimierungsalgorithmen sind, können je nach Verlustfunktion, Ansatz, Observable und Optimierungsalgorithmus viele Ausführungen nötig sein, bis das Minimum gefunden wurde. Falls die Verlustfunktion mehrere Minima besitzt, kann der Algorithmus auch in einem lokalen Minimum stecken bleiben.

Auch wenn einige Optimierungsverfahren, wie der Nelder-Mead Algorithmus [NM65], ohne Gradienteninformationen optimieren können, liefern gradientenbasierte Verfahren oft effizientere und bessere Lösungen. Um nun auch mit gradientenbasierten Optimierungsverfahren Probleme des maschinellen Lernens auf Quantencomputern zu lösen, gilt es zu klären, ob sich möglichst genaue Quantengradienten mit akzeptablem Aufwand berechnen lassen.

2 Quantengradientenverfahren

2.1 Problemdefinition

Gegeben sei ein Schaltkreis $U(\theta)$ auf beliebig vielen Qubits, der auf den Startzustand $|0\rangle$ (alle Qubits sind im Zustand 0) angewandt wird. Die Gatter des Schaltkreises können dabei von den k Parametern $\theta \in \mathbb{R}^k$ abhängen. Der resultierende Zustand nach Anwendung von $U(\theta)$ ist der Zustand $|\psi(\theta)\rangle$. Danach misst man in Observable $O(\omega)$, die von m Parametern $\omega \in \mathbb{R}^m$ abhängt, mit Erwartungswert $\langle\psi(\theta)|O(\omega)|\psi(\theta)\rangle$.

Man kann den Gradienten des Erwartungswerts bezüglich Observablenparametern ω oder Schaltkreisparametern θ berechnen. Der Fokus der Arbeit liegt auf Gradientenberechnung bezüglich Schaltkreisparametern θ .

In der Arbeit wird davon ausgegangen, dass der Erwartungswert der Messung in $O(\omega)$ als Verlustfunktion verwendet wird. Die Verlustfunktion kann auch eine Funktion eines oder mehrerer Erwartungswerte sein. Ein Beispiel dafür ist eine übliche Verlustfunktion des maschinellen Lernens, bei der für jedes Eingabedatum der Erwartungswert des Schaltkreises ausgewertet wird und der mittlere quadratische Fehler zu gewünschten Zieldatenpunkten als Verlustfunktion $L(\theta)$ fungiert. Die konkrete Ableitung kann dann über die Kettenregel bestimmt werden, wobei zusätzliche klassische Berechnungen hinzukommen. Da unsere Messergebnisse nur Schätzer der Erwartungswerte sind, kann es hier zu Schwierigkeiten kommen, falls nicht-lineare Funktionen verwendet werden [Sw20].

Im Weiteren werden Approximationen der partiellen Ableitung des i -ten Parameters θ_i als g_i bezeichnet. Zur einfacheren Notation wird im Weiteren bei der Berechnung von g_i am Parameterpunkt θ die Messung der Verlustfunktion $L(\theta + e_i h)$, also an einer Stelle, an der nur der Parameter, bezüglich dessen gerade die partielle Ableitung berechnet wird, um einen Skalar h vom Ausgangspunkt verschoben wurde, mit $L(\theta + h)$ abgekürzt.

2.2 Finite Differenzen

Finite Differenzen Methoden erlauben das Approximieren von Ableitungen und finden vor allem Anwendung in numerischen Methoden zur Lösung von Differenzialgleichungen [Sm85].

Aus der Taylor-Reihenentwicklung ergeben sich die zentralen finiten Differenzen:

$$g_{iCFD} = \frac{L(\theta + h) - L(\theta - h)}{2h} + O(h^2) \quad (1)$$

Übertragen auf den Quantencomputer lassen sich so mit zwei Messungen pro Parameter die einzelnen partiellen Ableitungen $\frac{\partial L}{\partial \theta_i}$ des Gradienten approximieren.

2.3 Simultaneous Perturbation Stochastic Approximation

Ein weiteres numerisches Verfahren zur Gradientenapproximation ist Simultaneous Perturbation Stochastic Approximation (SPSA) [Sp87]. Im Gegensatz zu zentralen Differenzen benötigt SPSA nur 2 Funktionsaufrufe für beliebig viele Parameter. Dazu wählt man einen zufällig gewählten Vektor Δ , der, mit einem festen Abstand h skaliert, als Abstandsvektor fungiert. Die einzelnen Gradiententerme werden dann ähnlich zu zentralen Differenzen berechnet, wobei die zwei Funktionswerte jedoch für alle Ableitungen gleich bleiben und nur die Skalierung mithilfe der einzelnen Elemente $h\Delta_i$ des Abstandsvektors variiert wird:

$$g_i(\Delta)_{SPSA} = \frac{L(\theta + h\Delta) - L(\theta - h\Delta)}{2h\Delta_i} \quad (2)$$

Dies ist meist eine gröbere Abschätzung des Gradienten, aber bei passend gewählter Wahrscheinlichkeitsverteilung für Δ ist g_{SPSA} ein erwartungstreuer Schätzer des Gradienten. Hier wurden die Einträge Δ_i unabhängig voneinander zufällig aus $\{\pm 1\}$ anhand einer Bernoulliverteilung mit Wahrscheinlichkeit $\frac{1}{2}$ für beide Fälle gezogen.

Für verrauschte Funktionen, wie Messungen auf Quantencomputern, wird empfohlen, in jeder Iteration den Gradienten genauer zu approximieren, indem man den Durchschnitt mehrerer solcher Gradientenapproximationen berechnet.

2.4 Parameter-Shift

Das bekannteste Verfahren zur analytischen Gradientenberechnung auf Quantencomputern ist das Parameter-Shift Verfahren [Sc19].

Sei lediglich das Gatter $\mathbb{G}(\theta)$ vom Parameter θ_j abhängig. Jede unitäre Matrix lässt sich als Matrixexponential e^{iG} der hermiteschen Generatormatrix G schreiben. Wenn sich $\mathbb{G}(\theta_j)$ schreiben lässt als $e^{i\theta_j G}$ und für den Generator G gilt, dass $G^2 = r^2 I$, dann gilt Gleichung (3).

$$g_{iPS} = \frac{\partial L}{\partial \theta_j} = r[L(\theta_j + \frac{\pi}{4r}) - L(\theta_j - \frac{\pi}{4r})] \quad (3)$$

Der Name des Verfahrens ergibt sich daraus, dass nur die Parameterwerte verschoben werden, der sonstige Schaltkreis aber komplett identisch bleibt. Man kann in diesem Fall also den Gradienten eines Schaltkreises mit zwei Anwendungen desselben Schaltkreises berechnen.

Falls die Generatorbedingung $G^2 = r^2 I$ nicht erfüllt wird, aber G genau zwei Eigenwerte λ_1, λ_2 mit $2r' = \lambda_1 - \lambda_2$ besitzt, können mittels zusätzlicher Phasenverschiebung die Eigenwerte auf $\lambda_1 = -\lambda_2 = r'$ gesetzt werden, sodass das Verfahren angewandt werden kann. Alle einzelnen Paulistrings erfüllen diese Eigenschaft als Generator, somit lässt sich beispielsweise der Gradient von den 1-Qubit Rotationsmatrizen R_x, R_y, R_z mittels Parameter-Shift berechnen.

Interessant ist die Ähnlichkeit zur zentralen finiten Differenzen Gleichung (1). Die beiden Verfahren unterscheiden sich nur im Bezug auf den Skalierungsfaktor der Differenz der beiden Messwerte.

Wenn es Verfahren ähnlich zu zentralen Differenzen gäbe, wären Verfahren ähnlich zu vorwärts oder rückwärts Differenzen von großem Vorteil, da der Messwert $L(\theta)$ an der Samplestelle dann für alle partiellen Ableitungen verwendet werden kann, also man sich die Hälfte aller Messungen sparen könnte. In [Hu22] wurde jedoch gezeigt, dass ein solches Verfahren nicht existieren kann.

3 Ergebnisse

3.1 Experimente

Als Experimente wurden die Gradientenverfahren in Qiskit [AN21] implementiert und mittels QASM-Simulator getestet. Viele verschiedene Faktoren spielen bei der Gradientenbestimmung eine Rolle, wie die Observable und der Ansatz, die verwendet werden. Diese können auch wieder Hyperparameter haben. VQE-Ansätze zum Beispiel sind häufig als sich wiederholende Schichten aufgebaut, mit denen man die Tiefe des Schaltkreises skalieren kann. Bei der Observable ist die Dimensionalität selbst relevant, welche die Mindestanzahl an benötigten Qubits impliziert, aber auch die Anzahl und Orientierungen der Eigenvektoren, sowie die Größenordnungen der Eigenwerte.

In den Experimenten wurde der in Qiskit als EfficientSU2 bezeichnete Schaltkreis verwendet, der konzeptionell dem Hardware Efficient Ansatz aus [Ka17] ähnelt. Der Schaltkreis besteht aus alternierenden 1-Qubit Gatter- und Verschränkungsschichten. Als 1-Qubit Gatter wurden die Rotationsgatter R_x und R_z , sowie CNOT-Gatter als Verschränkungsgatter angewandt (siehe Abb. 1). Da als parametrisierte Gatter nur einfache Rotationsgatter verwendet wurden, lässt sich auf allen Gattern das Parameter-Shift Verfahren anwenden.

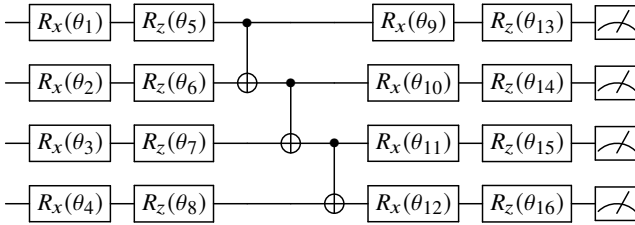


Abb. 1: EfficientSU2 Schaltkreis mit CNOT-Gattern in der Verschränkungsschicht, sowie R_x und R_z Gattern in der SU2-Schicht

Als Observable wurde in den Experimenten eine einfache Diagonalmatrix, wie in Gleichung (4) dargestellt, verwendet. Die Eigenwerte wurden als $\lambda_i = 2i$ gewählt, ähnlich zur globalen Observable im Variational Quantum State Eigensolver [Ce20], bei der die Eigenwerte ebenfalls monoton steigen ($\lambda_i < \lambda_{i+1}$).

$$O_{Diag}(\lambda_1, \dots, \lambda_N) = \begin{pmatrix} \lambda_1 & 0 & \dots & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & \lambda_N \end{pmatrix} \quad (4)$$

Der Gradient ist immer abhängig von der Parameterposition an der gemessen wird. Eine Abtastung des gesamten Parameterraumes mit einem feinen Gitter, um allgemeine Aussagen

treffen zu können, ist anhand der Laufzeiten einzelner Messungen praktisch nicht möglich. Daher wurden in dieser Arbeit gleichverteilte zufällige Parameterpunkte für die Experimente genutzt.

Da die Experimente als Grundlage zur Anwendung auf NISQ-Quantencomputern dienen sollen, kommen noch zusätzliche Faktoren wie begrenzte Anzahl Shots und verrauschtes Ausführen der Schaltkreise hinzu. Da alle Experimente auf dem Simulator ausgeführt wurden, spielt Rauschen für die Experimente keine Rolle. In der Praxis sollten die einfachen Verfahren ähnlich von Rauschen und ungenauen Gattern betroffen sein, da die untersuchten Methoden die gleichen Schaltkreise verwenden und sich nur in Parameterwerten und klassischen Nachberechnungen unterscheiden.

In den Experimenten wurde im Detail untersucht, wie sich verschiedene Shotzahlen und die Wahl der Hyperparameter der Verfahren auf die Gradientenapproximation auswirken.

Für SPSA und zentrale finite Differenzen wurde ein Abstand von $h = 0.01$ gewählt. Da SPSA weniger Shots als die anderen Verfahren verwendet, wurde der Mittelwert über 10 Ausführungen gebildet.

Die hier besprochenen Ergebnisse wurden mit 4 Qubits und dem Ansatz aus Abb. 1 mit einer zusätzlichen Verschränkungs- und 1-Qubit Gatter Schicht generiert. Es wurden verschiedene Anzahlen an Schichten mit unterschiedlichen Arten von Verschränkungsschichten und 1-Qubit Gattern getestet. Dies schien aber keinen signifikanten Einfluss auf das Verhältnis zwischen den Fehlern der verschiedenen Gradientenverfahren zu haben.

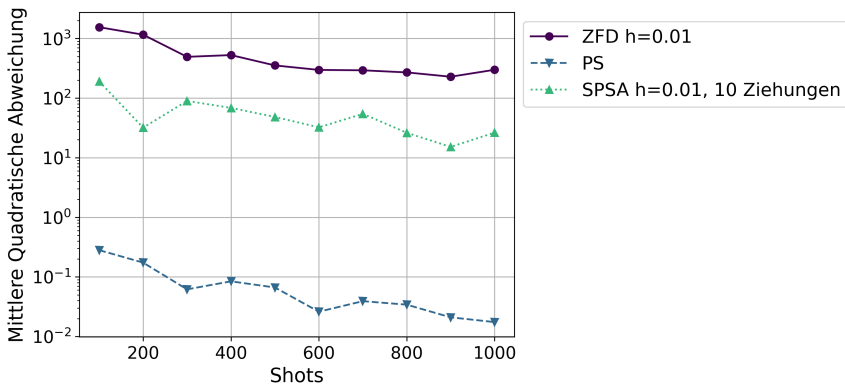


Abb. 2: Durchschnittlicher quadratischer Fehler über alle partiellen Ableitungen an zufälligem Parameterpunkt für Parameter-Shift (PS), zentrale finite Differenzen (ZFD) und SPSA (gemittelt über 10 Approximationen). Die Anzahl an Shots wurde für jede individuelle Messung verwendet.

Aus den in Abb. 2 gezeigten Ergebnissen der Experimente lassen sich einige Hypothesen über den allgemeinen Einsatz der Verfahren formulieren:

Hypothese 1: Parameter-Shift berechnet deutlich bessere Approximationen als die numerischen Verfahren.

Hypothese 2: SPSA berechnet bessere Approximationen als finite Differenzen.

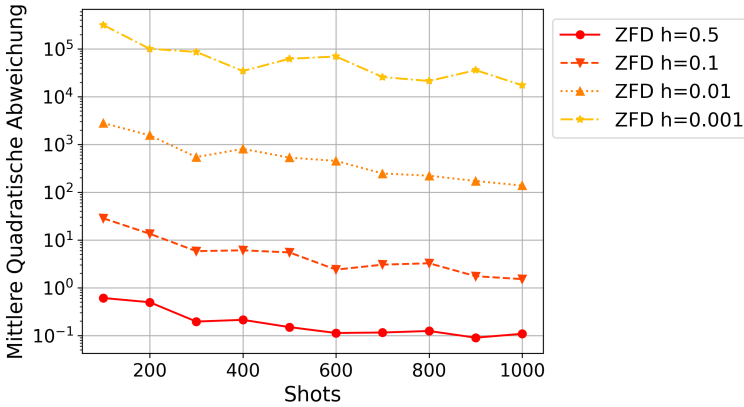


Abb. 3: Durchschnittlicher quadratischer Fehler über alle partiellen Ableitungen an zufälligem Parameterpunkt für zentrale finite Differenzen (ZFD) mit verschiedenen Abstandswerten h . Die Anzahl an Shots wurde für jede individuelle Messung verwendet.

Außerdem ließ sich aus weiteren Experimenten, deren Ergebnisse in Abb. 3 dargestellt werden, die folgende Hypothese folgern:

Hypothese 3: Finite Differenzen mit kleinerem h berechnet schlechtere Approximationen.

Hypothese 2 entspricht dabei nicht den Erwartungen zu SPSA in anderen Problemstellungen, da der eigentliche Vorteil von SPSA darin besteht, schlechtere, aber dafür weniger aufwendigere Approximationen iterativ auszugleichen. Auch hat SPSA bei den Versuchen insgesamt weniger Shots benötigt als alle anderen Verfahren. Bei $k = 24$ Parametern benötigten die anderen Verfahren $2k = 48$ Messungen. SPSA benötigt zwei Messungen pro Gradientenapproximation. Da über 10 Approximationen der Mittelwert gebildet wurde, wurde insgesamt also nur 20 gemessen.

Auch Hypothese 3 scheint unintuitiv, da man normalerweise $0 < h < 1$ möglichst klein wählt, um den Fehlerterm $O(h^2)$ in Gleichung (1) zu verringern. Wenn man spezifisch zentrale finite Differenzen als den Differenzenquotienten der Ableitungsdefinition betrachtet, ergibt die Wahl eines kleineren Abstands ebenfalls Sinn, da ein kleineres h einem besseren Annähern des Grenzwertes, also der exakten Ableitung entspricht. Auf einem Computer entstehen ab einer gewissen Nachkommastelle, aufgrund der begrenzten Darstellungsmöglichkeit von Fließkommazahlen, Rundungsfehler. Wenn Abstände h diese Größenordnung erreichen, können diese Rundungsfehler größere Approximationsfehler zur Folge haben. In den

Experimenten führten kleinere h aber schon zu immer schlechteren Ergebnissen, bevor Rundungsfehler eine Rolle spielten.

Auffällig ist, dass dieser Fehler kontinuierlich mit mehr Shots sinkt. Weitere Experimente zeigten, dass finite Differenzen mit kleineren Abständen deutlich mehr Shots benötigt um zu konvergieren.

3.2 Analyse

Aufgrund des durch die Messungen entstehenden Zufalls lassen sich die konkreten Gradientenberechnungen als Zufallsvariablen auffassen und deren Fehler in Bias und Varianz aufteilen. Der Bias entspricht dabei dem Fehler des Erwartungswertes des Verfahrens. Die Varianz beschreibt die zu erwartende quadratische Abweichung vom Bias. Messungen mit mehr Shots verringern die Varianz des Messergebnisses. Für unendlich Shots geht die Varianz gegen 0; man misst also sicher den Erwartungswert und es bleibt lediglich der Bias als Restfehler.

In den Experimenten mit NISQ-üblichen Shotzahlen von bis zu 1000 Shots war jedoch nicht der Bias, sondern die Varianz für den Großteil des Fehlers verantwortlich. Der Bias verhält sich den Erwartungen entsprechend: SPSA hat dort den größten Fehler, Parameter-Shift ist exakt und für finite Differenzen sinkt der Bias mit geringerem Abstand h .

Die Varianz der Messung eines Quantenschaltkreises $L(\theta)$ ist bekannt und wird auch als One-Shot Varianz [Hu22] bezeichnet, da es die Varianz bei einer Messung mithilfe eines einzigen Shots beschreibt. Daraus lassen sich direkt die Varianzen von Parameter-Shift und zentralen finiten Differenzen erschließen, wenn in den Messungen nur ein einzelner Shot verwendet wird.

$$\text{Var}(g_{iPS}) = r^2(\text{Var}(L(\theta + \frac{\pi}{4r})) + \text{Var}(L(\theta - \frac{\pi}{4r}))) \quad (5)$$

$$\text{Var}(g_{iCFD}) = \frac{1}{4h^2}(\text{Var}(L(\theta + h)) + \text{Var}(L(\theta - h))) \quad (6)$$

Die Varianz nach n Shots pro Messung lässt sich daraus als $\frac{\text{Var}(g_i)}{n}$ berechnen. Solange nicht $h = \frac{\pi}{4r}$ gewählt wird, messen Parameter-Shift und zentrale finite Differenzen an unterschiedlichen Parameterpunkten. Man kann dann die beiden Varianzen daher eigentlich nicht direkt vergleichen, da sie abhängig von den Varianzen an den Messpunkten sind. Folgende Annahme aus [MBK21] erlaubt jedoch eine andere Perspektive:

Annahme 1: Die Varianz der Messung in einer Observable hängt nur schwach von der Verschiebung des Parameterwertes ab, sodass $\text{Var}(L(\theta + h)) + \text{Var}(L(\theta - h)) \simeq 2\text{Var}(L(\theta))$ für alle Werte von h gilt.

Für diese Annahme lässt sich leicht ein Gegenbeispiel konstruieren. Eine empirische Untersuchung in Form einer Stichprobe an verschiedenen zufälligen Parameterpunkten unterstützt jedoch die Annahme, gerade bei kleinen Abständen. Oft unterscheiden sich $\text{Var}(L(\theta + h))$, $\text{Var}(L(\theta - h))$ und $\text{Var}(L(\theta))$ kaum voneinander.

Eine mögliche Erklärung für dieses Verhalten könnte hier das Barren Plateau Phänomen [Mc18] sein. Die Anzahl an Zuständen, deren Erwartungswerte sich vom durchschnittlichen Erwartungswert signifikant unterscheiden, sinkt exponentiell mit der Anzahl an Qubits. In Plateaus aus Zuständen mit durchschnittlichen Erwartungswerten ist dann der Gradient sehr gering, jedoch kein Extremum vorhanden.

Eine Idee ist daher, auch wenn das Problem mit 4 Qubits noch relativ klein ist, das dieses Phänomen hier bereits Auswirkungen hat. Erst wenn man sich an dem Punkt θ , beziehungsweise den Verschiebungen um h , einem Eigenzustand der Observablen nahe genug annähert, unterscheidet sich die Varianz der Messung deutlich von der durchschnittlichen Varianz.

Wenn man davon ausgeht, dass ein gleich verteilter zufälliger Parameterpunkt circa zu einem gleich verteilten zufälligen Zustand führt, ist die Chance deutlich höher einen Zustand mit durchschnittlicher Varianz zu treffen, als einen Zustand in der Nähe eines Eigenvektors. Aufgrund der empirischen Ergebnisse wird die Annahme als zutreffend angenommen und für die weitere Analyse verwendet.

$$\text{Var}(g_{iPS}) \simeq 2\sigma^2 r^2 \quad (7)$$

$$\text{Var}(g_{iCFD}) \simeq \frac{\sigma^2}{2h^2} \quad (8)$$

Gleichung (7) und Gleichung (8) zeigen nun die Varianzen der Verfahren mit fester Varianz σ^2 für alle Messungen. Wie zu sehen, hängt die Varianz nun nur von den Faktoren r und $\frac{1}{2h}$, die in den ursprünglichen Gradientenformeln (Gleichung (1), Gleichung (3)) die Messwerte skalieren, ab. Für die numerischen Verfahren wird die Varianz mit $O(h^{-2})$ skaliert. Da üblicherweise ein Abstand $0 < h \ll 1$ gewählt wird, um einen geringeren Bias zu produzieren, wird die Varianz aufgebläht. Wenn man die Parameter aus den Experimenten ($h = 0.01$, $r = \frac{1}{2}$ für Rotationsgatter) einsetzt, erzielt das zentrale finite Differenzen Verfahren erst mit 10000 Shots die Varianz, die Parameter-Shift mit einem einzigen Shot generiert.

Dadurch lassen sich die Hypothesen 1 und 3 erklären. Kleinere Abstände führen zu größeren Varianzen, die mehr Shots benötigen, um sich dem Erwartungswert anzunähern. Da in den Experimenten der Fehler aus der Varianz dem des Bias überwiegt, lieferten die größeren Abstände bessere Ergebnisse. Die guten Resultate von Parameter-Shift lassen sich auf das kleine r der Rotationsgatter zurückführen. Das Verfahren hat keine allgemein geringere Varianz als die numerischen Verfahren.

SPSA muss gesondert betrachtet werden, da es zwei Arten von Varianzen hat. Je nach Abstandsvektor Δ unterscheidet sich der Erwartungswert und die Varianz von g_i , da an unterschiedlichen Parameterpunkten gemessen wird. Analog zum Vorgehen im vorherigen Abschnitt lässt sich die Varianz einer Approximation g_i bestimmen, sobald ein spezifisches Δ gewählt wurde:

$$\text{Var}(g_i(\Delta)_{SPSA}) = \frac{\text{Var}(L(\theta + h\Delta)) + \text{Var}(L(\theta - h\Delta))}{4h^2} \quad (9)$$

Die erste Art von Varianz ist $\sigma_M^2 = \mathbb{E}_\Delta[\text{Var}(g_i(\Delta)_{SPSA})]$, die zu erwartende Varianz eines $g_i(\Delta)$ aufgrund der inhärenten Varianz der Messungen.

Aber auch die Erwartungswerte der Approximationen $\mathbb{E}[g_i(\Delta)]$ schwanken um den gesamten Erwartungswert $\mathbb{E}_\Delta[\mathbb{E}[g_i(\Delta)]]$ über alle möglichen Δ . Diese zweite Art der Varianz $\sigma_\Delta^2 = \mathbb{E}_\Delta[(\mathbb{E}[g_i|\Delta] - \mathbb{E}_\Delta[\mathbb{E}[g_i(\Delta)]])^2]$ sagt aus, wie groß die zu erwartende quadratische Abweichung zum Erwartungswert ist, wenn alle Messungen perfekt wären.

Die Gesamtvarianz von SPSA ergibt sich dann als:

$$\text{Var}(g_i_{SPSA}) = \sigma_\Delta^2 + \sigma_M^2 \quad (10)$$

In den Experimenten ging der überwiegende Teil der Varianz von der durchschnittlichen Messvarianz σ_M^2 aus. Unter der vorherigen Annahme, dass die Varianz einzelner Messungen konstant mit σ^2 ist folgt:

$$\sigma_M^2 = \mathbb{E}_\Delta[\text{Var}(g_i(\Delta)_{SPSA})] = \sum_{j=0}^{2^k} p(\Delta_j) \text{Var}(g_i(\Delta_j)_{SPSA}) \simeq \frac{1}{2^k} \sum_{j=0}^{2^k} \frac{2\sigma^2}{4h^2} = \frac{\sigma^2}{2h^2} \quad (11)$$

σ_M^2 ist somit identisch zur Varianz von zentralen finiten Differenzen. Der entscheidende Punkt, warum SPSA eine geringere Varianz besitzt, besteht darin, dass jede Messung die Varianz in allen Dimensionen verringert. Während bei zentralen finiten Differenzen $2k$ Shots benötigt werden, um die Messvarianz σ_M^2 in jedem Parameter zu erzeugen, braucht SPSA nur 2 Messungen und generiert zusätzlich nur die Varianz σ_Δ^2 , welche basierend auf den Experimenten vernachlässigbar ist. Somit lässt sich schlussendlich auch Hypothese 2 erklären.

4 Zusammenfassung und Ausblick

In dieser Arbeit wurden verschiedene Verfahren zur Gradientenberechnung auf Quantencomputern untersucht. Als numerische Verfahren wurden finite Differenzen, sowie das

Gradientenapproximationsverfahren des SPSA Algorithmus und als quantencomputerspezifisches analytisches Verfahren das Parameter-Shift Verfahren vorgestellt.

In Experimenten wurden die Verfahren verglichen und Beobachtungen als Hypothesen formuliert und weiter analysiert. Zuerst fiel auf, dass der größte Fehler (bei NISQ üblichen Shotzahlen) durch die Varianz entsteht und von den Hyperparametern der Verfahren abhängt. Bei finiten Differenzen führt dies zu einem Dilemma bei der Wahl des Abstands h , da dort die Varianz mit $O(h^{-2})$, der Bias jedoch mit $O(h^2)$ skaliert. Diese Behauptungen stützen sich auf eine Annahme aus [MBK21], wonach die Varianzen von Messungen an einem Parameterpunkt ähnlich sind zu denen an Parameterpunkten nach beliebigen Shifts.

Im Detail wurde die Zusammensetzung der Varianz von SPSA besprochen. Auch hier überwiegt die inhärente Varianz der Messungen, welche ebenfalls mit $O(h^{-2})$ skaliert. SPSA erzeugt diese Varianz unabhängig von der Anzahl an Parametern mit nur zwei Messungen, wodurch im Verhältnis mehr Shots auf einzelne Messungen angewandt werden können, was die Varianz in allen Parametern reduziert. Diese Unabhängigkeit von der Parameteranzahl kann gerade für zukünftige, größere, variationelle Quantenalgorithmen hilfreich sein.

Neben Parameter-Shift gibt es Verfahren, die bezüglich beliebiger Gatter ableiten können [Cr19][Sc19][BC21], welche jedoch in dieser Arbeit nicht weiter behandelt wurden. Es gilt die Qualität und den Aufwand dieser Verfahren mit den Resultaten der in dieser Arbeit besprochenen numerischen Verfahren in Vergleich zu setzen.

Es bleibt offen die Ergebnisse dieser Arbeit und der unterliegenden Annahme [MBK21] mit dem Barren Plateau Phänomen in Kontext zu setzen. Außerdem gibt es auch Ideen, wie man Barren Plateaus bei der Optimierung umgehen kann. So kann beispielsweise durch Entropiemessungen festgestellt werden, ob man sich einem Barren Plateau nähert und so versucht werden, diesen auszuweichen [Sa22]. Eine andere Idee sind spezialisierte, problemspezifische Ansätze, ähnlich zu problemspezifischen Architekturen von neuronalen Netzen. Kombiniert mit einem problemspezifischen Initialzustand, könnte dies zu einer lokaleren Optimierung führen, die nicht von Barren Plateaus betroffen ist.

Literatur

- [AN21] ANIS, M. S. et al.: Qiskit: An Open-source Framework for Quantum Computing, 2021.
- [BC21] Banchi, L.; Crooks, G. E.: Measuring Analytic Gradients of General Quantum Evolution with the Stochastic Parameter Shift Rule. Quantum 5/, S. 386, 2021.
- [Ce20] Cerezo, M.; Sharma, K.; Arrasmith, A.; Coles, P. J.: Variational quantum state eigensolver. arXiv preprint arXiv:2004.01372/, 2020.
- [Cr19] Crooks, G. E.: Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. arXiv preprint arXiv:1905.13311/, 2019.

- [Du20] Du, Y.; Hsieh, M.-H.; Liu, T.; Tao, D.: Expressive power of parametrized quantum circuits. *Phys. Rev. Research* 2/, Juli 2020.
- [FGG14] Farhi, E.; Goldstone, J.; Gutmann, S.: A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*/, 2014.
- [Hu22] Hubregtsen, T.; Frederik, W.; Qasim, S.; Eisert, J.: Single-component gradient rules for variational quantum algorithms. *Quantum Science and Technology* 7/, Apr. 2022.
- [Ka17] Kandala, A.; Mezzacapo, A.; Temme, K.; Takita, M.; Brink, M.; Chow, J.; Gambetta, J.: Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* 549/, S. 242–246, Sep. 2017.
- [MBK21] Mari, A.; Bromley, T. R.; Killoran, N.: Estimating the gradient and higher-order derivatives on quantum hardware. *Phys. Rev. A* 103/, Jan. 2021.
- [Mc18] McClean, J. R.; Boixo, S.; Smelyanskiy, V. N.; Babbush, R.; Neven, H.: Barren plateaus in quantum neural network training landscapes. *Nature communications* 9/1, 2018.
- [NM65] Nelder, J. A.; Mead, R.: A simplex method for function minimization. *The computer journal* 7/4, S. 308–313, 1965.
- [Pe14] Peruzzo, A.; McClean, J.; Shadbolt, P.; Yung, M. H.; Zhou, X.; Love, P.; Aspuru-Guzik, A.; O’Brien, J.: A variational eigenvalue solver on a photonic quantum processor. *Nature communications* 5/1, 2014.
- [Pr18] Preskill, J.: Quantum Computing in the NISQ era and beyond. *Quantum* 2/, S. 79, Aug. 2018.
- [Sa22] Sack, S. H.; Medina, R. A.; Michailidis, A. A.; Kueng, R.; Serbyn, M.: Avoiding barren plateaus using classical shadows. *arXiv preprint arXiv:2201.08194*/, 2022.
- [Sc19] Schuld, M.; Bergholm, V.; Gogolin, C.; Izaac, J.; Killoran, N.: Evaluating analytic gradients on quantum hardware. *Phys. Rev. A* 99/, März 2019.
- [Sm85] Smith, G. D.: Numerical solution of partial differential equations: finite difference methods. Oxford university press, 1985.
- [Sp87] Spall, J. C.: A stochastic approximation technique for generating maximum likelihood parameter estimates. In: 1987 American control conference. IEEE, S. 1161–1167, 1987.
- [Sw20] Sweke, R.; Wilde, F.; Meyer, J.; Schuld, M.; Fährmann, P.; Meynard-Piganeau, B.; Eisert, J.: Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum* 4/, S. 314, Aug. 2020.

Gaining insights into the information distribution of Light Fields and enabling adaptive Light Field processing

Robin Kremer ¹

Abstract: Thanks to smartphones with several cameras, capturing a scene from multiple view points has become increasingly more available. Together with the evolving computing capabilities of modern hardware, light field processing has gained a lot of attention in the last years [Br20; F119; Mi20]. These techniques rely on neural networks to generate representations of the light field data. Other work assumes certain scene properties to enable light field processing (like lambertian radiation). The work shown here uses depth maps to transform the light field into a froxel (frustum + voxel)[Ev15] centered representation enabling unique post processing steps and analysis of the ray distribution in a scene. But more importantly it paves the way to quantify the information distribution within a scene. Based on this information appropriate adaptive filtering techniques can be applied. The transformation into the froxel centric representation is compatible with techniques like NERF.

Keywords: Lightfields; Froxels; Light Fields; Frustum; Voxel; Neural Radiance Field; Ray Classification

1 Introduction

Thanks to the increase in processing power in the last years, concepts like neural networks have gained a lot of popularity even though their theory has been around for much longer. A similar development is occurring on the topic of plenoptic capture and light field processing. Recent approaches even combined the two fields [F119] [Br20] [Mi20]. Other approaches applied more traditional processing techniques to achieve results like super resolution [LS20] or denoising [AS17]. The later techniques assume that a scene consists out of lambertian radiators, that is objects don't change color when viewed from different angles. This assumption may be false and can lead to visual artifacts.

With the proposed transformation to a froxel centered representation, the information content of regions in the scene can be identified. Furthermore, objects can be classified as lambertian or non-lambertian allowing finer control of succeeding processing steps. Traditionally the most common way to represent light field images is the two-plane parameterization $P = P(s, t, u, v)$ [DB03] ². Which itself is a simplification of the 7 dimensional plenoptic function $P = P(V_x, V_y, V_z, \Phi, \Theta, \lambda, t)$ [BA91]. This reduction is done by converting the angular coordinates Φ and Θ to Cartesian coordinates u and v . One spatial dependency is removed because light fields are only captured at a certain plane (by a camera gantry or

¹ Saarland Informatics Campus, Telecommunications Lab, Saarland University, 66123 Saarbrücken, Germany

² <https://github.com/doda42/LFToolbox>

plenoptic camera) and it is assumed that the rays don't significantly change while travelling from a scene object to the camera. The two remaining spatial dependencies are renamed to s and t . The time dependency is removed because most of the work is performed on still images while the wave length dependency is removed by creating a light field for each captured color channel.

A recent alternative to the two-plane parameterization are Multi-plane images (MPI) [Fl19] or Multi-sphere images (SPI) [Br20]. In these techniques the information of the light field is stored in layered meshes. The shape and color of these meshes is generated by a neural network which tries to minimize the difference between the original light field images and the reconstructed ones. While these approaches as a whole can process certain non-lambertian surfaces the data stored in each mesh layer is not view point dependent. As noted by the authors this leads to problems in displaying some visual phenomena like curved reflective surfaces.

In Contrast NeRFs [Mi20] store the information of the light field in the weights of a trained network. Rendering a scene works by sampling this network for points along camera rays. An advantage over layered mesh approaches is that NeRFs store view point dependent colors for a scene point. Which greatly improves the visual quality of non-lambertian radiators and can handle phenomena like curved reflective surfaces. Furthermore NeRFs don't store the color information at one specific scene point but work with a density based approach. This allows rays to change color while travelling from an object to the observer and approximating the underlying plenoptic function more closely.

As mentioned earlier the ongoing advances in computational power enabled the processing of light field images. With modern hardware even the processing of light field videos becomes feasible and thereby reintroducing the time dependency t . Taking light fields to 5 Dimensions. While the resulting amount of data increases significantly so do the available processing techniques. As an example the cameras of a light field video capture array don't have to be triggered at the same time. It is possible to offset different groups of cameras and thereby increasing the temporal resolution of the whole array. This idea is introduced as the concept of sub-framing.

2 Theory of Foxel representation

The distribution of information in a light field heavily depends on the scene geometry and the surface properties of the objects present. In the typical two-plane representation, information about both of these aspects is hard to extract. While depth maps give exceptional insights into the geometries in a scene, surface properties are still a problem. Combining both the two-plane parameterization and depth maps into a foxel centered representation gives direct access to both scene geometry and surface properties. This enables filtering techniques which are suited to the information content available and helps to identify parts of a scene where lots of redundancy is present (lambertian radiators).

The first step of the transformation is the creation of a discretization raster for the scene.

This consists out of froxels whose sizes are chosen in such a way as to perfectly match the capabilities of the capturing system. The width w_{froxel} and height h_{froxel} of a froxel are proportional to the pixel size p_{pixel} , the distance of the froxel from the camera D_{plane} and inversely proportional to the focus distance f_d of the capturing system (for square pixels $w_{froxel} = h_{froxel}$).

$$w_{froxel} = h_{froxel} = \frac{p_{pixel} D_{plane}}{f_d} \quad (1)$$

The depth of a froxel is based on disparity achieved by the capture system.

$$d_{froxel} = D_{plane}^2 / \left(\frac{f_d \cdot \max(a_{max} \cdot a_{spacing}, b_{max} \cdot b_{spacing})}{p_{pixel}} - D_{plane} \right) \quad (2)$$

Equation 2 is used to calculate the depth of a froxel at a distance D_{plane} from the capture system. $\max(a_{max} \cdot a_{spacing}, b_{max} \cdot b_{spacing})$ is the largest distance between two capture points of the system. Figure 1 illustrates that the largest baseline distance is the defining baseline for the froxel depth. This approach currently only works on planar capture systems. The resolution of the resulting discretization raster will perfectly match the capabilities of the capture array, such that no two rays captured by one camera will be assigned to the same froxel. The allocating of rays to froxels is the transformation from the two-plane parameterization to the froxel centered representation. In order to assign the rays to the froxels the s, t, u, v parameters and a depth map of the scene are used. With this the origin of a ray in the scene can be calculated. The previously generated discretization raster is then applied to these positions to assign all rays to their corresponding froxel. The result is a new representation in the way of a froxel centric data structure where a ray is not identified by a camera and pixel position (like in the two-plane parameterization) but identified by its origin in the scene. Note that for reconstruction purposes the camera number which captured the ray is stored alongside the color information. A scene point sampled by different cameras will be assigned to the same froxel after the transformation. This allows direct insight into which parts of a scene are only sampled by a few cameras and therefore unsuited for techniques like super resolution or denoising, while scene points for which lots of samples exist can be further analysed. For example the color distribution of all rays from a single froxel gives insight into the lambertian properties of the corresponding scene point. Transforming the light field from the froxel representation back to the two-plane parameterization or generating view points of the light field is done by selecting all froxels in a given view frustum. The rays assigned to these froxels are then projected onto a virtual camera sensor placed at the wanted view point. Note that this view point doesn't have to be at a position originally sampled by the capture system but can also be a novel one.

3 Implementation

A first implementation was done in Matlab[®] where concepts like the transformation to the froxel representation and further analysis were explored. Due to the chosen underlying data

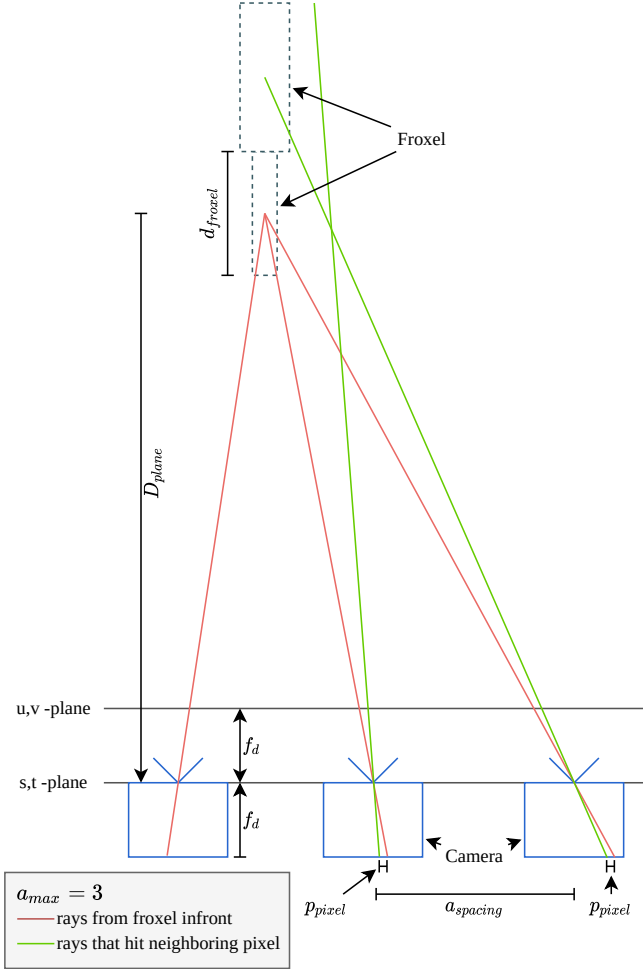


Fig. 1: Top view onto a multi-camera array with two froxels shown

structure and limitations of Matlab[®] this implementation turned to be prohibitively slow for further development.

After a small experiment to compare different programming languages, python together with the just-in-time compiler numba was used for a reworked implementation. Thanks to the performance improvements this implementation is also capable of working with 5D light fields (light field videos) where the dependency on time t is reintroduced. As the underlying data structure a dictionary/map is used, for which the keys are generated based on the spatial and temporal position of a froxel. This allows for efficient storage of the sparsely sampled scene space. After the transformation, the found froxels can be categorised based on the

color distribution of the contained ray-set. This is currently done with a standard deviation and z-score threshold where the first one classifies non-lambertian froxels and the second one finds outliers in non-lambertian froxels. In the 5D implementation this analysis can also be done over time. After this classification different post processing techniques can be applied. As an example a ray reduction filter is implemented which reduces the number of rays saved per froxel by applying a median filter to all rays classified as lambertian. While rays classified as non-lambertian or outliers are not reduced. The rendering of light field images can then be performed on this reduced set of total rays without a significant loss in visual quality.

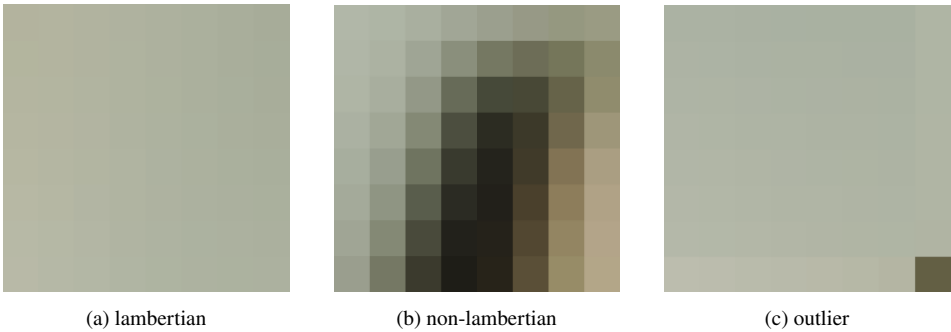


Fig. 2: Example data stored in a froxel from a 4D light field captured on a 8×8 camera array.

The quality of the depth maps is important so that one scene point sampled by different cameras is assigned to the same froxel. Due to this the development has been done with scenes created in blender where near perfect depth maps are available.

4 Results

The implementation (compare figure 3) was developed and tested on blender demo scenes³. A blender add-on [Ho16] was used to render the color data and depth maps for a 9×9 camera array with a baseline between neighbouring cameras of 70 mm.

After converting the light field from the two-plane parameterization into the froxel representation a fristogram (froxel + histogram) [He21] can be generated. Fristograms give first insights into how data is distributed in a light field. Figure 5 shows a CDF fristogram of the classroom scene. One observation is that over half of the froxels are seen by all 16 cameras of the 4×4 camera array. Furthermore distinct steps each four additional cameras are visible. This occurs because on a regular camera array vertical or horizontal occlusions often effect a whole column or row.

³ <https://www.blender.org/download/demo-files/>

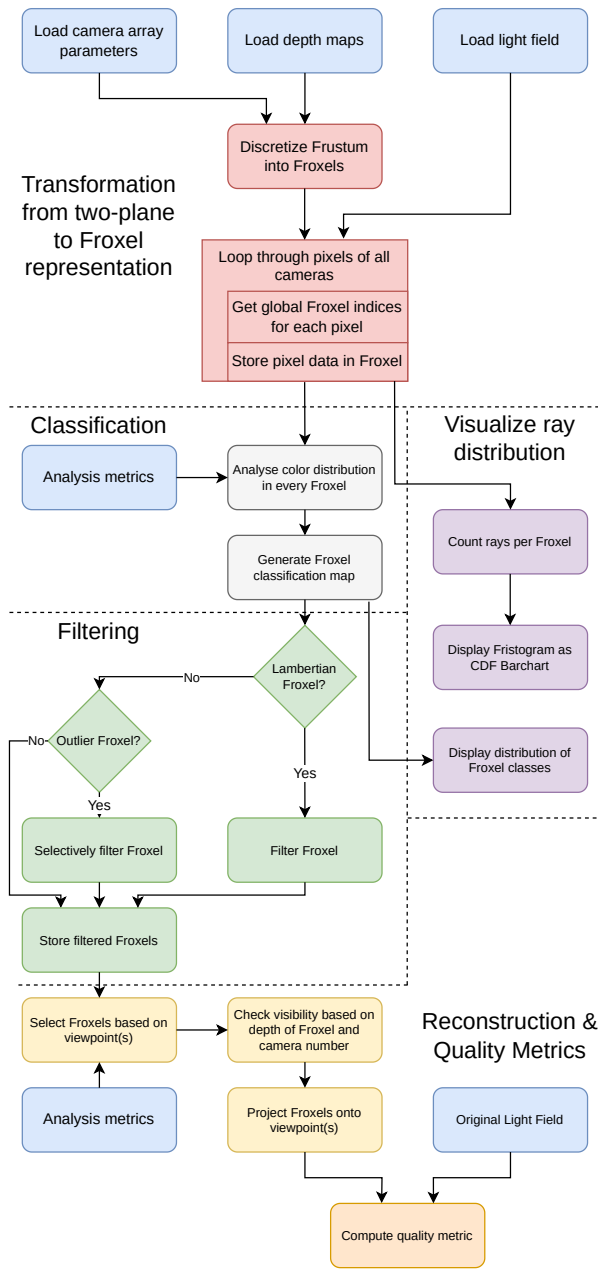


Fig. 3: Flowchart of the foxel transformation



Fig. 4: Classroom scene (left) and BMW scene (right) from blender which were used to develop the pipeline

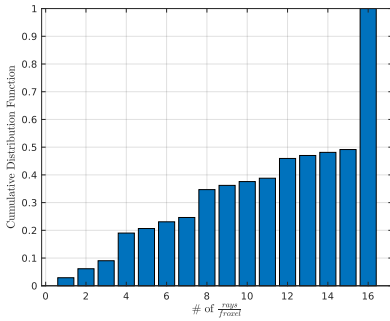


Fig. 5: Fristogram of the classroom scene captured by a 4×4 camera array. Empty froxels are omitted

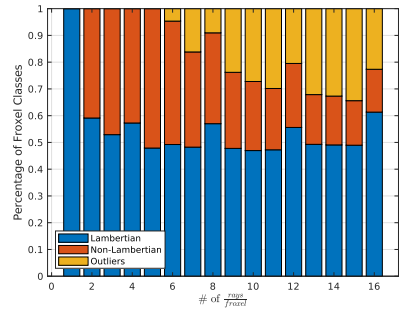


Fig. 6: Distribution of the ray classes over number of rays per froxel (classroom on 4×4 camera array)

In theory without further processing steps like ray reduction a transformation back to the two-plane parameterization should be lossless. As shown in figure 7b the SSIM of the reconstruction shows a slight degradation. This is most likely due to the rounding that happens during discretization and view reconstruction.

Since over half of the froxel are seen by all cameras, there is redundancy present in the light field data. By applying a median or mean filter to all rays present in the froxels this redundancy can be reduced. The total number of rays for the classroom scene captured on a 4×4 camera array is reduced from $4 \times 4 \times 1920 \times 1200 = 36,864,000$ to 3,185,991 (8.64%). Figure 7c shows the result of a mean filter. This technique assumes that one ray is sufficient to describe the color of a froxel, this is true for lambertian scene points (e.g. the wall). In contrast, non-lambertian froxels need more rays to be displayed accurately which is why this reduction leads to visible artifacts.

Because froxels are directly linked to scene points the ray color distribution stored inside each froxel gives insight into the surface properties of that scene point. A low standard deviation among all ray colors of a froxel is an indication that the underlying scene point acts



(a) Original, SSIM: 1.0



(b) No ray reduction, SSIM: 0.9987



(c) Reduction to one ray per Froxel, SSIM: 0.9641



(d) Ray class aware reduction, SSIM: 0.9838

Fig. 7: Image of Classroom scene. The reduction to one ray per Froxel (Figure 7c shows artifacts on non lambertian surfaces. Detail is lost in the reflections of the table and the metal bars.

like a lambertian surface. While a large deviation most likely comes from non-lambertian properties of the surface like reflections or transparency. Applying a threshold to this deviation therefore makes it possible to classify scene points. During the development an outlier class was introduced next to lambertian and non-lambertian for froxels where only very few rays deviate from the mean (compare figure 2c). This classification is based on the z-score of all rays. With this information a surface property aware ray reduction is possible. By choosing a standard deviation threshold of 2.0 and a z-score threshold of 4.0 the total number of rays left after the reduction increases to 10,483,959 for the classroom scene. Which still is only 28.4% of the original ray count. This increase yields an SSIM improvement from 0.96419 to 0.98384 (figure 7d). The distribution of the ray classes over the number of rays per froxel can be seen in figure 6.

For 5D light fields the froxel analysis can also be extended over the time domain. This enables finer classification of scene points. As a demonstration a few moving objects were added to the classroom scene and a short animation was rendered (compare figure 8). This leads to a new class of objects that change their appearance not only when viewed from a different angle but also change it over time (compare figure 9). This information can be utilized by further post processing steps like a 5D ray reduction. The froxel representation also provides direct insight into how sub-framing a camera

array trades spatial resolution for temporal resolution. As a demonstration the same animation as before was rendered but with a sub-frame pattern applied to the camera array. The chosen sub-frame pattern consists out of four groups (each array column is one group). The trigger time points were then uniformly distributed in one frame interval. This practically increases the temporal resolution by a factor of four while each camera still captures at the original frame rate. The resulting froxel data is shown in figure 10.

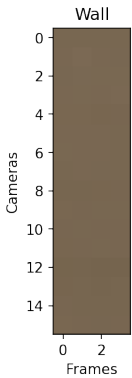


(a) First picture of the animation

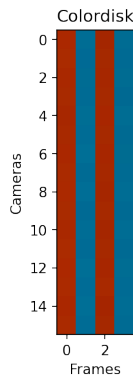


(b) Last picture of the animation

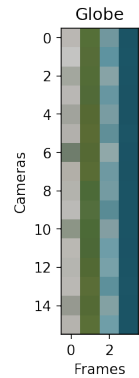
Fig. 8: As a demonstration for the 5D capabilities a few moving objects were added to the classroom scene. Note that color wheel looks static because it's rotational speed is synchronized to the frame rate.



(a) Wall - lambertian and not changing over time



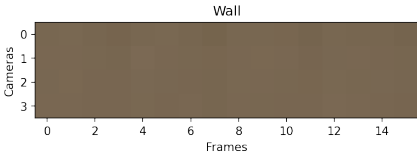
(b) Color disk - lambertian and changing over time



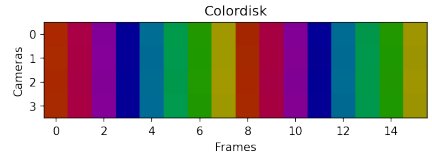
(c) Globe - non-lambertian and changing over time

Fig. 9: All 16 cameras of the 4×4 camera array are triggered at the same time and capture the scene with a constant frame rate (only 4 frames shown). Along the y-axis the ray color captured by each camera is display while the x-axis shows the change over time.

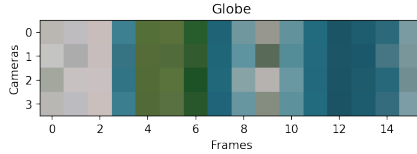
In conclusion the proposed transformation to a froxel centric light field representation allows for deeper insights into how data is distribution in a light field. It not only allows to identify regions of a scene which are suitable for certain post processing techniques like super resolution but also is able to classify the surface properties of these scene points. As a demonstration of the capabilities the total number of rays of a light field were reduced



(a) Wall - lambertian and not changing over time



(b) Color disk - lambertian and changing over time



(c) Globe - non-lambertian and changing over time

Fig. 10: The camera array columns are staggered against each other increasing the temporal resolution by a factor of four but decreasing the spatial resolution. Note that the overall captured time is the same for both cases.

significantly while retaining a very high visual quality. Furthermore the proposed technique can also be applied to light field videos which increases the possible post processing options even more.

5 Future Work

A big advantage of the proposed froxel based representation is the compatibility with other formats. As mentioned the implementation is currently only tested on virtual scenes because of the high quality depth maps. For real scenes it is proposed to use depth maps created by a NeRF network. While with the introduction of NeRFs to the overall pipeline the froxel based representation may seem unnecessary it should be noted that rendering views from NeRFs is very computationally expensive. This is because a NeRF MLP has to be queried more than 100 times for a single pixel resulting in more than 100 million queries for high quality images "on an NVIDIA V100, this takes approximately 30 seconds per frame" [Mi20]. In contrast rendering a view point from a froxel based representation in a comparable quality takes 200 ms on a single core of an Intel i7-7700 CPU @ 3.60GHz.

Another feature of NeRFs is the use of a volume rendering approach. While the current view rendering of the froxel based representation works by only rendering the froxels closest to the camera. NeRFs therefore stay more true to the underlying plenoptic function where rays can change while travelling from an object to the observer. A volume density aware renderer is also compatible with the idea of the froxel based representation. Making it possible to further combine both ideas in order to achieve good visual quality and low computational complexity.

The current implementation together with the used data sets will be made available as open source.

References

- [AS17] Alain, M.; Smolic, A.: Light field denoising by sparse 5D transform domain collaborative filtering. In: 2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP). IEEE, pp. 1–6, 2017.
- [BA91] Bergen, J. R.; Adelson, E. H.: The plenoptic function and the elements of early vision. *Computational models of visual processing* 1/, p. 8, 1991.
- [Br20] Broxton, M.; Flynn, J.; Overbeck, R.; Erickson, D.; Hedman, P.; Duvall, M.; Dourgarian, J.; Busch, J.; Whalen, M.; Debevec, P.: Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)* 39/4, pp. 86–1, 2020.
- [DB03] Dansereau, D.; Bruton, L.: A 4D frequency-planar IIR filter and its application to light field processing. In: *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03. Vol. 4*, IEEE, pp. IV–IV, 2003.
- [Ev15] Evans, A.: Introduction of the term Foxel, <https://www.realtimerendering.com/blog/tag/siggraph-2015/>, Accessed: 2022-05-02, 2015.
- [Fl19] Flynn, J.; Broxton, M.; Debevec, P.; DuVall, M.; Fyffe, G.; Overbeck, R.; Snavely, N.; Tucker, R.: Deepview: View synthesis with learned gradient descent. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Pp. 2367–2376, 2019.
- [He21] Herfet, T.; Chelli, K.; Lange, T.; Kremer, R.: Fristograms: Revealing and Exploiting Light Field Internals. *arXiv preprint arXiv:2107.10563*, 2021.
- [Ho16] Honauer, K.; Johannsen, O.; Kondermann, D.; Goldluecke, B.: A dataset and evaluation methodology for depth estimation on 4D light fields. In: *Asian Conference on Computer Vision*. Springer, 2016.
- [LS20] Le Pendu, M.; Smolic, A.: High resolution light field recovery with fourier disparity layer completion, demosaicing, and super-resolution. In: *2020 IEEE International Conference on Computational Photography (ICCP)*. IEEE, pp. 1–12, 2020.
- [Mi20] Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: *European conference on computer vision*. Springer, pp. 405–421, 2020.

Methode für Vorhersagen über die Fortführung von Handbewegungen

Philipp Rall,¹ Nicolas Bender²

Abstract: Die vorliegende Arbeit befasst sich mit der Entwicklung einer Methode zur Echtzeit-Vorhersage von Trajektorien seitlicher Greifbewegungen zur Kollisionsvermeidung in der kollaborativen Robotik. Ein Neuronales Netz sagt hierfür anhand des Verlaufs der Anfangsbewegung in einem Regressionsansatz die Endposition und Dauer des gesamten Greifvorgangs voraus. Durch das Minimum Jerk Model für gekrümmte Punkt-zu-Punkt-Bewegungen lässt sich daraufhin der weitere Verlauf der Trajektorie präzise berechnen. Die Arbeit legt besonderen Fokus auf die Entwicklung einer automatisierten Pipeline zur Datenvorverarbeitung, die aufgenommene Rohdaten von natürlichen Greifbewegungen in mehreren modularen Verarbeitungsphasen zur qualitativ hochwertigen und vereinheitlichten Trainingsdaten transformiert sowie fehlerbehaftete Messdaten aussortiert.

Keywords: Mensch-Maschine Interaktion; Minimum Jerk Model; Datenvorverarbeitung; Kollaborative Robotik; Maschinelles Lernen

1 Einleitung

Mit dem Fokus auf innovative Produktionsmethoden und technologische Strategien im Zeichen von Industrie 4.0 werden unter dem Stichwort Human-Robot-Collaboration Szenarien zunehmend realistischer, in denen sich Menschen und Roboter einen gemeinsamen Arbeitsplatz teilen, um kollaborativ Aufgaben bearbeiten zu können.[Ke20] Um diese geteilten Arbeitsplätze für Menschen sicher zu gestalten, ist es unabdingbar, technische Vorsichtsmaßnahmen zu implementieren. Damit soll verhindert werden, dass Menschen von leistungsstarken Robotern verletzt oder empfindliche Roboter von Menschen beschädigt werden.[Ma16]

Eine Möglichkeit eines Kontrollmechanismus ergibt sich aus der Beobachtung der menschlichen Bewegung, die in den Kontext der Bewegungen des Roboters gesetzt werden. Bewegungsvorgänge des Roboters haben den Vorteil gegenüber den menschlichen Bewegungen, dass diese präzise gesteuert, lokalisiert und vorhergesagt werden können. Schwieriger gestaltet sich die Betrachtung der Bewegung des Menschen, die nach von außen nicht zwingend ersichtlichen Einflüssen erfolgt und deshalb kaum von außerhalb kontrolliert werden kann. Die Kontrollmechanismen des Roboters müssen sich daher an die Spontanitäten der menschlichen Bewegungen anpassen und Prognosen für beispielsweise Handbewegungen treffen, um angemessen darauf reagieren zu können.

¹ Technische Universität Darmstadt, Karolinenpl. 5, 64289 Darmstadt, mail@philipp-rall.de

² Universität Heidelberg, Grabengasse 1, 69117 Heidelberg, nicolas.bender@posteo.de

Das Ziel dieser Arbeit ist die Entwicklung und Bewertung einer Methode, mit der sich die Fortführung von Handgelenken vorhersagen lässt. Dabei soll auf Grundlage einer kurzen Bewegungssequenz der weitere Verlauf dieser Bewegung berechnet werden. Unter Zuhilfenahme von Maschinellem Lernen besteht das Ziel darin, ein Modell zu entwickeln, mit dem aus aufgenommenen Anfangsbewegungsdaten einer Armbewegung die Endposition des Handgelenks und Dauer der Bewegung prognostiziert und die Trajektorie berechnet werden kann.

Abbildung 1 stellt das angestrebte Gesamtszenario der Kollisionsvermeidung im Bereich der kollaborativen Robotik dar. Die Kollisionsberechnung sowie Ermittlung von Handlungskonsequenzen für den Roboter sind nicht Thema der Arbeit.

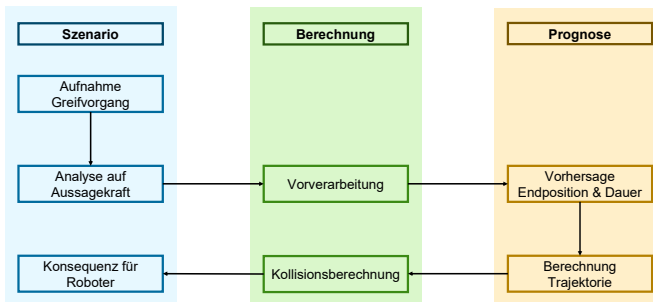


Abb. 1: Angestrebtes Gesamtszenario

2 Grundlagen

2.1 Experimentelles Setup

Zur Datenaufnahme wird ein Szenario der kollaborativen Robotik mit einem Roboter und einer Testperson simuliert. Ziel des Szenarios ist die Übergabe eines Paketes vom Roboter an den Menschen. Als Roboterarm wird das Modell UR5e von Universal Robots verwendet. Der Roboter positioniert das Paket an einer von 346 vorgegebenen Positionen im Raum. Dazu ist der Raum wie von Rettig et al. beschrieben in Würfel mit einer Kantenlänge von 10cm gleichmäßig aufgeteilt.[Re21] Nach Abschluss der Positionierung kann die Testperson das Paket mit beiden Händen greifen und auf einem Tisch ablegen. Diese zielgerichtete seitliche Greifbewegung ist die für die Datenaufnahme relevante Bewegung.

Die Aufnahme der Bewegungsdaten erfolgt über am Körper der Testperson angebrachte reflektierende Cluster aus Markern und ein 120-Hz-Vicon-Kamerasystem. Dieses besteht aus acht Infrarotkameras, welche das von den Markern reflektierte Infrarotlicht aus mehreren Perspektiven erkennen können. Die Cluster werden entsprechend an den in der Arbeit von Rettig et al. [Re21] sowie Murray et al. [Mu99] genannten Positionen des Torsos angebracht.

2.2 Modellierung von Armbewegungen

Die Grundproblematik in der Betrachtung von menschlichen Armbewegungen liegt darin, dass es beliebig viele Möglichkeiten gibt, einen Pfad und die Geschwindigkeit von einem Startpunkt zu einem Endpunkt zu wählen. Für hindernisfreie Bewegungsvorgänge sieht die Literatur die Gemeinsamkeiten für die physikalische Beschreibung in den folgenden vier Aspekten: Ein möglichst glatter und geradliniger Pfad, eine ruckfreie Bewegung, ein unimodales glockenförmiges Geschwindigkeitsprofil und eine Skalierung der Spitzengeschwindigkeit und -beschleunigung sowie deren Zeitpunkte mit der Bewegungsamplitude.[MK99][FH85]

Es wird davon ausgegangen, dass sich die Berechnung eines Bewegungsvorgangs durch das menschliche Gehirn anhand dieser Faktoren annähern lässt. Für den möglichst glatten und geradlinigen Pfad sind Szenarien gemeint, in denen bei einem Greifvorgang ein Endpunkt, eventuell in Form eines Zielobjekts, anvisiert und eine Hand hindernisfrei darauf zu bewegt werden kann. Der Geschwindigkeitsverlauf der Hand ist dabei glockenförmig und enthält in der Spitze eine maximale Geschwindigkeit. Diese Höchstgeschwindigkeit und die maximale Beschleunigung und deren Zeitpunkte sind abhängig von der Entfernung des Endpunkts.

In der Forschung findet sich als etabliertes Modell der Vorhersage von menschlichen Bewegungen das Minimum Jerk Model von Flash und Hogan.[FH85] Diese Methode soll eine mathematische Herangehensweise zur Verfügung stellen, die ohne Einbezug der dahinterliegenden anatomischen Faktoren das Verhalten von willkürlichen menschlichen Bewegungen präzise modellieren kann. Der Grundgedanke dieses Modells ist die Eigenschaft, dass eine menschliche Bewegung einen Bewegungsablauf durchführt, der den kleinsten möglichen Ruck verursacht. Der Ruck ist dabei die dritte Ableitung des Ortes nach der Zeit und physikalisch gesehen damit die Änderung der Beschleunigung. Das Minimum Jerk Model sieht vor, die Trajektorie der Hand bei einem hindernisfreien geradlinigen Greifvorgang zu approximieren und weist als Rahmenbedingung auf, dass die Bewegung aus dem Stillstand startet und sich nur in einer zweidimensionalen Ebene abspielt. Das Modell lässt sich auch für Bewegungen im dreidimensionalen Raum anwenden.[FMv12] Für die Beschreibung eines Bewegungsvorgangs muss das Integral über die Quadrate des Rucks der einzelnen Dimensionen minimiert werden. Somit lässt sich als zu minimierende Kostenfunktion C in Gleichung 1 definieren:

$$C = \frac{1}{2} \int_0^{t_f} \left(\left(\frac{d^3x}{dt^3} \right)^2 + \left(\frac{d^3y}{dt^3} \right)^2 + \left(\frac{d^3z}{dt^3} \right)^2 \right) dt \quad (1)$$

Durch die Minimierung dieser Kostenfunktion, in der x , y und z Funktionen über der Zeit t sind, ergeben sich nach Flash und Hogan [FH85] für verschiedene Arten von Armbewegungen jeweils Funktionen für jede Koordinate, die eine Bewegung mit minimalem Ruck beschreiben. Im Rahmen dieser Arbeit sollen gekrümmte Punkt-zu-Punkt Bewegungen betrachtet werden.

Gekrümmte Punkt-zu-Punkt Bewegungen stellen Bewegungskurven von einem Startpunkt über einen Zwischenpunkt zu einem Zielpunkt dar und können somit einem Hindernis ausweichen. Der Zwischenpunkt sollte dabei die maximale Distanz zur direkten Verbindung von Start- und Endpunkt aufweisen. Zur Beschreibung der Trajektorie ergeben sich durch die Optimierung je Koordinate zwei Funktionen. Eine Funktion für die Punkte vor dem Zwischenpunkt und eine für die Punkte danach.[FH85]

2.3 Datenvorverarbeitung

Die Bestimmung der Geschwindigkeit muss bei Messdaten aufgrund von diskreten Werten numerisch erfolgen. Eine Bildung von Vorwärtsdifferenzen ist in der Literatur dabei als ausreichend angesehen.[Di11] Extreme Ausreißer im Geschwindigkeitsprofil können anhand des Ausreißertests nach Grubbs erkannt werden.[Gr50]

Für die Rauschentfernung der aufgenommenen Messdaten nutzen Experimente den Savitzky-Golay Filter.[Zh21] Darüber hinaus werden auch aufwendige Verfahren wie der Kalman Filter verwendet, die zufriedenstellende Ergebnisse für sehr verrauschte Daten liefern können.[AKG04][Ed17]

Um den Beginn einer Bewegung zu definieren, wird von Zhao et al. [Zh21] vorgeschlagen, alles unter 1% der Maximalgeschwindigkeit des Bewegungsvorgangs als Stillstand anzunehmen. Damit lässt sich ausgehend vom Punkt der höchsten Geschwindigkeit zeitlich die Bewegung bis zu den Punkten eingrenzen, an denen der Stillstand erreicht wird.

3 Methode

Die vorgeschlagene Methode besteht aus den drei Teilen Datenvorverarbeitung, Vorhersage und der Pfadberechnung.

3.1 Datenvorverarbeitung

Die entwickelte Pipeline zur automatisierten Vorverarbeitung von Messdaten natürlicher Handgelenkstrajektorien beinhaltet acht Phasen. Ziel ist die Gewinnung qualitativ hochwertiger und einheitlicher Trainingsdaten.

Interpolation und Glättung Zu Beginn der Pipeline werden bis zu drei aufeinanderfolgende Not-a-Number-Werte (NaN) in der Messreihe der Trajektorie durch ein Polynom dritten Grades interpoliert. Treten in den Messdaten längere NaN-Folgen auf, so ist eine qualitativ aussagekräftige Interpolation nicht mehr möglich. In Messdaten vorliegendes Rauschen kann

mit dem Savitzky-Golay Filter geeignet geglättet werden, der auf polynomialer Regression anhand von Stützstellen basiert. Die Verwendung eines Polynoms dritten Grades und 13 Stützstellen liefert für die vorliegenden Datensätze gute Ergebnisse.

Beschneidung Die Rohdaten der Armbewegungen bestehen aus unterschiedlich langen Datensätzen, die grob zugeschnitten vorliegen. Die Datensätze enthalten einen Abschnitt zu Beginn, in dem sich die Position der Handgelenke, der Ellenbogengelenke und der Schultergelenke kaum verändern. Dieser Zustand spiegelt das lose Hängen der Arme und Hände der Testperson vor der eigentlichen Greifbewegung wider. Darauf folgend beginnt die Phase des Greifens, in der sich die Position der jeweiligen Gelenke verändert. Da für die Auswertung des Greifvorgangs die Daten ohne Bewegung irrelevant sind, sollen diese aus den Datensätzen ausgeschnitten werden. Der Grundgedanke besteht hierbei darin, dass der Zeitraum des Greifvorgangs über das Geschwindigkeitsprofil der Trajektorie ermittelt werden kann.

Angelehnt an das Vorgehen von Zhao et al. [Zh21] soll zunächst ein Schwellwert auf Basis der Maximalgeschwindigkeit der Trajektorie bestimmt werden, der als Stillstand der Bewegung angenommen wird. Dafür wird der initiale Schwellwert auf 1% der Maximalgeschwindigkeit gesetzt. Ausgehend vom Punkt der höchsten Geschwindigkeit kann anschließend der nächste Punkt vor und nach diesem Zeitpunkt ermittelt werden, zu dem die nach dem Schwellwert als Stillstand angenommene Bewegung auftritt. Damit lässt sich der Datensatz soweit beschneiden, dass der erste bzw. der letzte Punkt des Datensatzes den Beginn bzw. das Ende des Greifvorgangs darstellt.

Statische Filterung Diese Phase filtert fehlerhafte Messdaten basierend auf den Kriterien der Länge der Messreihe, Anzahl an NaN-Werten je Messreihe und der Maximalgeschwindigkeit je Messreihe heraus.

Die obere und untere Schranke der Messreihenlänge ergibt sich aus Beobachtungen durchschnittlicher Längen korrekt gemessener Messreihen. Alle Messreihen, die nach der vorangegangenen Interpolation noch NaN-Werte enthalten, besitzen nicht korrigierbare Messfehler und werden aussortiert. Als Maximalgeschwindigkeit für die Bewegung eines menschlichen Handgelenks kann realistisch 5m/s [E114] festgelegt werden.

Spiegelung und Translation In einem Teil der vorliegenden Datensätze sind für Greifvorgänge mit beiden Händen die Bewegungspunkte für beide Körperhälften vorhanden, in vielen Fällen jedoch nur für eine. Zur Vereinheitlichung und Vergrößerung der Datenmenge sollen alle Datensätze durch eine Spiegelung auf den Greifvorgang der rechten Körperhälfte abgebildet werden.

Für die Umsetzung der Spiegelung wird folgende Vorgehensweise vorgeschlagen. Liegen für einen Greifvorgang Datensätze zu beiden Körperhälften vor, soll eine Spiegelebene anhand dieser beiden Datensätze ausgerichtet werden. Ist nur eine Körperhälfte vorhanden, wird

die Spiegelung an einer Ebene durchgeführt, die durch Start- und Endpunkt der Bewegung verläuft und senkrecht zur größten Auslenkung der Trajektorie steht.

Nach der Spiegelung erfolgt eine Verschiebung aller Trajektorien, sodass sie jeweils im Ursprung starten.

Metrikbasierte Filterung Die weitere Selektion qualitativ hochwertiger Messdaten lässt sich mit Hilfe von drei Metriken und dynamisch gebildeten Grenzenwerten durchführen.

1. Handgelenk-Ellenbogen-Distanz: Arithmetisches Mittel der euklidischen Distanzen zwischen Ellenbogen und Handgelenk zu allen Zeitpunkten n , sollte annähernd konstant sein.
2. Handgelenk-Handgelenk-Distanz: Arithmetisches Mittel der euklidischen Distanzen beider Handgelenke zu allen Zeitpunkten n , sollte aufgrund der vorhergehenden Spiegelung der Trajektorien der linken Körperhälfte möglichst gering sein.
3. Vergleich der Z-Koordinaten: Arithmetisches Mittel der absoluten Differenzen der z-Koordinaten der Handbewegungen beider Körperseiten zu allen Zeitpunkten n , sollte nahezu Null sein.

Die Schwellwerte der Metriken, ab welchen ein Messdatensatz als fehlerbehaftet angesehen wird, müssen aufgrund der anatomischen Unterschiede der Testpersonen jeweils anhand der jeweiligen Messdaten ermittelt werden. Der untere und obere Schwellwert κ_m je Metrik m ergibt sich aus dem Quantil der jeweiligen Metrikwertverteilung, wobei der Unterschreitungsanteil des Quantils abhängig von der Standardabweichung s_m und einer Skalierungskonstante δ_m gewählt wird. Hierdurch ist sichergestellt, dass stark verrauschte Daten stärker aussortiert werden. Für die drei vorgestellten Metriken lassen sich folgende Skalierungskonstanten verwenden: $\delta_1 = 0.009$, $\delta_2 = 0.005$, $\delta_3 = 0.008$

$$\kappa_m^{unten} = Q(\delta_m s_m) \quad \text{mit } \delta_m s_m \in [0, 0.5] \quad (2)$$

$$\kappa_m^{oben} = Q(1 - \delta_m s_m) \quad \text{mit } \delta_m s_m \in [0, 0.5] \quad (3)$$

Nach der Berechnung der Schwellwerte je Metrik und Messreihe erfolgt in einem mehrstufigen Verfahren die Entscheidung über die Aussortierung eines Datensatzes. Hierzu wird bei auffälligen Metrikwerten außerhalb der Schwellwerte der normierte Abstand zum Mittelwert der jeweiligen Verteilung als Ausreißerintensität ρ_m herangezogen. Das Aussortierungsverfahren bildet anschließend wie in Algorithmus 1 eine gewichtete Summe der Ausreißerintensitäten je Metrik und vergleicht diese mit einer Fehlerschranke.

Die Wahl der Gewichte spiegelt die Relevanz der einzelnen Metriken wider. Höchste Relevanz haben die Ergebnisse der Handgelenk-Handgelenk-Metrik, da Ausreißer in dieser Metrik klar auf Fehler in den für das maschinelle Lernen relevanten Handgelenkdaten zurückzuführen sind.

für alle translatierten Dateien **tue**

```

     $\epsilon \leftarrow 0$ ;
    wenn Datei in Handgelenk-Ellenbogen-Metrik als Ausreißer markiert dann
        |  $\epsilon \leftarrow \epsilon + 2 \cdot (1 + \text{Ausreißerstärke } \rho_1 \text{ der Datei})$ ;
    Ende
    wenn Datei in Handgelenk-Handgelenk-Metrik als Ausreißer markiert dann
        |  $\epsilon \leftarrow \epsilon + 4 \cdot (1 + \text{Ausreißerstärke } \rho_2 \text{ der Datei})$ ;
    Ende
    wenn Datei in Z-Differenz-Metrik als Ausreißer markiert dann
        |  $\epsilon \leftarrow \epsilon + 2 \cdot (1 + \text{Ausreißerstärke } \rho_3 \text{ der Datei})$ ;
    Ende
    wenn  $\epsilon > 4$  dann
        | Sortiere Datei aus;
    Ende

```

Ende

Algorithmus 1: Gewichtete Aussortierung fehlerhafter Datensätze

Rotation Diese Phase rotiert die Handgelenkstrajektorien, sodass die Anfangsverläufe der Bewegungen einheitlich in x-Achsenrichtung zeigen. Der Anfangsverlauf ist hierbei die feste Anzahl an Datenpunkten, die als Eingang für das zu trainierende Neuronale Netz gilt. Der letzte Datenpunkt des Anfangsverlaufes lässt sich als Rotationspunkt bezeichnen, dessen Ortsvektor die Richtung des Anfangsverlaufs darstellt. Für die Rotation gilt der Drehwinkel ψ zwischen der x-Achse und dem Ortsvektor \vec{d} des Rotationspunktes. Die Rotationsachse stellt die Normale der von der x-Achse und \vec{d} aufgespannten Ebene dar. Die rotierten Daten ergeben sich durch Multiplikation mit einer entsprechenden Rotationsmatrix.

3.2 Vorhersage

Die Vorhersage der zur Pfadberechnung relevanter Angaben der Endposition und Dauer der Handgelenkbewegung erfolgt über zwei separate Neuronale Netze. Als Eingangsmerkmale lassen sich die drei Koordinaten der Zeitpunkte der zuvor in der Rotation definierten Anfangsbewegung verwenden. Diese hat im Beispiel eine Länge von 85 Zeitpunkten, d.h. es ergeben sich 255 Eingangsmerkmale.

Für die Vorhersage der Endposition hat sich empirisch ein Neuronales Netz mit Dense-Layern als sinnvoll erwiesen, siehe dazu Abbildung 2. In den verborgenen Schichten verwendet das Netz als Aktivierungsfunktion Rectified Linear Units (ReLU) und in der Ausgabeschicht eine lineare Aktivierungsfunktion. Die Fragezeichen in der Abbildung stehen für eine beliebige Anzahl an Datensätzen. Vor dem Training erfolgt eine Skalierung der Daten in den einheitlichen Wertebereich von 0 bis 1. Als Kostenfunktion wird zur Vorhersage der Endpunkte die quadrierte euklidische Distanz zwischen dem vorhergesagten und dem tatsächlichen Endpunkt verwendet. Für die Vorhersage der Dauern erweist sich der Mean Absolute Error als geeignete Kostenfunktion, da keine räumlich interpretierbaren Daten vorliegen.

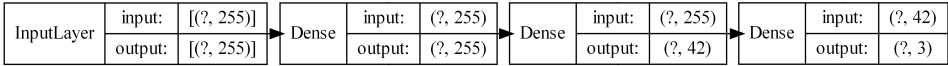


Abb. 2: Architektur des neuronalen Netzes zur Vorhersage der Endpositionen

Das Training des Neuronalen Netzes geschieht über Datensätze, die nach der in der vorliegenden Arbeit ausgeführten Methode vorverarbeitet sind. Die Datensätze bestehen aus Greifbewegungen von verschiedenen Probanden, die nach dem in Abschnitt 2.1 genannten Szenario aufgenommen sind.

3.3 Pfadberechnung

Mittels der in diesem Kapitel prognostizierbaren Endpunkte und Bewegungsdauern wird mit dem Minimum Jerk Modell für gekrümmte Bewegungen der Verlauf der Trajektorie anhand der aufgenommenen Anfangsbewegung vorausberechnet. Notwendig hierfür ist neben der Information des Start- und des Endpunkts P_1 und P_2 der Bewegung sowie der Dauer des Greifvorgangs auch die Angabe eines dritten Punkts P_C , durch den die Bahnkurve verlaufen soll. In der idealen Modellierung der realen Trajektorie mit diesem Verfahren ist dies der Punkt der größten Entfernung von der Strecke zwischen Start- und Endpunkt. Da dieser bei der Betrachtung der Anfangsbewegung noch nicht aufgetreten sein muss, soll die Annahme getroffen werden, dass derjenige Punkt der Anfangsbewegung mit dem größten Abstand von der Strecke zwischen Startpunkt und dem durch das ML-Modell prognostizierten Endpunkt diesen dritten Punkt P_C darstellt. Das Vorgehen zur Bestimmung der gekrümmten Trajektorie sieht vor, eine Berechnung der Bahnkurve zeitlich vor und nach dem Punkt P_C durchzuführen.

4 Ergebnisse

Als bestmögliche Konfiguration erweist sich eine Anfangsbewegung der Länge 85, somit werden durchschnittlich 65% der Trajektorie als Eingangsdaten zur Vorhersage der Endposition benötigt. Der Rotationspunkt, nach dem sich die Rotation ausrichtet, liegt dabei beim 70. Punkt der Trajektorie. Als Epochenanzahl des Trainings stellt sich 350 als sinnvolle Größe heraus, da bei dieser Epochenanzahl das Neuronale Netz bei Testdaten ähnlich gute Ergebnisse wie bei Trainingsdaten erzielt und noch keine starke Überanpassung auf die Trainingsdaten vorliegt. Als Batch-Size lässt sich 250 festlegen, von den Trainingsdaten werden 25% als Testdaten und 5% als Validierungsdaten verwendet. Bezüglich der Metrikergebnisse können bei bereits beschriebener Trainingskonfiguration die Endpunkte durchschnittlich bei den Trainingsdaten auf 10.7cm und bei den Testdaten auf 13.7cm genau vorhergesagt werden, die Werte ergeben sich als durchschnittliche euklidische Distanz der letztendlich vorhergesagten und tatsächlichen Endpunkte. Bewegungsdauern können auf bis zu zehn Zeitpunkte genau bei Trainings- und Testdaten vorhergesagt werden.

Die Berechnung der Bahnkurve durch das Minimum Jerk Model für gekrümmte Punkt-zu-Punkt Bewegungen anhand der Anfangsbewegung und dem vorhergesagten Endpunkt und Dauer ergibt einen ähnlichen Verlauf im Vergleich zur tatsächlichen Trajektorie. In Abbildung 3 ist eine beispielhafte Gegenüberstellung zu sehen. Abgebildet ist zum einen der Verlauf der im Versuch aufgenommenen Trajektorie in blau und zum anderen die anhand dieser Daten und der Vorhersagen des Neuronalen Netzes berechneten Bahnkurve in orange.

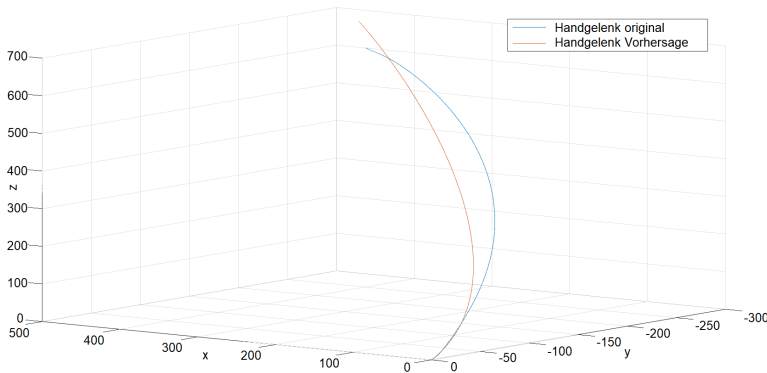


Abb. 3: Vergleich der tatsächlichen (blau) zur berechneten Trajektorie (orange)

5 Diskussion

Datenvorverarbeitung In Bezug auf die Datenvorverarbeitung lässt sich in der Phase der Beschneidung kritisieren, dass sich die Bestimmung der Qualität der Beschneidung lediglich auf empirisch bestimmte Werte begründet. Eine Aussage über die tatsächliche Güte der Datensätze nach der Beschneidung findet nur durch eine manuelle Analyse der Geschwindigkeitsverläufe und der Trajektorien statt. Diese entspricht allerdings nicht dem Gedanken der vollständigen Automatisierung der Datenvorverarbeitung und ist mit erheblichem Aufwand verbunden.

Durch stichprobenartig visualisierte Datensätze fällt auf, dass sich fehlerbehaftete Daten vor allem durch Sprünge in den Geschwindigkeitsprofilen, harte Kanten oder sonstige ungewöhnliche Form auszeichnen. Die Filterung deckt diese Kriterien bisher nicht vollständig ab, beispielsweise wäre die Ergänzung um eine Stetigkeitsanalyse der Geschwindigkeitsverläufe sinnvoll.

Machine Learning Es lässt sich der generelle Erfolg des Maschinellen Lernens diskutieren, da fast zwei Drittel der Trajektorie als Eingangsdaten benötigt werden, um den Endpunkt und die Dauer vorherzusagen. Zudem ist eine Vorhersage der Dauer mit einer Präzision von 10 Zeitpunkten unter der Betrachtung, dass aufgrund der statischen Filterung noch

maximal 90 Zeitpunkte, durchschnittlich jedoch lediglich 45 weitere Messpunkte folgen, relativ unpräzise. Da der Fokus dieser Arbeit allerdings auf der Datenvorverarbeitung liegt, ist dieses Ergebnis akzeptabel. Es lassen sich trotz verbesserungsfähiger Präzision gute Ergebnisse in der Berechnung der weiteren Trajektorie erzielen. Zudem gilt zu beachten, dass in der Praxis zur Kollisionsvermeidung ohnehin ein Sicherheitsbereich um die berechnete Trajektorie gebildet werden muss. Dadurch lassen sich Ungenauigkeiten in der Berechnung etwas ausgleichen.

Praxistauglichkeit Für die Anwendung in der Praxis ist eine Vorverarbeitung der Daten der Anfangstrajektorie erforderlich. Durchschnittlich folgen auf den 85. Messwert noch ca. 45 weitere, bis die Bewegung die Endposition erreicht. Bei einem verwendeten Kamerasystem mit 120Hz bleiben zur Datenverarbeitung und Endpunktvorhersage lediglich 375ms Zeit. Deshalb müssen die Phasen der Datenvorverarbeitung priorisiert werden. Höchste Priorität haben dabei Phasen, die der Vereinheitlichung der Daten dienen, wie die Translation auf den Ursprung und Rotation der Anfangsrichtung in x-Achsenrichtung, da ohne diese Phasen die gemessenen Eingangsdaten nicht mit den Trainingsdaten vergleichbar sind. In Priorität zwei sind Verarbeitungsschritte, die lediglich die Datenqualität steigern. Darunter fallen beispielsweise Interpolation, Glättung und Filterung der Daten.

In Tests können die beschriebenen Verarbeitungsschritte der höchsten Priorität, die Vorhersage des Endpunkts und der Dauer sowie anschließende Retransformation in durchschnittlich 52ms durchgeführt werden. Die Berechnung der vorherzusagenden Bahnkurve der Hand über das Minimum Jerk Model nimmt in Tests etwa 250ms in Anspruch. Über eine numerische Optimierung der Gleichung und der Berechnung sowie Approximationen, wie beispielsweise eine Betrachtung jedes zehnten Frames, ließe sich die Berechnungszeit auf schätzungsweise ein Zehntel davon minimieren. Die gesamte Vorausberechnung der Trajektorie der Hand liegt damit ungefähr bei 80ms und ist somit praxistauglich.

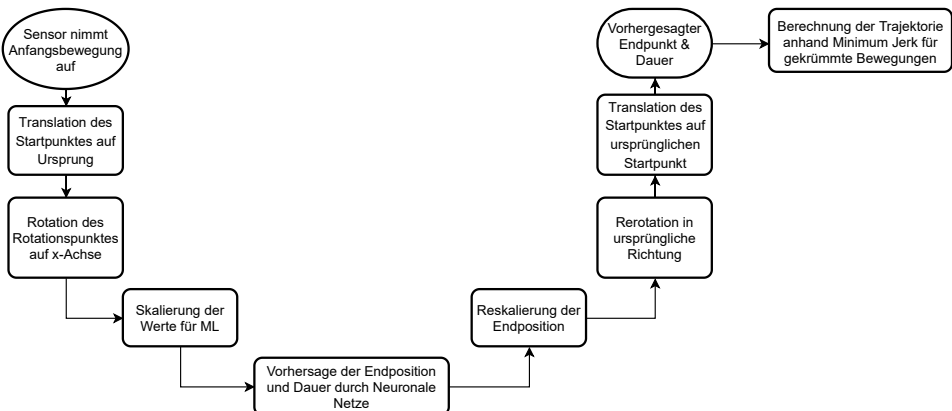


Abb. 4: Ablauf zur Anwendung in der Praxis

Es lässt sich der generelle Ansatz der Kollisionsvermeidung durch Vorhersage der Bewegungsabläufe des Menschen und anschließenden Korrektur der Roboterbewegung kritisch betrachten. Der Ansatz basiert auf der Annahme, dass der Mensch seine Handbewegung unabhängig von Einflüssen der Umwelt deterministisch durchführt. Menschen reagieren allerdings auf äußere Veränderungen, wie beispielsweise eine Routenanpassung oder Pausierung der Roboterbewegung, und können daraufhin ihre Bewegung unmittelbar auch ändern, sodass sie nicht mehr der vorausberechneten Trajektorie entspricht.

Dieser Rückkopplungseffekt muss bei weiteren Überlegungen und dem praktischen Einsatz des Verfahrens bedacht werden.

6 Fazit und Ausblick

Zusammenfassend zeigt die vorliegende Arbeit ein Gesamtkonzept zur Vorausberechnung von Handbewegungen auf. Hierbei deckt die Arbeit von der Datenaufnahme, über die Datenvorverarbeitung und dem letztendlichen Training eines Neuronalen Netzes zur Vorhersage von Endpunkt und Dauer von Handbewegungen alle wichtigen Phasen des Maschinellen Lernens ab. Dieser Prozess wird zudem durch die anschließende mathematische Berechnung der Trajektorie in das praktische Anwendungsszenario der Kollisionsvermeidung in der kollaborativen Robotik eingebettet.

Mögliche Erweiterungen und Fortführungen bestehen zum einen in einer weiteren Verbesserung der Datenvorverarbeitung durch beispielsweise eine Erweiterung der Filterung um die Glattheit der Kurven. Grundsätzlich bietet sich auch Maschinelles Lernen als Unterstützung zur Aussortierung von fehlerbehafteten Rohdaten an. Anhand der durch die Pipeline erzeugten gelabelten Daten ließe sich in einem überwachten Lernverfahren zu diesem Zweck ein Klassifikator trainieren.

Zum anderen kann über eine Diskretisierung des Raums in Würfel mit Kantenlänge von 10cm und diskrete Vorhersage der Endposition nachgedacht werden. Aufgrund des erforderlichen Sicherheitsabstandes ist für die Praxis eine exakte kontinuierliche vorhergesagte Position nicht erforderlich.

Auch die Verwendung von Recurrent Neural Networks oder Convolutional Neural Networks lässt sich als Alternative zu den verwendeten Dense-Layern zur Vorhersage der Endpositionen heranziehen.

Für die Anwendbarkeit in der Praxis ist eine Umsetzung ohne markerbasierte Handgelenkserkennung durch ein Kamerasystem und Software wie *OpenPose*, einem Open-Source-Echtzeitsystem zur 2D-Positionserkennung von anatomischen Schlüsselpunkten wie beispielsweise Handgelenken denkbar.[Ca21] Die 2D-Positionsdaten müssten beim aktuell trainierten Modell allerdings durch mehrere Kameras in 3D-Positionsdaten umgerechnet werden.

Darüber hinaus kann die Möglichkeit der Aufnahme von Handbewegungen über eine Kinect-Kamera wie von Elgendi et al. [El14] beschrieben für eine bessere Praxistauglichkeit weiter verfolgt werden.

Literaturverzeichnis

- [AKG04] Admiraal, Marjan A.; Kusters, Martijn J. M. A. M.; Gielen, Stan C. A. M.: Modeling kinematics and dynamics of human arm movements. 8(3):312–338, 2004. Journal Article.
- [Ca21] Cao, Zhe; Hidalgo, Gines; Simon, Tomas; Wei, Shih-En; Sheikh, Yaser: OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. 43(1):172–186, 2021. Journal Article.
- [Di11] Ding, Hao; Reissig, Gunther; Wijaya, Kurniawan; Bortot, Dino; Bengler, Klaus; Stursberg, Olaf: Human arm motion modeling and long-term prediction for safe and efficient Human-Robot-Interaction. In: 2011 IEEE International Conference on Robotics and Automation (ICRA). IEEE / Institute of Electrical and Electronics Engineers Incorporated, S. 5875–5880, 2011.
- [Ed17] Edvarsson, Andreas: Online Predictions of Human Motion. 2017.
- [El14] Elgendi, Mohamed; Picon, Flavien; Magnenat-Thalmann, Nadia; Abbott, Derek: Arm movement speed assessment via a Kinect camera: a preliminary study in healthy subjects. 13:88, 2014. Journal Article Research Support, Non-U.S. Gov't.
- [FH85] Flash, Tamar; Hogan, Neville: The coordination of arm movements: an experimentally confirmed mathematical model. 5(7):1688–1703, 1985.
- [FMv12] Fligge, Nadine; McIntyre, Joseph; van der Smagt, Patrick: Minimum jerk for human catching movements in 3D. In (Author, Ieee Corporate, Hrsg.): 2012 4th IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics. IEEE, S. 581–586, 6/24/2012 - 6/27/2012.
- [Gr50] Grubbs, Frank E.: Sample Criteria for Testing Outlying Observations. 21(1):27–58, 1950.
- [Ke20] Keibel, Andreas: Mensch-Roboter-Kollaboration in der Medizin. In (Buxbaum, Hans-Jürgen, Hrsg.): Mensch-Roboter-Kollaboration, S. 133–143. Springer Fachmedien Wiesbaden, 2020.
- [Ma16] Markis, Alexandra; Montenegro, Harald; Neuhold, Michael; Oberweger, Andreas; Schlosser, Christian; Schwald, Christoph; Sihn, Wilfried; Ranz, Fabian; Edtmayr, Thomas; Hold, Philipp et al.: Sicherheit in der Mensch-Roboter-Kollaboration. 2016.
- [MK99] Messier, Julie; Kalaska, John F.: Comparison of variability of initial kinematics and endpoints of reaching movements. 125(2):139–152, 1999. Comparative Study Journal Article Comparative Study Journal Article.
- [Mu99] Murray, Ingram Andrew: Determining upper limb kinematics and dynamics during everyday tasks. 1999.
- [Re21] Rettig, Oliver; Müller, Silvan; Strand, Marcus: Determination of posture comfort zones for robot-human handover tasks. 2021.
- [Zh21] Zhao, Jing; Gong, Shiqiu; Xie, Biyun; Duan, Yaxing; Zhang, Ziqiang: Human arm motion prediction in human-robot interaction based on a modified minimum jerk model. 35(3-4):205–218, 2021.

Short Papers

TD-Browser – A Beginner-friendly Web-Client for the Web of Things

Osama Hanoun¹

Abstract: In this paper, the author introduce the TD-Browser that enables visual interactions with Web Things by generating user interface elements from Thing Descriptions. The target group are beginners who did not yet understand the concept of the Web of Things. Currently, most available tools focus on scientific purposes with a lack of documentation which makes it hard for newcomers to gain practical experience. For evaluation, I conducted a study using the Concurrent Think-aloud method with one subject to uncover first design flaws. Although TD-Browser cannot be used for teaching beginners the concepts of the Web of Things for now, its clean user interface enables users to work and interact effortlessly with Web Things.

Keywords: Prototyping; Semantic Web; Thing Description; Web of Things

1 Introduction

The Web of Things (WoT) is a subset of the Internet of Things (IoT) which aims to provide universal protocol interoperability by leveraging standard Web protocols such as the HyperText Transfer Protocol (HTTP). The W3C WoT Task Force defines three main components for the WoT [Ka20]: (1) A *Thing* (or a Web Thing) which is an abstraction of a physical or a virtual entity, (2) a *Thing Description* (TD) which is both a machine- and human-readable way to describe the metadata and the interface of a Thing, and (3) a so-called *Consumer* which is an entity that can process WoT Thing Descriptions.

Although the first public Working Draft for the WoT-TD was published in 2017², there is still a scarcity of beginner-friendly tools for the Web of Things (WoT) that can be used for prototyping and testing. In order for users to be able to interact with a Thing, it is required from them to develop a Consumer that can process the TD for the targeted Thing. Developing a Consumer requires the comprehension of the WoT architecture³ and proficient programming skills. Therefore, it is challenging for students and scientists who are newcomers, I will describe them as beginners through this paper, to the WoT architecture, whenever they would like to try interacting with published Things or to test a Thing that they have created. As a solution, I introduce TD-Browser, a Web-client and WoT Consumer

¹ Friedrich-Alexander-Universität Erlangen-Nürnberg, Lange Gasse 20, 90403, Nuremberg, Germany, osama.hanoun@fau.de

² <https://www.w3.org/TR/2017/WD-wot-thing-description-20170914/>, last accessed on April 25th, 2022.

³ <https://www.w3.org/TR/wot-architecture/>, last accessed on April 25th, 2022.

which can generate a graphical user interface (GUI) on the basis of a TD. This allows users to start interacting with a Thing graphically with reduced effort by overcoming most technical barriers and minimizing the required knowledge of the WoT Scripting API⁴ implementation.

2 Related Work

The WoT architecture defines three interaction affordances [Ka20]: (1) Properties, (2) Actions, and (3) Events. Properties expose the state of a Thing, for example, the model name of a device or the current data of a sensor. Actions are used to invoke a function of a Thing, for example, asking a printer to print a document. Events are asynchronous data sent by a thing to all consumers, for example, a Smart Radiator alerting all Consumers when its heat increases.

There are related projects, as mentioned in the developers page on the W3C Web of Things website⁵, such as Web of Things API Development Environment (W-ADE)⁶, WoT-FXUI⁷, and Browsified node-wot⁸. However, all of the mentioned projects do not support the latest WoT standard.

W-ADE [Sc20] provides many features including TDs parsing, interpreting and editing, communicating over various protocols (e.g., MQTT), adding security credentials and providing timing performance testing. Browsified node-wot allows users to read properties, invoke actions, and subscribe to events. But, it has a poorly structured user interface and does not support all the affordance interaction like observing a property and reading all properties. In contrast to the previously mentioned projects, TD-Browser covers all the interaction affordances and provides a beginner friendly user interface as will be discussed in the evaluation section.

3 Design and Implementation

TD-Browser⁹ addresses users with limited knowledge of the WoT, where having a clear user interface plays a significant role especially when it is the user's first interaction with the WoT. Following Don Norman's principles of design (see Table 1), TD-Browser ensures *visibility* of all the features by using clear descriptive buttons that correspond to the WoT Scripting API. Users receive an instant *feedback* for any action that they perform. For example, when TD-Browser consumes a TD, the user will be informed by either a green success message or a red error message. TD-Browser puts *constraints* on users by preventing accessing

⁴ <https://www.w3.org/TR/wot-scripting-api/>, last accessed on April 25th, 2022.

⁵ <https://www.w3.org/WoT/developers/>, last accessed on April 25th, 2022.

⁶ <https://github.com/tum-esi/wade/>, last accessed on April 25th, 2022.

⁷ <https://github.com/danielpeintner/wot-fxui/>, last accessed on April 25th, 2022.

⁸ <http://plugfest.thingweb.io/webui/>, last accessed on April 25th, 2022.

⁹ <https://github.com/wintechis/TD-Browser/>

more than one interaction affordance at a given time. Having representative icons allows TD-Browser to provide an excellent *mapping* between icons and their linked effects. If a user wants to invoke an action or maybe write a property, then the user will take *consistent* steps in order to achieve the goal. In TD-Browser, users will experience consistent color change of buttons when interacting with them. Figure 1 shows a screenshot of TD-Browser.

Tab. 1: The Don Norman principles of design which used for TD-Browser [No02].

Principle	Meaning
Visibility	Ensuring that users are able to find the features that they are looking for and to know what to do next.
Feedback	Providing users with feedback for what action has been invoked and what has been accomplished.
Constraints	Restricting user interactions with the product at a given moment.
Mapping	Implementing design with a clear relationship between controls and the effect they have.
Consistency	Similar tasks should have similar operations and similar elements.
Affordance	An attribute of an object that should give the user a hint on how to use it.

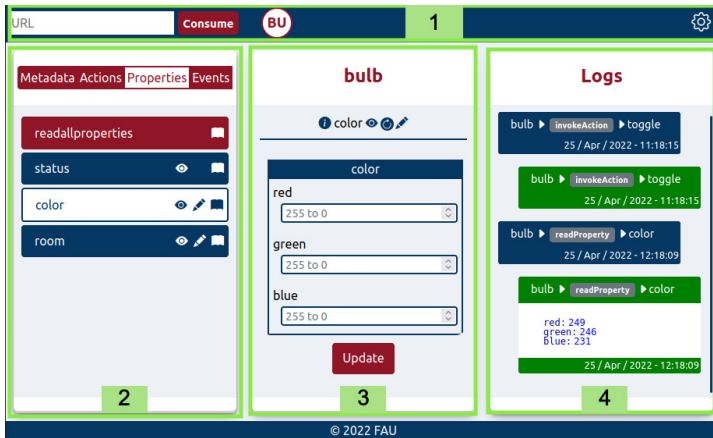


Fig. 1: Screenshot of TD-Browser on desktop showing the main interface components: (1) The Navbar which contains a form for consuming TDs, switching between consumed Things, and changing settings. (2) The Left-frame which contains the metadata, the affordances, and the interaction affordances for the active consumed Thing. (3) The Middle-frame where the user can interact with selected interaction affordance and view the responses from a Thing. (4) The Right-frame where the user can see all requests and responses that happened during a session.

TD-Browser runs on browsers, on both smartphones and desktops, by using web development languages like Hypertext Markup Language revision 5 (HTML5), Cascading Style Sheets level 3 (CSS3), and JavaScript. Also, TD-Browser uses the Eclipse Thingweb node-wot package¹⁰ which is a W3C Web of Things implementation with NodeJS. TD-Browser can

¹⁰ <https://github.com/eclipse/thingweb.node-wot/>, last accessed on April 25th, 2022

be used with no internet connection if a user has the source code and decides to only interact through a local network.

The following scenario will be used to explain the TD-Browser architecture and how it works. A user provides TD-Browser with a TD of a Thing then TD-Browser generates a graphical user interface that corresponds to the provided TD. When the user interacts with an interaction affordance, TD-Browser will convert any entered data and the performed interaction to an acceptable syntax by Eclipse Thingweb node-wot package. After that, the package sends a request to the Thing and when the package receives a response then it will be provided to TD-Browser. In the end, the TD-Browser displays the response to the user.

TD-Browser provides the following main features: (1) Supporting HTTP protocol. (2) Consuming a local or web-based JSON/JSON-LD file. (3) Consuming multiple TDs at the same time. (4) Reading, observing, and writing properties. (5) Invoking actions. (6) Subscribing and unsubscribing to events. (7) Validating inputs according to the consumed TD. (8) Accessible on the browser by both desktop and smartphone with a responsive view.

4 Evaluation

TD-Browser targets beginner users of the WoT with a beginner-friendly interface. In order to test its eligibility for beginners, an evaluation is required. In addition, the evaluation is important to uncover any design flaws and relevant usability elements.

In this evaluation I used the Concurrent Think-aloud method (CTA) [MH12] which is an evaluation method where subjects verbalize their thinking, doing, or feeling while completing a set of predefined tasks. According to an international survey done by McDonald, Edwards, and Zhao [MEZ12] showed that CTA is the most frequently used approach. Also, CTA is easy and fast to conduct.

I recruited one female computer engineering student from Jordan (22 years old), by word of mouth, who did not have prior knowledge of the WoT architecture and was not associated with my research work. Also, she did not receive any compensation and was just participating out of interest. She was told that TD-Browser allows users to control devices connected to the internet by providing the device URI (Uniform Resource Identifier) and she was given a set of tasks as a scenario to complete that covered virtual Thing features. The virtual Thing used in evaluation was a virtual light bulb with interaction affordances which is listed in Table 2. In addition, I provided her with a Web page that simulates the behavior of the virtual light bulb (see Figure 2). The total evaluation lasted one hour.

The subject was given a scenario and a set of tasks as listed in Table 3. The scenario was that the subject gave her grandmother a new smart light bulb as a gift. Then before she goes back home, she connects the light bulb for her grandmother to the Wi-Fi. After she arrived home, her grandmother calls her back in order to help her grandmother. Since, the

Tab. 2: Interaction affordances of the virtual light bulb used in the evaluation

Affordance Class	Interaction Affordance	Description
Action	toggle	Toggle the light bulb (on off)
	reset	Reset the light bulb settings
Property	status	Current status of the light bulb (on off)
	color	Current RGB color of the light bulb
	room	Current Location of the light bulb
Event	power_on	Fire the event when the power is back on

light bulb can be controlled by TD-Browser from her home, therefore, she shall help her grandmother by completing the tasks.

The subject was able to complete all the tasks without any problems. As I observed during the evaluation, the subject would solve a task by looking for mutual words between a task and the TD-Browser interface. In case no mutual words were found, she would hover over the icons in the TD-Browser interface and read their tooltip. The approach that the subject used to complete the tasks relies on the clarity of the TD-browser interface and its self-descriptive buttons and icons. Regarding tasks 2, 9, 13 and 14 in Table 3, even though the subject did not face any problems in completing these tasks, she had trouble to understand the phrased tasks with the WoT terminology. In summary, she could interact with WoT Things without understanding the WoT architecture. Therefore, I conclude that TD-Browser can be valuable to users who recently have started with the WoT, but TD-Browser does not teach users the concept of the WoT architecture. Future work should focus on developing a didactic concept to teach the WoT architecture to beginners while interacting with TD-Browser.

5 Conclusion

In order for more people to adhere to the W3C standards of WoT, more development tools and packages are needed to encourage them to adapt these standards. Users who are new to the Web of Things environment struggle to get started with available tools due to their lacking documentation. To support beginners, I introduced the TD-Browser which enables visual interactions with Web Things. I conducted a study using the Concurrent Think-aloud method with one subject to uncover first design flaws. Although TD-Browser cannot be used for teaching beginners the concepts of the Web of Things, its clean user interface enables users to work and interact with Web Things nonetheless. Future work aims to close the gap between applying the WoT and understanding it.

Tab. 3: The tasks which were used for the evaluation

#	Task
1	Provide the app with the URI that was given to you in order to gain control over the light bulb.
2	State the type of security that is used for the light bulb.
3	State the available actions.
4	Show the description of the toggle action.
5	Invoke the toggle action.
6	State the available properties.
7	Read the status property.
8	Change the color of the light bulb to yellow (R:255, G:255,B:0).
9	Observe the status property.
10	Change the location of the bulb from the room property to "bedroom".
11	State the available events.
12	Subscribe to the power_on event.
13	Unsubscribe to the power_on event.



Fig. 2: A screenshot of the virtual light bulb used in the evaluation

6 Acknowledgements

I would like to thank the Chair of Technical Information Systems at Friedrich-Alexander-Universität Erlangen-Nürnberg. Without the support of the Chair this paper and this project would not have been possible.

Bibliography

- [Ka20] Kajimoto, Kazuo; Lagally, Michael; Kawaguchi, Toru; Matsukura, Ryuichi; Toumura, Kunihiro: Web of Things (WoT) Architecture 1.1. W3C working draft, W3C, November 2020. <https://www.w3.org/TR/2020/WD-wot-architecture11-20201124/>.
- [MEZ12] McDonald, Sharon; Edwards, Helen; Zhao, Tingting: Exploring Think-Alouds in Usability Testing: An International Survey. IEEE Trans. Prof. Communication, 55:2–19, 03 2012.
- [MH12] Martin, Bella; Hanington, Bruce: Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions. Rockport Publishers, hardcover edition, 2 2012.
- [No02] Norman, Donald A.: The design of everyday things. Basic Books, [New York], 2002.
- [Sc20] Schlott, Verena Eileen; Korkan, Ege; Kaebisch, Sebastian; Steinhorst, Sebastian: W-ADE: Timing Performance Benchmarking in Web of Things. In (Bielikova, Maria; Mikkonen, Tommi; Pautasso, Cesare, eds): Web Engineering. Springer International Publishing, Cham, pp. 70–86, 2020.

Identifying Alternatives and Deciding Factors for a Data Mesh Architecture

Clara Voß¹

Abstract: The data mesh was introduced in 2019 as a new type of data architecture. It promises a more democratic and scalable way of data production and consumption, while also solving current data engineering problems. These problems consist of siloed and hyper-specialized data engineering knowledge, a growing number of dependencies within data pipelines, and the rigidity of centralized monoliths. This paper uses expert interviews to identify the most significant current alternatives to the data mesh and abstract factors, with which companies can evaluate whether a data mesh can advance their move towards a data-driven, democratized future. The results show that the company's motivation, culture, structure, as well as IT history and IT structure should be evaluated before implementing a data mesh. This paper is based on a bachelor thesis.

Keywords: data mesh, data architectures, data integration, expert interviews

1 Introduction

In a world where the importance of data and its analysis continuously grows, choosing a fitting data architecture is crucial to the success of a company's data strategy. While many data architectures have been proposed since the start of enterprise data solutions in the 1970s, a lot of companies still struggle to find suitable solutions to their specific problems and to overcome challenges like the need for a growing number of increasingly specialized data engineers and a growing burden of dependencies.

In an attempt to solve these common architecture failure modes, Zhamak Dehghani presented the data mesh [De19]. This is a data architecture that is not focused on a centralized monolithic data architecture but instead provides a more distributed, democratized approach. Supported by a self-serve tooling and data platform, the business is separated into independent, empowered units, which create autonomous data products that are shared in a company-wide data marketplace. While the data mesh has started a big conversation about distribution among the data engineering community, many companies are still hesitant to implement it, because little work has been done to identify advantages and disadvantages. The findings of this paper are based on expert advice and practitioners' experiences and offer use cases, alternatives, and tangible deciding factors to allow for a greater understanding of the data mesh concept.

¹ Hochschule für angewandte Wissenschaften München, Fakultät für Informatik und Mathematik, Lothstr. 34, 80335 München, clara.voss97@gmail.com

2 Theoretical Foundation

2.1 Data Mesh

To define the data mesh, its idea and core concepts as presented in [De19, De20] will be summarized in the following paragraph: Central to the definition of the data mesh are its four main concepts: Domain Ownership, Data as a Product, Self-Serve Infrastructure Platform, and Federated Governance. According to Domain Ownership, a company should be organized into inter-disciplinary teams, that are formed around business functionality, have domain knowledge and are responsible for the full life cycle of its functionality. This pushes responsibility from the central IT or data engineering department to these independent teams. Each team can build data products, the smallest independent measure of data assets. These include the data, the describing metadata, the code leading to its finished product, the documentation of the infrastructure it is hosted on, documentation, and data lineage. Full responsibility for the data product, including the fulfillment of the DATSIS characteristics (discoverable, addressable, trustworthy, self-describing, interoperable, secure), stays with the team that created it. Data products are hosted on a company-wide data marketplace from which other teams can use them for analysis, further development, and creation of new data products (Data as a Product). To enable all teams to build data products without needing specialized technological knowledge, a self-serve infrastructure platform is built. This platform abstracts the complexity of choosing and provisioning technologies and allows for teams to independently choose the technologies that best suit their challenges. Lastly, federated governance is the principle that sets global standards, monitors the mesh, and is responsible for defining the balance between centralized guidelines and distributed responsibility / domain team autonomy.

Aspects of existing trends can be found in the main concepts leading to the data mesh. This shows that the data mesh can be seen as a natural progression, consolidation and application of these trends: Domain Driven Design (organizing teams around functionality), agile development (self-organized and cross-functional teams), cloud computing (abstraction of physical infrastructure and easier infrastructure administration), DevOps (collaboration between technical and business experts and removing traditional separation e.g. by organizational stages), microservices (splitting a monolithic system into small, independent, loosely coupled parts), data democracy (empowering a large, not necessarily technically trained group of people to use and produce data).

2.2 Alternatives

The most important alternative data architectures are the data warehouse, data lake, data lakehouse, data hub, and data fabric, which can be sorted into two main categories:

The data warehouse and data lake specify the way data is stored, the data scheme used and are closely connected to technologies like TeraData or Hadoop. Specialized to work with structured data, the data warehouse stores its data in relational databases and strictly enforces consistency with its scheme-on-write [Wh21]. The growing amount of semi- or unstructured data challenged the data warehouses' focus on structured data and its rigid consistency enforcement. This led to the development of the data lake, which stores data in its raw format, without transformations or enforced consistency [KW18].

In comparison, the data lakehouse, data hub, data mesh and data fabric do not predominately make suggestions about the way data is stored but focus on connecting distinct data sources to offer users a single point of access. The data hub is based on a hub-and-spoke layout and is optimized for sharing data between multiple companies, while conforming to data privacy regulations [Bh15]. Connecting the database management system functionalities and benefits of the data warehouse and the data lake has been the focus of the data lakehouse, which was proposed by Databricks [Ar21].

The data fabric is the closest alternative to the data mesh. Both are not technologies, but emerging data architecture concepts, which identify the same problem of managing data at scale. While the data mesh focusses on distributing the company into independent data domain teams with a federated governance, the data fabric still enforces central data management and control. Additionally, the concept of the data fabric relies on the evolution of artificial intelligence (AI) and knowledge graphs to solve data engineering problems and automatically create inference between data using AI [GB19].

3 Method: Expert Interviews

This paper uses semi-structured, guideline-based expert interviews to extract their experiences with alternatives and transfer the experts' knowledge onto the new topic of the data mesh. The introductory questions centered around the experts' own definition of the data mesh and its alternatives as well as data mesh implementations in past projects. The focus of the interviews was using the ISO 25010 criteria to consider different aspects of the data mesh concept and how they compare to the alternatives. Lastly, the experts were invited to contemplate the future of data architectures and the data mesh concept's place in it.

Planning and execution of the interviews followed the steps described by Mieg and Näf's [MN05], while the interviews' transcripts were analyzed using Mayring's method of qualitative content analysis [Ma15]. The resulting category system built the basis of the analysis and discussion.

The interviewed experts can be separated into three categories based on their background (managerial, technical, and academic), to allow for a broad discussion of the data mesh concept and include different experiences in implementing it. The interviews were conducted in English and German via video calls from 01.11.2021 to 30.11.2021.

Name	Viewpoint	Role	Experience with data integration	Interview length
Roy Kronester	Management	Director Technical Advisory	30 years	1:02:23 min
Dael Williamson	Management	European CTO for Data & AI	25 years	1:05:57 min
-- (anonymous)	Technical, academic	Professor & Hub Lead	11 years	1:01:21 min
-- (anonymous)	Technical	Data Engineering Senior Analyst	4 years	0:29:40 min
Inês Machado	Academic	Author of the first paper on the data mesh	5 years	0:47:48 min
Yvonne Niedling	Technical	Data Engineering Manager	10 years	0:36:30 min

Tab. 1: Overview of the experts

4 Findings

The category system, that resulted from the qualitative content analysis, consolidates the content of the transcribed interviews, and is summarized in the following paragraphs.

4.1 Data architectures in comparison

The data hub, data lakehouse and data fabric are more contemporary data architectures and solve the same use case of connecting distinct data sources. None the less, especially the older data warehouse is mentioned as a suitable alternative to the data mesh. The consolidating nature of the data warehouse allows for representation of all business units within a company and the connections between them. The development of a “data swamp”, a non-functioning data lake that lacks governance and standards, can also be mirrored in a data mesh leading to a “data mess”. Modern architectures like the data lakehouse and data fabric have the added challenge, that much of their vision is based on future advancement of technologies like AI. The level of AI necessary to implement a data fabric has not been reached yet, limiting the evaluation of these data architectures for practical use cases and the implementation itself.

4.2 Factors in the decision for a data mesh

The main factors a company should consider before implementing a data mesh can be grouped into “motivation”, “IT history”, “company culture”, “IT structure”, and “company structure”.

- *Motivation*: It is crucial to evaluate who proposes the implementation of a data mesh and why. Because of the new distribution of responsibility to the business units, they must be fully aware of the effects a data mesh has on their workload. The work of the data engineering and IT departments becomes less stressful, and they can refocus on their intended assignments, creating a big incentive to implement a data mesh. That is why the motivation must be closely analyzed, if the idea of implementing a data mesh is brought forth by the IT department and not fully understood by the business units.
- *IT history*: A data mesh can be a very useful architecture, when the company has conducted many IT and data experiments leading to isolated systems, that otherwise cannot be connected to the main system or whose data cannot be accessed.
- *Company culture*: To support the successful implementation of a data mesh, it is important to make data a priority in the company and allow for flexible problem solving. Additionally, management must be willing to give up responsibilities to the self-governing independent teams and support them in their local decision-making. If this empowerment of the teams cannot be guaranteed by management, then a data mesh should not be implemented.
- *IT structure*: According to the experts’ opinions, a data mesh is especially useful in connecting isolated or legacy systems, clouds or data silos, which have a great amount of friction in data workflows and pipelines.
- *Company structure*: The data architecture should mimic the companies’ own structure: Geographically distributed companies or companies that are comprised of loosely coupled, independent business units can reap a greater benefit, because the domain boundaries can be easily derived. Consolidating these independent business units in a centralized solution would lead to more organizational overhead and problems. Additionally, the company size and amount of data should be considered, as small and mid-sized companies with a small amount of data might be overwhelmed by the organizational changes necessary to implement a data mesh.

5 Conclusion

5.1 Theoretical and Practical Implications

The compiled factors can be used to help companies in deciding whether a data mesh can solve the data engineering challenges and further the usage of data or whether the overhead of implementing a data mesh is too much in comparison to its benefits. If the data mesh is not feasible, the company might consider the mentioned alternatives to find a suitable data architecture solution.

5.2 Limitations and Outlook

Because the conversation about the data mesh has only started in 2019, not many implementation tools have been developed and distributed widely and the topic has not been fully academically discovered, many questions remain unanswered. Some companies report anecdotal successes and benefits due to the implementation of a data mesh, but an academic evaluation is still missing. To conclude, the conversation of moving away from monolithic data architectures might highlight the data usage within companies and bring on a new era of data integration.

6 Acknowledgments

I would like to thank the supervisor of my bachelor thesis, Prof. Dr. Johannes Ebke, for his continuous feedback and the encouragement to take part in this program. Additionally, I want to give thanks to the interview partners, who shared their experiences and invested time. Lastly, I would like to thank Avanade, the company that initially proposed the topic of the data mesh and supported my work along the way.

7 References

- [Ar21] Armbrust, G. et al: Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. In: 11th Annual Conference on Innovative Data Systems Research (CIDR '21), 2021.
- [Bh15] Bhardwaj, K. et al: Collaborative Data Analysis with DataHub. In (VLDB Endowment): Proceedings of the VLDB Endowment, Vol. 8, No. 12, S. 1916-1919, 2015.
- [De19] Dehghani, Z.: How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh, <https://martinfowler.com/articles/data-monolith-to-mesh.html>, Stand: 09.05.2022.
- [De20] Dehghani, Z.: Data Mesh Principles and Logical Architecture, <https://martinfowler.com/articles/data-mesh-principles.html>, Stand: 09.05.2022.

- [GB19] Ghiran, A., Buchmann, R. A.: A Model-Driven Enterprise Data Fabric: A Proposal Based on Conceptual Modelling and Knowledge Graphs. In (Springer, Cham): Knowledge Science, Engineering and Management. KSEM 2019. Lecture Notes In Computer Science, vol. 11775, S. 572-583, 2019.
- [KW18] Khine, P. P., Wang, Z. S.: Data Lake: A new ideology in big data era. In: ITM Web Conferences, vol. 17, 030525, 2018.
- [Ma15] Mayring, P.: Qualitative Inhaltsanalyse. Grundlagen und Techniken, Beltz, Weinheim, 2015.
- [MN05] Mieg, H. A, Näf, M.: Experteninterviews, 2. Aufl., Institut für Mensch-Umwelt-Systeme (HES), ETH Zürich, 2005.
- [Wh21] Oracle (Hg.): What is a Data Warehouse?, <https://www.oracle.com/database/what-is-a-data-warehouse/>, Stand: 09.05.2022

Theoretische Informatik

The problem of packing modification-disjoint P_3 – an overview and an improved heuristic approach

Jona Dirks,¹ Enna Gerhard²

Abstract: A P_3 is a path on three vertices. Two P_3 are *modification-disjoint* if they share at most one vertex. In this work we consider the problem of packing modification-disjoint induced P_3 .

To the best of our knowledge it is not known whether the problem of finding a maximum packing of modification-disjoint P_3 can be solved in polynomial time. We provide new insights towards this question.

We provide an overview and new insights for locating modification-disjoint P_3 packing within the complexity hierarchy. Accordingly, we further look into conflict graphs.

Complementing our theoretical results, we present a significantly improved heuristic based on the approach of Spinner [Sp19]. We analyze its efficiency empirically on a selection of generated and public datasets. Our results improve on existing heuristics when comparing solution size and running time.

Keywords: P_3 Packing; Cluster Editing; Graph Theory; Heuristic; Complexity

1 Introduction

A P_3 is an induced path on three vertices. Two P_3 are called modification-disjoint if they share at most one vertex. The modification-disjoint P_3 packing problem (PPP) is to find a largest set of pairwise modification-disjoint P_3 within a given graph.

To the best of our knowledge, PPP has not been studied in the literature. It is not even known whether PPP is solvable in polynomial time.

PPP is very relevant for the cluster editing problem (CEP). The CEP is the problem of finding a $M \subseteq E$ such that $(V, E \Delta M)$ is a cluster graph with all components being cliques such that $|M| \leq k$. PPP can be used to find a lower bound for CEP, which can be applied in branching algorithms. CEP has many applications, for example in bioinformatics, image processing and circuit design, see e.g. [SST04]. Solving CEP efficiently was the goal of the PACE 2021 Challenge in which we submitted the best student solver [Di21b].

¹ University of Bremen, Faculty 3, Bibliothekstraße 1, 28334 Bremen, Germany
dirks2@uni-bremen.de

² University of Bremen, Faculty 3, Bibliothekstraße 1, 28334 Bremen, Germany
gerhard@uni-bremen.de

After introducing some notation in Sect. 2, we discuss results regarding the complexity of PPP in Sect. 3. One possible way to show complexity bounds could be to understand the graph class of conflict graphs. We will study these in Sect. 3.2. Afterwards, we use proven characteristics of these conflict graphs to improve upon known heuristics in Sect. 4.

2 Definitions and notation

All graphs in this paper are undirected and do not contain double edges or loops. We use standard graph notation. We write ab for an edge between vertices a and b . A $K_{a,b}$ is the complete bipartite graph partitioned into a and b vertices. W_6 is the six wheel. See Fig. 1.

We represent a P_3 via the set of vertices that it contains. For simplicity such a set over the vertices a, b and c is denoted as abc . We require that P_3 are induced. In most cases we also assume that the non-edge is ac . The set of all induced P_3 of a graph G is denoted with $P_3(G)$, see Fig. 2 for $P_3(W_6)$.

Two P_3 , abc and def are modification-disjoint if they share one or less vertices, that is, if $|abc \cap def| \leq 1$. In this case, they share neither an edge ab or bc nor the non-edge ac . We say that two P_3 are in conflict, if they are not modification-disjoint. The conflict graph for a graph G is induced by the edges $\{pq \mid p, q \in P_3(G), p \neq q, p \text{ in conflict with } q\}$. In case of the six wheel W_6 , this becomes a Petersen graph, see Fig. 3.

The class of F -free graphs for a set of graphs F is the set of graphs that do not contain any graph from F as induced subgraphs. A *cluster graph* is a graph of fully connected components. The class of cluster graphs is exactly the class of P_3 -free graphs.

3 Observations regarding the complexity of PPP

This section discusses the decision problem variant DPPP where (G, k) is a yes-instance, if there exists a modification-disjoint P_3 packing in G of size at least k . To the best of our knowledge, it is yet unknown whether DPPP is NP-complete or solvable in polynomial time. There are similar problems that are within either of these classes. The most promising approach for solving PPP is a reduction to independent set on the restricted graph class of conflict graphs which we will explore in Sect. 3.2.

3.1 Related problems

There are several related problems for which the complexity is known. For example if we ignore the non edge and only search for edge disjoint P_3 packings, Algorithm 1 runs in polynomial time [DS21]. A maximum matching in G can be computed using the Blossom-Algorithm by Edmonds [Ed65].

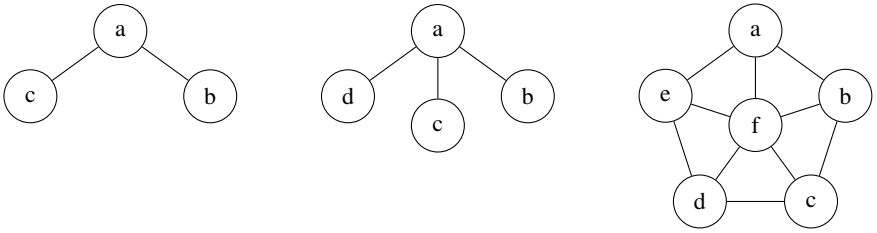


Fig. 1: $K_{1,2}$ containing one induced P_3 , the claw $K_{1,3}$ and the six wheel W_6

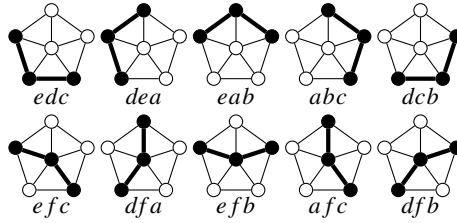


Fig. 2: $P_3(W_6)$, all induced P_3 on the six wheel W_6

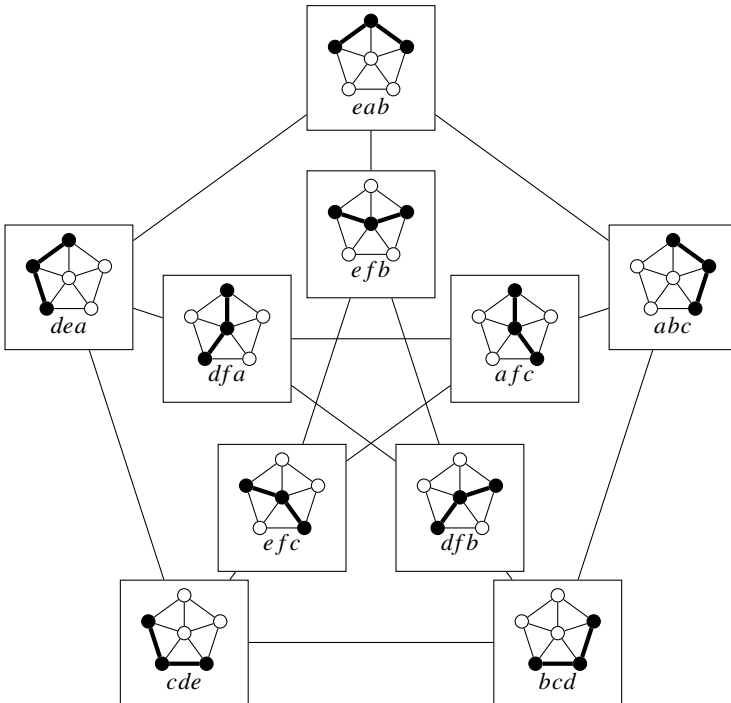


Fig. 3: Conflict graph of W_6

Algorithm 1: Polynomial ($O(|E|^4)$) time edge disjoint P_3 packing algorithm**Input:** Graph G , size k **Output:** A packing of size k exists $EG := (E(G), \{e_1e_2 \mid e_1, e_2 \in E(G) \text{ and } e_1 \text{ adjacent to } e_2\})$ ▷ The edge graph**for each** $e_1e_2 \in E(EG)$ **do** **if** $e_1 \cup e_2 \notin P_3(G)$ **then** Remove edge e_1e_2 from EG $S := \emptyset$ **for each matching** $\{e_1, e_2\} \in \text{maximumMatching}(EG)$ **do** add $e_1 \cup e_2$ to S ▷ The set $e_1 \cup e_2$ induces a P_3 **return** $|S| \geq k$

On the other hand, if we require that all vertices have to be covered by vertex-disjoint P_3 , then the problem becomes NP-complete [Pa94].

3.2 Characteristics of conflict graphs

Observation 3.1. *A maximum modification-disjoint P_3 packing corresponds directly to an independent set in the conflict graph.*

While the independent set problem is NP-complete in general, it can be solved efficiently in many restricted graph classes, e.g. in $K_{1,3}$ -free (claw-free) graphs. This motivates our study of the class of conflict graphs. A very interesting property of conflict graphs is the neighbour covering of cliques which can be formulated as:

Theorem 3.2. *For each vertex v of a conflict graph C of G there exist $Q, R, S \in \text{clique}(v)$ such that each $u \in N(v)$ lies in $Q \cup R \cup S$. Here, $\text{clique}(v)$ denotes the set of all cliques which contain v .*

Proof. For each vertex in C there exists a P_3 ijk in G . This P_3 has three possible conflicts. If we collect all P_3 that are in conflict via a (non-)edge ab into one set $[ab]$, then this P_3 has three such sets, namely $[ij]$, $[jk]$ and $[ki]$. The P_3 of each set form a clique in the conflict graph C , because they are all in conflict via the name giving edge. Therefore, we can choose these sets as the cliques Q, R and S [Di21a]. □

We next prove that conflict graphs cannot be described as F -free for any set F . This is unfortunate, as F -free graphs are well studied. For example if conflict graphs would be $K_{1,3}$ -free, which they are not, then DPPP would be solvable in polynomial time due to the mentioned connection with independent set [Sb80]. On the other hand, if they were characterized as the class of $K_{1,4}$ -free graphs, then independent set would be NP-complete on them [Mi80]. To prove this, we first show inclusion and exclusion of certain subgraphs as conflict graphs:

Lemma 3.3. *If a graph G has two not necessarily induced P_3 abc and adc then $abc \in P_3(G) \Leftrightarrow adc \in P_3(G)$.*

Proof. It is obvious that we only have to show one implication, because the other one is analog. Let $abc \in P_3(G)$. If adc was not induced, then there would have to be an edge ac . The P_3 abc would not be induced either. \square

Lemma 3.4. *$K_{1,3}$ is not a conflict graph.*

Proof. We can show that there is just a single possibility for a $K_{1,2}$ to be present without structures that are in conflict with a $K_{1,3}$. We do so by looking at three cases with subcases that are visualized in Fig. 4. We then show that there is no structure that leads to a $K_{1,3}$.

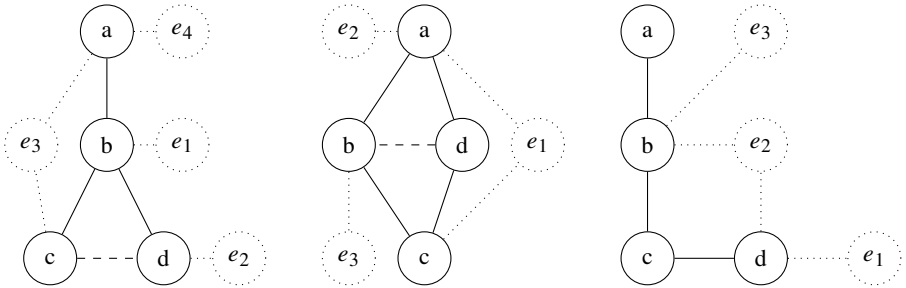


Fig. 4: The possible ways of extending conflict graphs with a $K_{1,1}$

Case 1. There are two P_3 in conflict via one edge. So without loss of generality, $abc, abd \in P_3(G)$. Then there also has to be an edge dc , otherwise the conflict graph would contain a triangle. Now there are four different possibilities that potentially lead to a $K_{1,2}$.

Case 1.1. There exists a P_3 abe_1 . This P_3 is in conflict with both abc and abd . Therefore, the resulting graph can never be a $K_{1,3}$, even if further edges or vertices are present.

Case 1.2. There exists a P_3 e_2db (the case e_2cb is analog). To avoid the P_3 e_2dc , there has to be an edge e_2c . However, the P_3 e_2cd must exist – because of Lemma 3.3 the P_3 e_2db would not be possible otherwise.

Case 1.3. There exists a P_3 ae_3c (the case de_3a is analog). To avoid the P_3 bae_3 , the edge be_3 has to exist. Now the edge de_3 has to exist to avoid the P_3 dbe_3 . Because of Lemma 3.3, the P_3 de_3a must be present, otherwise abd could not exist.

Case 1.4. There exists a P_3 e_4ab . In this case, it is possible to construct a $K_{1,2}$. We will discuss this case at the end of the proof.

Case 2. There are two P_3 in conflict via the non-edge. Without loss of generality, $abc, adc \in P_3(G)$. Then there also has to be an edge bd . Now there are three different possibilities that potentially lead to a $K_{1,2}$.

Case 2.1. There exists a P_3 ae_1c . This P_3 is in conflict with both abc and adb . Therefore, the resulting graph can never be a $K_{1,3}$, even if further edges or vertices are present.

Case 2.2. There exists a P_3 e_2ab (the case e_2cd is analog). To avoid the P_3 e_2ad the edge de_2 has to exist. Because of Lemma 3.3, the P_3 bde_2 must be present, otherwise the P_3 e_2ab could not exist.

Case 2.3. There exists a P_3 e_3ba (the case e_3da is analog). To avoid the P_3 dbe_3 the edge de_3 has to be present. Because of Lemma 3.3, the P_3 e_3da must be present, otherwise the P_3 e_3ba could not exist.

Case 3. $abc, bcd \in P_3(G)$

Case 3.1. There exists a P_3 cde_1 (the case e_1ab is analog). This would produce a P_3 as a conflict graph, which cannot be part of a $K_{1,3}$.

Case 3.2. The P_3 be_2d or ae_2c exist. This case is analog to Case 2.2.

Case 3.3. The P_3 abe_3 or dce_3 exist. This case is analog to Case 1.4.

We return to Case 1.4. To extend the conflict graph of it to a $K_{1,3}$ the new P_3 has to contain e . Otherwise, a $K_{1,2}$ would exist, which has been covered above. We can now look at three cases where another P_3 could appear, which can be seen in Fig. 5.

Case 1. There exists a P_3 $ae f_1$. This would produce a path of length 4 (P_4) as a subgraph in the conflict graph.

Case 2. There exists a P_3 f_2ae . To avoid a P_3 baf_2 , there have to be edges bf_2 , cf_2 and df_2 . Now because of Lemma 3.3, the P_3 df_2a cannot be avoided with abd present.

Case 3. There exists a P_3 $b f_3e$. We then need edges af_3 , cf_3 , df_3 . Now we cannot avoid the P_3 df_2a with abd existing.

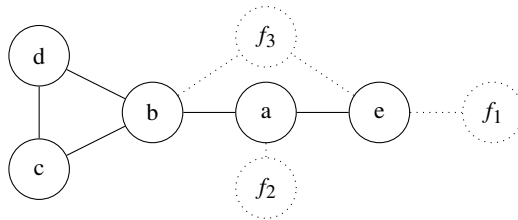


Fig. 5: The possible ways of extending a $K_{1,2}$

□

However this does not rule out a $K_{1,3}$ as an induced subgraph:

Lemma 3.5. *There are non- $K_{1,3}$ -free conflict graphs.*

Proof. The conflict graph of W_6 (Fig. 3) has the induced $K_{1,3}$ over the vertices efb , dfb , eab and efc . \square

With this we can prove the following theorem:

Theorem 3.6. *Conflict graphs cannot be described as F -free for any graph set F .*

Proof. Assume towards a contradiction that there is an F such that conflict graphs are exactly the class of F -free graphs. By definition of the induced subgraph relation any induced subgraph of a conflict graph has to be also F -free. Because of Lemma 3.5 the graph $K_{1,3}$ is then also F -free. This would mean that $K_{1,3}$ is a conflict graph, which was disproved in Lemma 3.4. \square

3.3 Reduction to independent set on conflict graphs

We know that if we can solve independent set on the restricted graph class of conflict graphs in polynomial time, we can also solve DPPP. Unfortunately, as we have seen in the previous section, this graph class is hard to describe and does not fit in the well studied concept of F -free graphs. Even intuitively, this problem is hard to narrow down, as it is similar both to problems that are solvable in polynomial time and problems that are NP-complete.

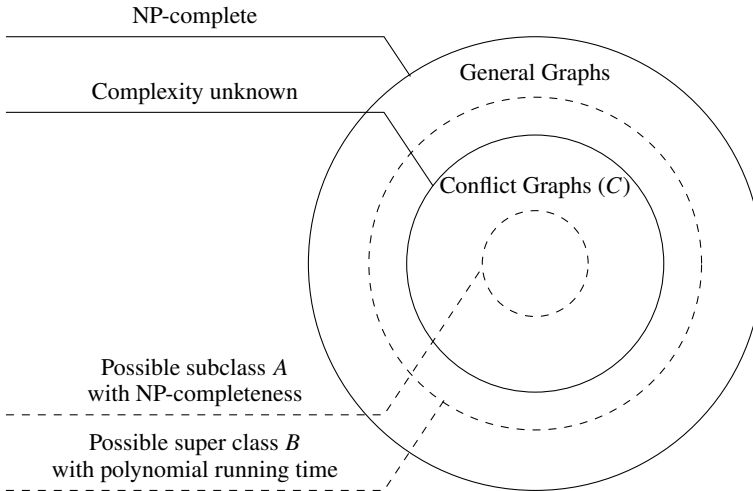


Fig. 6: Complexity of the independent set problem on different graph classes. By Theorem 3.6 we get $A \neq C \neq B$, if A and B are described as F -free

There might be a subclass of conflict graphs where solving independent set is NP-complete or a super class for which a polynomial time algorithm exists. This relationship can be seen in Fig. 6. Independent set remaining NP-hard on conflict graphs would not imply the same

for DPPP as we haven't been able to reconstruct the original graph for a conflict graph in the general case. As a result, it is still unknown to which complexity class this problem belongs.

4 Heuristic approaches to PPP

As the complexity of PPP remains unknown, our best possible approach is to use heuristics. In this section, we will introduce known heuristics and show that with our results about conflict graphs we are able to improve upon these.

4.1 Known heuristics

A possible way to solve this problem is to reduce it to an independent set instance and then solve this one. One such local search heuristic was developed by Nogueira et al. [NPS18]. The heuristic is based on swaps: An (a, b) -swap removes a vertices from the solution and adds b to the solution. The heuristic searches for $(w, 1)$ -swaps and $(1, 2)$ -swaps. The first case can only happen in weighted instances and thus is left out in our adaptation. The starting solution is found greedily. Nogueira et al.'s heuristic also makes a perturbation step that modifies the solution. These steps are repeated until some termination criteria are met.

Spinner later adopted this heuristic to find packings for various graphs [Sp19]. However, for some unknown reason he does not make use of the perturbation step and also simplified the termination criteria. In our implementation the $(w, 1)$ -swap was also left out.

4.2 Novel heuristic

Especially the results from Theorem 3.2 allow us to improve on Spinners heuristic. The algorithm calculates these cliques while searching for the set of P_3 . It can then compute possible $(1, 2)$ -swaps faster, because any two vertices of a swap have to come from two different cliques in the conflict graph. While in theory such a swap can be found in any combination of cliques, we decided to ignore the one that is produced by the non-edge, as this did not improve results.

The resulting practical time improvement allows us to include other attempts to increase solution size while still being faster than the heuristic developed by Spinner:

- If we found a $(1, 2)$ -swap, we can try to improve it to a $(1, 3)$ -swap.
- We attempt applying non-improving $(1, 1)$ -swaps, at least once, or as long as further improvements are found. This is a minimalistic version of the perturbation step done by [NPS18].

- We can search for simplicial vertices and force add them into the solution. A vertex is called simplicial, if its neighbors form a clique. This is based on a reduction rule developed by [St16]. Theorem 3.2 allows us to check whether all but one of the cliques are empty.

The full heuristic in pseudocode is shown in Algorithm 2. A $(1, 3)$ - or $(1, 2)$ -swap can be calculated with Algorithm 3. A $(1, 1)$ -swap is trivial, it replaces a P_3 with a different one, thus retaining solution size. The greedy packing can be calculated by iterating through all triples of vertices and adding them to P if they are a P_3 and not in conflict with any previously added P_3 . The cliques from Theorem 3.2 ($[ab]$, $[bc]$, $[ca]$ for $P_3 abc$) can also be calculated in this step. A further improvement to running time in practise is to keep track of the number of P_3 in P that any P_3 is in conflict with.

Algorithm 2: Improved heuristic

Input: A Graph G

Output: Packing P

Function $isSimplicial(p)$ **is**

return $||[ab]|| > 0$ and $||[bc]|| = 0$ and $||[ca]|| = 0$ for pairwise unequal $a, b, c \in p$

$P := greedyPacking(G)$

$fixed(p) := isSimplicial(p), \forall p \in P$

for each $p \in (P_3(G) \setminus P)$ with $isSimplicial(p)$ **do**

$fixed(p) := true$

 remove all $[ab]$ from P

$attempt := 0$

while $attempt \leq 1$ **do**

$attempt += 1, P_{last} := \emptyset$

while $|P_{last}| < |P|$ **do**

$P_{last} := P$

$S := swapSearch(P, fixed)$

▷ Algorithm 3

if S exists **then**

 apply S to P

$attempt := 0$

$S := find(1, 1)\text{-swap}$

if S exists **then**

 apply S to P

else

return P

return P

We obtain a worst case running time of $O(|P_3(G)|^4) \cdot O(|P_3(G)|) = O(|P_3(G)|^5)$, where the first factor stems from Algorithm 2 and the second from Algorithm 3. The amount of $P_3(G)$ can be at most $|V|^3$. This is a worst case estimate and there are hidden factors that drastically decrease running time in practice. If we have large cliques in the conflict graph and thus a high running time of Algorithm 3, we typically obtain a small maximum packing and therefore a smaller amount of iterations in Algorithm 2 will be sufficient.

Algorithm 3: Swap search for (1, 2)-swaps and (1, 3)-swaps**Input:** A packing P , status *fixed***Output:** A suggested swap, if one exists**Function** *conflict*(p, P) **is**

```

|   return  $|\{p' \mid p' \in P : p' \text{ is in conflict with } p\}|$ 
for each  $ijk \in P$  and  $\neg \text{fixed}(ijk)$  do
|   for each  $a \in [ij]$ ,  $a, \text{conflict}(p, P) \leq 1$  and  $\neg \text{fixed}(a)$  do
|       for each  $b \in [jk]$ ,  $\text{conflict}(a, P) \leq 1$  and  $\text{conflict}(b, \{a\}) = 0$  and  $\neg \text{fixed}(b)$  do
|           for each  $c \in [ik]$ ,  $\text{conflict}(c, P) \leq 1$  and  $\text{conflict}(c, \{a, b\}) = 0$  and  $\neg \text{fixed}(c)$ 
|               do
|                   return swap  $ijk$  for  $a, b, c$ 
|           return swap  $ijk$  for  $a, b$ 
return no swap exists

```

4.3 Empiric results

To show that our heuristic performs better than previously known heuristics, we evaluated it on two types of graphs. First we used randomly generated relatively small graphs in order to show the relation to the slow heuristics. Secondly we used the smallest 50 public instances of the PACE Challenge [Ke21]. We generated 50 graphs of the sizes $n = 10, m = 20$ (small), $n = 20, m = 80$ (medium), $n = 20, m = 100$ (large) and $n = 40, m = 300$ (huge).

The results are shown in Tab. 1. For evaluating the running time we used an AMD Ryzen 5 3600 6-Core processor with 16 gigabyte of RAM. For the ILP formulation we used a translation of the folklore ILP for independent set on the conflict graph: $\max \sum_{p \in P_3(G)} x_p$ such that $(x_p + x_{p'} \leq 1, \forall p, p' \in P_3(G) : p \text{ in conflict with } p')$ and $(x_p \in \{0, 1\}, \forall p \in P_3(G))$ (see [BB13]).

The greedy heuristic is fastest, but its solution sizes get outperformed on all but the smallest generated graphs. While Nogueira et al. produces almost perfect solutions, which compare to the exact ILP, it is very slow. On most graph types it is even slower than the ILP. We were not able to compute Nogueira nor the ILP on the huge graphs of the PACE Challenge within reasonable time. Thus, both algorithms seem too slow to solve PPP on real world instances.

With respect to the running time, the novel heuristic is faster on all graphs compared to Spinner. On the large graphs the novel heuristic has a speedup of a magnitude of at least ten. On the pace graphs this time advantage is only around 10%, although it is still clearly visible.

Regarding the packing size, the results are not so clear but still indicate a lead for the novel heuristic. In two categories the novel heuristic produces larger packings, otherwise it performed equally well.

However, the PACE dataset is highly irregular, and a simple average hides this. There is more than a 100-fold increase in edge count between the smallest and the largest graphs. A

Tab. 1: Empirical results, average over tested instances of the respective sizes

Graph Type	Solution size				
	Greedy	Spinner	Novel	Nogueira	ILP
small	8	8	9	9	9
medium	37	38	39	39	40
large	45	47	48	49	50
huge	146	149	149	150	150
pace	677	712	712	timeout	timeout

Graph Type	Running time in milliseconds				
	Greedy	Spinner	Novel	Nogueira	ILP
small	0	2	0	54	11
medium	0	86	6	2074	271
large	0	142	10	3395	528
huge	0	2449	444	63354	63978
pace	19	400903	363999	timeout	timeout

small disadvantage on one of the larger graphs could cover advantages on many smaller graphs. The novel heuristic has better results on 23 graphs while Spinners heuristic performs better on 21 graphs, giving the novel heuristic a small lead. The novel heuristic is faster on all but three graphs regarding the running time.

5 Conclusion and further research

We have explored PPP and explained several of its properties.

There are many similar problems that are NP-complete or in P. The exact classification of PPP remains interesting for further research.

We have found quite a few characteristics about conflict graphs, such as the partition of neighbours of each vertex into three cliques. Although we were only able to show that conflict graphs can not be described with commonly used techniques, we have added insights for further developments. We were able to use our results to improve upon know heuristics both in terms of speed and of solution size, and we were able to demonstrate these gains.

References

- [BB13] Böcker, S.; Baumbach, J.: Cluster Editing. In (Bonizzoni, P.; Brattka, V.; Löwe, B., eds.): The Nature of Computation. Logic, Algorithms, Applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 33–44, 2013, ISBN: 978-3-642-39053-1, URL: <https://doi.org/10.1007/978-3-642-39053-1>.

- [Di21a] Dietrich, K.; Dirks, J.; Gahde, J.; Heydt, O.; Schnaubelt, Y.; Sczuka, F.; Siering, N.; Sonneborn, M.; Stichternath, L.; Tat, J.; Freese, T.: Parametrisierte Algorithmen auf Graphen, Project Report, Bremen, Germany, July 2021.
- [Di21b] Dirks, J.; Grobler, M.; Rabinovich, R.; Schnaubelt, Y.; Siebertz, S.; Sonneborn, M.: PACE Solver Description: PACA-JAVA. In (Golovach, P. A.; Zehavi, M., eds.): 16th International Symposium on Parameterized and Exact Computation (IPEC 2021). Vol. 214, Dagstuhl, Germany, 30:1–30:4, 2021.
- [DS21] Dirks, J.; Schnaubelt, Y.: Modellbasierte Parameterkonfiguration für parametrisierte Graphprobleme, (Sieberts, S.; Wright, M., supv.), Bachelor Thesis, Bremen, Germany: University of Bremen, Sept. 2021.
- [Ed65] Edmonds, J.: Paths, Trees, and Flowers. *Canadian Journal of Mathematics* 17/1, pp. 449–467, 1965, URL: <https://doi.org/10.4153/CJM-1965-045-4>.
- [Ke21] Kellerhals, L.; Koana, T.; Nichterlein, A.; Zschoche, P.: The PACE 2021 parameterized algorithms and computational experiments challenge: Cluster editing. In. Vol. 214, 2021, URL: <https://doi.org/10.4230/LIPIcs.IPEC.2021.26>.
- [Mi80] Minty, G. J.: On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B* 28/3, pp. 284–304, 1980, ISSN: 0095-8956, URL: [https://doi.org/10.1016/0095-8956\(80\)90074-X](https://doi.org/10.1016/0095-8956(80)90074-X).
- [NPS18] Nogueira, B.; Pinheiro, R. G. S.; Subramanian, A.: A hybrid iterated local search heuristic for the maximum weight independent set problem. *Optimization Letters* 12/3, pp. 567–583, 2018, URL: <https://doi.org/10.1007/s10878-014-9756-7>.
- [Pa94] Pantel, S.: Graph Packing Problems, (Hell, P.; Alspach, B.; Brewster, R., supv.), Master Thesis, Manitoba, Canada: University of Manitoba, Oct. 1994.
- [Sb80] Sbihi, N.: Algorithme de recherche d’un stable de cardinalité maximum dans un graphe sans étoilé. *French, Discrete Math.* 29/, pp. 53–76, 1980, ISSN: 0012-365X, URL: [https://doi.org/10.1012-365X\(90\)90287-R](https://doi.org/10.1012-365X(90)90287-R).
- [Sp19] Spinner, J.: Weighted F-free Edge Editing, (Hamann, M.; Gottesbüren, L., supv.), Bachelor Thesis, Karlsruhe, Germany: Karlsruhe Institute of Technology, 2019, URL: <https://doi.org/10.5445/IR/1000135117>.
- [SST04] Shamir, R.; Sharan, R.; Tsur, D.: Cluster graph modification problems. *Discrete Applied Mathematics* 144/1, *Discrete Mathematics and Data Mining*, pp. 173–182, 2004, ISSN: 0166-218X, URL: <https://doi.org/10.1016/j.dam.2004.01.007>.
- [St16] Strash, D.: On the Power of Simple Reductions for the Maximum Independent Set Problem. In (Dinh, T. N.; Thai, M. T., eds.): *Computing and Combinatorics*. Springer International Publishing, Cham, pp. 345–356, 2016, ISBN: 978-3-319-42634-1, URL: https://doi.org/10.1007/978-3-319-42634-1_28.

Computing Treewidth with Constraint Programming

Florentina Voboril¹

Abstract: In this paper, we revisit a known SAT encoding for the fundamental combinatorial treewidth problem. Based on this encoding, we rework it within the constraint modeling language MiniZinc. Two MiniZinc encodings for treewidth are created and their performance is compared in an experimental evaluation. A further dimension for comparison is added by choosing between different back-end solvers supported by MiniZinc.

Keywords: Treewidth; SAT; Constraint Programming; MiniZinc

1 Introduction

Several hard combinatorial problems have been successfully solved in recent years by encoding them into an instance of the propositional satisfiability problem (SAT), which determines if there exists an assignment that satisfies a given propositional formula. The aim of this paper is to revisit a known SAT encoding of the treewidth problem, which is an important problem from graph theory, and rework it within the MiniZinc constraint modeling framework². This provides a more high-level perspective on the encoding. One can immediately see and compare the performance of different modeling choices. Moreover, by choosing among several back-end solvers supported by MiniZinc, one obtains a further dimension for comparison and experimentation.

The paper provides relevant background information about constraint solving, the MiniZinc constraint modeling language and the graph input format needed for MiniZinc in Section 2. In Section 3 we look at the definition of treewidth and its known SAT encoding. In Section 4 we describe our two MiniZinc encodings. In the experimental evaluation in Section 5 we compare those encodings and observe that the SAT encodings from literature performs better than the numeric encoding. We furthermore compare the performance of three different MiniZinc solvers. Finally, Section 6 concludes and sketches further work.

¹ TU Wien, Fakultät für Informatik, Favoritenstraße 9-11, 1040 Vienna, Austria florentina.voboril@tuwien.ac.at

² <https://www.minizinc.org/>

2 Preliminaries

2.1 Conjunctive Normal Form

A Conjunctive Normal Form (CNF) formula just consists of conjunctions of clauses, where a clause is a disjunction of literals. A literal is a (possibly negated) propositional variable, which can be assigned to true or false. A CNF formula is satisfiable if there is an assignment such that if every clause contains a variable assigned to true or a negated variable assigned to false. Determining whether a formula is satisfiable is known as the SAT problem. This problem is known to be NP-complete.

2.2 Constraint Satisfaction Problem

A constraint satisfaction problem (CSP) is represented by decision variables and constraints. The constraints specify relations between the variables. They forbid some subsets of variable combinations. In the context of MiniZinc, a constraint corresponds to a CNF clause. A CSP can be solved by a constraint solver. A solution to a CSP is an assignment to all decision variables which satisfies the given constraints. The solver can either search systematically with backtracking or branch and bound algorithms, or use forms of local search. It can be very hard to find a good encoding for the chosen solver to solve a problem effectively [RvBW06].

It is hard to pinpoint the origins of constraint solving. As argued by Freuder and Mackworth [FM06], real world problems of this kind have always been with us. A form of backtracking search, that has become a central tool for constraint satisfaction, had already appeared in Mythology. It is part of folklore, that Theseus used this kind of search to find a way out of the labyrinth of the Minotaur. The 8-queens problem, which now is a famous problem in the field of constraint satisfaction, has presumably been proposed as far back as 1848.

2.3 MiniZinc

MiniZinc is an open-source constraint modeling language. It can be used for constraint satisfaction problems, as well as optimization problems. The models can be created independently of the solvers, which range from SAT solvers, to Constraint Programming (CP) solvers and Mixed Integer Linear Programming (MIP) solvers. In our experimental evaluation we will use the solvers Chuffed, COIN-BC and Gecode. Chuffed³ is a CP solver, which makes use of lazy clause generation and SAT solving techniques, including activity-based search heuristics, watched literal propagation and conflict clause learning. This often makes it much more effective than other CP solvers [SMT20]. Gecode⁴ is an

³ <https://github.com/chuffed/chuffed>

⁴ <https://github.com/Gecode/gecode>

open source C++ toolkit, which provides a CP solver. Many of the global constraints and search annotations of the MiniZinc standard library are supported by Gecode. Support for integer, float, and set variables is also provided [SMT20]. COIN-BC⁵ is an open-source solver for Mixed Integer Programming, which has warm starts implemented. This means, that it can make use of previous solutions found for similar problems [SMT20].

MiniZinc also offers a library of many global constraints. `all_different(x)` constrains all variables of the array x to be different from one another. `count_geq(x, y, c)` restricts a variable c to be greater than or equal to the number of occurrences of an element y in an array x . An easy way to use MiniZinc is the MiniZinc IDE. Here, the models and data files can be edited and solved with one of the provided solvers. An introduction to the modeling language, as well as detailed examples can be found in the MiniZinc Handbook [SMT20].

2.4 Graph Input Format

The input data for MiniZinc models have to be in files of data type `.dzn`. We use a simple format for the representation of input graphs. We just have to specify the number of vertices n , the number of edges m and an array of all edges. The labeling of the vertices starts with 1 and goes up to n . For all edges, the vertex with smaller index is written first. The following example represents the Petersen graph.

```
n = 10;
m = 15;
edges = [{1, 3}, {1, 4}, {1, 6}, {2, 4}, {2, 5}, {2, 7}, {3, 5}, {3, 8},
        {4, 9}, {5, 10}, {6, 7}, {6, 10}, {7, 8}, {8, 9}, {9, 10}];
```

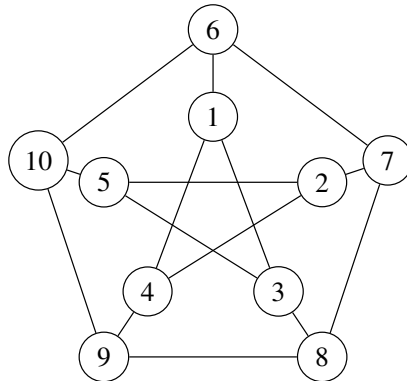


Fig. 1: Petersen graph

⁵ https://github.com/MiniZinc/libminizinc/blob/master/solvers/MIP/MIP_osicbc_wrap.cpp

3 Treewidth

The concept of treewidth was first introduced by Bertelè and Brioschi in 1972 under the name *dimension* [BB73]. Independent of this, in 1984 the term *treewidth* was introduced by Robertson and Seymour [RS84]. This parameter is an indicator for “tree-likeness” of an undirected graph. Since many NP-hard combinatorial problems are easy to solve on trees, this knowledge can be used in order to solve instances with low treewidth more efficiently [SV09].

In order to define the treewidth, we need to introduce tree decompositions. The following definition is given by Robertson and Seymour [RS84]. A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (V', E')$ where each node $t \in V'$ is associated with a “bag” $\chi(t) \subseteq V$ containing vertices of G , satisfying the following properties:

- (T1) $\bigcup_{t \in V'} \chi(t) = V$
- (T2) For every edge $e \in E$, there is some node $t \in V'$ such that both ends of e are in $\chi(t)$
- (T3) For all nodes $t_1, t_2, t_3 \in V'$ such that t_2 is on the path between t_1 and t_3 in T , it applies that $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$

The *width* of a tree decomposition is given by the size of its largest bag minus 1, i.e. $\max_{t \in V'} |\chi(t)| - 1$. The minimum width over all tree decompositions of a graph yields its *treewidth*. It is NP-hard to determine the treewidth of a graph. Figure 2 shows an example of a graph with treewidth of 2 and one of its tree decompositions of width 2.

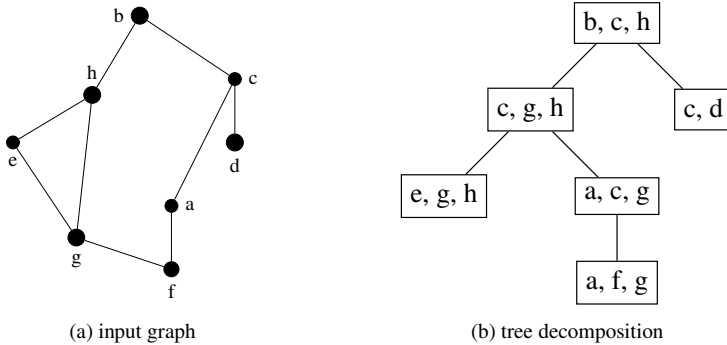


Fig. 2: Graph with treewidth of 2 and corresponding tree decomposition

The encoding of a treewidth problem to SAT by Samer and Veith [SV09] is based on an alternative characterization, an elimination ordering, which is described by Bodlaender [Bo05]. Consider a graph $G = (V, E)$. The treewidth of G can be determined directly from the elimination ordering of its vertices such that no tree decomposition is needed. In an elimination ordering, we have all vertices v_1, v_2, \dots, v_n of V . v_j is a *predecessor* (resp.

successor) of v_i if $v_i v_j \in E$ and $j < i$ (resp. $j > i$). For all vertices v_i and v_j , which have a common predecessor, we iteratively add the edge $v_i v_j$ to E . Such added edges are called fill-in edges. The width of the corresponding tree decomposition is the maximum number of successors over all vertices in the filled-in graph, i.e., $\max_{v_i \in V} |\{v_i v_j \in E : j > i\}|$.

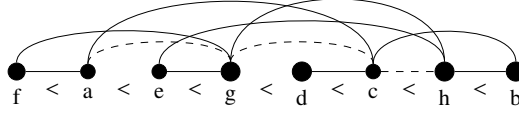


Fig. 3: Elimination ordering of graph with width of 2

Figure 3 shows one of the elimination orderings corresponds to the tree decomposition in Figure 2. The dashed lines represent fill-in edges. For example, the fill-in edge ag is added, because both a and g have vertex f as predecessor. Adding ag causes adding the fill-in edge gc , because vertex a is predecessor of vertices c and g . Since there is no vertex with more than two outgoing edges, the width for the corresponding tree decomposition is 2.

4 MiniZinc Encoding of Treewidth

4.1 Boolean Encoding Based on SAT Clauses (SAT-B)

The following MiniZinc encoding is the CNF formulations for treewidth by Samer and Veith [SV09] as introduced above. Let $G = (V, E)$ be the given graph with $n = |V|$ and $m = |E|$, where all vertices are enumerated from 1 to n . Let k be a given integer. Samer and Veith construct a propositional CNF formula $F(G, k)$, which is satisfiable if and only if $\text{treewidth}(G) \leq k$. They need $O(k|V|^2)$ variables and $O(|V|^3)$ clauses with each clause having at most 4 literals. In the following, this encoding is explained in more detail, together with the equivalent formulations in MiniZinc format.

To encode the ordering of the vertices, which gives a direction to all edges, we need $n(n-1)/2 = \binom{n}{2}$ Boolean variables $\text{ord}[i, j]$ with $1 \leq i < j \leq n$ where

$$\text{ord}[i, j] = \begin{cases} \text{true} & \text{if vertex } v_i \text{ precedes vertex } v_j \text{ in the elimination ordering;} \\ \text{false} & \text{otherwise.} \end{cases}$$

```
array[1..(n-1), 1..n] of var bool: ord;
```

To enforce transitivity of the ordering we need the following $n(n-1)(n-2)$ clauses.

$$\bigvee_{i \neq j \neq l} (\text{ord}[i, j] \wedge \text{ord}[j, l] \implies \text{ord}[i, l])$$

```

constraint forall(i, j, l in 1..n where i != j /\ i != l /\ j != l) (
  ( if i < j then ord[i, j] else not ord[j, i] endif
  /\ if j < l then ord[j, l] else not ord[l, j] endif )
  -> if i < l then ord[i, l] else not ord[l, i] endif
);

```

To represent the successor relation induced by this elimination ordering, we need n^2 Boolean variables $\text{arc}[i, j]$ with $1 \leq i, j \leq n$ where

$$\text{arc}[i, j] = \begin{cases} \text{true} & \text{if there is an arc from vertex } i \text{ to vertex } j; \\ \text{false} & \text{otherwise.} \end{cases}$$

```

array[1..n, 1..n] of var bool: arc;

```

An arc variable is called *active* if it is assigned to true and *inactive* otherwise. The maximum number of active outgoing arcs is the maximum number of successors. This number equals the width of the corresponding tree decomposition. For every edge of the input graph an arc variable must be activated. Therefore we need the following $2m$ clauses.

```

constraint forall(e in 1..m where edges[e][1] < edges[e][2])(
  ord[edges[e][1], edges[e][2]] -> arc[edges[e][1], edges[e][2]]
);

constraint forall(e in 1..m where edges[e][1] < edges[e][2])(
  not ord[edges[e][1], edges[e][2]] -> arc[edges[e][2], edges[e][1]]
);

```

Furthermore, we need to encode the fill-in edges. For this, we need $n(n-1)(n-2)$ clauses of the following form.

```

constraint forall(i, j, l in 1..n where i != j /\ i != l /\ j < l)(
  (arc[i, j] /\ arc[i, l] /\ ord[j, l]) -> arc[j, l]
);

constraint forall(i, j, l in 1..n where i != j /\ i != l /\ j < l)(
  (arc[i, j] /\ arc[i, l] /\ not ord[j, l]) -> arc[l, j]
);

```

We also can add these $n(n-1)(n-2)/2$ clauses. They are redundant, but accelerate the running time of SAT.

$$(\text{arc}[i,j] \wedge \text{arc}[i,l]) \implies (\text{arc}[j,l] \vee \text{arc}[l,j])$$

```
constraint redundant_constraint(
  forall(i, j, l in 1..n where i != j /\ i != l /\ j < l)(
    not arc[i, j] \/ not arc[i, l] \/ arc[j, l] \/ arc[l, j]
  )
);
```

In order to neglect self loops, the following n unit clauses are added.

```
constraint forall(i in 1..n)( not arc[i, i] );
```

Now, that all constraints for adding arcs are encoded, we have to count the number of outgoing arcs from each vertex. Samer and Veith [SV09] used CNF clauses in order to encode a sequential unary counter. Their counter is based on a proposal by Sinz [Si05]. In MiniZinc we can make use of the global constraint `count_geq(x, y, c)`, to make sure that the number of active outgoing arcs does not exceed k .

```
var int: k;
constraint forall(i in 1..n)( count_geq(arc[i, 1..n], true, k) );
```

The encoding of Samer and Veith is a decision problem. The clauses are satisfiable if and only if the treewidth of the input graph is at most k . In our MiniZinc encoding we consider the optimisation problem of minimizing k instead.

```
solve minimize k;
```

4.2 Numeric Encoding Based on SAT Encoding (SAT-N)

The second encoding works according to the same principles as the first encoding. The only difference is, that we use an one-dimensional integer array for the variable `ord` instead of a two-dimensional array of Boolean variables. The position in the elimination ordering corresponds to the index in the array.

```
array[1..n] of var 1..n: ord;
```

In order that every vertex appears only once in the encoding, we use the `alldifferent` constraint.

```
constraint all_different(ord);
```

Since the `ord` array in this encoding is numeric, we do not need the clauses for transitivity of elimination ordering from the previous encoding. To encode the edges of the input graph we have the following constraint.

```
constraint forall(e in 1..m where edges[e][1] < edges[e][2], i,j in 1..n
  where ord[i] == edges[e][1] /\ ord[j] == edges[e][2])(
  if i < j
    then arc[edges[e][1], edges[e][2]]
    else arc[edges[e][2], edges[e][1]]
  endif
);
```

To enforce the additional edges caused by the elimination ordering we add the following constraint.

```
constraint forall(i, j, l in 1..n where i != ord[j] /\ i != ord[l]
  /\ j != l /\ arc[i, ord[j]] /\ arc[i, ord[l]])(
  if j < l then arc[ord[j], ord[l]] else arc[ord[l], ord[j]] endif
);
```

The other clauses are the same as in the previous encoding.

5 Experimental Evaluation

5.1 Setup

All experiments are executed on a computer with 16GB RAM and an Intel(R) Core(TM) i7, 1.80 GHz Quad Core processor. For the benchmarking, MiniZinc version 2.5.3 and the three solvers Chuffed 0.10.4, COIN-BC 2.10.5/1.17.5 and Gecode 6.3.0 are used. As benchmark instances we use graphs from van den Broek and Bodlaender [vdBB10]. We pick only those instances with 2 to 20 vertices and at least one edge. The number of vertices of a graph is called its *order*. The graphs are transformed to the .dzn format as described in Section 2.4 and sorted by their order. Table 1 shows the number of instances with a certain order.

order	2	3	4	5	6	7	8	9	10
#	4	13	25	47	29	16	17	10	9

order	11	12	13	14	15	16	17	18	19	20
#	9	12	6	9	8	25	11	7	10	8

Tab. 1: Number of instances with certain number of vertices

There are six possible combinations of the two encodings and the three solvers. However, with the Gecode solver errors occur in the SAT-N encoding. Thus, we omit this combination,

leaving behind five combinations. We then group all instances of a certain order and run them sequentially with a total timeout of 30 minutes within each group for each of the five combinations. The input instances, the Python programs which executes the benchmarking, and the resulting output files are publicly available ⁶.

5.2 Comparison of the Solvers

Comparing the performance of the three solvers leads to very similar results for both encodings. Table 2 shows the results for the SAT-B encoding. The rows correspond to the different orders. The columns correspond to the used solver. The cells state the number of input instances with corresponding order, which could be solved in 30 minutes by the corresponding solver. For those cases, where all given instances have been solved, we divide 1800 seconds by the average running time. This estimates how many instances with this running time could be solved within 30 minutes, if there was a sufficiently large number of input instances with this order.

	Chuffed	COIN-BC	Gecode
2	3428	1925	1769
3	2500	4500	2321
4	1706	4193	1976
5	1964	1883	1868
6	1809	707	1891
7	1784	312	1818
8	1391	132	1510
9	1573	45	653
10	1375	2	57
11	1273	1	1
12	1095	0	0
13	733	0	0
14	590	0	0
15	552	0	0
16	23	0	0
17	50	0	0
18	7	0	0
19	9	0	0
20	1	0	0

Tab. 2: SAT-B encoding: Number of solved input instances of different graph orders with different solvers

For the SAT-B encoding, the COIN-BC solver delivers the fastest results for small instances with 3 and 4 vertices. But for bigger instances the performance is much slower. Within 30 minutes it just solves two instances of order 10. Instances with order 12 or bigger cannot be solved. For the Gecode solver also no instances of order 12 or larger can be solved. But for

⁶ <https://www.ac.tuwien.ac.at/files/resources/results/minizinc-width.zip>

instances with the order of 6 to 10 it has an obviously better performance than the COIN-BC solver. The Chuffed solver has the overall best performance. It can solve instances of all orders. Figure 4 compares the number of solved instances for different solvers and different encodings.

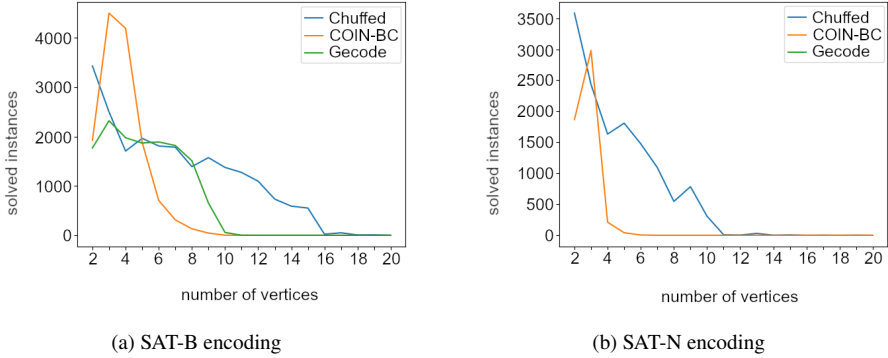


Fig. 4: Number of solved instances by the different solvers for the different encodings

5.3 Comparison of the Encodings

Since the Chuffed solver performs the best and no encoding leads to an error with this solver, we compare the running times of all encodings, solved by Chuffed. We have two encodings: The Boolean encoding based on SAT clauses (SAT-B, Section 4.1) and the numeric encoding based on SAT encoding (SAT-N, Section 4.2). Figure 5a shows how many input instances of different graph orders can be solved with the two different encodings. For each graph order there is a timeout of 30 minutes. For those graph orders, where all given instances are solved within the timeout, we estimate how many instances could have been solved within 30 minutes with respect to the average running time.

Besides graphs of order 2, where the SAT-N encoding is just a little bit quicker, the SAT-B encoding performs better overall. This difference is especially visible for instances of order larger than 10. For graph order 11, for example, the SAT-B encoding can solve 1273 instances whereas the SAT-N encoding can only solve 9 instances. It is interesting to note, that the running times for these 9 instances differ a lot. Most of them are solved within 10 seconds, but the instance called "11_myciel3.dzn" takes 17 minutes, while the instance "11_Grotzsch.dzn" takes about 10 minutes. For the SAT-B encoding however, there are no such huge differences in the individual running times.

Figure 5b makes a comparison of the two encodings using a cactus plot. This means that the solver runs for 30 minutes for each encoding and tries to solve instances of all orders, one after another. The input instances are sorted by the number of vertices. For the individual graph orders there are 4 to 25 instances each. More details can be found in Table 1. The

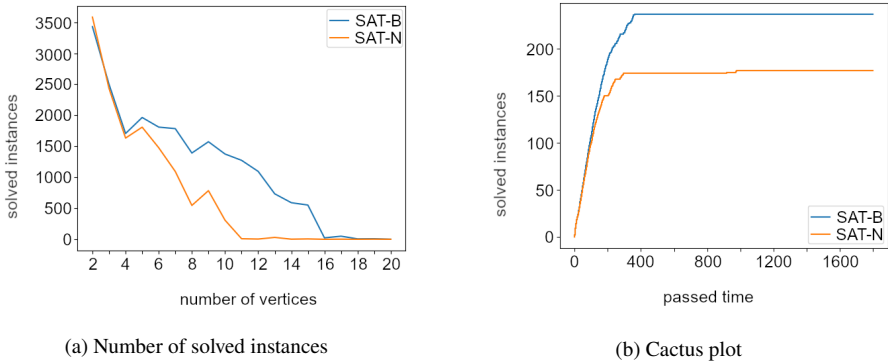


Fig. 5: Performance comparison of the different encodings with the Chuffed solver

cactus plot shows how many instances are solved until a certain time. Again, we can see that the SAT-B encoding performs better than the SAT-N encoding. For the instances with small order, the differences between both encodings are not huge. But for bigger instances, a significant difference can be seen. The two long horizontal lines for the numeric encoding are caused by the two above mentioned instances of graph order 11 with running times of about 10 and 17 minutes. The long horizontal line of the SAT-B encoding is caused by one instance of order 16, that can not be solved within the timeout.

5.4 Discussion

The comparison of the three used solvers shows that Chuffed has the best performance. This might be because of its use of lazy clause generation and SAT solving techniques, as described in Section 2.3. It however, could just solve a single instance of order 20 within a time limit of 30 minutes. When we compare the encodings, we can see that the Boolean SAT encoding, which is presented in the literature, performs better than the numeric encoding. This shows that it makes sense to use just Boolean variables. A reason for this might be that the Chuffed solver internally works with Boolean variables and techniques from SAT solving [SMT20]. By directly providing Boolean variables and SAT instances, less overhead is created. Other solvers would perhaps lead to different results.

6 Conclusion and Future Work

In this paper we revisited a known SAT encoding of treewidth and rework it within the MiniZinc constraint modeling framework. We used the SAT-B encoding based on Samer and Veith. For reasons of comparability we also formulated a numeric encoding of the same problem. In the experimental evaluation it could be seen that the Boolean encoding

performed better. We also compared the performance of the three solvers Chuffed, COIN-BC and Gecode and could show that Chuffed leads to the best results. Besides Chuffed, COIN-BC and Gecode, there are also other solvers available for MiniZinc, such as Gurobi and Globalizer. In the annual MiniZinc Challenge constraint programming solvers compete against each other. In future work, one could run the encodings with one of the best state-of-the-art solvers from the competition to compare the performance.

Furthermore, MiniZinc provides multiple tools for improving the performance of solving. One of them is search annotations, which specify how to search so that a solution to the problem is found quicker. Another modelling technique is symmetry breaking, where different symmetric copies of the same solution are ruled out by adding symmetry breaking constraints. Redundant constraints are another possibility to improve the encodings [SMT20]. In further work we could use such tools for other encodings and analyze whether they improve the performance of the encodings.

It could also be interesting to compare our encodings with a purely SAT-based formulation and a state-of-the-art SAT solver. By doing this, one could investigate how much performance is lost by using a more user-friendly formalism.

Bibliography

- [BB73] Bertelè, Umberto; Brioschi, Francesco: On non-serial dynamic programming. *Journal of Combinatorial Theory, Series A*, 14(2):137–148, 1973.
- [Bo05] Bodlaender, Hans L.: Discovering Treewidth. In (Vojtáš, Peter; Bielíková, Mária; Charron-Bost, Bernadette; Sýkora, Ondrej, eds): *SOFSEM 2005: Theory and Practice of Computer Science*. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 1–16, 2005.
- [FM06] Freuder, Eugene C.; Mackworth, Alan K.: Constraint Satisfaction: An Emerging Paradigm. In: *Foundations of Artificial Intelligence*, volume 2, pp. 13–27. Elsevier, 2006.
- [RS84] Robertson, Neil; Seymour, P.D.: Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, February 1984.
- [RvBW06] Rossi, Francesca; van Beek, Peter; Walsh, Toby: Introduction. In: *Foundations of Artificial Intelligence*, volume 2, pp. 3–12. Elsevier, 2006.
- [Si05] Sinz, Carsten: Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In (van Beek, Peter, ed.): *Principles and Practice of Constraint Programming - CP 2005*. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 827–831, 2005.
- [SMT20] Stuckey, Peter J; Marriott, Kim; Tack, Guido: *MiniZinc Handbook*. p. 488, 2020.
- [SV09] Samer, Marko; Veith, Helmut: Encoding Treewidth into SAT. In (Kullmann, Oliver, ed.): *Theory and Applications of Satisfiability Testing - SAT 2009*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 45–50, 2009.
- [vdBB10] van den Broek, J.W.; Bodlaender, H.: , *TreewidthLIB - a benchmark for algorithms for treewidth and related graph problems*. Technical report. <http://www.staff.science.uu.nl/~bodla101/treewidthlib/>, 2010. Accessed: 2022-02-08.

Bisecting K-Prototypes: Effizientes hierarchisches Clustering gemischter Datensets

Hannes Dröse ¹

Abstract: Dieses Paper stellt ein hierarchisches Top-down-Clustering-Verfahren für gemischte Datensets mit linearer Laufzeit vor: Bisecting K-Prototypes. Der Algorithmus ist speziell für die Verarbeitung komplexer (numerischer und kategorialer) Datensets mit vielen fehlenden Werten geeignet. Dabei ist keine exzessive Vorverarbeitung des Datensets nötig. Zusätzlich werden Erweiterungen des Algorithmus vorgestellt, welche für die Verarbeitung von Multi-Select- und Freitext-Feldern (multi-kategoriale und String-Attribute) geeignet sind. Der Algorithmus wurde implementiert und gegen ein entsprechendes Datenset getestet und evaluiert.

Keywords: Clusteranalyse; Product-Information-Management; hierarchisches Clustering; Bisecting K-Means; K-Prototypes; gemischte Datensets; fehlende Werte

1 Einleitung & Motivation

Die Clusteranalyse ist ein Verfahren des maschinellen Lernens, bei dem in einem Datenset nach “Ähnlichkeitsgruppierungen” gesucht wird [13, Kap. 5.2.3 Unüberwachtes Lernen]. Die einzelnen Objekte eines Datensets werden dabei von einem Clustering-Algorithmus in Gruppen sortiert, “[. . .] sodass sich die Individuen innerhalb einer Gruppe auf eine Art und Weise ähnlich sind und unähnlich denen in anderen Gruppen” [9, Kap. 1.1 What Is a Cluster?]. Mit Hilfe dieser gefundenen Gruppen oder Cluster können im Anschluss neue Erkenntnisse und Anwendungen abgeleitet werden. [13, Kap. 5.2.3 Unüberwachtes Lernen]

Speziell im Bereich E-Commerce wurde die Clusteranalyse z.B. für die Entwicklung von Empfehlungsalgorithmen verwendet ([12], [3] und [11]). Ebenso gibt es eine interessante Arbeit von Kou und Lou [10], welche mithilfe von Clusteranalyse die Suchergebnisse einer Web-Suchmaschine verbesserten.

Inspiriert durch diese Ansätze beschäftigte sich der Autor nun mit der Frage, ob es sinnvolle Anwendungen für die Clusteranalyse im Zusammenhang mit Product-Information-Management-Systemen (PIM-Systemen) gibt. Dabei handelt es sich um spezielle IT-Systeme, welche i.d.R. im Zentrum der Architektur von E-Commerce-Unternehmen zu finden sind. Sie fungieren als zentraler Produktkatalog (Single Source of Truth) für die Verwaltung, Speicherung, Anreicherung und Aufbereitung von Produktdaten. Aus diesen PIM-Systemen erfolgt die Bereitstellung des Produktsortiments für andere Anwendungen wie ERP-Systeme,

¹ FH Erfurt, Angewandte Informatik, Altonaer Str. 25, 99085 Erfurt, Deutschland hannes.droese@fh-erfurt.de

die Bestellabwicklung oder das Marketing [14]. Mögliche Einsatzgebiete von Clustering könnten das automatische Kategorisieren von Produkten, Anomalie- und Duplikaterkennung, Anwendungen für die Produktempfehlung oder die Warenkorbanalyse sein.

Während der Recherche wurde aber deutlich, dass die Produktdaten in PIM-Systemen nicht für die klassischen Verfahren der Clusteranalyse geeignet sind. Typische Vertreter wie der K-Means-Algorithmus arbeiten nur mit Datensets, die aus numerischen Vektoren bestehen [5]. Produkte in PIM-Systemen weisen hingegen deutlich komplexere Strukturen auf. So gibt es Werte wie Single- und Multi-Select-Felder, Freitext-Attribute oder Produktbilder. Außerdem sind Produktdaten in der Praxis durchzogen mit fehlenden Werten (null-Values), was etablierten Clustering-Bibliotheken Probleme bereitet. [17]

Dadurch verschob sich der thematische Schwerpunkt: Ziel war es nun zuerst ein Clustering-Verfahren zu entwickeln, welches mit Produktdaten in PIM-Systemen umgehen kann und sinnvolle Cluster auf diesen generiert. Dabei entstand der *Bisecting K-Prototypes*. Im folgenden wird dieses Verfahren hergeleitet und seine Eigenschaften dargelegt. Anschließend erfolgt eine praktische Evaluation mit einem entsprechenden Datenset.

2 Clusteranalyse

2.1 Notation & Überblick

Die Objekte (nachfolgend *Datenpunkte* bzw. *Produkte* genannt), welche es zu clustern gilt, sind Vektoren. Jedes Element im Vektor steht für die Wertausprägung eines spezifischen Attributes (z.B. Preis, Titel, Gewicht etc.). Mittels Superskript werden spezifische Attribute eines Datenpunktes angesprochen. x^i bezeichnet also das i -te Attribut von x .

Die Produkte oder Datenpunkte sind Teil eines *Datensets* X . Hierbei handelt es sich um eine simple Menge dieser Datenpunkte $x \in X$. Mittels Subskript werden einzelne Punkte des Sets angesprochen (z.B. x_1 , x_2 oder x_i). Wenn nicht anders angegeben, gilt $n = |X|$.

Für die Einteilung des Datensets in Cluster muss die “Ähnlichkeit” zwischen den Datenpunkten quantifiziert werden. Dies erfolgt über die Berechnung der “Nähe” (engl. proximity) der Objekte zueinander. Dazu werden sog. Abstands- bzw. Distanzmaße verwendet. [8, Kap. 1.2 Types of Data and How to Handle Them]

Der Abstand zwischen zwei Datenpunkten wird hier mit $d(x_1, x_2)$ bezeichnet. Der kleinstmögliche Abstand ist 0, was absolute Deckungsgleichheit der Punkte signalisiert (daher ist $d(x_1, x_1) = 0$). Je größer der berechnete Abstand, desto unähnlicher sind sich die beiden Datenpunkte. Es existieren verschiedenste Abstandsmaße, welche für unterschiedliche Arten und Typen von Daten geeignet sind. Die Abstände ausschließlich numerischer Vektoren können z.B. mittels Maßen aus der Minkowski-Familie wie dem euklidischen Abstand oder dem Manhattan-Abstand berechnet werden. [8, Kap. 1.2 Types of Data and How to Handle Them]

2.2 Clustering-Verfahren

Mithilfe einer geeigneten Distanzfunktion generiert ein Clustering-Verfahren nun eine Cluster-Zuteilung des Datensets X . Es existieren vielfältige Verfahren. Die meisten lassen sich in partitionierende und hierarchische Verfahren unterteilen. [8, Kap. 1.3 Which Clustering Algorithm to Choose]

2.2.1 Partitionierende Verfahren

Partitionierende Verfahren teilen ein Datenset in eine vorher festgelegte Anzahl an Cluster (i.d.R. als k bezeichnet) und jeder Datenpunkt gehört zu genau einem der Cluster. [8, Kap. 1.3.1 Partitioning Methods]

Eines der bekanntesten Verfahren dieser Kategorie ist der *K-Means-Algorithmus*, welcher häufig als Inbegriff der Clusteranalyse selbst gilt [5]. Jedes Cluster wird in diesem Verfahren durch einen Mittelpunkt oder Centroid repräsentiert. Dieser Mittelpunkt ist kein tatsächlicher Punkt des Datensets [18]. Stattdessen berechnet sich jeder Attributwert aus dem Durchschnitt (engl. mean) der Werte der Cluster-Mitglieder für das jeweilige Attribut. [9, Kap. 4.5 K-Means Algorithm]

Steinbach et al. [18] beschreiben den grundsätzlichen Ablauf dieses Algorithmus wie folgt:

1. zufällige Wahl von k Punkten aus dem Datenset als initiale Mittelpunkte
2. Zuordnung aller Datenpunkte des Datensets zum jeweils nächstgelegenen Mittelpunkt
3. Neuberechnung aller Mittelpunkte aus den zugeordneten Datenpunkten
4. Wiederholung der Schritte 2 und 3; solange, bis sich die Cluster-Zuordnungen nicht mehr verändern oder ein Höchstlimit an Iterationen überschritten ist

Der große Vorteil dieser Verfahren ist eine lineare Laufzeit von $O(n)$ [5]. Nachteilig ist, dass die gesuchte Anzahl an Clustern vorher bekannt sein muss und jeder Datenpunkt nur genau einem Cluster zugeordnet sein kann. [8, Kap. 1.3.1 Partitioning Methods]

Um den K-Means auf gemischte (numerische und kategoriale) Datensets anwenden zu können, entwickelte Huang [5] den *K-Prototypes-Algorithmus*. Dieser arbeitet grundsätzlich wie der K-Means. Zur Berechnung der Cluster-Mittelpunkte wird aber für die kategorialen Attribute nicht der Durchschnitt, sondern der Modus – also der am häufigsten auftretende Wert – verwendet. Für die Berechnung des Abstands zwischen Mittel- und Datenpunkten wird eine kombinierte Abstandsfunktion genutzt:

$$d(x_1, x_2) = d_{num}(x_1^{num}, x_2^{num}) + w \cdot d_{cat}(x_1^{cat}, x_2^{cat}) \quad (1)$$

Die numerischen und kategorialen Werte werden also separat mit einer jeweils geeigneten Abstandsfunktion verarbeitet und die Ergebnisse addiert. Über den Faktor w lässt sich die Gewichtung beider Abstände zueinander anpassen. [5]

2.2.2 Hierarchische Verfahren

Hierarchische Verfahren produzieren eine ineinander verschachtelte Struktur von Clustern. Diese Cluster werden entweder nach dem Top-down- oder Bottom-up-Ansatz generiert [8, Kap. 1.3.2 Hierarchical Methods]:

- *Bottom-up- oder agglomerative Verfahren* starten mit jedem Datenpunkt in einem eigenen Cluster. Anschließend werden die beiden nächstgelegenen Cluster zu einem größeren kombiniert. Dieser Prozess wird solange wiederholt, bis im letzten Schritt die beiden verbliebenen Cluster zu einem großen verschmolzen werden, welches also alle Datenpunkte enthält.
- *Top-down- oder divisive Verfahren* arbeiten genau andersrum. Sie starten mit allen Datenpunkten in einem Cluster. Anschließend wird immer wieder das größte der verbleibenden Cluster in zwei kleinere aufgesplittet, bis schließlich jeder Datenpunkt seinem eigenen Cluster zugeordnet ist.

Durch dieses Vorgehen wird faktisch eine Cluster-Zuteilung für jede mögliche Anzahl an Clustern ($1 \leq k \leq n$) generiert. Jeder Datenpunkt gehört dadurch mehreren Clustern auf den unterschiedlichen Hierarchieebenen an, was umfangreichere Analysen der Cluster erlaubt [4]. Eine alternative Visualisierung der Ergebnisse besteht in Form einer Baumstruktur – einem sog. Dendrogramm. [18]

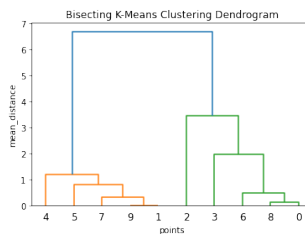


Abb. 1: Bsp. eines Dendrogramms

Der größte Nachteil dieser Verfahren ist die Laufzeit. Meistens liegt diese bei mindestens $O(n^2)$ bis teilweise $O(n^3)$. [4]

Deshalb entwickelten Steinbach et al. [18] den *Bisecting K-Means-Algorithmus*. Dabei handelt sich um ein Top-down-Clustering-Verfahren, welches für den Split eines größeren Clusters in zwei kleinere den K-Means mit $k = 2$ benutzt. Da nur das initiale Cluster alle

Datenpunkte enthält und mit den folgenden Splits die Cluster immer kleiner werden, ist die Laufzeit linear ($O(n)$).

3 Bisecting K-Prototypes

3.1 Herleitung

Für die beschriebenen möglichen Anwendungen der Clusteranalyse in PIM-Systemen sind hierarchische Verfahren zu bevorzugen, da die ineinander verschachtelte Cluster-Hierarchie eine umfassendere Analyse ermöglicht [4]. Außerdem entfällt eine vorherige Bestimmung der erwarteten Anzahl an Clustern, wie sie partitionierende Verfahren benötigen.

Gleichzeitig ist die quadratische oder noch schlechtere Laufzeit vieler hierarchischer Verfahren problematisch. Produktkataloge umfassen häufig mehrere tausend oder sogar einige Millionen Produkte, sodass eine möglichst geringe Laufzeit zu bevorzugen ist.

Der naheliegendste Ansatz besteht nun darin, den Bisecting K-Means-Algorithmus zu verwenden. Allerdings kann dieser nur numerische Vektoren verarbeiten. Grundsätzlich können kategoriale Werte in numerische umgewandelt werden (z.B. durch Nummerierung also Rot = 1, Grün = 2, Blau = 3). Solche Umwandlungen verzerren aber das Datenset (rote und blaue Produkte unterscheiden sich nun stärker voneinander als rote und grüne). [8, Kap. 1.2.6 Mixed Variables]

Schließlich wurde zur Lösung die Grundidee des Bisecting K-Means mit dem K-Prototypes-Algorithmus kombiniert. Das Clustering erfolgt also nach dem Ablauf des Bisecting K-Means, allerdings wird für den Zweier-Split eines großen Clusters in zwei kleineren stattdessen der K-Prototypes-Algorithmus verwendet. Das daraus resultierende Verfahren *Bisecting K-Prototypes* ist ein hierarchisches Top-down-Clustering-Verfahren für gemischte Datensets mit linearer Laufzeit.

3.2 Erweiterungen

3.2.1 fehlende Werte

Mit dem Bisecting K-Prototypes lassen sich bereits viele Attribute der Produkte in PIM-Systemen für das Clustering verwenden. Klassische Implementierungen von Clustering-Verfahren können allerdings nicht mit fehlenden Werten umgehen [siehe z.B. 17]. Füllt man diese mit einem Pseudowert (z.B. dem Mittelwert), so werden die Datensets wieder verzerrt. In PIM-Systemen kommen häufig Produkte aus verschiedenen Kategorien mit dadurch sehr unterschiedlichen Attributen vor. Dadurch weisen die meisten Produkte in den meisten Attributen keinen Wert auf. Eine gesonderte Betrachtung fehlender Werten ist also vonnöten.

Die folgende Formel zeigt die Distanzfunktion, welche für das Clustering verwendet wurde:

$$d(x_1, x_2) = \frac{\sum d'(x_1^i, x_2^i)}{|x_1^{\text{non null}} \cup x_2^{\text{non null}}|} \quad (2)$$

$$d'(x_1^i, x_2^i) = \begin{cases} 0 & , x_1^i \text{ is null} \wedge x_2^i \text{ is null} \\ 1 & , x_1^i \text{ is null} \vee x_2^i \text{ is null} \\ |x_1^i - x_2^i| & , i \text{ is numerical} \\ 0 & , i \text{ is categorical} \wedge x_1^i = x_2^i \\ 1 & , i \text{ is categorical} \wedge x_1^i \neq x_2^i \end{cases} \quad (3)$$

Zwei Produkte werden Attribut für Attribut miteinander verglichen. Dabei liegt der Abstand für jedes Paar von Attributwerten im Intervall $[0, 1]$. Die Abstände auf Basis der einzelnen Attribute werden anschließend summiert und durch die Menge an Attributen geteilt. Dadurch liegt der Abstand zweier Produkte insgesamt ebenfalls im Intervall $[0, 1]$.

Weisen beide Produkte in einem Attribut keinen Wert auf, so wird dieses sowohl über als auch unter dem Bruchstrich ignoriert. Fehlt der Wert nur bei einem der beiden Produkte, so wird der maximale Abstand von 1 addiert. Dieses Vorgehen ist durch den Jaccard-Koeffizienten inspiriert, welcher für asymmetrische binäre Attribute verwendet wird. Er ignoriert ebenfalls alle Attribute, bei denen beide Werte *false* sind. [siehe 8, Kap. 1.2.4 Binary Variables]

Die numerischen Attribute werden mittels Manhattan-Abstand verrechnet. Damit auch hier die Abstände auf Attributbasis im Intervall $[0, 1]$ liegen, müssen die Werte vorher auf ebendieses Intervall skaliert werden. Eine solche Normalisierung sorgt außerdem dafür, dass alle Attribute in etwa das gleiche Gewicht zueinander haben. [8, Kap. 1.2.1 Interval-Scaled Variables]

Die kategorialen Attribute werden nach dem Prinzip des Simple Matchings verarbeitet. Weisen beide Produkte den gleichen Wert auf, so wird 0 und im Falle von Ungleichheit 1 addiert.

3.2.2 Multi-kategoriale Attribute

Eine weitere Form von Attributen in PIM-Systemen sind sog. Multi-Select-Felder. Das sind Attribute, wo eine Liste an Auswahlmöglichkeiten definiert wird. Anschließend kann für ein Produkt ein oder mehrere dieser Optionen ausgewählt werden. Dies könnte z.B. die Farbe sein, da Produkte auch mehrfarbig sein können.

Diese Attribute könnten zum einen in einfache kategoriale Attribute umgewandelt werden, indem die gewählten Optionen sortiert und mittels Konkatination verbunden werden.

Dadurch würden sich aber zwei Produkte, die z.B. {Rot} und {Rot, Grün} sind, maximal voneinander unterscheiden.

Daher wurde ein alternativer Ansatz erdacht und die Distanzfunktion wie folgt erweitert:

$$d'(x_1^i, x_2^i) = \begin{cases} \dots \\ 1 - \frac{|x_1^i \cap x_2^i|}{|x_1^i \cup x_2^i|} \end{cases}, i \text{ is multi-categorical} \quad (4)$$

Diese multi-kategorialen Attribute werden mit einem “inversen Jaccard-Koeffizient” berechnet. Normalerweise wird der Jaccard-Koeffizient zur Bewertung der Ähnlichkeit zweier Produkte als ganzes verwendet. Hier wird für jedes multi-kategoriale Attribut der Jaccard-Koeffizient für die Werte des jeweiligen Attributes einzeln berechnet und aufaddiert.

3.2.3 String-Attribute

Schließlich gibt es noch Freitext-Felder in den Produktdaten wie z.B. den Produkttitel. Dies sind keine klassischen kategoriale Werte, da kaum ein Wert dem anderen gleicht. [15]

Zur Verarbeitung werden die Strings zunächst in Tokens zerlegt (Tokenization), ihre Endungen entfernt (Stemming) und schließlich Stop-Words entfernt (Stop-Word-Removal) – alles typische Verarbeitungsschritte aus dem Document Retrieval [2]. Aus einem Titel wie “Samsung Galaxy S20 128GB” wird nun {samsung, galaxi, s20, 128gb}.

Durch die Umwandlung in Tokens, sind die String-Attribute equivalent zu den multi-kategorialen Attributen und können mit dem gleichen Verfahren verarbeitet werden.

4 Experimentelle Überprüfung

4.1 Überblick

Für die praktische Evaluation ist Akeneo-PIM [1] (ein Open-Source PIM-System mit weiter Verbreitung) verwendet worden. Dieses System wurde mit Produkten aus dem sehr umfangreichen Online-Katalog Icecat [7] gefüllt, wo Hersteller aus der ganzen Welt ihre Produktdatenblätter für die Verteilung an Händler hochladen.

Anschließend wurde das hergeleitete Clustering-Verfahren implementiert. Nun wurde das Datenset geclustert, wobei verschiedene Kombinationen an Attributen für das Clustering verwendet worden sind. Diese Clustering-Ergebnisse wurden schließlich mittels einiger Metriken evaluiert.

4.2 Datenset

Es wurden 42 Smartphones der Samsung Galaxy S-Reihe importiert. Dabei waren 17 Smartphones der Generation S20, 21 der Generation S21 und 4 der Generation S22 vertreten. Es wurden sowohl Modelle der Produktausführungen Standard, Plus und Ultra sowie der sog. Fan Edition (FE) importiert. Die Smartphones zeichnen sich durch eine große Menge an verschiedenen Attributen der unterschiedlichsten Typen aus. Die folgende Tabelle gibt eine Übersicht dazu:

Tab. 1: Übersicht zu den Attributen der 42 Smartphones

Typ	Anzahl	Ø non-null	Ø unique	Beispiele
numerisch	56	21.8	3.7	Weight, Width, Depth, Height
kategorial	106	28.2	1.3	OS installed, SIM Card Type
multi-kat.	22	25.2	3.5	Product Color, 3G standards
String	11	28.4	13.5	Title, Description
<i>alle:</i>	<i>195</i>	<i>25.9</i>	<i>2.9</i>	

Die Spalte “Anzahl” zeigt, wie viele Attribute je Typ vorkommen. Die numerischen und kategorialen Attribute dominieren hierbei das Datenset. Die Spalte “Ø non-null” zeigt, wie viele der 42 Produkte im Schnitt in diesem Typ einen Wert aufweisen. Es kommen also fast so viele fehlende wie gefüllte Werte vor. Die Spalte “Ø unique” gibt an, wie viele verschiedene Wertausprägungen im Schnitt in der jeweiligen Attributart vorkommen. Die Strings weisen hierbei die höchste Menge an unterschiedlichen Werten auf, was in der Natur dieses Types liegt. Aufgrund des recht kleinen Datensets mit vielen ähnlichen Produkten, ist die Variation an Werten gering.

4.3 Evaluation & Metriken

Für die Bewertung des Clustering-Verfahrens sind verschiedene Metriken verwendet worden:

Die **Stabilität** zeigt die Übereinstimmung der Clustering-Ergebnisse von zehn verschiedenen Durchläufen des Bisecting K-Prototypes. Da das Verfahren mit zufälligen Startpunkten arbeitet, sind die Ergebnisse von zwei Durchläufen nicht immer deckungsgleich. Ein sinnvolles Clustering-Verfahren sollte aber dennoch einen gewissen Determinismus aufweisen. Die Übereinstimmung der Clusterings wurde mittels Adjusted-Rand-Index [6] berechnet. Er liefert Werte zwischen -1 und 1 und je höher er liegt, desto höher ist die Übereinstimmung zweier Cluster-Zuteilungen.

Die **Qualität** wird mittels des Silhouetten-Koeffizienten [16] gemessen. Er berechnet den durchschnittlichen Abstand der Datenpunkte zu den Punkten im selben Cluster im Verhältnis

zu den Punkten des unmittelbar benachbarten Clusters. Auch hier können Werte zwischen -1 und 1 entstehen. Je näher an 1 , desto besser sind die Cluster voneinander getrennt.

Mit der **Erkennung** wird geprüft, ob das Clustering-Verfahren die inhärenten Strukturen des Datensets erkennen kann. Die Smartphones stammen aus drei verschiedenen Generationen und es gibt elf verschiedene Modelle (unter Berücksichtigung der Generationen). Auf der Hierarchiestufe $k = 3$ sollten also die Smartphones der gleichen Generationen dem gleichen Cluster zugeordnet werden. Ebenso bei $k = 11$ für die Smartphone-Modelle. Die Übereinstimmung der berechneten Cluster-Zuteilung mit der erwarteten wurde mittels Adjusted-Rand-Index berechnet.

Schließlich befinden sich im Datenset sechs Paare an **Duplikaten** – also das gleiche Produkt unter verschiedenen Ids. Ein hierarchisches Clustering-Verfahren sollte die Duplikate stets in die gleichen Cluster sortieren und erst beim letzten Split die Duplikate schließlich voneinander trennen. Ist dies der Fall, so wird ein Duplikat als erfolgreich erkannt gezählt.

5 Auswertung

Die folgende Tabelle zeigt die Ergebnisse des Clustering für verschiedene (ausgewählte) Kombinationen an Attributen. Im Versuch “Strings” sind nur die String-Attribute für das Clustering benutzt worden – bei “num+kat” die numerischen und kategorialen Attribute. Alle weiteren Versuche (mit “+” beginnend) haben die numerischen und kategorialen um multi-kategoriale (mul) oder die String-Attribute (str) erweitert. Im Versuch “mul_k” sind die multi-kategorialen Attribute nicht nach dem hergeleiteten Verfahren verarbeitet, sondern in einfache kategoriale Attribute umgewandelt worden.

Tab. 2: Ergebnisse des Clustering

Versuch	Stabilität	Qualität	Erkennung				Duplikate
		\emptyset	$k = 3$	$k = 11$	$k = 3$	$k = 11$	
Strings	0.91	0.34	0.26	0.40	0.64	0.87	6 / 6
num+kat	0.98	0.43	0.31	0.60	0.65	0.71	5 / 6
+mul	0.98	0.41	0.37	0.57	0.16	0.70	5 / 6
+mul_k	0.97	0.40	0.30	0.53	0.52	0.68	5 / 6
+str	0.98	0.41	0.30	0.58	0.65	0.66	5 / 6
+mul+str	0.92	0.39	0.35	0.56	0.16	0.70	6 / 6

Die Stabilität und die Erkennung von Duplikaten erreicht für alle Versuche sehr hohe Werte.

Die Erkennung von Smartphone-Generationen ($k = 3$) ist mit Werten zwischen 0.16 und 0.65 eher mittelmäßig. Die Smartphone-Modelle ($k = 11$) erreichen höhere Werte zwischen

0.66 und 0.87. Abbildung 2 zeigt das Dendrogramm zum Clustering mit numerischen und kategorialen Attributen. Hier wird deutlich, dass die Trennung in die Generationen S20 und S21 sowie deren Modelle sehr gut funktioniert. Lediglich in der Gruppe mit den S22-ern (orange) finden sich auch Smartphones, die nicht in diese Generation gehören.

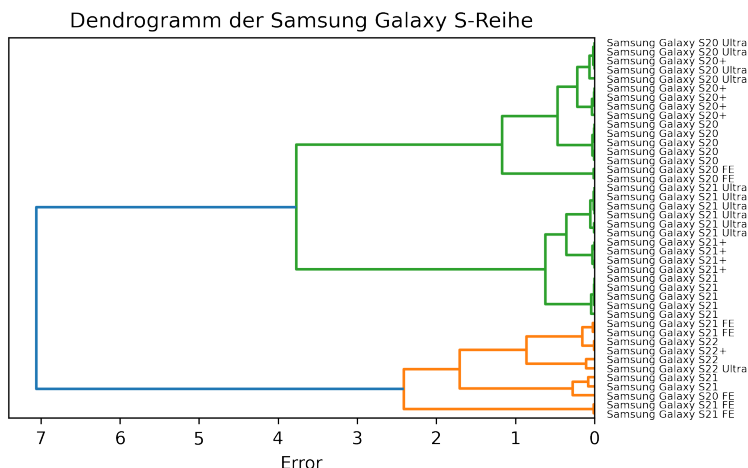


Abb. 2: Ergebnis des Clustering mit nur numerischen und kategorialen Attributen

Die Qualität ist einmal als Durchschnitt über alle Werte von k gegeben sowie für die beiden relevanten Hierarchiestufen. Sie liegt in allen Versuchen eher in mittelmäßigen Bereichen zwischen 0.26 und 0.60. Dies könnte daran liegen, dass sich die Produkte des Datensets insgesamt ziemlich ähnlich sind (alle vom gleichen Hersteller, recht geringe Anzahl an verschiedenen Wertausprägungen).

Auffällig ist, dass die ausschließliche Verwendung von numerischen und kategorialen Attributen auf fast allen Metriken die besten Werte liefert. Die Hinzunahme von multi-kategorialen und String-Attributen hingegen verschlechtert diese Ergebnisse wieder.

Die Verarbeitung der multi-kategorialen Attribute nach dem hergeleiteten Verfahren (+mul) bringt etwas bessere Stabilität und Qualität als die Umwandlung in einfache kategoriale Attribute (+mul_k). Die Erkennung der Generationen ist mit dem hergeleiteten Verfahren drastisch schlechter. Die Modelle werden dann wiederum etwa gleich gut erkannt. Insgesamt haben die multi-kategorialen Attribute keinen positiven Einfluss auf das Clustering. Eine klare Überlegenheit des hergeleiteten Verfahrens zur Umwandlung in einfache kategoriale Attribute ist ebenfalls nicht erkennbar.

Die String-Attribute mit den numerischen und kategorialen Attributen zu kombinieren verschlechtert die Clustering-Ergebnisse. Werden allerdings nur die String-Attribute verwendet, so bilden sich adäquate Cluster. Zwar ist die Qualität dieser Cluster im Vergleich sehr schlecht, aber vor allem die Erkennung und die Duplikate erreichen etwas bessere

Werte als die Kombination aus numerischen und kategorialen Attributen. Dies lässt sich dadurch erklären, dass z.B. im Produkttitel die wesentlichen Informationen des Produktes komprimiert – wenn auch unstrukturiert – hinterlegt sind.

6 Fazit und Ausblick

In diesem Paper wurde ein hierarchisches Clustering-Verfahren für gemischte Datensets mit linearer Laufzeit hergeleitet: *Bisecting K-Prototypes*. Das Verfahren wurde erweitert, um ohne Verzerrung der Daten mit fehlenden Werten umgehen zu können. In der anschließenden praktischen Evaluation zeigte sich, dass das Verfahren funktioniert und sinnvolle Cluster für ein Datenset mit recht komplexen Datenpunkten (Smartphones mit über 100 verschiedenen Attributen) finden kann.

Es ist außerdem eine Erweiterung des Verfahrens für sog. multi-kategoriale Attribute vorgestellt worden, welche sich ebenfalls für String-Attribute verwenden lässt. In der praktischen Evaluation hat die Hinzunahme dieser Attributtypen die Cluster allerdings nicht verbessert. Werden ausschließlich String-Attribute für das Clustering verwendet, so entstehen ähnlich gute Cluster wie mit ausschließlich numerischen und kategorialen Attributen. U.U. ist das hergeleitete Verfahren besonders für die Verarbeitung von Datensets geeignet, welche aus Freitext-Feldern bestehen. Hier könnte zukünftige Forschung anknüpfen.

Es ist wichtig zu beachten, dass das verwendete Datenset für die Evaluation sehr klein gewesen ist und eine sehr beschränkte Auswahl an Produkten enthalten hat. Alle Ergebnisse dieser Arbeit sind also bestenfalls als vorläufig anzusehen und sollten in Zukunft mit weiteren Datensets überprüft werden. Speziell sollte dabei geprüft werden, ob die multi-kategorialen Attribute in anderen Datensets einen positiven Einfluss auf das Clustering haben.

Weitere Versuche in der Zukunft könnten den Bisecting K-Prototypes mit alternativen Verfahren vergleichen. Im Speziellen wäre hier ein Vergleich der nötigen Vorverarbeitungen der Datensets für die verschiedenen Verfahren und deren Einfluss auf die Cluster-Ergebnisse interessant.

Erweisen sich diese Versuche als erfolgreich, so könnte mit dem Bisecting K-Prototypes ein solides, effizientes und flexibles Verfahren für das hierarchische Clustern einer Vielzahl von Datensets gefunden worden sein.

Literatur

- [1] Akeneo. *About Akeneo*. März 2022. URL: <https://www.akeneo.com/de/about-us/> (besucht am 04. 04. 2022).
- [2] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg u. a. „A Comparison of String Distance Metrics for Name-Matching Tasks“. In: *IJWeb*. Bd. 3. Citeseer. 2003, S. 73–78.

- [3] Yimin Cui. „Intelligent recommendation system based on mathematical modeling in personalized data mining“. In: *Mathematical Problems in Engineering* 2021 (2021).
- [4] Alican Dogan und Derya Birant. „K-centroid link: a novel hierarchical clustering linkage method“. In: *Applied Intelligence* 52.5 (2022), S. 5537–5560.
- [5] Zhexue Huang. „Extensions to the k-means algorithm for clustering large data sets with categorical values“. In: *Data mining and knowledge discovery* 2.3 (1998), S. 283–304.
- [6] Lawrence Hubert und Phipps Arabie. „Comparing partitions“. In: *Journal of classification* 2.1 (1985), S. 193–218.
- [7] Icecat. *About Icecat*. Nov. 2021. URL: <https://icecat.com/about-us/> (besucht am 05.04.2022).
- [8] Leonard Kaufman und Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Bd. 344. John Wiley & Sons, 2009.
- [9] Ronald S King. *Cluster analysis and data mining: An introduction*. Stylus Publishing, LLC, 2015.
- [10] Gang Kou und Chunwei Lou. „Multiple factor hierarchical clustering algorithm for large scale web page and search engine clickstream data“. In: *Annals of Operations Research* 197.1 (2012), S. 123–134.
- [11] Ravi Kumar u. a. „Recommendation systems: A probabilistic analysis“. In: *Journal of Computer and System Sciences* 63.1 (2001), S. 42–61.
- [12] Yoori Oh und Yoonhee Kim. „A resource recommendation method based on dynamic cluster analysis of application characteristics“. In: *Cluster Computing* 22.1 (2019), S. 175–184.
- [13] Stefan Papp u. a. *Handbuch Data Science: Mit Datenanalyse und Machine Learning Wert aus Daten generieren*. Carl Hanser Verlag GmbH Co KG, 2019.
- [14] Pimcore. *What, why and how of product information management*. Aug. 2021. URL: <https://pimcore.com/en/what-is-pim> (besucht am 04.04.2022).
- [15] N Rajalingam und K Ranjini. „Hierarchical Clustering Algorithm - A Comparative Study“. In: *International Journal of Computer Applications* 19.3 (2011), S. 42–46.
- [16] Peter J Rousseeuw. „Silhouettes: a graphical aid to the interpretation and validation of cluster analysis“. In: *Journal of computational and applied mathematics* 20 (1987), S. 53–65.
- [17] scikit-learn. 2.3. *Clustering – scikit-learn 1.0.2 documentation*. URL: <https://scikit-learn.org/stable/modules/clustering.html> (besucht am 14.04.2022).
- [18] Michael Steinbach, George Karypis und Vipin Kumar. „A Comparison of Document Clustering Techniques“. In: *Proceedings of the International KDD Workshop on Text Mining* (Juni 2000).

Natural Language Processing

Automatische Transformierung multilingualer Spracheingaben in Datenbankabfragen

Marcel Franzen¹

Abstract: Um die Nutzung von Datenbanken zu vereinfachen, werden im Rahmen dieser Arbeit SQL-Abfragen auf Basis einer Fragestellung und dem Datenbankschema in Form der Spaltennamen erzeugt. Hierzu werden mehrere Neuronale Netze eingesetzt, die einzelne Teile der SQL-Abfrage vorhersagen. Darüber hinaus werden multilingualen Worteinbettungen zur Repräsentation der Frage und Tabellenspalten verwendet. Durch die Nutzung der Worteinbettungen können auch Synonyme auf die Spaltennamen abgebildet werden. Dadurch ist die im Trainingsprozess verwendete Sprache irrelevant. Die Ergebnisse zeigen, dass das entstandene Modell sowohl tabellenunabhängig als auch sprachunabhängig funktioniert. Demnach erfordert die Nutzung einer Datenbank nur noch wenig Wissen über das Datenbankschema und insbesondere über die Sprache der Spaltennamen.

Keywords: Maschinelles Lernen; Neuronale Netze; Natural Language Processing; SQL-Abfragen; Multilingualität

1 Einleitung

In den vergangenen Jahrzehnten haben Menschen eine große Menge von Daten erzeugt, die gespeichert werden müssen, zumal die Menge der Daten in den kommenden Jahren weiter steigen wird [ID21]. Im Zuge dessen wird die Entwicklung moderner Datenbanksysteme als eine der wichtigsten Aspekte der Digitalisierung angesehen. In allen Bereichen von Kommunikation über Banken, Transportwesen, Versicherungen bis zu Universitäten, sind Datenbanken unverzichtbar [Ga17]. Auch wenn es den Nutzern häufig nicht bewusst ist, sind Datenbanken in beinahe jeder alltäglichen Aktivität involviert. Dementsprechend speichern Datenbanken eine große Menge heutiger Informationen und bilden die Basis für moderne Technologien.

Gleichzeitig führt die steigende Menge von Daten zu immer komplexer werdenden Datenbanken, deren Abfrage von einem Nutzer Kenntnisse über Datenbanksprachen wie SQL erfordert [ZXS17]. Dieser Umstand führt zu einer Barriere zwischen Nutzern ohne technische Kenntnisse und den Vorteilen, die eine Datenbank aufweist [PSC21]. Aus diesem Grund beschäftigt sich diese Arbeit mit der Erzeugung von SQL-Abfragen anhand von Fragen in natürlicher Sprache, die ein Nutzer zu einem unbekannten Datenbankschema stellen kann. Hierfür kommen Techniken des Maschinellen Lernens und der natürlichen Sprachverarbeitung zum Einsatz. Somit kann jeder Informationen von einer Datenbank abfragen, ohne die SQL-Abfrage selber formulieren zu müssen. Diese Arbeit greift dazu die

¹ Universität Bremen, Bibliotheksstraße 1, 28359 Bremen, Germany mfranzen@uni-bremen.de

bereits in Vielzahl vorhandenen Ansätze auf und entwickelt sie in den folgenden Aspekten weiter:

1. Bei der Speicherung von Nutzerdaten sind aufgrund der DSGVO Aspekte der Privatsphäre und Sicherheit zu berücksichtigen. Daher verfolgt diese Arbeit das Ziel der vollständigen Aufrechterhaltung der Privatsphäre. Folglich basiert der in dieser Arbeit vorgestellte Ansatz ausschließlich auf der Frage eines Nutzers in natürlicher Sprache und dem Datenbankschema in Form der Spaltennamen. Der eigentliche Inhalt einer Datenbank wird dabei nicht verwendet, um die SQL-Abfrage zu erzeugen.
2. Da die Datensätze zur Lösung dieser Aufgabe in Englisch vorhanden sind [ZXS17, Yu18], präsentiert diese Arbeit einen auf multilingualen Sprachmodellen basierenden Ansatz, um sowohl Englisch als auch Deutsch verwenden zu können. Im Zuge dessen wird untersucht, inwiefern eine sprachunabhängige Repräsentation der Eingabedaten zu sprachunabhängigen Ergebnissen führt.

Folglich können die Ergebnisse dieser Arbeit dazu verwendet werden, um jedem den Zugang zu Datenbanken und deren Nutzung zu ermöglichen.

2 Verwandte Arbeiten

In [ZXS17] wird der WikiSQL-Datensatz vorgestellt, welcher die Grundlage zur Lösung des Problems mit lediglich einer Tabelle darstellt. Dieser besteht aus insgesamt 80654 englischen Beispielen, welche eine hand-annotierte Sammlung von Fragen über den Inhalt von Datenbanken mit entsprechenden SQL-Abfragen bilden. Dabei sind verschiedenste Fragearten enthalten und die Länge der Fragen sowie der finalen SQL-Abfragen und die Anzahl der Spalten pro Tabelle variieren. Die SQL-Abfragen im Datensatz folgen stets dem selben Schema.

Die vorhandenen Ansätze zur Erzeugung von SQL-Abfragen anhand natürlicher Sprache können grundsätzlich in zwei verschiedene Kategorien unterteilt werden [Hu21]:

Erstens gibt es die erzeugungs-basierte Methode, die sich durch einen Sequenz-zu-Sequenz-Prozess auszeichnen. Bei diesem Ansatz wird die Fragestellung als Sequenz von Wörtern auf eine Sequenz abgebildet, welche die fertige SQL-Abfrage darstellt [SVL14]. Beispiele für die Nutzung der erzeugungs-basierten Methode sind [ZXS17, Su18, KDG17].

Zweitens gibt es eine skizzenbasierte Methode, welche die Erzeugung der SQL-Abfrage in Submodule unterteilt, die einzelne Aspekte der SQL-Abfragen wie zum Beispiel die zu selektierende Spalte erzeugen. Diese erreicht grundsätzlich bessere Ergebnisse auf dem WikiSQL-Datensatz [Hu21]. Beispiele hierfür sind [GG20, XLS17, He19]. Im Vergleich zu [ZXS17] nutzen die Modelle in [GG20] nicht ausschließlich nur die Frage und das

Tabellenschema in Form der Spaltennamen. Um das Modell zur Erzeugung der SQL-Abfragen mit weiteren Informationen bezüglich der Beziehungen zwischen Frage und Tabellenspalten anzureichern, stellt [GG20] zwei neue Algorithmen vor. Zum einen wird der Tabelleninhalt mit der Frage verglichen. So entsteht ein Vektor mit der Länge der Frage, in dem alle Wörter markiert sind, die als Inhalt einer Zelle der Tabelle gefunden werden konnten. Zum anderen wird ein weiterer Vektor mit der Länge des Tabellenkopfes erzeugt, in dem alle Spalten markiert sind, die in der Frage erwähnt werden. Weiter werden in diesem Vektor noch die Spalten markiert, deren Zellen im vorherigen Algorithmus in der Frage wiedergefunden werden konnten. Durch diese zusätzlichen Informationen werden bessere Ergebnisse als in bisherigen Arbeiten erzielt. Als Repräsentation der Frage und der Spaltennamen wird ein BERT-Modell [De19] verwendet.

Der Ansatz aus [PSC21] präsentiert ebenfalls einen skizzenbasierten Ansatz, welcher zusätzliche Informationen wie [GG20] einbindet, dabei jedoch den Inhalt der Tabelle vernachlässigt, um die Privatsphäre der Daten zu wahren. Anstatt die Repräsentation der Eingabewerte durch ein BERT-Modell umzusetzen, nutzt dieses Paper die Weiterentwicklung RoBERTa [Li19].

In dieser Arbeit wird ein skizzenbasierter Ansatz vorgestellt, bei dem ebenfalls zusätzliche Informationen wie in [GG20] verwendet werden, wobei die Algorithmen dazu auf [PSC21] basieren, da die Privatsphäre der Daten auch in dieser Arbeit berücksichtigt wird. Diese Algorithmen sind jedoch nicht in der Lage, Synonyme aus der Frage mit der Tabelle zu verknüpfen und schlagen darüber hinaus fehl, wenn die Frage in einer anderen Sprache (z.B. Deutsch) als die Spaltennamen der Tabelle (z.B. Englisch) vorliegt. Dementsprechend werden die bereits existierenden Ansätze in dieser Arbeit aufgegriffen, aber hinsichtlich einer multi-lingualen Anwendung ergänzt.

3 Methodik

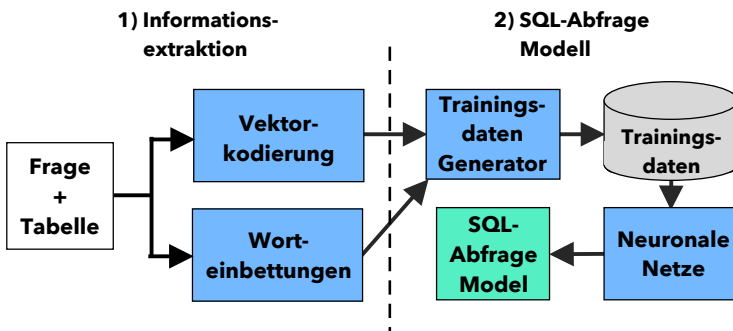


Abb. 1: Eine Übersicht über die Herangehensweise zur Erzeugung der SQL-Abfragen (Grafik inspiriert durch [MGD21]).

Die Herangehensweise in dieser Arbeit unterteilt sich in zwei Schritte: 1) Informationsextraktion und 2) ML-Model Generierung, welche in der Abbildung 1 dargestellt sind.

1. Informationsextraktion: Zunächst müssen die Einträge aus dem Datensatz zur weiteren Verarbeitung aufbereitet werden. Hierzu werden die Frage und das Tabellenschema in Form der Spaltennamen verwendet, um nützliche Informationen zu extrahieren.
2. SQL-Abfrage Modell: Anhand der Daten aus der Informationsextraktion wird ein finaler Datensatz gebildet, welcher von mehreren Neuronalen Netzen im Training genutzt wird. Aus diesen einzelnen Neuronalen Netzen wird dann das Gesamtmodell gebildet, welches SQL-Abfragen erzeugen kann.

3.1 Informationsextraktion

Zur Verwendung des Datensatzes müssen informationsreiche Merkmale extrahiert werden, in denen die Frage und die Spaltennamen enthalten sind, aber auch die Beziehungen untereinander erkenntlich sind. Hierzu wird der WikiSQL-Datensatz zunächst in eine numerische Repräsentation durch Worteinbettungen transformiert, welche als Eingabe für die Neuronalen Netze verwendet wird. Darüber hinaus werden weitere Vektoren aus dem Datensatz erzeugt, in denen zusätzliche Merkmale hinsichtlich der Beziehung zwischen Frage und Spalten enthalten sind. Nachfolgend werden die genannten Schritte näher erklärt:

3.1.1 Kodierung zusätzlicher Featurevektoren

Da der Fokus in dieser Arbeit unter anderem auf die Privatsphäre der Daten gerichtet ist, steht der Inhalt der Tabellen zur zusätzlichen Kodierung nicht zur Verfügung. Dadurch werden die Modelle generalisierter und unabhängig von Tabelleninhalten nutzbar. Im Folgenden werden zwei neue Algorithmen vorgestellt, die auf den Ideen aus [GG20] und [PSC21] aufbauen, aber hinsichtlich der Art des Vergleiches weiterentwickelt wurden.

Mit dem ersten Algorithmus wird ein Vektor erzeugt, in dem die Wörter der Frage mit dem Tabellenkopf verglichen werden. Hierzu wird die Frage vorher in einzelne Tokens unterteilt, wodurch sich jeder Index im Vektor auf einen Token der Frage bezieht. Wenn sich ein Token der Frage auf den Namen einer Spalte bezieht, werden die entsprechenden Indexe im Vektor markiert, die das gefundene Wort in der Frage repräsentieren. Die Unterteilung der Frage in Tokens wird durchgeführt, da die numerische Repräsentation in Kapitel 3.1.2 ebenfalls auf den einzelnen Tokens basiert. Für den Vektor wird eine One-Hot-Kodierung angewendet, bei der ein Index lediglich dann markiert wird, wenn der entsprechende Token in dem Namen einer Spalte des Tabellenkopfes wiedergefunden wird. Wie in [GG20] wird auch hier der erzeugte Vektor als *QV* (Question mark vector) bezeichnet.

Der oben angesprochene Unterschied bezüglich der Vergleichsmethode liegt darin, dass im ursprünglichen Paper [GG20] lediglich Wort für Wort verglichen wird. In diesem Algorithmus werden hingegen Worteinbettungen verwendet, die einerseits einen 1:1 Vergleich ermöglichen. Andererseits können so aber durch Synonyme auch ähnliche Wörter markiert werden. Hierzu wird die Konsinusähnlichkeit zwischen den Worteinbettungen ermittelt. Sollte diese bei über 50% Ähnlichkeit liegen, wird der entsprechende Index im Vektor markiert. Dadurch kann sich die Frage von dem gegebenen Tabellenschema stärker unterscheiden, als es in bisherigen Arbeiten [GG20, PSC21] möglich ist. Da der Nutzer oftmals kein Wissen über das zugrunde liegende Tabellenschema und den Namen der Spalten hat, wird diese Hürde durch die Worteinbettungen gelöst.

Des Weiteren kann die Nutzung muti-lingualer Worteinbettungen dazu genutzt werden, dass sich die Sprache der Frage von der, der Spaltennamen unterscheiden kann und die Markierung trotzdem möglich ist. Eine Voraussetzung dafür ist das Vorhandensein angeglicherter Worteinbettungen, in denen gleiche Wörter in unterschiedlichen Sprachen nah beieinander liegen, wie sie durch [fa17] bereitgestellt werden. Der zweite Algorithmus erzeugt einen Vektor, in dem alle Spalten markiert sind, die sich auf die Frage beziehen. Dieser wird fortan *HV* genannt (Table header mark vector).

In der Tabelle 1 werden exemplarisch zwei Beispiele gezeigt, anhand derer die Funktionsweise der oben ausgeführten Algorithmen verdeutlicht wird. Im ersten Beispiel bezieht sich das Wort „notes“ in der Frage auf die gleichnamige Spalte der Tabelle. Aus diesem Grund sind sowohl der entsprechende Index im QV als auch HV-Vektor mit einer 1 markiert.

Frage:	Tell me what the notes are for south australia?
Tabellenkopf:	[„territory“, „background colour“, „slogan“, „series“, „notes“]
QV:	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
HV:	[0, 0, 0, 0, 1]
Frage:	In welchem Büro sitzt Christopher Metz?
Tabellenkopf:	[„office“, „employee“, „mail“, „telephone“]
QV:	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
HV:	[1, 0, 0, 0, 0]

Tab. 1: Zwei Beispiele die zeigen, wie die QV -und HV Vektoren anhand einer Frage und einem Tabellenkopf erzeugt werden.

3.1.2 Transformierung in eine latente Repräsentation durch Worteinbettungen

Zur Verarbeitung von natürlicher Sprache ist eine latente Repräsentation als Vektoren notwendig, die durch Worteinbettungen erreicht werden kann. Wie bereits in 3.1.1 erwähnt, werden in dieser Arbeit diesbezüglich die multilingualen Worteinbettungen des Facebook MUSE Projektes verwendet, die auf fastText [Bo16] basieren und für ein Wort, einen Vektor mit 300 Werten zur Verfügung stellen.

Zunächst wird der Text durch das Sprachverarbeitungsframework spaCy [o.oJ] in die einzelnen Tokens zerlegt, wobei diese in Kleinbuchstaben umgewandelt werden und neben dem originalen Token noch die Grundform (Lemma) betrachtet wird. Für jeden dieser Tokens wird dann geprüft, ob es in dem WordEmbeddingsmodell den passenden Vektor gibt. Falls nicht, wird geprüft, ob es einen Vektor für die Grundform des Tokens gibt. Jeder passende Vektor wird anschließend in eine Liste von Vektoren eingefügt, woraus die latente Repräsentation des gesamten Textes mit den Dimensionen $(n, 300)$ gebildet wird, wobei n der Anzahl der Token entspricht. Sollte ein Token nicht in den WordEmbeddings enthalten sein, wird der Token gegen einen speziellen Token *UNK* ersetzt, welcher aus 300 Nullen besteht.

Durch diese Methode werden die Frage und die Spaltennamen des Tabellenkopfes für jeden Eintrag im Datensatz transformiert. Die Spaltennamen werden zu diesem Zweck zu einem Text konkateniert und jeweils mit dem Token `</s>` getrennt. Da diese multilingualen WordEmbeddings nur für eine endliche Menge von Wörtern existieren, sollten die Spaltennamen in der Regel aus Wörtern bestehen, für die es WordEmbeddings gibt. Andernfalls würde die Repräsentation der Spaltennamen überwiegend aus Nullen bestehen und somit den Informationsgehalt stark reduzieren. Dies gilt ebenfalls für die Fragen, wobei diese oftmals länger sind als die Spaltennamen und somit mehr Wörter enthalten, wodurch die unbekannten Wörter einen geringeren Einfluss auf die gesamte Repräsentation haben.

3.2 SQL-Abfrage Modell

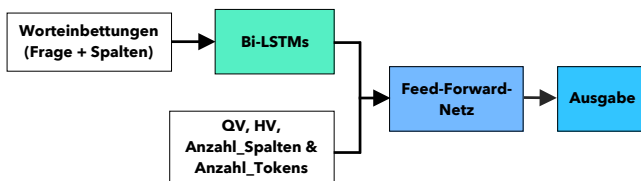


Abb. 2: Die Referenzarchitektur zur Vorhersage einzelner Aspekte einer SQL-Abfrage.

Die Erzeugung der SQL-Abfragen in der zugrundeliegenden Bachelorarbeit, auf der dieses Paper basiert, wird nur für einen konkreten Anwendungsfall benötigt. So wird die Komplexität des Problems stark reduziert und die Modelle müssen nur für kleine Tabellen funktionieren.

Als Grundlage zur Modellierung des Problems in dieser Arbeit dient [GG20], worin die Beschaffenheit des WikiSQL-Datensatzes genutzt wird, um das Problem der Abfrage-Erzeugung in kleinere Teil-Probleme zu zerlegen, da die SQL-Abfragen stets dem selben Schema folgen. Dadurch wird die Komplexität des Problems verringert, weshalb der Entwurf der Neuronalen Netze leichter und effizienter ist. Mehrere Modelle werden so trainiert, dass sie die einzelnen Attribute der Trainingsdaten vorhersagen können, um letztlich die einzelnen Lücken in der Abfrage zu füllen. Die Abbildung 2 zeigt die Referenzarchitektur dieser Arbeit, auf der die einzelnen Modelle mit geringfügigen Änderungen aufbauen.

Die bereits erwähnten Worteinbettungen werden genutzt, um für jeden Token der Frage sowie der Spalten einen latenten Vektor zu erzeugen. Darüber hinaus werden die zusätzlichen QV- und HV Vektoren aus Kapitel 3.1.1 als weitere Eingabewerte verwendet sowie die Anzahl der Fragetokens und die Anzahl der Spalten im Tabellenkopf.

Die Worteinbettungen werden zu einem Vektor konkateniert und durch zwei bidirektionale LSTMs (*Bi-LSTMs*) verarbeitet, damit der gesamte Kontext der Sequenz erfasst werden kann. Durch die gemeinsame Verarbeitung von Frage und Tabelle soll das Modell ein tieferes Verständnis für den Zusammenhang zwischen beiden Komponenten erlangen. Eine alternative Architektur, in der die Frage und die Tabellenspalten zunächst durch getrennte Bi-LSTMs verarbeitet werden, wurde ebenfalls in Betracht gezogen, aber aufgrund nicht signifikant besserer Ergebnisse wieder verworfen.

Zusammen mit den anderen Eingabedaten wird die Ausgabe der Bi-LSTMs anschließend von einem einfachen Feed-Forward-Netz verarbeitet. Die gewünschte Ausgabe unterscheidet sich hierbei jeweils hinsichtlich des jeweiligen Aspektes der SQL-Abfragen, den das Modell erzeugen soll:

SELECT-Spalte (sel): In dieser Arbeit werden nur Beispiele aus dem Datensatz verwendet, die maximal fünf Spalten in der Tabelle enthalten. Aus diesem Grund werden zur Vorhersage der zu selektierenden Spalte fünf Ausgabeneuronen benötigt. Folglich wird das Problem der zu selektierenden Spalte als einfache Klassifikation betrachtet.

Aggregatfunktion (agg): Analog zur Vorhersage der SELECT-Spalte, wird zur Vorhersage der Aggregatfunktion ebenfalls eine Klassifikation verwendet, die sechs mögliche Klassen enthält (MAX, MIN, AVG, SUM, COUNT & NONE).

Anzahl WHERE-Klauseln: Die SQL-Abfragen in dieser Arbeit enthalten stets entweder eine oder maximal zwei WHERE-Klauseln, weshalb das Problem binär betrachtet werden kann und lediglich ein Ausgabeneuron notwendig ist.

WHERE-Spalten: Wenn zwei WHERE-Klauseln benötigt werden, ist die Reihenfolge dieser irrelevant. Aus diesem Grund wird für jede WHERE-Klausel ein eigenes Trainingsbeispiel erzeugt, welches im One-Hot-Vektor der gewünschten Ausgabe die entsprechende Spalte markiert. Die Ausgabe des Modells erzeugt folglich keine Wahrscheinlichkeitsverteilung per Softmax-Funktion, sondern wendet die Sigmoid-Funktion für jedes Neuron einzeln an. Falls nur eine WHERE-Klausel gesucht wird, kann der größte Wert in der Ausgabe betrachtet werden, deren Index die Spalte der WHERE-Klausel angibt. Wenn zwei WHERE-Klauseln gesucht werden, werden die zwei größten Werte mit deren Indexen betrachtet. Im Gegensatz zur oben vorgestellten Architektur ist die Anzahl der WHERE-Klauseln ein zusätzlicher Eingabewert des Modells.

WHERE-Werte: Die Vergleichswerte in den WHERE-Klauseln sind Ausschnitte aus den ursprünglichen Fragen. Aus diesem Grund werden zur Vorhersage der Vergleichswerte in den WHERE-Klauseln zwei getrennte Modelle verwendet. Ersteres bestimmt den Anfangstoken aus der Frage und das zweite Modell den Endtoken aus der Frage. Die Vorhersage der Vergleichswerte ist neben den üblichen Eingabewerten noch zusätzlich von der Spalte in der WHERE-Klausel abhängig, wodurch die Modelle sowohl für ein als auch für zwei WHERE-Klauseln genutzt werden können. Der Operator in der WHERE-Klausel wird in dieser Arbeit nicht betrachtet, da nur Beispiele aus dem Datensatz entnommen wurden, die den Gleichheits-Operator enthalten.

Jedes dieser Modelle wird verwendet, um einen bestimmten Aspekt der SQL-Abfrage zu erzeugen. Folglich wird die finale SQL-Abfrage anhand der Ausgaben aller einzelnen Modelle zusammengesetzt, wie es in der Abbildung 3 dargestellt ist.

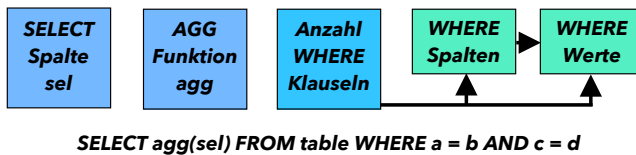


Abb. 3: Die einzelnen Modelle zur Vorhersage verschiedener Aspekte einer SQL-Abfrage und deren Zusammensetzung.

4 Ergebnisse

Die Modelle sind aufgrund der Datenprivatsphäre generalisiert und können grundsätzlich für jede Tabelle und jede Frage genutzt werden, solange sie nicht größer als die im Training verwendeten Datenbanken sind. Nichtsdestotrotz führt die Privatsphäre der Daten und die daraus folgenden Nichtberücksichtigung der Tabelleninhalte auch dazu, dass wichtige Informationen in den HV -und QV Vektoren nicht enthalten sind, von denen die Modelle profitieren könnten. Dennoch wurden, wie in der Tabelle 2 zu sehen, vielversprechende Ergebnisse erreicht:

In der Tabelle 2 sind die Ergebnisse der einzelnen Modelle auf einem Testdatensatz von WikiSQL zusammengefasst. Mit einer 70.48% Genauigkeit für die Vorhersage der zu selektierenden Spalte sowie 84.37% für die Aggregatfunktion funktionieren diese Aspekte der SQL-Abfrage besonders gut. Auch die Ergebnisse für die Anzahl der WHERE-Klauseln sind mit 88.86% gut, obwohl insgesamt mehr Beispiele für nur eine WHERE-Klausel im Testdatensatz enthalten sind und die Genauigkeit durch diese Unausgewogenheit im Datensatz beeinflusst wird. Mit einer Genauigkeit von 70% funktioniert die Vorhersage der Vergleichswerte ebenfalls gut. Bei der Verwendung realer Daten werden die Ergebnisse für die Vergleichswerte der WHERE-Klauseln schlechter ausfallen, da hierfür die Spalte der

WHERE-Klausel als Eingabe verwendet wird. Da das Modell hierfür jedoch lediglich eine Genauigkeit von 46.83% aufweist, liefert es überwiegend falsche Ergebnisse, welche sich negativ auf die darauffolgenden Modelle auswirken.

Aufgabe	Genauigkeit auf dem Testdatensatz
SELECT-Spalte	70.48%
Aggregatfunktion	84.37%
Anzahl der WHERE-Klauseln	88.86%
SELECT-Spalte der WHERE-Klauseln	46.83%
Anfangstoken des Vergleichswerts in den WHERE-Klauseln	70.76%
Endtoken des Vergleichswerts in den WHERE-Klauseln	70.01%

Tab. 2: Die Ergebnisse der Modelle auf dem Testdatensatz.

Die Experimente haben gezeigt, dass die zusätzlichen Vektoren bezüglich der Verknüpfung zwischen der Frage und den Tabellenspalten eine starke Gewichtung in der Vorhersage der Spalten haben und folglich eine sinnvolle Erweiterung der Eingabedaten sind. Darüber hinaus wurde gezeigt, dass die Nutzung von Worteinbettungen für die zusätzlichen Vektoren die Fähigkeit, die Frage mit dem Tabellenschema zu verknüpfen, verstärken, da so auch Synonyme erkannt werden können und der Nutzer nicht zwingend Kenntnisse über die Tabelle benötigt. Außerdem ermöglicht dies, dass der Nutzer die Frage in mehr Variationen stellen kann und dennoch gleiche Ergebnisse erhält.

1:

Frage:	What is Terrence Ross nationality?				
Tabelle:	player	nationality	position	years in toronto	team
Zielausgabe:	SELECT nationality FROM table WHERE player='terrence ross'				
Ausgabe:	SELECT nationality FROM table WHERE player='terrence ross'				

2:

Frage:	What is the average length of Lucifer?				
Tabelle:	series	episode	title	directed by	written by
Zielausgabe:	SELECT AVG(length) FROM table WHERE series = 'lucifer'				
Ausgabe:	SELECT AVG(length) FROM table WHERE series = 'of lucifer'				

Tab. 3: Ergebnisse anhand deutscher- und englischer Fragen / Spaltennamen (1).

Bei den Aggregatfunktionen ist aufgefallen, dass manchmal falsche Aggregatfunktionen vorhergesagt werden, wenn die Eingabesprache eine andere ist, als die im Training verwendete. So konnte das Neuronale Netz den Zusammenhang zwischen bestimmten Worteinbettungen und den richtigen Aggregatfunktionen erlernen wie zum Beispiel *how many* und *COUNT*. Da die Worteinbettungen zwar stark angeglichen sind, aber dennoch Abweichungen haben, hatten sie zwischen Deutsch und Englisch teilweise eine zu geringe Ähnlichkeit, damit das Modell auch hier den Zusammenhang zu den Aggregatfunktionen herstellen konnte. Die Tabellen 3 und 4 zeigen einige Ergebnisse anhand verschiedener Kombinationen aus deutschen und englischen Fragen bzw. Spaltennamen. Folglich ist die Herangehensweise

mit multilingualen Sprachmodellen einerseits zielführend, andererseits werden noch stärker angegliche Sprachmodelle die Ergebnisse weiter verbessern.

3:

Frage:	Was ist Terrence Ross Nationalität?				
Tabelle:	spieler	nationalität	position	jahre in toronto	verein
Zielausgabe:	SELECT nationalität FROM table WHERE spieler = 'terrence ross'				
Ausgabe:	SELECT nationalität FROM table WHERE spieler = 'terrence ross'				

4:

Frage:	Was ist die durchschnittliche Länge von Lucifer?				
Tabelle:	series	episode	title	directed by	length
Zielausgabe:	SELECT AVG(length) FROM table WHERE series = 'lucifer'				
Ausgabe:	SELECT AVG(length) FROM table WHERE length = 'länge von lucifer'				

Tab. 4: Ergebnisse anhand deutscher- und englischer Fragen / Spaltennamen (2).

5 Fazit und Ausblick

Diese Arbeit beschäftigt sich mit der Erzeugung von SQL-Abfragen anhand einer gegebenen Frage und einer Tabelle. Hierzu wurde der WikiSQL Datensatz verwendet und hinsichtlich eines vereinfachten Trainings optimiert, sodass die Komplexität des Problems reduziert werden konnte. Der Anspruch der Datenprivatsphäre konnte erfüllt werden und folglich werden die Ergebnisse unabhängig vom Inhalt der Tabelle erzielt, wodurch das Modell insgesamt deutlich generalisierter ist. Darüber hinaus wurde in dieser Arbeit untersucht, inwiefern multilinguale Sprachmodelle dazu genutzt werden können, um die Ergebnisse Neuroner Netze bei Problemen der natürlichen Sprachverarbeitung unabhängig von Sprachen zu erzielen. Diesbezüglich haben die Experimente gezeigt, dass dies möglich ist und dabei oftmals identische Ergebnisse erzielt werden können. Insbesondere konnten verschiedene Sprachkombinationen getestet werden, welche sehr ähnliche Ergebnisse lieferten. Da die Modelle für die Spalten einen starken Fokus auf die zusätzlichen Vektoren legen, funktioniert die Vorhersage der Spalten multilingual besonders gut. Dies ist darin begründet, dass mit Hilfe der angeglichenen Worteinbettungen oftmals die selben Eingabevektoren über verschiedene Sprachen hinweg erzeugt werden können. Noch stärker angegliche Worteinbettungen werden potenziell zu noch besseren Ergebnissen führen. Die selben Modelle können für unterschiedliche Eingabesprachen verwendet werden, obwohl die Trainingsdaten lediglich in Englisch vorlagen.

Zusammengefasst konnten alle Ziele der Arbeit erreicht werden und es ist ein Modell zur Erzeugung von SQL-Abfragen entstanden, welches einerseits unabhängig vom Inhalt einer Tabelle und andererseits unabhängig von der Eingabesprache funktioniert. Die vorgestellten Algorithmen und die Nutzung der angeglichenen Worteinbettungen als Repräsentation der Eingabedaten kann potenziell auf andere Sprachen ausgeweitet werden, ohne einen signifikanten Verlust der Zuverlässigkeit zu haben. Die Stärken und Schwächen des Ansatzes

konnten herausgearbeitet werden und insbesondere wurde versucht, die noch vorhandenen Schwierigkeiten zu erklären.

In einer hierauf aufbauenden Arbeit kann versucht werden, die Ergebnisse weiter zu verbessern. Folgende Ansätze könnten dabei verfolgt werden:

- **Bessere Sprachmodelle:** Die Verwendung besserer Sprachmodelle kann dazu genutzt werden, um eine allgemeinere Repräsentation der Frage und Tabellenspalte zu erreichen. Die Nutzung der Worteinbettungen ist unabhängig vom Kontext der Frage und entsprechend sehr abhängig vom genauen Wortlaut. Andere Arbeiten zu diesem Thema schlagen die Nutzung von BERT vor, welches nicht immer dieselben Vektoren für Wörter zuweist, sondern dies abhängig vom Gesamtkontext macht. Insbesondere könnte dadurch das Problem gelöst werden, dass die Worteinbettungen in verschiedenen Sprachen teilweise nicht ähnlich genug waren, um multilingual dieselben Ergebnisse zu erzielen.
- **Mehr Informationen über den Zusammenhang von Frage und Tabelle:** Die Vernachlässigung der Tabelleninhalte hatte positive und negative Auswirkungen. Die Algorithmen könnten hier dahingehend verbessert werden, dass die Wortart der Tokens berücksichtigt wird, sodass beispielsweise eine Ortsangabe mit einer dafür vorgesehenen Spalte verknüpft wird.

6 Danksagung

Dieses Paper basiert auf einer Bachelorarbeit, welche an der Universität Bremen in der Arbeitsgruppe Rechnerarchitektur von Prof. Dr. Rolf Drechsler entstand. Diese Arbeit wurde durch das Data Science Center der Universität Bremen (DSC@UB) unterstützt. Ein besonderer Dank gilt Christopher Metz, welcher als Betreuer der Bachelorarbeit fungierte.

Literaturverzeichnis

- [Bo16] Bojanowski, Piotr; Grave, Edouard; Joulin, Armand; Mikolov, Tomáš: Enriching Word Vectors with Subword Information. CoRR, abs/1607.04606, 2016.
- [De19] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina: , BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.
- [fa17] facebookresearch/MUSE, Juli 2017. Abgerufen: 14.07.2021.
- [Ga17] Gabriel, CRISTIAN Mihai: The Importance Of Databases In Economy - Some General Coordinates. Revista Economica, 69(4):92–98, November 2017.
- [GG20] Guo, Tong; Gao, Huilin: , Content Enhanced BERT-based Text-to-SQL Generation, 2020.

- [He19] He, Pengcheng; Mao, Yi; Chakrabarti, Kaushik; Chen, Weizhu: X-SQL: reinforce schema representation with context. CoRR, abs/1908.08113, 2019.
- [Hu21] Hui, Binyuan; Shi, Xiang; Geng, Ruiying; Li, Binhua; Li, Yongbin; Sun, Jian; Zhu, Xiaodan: , Improving Text-to-SQL with Schema Dependency Learning, 2021.
- [ID21] IDC, Statista: , Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025 (in zettabytes). <https://www.statista.com/statistics/871513/worldwide-data-created/>, 2021. Abgerufen: 10.07.2022.
- [KDG17] Krishnamurthy, Jayant; Dasigi, Pradeep; Gardner, Matt: Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Copenhagen, Denmark, S. 1516–1526, September 2017.
- [Li19] Liu, Yinhan; Ott, Myle; Goyal, Naman; Du, Jingfei; Joshi, Mandar; Chen, Danqi; Levy, Omer; Lewis, Mike; Zettlemoyer, Luke; Stoyanov, Veselin: RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR, abs/1907.11692, 2019.
- [MGD21] Metz, Christopher A.; Goli, Mehran; Drechsler, Rolf: Early Power Estimation of CUDA-Based CNNs on GPGPUs: Work-in-Progress. CODES/ISSS '21, Association for Computing Machinery, New York, NY, USA, S. 29–30, 2021.
- [o.oJ] o.A.: , spaCy: Industrial-Strength Natural Language Processing, o.J. Abgerufen: 07.10.2021.
- [PSC21] Pal, Debaditya; Sharma, Harsh; Chaudhari, Kaustubh: , Data Agnostic RoBERTa-based Natural Language to SQL Query Generation, 2021.
- [Su18] Sun, Yibo; Tang, Duyu; Duan, Nan; Ji, Jianshu; Cao, Guihong; Feng, Xiaocheng; Qin, Bing; Liu, Ting; Zhou, Ming: Semantic Parsing with Syntax- and Table-Aware SQL Generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Melbourne, Australia, S. 361–372, Juli 2018.
- [SVL14] Sutskever, Ilya; Vinyals, Oriol; Le, Quoc V.: , Sequence to Sequence Learning with Neural Networks, 2014.
- [XLS17] Xu, Xiaojun; Liu, Chang; Song, Dawn: SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. CoRR, abs/1711.04436, 2017.
- [Yu18] Yu, Tao; Zhang, Rui; Yang, Kai; Yasunaga, Michihiro; Wang, Dongxu; Li, Zifan; Ma, James; Li, Irene; Yao, Qingning; Roman, Shanelle; Zhang, Zilin; Radev, Dragomir R.: Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. CoRR, abs/1809.08887, 2018.
- [ZXS17] Zhong, Victor; Xiong, Caiming; Socher, Richard: Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. CoRR, abs/1709.00103, 2017.

Comparing Link Grammars and Dependency Grammars for parsing German histological reports

Julian Dörenberg¹

Abstract: The availability of structured data is becoming an increasingly critical factor in medical research. Still, pathologists in Germany document their findings in running text instead of in a structured form. In order to obtain structured data from these report texts, they have to be converted to a more useful form. Link Grammars (LGs) and Dependency Grammars (DGs) both can be used to parse the texts. Hence, LGs and DGs can be used for information extraction on histological reports. This paper aims to compare LGs and DGs, to show why DGs are superior and to evaluate the performance of a DG parser on a corpus of 200 histological reports randomly selected from breast biopsy reports. The DG parser achieved an Unlabelled Attachment Score of 96, a Labelled Accuracy of 95 and a Labelled Attachment Score of 93. Further evaluation shows that the occurrence of medical words which have not been part of the training data does not affect the parsers performance.

Keywords: Natural Language Processing, Text-Mining, AI, Dependency Grammars, Link Grammars, Medical Informatics

1 Introduction

In modern medical research, the availability of structured data is becoming increasingly critical. Data from clinical practice generated from histological reports are of particular interest for cancer research. Although synoptic reporting is on the rise in European medical research [Ka07], pathologists in Germany document most of their findings in running text rather than in a structured form. In order to make the information within these reports available for research, it is critical to convert them into a structured form. One way to perform this information extraction is to use a grammar for a natural language – German here –, to parse the sentences in the reports and to use a downstream application to extract the required information. This paper deals with the comparison of two such grammars, both of which can be obtained by making use of machine learning techniques. After training them, grammars for natural languages typically are used to parse a sentence into a graph. This graph consists of the words within the sentence – its nodes – and grammatical relations between two words in the sentence – its edges. Due to this second aspect, these graphs are also called (*grammatical*) *relation graphs*.

¹ RWTH Aachen University and University Hospital Aachen,
julian.doerenberg@rwth-aachen.de or jdoerenberg@ukaachen.de

The first grammar type to be discussed here are Link Grammars (LGs). In an LG there is a dictionary that maps each word that is featured in the grammar to a list of possible grammatical contexts that the word can appear in. As these grammars can be denoted in propositional logic in disjunctive normal form, such a grammatical context is called a *disjunct*. Each disjunct contains so-called *connectors*, which denote the option to establish a grammatical relation to a different word within the same sentence.

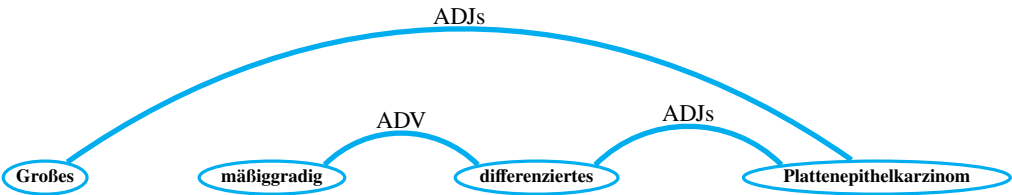
The second grammar type to be discussed here are Dependency Grammars (DGs). DGs also follow the idea that there are grammatical relations between the words, but they differ from LGs in two properties. Firstly, LGs consider the word class – for instance noun or verb –, whereas DGs concentrate on the role a word plays within a sentence – for instance if a noun is the sentence's subject or an accusative object. Secondly, the relation graphs generated by a DG are always trees. For LGs, this condition generally does not hold. They allow cycles and – in general – undirected edges.

An example sentence annotated according to both grammar types is given in Figure 1. In the example, the – partial – sentence *Großes mäßiggradig differenziertes Plattenepithelkarzinom* is annotated by using both grammar formalisms. The most important difference is that the relation graphs of DGs are trees whereas there is no hierarchy in LG relation graphs. This results in the property that LGs allow cycles in their relation graphs while DGs do not. The example illustrates two further differences. Firstly, LGs also establish relations to the final dot of a sentence whereas DGs use a root node to model the end of a sentence. Secondly, the tags of the relations denote different linguistic aspects. Discussing these is out of scope here. Intuitively, the tags in DGs denote the grammatical role the word in the child node has in a sentence – and hence takes into account the whole sentence to determine the tag –, while the tag in the LG focuses on the kind of the relationship between the two words which are related to each other.

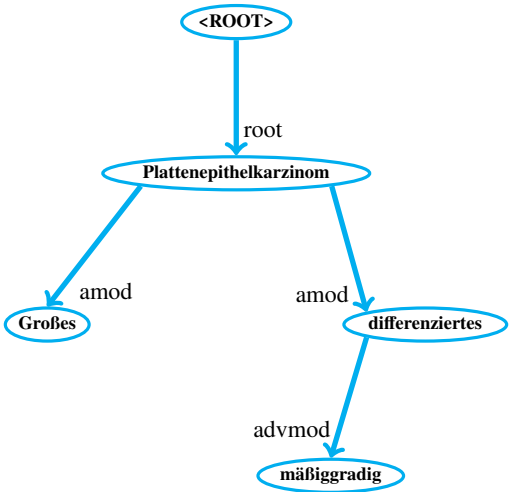
2 Related work

LG were first described in 1996 by Daniel Sleator and Davy Temperley in the context of the English language [ST95]. For linguistic reasons that are outside of the framework of this paper, adaptations to the formalism had to be made in order to make them useable for the German language. This was done by Sandra Kübler in 1998 [Kü02]. For instance, she introduced changes to the connectors, which resulted in the grammatical relation graph being directed. Since then, LGs have been used in multiple information extraction projects on English medical reports. Most of them have since remained in an experimental state and have never been used in real world applications [Zh06]. For the context of the German language, there is no further usage of LGs other than the work of Kübler to the best of my knowledge.

Link Grammar:



Dependency Grammar:



Tab. 1: The Figure shows how the sentence *Großes mäßiggradig differenziertes Plattenepithelkarzinom* is annotated in the Link Grammar formalism – the upper part – and in the Dependency Grammar formalism – the lower part. Each word of the sentence is represented as one node of a relation graph. The edges of the graphs denote the occurrence of a grammatical relations between the words. DGs form trees whereas LGs form non-hierarchical relation graphs.

DGs have their foundation in 1959 when Lucien Tesnieres’ work was published posthumously [Te76]. In order to facilitate a modern framework for working with DGs, Christopher Manning, Mihai Surdeanu et al. implemented the Stanford Parser as part of the Stanford CoreNLP pipeline in 2014 [CM14; Ma14]. In 2017, Christopher Manning published a paper together with Timothy Dozat where they trained a recurrent neural network (RNN) to parse sentences by using the DG formalism [DM17; Pa19]. They chose their training

data from the Universal Dependencies project [Ma21; Mc13]. Afterwards, they evaluated the model on data from the Universal Dependencies project in different languages. The performance for German was satisfying for use in real world application [DM17]. Dozat and Mannings' neural parser is available for usage with the framework supar, which was developed by Yu Zhang and published in 2018 [Zh18].

Since the first description of DGs, experiments in parsing German medical reports have been done. For instance, Elif Kara, Tatjaa Zeen et al. trained a DG parser on nephrological reports and evaluated its performance by using the Stanford parser [Ka18]. Unfortunately, the parsers performance was too poor to use it in a real world application. Currently, there is a tool in development which utilizes a DG parser based on Dozat and Mannings' work to extract information from histological reports [Dö22]. Figure 1 shows the pipeline executed by this tool.

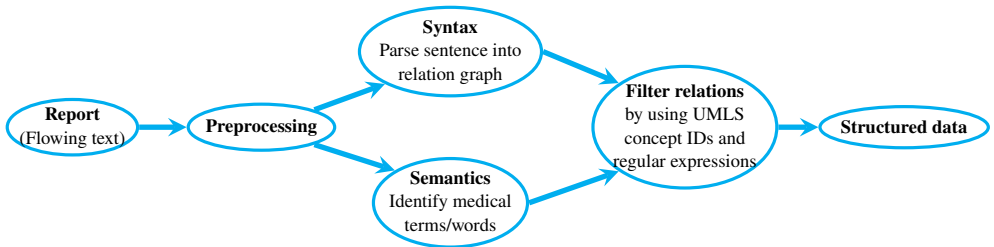


Abb. 1: The pipeline that extracts relations from German histological reports.

After parsing each sentence of the histological report into relation graphs, the tool generates relations of higher arity by attaching relations to each other. For instance, the two relations $(6.5\text{cm}, \text{gro\ss})$ and $(\text{gro\ss}, \text{Karzinom})$ are combined to $(6.5\text{cm}, \text{gro\ss}, \text{Karzinom})$. Finally, these relations are filtered by using regular expressions and the ontology database UMLS [Bo04] and exported to a csv table. Alternatively to UMLS other ontology databases such as SNOMED CT [St01] can be used. The tool has been evaluated on a data set featuring hepatocellular carcinoma [Dö22]. On this small evaluation it exhibits good performance and is able to extract 98% of the requested information correctly [Dö22]. Table 2 shows their evaluation data.

As given in the table, there is a single error caused by the Dependency Grammar parser. The information *inflammation degree* is not found. This is caused by the – partial – sentence *mit milder entzündlicher Aktivität und portaler sowie septenbildender Fibrose mit Architekturstörung (Grad 2, Stadium 3 nach Desmet)*. As it is syntactically ambiguous whether *Grad 2* refers to *entzündlicher Aktivität*, to *Fibrose* or to *Architekturstörung*, the parser cannot be expected to parse this construct correctly.

# HCCs	Fibrosis	Vascular invasion	Tumor diameter	Inflammation	Inflammation degree	Distance to reSection area	Desmet stage	Steatosis	Cirrhosis
1			1,4cm			1mm			TRUE
1		TRUE	5,5cm			0,3cm			FALSE
1	FALSE		4,2cm	FALSE		0,3cm			FALSE
1	TRUE	TRUE	8,5cm	TRUE	—		3		
1			16cm			0,1cm			
1	TRUE	TRUE	4,2cm	TRUE		1,5mm		TRUE	FALSE
1									
1		FALSE	9,5cm			1cm			
1		FALSE	8,5cm	TRUE				TRUE	
1	TRUE		3,6cm			0,2cm	1-2		

Tab. 2: From [Dö22]. The Table shows a dataset extracted by the tool developed in this thesis. Each row represents one report. The column names are information defined to be relevant for researching HCC by the clinic of surgery within the University Hospital Aachen. Information printed in black were extracted correctly. The data can have the two data types boolean, integer or measurement value with respective unit. The Information in the column *Inflammation degree* was extracted wrong, which is denoted by the red dash.

3 Methods

The comparison between LGs and DGs will be purely argumentative. However, the performance of Dozats and Mannings DG parser will be evaluated by following an idea of Carlos Gomez-Rodriguez, Iago Alonso-Alonso and David Vilares [GAV19]. In particular, they recommend to evaluate the performance of a DG parser in two parts.

Firstly, three standard metrics shall be used. For a test corpus, let n_{words} be the number of words in the corpus, n_{rels} be the number of words that correctly have been attached to their parent in the relation graph tree correctly and n_{tags} be the number of words that have been tagged correctly by the parser. Finally, let n_{rels_tags} be the number of words that have been attached to their parent in the relation graph tree and have been tagged correctly. Then the three standard metrics are computed as follows: Firstly, The Unlabelled Attachment Score (UAS) is defined as $\frac{n_{rels}}{n_{words}}$. It denotes the proportion of correctly attached grammatical relations to the words. Secondly, the Labelled Accuracy (LA) is defined as $\frac{n_{tags}}{n_{words}}$. It denotes the proportion of correctly tagged words. Thirdly, the Labelled Attachment Score (LAS) is defined as $\frac{n_{rels_tags}}{n_{words}}$. It denotes the proportion of words being tagged and attached to their parent in the relation graph tree correctly. By definition, LAS combines UAS and LA and it holds $uas \geq las$ and $la \geq las$.

Secondly, the parsing performance shall be evaluated based on the particular use case of the DG parser. In case of the tool presented in Section 2, this is the task of information extraction performed via the pipeline given in Figure 1 where relations of higher arity are generated by attaching relations to each other [Dö22]. Accordingly, this part of the evaluation will be performed by generating the 2-, 3-, and 4-ary relations by using the downstream application. In order to investigate the effect of medical words that were not included in the training data the resulting set of relations will then be filtered for

those relations, that contain at least one medical word. The proportion of correctly extracted relations then is used as the evaluation metric.

4 Results

In order to compare LGs and DGs, it is critical to value the linguistic motivation behind both grammar types and investigate properties which firstly affect their parsing performance and secondly their usability.

Both types of grammar model grammatical relations between words. Whereas relation graphs in LGs are non-hierarchical, DGs force their relation graphs to being trees. The problem about non-hierarchical relation graphs is that they allow for cycles, although this is not supported by linguistics. Accordingly, the motivation behind LGs is worse than the one behind DGs, because LGs allow relation graphs which are not supported by linguistics, whereas in DGs this is impossible.

After discussing the linguistic motivation of both grammar types, properties which firstly affect their parsing performance and secondly their usability are compared between LGs and DGs. Table 3 shows and compares the properties of LGs and DGs which affect their performance and usability.

Property	LGs	DGs
Lexicalized	Yes	No
Adaptations for German required	Yes	No
Public Parser for German available	No	Yes
Public training data available	No	Yes
Neural parser available	No	Yes

Tab. 3: Comparison of properties of LGs and DGs. The entry whether one of the grammar types has a property is printed boldly if this grants an advantage over the other grammar type. Kübler implemented an LG parser for German, but due to DG parsers being superior in his evaluation, it has never been published.

The first aspect to be discussed here is lexicalisation. LGs are an example of lexicalized grammars that means that the grammar consists of a dictionary which contains each word known to the grammar. A sentence containing a word that is not part of the grammar cannot be parsed. In opposition to this, DGs are not lexicalised, because they can make use of word embeddings which support unknown words [Bo17]. This makes them superior to LGs, because they are able to handle unknown words. Additionally, typos cannot be recognized by LGs, because the misspelled version of the word is not part of the grammar's dictionary. DGs can bypass this issue by handling the misspelled word as an unknown word.

The second aspect that affects the grammar's parsing performance is the need for specific adaptations of the formalism in order to support the German language. For LGs, Kübler argued that German has a freer word order within the sentences than English. According to its motivation to support the English language, the original formalism strongly preferred creating grammatical relations between words with minimal distance. Hence, Kübler described the changes to the formalism required for the German language [Kü02]. DGs already have been designed based on the idea to parse multilingual texts. Hence, no adaptations are required for their usage in German. This has the advantage that there is no need for an adapted implementation of the parser. As described in Section 2, there are already numerous DG parsers available. For LGs however, this does not hold. Kübler implemented an adapted parsers, but due to the lexicalisation of LGs and her further research on DGs, it has not been published.

Another property affecting the usability of the grammars is the availability of public training data. The Universal Dependencies project has been founded with the aim to annotate corpora with the DG formalism and publish them. Accordingly, numerous corpora have been annotated and published in a number of languages. For instance, there are German corpora based on Twitter data, Google reviews and a number of newspaper articles [In; Mc13]. The data are stored in the CoNLL-U format which simplifies combining multiple corpora [St]. In opposition to this, no training data for German LGs are available to the public to the best of my knowledge.

Data in the CoNLL-U format can simply be used as input data for the training of neural nets. As described in Section 2, modern DG parser implementations are based on neural networks. When training them, it is possible to rely on the advancements the training of neural nets and the availability of the required hardware have made in the past decades. Hence, training can be executed in a very efficient way without the need to implement particular training algorithms. In opposition, no neural parser is available for LGs. In her work, Kübler, however, described a training algorithm that she also implemented. But due to her further work with DGs it has not been published.

In conclusion, DGs are superior to LGs. Hence, it cannot be recommended to use LGs in a real world application. DGs offer a wide range of advantages over LGs and should be preferred.

Accordingly, solely DG parsers are evaluated in this paper. This is done based on the RNN implemented and trained by Dozat and Manning [DM17] which is used in its pretrained version provided by supar [Zh18]. Table 4 shows the evaluation results for the metrics described in Section 2.

Metric	Full corpus [%]	Without localisations [%]
UAS	94	96
LA	92	95
LAS	90	93
2-ary relations	95	97
3-ary relations	91	93
4-ary relations	88	89

Tab. 4: From [Dö22]. Evaluation of the DG Parser trained by Dozat and Manning [DM17]. The metrics described in Section 2 have been computed for two corpora. The corpus including the localisation sentences – left column – consists of 200 sentences, whereas the corpus excluding them – right column – contains 165 sentences.

The model was trained on a number of corpora from the Universal Dependencies project. These include Bulgarian, Catalan, Czech, German, English, Spanish, French, Italian, Dutch, Norwegian, Romanian/Moldavian¹ and Russian corpora [DM17; Mc13]. The evaluation data, however, were explicitly annotated to evaluate Dozat and Manning’s model. 200 sentences were randomly selected from breast biopsy reports written by two senior pathologists. Eventually, two sub-corpora of these were evaluated. The left column in Table 4 shows the evaluation results for the full corpus. The right column shows the evaluation results for the same corpus, when sentences on tumor localisation within the affected organ had been removed. Usually, histological reports start with such a sentence – for instance with *links oben außen* – but they do not occur in the further report. From a linguistic point of view, these sentences do not have an unambiguous parsing. In order to not affect the parsing performance, they were removed from the corpus. Afterwards, 165 sentences remained. Overall, the parser showed very good performance on the corpus. In particular, for the most important metric – the UAS – it achieved a score of 96. When restricting the UAS to discard each relation that does not contain a medical word, the score for the 2-ary-relations is obtained. This was computed as 97. This demonstrates that the occurrence of medical words does not affect the parsing performance negatively, although these words have neither been included in the training data, nor in the dictionary of the words embeddings.

Medical words sometimes occur as Multi Word Expressions (MWEs), such as *Carcinoma in situ* for instance. Semantics of MWEs are determined from the combination of the words they contain. In the given example *Carcinoma* does not have a semantics on its own in German. It is the Latin word for the German word *Karzinom*. The whole expression *Carcinoma in situ* however has a semantics in German. It is reasonable to assume that medical MWEs reduce the parsing performance, because neither the whole expression nor the separate medical words are featured in the training data. Hence, it is sensible to set up the hypothesis that the DG parser will not parse them correctly. This hypothesis has turned

¹ Using the ISO 639-1 code it remains ambiguous whether the data include sentences in Romanian, Moldavian or both and Dozat and Manning did not elaborate on that in their paper.

out to be true: In the few MWEs that occur in the 200 breast biopsy sentences, neither of the MWEs has been parsed correctly. In each case either the grammatical relation or its type was assigned incorrectly to at least one of the separate words. Unfortunately, the corpus is too small to allow for a proper analysis of the errors and to find a pattern behind the parsing errors.

5 Conclusion

LGs have shown to be inferior to DGs on a conceptual level for two reasons. Firstly, they are lexicalized and hence can neither handle unknown words nor typos. DGs resolve this problem by supporting word embeddings [DM17; Pa19; Zh18]. Complementing this issue, the set of linguistic properties modelled in LGs is incomplete: The underlying formalism allows for cycles in the relation graph, although this is impossible from a linguistic point of view. This applies to both, the German as well as the English language. In conclusion, it is not recommended to use LGs for any kind of application, but it is preferable to use DGs instead.

In order to investigate multiple DG parsers, the framework *supar* can be used [Zh18]. One of these parsers pretrained by Dozat and Manning [DM17] shows good results in parsing histological reports on the chosen corpus of 200 sentences randomly selected from breast biopsy reports. However, the corpus is quite small, thereby limiting the evaluation of the parsing performance on histological reports. Hence, more data will be annotated and the evaluation of the extension is ongoing. Currently, there is evidence that the DG parser lacks in precision when parsing sentences which contain MWEs. Hence, the training data corpus of Dozat and Manning will be extended by adding sentences that contain medical MWEs. Afterwards, Dozats and Mannings training will be repeated on the enlarged corpus. The goal behind this is to increase the performance of the DG parser when parsing MWEs and identify the reason why they cause the parser to parse them incorrectly.

Nevertheless, it has been shown that Dependency Grammar parsers can already be helpful in their current state of development, even without above modifications [Dö22]. But if the advances described above are successful and the DG parser's performance will be verified on a larger evaluation data corpus, DGs will eventually significantly advance the parsing of medical reports.

Literatur

- [Bo04] Bodenreider, O.: The Unified Medical Language System (UMLS): integrating biomedical terminology. en, *Nucleic Acids Research* 32/90001, S. 267D–270, Jan. 2004, ISSN: 1362-4962, URL: <https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkh061>, Stand: 09. 11. 2021.
- [Bo17] Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T.: Enriching Word Vectors with Subword Information. arXiv:1607.04606 [cs]/, arXiv: 1607.04606, Juni 2017, URL: <http://arxiv.org/abs/1607.04606>, Stand: 15. 11. 2021.
- [CM14] Chen, D.; Manning, C. D.: A fast and accurate dependency parser using neural networks. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. S. 740–750, 2014.
- [DM17] Dozat, T.; Manning, C. D.: Deep Biaffine Attention for Neural Dependency Parsing. arXiv:1611.01734 [cs]/, März 2017, URL: <http://arxiv.org/abs/1611.01734>, Stand: 28. 10. 2021.
- [Dö22] Dörenberg, J.: Extraction of HCC-related data from histological reports by using a Dependency Grammar, English, Presentation at 105. Jahrestagung der Deutschen Gesellschaft für Pathologie, Münster, Germany, Juni 2022, URL: https://www.cbmb.ukaachen.de/wp-content/uploads/2022/06/DGP2022_Praesentation_Doerenbergetal.pdf.
- [GAV19] Gómez-Rodríguez, C.; Alonso-Alonso, I.; Vilares, D.: How important is syntactic parsing accuracy? An empirical evaluation on rule-based sentiment analysis. en, *Artificial Intelligence Review* 52/3, S. 2081–2097, Okt. 2019, ISSN: 0269-2821, 1573-7462, URL: <http://link.springer.com/10.1007/s10462-017-9584-0>, Stand: 28. 10. 2021.
- [In] Ines Rehbein and Josef Ruppenhofer and Bich-Ngoc Do: tweeDe – A Universal Dependencies treebank for German tweets, URL: <https://aclanthology.org/W19-7811.pdf>.
- [Ka07] Karim, R. Z.; Van Den Berg, K. S.; Colman, M. H.; McCarthy, S. W.; Thompson, J. F.; Scolyer, R. A.: The advantage of using a synoptic pathology report format for cutaneous melanoma: Synoptic pathology reports in melanoma. en, *Histopathology* 52/2, S. 130–138, Dez. 2007, ISSN: 03090167, URL: <http://doi.wiley.com/10.1111/j.1365-2559.2007.02921.x>, Stand: 02. 05. 2021.
- [Ka18] Kara, E.; Zeen, T.; Gabryszak, A.; Budde, K.; Schmidt, D.; Roller, R.: A Domain-adapted Dependency Parser for German Clinical Text. In: *A Domain-adapted Dependency Parser for German Clinical Text*. Verlag der Österreichischen Akademie der Wissenschaften, 0xc1aa5576_0x003a12bd, 2018, ISBN: 978-3-7001-8437-9, URL: <https://hw.oeaw.ac.at/8437-9>, Stand: 24. 01. 2022.
- [Kü02] Kübler, S.: *Learning a Lexicalized Grammar for German*. *New Methods in Language Processing and Computational Natural Language Learning*/, 2002.

- [Ma14] Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J. R.; Bethard, S.; McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. S. 55–60, 2014.
- [Ma21] de Marneffe, M.-C.; Manning, C. D.; Nivre, J.; Zeman, D.: Universal Dependencies. en, Computational Linguistics/, <https://universaldependencies.org/>, S. 1–54, Mai 2021, ISSN: 0891-2017, 1530-9312, URL: https://direct.mit.edu/coli/article/doi/10.1162/coli_a_00402/98516/Universal-Dependencies, Stand: 14. 11. 2021.
- [Mc13] McDonald, R.; Nivre, J.; Quirmbach-Brundage, Y.; Goldberg, Y.; Das, D.; Ganchev, K.; Hall, K.; Petrov, S.; Zhang, H.; Täckström, O. et al.: Universal dependency annotation for multilingual parsing. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). S. 92–97, 2013.
- [Pa19] Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In (Wallach, H.; Larochelle, H.; Beygelzimer, A.; Alché-Buc, F. d.; Fox, E.; Garnett, R., Hrsg.): Advances in Neural Information Processing Systems. Bd. 32, Curran Associates, Inc., 2019, URL: <https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf>.
- [St] Stenstrom, E.: CoNLL-U, URL: <https://github.com/EmilStenstrom/conllu>.
- [St01] Stearns, M. Q.; Price, C.; Spackman, K. A.; Wang, A. Y.: SNOMED clinical terms: overview of the development process and project status. eng, Proceedings. AMIA Symposium/, S. 662–666, 2001, ISSN: 1531-605X.
- [ST95] Sleator, D.; Temperley, D.: Parsing English with a Link Grammar. CoRR abs/cmp-lg/9508004/, 1995.
- [Te76] Tesnière, L.: Éléments de syntaxe structurale. Klincksieck, Paris, 1976, ISBN: 978-2-252-01861-3.
- [Zh06] Zhou, X.; Han, H.; Chankai, I.; Prestrud, A.; Brooks, A.: Approaches to text mining for clinical medical records. In: Proceedings of the 2006 ACM symposium on Applied computing - SAC '06. ACM Press, Dijon, France, S. 235, 2006, ISBN: 978-1-59593-108-5, URL: <http://portal.acm.org/citation.cfm?doid=1141277.1141330>, Stand: 01.05. 2021.
- [Zh18] Zhang, Y.: Supar, 2018, URL: <https://github.com/yzhangcs/parser>.

Autor:innenverzeichnis

B

Bender, Nicolas, 73

Berner, Lukas, 11

D

Dirks, Jona, 103

Dörenberg, Julian, 153

Dröse, Hannes, 127

F

Franzen, Marcel, 141

G

Gerhard, Enna, 103

H

Hanoun, Osama, 87

K

Kremer, Robin, 61

R

Rall, Philipp, 73

Rentz, Bjarne Valentin, 35

S

Sander, Jurek, 23

Schmidt, Moritz, 49

V

Voboril, Florentina, 115

Voß, Clara, 93