Oktolineare Darstellungen von Metrokarten aus GTFS-Daten mit disjunkten Tarifzonen mittels Mixed Integer Linear Programming

Bachelorarbeit

Cedric Lippke
456900

15. November 2024

Betreuer: Prof. Dr. Benjamin Blankertz
Prof. Dr. Marc Alexa

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Institut für Softwaretechnik und Theoretische Informatik
Fachgebiet Neurotechnologie

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit eigenständig ohne Hilfe Dritter und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe. Alle Stellen die den benutzten Quellen und Hilfsmitteln unverändert oder sinngemäß entnommen sind, habe ich als solche kenntlich gemacht.
Sofern generative KI-Tools verwendet wurden, habe ich Produktnamen, Hersteller, die jeweils verwendete Softwareversion und die jeweiligen Einsatzzwecke (z.B. sprachliche Überprüfung und Verbesserung der Texte, systematische Recherche) benannt. Ich verantworte die Auswahl, die Übernahme und sämtliche Ergebnisse des von mir verwendeten KI-generierten Outputs vollumfänglich selbst.
Die Satzung zur Sicherung guter wissenschaftlicher Praxis an der TU Berlin vom 8. März 2017. `https://www.static.tu.berlin/fileadmin/www/10000060/FSC/Promotion___Habilitation/Dokumente/Grundsaetze_gute_wissenschaftliche_Praxis_2017.pdf` habe ich zur Kenntnis genommen.
Ich erkläre weiterhin, dass ich die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Berlin, den 15. November 2024

Cedric Lippke

Digital unterschrieben von Cedric Lippke
Datum: 2024.11.15 22:22:22 +01'00'

Kurzfassung

Existierende Ansätze zum Berechnen von Metro Karten benötigen eigens dafür aufbereitete Daten und zeigen nichts neben den Linien an. In dieser Arbeit wird das weit verbreitete Datenformat GTFS als Grundlage für die Berechnung genommen und Tarifzonendaten in die Berechnung mit einbezogen. Die GTFS-Daten werden mit SQL gefiltert und anschließend zu einem Graph zusammengefügt, um dann mit einem Ansatz von Nöllenburg und Wolff mittels Mixed Integer Linear Programming zu einer Grafik zu kommen. Es wird ein Weg beschrieben, mit dem der für den Ansatz von Nöllenburg und Wolff benötigte Graph aufgebaut werden kann und im Fall von Tarifzonen in Form von konzentrischen Kreisen auch diese in den Graphen integriert werden können. Die Probleme sind für die einfache Version des Ansatzes von Nöllenburg und Wolff zu komplex und kommen nicht zu optimalen Lösungen. Weitere Forschung wird benötigt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Es gibt unzählige Transitsysteme auf der Welt von einer Buslinie, die ein paar Dörfer miteinander verbindet, hin zu einem komplexen Metrosystem, wie es in Tokyo gefunden werden kann. Um sich insbesondere in komplexen Netzwerken zurechtzufinden, werden Karten erstellt. Es gibt grob zwei Kartenarten: Geographische Karten, über die die Linien darübergelegt werden, und topologische Karten, bei denen die Linien und ihre Umstiege in den Fokus rücken. Ein berühmtes Beispiel für geographische Karten findet sich in New York[3] (Abbildung 1.1). Für topologische Karten wird gerne London[5] als Beispiel genommen, da man dort den Ursprung für diese Kartenart ausgemacht hat. 1933 wird hier die erste Karte dieser Art benutzt, designt von Henry C. Beck[8] (Abbildung 1.2). Doch ist diese nicht die erste ihrer Art. So gibt es eine Karte des Berliner S-Bahnnetzes aus dem Jahr 1931 in ähnlichem Stil[6, 1].
Geographische Karten können oft einfacher zu erstellen sein, da die meisten Elemente eindeutig sind. Es muss lediglich darauf geachtet werden, dass alle Haltestellen lesbar beschriftet werden können. Topologische Karten hingegen haben kein eindeutiges Aussehen, das aus dem Netz notwendigerweise hervorgeht. In dieser Arbeit werden für das Design einer topologische Karte die Regeln von M. Nöllenburg und A. Wolff verwendet[9].
Aufgrund des höheren Arbeitsaufwandes beim Erstellen einer topologischen Karte, trifft man sie häufig nur in größeren Systemen mit großer Nutzerschaft an. Eine Möglichkeit dies zu ändern könnte die automatische Generierung dieser sein.
In dieser Arbeit soll der Ansatz von M. Nöllenburg und A. Wolff ergänzt werden. Die Generierung soll auf Basis von GTFS Daten (siehe Kapitel 2.1) erfolgen, welche bei vielen Transportunternehmen bereits vorliegen. Des weiteren sollen Tarifzonen in die Darstellung aufgenommen werden. Dabei sollen in dieser Arbeit lediglich konzentrische ringförmige Tarifzonen betrachtet werden.
Der Ansatz von M. Nöllenburg und A. Wolff wurde bereits in mehreren Formen weiterentwickelt, doch weder um Tarifzonen ergänzt noch direkt aus GTFS-Daten abgeleitet. Es wurden stattdessen diverse Optimierungen untersucht. Untersucht wurde zum einen eine Optimierung für sehr große Netzwerke[10] und daneben ein auf Gittern basierendes Verfahren, das polinomielle Laufzeiten erreicht[7].
Diese Arbeit wird sich zunächst mit den GTFS-Daten beschäftigen und wie aus diesen ein Graph für den Lösungsansatz von Nöllenburg und Wolff konstruiert werden kann. Dies schließt Tarifzonen bereits ein, da sie mit in den Graphen integriert werden. Im Anschluss wird dieser Lösungsansatz von Nöllenburg und Wolff vorgestellt. Danach werden die Ergebnisse vorgestellt und diskutiert.

Abbildung 1.1: Offizielle Karte der New York Subway von 2013 (`https://commons.wikimedia.org/wiki/File:Official_New_York_City_Subway_Map_2013_vc.jpg`)

Abbildung 1.2: Henry C. Becks Karte des London Underground von 1933 (`https://commons.wikimedia.org/wiki/File:Beckmap1.jpg`)

2 Methoden

In dieser Arbeit wird ein Algorithmus zum Generieren eines gemischt-ganzzahligen Optimierungsproblems (MILP) vorgestellt. Als Eingabe fungieren GTFS-Daten, welche zunächst näher vorgestellt werden.

2.1 GTFS

GTFS steht für General Transit Feed Specification und ist ein Standard[4] für das Beschreiben von nach Fahrplan verkehrendem Personenverkehr. Der Standard beschreibt ein relationales Datenmodell, dessen Daten größtenteils - und für dieses Problem ausschließlich - in Textdateien im CSV-Format gespeichert werden. Auf das CSV-Format wird hier nicht näher eingegangen. Diese Dateien werden in einem ZIP-Archive zusammengefasst und stellen dann GTFS-Daten dar. Die folgenden Dateien sind in jedem GTFS-Datensatz enthalten: agency.txt, routes.txt, trips.txt, stops.txt, stop_times.txt, calendar.txt und calendar_dates.txt. Davon wird nur agency.txt nicht für dieses Problem benötigt, jedoch kann es zum Filtern von Teilnetzen aus großen GTFS-Datensammlungen genutzt werden. Für Tarifzonen werden zusätzlich die Dateien areas.txt und stop_areas.txt benötigt.

2.1.1 Qualität der Daten am Beispiel VBB

Der Verkehrsverbund Berlin Brandenburg (VBB) ist der Verwaltungsverbund des öffentlichen Personennahverkehrs in der Region Berlin Brandenburg. Sie stellen für das Gesamtnetz einen Fahrplan im GTFS-Format zur Verfügung, der im Folgenden verwendet wurde. Zentraler Angelpunkt für Metrokarten sind die Linien in routes.txt. Dort sind Regio-, U-Bahn-, S-Bahn, Bus-, Tram- und Fährlinien aufgelistet mit ID, Bezeichner und Typ. Da eine Route in GTFS nicht gemischtem Bus- und Bahnbetrieb zugeordnet sein kann, ist Schienenersatzverkehr (SEV) als eine separate Route zu der Route, die er ersetzt, gelistet. Problematisch wird dies insbesondere, wenn an einem Tag auf einer Strecke sowohl SEV als auch reguläre Züge verkehren, da dies oft zu impliziten Kreuzungspunkten führt, die vom Ansatz von Nöllenburg und Wolff unterdrückt werden und deswegen explizit im Eingabegraphen sein müssten. Diese Arbeit wird diese Erkennung von impliziten Kreuzungspunkten nicht betrachten. Die eigentliche Linienführung ist in keiner Tabelle explizit gelistet. Stattdessen gibt es eine Liste von Fahrten in trips.txt, die an einem Tag stattfinden. Diese erstrecken sich teilweise nicht über die ganze Strecke, insbesondere dann, wenn aufgrund von SEV keine Züge auf der ganzen Strecke verkehren können oder auf Teilen der Strecke Pendelverkehr eingerichtet ist[11].
Ein weiteres Problem mit den Fahrten in trips.txt ist, dass je nach Fahrtrichtung die Stationen sich unterscheiden können. Bei der U-Bahn ist dies häufig eine übersprungene Station in einer Fahrtrichtung; bei Tramlinien gibt es einzelne Stationen, die in zwei verschiedene Stationen aufgeteilt sind, wie das auf der Linie 27 am Halt Am Faulen See der Fall ist[1]; und bei Bushaltestellen ist jede Halteposition als getrennte Station in den Daten gelistet.

2.1.2 Ein View für Linien

Um besser mit den Daten zu arbeiten, wurden sie in eine SQLite-Datenbank übertragen. Dort wurde ein View angelegt, der für alle Routen eine Liste von gerichteten Kanten abbildet. Dieser Prozess kann in vier Schritte unterteilt werden.

1. Da GTFS-Datensätze über lange Zeiträume gültig sind, können einige Änderungen der Linienführung enthalten sein. Um eine brauchbare Metrokarte zu erhalten, muss ein Tag festgelegt werden, für den die Karte generiert wird. Es kann möglicherweise nötig sein die Zeit noch weiter einzuschränken, da baubedingte Fahrplanänderungen des Öfteren im Laufe eines Betriebstages beginnen oder enden. So beginnen Sperrungen im Berliner S-Bahnnetz häufig um 22 Uhr.

2. stops.txt enthält nicht nur Stationen, sondern auch untergeordnete Haltepunkte in diesen Stationen. Dies können einzelne Gleise oder Ebenen sein. Daher werden über das Feld parent_station die Wurzelknoten der Stationenbäume ermittelt.

3. Das Format für Fahrten in stop_times.txt gibt die Reihenfolge über das Feld stop_sequence an. Nach dem Zusammenführen der verschiedenen Fahrten können jedoch getrennte Linienführungen entstehen, die in diesem Format nicht ausdrückbar sind. Daher werden statt den Halten die Kanten zwischen den Halten ermittelt.

4. Zuletzt werden doppelte Kanten der selben Linie entfernt.

Daraus resultiert zum einen Λ – die Menge der Linien, wobei jede Linie eine Menge der Kanten darin ist – und der Liniengraph, bei dem auch die doppelten Kanten über verschiedene Linien entfernt werden.

2.2 Tarifzonengraph

In der zu generierenden Grafik sind Tarifzonen durch Kanten an ihren Grenzen dargestellt. Für den Ansatz von Nöllenburg und Wolff muss der zu zeichnende Graph planar sein. Um dies zu gewährleisten, werden alle tarifzonenüberschneidenen Kanten im Liniengraphen ermittelt. Diese werden anschließend zyklisch um eine Tarifzone verbunden.
Erreicht wird dies durch Konstruieren eines Tarifzonengraphs. Es handelt sich dabei um einen Multigraphen, dessen Knoten Tarifzonen beschreiben und Kanten Verbindungen zwischen Tarifzonen repräsentieren. Dabei sind die ausgehende Kanten in der Adjazenzliste geordnet. Konstruiert wird er aus dem Liniengraph. Dabei werden alle Knoten des Liniengraphs aus Kapitel 2.1.2 zunächst in den Tarifzonengraph übernommen. In Adjazenzlisten werden die ausgehenden Kanten nach Richtung im oder gegen den Uhrzeigersinn sortiert. Im nächsten Schritt werden nacheinander alle Knotenpaare (v_1, v_2), die mit einer Kante verbunden sind und in der selben Tarifzone liegen, verbunden, indem ihre ausgehenden Kanten verkettet werden. Diese Verkettung erfolgt, indem die verbindenden Kanten zwischen den Knoten identifiziert werden. Die auf der Adjazenzliste von v_2 nach der Verbindungskante nachfolgenden Kanten werden in der Adjazenzliste von v_1 statt der Verbindungskante eingefügt. Im Anschluss werden alle Knotenpaare (v_1, v_2) verbunden, die in der gleichen Tarifzone liegen aber keine gemeinsame Kante haben. Hierfür wird die Adjazenzliste von v_2 an die von v_1 angehängt. Im resultierenden Graphen sind nur die Kanten enthalten, die eine Tarifzonengrenze queren. Durch ihre

Sortierung in der Adjazenzliste der Knoten haben die Kanten eine zyklische Ordnung, mit der sie verbunden werden können, um die Tarifzonengrenze in den Liniengraph zu integrieren.
Im Liniengraphen werden entsprechend alle Kanten, die auch im konstruierten Tarifzonengraphen enthalten sind, in zwei Kanten mit einem Tarifzonengrenzknoten geteilt. Unter der Annahme, dass die Tarifzonen als konzentrische Kreise organisiert sind und Linien keinen Kreis überspringen, können die Tarifzonengrenzknoten entsprechend der Sortierung ihrer zugrundeliegenden Kanten in den Adjazenzlisten der Tarifzonenknoten verbunden werden.

2.3 Gemischt-Ganzzahlige Optimierungsprobleme (MILP)

Für den Ansatz zum Generieren von Metrokarten aus Nöllenburg und Wolff wird MILP eingesetzt. Dies ist eine Beschreibung für Probleme, die diese als Optimierung einer Menge Variablen unter linearen Constraints darstellt. Zusätzlich kann an Variablen der Anspruch der Ganzzahligkeit gestellt werden.
Gelöst werden diese Problem durch Solver wie beispielsweise CPLEX, der für diese Arbeit verwendet wurde.

2.3.1 Variablen

In dem Ansatz von Nöllenburg und Wolff werden einige Variablen benötigt, die im Folgenden aufgelistet sind. Zusätzlich werden die $bound$-Variablen eingeführt.

Ganzzahlige

- dir_e - modelliert die oktolineare Richtung einer Kante e. Somit gilt für alle e, dass $0 \leq dir_e < 8$.
- $\alpha_{e,i}$ - sind Hilfsvariablen zu einer Kante e, die 1 annimmt, wenn die Richtung i gewählt wird, wobei die Richtung i relativ zur optimalen oktolinearen Richtung ist. Für alle e, i gilt, dass $\alpha_{e,i} \in \{0, 1\}$.
- $\beta_{v,i}$ - sind Hilfsvariablen für die zyklische Ordnung der Kanten um einen Knoten v. Eine Variable ist 1 an der Stelle des Zyklenschlusses in der Ordnung. Für alle v gilt, dass $0 \leq i < deg(v)$. Für alle v, i gilt, dass $\beta_{v,i} \in \{0, 1\}$.
- $\gamma_{e_1,e_2,i}$ - sind Hilfsvariablen zur Abstandshaltung von Kanten, wobei zwei Kanten e_1 und e_2 immer in einer Richtung i einen Mindestabstand haben müssen, damit gesichert ist, dass sie sich nicht schneiden. Für alle e_1, e_2, i gilt, dass $\gamma_{e_1,e_2,i} \in \{0, 1\}$.
- $bd_{u,v,w}$ - die Knickstärke an einem Knoten v auf dem Pfad von u nach w.
- $\delta'_{u,v,w}$ und $\delta''_{u,v,w}$ - sind Hilfsvariablen zum Beschreiben von bd. Sie sind entweder 1 oder 0.

Reelle

- x_v - die x-Position eines Knoten v in der Grafik.
- y_v - die y-Position eines Knoten v in der Grafik.

- λ_e - die Länge einer Kante e.
- $bound_{top}, bound_{bottom}, bound_{left}, bound_{right}$ - sind die jeweils minimalen und maximalen x- und y-Werte.

2.3.2 Constraints

Nöllenburg und Wolff formulieren in ihrem Ansatz vier Constraintgruppen. Diese werden abgeleitet aus den Design-Anforderungen an die Karte:

- Oktolinearität,
- korrekte Sortierung der Kanten um einen Knotenpunkt,
- eine minimale Kantenlänge und
- keine ungewollten Schnittpunkte.

Im Anschluss folgen noch zwei Constraintgruppen für Design-Anforderungen, nach denen optimiert wird:

- Knickvermeidung und
- kurze Kanten.

Zusätzlich wurde in dieser Arbeit noch eine weitere Constraintgruppe eingeführt.

Oktolinearität und minimale Kantenlänge

Es sei $sec(e)$ die oktolineare Richtung einer Kante e. Dabei werden die tatsächlichen Richtungen basierend auf den Koordinaten der Endpunkte zur nächsten oktolinearen Richtung gerundet. Für dir_e ergeben sich dann die Constraints

$$\sum_{i \in I} \alpha_{e,i} = 1,$$
$$dir_e = \sum_{i \in I} ((sec(e) + i + 8) \mod 8) \cdot \alpha_{e,i}.$$

Der Term $(sec(e) + i + 8) \mod 8$ ist konstant, weshalb dir_e durch ein lineares Constraint bestimmt ist.
Für alle möglichen $\alpha_{e,i}$, wobei $e = (u, v)$, betrachte man die Achsen $a_1(u), a_1(v)$ und $a_2(u), a_2(v)$, wobei diese sich aus x und y zusammensetzen in Abhängigkeit von der Richtung, für die $\alpha_{e,i}$ steht. Sei a_1 die Achse, auf der bei Richtungswahl $\alpha_{e,i}$ die Knoten u und v auf einer Höhe liegen müssen. So ergeben sich die Constraints

$$\begin{aligned} a_1(u) - a_1(v) &\leq M_L(1 - \alpha_{e,i}), \\ a_1(v) - a_1(u) &\leq M_L(1 - \alpha_{e,i}), \\ a_2(u) - a_2(v) &\geq -M_L(1 - \alpha_{e,i}) + L. \end{aligned}$$

L und M_L sind Konstanten, wobei M_L die maximale und L die minimale Länge einer Kante ist.

Schnittpunktunterbindung

Damit keine ungewollten Schnittpunkte entstehen, müssen Kanten in wenigstens einer der oktolinearen Richtungen auseinander liegen[9]. Für die entsprechenden Entscheidungsvariablen gilt für alle Kantenpaare e_1, e_2, dass

$$\sum_{i \in \{0,\dots,7\}} \gamma_{e_1,e_2,i} = 1.$$

Für jede der acht oktolinearen Richtungen, gilt für eine beliebige Richtung a und zwei Kanten $e_1 = (u_1, v_1)$ und $e_2 = (u_2, v_2)$, dass

$$\begin{aligned}
a(u_2) - a(u_1) &\leq M_D(1 - \gamma_{e1,e2,i}) - D,\\
a(u_2) - a(v_1) &\leq M_D(1 - \gamma_{e1,e2,i}) - D,\\
a(v_2) - a(u_1) &\leq M_D(1 - \gamma_{e1,e2,i}) - D,\\
a(v_2) - a(v_1) &\leq M_D(1 - \gamma_{e1,e2,i}) - D.
\end{aligned}$$

D und M_D sind Konstanten, wobei M_D der maximale und D der minimale Abstand zweier Kanten ist.

Korrekte Kantenordnung um einen Knotenpunkt

Entsprechend der Ordnung der Kanten, wie durch ihre Richtung in den Eingabedaten festgelegt, soll diese auch in der Ausgabe erhalten bleiben. Die dir_e der Kanten ist im Kreis immer aufsteigend mit genau einer Ausnahme. Die Ausnahme wird modelliert mittels einer Hilfsvariablen für alle Knoten v, für die gilt

$$\sum_{0 \leq i < deg(v)} \beta_{v,i} = 1.$$

Seien $u_1, \dots, u_{deg(v)}$ die Endknoten der Kanten um v, wie die oktolinearen Richtungen gegen den Uhrzeigersinn sortiert, dann gilt für alle $1 \leq i \leq deg(v)$

$$dir_{(v,u_i)} \leq dir_{(v,u_{(i+1) \mod dir(v)})} - 1 + 8\beta_{v,i}$$

Kantenlänge

Für die Optimierung der Kantenlänge muss eine Hilfsvariable für jede Kante mit der Länge dieser berechnet werden. Dafür gibt es für jede Kante $e = (u, v)$ folgende Constraints

$$\begin{aligned}
x_u - x_v &\leq \lambda_e,\\
x_v - x_u &\leq \lambda_e,\\
y_u - y_v &\leq \lambda_e,\\
y_v - y_u &\leq \lambda_e.
\end{aligned}$$

Anzahl und Stärke der Knicke in Linienverläufen

Die Stärke eines Knickes zwischen zwei Kanten bestimmt sich durch die Differenz zwischen den jeweiligen Werten von dir. Seien die Knoten u, v, w Teil von zwei verschiedenen Kanten der Form

(u, v) und (v, w), dann beschreiben folgende Gleichungen von Nöllenburg und Wolff die Hilfsvariable bd:

$$\begin{aligned} -bd_{u,v,w} &\leq dir_{(u,v)} - dir_{(v,w)} - 8\delta'_{u,v,w} + 8\delta''_{u,v,w}, \\ bd_{u,v,w} &\geq dir_{(u,v)} - dir_{(v,w)} - 8\delta'_{u,v,w} + 8\delta''_{u,v,w}. \end{aligned}$$

Kartendimensionen

Für eine weitere Optimierung wird die Größe der Karte herangezogen. Daher gilt für alle Knoten v folgendes:

$$\begin{aligned} bound_{top} &\leq y_v, \\ bound_{bottom} &\geq y_v, \\ bound_{left} &\leq x_v, \\ bound_{right} &\geq x_v. \end{aligned}$$

2.3.3 Optimierungsfunktion

Die Optimierungsfunktion bei Nöllenburg und Wolff setzt sich aus 3 Teilen zusammen. Es wird optimiert nach möglichst wenigen und schwachen Knicken, möglichst häufige Nutzung der rechnerisch korrekten oktolinearen Richtung und eine möglichst geringe Gesamtlänge aller Kanten. Eine weitere Optimierung $cost_4$ in dieser Arbeit ist die Minimierung des Umfanges. $cost_2$ ist äquivalent zu Nöllenburg und Wolff, aber ohne zusätzliche Variablen und Constraints.
Sei Λ wie in Kapitel 2.1.2 definiert, dann seien

$$\begin{aligned} cost_1 &= \sum_{L \in \Lambda} \sum_{(u,v),(v,w) \in L} bd_{u,v,w}, \\ cost_2 &= \sum_{e} \sum_{i \neq 0} \alpha_{e,i}, \\ cost_3 &= \sum_{e} \lambda_e, \\ cost_4 &= bound_{bottom} - bound_{top} + bound_{right} - bound_{left}. \end{aligned}$$

Diese Optimierungsfunktionen werden nun gewichtet und kombiniert. In dieser Arbeit wurden sie immer zu gleichen Teilen gewichtet.

3 Ergebnisse

Die Methoden wurden anhand des Berliner U-Bahnnetzes untersucht (Abbildung 3.1). Einmal wurden sie ohne die Inklusion von Tarifzonen betrachtet und einmal mit und dabei sowohl mit Schnittpunktunterbindung, als auch ohne. Die verschiedenen Kombinationen von Randbedingungen sind in Tabelle 3.1 zusammengefasst und werden in den folgenden Kapiteln einzeln betrachtet.
Die Berechnungen wurden auf dem Hydra Cluster der TU Berlin ausgeführt[2] mit einer maximalen Ausführungsdauer von sieben Tagen. Primär wurden die Berechnungen auf Maschinen mit 56 Kernen und 2-fach-Hyperthreading ausgeführt. Diese hatten je 2048 GB RAM zur Verfügung. Eine Maschine hatte 6000 GB RAM zur Verfügung mit 18 Kernen und 4-fach-Hyperthreading.
Die von Nöllenburg und Wolff eingeführten Qualitätsfunktionen, sowie die Platzsparsamkeitsoptimierung wurden zu je gleichen Teilen in der Optimierungsfunktion gewichtet.
Es konnten in der Zeit von sieben Tagen keine Lösungen für Probleme gefunden werden, bei denen die Schnittpunktunterbindung aktiv war. Diese sind gemeinsam in Kapitel 3.1 behandelt. Ein Plan wurde ohne Tarifzonengrenzen und ohne Schnittpunktunterbindung generiert und wird in Kapitel 3.2 genauer behandelt. Mit Tarifzonengrenzen aber ohne Schnittpunktunterbindung wurden Pläne einmal mit 1,95 TB und einmal mit 6 TB RAM generiert. Dies wird in Kapitel 3.3 behandelt.
Die Grafiken, die für gültige Lösungen resultieren, bestehen aus schwarzen und roten Kreisen, welche die Knoten des Netzgraphen darstellen. Schwarz sind Stationen, an denen eine Linie hält und rot sind die Knoten der Tarifzonengrenze. Die Kanten sind als schwarze Geraden eingezeichnet.

3.1 Schnittpunktunterbindung

Es gab 4 Versuche Karten mit Schnittpunktunterbindung zu generieren. Davon waren drei mit Tarifzonengrenzen. In allen Fällen wurden keine Ergebnisse vor der Sieben-Tagesfrist gefunden. Das kleinste Problem beginnt mit 2 Millionen Constraints und das größte hat 6,2 Millionen. Bei der Variablenanzahl beläuft sich die Spanne auf 488 Tausend bis 1,5 Millionen. Zur besseren Übersicht ist

Kapitel	Zonen?	Linien	Schnittpunkt-unterbindung	Threads	RAM	Ergebnis	Zwischen-ergebnisse
3.2	Nein	U1–9	Nein	112	1,95 TB	Speicherüberlauf	15
3.1	Nein	U1–9	Ja	32	1,8 TB	Zeitüberschreitung	0
3.1	Ja	U1–9	Ja	112	1,95 TB	Zeitüberschreitung	0
3.3	Ja	U1–9	Nein	112	1,95 TB	Speicherüberlauf	15
3.3	Ja	U1–9	Nein	72	6 TB	Speicherüberlauf	15
3.1	Ja	U5–9	Ja	112	2 TB	Zeitüberschreitung	0
3.1	Ja	U5–7+9	Ja	112	2 TB	Zeitüberschreitung	0

Tabelle 3.1: Kurzzusammenfassung der in den Unterkapiteln behandelten Ergebnisse

Linien	Zonen?	Anzahl der Constraints	Anzahl der Variablen
U1-9	Nein	4423015	1075510
U1-9	Ja	6153632	1495195
U5-9	Ja	2941824	714394
U5-7+9	Ja	2007311	487649

Tabelle 3.2: Übersicht der Anzahl von Constraints und Variablen der Probleme mit Schnittpunktunterbindung

dies in Tabelle 3.2 dargestellt.

3.2 Pläne ohne Tarifzonengrenzen

Es wurde einmal versucht ein Plan für das U-Bahnnetz ohne die Inklusion von Tarifzonen zu generieren, bei dem die Schnittpunktunterbindung nicht beachtet wurde. Daraus resultierten für das zu lösende Problem 7421 Constraints und 4159 Variablen. Die erste gefundene Zwischenlösung, die in Abbildung 3.2 zu sehen ist, wurde nach 10 Sekunden gefunden. Die letzte gefundene Zwischenlösung, die in Abbildung 3.3 zu sehen ist, wurde nach knapp 4 Stunden gefunden. Nach knapp 13 Stunden brach die Suche ab, da der Arbeitsspeicher übergelaufen war.
Wie in Abbildung 3.2 exemplarisch sichtbar, sind die ersten Lösungen, die gefunden wurden, sehr gestreckt. Die meisten Kanten sind verglichen mit der theoretisch korrekten Richtung um 45 Grad gegen den Uhrzeigersinn gedreht. Nach einigen Zwischenergebnissen in dieser Form gibt es einen abrupten Wechsel zu der gewohnten Orientierung, wie sie im offiziellen S- und U-Bahnnetzplan existiert[1]. Hier ist dies nach dem 6. Zwischenergebnis geschehen.
Im letzten Zwischenergebnis bevor das Programm abbricht sind zwei Kreuzungspunkte von Linien, die sich in Wirklichkeit an diesen Stellen nicht kreuzen. In Abbildung 3.3 ist dies mit roten Kreisen markiert.

3.3 Pläne mit Tarifzonengrenzen

Mit der Inklusion von Tarifzonengrenzen hat das resultierende Problem 8450 Constraints und 4237 Variablen. Dieses Problem wurde einmal mit 112 Threads und 1,95 TB RAM und einmal mit 72 Threads und 6 TB RAM ausgeführt. Die jeweils erste Lösung ist bis auf die Positionierung von Stationsknoten auf langen Geraden identisch und in Abbildung 3.4 dargestellt. Diese Lösung wurde auf der 1,95 TB-Maschine in 15 und auf der 6 TB-Maschine in 10 Sekunden gefunden. Die jeweils beste Lösung wurde auf der 1,95 TB-Maschine nach 10 Minuten und 19 Sekunden (Abbildung 3.5) und auf der 6 TB-Maschine nach einem Tag und 18 Stunden gefunden (Abbildung 3.6). Mit 1,95 TB konnte 21 Stunden und 44 Minuten und mit 6 TB 2 Tage und 16 Stunden lang gesucht werden, bevor jeweils der Arbeitsspeicher nicht mehr ausreichte.
Wie in Kapitel 3.2 kann in den ersten Zwischenergebnissen die Rotation um 45 Grad gegen den Uhrzeigersinn beobachtet werden (Abbildung 3.4). Wieder gibt es nach ein paar Zwischenergebnissen ebenfalls den plötzlichen Wechsel zur normalen Rotation (Abbildungen 3.5 und 3.6).

Abbildung 3.1: Berliner U-Bahnnetz von Arbalete (`https://commons.wikimedia.org/wiki/File:U-Bahn_Berlin_-_Netzplan.svg`)

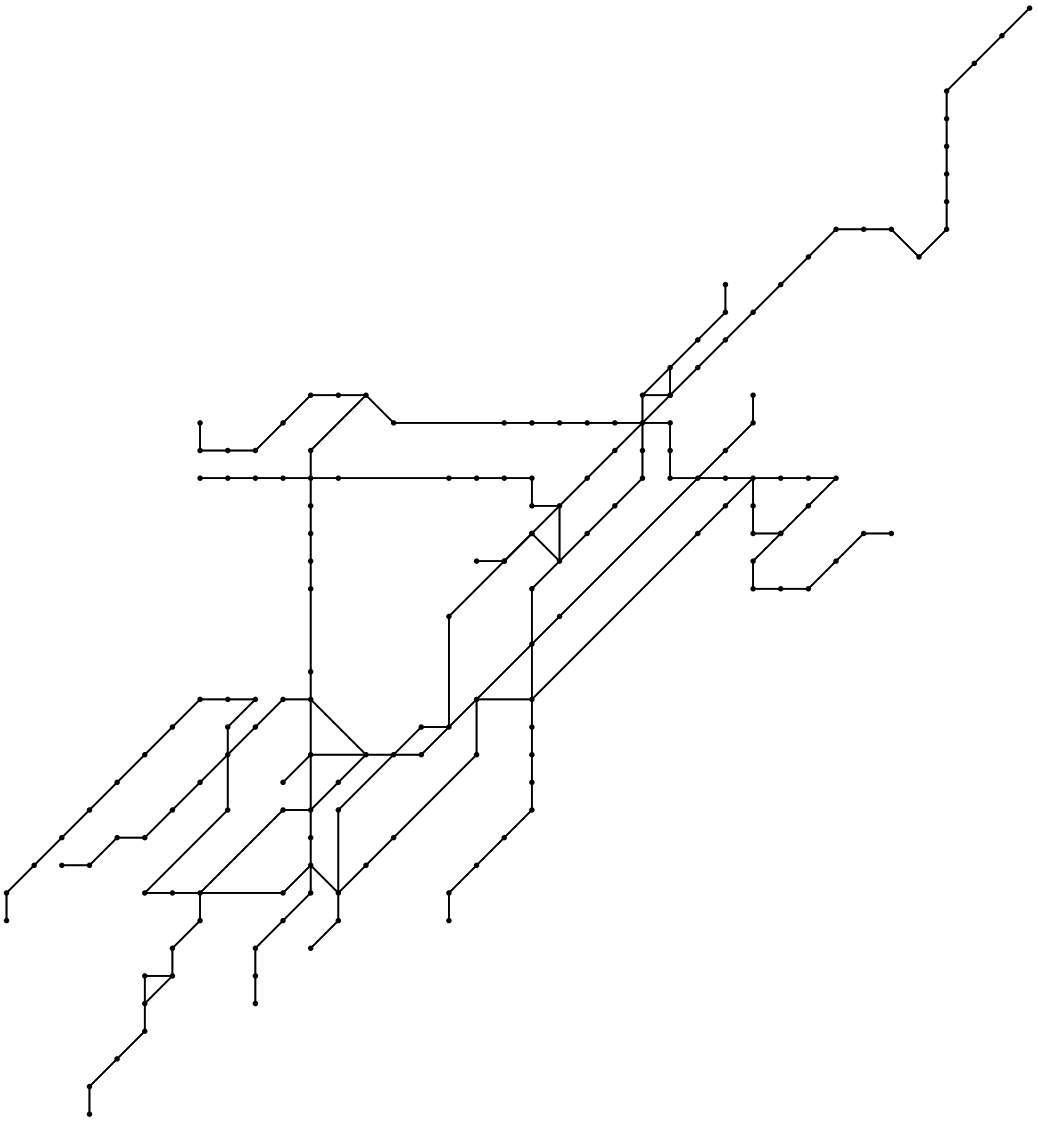

Abbildung 3.2: Erste Zwischenlösung für das ganze Berliner U-Bahnnetz ohne Tarifzonengrenzen und ohne Schnittpunktunterbindung

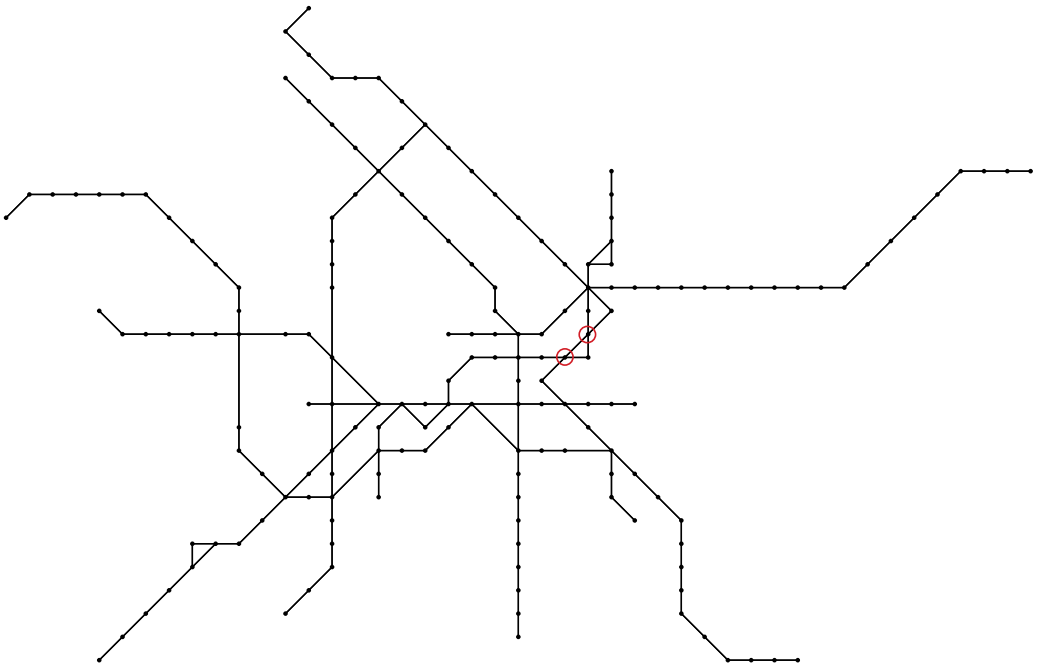

Abbildung 3.3: Beste Zwischenlösung für das ganze Berliner U-Bahnnetz ohne Tarifzonengrenzen und ohne Schnittpunktunterbindung; rote Kreise markieren fehlerhafte Kreuzungspunkte

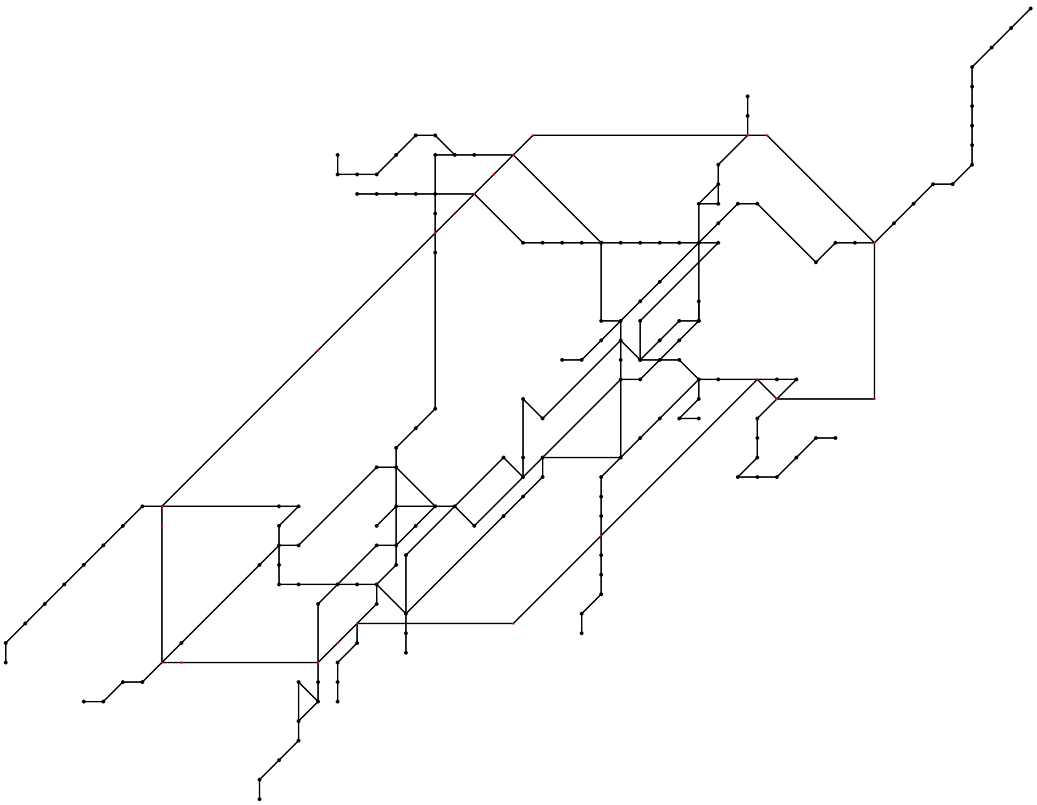

Abbildung 3.4: Erste Zwischenlösung für das ganze Berliner U-Bahnnetz mit Tarifzonengrenzen aber ohne Schnittpunktunterbindung

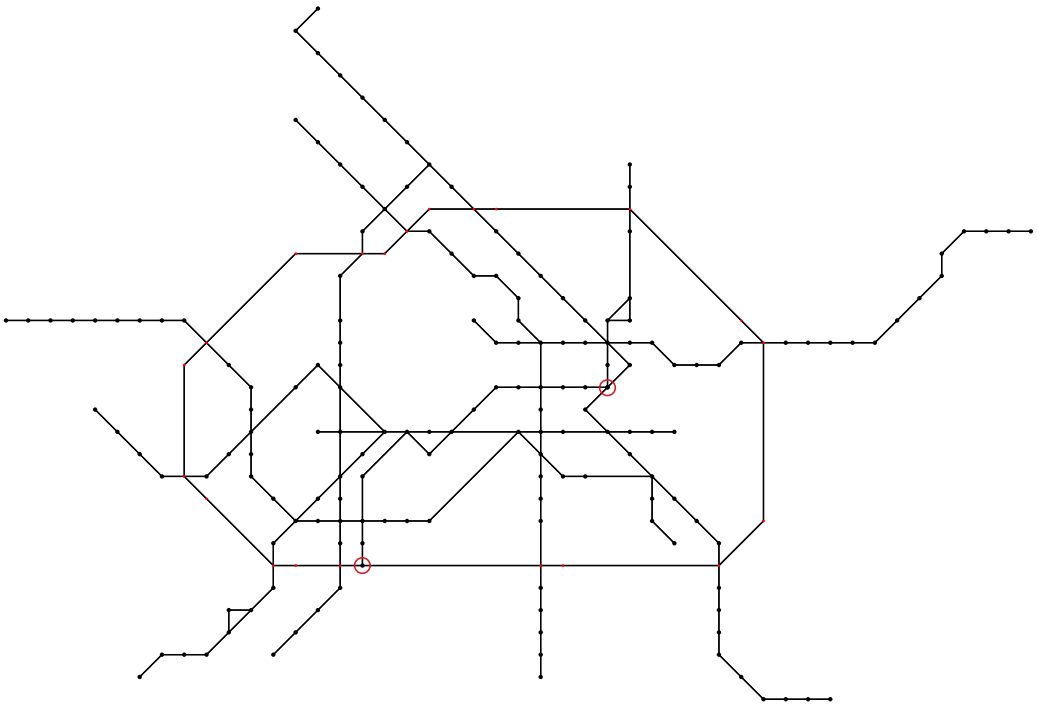

Abbildung 3.5: Beste Zwischenlösung mit 1,95 TB RAM für das ganze Berliner U-Bahnnetz mit Tarifzonengrenzen aber ohne Schnittpunktunterbindung; rote Kreise markieren fehlerhafte Kreuzungspunkte

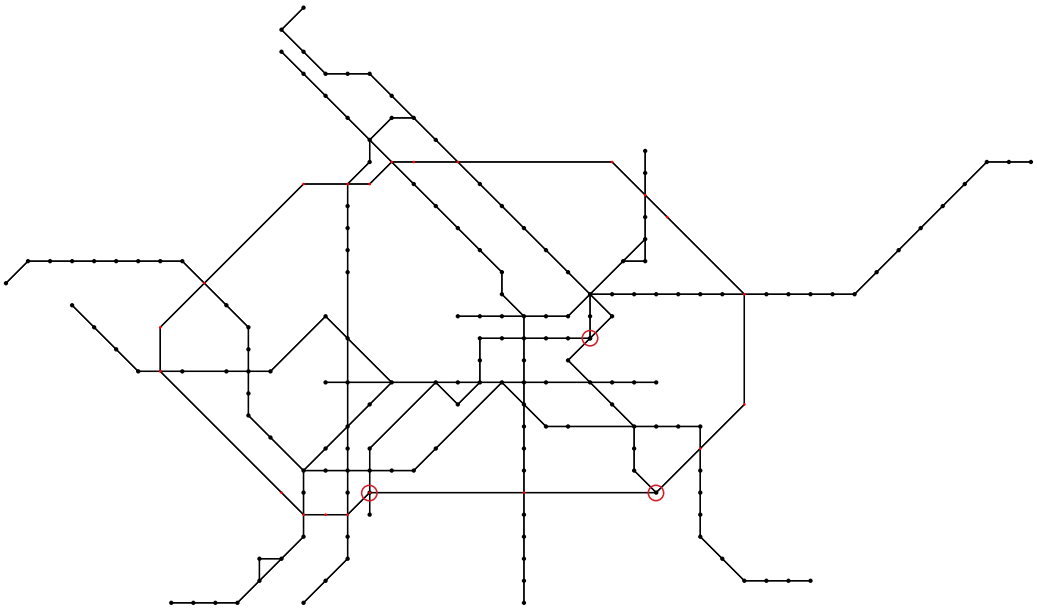

Abbildung 3.6: Beste Zwischenlösung mit 6 TB RAM für das ganze Berliner U-Bahnnetz mit Tarifzonengrenzen aber ohne Schnittpunktunterbindung; rote Kreise markieren fehlerhafte Kreuzungspunkte

4 Diskussion

Die Tarifzonen wurden erfolgreich eingezeichnet. In Abbildung 3.6 kann man zwar zwei Punkte sehen, die auf der Tarifzonengrenze liegen, obwohl sie innerhalb sein müssten, doch ist dies auf die nicht aktive Regel zur Schnittpunktunterbindung zurückzuführen.
Obwohl keines der Probleme bis zum Ende gerechnet werden konnte, sind sie den offiziellen Karten[1] ähnlich. Der Wert der berechneten Karten ist jedoch stark geschmälert, da eine nicht-vertraute Nutzer*in aufgrund des Mangels an Beschriftungen keine Stationen finden könnte. Nöllenburg und Wolff hatten hierfür bereits einen Ansatz vorgeschlagen, der in dieser Arbeit nicht behandelt wurde[9]. Ein weiteres Problem stellt die fehlende Differenzierung der Linien durch Farben oder ähnliches dar.
Ein großes Problem war das Vermeiden von zusätzlichen Schnittpunkten an Stellen, bei denen keine sein sollten. Wie Nöllenburg und Wolff bereits beobachteten, gibt es nur wenige Kanten, die sich überhaupt kreuzen könnten[9]. Dennoch wurde in dieser Arbeit der „naive Ansatz“ von Nöllenburg und Wolff verwendet, der die Anzahl Constraints und Variablen mit der Anzahl Kanten quadratisch wachsen lässt.
In dieser Arbeit wurden die Karten direkt aus GTFS-Daten abgeleitet statt manuell vorbereitet zu werden. Dies geht allerdings mit einigen Einschränkungen einher. Damit die Methode von Nöllenburg und Wolff funktioniert, muss der Eingabegraph planar sein. Doch diese Arbeit zeigt keinen Weg auf dies zu erreichen. Wird eine Station nur in eine Richtung bedient oder es halten nicht alle Linien an einer Station, durch die gefahren wird, so werden in der Praxis diese Linien trotzdem parallel geführt. Dies wurde in dieser Arbeit nicht betrachtet und findet sich in Form von kleinen Dreiecken daher in den Ergebnissen wieder.

5 Fazit

Ziel der Arbeit war es aus GTFS-Daten eine oktolineare Darstellung des Netzwerkes inklusive seiner Tarifzonengrenzen mittels eines Mixed Integer Linear Programming Ansatzes von Nöllenburg und Wolff zu entwickeln. Dies ist in Teilen gelungen. Es funktioniert mit dem Berliner U-Bahnnetz, doch funktioniert es nicht mit beliebiger Netztopologie. Insbesondere die Tarifzonen müssen konzentrisch angeordnet sein und dürfen nicht beliebig sein.
Es wäre zu untersuchen, ob mit den Optimierungen zur Beschleunigung der Berechnung von Nöllenburg und Wolff oder durch die Optimierungen von Onda et al.[10] eine ausreichende Verbesserung erreichbar wäre, sodass das Netzwerk auch mit Schnittpunktunterbindung berechnet werden könnte. Eine Verallgemeinerung auf mehr Netztopologien wäre ebenfalls zu betrachten.
Die Einschränkungen bedeuten, dass ein Einsatz in der Praxis auch wegen der langen Laufzeiten noch unpraktikabel ist und noch weiterer Forschungsbedarf besteht.

Literaturverzeichnis

[1] Alte netzspinnen und planungen - berlin:verkehr. https://berliner-verkehr.de/html_neu/hauptstrasse83f.de/liniennetz/alte-netzspinnen/index.html. Accessed: 12.11.2024.

[2] Documentation of the hydra cluster. https://git.tu-berlin.de/ml-group/hydra/documentation/-/blob/main/README.md. Accessed: 13.11.2024.

[3] Mta maps. https://new.mta.info/maps. Accessed: 12.11.2024.

[4] Reference - general transit feed specification. https://gtfs.org/documentation/schedule/reference/. Accessed: 13.11.2024.

[5] Standard tube map. https://content.tfl.gov.uk/standard-tube-map.pdf. Accessed: 12.11.2024.

[6] Beck at 90. *The Cartographic Journal*, 60(4):264–281, 2023.

[7] Hannah Bast, Patrick Brosi, and Sabine Storandt. Metro maps on octilinear grid graphs. *Computer Graphics Forum*, 39(3):357–367, 2020.

[8] Alexander J. Kent. When topology trumped topography: Celebrating 90 years of beck's underground map. *The Cartographic Journal*, 58(1):1–12, 2021.

[9] Martin Nollenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011.

[10] Masahiro ONDA, Masaki MORIGUCHI, and Keiko IMAI. Automatic drawing of complex metro maps. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E104.A(9):1150–1155, 2021.

[11] RBB24. U2 am alexanderplatz bleibt bis ende august unterbrochen. https://www.rbb24.de/panorama/beitrag/2023/01/u2-alexanderplatz-bis-ende-sommerferien-unterbrochen.html, 2023. Accessed: 15.11.2024.