Technische Universität München

Zentrum Mathematik

k-Center in Verkehrsnetzwerken – ein Vergleich geometrischer und graphentheoretischer Ansätze

Diplomarbeit von Valentin Breuß

Aufgabensteller:	Prof. Dr. Peter Gritzmann
Betreuer:	Dr. René Brandenberg
	Dipl.-Math. Stefan König
Abgabe:	1. Juni 2010

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

München, den 1. Juni 2010

Valentin Breuß

Inhaltsverzeichnis

1 Einführung

Die vorliegende Diplomarbeit beschäftigt sich mit dem Vergleich der geometrischen und graphentheoretischen Varianten des Standortproblems k-CENTER. Dieses Problem besteht darin, zu gegebenen Standorten (oder Kunden) eine Menge von maximal k Zentren (oder Lagern) zu finden, sodass der Maximalabstand aller Standorte zum jeweils nächsten Zentrum minimiert wird.

Problem 1.1 *(*k*-Center)*
Gegeben seien eine Grundmenge M mit einer Distanzfunktion $\delta : M \times M \to \mathbb{R}^+$, eine Standortmenge $S \subseteq M$, sowie $k \in \mathbb{N}$. Das Problem eine höchstens k-elementige Menge von Zentren $C \subseteq M$ zu finden, sodass

$$\max_{s \in S} \min_{c \in C} \delta(c, s) \tag{1.1}$$

minimiert wird, heißt k-CENTER.

In Problem 1.1 ist sowohl die Grundmenge M als auch die Distanzfunktion δ bewusst sehr allgemein gehalten. Durch deren Wahl können verschiedene k-Center Varianten beschrieben werden.
So verwendet das geometrische k-Center als Grundmenge einen normierten Raum, wobei die Distanzfunktion $\delta(x, y) := \|x - y\|$ von der Norm induziert wird. Das graphentheoretische k-Center benötigt hingegen einen vollständigen, gewichteten Graphen $G = (V, E, \delta)$. Dabei entspricht die Grundmenge der Knotenmenge V und die Distanzfunktion wird direkt durch die Kantengewichte bestimmt.

1.1 Motivation

Sollen beispielsweise in der Städteplanung k Krankenhäuser gebaut werden, sodass im Ernstfall die maximale Strecke der Rettungshubschrauber zur Unglücksstelle möglichst gering wird, entspricht dies – unter der Annahme, dass die Krankenhäuser beliebig platziert werden können – dem geometrischen k-Center mit allen möglichen Unglücksstellen als Standortmenge.
Werden die Patienten hingegen nicht mit Rettungshubschraubern, sondern mit Krankenwagen transportiert, ist die reale Entfernung entlang der Straßen zum Unglücksort, bzw. die bis zur Ankunft verstrichene Zeit relevant und nicht die Luftlinienentfernung. Dies kann durch einen gewichteten Graphen modelliert werden, wobei die Gewichtsfunktion je nach Bedarf der Weglänge oder der erwarteten Fahrtzeit entsprechen kann.

Für das geometrische k-Center haben Brandenberg und Roth in [3] einen Branch-and-Bound Ansatz vorgestellt, dessen Laufzeit kaum von der Anzahl der Standorte abhängt, dafür recht

empfindlich auf große k reagiert. Bei dem in Kapitel 4 vorgestellten Algorithmus von Ilhan & Pinar verhält es sich hingegen umgekehrt. Während der Algorithmus für zunehmendes k sogar effizienter wird, löst er eine Reihe von ganzzahligen linearen Programmen (ILPs), deren Dimension quadratisch mit der Standortanzahl wächst und kann damit nicht auf Probleme mit großen Standortmengen angewendet werden.
In Verkehrsnetzwerken kann eine Menge von Standorten als Instanz für beide Varianten betrachtet werden. Einerseits sind die Standorte als Punkte auf der Erdoberfläche ein Teil des (Minkowski-) Raums $\mathbb{R}^3$, andererseits kann durch ein zusammenhängendes Straßennetz ein vollständiger Graph erklärt werden, mit den kürzesten Verbindungen als Kantengewichten. Es stellt sich nun die Frage, wie gut eine graphentheoretische Lösung für das geometrische k-Center ist und umgekehrt. Beim geometrischen k-Center können die Zentren beliebig im Raum verteilt werden, während beim graphentheoretischen k-Center nur vorhandene Knoten verwendet werden dürfen. So muss beispielsweise der graphentheoretische Algorithmus in Abbildung 1.1 einen der beiden Knoten als Zentrum auswählen, während der geometrische Algorithmus das Zentrum zwischen den Knoten platzieren kann.

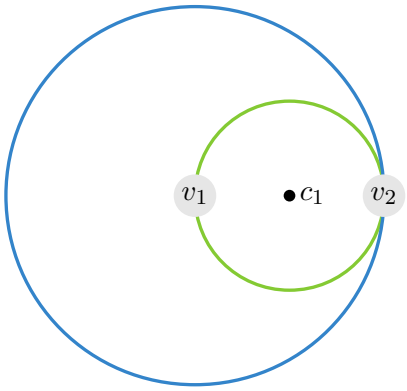

Abbildung 1.1: Geometrisches 1-Center c_1 und graphentheoretisches 1-Center v_1 mit eingezeichnetem Radius.

Um die beiden Varianten trotzdem vergleichbar zu machen, führen wir in Abschnitt 4.3 eine Verallgemeinerung des graphentheoretischen k-Centers ein, bei der zum Graphen zusätzliche Knoten hinzugefügt werden können, ohne dass diese von den Zentren überdeckt werden müssen. Indem einer dieser Knoten in c_1 platziert wird, ist es möglich im Graphen aus Abbildung 1.1 für beide Varianten dasselbe Ergebnis zu erhalten.

In den Abbildungen 1.2 und 1.3 ist zur selben Standortmenge ein 4-Center gelöst worden. Die gefundenen Zentren sind in schwarz markiert, während die 54 Standorte nach Zuordnung zum jeweils nächsten Zentrum eingefärbt wurden. In Abbildung 1.2 ist dabei eine Lösung des „klassischen“ k-Centers und in Abbildung 1.3 eine Verallgemeinerung mit 320 zusätzlichen Knoten auf den Kreuzungspunkten aller Kreisstraßen eingezeichnet.
Besonders auffällig ist, dass das klassische k-Center zwei Zentren benötigt, um die Standorte auf der rechten Seite zu überdecken, während die Verallgemeinerung mit nur einem Zentrum auskommt. Der Radius verringert sich dadurch um mehr als ein Viertel.

In diesem Kapitel werden zunächst noch einige grundlegende mathematische Begriffe definiert, bevor in Kapitel 2 dargestellt wird, wie aus Verkehrsnetzwerken gewichtete Graphen

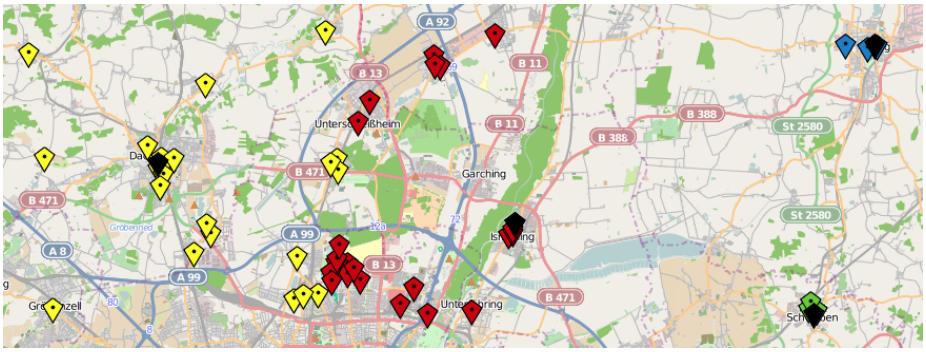

Abbildung 1.2: Klassisches graphentheoretisches 4-Center mit 10.5 Kilometer Radius.

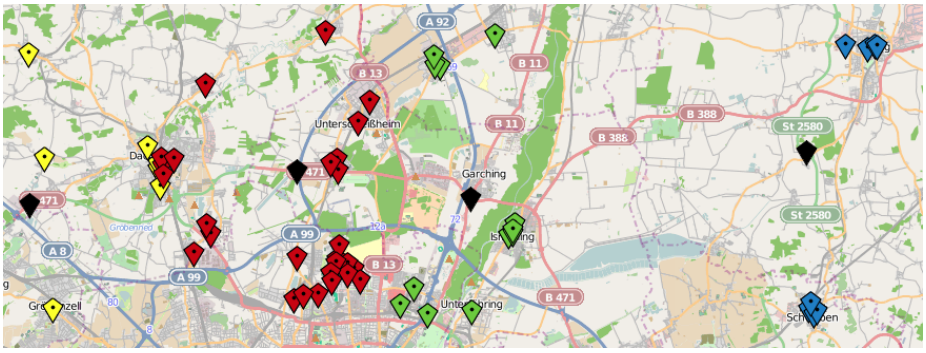

Abbildung 1.3: Verallgemeinertes graphentheoretisches 4-Center mit 8.0 Kilometer Radius.

gewonnen werden können. Als Datenquelle wird dabei die OpenStreetMap [18] verwendet.
In den Kapiteln 3 und 4 behandeln wir das geometrische bzw. das graphentheoretische k-Center. Neben einer allgemeinen Komplexitätsbetrachtung stellen wir jeweils die Algorithmen vor, mit denen das Problem gelöst werden soll.
Anschließend analysieren wir in Kapitel 5 die gewonnenen Ergebnisse. Neben der Laufzeit der Algorithmen betrachten wir insbesondere, um wieviel schlechter eine Optimallösung einer Variante, aufgefasst als Lösung der anderen Variante, ist. Obwohl wir zeigen werden, dass in beiden Richtungen theoretisch beliebig schlechte Ergebnisse erzielt werden können, halten sich die Abweichungen bei realen Beispielen in Grenzen.
Schlussendlich ziehen wir im letzten Kapitel ein Fazit und führen noch einige Erweiterungen für die in dieser Diplomarbeit betrachtete Problemstellung an.

1.2 Mathematische Grundlagen

Um auf eine einheitliche Notation zurückgreifen zu können, definieren wir im Folgenden einige grundlegende Begriffe. Die Definitionen orientieren sich an [1], [2] und [8].

Definition 1.2 *(Minkowski-Raum)*
Sei $\mathbb{X}$ ein d-dimensionaler reellwertiger Vektorraum und $\|\cdot\|$ eine Norm auf $\mathbb{X}$. Dann heißt $\mathbb{M} = (\mathbb{X}, \|\cdot\|)$ *Minkowski-Raum*. Mit $\mathbb{B} = \{x \in \mathbb{X} : \|x\| \leq 1\}$ wird die zugehörige Einheitskugel bezeichnet.

Definition 1.3 *(Container, Containernorm)*
$C \subseteq \mathbb{R}^d$ heißt *Container*, wenn C konvex und kompakt und $0 \in int(C)$ ist. Für symmetrische Container C ist die *Containernorm* $\|\cdot\|_C$ definiert durch:

$$\|x\|_C := \min\{\rho \geq 0 : x \in \rho C\}$$

Definition 1.4 *(Graph)*
Ein 3-Tupel $G = (V, E, \nu)$ mit zwei endlichen Mengen V und E mit $V \cap E = \emptyset$ und einer Abbildung

$$\nu : E \to \binom{V}{2}$$

heißt *Graph*. Dabei wird V als *Knotenmenge* und E als *Kantenmenge* bezeichnet.
Ist die Abbildung ν injektiv, so heißt G *einfach*, andernfalls wird G auch als *Multigraph* bezeichnet. Ist die Abbildung ν surjektiv, so heißt G *vollständig*.
Ist zusätzlich eine Distanzfunktion $\delta : V \times V \to \mathbb{R}^+$ gegeben, so heißt G *gewichtet*.
Die Abbildung ν wird häufig nicht explizit angeführt, wenn die Bedeutung aus dem Zusammenhang klar ist.

Definition 1.5 *(Nachbarschaft)*
Sei $G = (V, E, \nu)$ ein Graph und $U \subseteq V$. Dann bezeichnet

$$\mathcal{N}(U) = \{v \in V \setminus U : \exists e \in E, \exists u \in U : \nu(e) = \{u, v\}\}$$

die *Nachbarschaft von* U.

Definition 1.6 *(Stabile Menge)*
Sei $G = (V, E, \nu)$ ein Graph. Eine Menge $C \subseteq V$ heißt *stabil*, wenn keine Kante $e \in E$ existiert mit $\nu(e) \subseteq C$.

Definition 1.7 *(Dominierende Menge)*
Sei $G = (V, E, \nu)$ ein Graph. Eine Menge $D \subseteq V$ heißt *dominierend*, wenn

$$D \cup \mathcal{N}(D) = V$$

gilt.

Definition 1.8 *(Weg)*
Sei $G = (V, E, \nu)$ ein Graph. Ein *s-t-Weg* ist eine Sequenz

$$(s = v_0, v_1, \ldots, v_l = t)$$

sodass für $i = 1, \ldots, l$ ein $e_i \in E$ existiert mit $\nu(e_i) = \{v_{i-1}, v_i\}$. Dabei wird l auch als *kombinatorische Länge* bezeichnet.

Definition 1.9 *(Zusammenhang)*
Sei $G = (V, E, \nu)$ ein Graph. Zwei Knoten s und t heißen *zusammenhängend*, wenn ein s-t-Weg existiert. G heißt *zusammenhängend*, wenn je zwei Knoten zusammenhängend sind.

Definition 1.10 *(Distanz)*
Sei $G = (V, E, \nu, \delta)$ ein gewichteter Graph und $s, t \in V$.
Zu einem s-t-Weg $w = (s = v_0, v_1, \ldots, v_l = t)$ wird mit

$$\delta(w) := \sum_{i=1}^{l} \delta(v_{i-1}, v_i)$$

die *Länge* definiert. Die Länge eines kürzesten s-t-Wegs

$$\text{dist}(s,t) := \min\{\delta(w) | w \text{ ist ein } s\text{-}t\text{-Weg}\}$$

wird als *Distanz* von s und t bezeichnet.

Um die Distanz zweier Punkte in einem gewichteten Graphen $G = (V, E, \nu, \delta)$ zu bestimmen, existieren einige bekannte Algorithmen. Wir werden uns im Folgenden häufig auf den Algorithmus von Dijkstra beziehen, der die Distanz von einem Startknoten zu allen anderen Knoten berechnet.

Algorithmus 1.11 *(Dijkstra)*
Input: $G = (V, E, \nu, \delta), v^* \in V$
Output: $\text{dist} : V \to \mathbb{R}^+ \cup \infty$

(1) **foreach** $v \in V$ **do**
(2) $\quad \text{dist}(v) = \infty$
(3) **end-for**
(4) $\text{dist}(v^*) = 0$
(5) $Q = \{v\}$
(6) **while** $Q \neq \emptyset$ **do**
(7) $\quad u = \text{argmin}\{\text{dist}(v) : v \in Q\}$
(8) $\quad$ **if** $\text{dist}(u) = \infty$ **then**
(9) $\quad\quad$ **output** G ist nicht zusammenhängend
(10) $\quad$ **end-if**
(11) $\quad Q = Q \setminus \{u\}$
(12) $\quad$ **foreach** $n \in \mathcal{N}(\{u\})$ **do**
(13) $\quad\quad \text{dist}(n) = \min\{\text{dist}(n), \text{dist}(u) + \delta(\{u, n\})\}$
(14) $\quad$ **end-for**
(15) **end-while**

In [8] wird gezeigt, dass Algorithmus 1.11 für zusammenhängende Graphen die kürzesten Wege vom Startknoten v^* zu allen Knoten aus V in einer Laufzeit von $\mathcal{O}(|V|^2 + |E|)$ berechnet. Durch die Verwendung eines Fibonacci-Heaps[1] lässt sich diese auf $\mathcal{O}(|V| \log |V| + |E|)$ reduzieren (siehe [4]). In Straßengraphen haben die Knoten nur eine beschränkte Anzahl an Nachbarn, sodass dort eine Laufzeit von $\mathcal{O}(|V| \log |V|)$ gilt.

[1] Ein Fibonacci Heap ist eine Datenstruktur bei der Elemente mit einer Priorität versehen gespeichert werden. Dabei kann sowohl das Einfügen als auch das Entfernen eines Elements höchster Prioriät in konstanter Laufzeit durchgeführt werden.

2 OpenStreetMap

Die OpenStreetMap (OSM) ist ein Projekt zur Erstellung einer freien geographischen Karte. Im Gegensatz zu kommerziellen Produkten wie zum Beispiel Google Maps ist hierbei ein Zugriff auf die Rohdaten und nicht nur auf die fertig gerenderte Karte möglich.

Für die vorliegende Arbeit wird die OpenStreetMap als Quelle benutzt, um Graphen zu erzeugen, die ein reales Straßennetz möglichst realistisch darstellen. Dazu wird in Abschnitt 2.1 in einem ersten Schritt aus den OSM-Daten ein Graph erzeugt, welcher das Verkehrsnetzwerk modelliert. Auf diesem wird anschließend in Abschnitt 2.2 eine Distanzmatrix berechnet, die als Grundlage für den Vergleich der k-Center-Varianten verwendet wird.

2.1 Erstellung eines Graphen aus OSM-Daten

Die OSM-Daten stehen in Form einer XML-Datei[2] zur Verfügung. Diese besteht aus drei wesentlichen Elementtypen, welche im Folgenden kurz erläutert werden.

Nodes

Nodes repräsentieren Punkte im OSM-Datenschema. Sie sind im *World Geodetic System 1984* (WGS84, siehe [17]) gespeichert. In diesem elliptischen Koordinatensystem wird die Erde durch ein Ellipsoid, das sogenannte Referenzellipsoid, angenähert und jeder Punkt auf der Erdoberfläche durch die Angabe des Längengrads (Longitude) und des Breitengrads (Latitude) eindeutig festgelegt. Der Nullmeridian entspricht dabei nicht dem Greenwich-Meridian, sondern verläuft ca. 100 Meter weiter östlich.

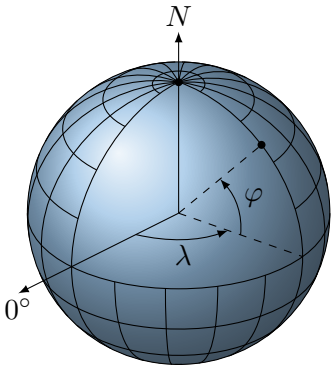

Abbildung 2.1: Definition von Longitude λ und Latitude φ.

[2] Unter der URL http://download.geofabrik.de/osm/ (Stand: 1. 5. 2010) befinden sich, aufgeteilt in einzelne Länder und Bundesländer, komprimierte XML-Dateien der OSM-Daten.

Nodes können unterschiedliche Aufgaben erfüllen: Einerseits gibt es Interpolationspunkte, welche einen Straßenverlauf festlegen, andererseits können Nodes auch ein Museum, eine Arztpraxis oder Ähnliches repräsentieren. Es ist daher möglich, zusätzliche Informationen in einem Node zu speichern.

Ways

Ways sind Sequenzen von Nodes, die beliebige Linien auf der Karte darstellen können. Neben der offensichtlichen Verwendung als Straßen, können Ways auch Flüsse, Landesgrenzen, etc. darstellen. Auch Gebiete werden durch geschlossene Ways eingegrenzt. Analog zu den Nodes können auch in Ways zusätzliche Informationen gespeichert werden.

In Abbildung 2.2 ist beispielhaft ein Ausschnitt der XML-Datei dargestellt, sowie eine gerenderte Karte dieses Gebiets. In der Karte wurden die Nodes markiert, um die Funktionsweise der Ways deutlich zu machen.

```
<osm version="0.6">
  <node id="2"lat="48.2669354"lon="11.6703136"/>
  <node id="3"lat="48.2668045"lon="11.6713725"/>
  <node id="4"lat="48.2667657"lon="11.6716525"/>
  <node id="5"lat="48.2667542"lon="11.6717361"/>
  <node id="6"lat="48.2670234"lon="11.6714143"/>
  <way id="10"visible="true">
    <nd ref="6"/>
    <nd ref="3"/>
    <tag k="highway"v="steps"/>
  </way>
  <way id="10"visible="true">
    <nd ref="2"/>
    <nd ref="3"/>
    <nd ref="4"/>
    <nd ref="5"/>
    <tag k="highway"v="unclassified"/>
    <tag k="maxspeed"v="50"/>
    <tag k="name"v="Lichtenbergstrasse"/>
  </way>
</osm>
```

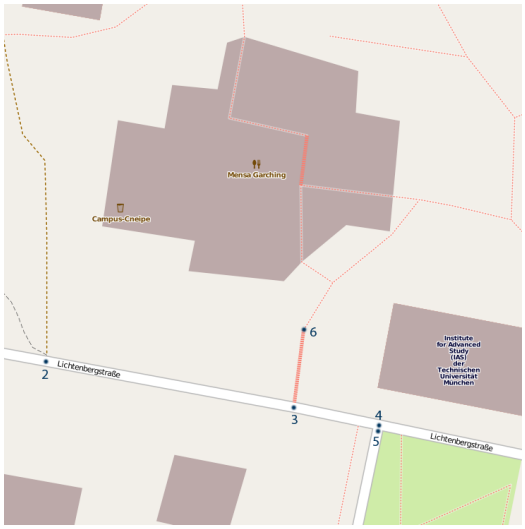

Abbildung 2.2: Ausschnitt einer XML-Datei und der daraus resultierenden Karte.

Relations

Relations gruppieren eine beliebige Anzahl von Nodes und Ways und weisen ihnen gemeinsame Eigenschaften zu. Damit ist es möglich räumlich getrennte Objekte, wie zum Beispiel Enklaven, zu verbinden. Für die Graphen, die im Folgenden aus der OpenStreetMap gewonnen werden sollen, werden Relations nicht berücksichtigt, da sie keine für die Entfernungsberechnung relevanten Informationen bereitstellen.

Um die WGS84-Koordinaten in einer zweidimensionalen Karte darzustellen, müssen diese zuerst in eine Ebene projiziert werden. Die OpenStreetMap verwendet hierfür die Mercator-Projektion.

Mercator-Projektion

Die Mercator-Projektion ist eine Zylinderprojektion, die in Nord-Süd-Richtung geeignet verzerrt wird, um eine winkeltreue Abbildung zu erhalten. Dabei wird eine große Verzerrung in polaren Regionen in Kauf genommen, sodass eine Anwendung jenseits des 70. Breitengrads nicht mehr sinnvoll ist. So entspricht beispielsweise die Ausdehnung Grönlands auf einer Mercator-Karte in etwa der Afrikas, obwohl es flächenmäßig 13,7 mal kleiner ist (siehe [13]). Aus gegebenen WGS84-Koordinaten mit Längengrad λ und Breitengrad φ lassen sich die auf den Erdradius normierten x,y-Koordinaten folgendermaßen berechnen:

$$\begin{aligned} x &= \lambda - \lambda_0 \\ y &= \operatorname{arctanh}(\sin\varphi) \end{aligned}$$

$\lambda_0 \in \mathbb{R}$ ist dabei ein ausgezeichneter Längengrad, das sogenannte Kartenzentrum[3].
Mit der inversen Mercator-Projektion, der sogenannten Gudermannfunktion, lassen sich x,y-Koordinaten in WGS84-Koordinaten zurückrechnen:

$$\begin{aligned} \lambda &= x + \lambda_0 \\ \varphi &= \arcsin(\tanh y) \end{aligned}$$

Im Folgenden sollen die XML-Daten in einen ungerichteten Graphen überführt werden. Dabei liegt es nahe, Nodes auf Knoten abzubilden und Kanten entsprechend der Ways einzufügen. Da die bei naivem Vorgehen aus der OpenStreetMap gewonnenen Graphen schon im Falle Bayerns mehr als 8,6 Millionen Knoten besitzen, müssen zusätzliche Verfahren angewandt werden, um Entfernungen vernünftig berechnen zu können.

Zuerst werden die für die Graphenberechnung relevanten Daten eines Gebiets in einer SQL-Datenbank gespeichert. Um die Knotenanzahl zu reduzieren, werden anschließend in einem Vorberechnungsschritt, wie in [6] vorgestellt, alle Zwischenknoten mit Grad zwei aus dem Graphen entfernt und stattdessen die beiden Nachbarknoten durch eine direkte Kante verbunden, deren Kantengewicht auf die Summe der beiden urspünglichen Kanten gesetzt wird (vergleiche Abbildung 2.3). Dabei werden eventuell eingetragene Richtungsinformationen der Kanten ignoriert und grundsätzlich ungerichtete Kanten eingetragen.

Nach diesem Vorberechnungsschritt steht ein ungerichteter, gewichteter Graph $G = (V, E, \delta)$ zur Verfügung, wobei $V \subseteq \mathbb{R}^2$ der Menge aller Kreuzungen im Straßengraphen entspricht und für jede Straße zwischen zwei Kreuzungen eine Kante eingefügt wurde, deren Kantenlänge der tatsächlichen Weglänge entspricht.

Trotz Vorberechnungsschritt sind die so erzeugten Graphen im Allgemeinen noch sehr groß. So hat ein Graph von Bayern immer noch mehr als zwei Millionen Knoten. Daher werden wir nun einige Strategien diskutieren, die Problemgröße für die Entfernungsberechnung zu reduzieren.

2.2 Entfernungsberechnung im OSM-Graphen

Gegeben sei ein gewichteter Graph $G = (V, E, \delta)$ mit $|V| = n$ und $|E| = m$, sowie eine ausgezeichnete Knotenmenge $S := \{v_i : 1 \leq i \leq s\} \subseteq V$. Unser Ziel ist es, den kürzesten

[3] In der OpenStreetMap wird λ_0 auf 0 gesetzt.

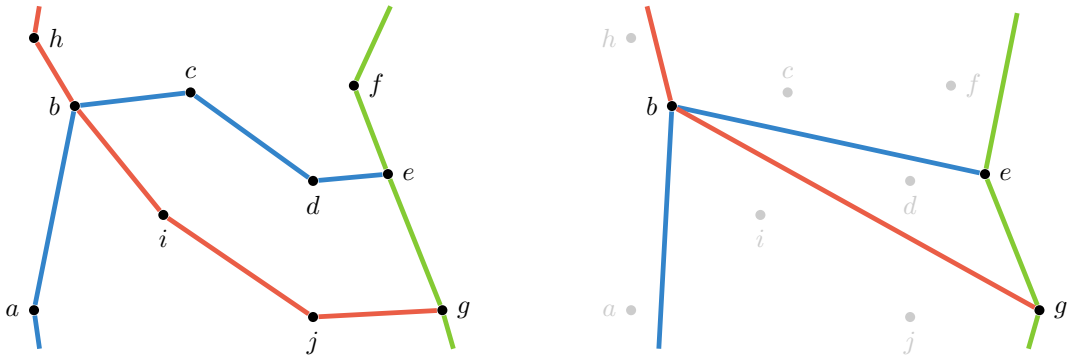

Abbildung 2.3: Vorberechnungsschritt für den OSM-Graphen.

Weg zwischen allen Knotenpaaren (v_i, v_j) aus S zu bestimmen. Dazu stellen wir nun einen Algorithmus vor, der als Basis den Dijkstra-Algorithmus 1.11 verwendet.

Algorithmus 2.1 *(Mehrfacher Dijkstra)*
Input: $G = (V, E, \delta), S \subseteq V$
Output: $D \in \mathbb{R}^{s \times s}$

(1) **foreach** $v_i \in S$ **do**
(2) dist $= dijkstra(G, v_i)$
(3) **foreach** $v_j \in S$ **do**
(4) $D(i, j) = \text{dist}(v_j)$
(5) **end-for**
(6) **end-for**

Lemma 2.2 *(Korrektheit und Laufzeit von Algorithmus 2.1)*
Wenn G zusammenhängend ist, findet Algorithmus 2.1 in $\mathcal{O}(s \cdot n \log(n))$ die minimalen Abstände zwischen allen Knotenpaaren aus S.

Beweis Die Korrektheit folgt direkt aus dem Dijkstra-Algorithmus, da die Kantengewichte im OSM-Graphen positiv sind. Die s-malige Ausführung des Dijkstra-Algorithmus bewirkt eine Laufzeit von $\mathcal{O}(s \cdot n \log(n))$. Hinzu kommt die Speicherung in der Distanzmatrix D mit einer Laufzeit von $\mathcal{O}(s^2)$. □

Bemerkung 2.3 *(Wahl des Basis-Algorithmus)*
Die Wahl fiel auf den Dijkstra-Algorithmus als Basis für Algorithmus 2.1, da damit pro Durchlauf eine ganze Zeile der Distanzmatrix D berechnet werden kann.
Als mögliche Alternative betrachten wir den A^*-Algorithmus (siehe [19]). Dieser verwendet eine Heuristik um die Entfernungsberechnung zu beschleunigen. Als einfach zu berechnende Heuristik bietet sich im Falle von OSM-Graphen die Luftlinienentfernung an. Da der A^*-Algorithmus jedoch nur die Distanz von einem Startknoten zu einem Zielknoten berechnet, wird pro Durchlauf nur ein Eintrag der Distanzmatrix bestimmt. Die theoretische Laufzeit beträgt daher nur noch $\mathcal{O}(s^2 \cdot n \log(n))$. Auch in realen Anwendungen ist die Laufzeit wesentlich schlechter.

In Abbildung 2.4 sind die Laufzeiten von Algorithmus 2.1 mit beiden Basisalgorithmen auf einem handelsüblichen Laptop[4] gegenübergestellt. Zu verschiedenen Standortmengen wurde die Distanzmatrix mit beiden Algorithmen berechnet. Der zugrunde liegende Graph besteht aus 61.427 Knoten und 63.935 Kanten. Der in grün dargestellte A^*-Algorithmus ist als Basis durchweg langsamer als der Dijkstra-Algorithmus in blau. Mit zunehmender Standortanzahl wächst das Verhältnis der Laufzeiten linear.

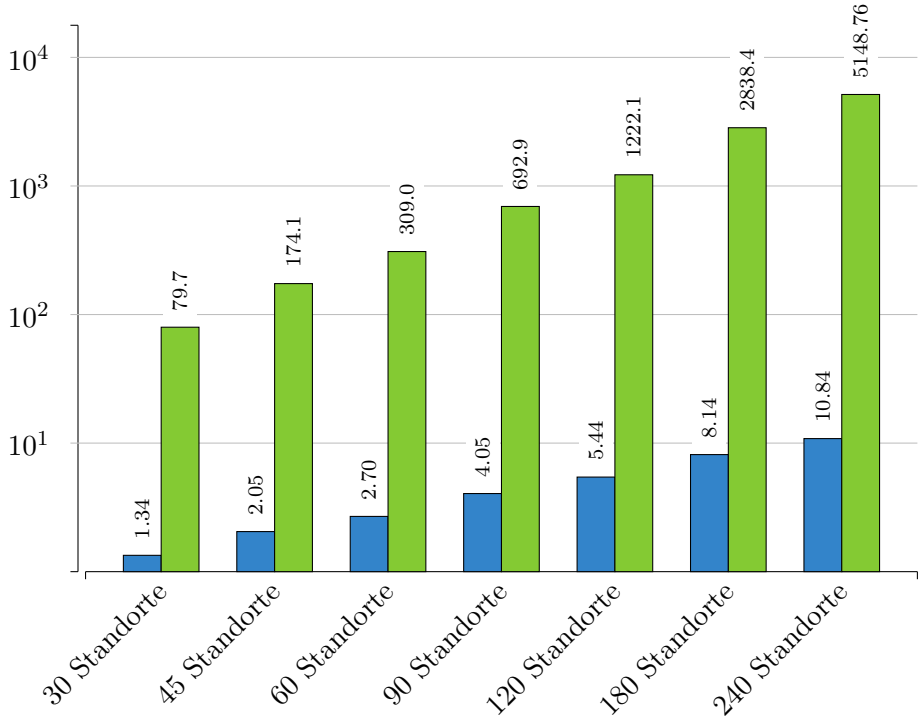

Abbildung 2.4: Vergleich der Laufzeiten in Sekunden des Dijkstra-Algorithmus (blau) und des A^*-Algorithmus (grün) als Basis von Algorithmus 2.1.

Bemerkung 2.4 *(Floyd-Warshall-Algorithmus)*
Der Floyd-Warshall-Algorithmus (siehe [8]) berechnet mit einer Laufzeit von $\mathcal{O}(n^3)$ die paarweisen Entfernungen zwischen allen Knoten aus V. Jedoch selbst für $S = V$ hat Algorithmus 2.1 eine bessere Laufzeit. Da S im Allgemeinen wesentlich kleiner ist als V und der Floyd-Warshall-Algorithmus zudem nicht an die veränderte Problemstellung angepasst werden kann, ist dieser hier nicht praktikabel.

Bemerkung 2.5 *(Eigenschaften der Distanzmatrix D)*
Die in Algorithmus 2.1 berechnete Distanzmatrix D ist symmetrisch und die durch D induzierte Gewichtsfunktion

$$\begin{aligned} \delta_D \ : \ & S \times S \to \mathbb{R}^+ \\ & (v_i, v_j) \mapsto D(i,j) \end{aligned}$$

erfüllt die Dreiecksungleichung.

[4]Intel Pentium T3200 (2.0 GHz), 2GB DDR2

Für eine große Anzahl an Knoten ist eine direkte Anwendung von Algorithmus 2.1 auf einen OSM-Graphen trotz der theoretisch guten Laufzeit nicht sinnvoll möglich. Betrachten wir beispielsweise die 233 in der OpenStreetMap eingetragenen Aldi-Filialen in Bayern, so dauert die Entfernungsberechnung mit Algorithmus 2.1 schon mehr als 16 Minuten. Mit zunehmender Knotenzahl steigt auch die Laufzeit linear.
Um trotzdem in vernünftiger Zeit auch zu größeren Knotenmengen ein Ergebnis zu erhalten, diskutieren wir nun zwei Ansätze, die jeweils nur einen Ausschnitt des Graphen betrachten.

2.2.1 Einschränkung des Gebiets

Da die Knoten im OSM-Graphen Punkte in der Ebene sind, besteht eine Möglichkeit den Graphen einzuschränken darin, nur Knoten zu betrachten, die sich in einem bestimmten Gebiet befinden. Sämtliche Kanten, die komplett oder teilweise außerhalb liegen werden ignoriert.
Im Folgenden werden Kreisscheiben als Gebiete betrachtet. Sei dazu

$$\mathscr{C} := \{c + \rho\mathbb{B}_2^2 : c \in \mathbb{R}^2, \rho > 0\}$$

die Menge aller Kreisscheiben in $\mathbb{R}^2$. Dabei bezeichnet c den Mittelpunkt, ρ den Radius und $\mathbb{B}_2^2 := \{x \in \mathbb{R}^2 : \|x\|_2 \leq 1\}$ die euklidische Einheitskreisscheibe. Zu einer gegebenen Punktemenge $S \subseteq \mathbb{R}^2$ sei $\mathcal{B}(S)$ die kleinste Kreisscheibe, die alle Punkte aus S enthält.

Definition 2.6 *(Eingeschränkter Graph)*
Zu einer gegebenen Kreisscheibe C wird der auf C eingeschränkte Graph $G|_C = (V|_C, E|_C)$ durch

$$\begin{aligned} E|_C &:= \{e \in E : \nu(e) \subseteq C\} \\ V|_C &:= \{v \in V : \exists e \in E|_C, v \in \nu(e)\} \end{aligned}$$

definiert.

Lemma 2.7 *(Algorithmus 2.1 in eingeschränkten Gebieten)*
Sei $G = (V, E)$ ein zusammenhängender Graph und $S \subseteq V$. Die kleinste S überdeckende Kreisscheibe sei gegeben durch $\mathcal{B}(S) = c^* + \rho^*\mathbb{B}_2^2$. Zu $\alpha \geq 1$ sei

$$\mathcal{B}_\alpha(S) := c^* + \alpha \cdot \rho^*\mathbb{B}_2^2$$

die um α skalierte Kreisscheibe $\mathcal{B}(S)$ und D_α das Ergebnis von Algorithmus 2.1 zur Eingabe $(G|_{\mathcal{B}_\alpha(S)}, S)$, also des auf $\mathcal{B}_\alpha(S)$ eingeschränkten Graphen.
Falls

$$\max_{i,j} D_\alpha(i,j) \leq 2 \cdot (\alpha - 1) \cdot \rho^* \tag{2.1}$$

ist, so gilt $D_\alpha = D_{\alpha'}$ $\forall \alpha' \geq \alpha$, das heißt, die Einträge in D_α entsprechen den kürzesten Entfernungen im nicht eingeschränkten Graphen.

Beweis Offensichtlich ist die Luftlinienentfernung zweier Punkte eine untere Schranke für den kürzesten Weg, das heißt $\|v_i - v_j\|_2 \leq D(i,j)$. Der Dijkstra-Algorithmus findet den optimalen Weg innerhalb der Kreisscheibe $\mathcal{B}_\alpha(S)$. Falls auf dem Gesamtgraphen ein kürzerer Weg existiert, muss dieser $\mathcal{B}_\alpha(S)$ an mindestens einer Stelle verlassen. Da sich jedoch nach Konstruktion sowohl der Startknoten als auch der Zielknoten in $\mathcal{B}(S)$ befinden, muss dieser global kürzeste Weg mindestens zweimal die Entfernung $(\alpha - 1) \cdot \rho^*$ zurücklegen. Ist dieser Wert für alle Knotenpaare größer als der gefundene Pfad, so kann ein kürzester v_i-v_j-Weg $\mathcal{B}_\alpha(S)$ nicht verlassen. Daraus folgt die Korrektheit aller kürzesten Wege auch auf dem nicht eingeschränkten Graphen. □

Bemerkung 2.8
Anstelle von Kreisscheiben kann der Graph auch auf Quadrate eingeschränkt werden. Mit $\mathbb{B}_2^2 \subseteq \mathbb{B}_\infty^2$ lässt sich eine zu Lemma 2.7 analoge Aussage zeigen. Dies nutzen wir für die Implementierung des Algorithmus aus, da sich die Datenbankabfragen dadurch beschleunigen lassen.

2.2.2 Einschränkung der Kanten

Wir wollen nun noch einen zweiten Ansatz betrachten, die Größe des OSM-Graphen zu reduzieren. Dieser verwendet die Angabe des Typs einer Straße, der in den OSM-Daten gespeichert ist. Damit ist es möglich nur Straßen eines bestimmten Typs (Autobahn, Landstraße, etc.) zu betrachten. Algorithmus 2.1 kann weiterhin angewendet werden, solange die Standortmenge S auch im eingeschränkten Graphen zusammenhängend ist.
Im Folgenden werden die Straßentypen in Level eingeteilt, wobei Level 0 dem gesamten OSM-Graphen entspricht und mit zunehmendem Level die Kantenmenge immer weiter ausgedünnt wird.

Definition 2.9 *(*k*-Levelgraph)*
Zu $k \in \mathbb{N}_0$ ist ein k-*Levelgraph* ein 4-Tupel $G = (V, E, \nu, \lambda)$ mit einem Multigraphen (V, E, ν) und einer Levelfunktion $\lambda : E \to \{0, 1, \ldots, k-1\}$. Der auf Level $i \in \{0, 1, \ldots, k-1\}$ eingeschränkte Graph G_i mit Knoten V_i und Kanten E_i ist durch

$$\begin{aligned} E_i &:= \{e \in E : \lambda(e) \leq i\} \\ V_i &:= \{v \in V : \exists e \in E_i : v \in \nu(e)\} \\ G_i &:= (V_i, E_i, \nu|E_i) \end{aligned}$$

definiert. Ist zusätzlich eine Kantengewichtsfunktion $\delta : E \to \mathbb{R}^+$ gegeben, so heißt der k-Levelgraph *gewichtet*.

Bemerkung 2.10 *(Knoten mit Grad zwei)*
Trotz des Vorberechnungsschritts enthält ein auf Level $i > 0$ eingeschränkter Graph viele Knoten vom Grad zwei, denn alle durch die Kreuzung einer Level i-Straße mit einer Straße niedrigeren Levels erzeugten Knoten haben in G_i Grad zwei.

Wir stellen nun einen Algorithmus vor, der die Überlegungen aus den Abschnitten 2.2.1 und 2.2.2 zu einem hierarchischen Ansatz verbindet. Dieser liefert zu einem k-Levelgraphen $G = (V, E, \nu, \lambda)$ und $S \subseteq V$ einen Graphen $G' = (V', E')$ zurück, wobei $S \subseteq V' \subseteq V$

und $E' \subseteq E$ gilt. Der Graph G' entsteht, indem ein global grober Graph lokal um die Standorte aus S verfeinert wird. Anschließend kann G' für die Entfernungsberechnung mit Algorithmus 2.1 verwendet werden.
Um den Graphen im $(i-1)$-ten Level in der Umgebung eines Standorts s einzuschränken, benötigen wir Punkte aus dem i-ten Level, die sich in der Nähe von s befinden. Dazu verwenden wir die nach Luftlinienentfernung nächsten Punkte.

Definition 2.11 *(*i*-te Umgebung)*
Seien $G = (V, E, \nu, \lambda)$ ein k-Levelgraph mit $V \subseteq \mathbb{R}^2$ und $0 \leq i < k$ gegeben. Zu $v \in V$ und $n_i(v) := \min\{\|v - w\|_2 : w \in V_i\}$ wird durch

$$N_i(v) := \{w \in V_i : \|v - w\|_2 \leq n_i(v)\}$$

die i-*te Umgebung* definiert.

Bemerkung 2.12
Falls $v \in V_{i^*}$, dann gilt für alle $i \leq i^*$ wegen $n_i(v) = 0$, dass $N_i(v) = \{v\}$. Insbesondere ist für alle $\alpha \geq 1$ der auf $\mathcal{B}_\alpha(N_i(v))$ eingeschränkte Graph $G|_{\mathcal{B}_\alpha(N_i(v))}$ leer.

Algorithmus 2.13 *(Hierarchische Verfeinerung)*
Input: k-Levelgraph $G = (V, E, \nu, \lambda)$, $S \subseteq V$, $\alpha \geq 1$
Output: $G' = (V', E')$

(1) $i := k - 1$
(2) $G' = (V_i|_{\mathcal{B}_\alpha(S)}, E_i|_{\mathcal{B}_\alpha(S)})$
(3) **while** $i \geq 0$ **do**
(4) **foreach** $s \in S$ **do**
(5) $G' := G' \cup (V_i|_{\mathcal{B}_\alpha(N_i(s))}, E_i|_{\mathcal{B}_\alpha(N_i(s))})$
(6) **end-for**
(7) $i := i - 1$
(8) **end-while**

In Abbildung 2.5 ist ein Graph abgebildet, welcher von Algorithmus 2.13 für $\alpha = 1,6$ erzeugt wird. Die Kanten, die in G' erhalten bleiben, sind farbig nachgezeichnet und die betrachteten Gebiete durch verschiedene Helligkeitsstufen dargestellt.
Man beachte, dass zwar bei den oberen drei Knoten die kürzeste Verbindung gefunden wird, bei den unteren beiden hingegen nicht. Der linke untere Knoten ist sogar mit keinem anderen Knoten verbunden.
Selbst wenn die Knoten im von Algorithmus 2.13 erzeugten Graphen verbunden bleiben, werden wir im folgenden Lemma sehen, dass die Länge des gefundenen Wegs beliebig stark von der eigentlich kürzesten Weglänge abweichen kann.

Lemma 2.14
Für $k \geq 1$ existiert für alle $\alpha \geq 1$ und für ein beliebiges $\xi > 1$ ein gewichteter k-Levelgraph $G = (V, E, \nu, \lambda, \delta)$ und $S \subseteq V$, sodass für alle von Algorithmus 2.13 zur Eingabe (G, S, α) berechnete Graphen G' und für alle u,$v \in S$

$$\frac{\text{dist}_{G'}(u, v)}{\text{dist}_G(u, v)} \geq \xi$$

gilt, wobei $\text{dist}_G(u, v)$ die Länge eines kürzesten Weges von u nach v im Graphen G bezeichnet.

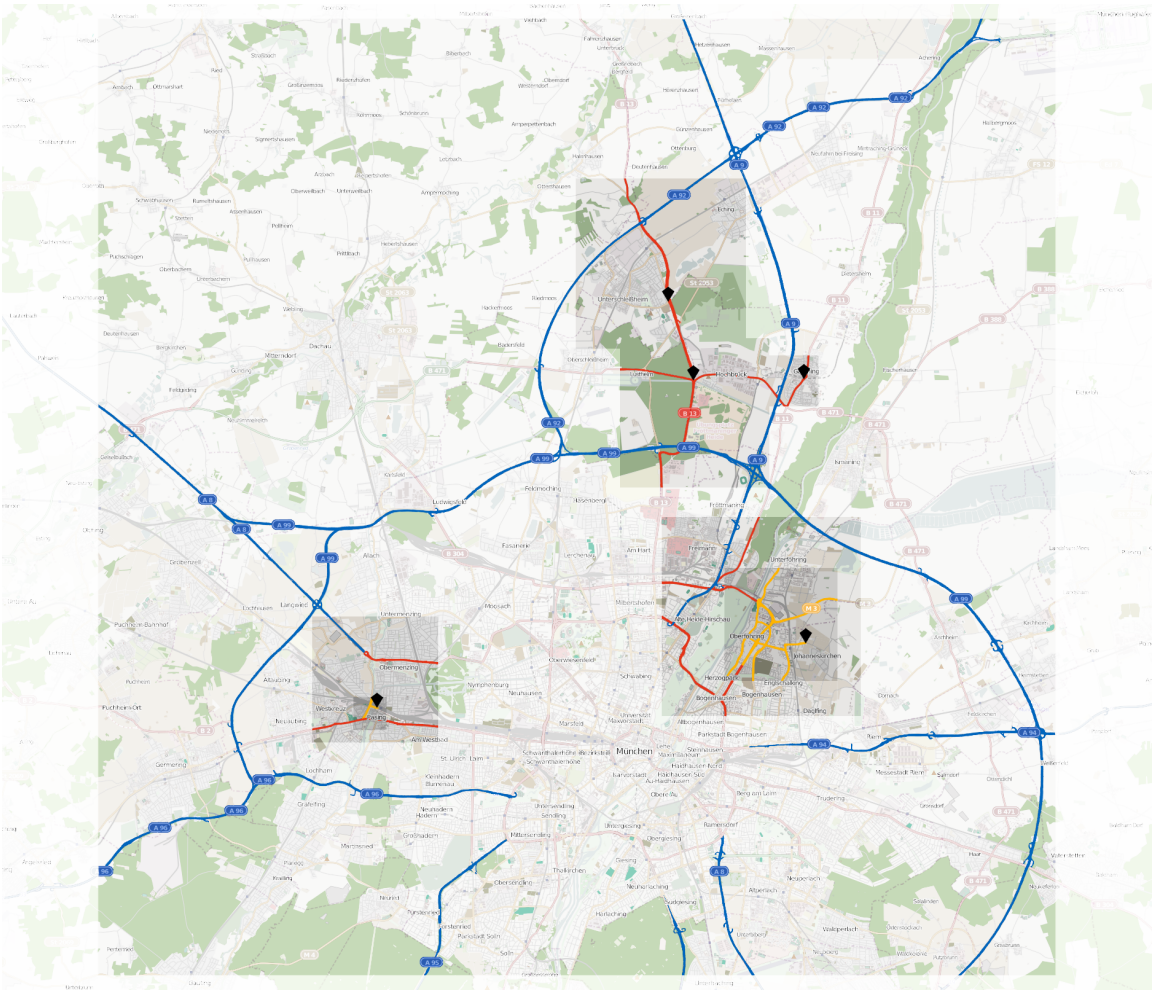

Abbildung 2.5: Hierarchische Verfeinerung von Algorithmus 2.13.

Beweis Es sei $\xi > 1$ beliebig. Wir betrachten den Graphen G aus Abbildung 2.6 mit Knotenmenge $V = \{u, v, w\}$. Wegen $k \geq 1$ existieren mindestens zwei Level, sodass die Kanten folgendermaßen eingefügt werden können:
Im 1. Level seien die blauen Kanten $\{u, w\}$ und $\{v, w\}$, jeweils mit Länge 1 vorhanden, im 0. Level zusätzlich die grüne Kante $\{u, v\}$ mit Länge $\varepsilon \leq \frac{2}{\xi}$.

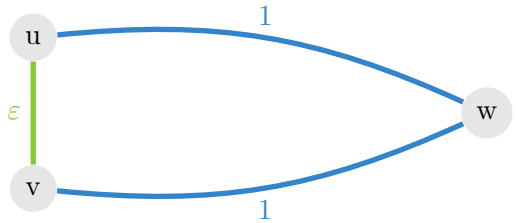

Abbildung 2.6: Problematische Instanz für Algorithmus 2.13.

Algorithmus 2.13 besetzt in der Initialisierungsphase den Graphen G' mit der kompletten

Knotenmenge und den blauen Kanten:

$$\begin{aligned} V' &= \{u, v, w\} \\ E' &= \{\{u, w\}, \{v, w\}\} \end{aligned}$$

Da sowohl u als auch v in V' liegen, gilt nach Bemerkung 2.12:

$$E_0|_{\mathcal{B}_\alpha(N_0(u))} = \emptyset = E_0|_{\mathcal{B}_\alpha(N_0(v))}$$

Somit wird die grüne Kante $\{u, v\}$ in keinem Verfeinerungsschritt eingefügt.
Für $S = \{u, v\}$ folgt also:

$$\frac{\text{dist}_{G'}(u, v)}{\text{dist}_G(u, v)} = \frac{2}{\varepsilon} \geq \xi$$

□

Das Lemma 2.14 zugrunde liegende Problem ist nicht nur akademischer Natur, sondern findet sich auch in realen OSM-Graphen.

Abbildung 2.7: Kürzeste Entfernung zwischen Dornbirn und Lustenau auf eingeschränkten Graphen.

Beispiel 2.15
Betrachtet man die kürzeste Entfernung zwischen Dornbirn und Lustenau, so ist der in Abbildung 2.7 blau eingezeichnete kürzeste Weg auf dem auf Autobahnen eingeschränkten Graphen 669.0 Kilometer lang. Der rot eingezeichnete kürzeste Weg auf Bundesstraßen hat hingegen nur noch eine Länge von 6.0 Kilometer und ist damit über 111 mal kürzer.

Bemerkung 2.16
Die Aussage von Lemma 2.14 gilt weiterhin, wenn für die Quadrate um die Standorte ein Mindestradius λ gefordert wird. Man betrachte dazu den Graphen aus Abbildung 2.8. Hier folgt analog, dass beliebig große Abweichungen vom tatsächlich kürzesten Weg möglich sind.

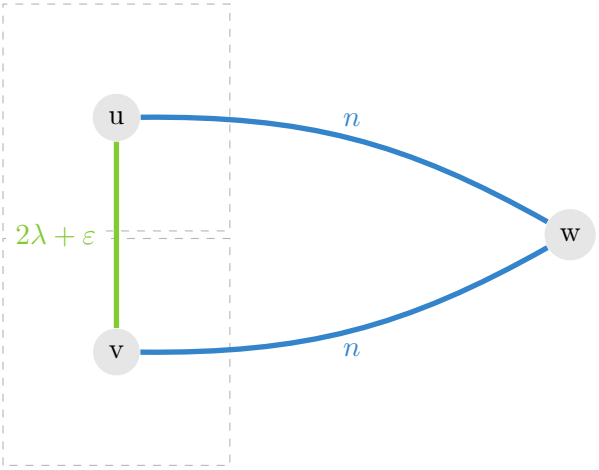

Abbildung 2.8: Problematische Instanz für Algorithmus 2.13 mit Mindestdurchmesser.

Für $n \geq \xi(\lambda + \frac{\varepsilon}{2})$ gilt nämlich wieder

$$\begin{aligned} \frac{\text{dist}_{G'}(u,v)}{\text{dist}_G(u,v)} &= \frac{2n}{2\lambda + \varepsilon} \\ &\geq \frac{2\xi(\lambda + \frac{\varepsilon}{2})}{2\lambda + \varepsilon} \\ &= \xi. \end{aligned}$$

Die einzige Möglichkeit diese Problematik zu umgehen, wäre zu fordern, dass sich alle Quadrate überlappen müssen. Da dann jedoch der gesamte Graph niedrigeren Levels betrachtet wird, führt dies zu keiner Reduktion der Größe des Graphen.

In diesem Kapitel haben wir gezeigt, wie wir aus den OSM-Daten einen gewichteten Graphen erzeugen und daraus die paarweisen Abstände aller Knoten einer Standortmenge in einer Distanzmatrix speichern können. Außerdem haben wir zwei Ansätze vorgestellt, die Problemgröße zu reduzieren, um auch bei großen Graphen ein Ergebnis berechnen zu können. Bei der Kombination der beiden Ansätze zu einem hierarchischen Verfahren wurde festgestellt, dass sehr große Fehler auftreten können. Daher wurden die im Rahmen dieser Diplomarbeit erzeugten Datensätze auf jeweils nur einem Level des Graphen ohne hierarchische Verfeinerung berechnet.

3 Geometrisches k-Center

Das geometrische k-Center ist ein Standortproblem, bei dem die Standorte Punkte in einem Minkowski-Raum (siehe Definition 1.2) sind und die Zentren beliebig im Raum verteilt werden können. Die Distanz zwischen Zentren und Standorten wird dabei durch die Norm festgelegt. Die folgende Definition findet sich in ähnlicher Form in [2].

Problem 3.1 *(Geometrisches* k*-Center)*
Gegeben sei ein Minkowski-Raum $\mathbb{M} = (\mathbb{X}, \|\cdot\|)$ mit Einheitskugel $\mathbb{B}$, eine Menge $S \subseteq \mathbb{M}$, sowie $k \in \mathbb{N}$. Zu $C \subseteq \mathbb{M}$ ist durch

$$\rho(C) := \min\{\rho \geq 0 : S \subseteq \bigcup_{c \in C} c + \rho\mathbb{B}\} \tag{3.1}$$

der *Radius von* C definiert. Das Problem, eine höchstens k-elementige Menge C mit minimalem Radius zu finden, heißt GEOMETRISCHES k-CENTER.

In Abbildung 3.1 ist eine Lösung eines geometrischen 5-Centers eingezeichnet. Die Zentren sind schwarz markiert, während die Standorte zum jeweils gleichen nächsten Zentrum in einer Farbe dargestellt sind.

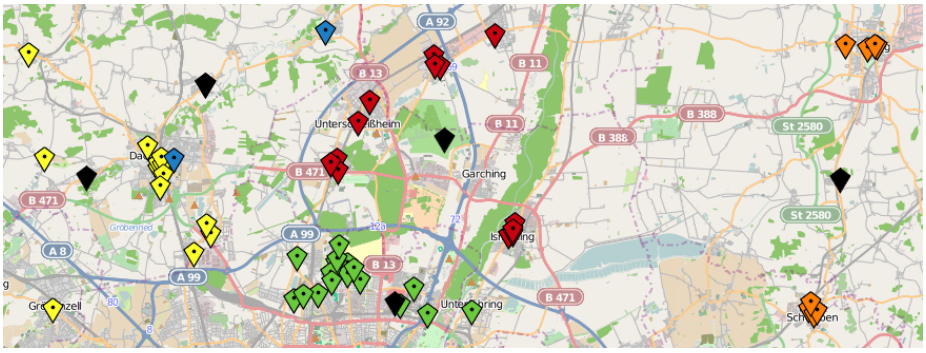

Abbildung 3.1: Beispiel eines geometrischen 5-Centers.

Bemerkung 3.2
Zu einer Menge $C \subseteq \mathbb{M}$ mit $|C| < k$ und für beliebige Punkte $x_1, \ldots, x_{k-|C|} \in \mathbb{M}$ gilt

$$\rho(\bigcup_{i=1}^{k-|C|} x_i \cup C) \leq \rho(C).$$

Eine Lösung mit weniger als k Zentren kann also unter Beibehaltung des Radius mit beliebigen Punkten aufgefüllt werden. Insbesondere kann in Problem 3.1 auch gefordert werden, dass C genau k Elemente enthält.

Bemerkung 3.3
Das Problem 3.1 mit dem Minkowski-Raum $\mathbb{M} = (\mathbb{R}^d, \|\cdot\|_2)$ wird auch als (d-dimensionales) EUKLIDISCHES k-CENTER bezeichnet.

Problem 3.4 *(Entscheidungsvariante des euklidischen* k*-Centers)*
Das Problem ENTSCHEIDUNGSVARIANTE DES EUKLIDISCHEN k-CENTERS besteht darin, zur Eingabe ($d \in \mathbb{N}$, $S \subseteq \mathbb{R}^d$, $k \in \mathbb{N}$) zu entscheiden, ob ein euklidisches k-Center mit maximalem Radius eins existiert.

3.1 Komplexität des geometrischen k-Centers

Megiddo hat in [14], durch eine Reduktion von 3-SAT (Definition siehe [5]) auf die ENTSCHEIDUNGSVARIANTE DES EUKLIDISCHEN 2-CENTERS, dessen $\mathbb{NP}$-Vollständigkeit gezeigt.

Satz 3.5
Die ENTSCHEIDUNGSVARIANTE DES EUKLIDISCHEN 2-CENTERS ist $\mathbb{NP}$-vollständig.

Beweis Das Problem ist in $\mathbb{NP}$, da durch die Angabe der Zentren einer Ja-Instanz diese in polynomieller Laufzeit $\mathcal{O}(k \cdot d \cdot |S|)$ verifiziert werden kann.
Es bezeichne $e^i = (\delta_{ij})_j \in \mathbb{R}^d$ den i-ten Einheitsvektor in $\mathbb{R}^d$. Für beliebige Dimension d und beliebige Vektoren $b \in \{\pm 1\}^d$ lässt sich die Menge $\{b_i e^i : i = 1, \ldots, d\}$ durch eine Kugel mit Radius $\rho_d = \sqrt{1 - \frac{1}{d}}$ und dem arithmetischen Mittel der Vektoren als Mittelpunkt überdecken:

$$\begin{aligned}\left\| b_i e^i - \frac{1}{d}\sum_{j=1}^{d} b_j e^j \right\|_2^2 &= b_i^2(1-\frac{1}{d})^2 + \sum_{\substack{j=1\\ j\neq i}}^{d}(\frac{b_j}{d})^2 \\ &= 1 - \frac{2}{d} + \frac{1}{d^2} + \frac{d-1}{d^2} \\ &= 1 - \frac{1}{d} \\ &= \rho_d^2\end{aligned}$$

Wir wollen eine Instanz der Entscheidungsvariante von 3-SAT mit Klauseln K_j, $j = 1, \ldots, m$ über jeweils paarweise verschiedenen Literalen aus $\{l_1, \bar{l}_1, \ldots, l_n, \bar{l}_n\}$ mit Hilfe des euklidischen 2-Centers lösen. Dazu betrachten wir den Raum $\mathbb{R}^d$ mit $d = n + 1$. Wähle $\alpha \in \mathbb{R}^+$ so, dass

$$12\alpha^2 - \frac{4\alpha}{d} + \frac{1}{d} < \rho_d^2 < 12\alpha^2 + \frac{1}{d} \tag{3.2}$$

erfüllt ist. Sei $P = \{p^j \in \mathbb{R}^d : j = 1, \ldots, m\}$, wobei

$$p_i^j = \begin{cases} \alpha, & \text{wenn } l_i \in K_j \\ -\alpha, & \text{wenn } \bar{l}_i \in K_j \\ 3\alpha, & \text{wenn } i = d \\ 0, & \text{sonst} \end{cases}$$

gesetzt wird. Wir zeigen nun, dass genau dann ein euklidisches 2-Center mit Radius ρ_d existiert, das die Punkte $P \cup \{\pm e^i : i = 1, \ldots, d\}$ überdeckt, wenn 3-SAT erfüllbar ist.

Angenommen `3-SAT` ist erfüllbar, dann betrachten wir eine Wahrheitsbelegung und setzen den Vektor $b \in \{\pm 1\}^d$ auf

$$b_i = \begin{cases} -1, & \text{wenn } l_i \text{ falsch ist} \\ +1, & \text{sonst} \end{cases}$$

(es gilt: $b_{n+1} = 1$). Damit gilt mit Gleichung (3.2) für alle $p^j \in P$, wobei τ die Anzahl an wahren Variablen in K_j bezeichnet:

$$\begin{aligned} \left\| p^j - \frac{1}{d} \sum_{i=1}^{d} b_i e^i \right\|_2^2 &= \tau(\alpha - \frac{1}{d})^2 + (3-\tau)(\alpha + \frac{1}{d})^2 + \frac{n-3}{d^2} + (3\alpha - \frac{1}{d})^2 \\ &= 12\alpha^2 - \frac{4\tau\alpha}{d} + \frac{1}{d} \\ &\leq 12\alpha^2 - \frac{4\alpha}{d} + \frac{1}{d} \\ &< \rho_d^2 \end{aligned}$$

Es lässt sich also $P \cup \{b_i e^i : i = 1, \ldots, d\}$ durch eine Kugel überdecken und die restlichen Einheitsvektoren $\{-b_i e^i : i = 1, \ldots, d\}$ durch eine zweite.

Angenommen es existiert ein 2-Center mit Radius ρ_d. Um alle Einheitsvektoren durch 2 Kugeln $\mathcal{B}_1$, $\mathcal{B}_2$ mit Radius ρ_d zu überdecken, müssen die Zentren in $c_1 = \frac{1}{d}\sum_{i=1}^{d} b_i e^i$ und $c_2 = \frac{1}{d}\sum_{i=1}^{d} -b_i e^i$ für ein $b \in \{\pm 1\}^d$ liegen. Dabei gelte ohne Einschränkung $b_{n+1} = +1$.
Wir belegen die Variable l_i mit wahr $\Leftrightarrow b_i = 1$. Angenommen es gäbe eine Klausel K_j, die nicht erfüllt ist, dann wäre sowohl

$$\begin{aligned} \left\| p^j - c_1 \right\|_2^2 &= 3(\alpha - \frac{1}{d})^2 + \frac{n-3}{d^2} + (3\alpha + \frac{1}{d})^2 \\ &= 12\alpha^2 + \frac{1}{d} \\ &> \rho_d^2 \end{aligned}$$

als auch

$$\begin{aligned} \left\| p^j - c_2 \right\|_2^2 &= 3(\alpha + \frac{1}{d})^2 + \frac{n-3}{d^2} + (3\alpha - \frac{1}{d})^2 \\ &= 12\alpha^2 + \frac{1}{d} \\ &> \rho_d^2. \end{aligned}$$

Der Punkt p^j wäre damit weder in $\mathcal{B}_1$ noch in $\mathcal{B}_2$. Also muss jede Klausel erfüllt sein. □

Bemerkung 3.6
Der Beweis von Satz 3.5 kann leicht für beliebige $k > 2$ verallgemeinert werden. Dazu werden $k - 2$ zusätzliche Punkte hinzugefügt, die zu allen anderen Punkten einen Abstand größer als ρ_d haben und somit ein eigenes Zentrum erfordern.

3.2 Lösung des geometrischen k-Centers

Brandenberg und Roth stellen in [3] einen Branch-and-Bound Algorithmus vor, der das geometrische k-Center zu verschiedenen Containernormen bei beliebiger Toleranz approximiert. Arnold hat dieses Verfahren in seiner Diplomarbeit [1] – allerdings nur für euklidische Container, da sich dann die Tiefe des Branch-and-Bound Baumes beschränken lässt – implementiert. Um diesen Algorithmus anzuwenden, kann daher als Distanz zweier Punkte $P = (\lambda_1, \varphi_1)$ und $Q = (\lambda_2, \varphi_2)$ nicht die Länge des Kreisbogens entlang der Erdoberfläche

$$\text{dist}(P, Q) = R_E \cdot \arccos(\sin(\varphi_1)\sin(\varphi_2) + \cos(\varphi_1)\cos(\varphi_2)\cos(\lambda_1 - \lambda_2)) \tag{3.3}$$

verwendet werden.
Die von der OpenStreetMap verwendete Mercator-Projektion verursacht, wie in Abbildung 3.2 dargestellt, ab einer geographischen Breite von 48.2 Grad[5] schon einen Fehler von mehr als 50%. Daher betrachten wir nun weitere Möglichkeiten den Abstand zweier Punkte durch eine euklidische Norm anzunähern.

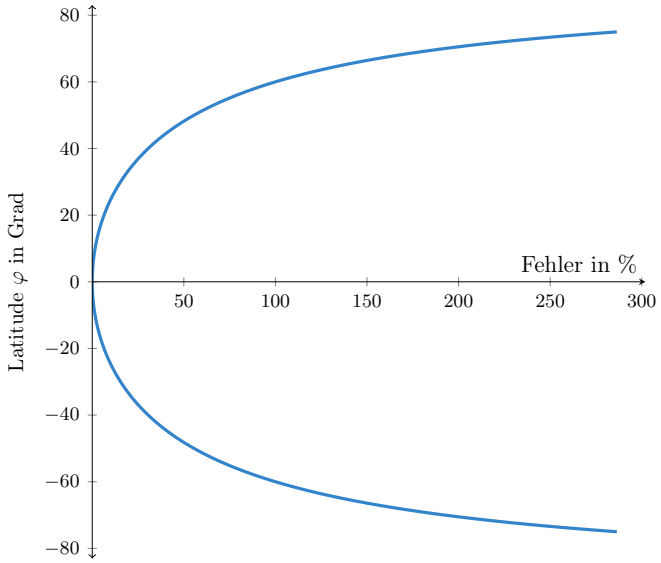

Abbildung 3.2: Fehler der Mercator-Projektion im Intervall $[-75°, 75°]$.

Zylinderprojektion

Fasst man die beiden sphärischen Parameter λ und φ als kartesische Koordinaten in der Ebene auf, kann die Entfernung der Punkte $P = (\lambda_1, \varphi_1)$ und $Q = (\lambda_2, \varphi_2)$ auf einfache Art durch den euklidischen Abstand $\|P - Q\|_2$ berechnet werden. Wie in Abbildung 3.3 zu erkennen ist, entspricht dies einer Zylinderprojektion der Punkte P und Q.
Da der Großkreis zur geographischen Breite φ jedoch nur einen Radius von $2\pi R_E \cdot \cos(\varphi)$ hat und dies bisher nicht berücksichtigt wurde, treten in polaren Gebieten große Verzerrungen auf. Um diesen Effekt zu kompensieren werden Punkte (λ, φ) in einem Vorberechnungs-

[5] Der Marienplatz in München hat eine geographische Breite von 48.137°.

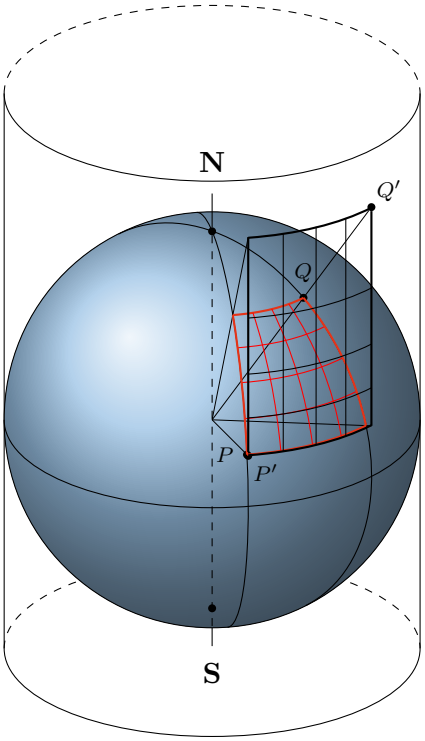

Abbildung 3.3: Verzerrung bei der Zylinderprojektion.

schritt auf $(\cos(\varphi)\lambda, \varphi)$ abgebildet. Dann kann mit der dadurch festgelegten Distanzfunktion

$$\begin{aligned}\text{dist}(P,Q) &= R_E \cdot \sqrt{(\cos(\varphi_1)\lambda_1 - \cos(\varphi_2)\lambda_2)^2 + (\varphi_1 - \varphi_2)^2} \\ &= R_E \cdot \left\| \begin{pmatrix} \cos(\varphi_1)\lambda_1 \\ \varphi_1 \end{pmatrix} - \begin{pmatrix} \cos(\varphi_2)\lambda_2 \\ \varphi_2 \end{pmatrix} \right\|_2\end{aligned}$$

der von Arnold implementierte Algorithmus verwendet werden.

Verwendung von 3D-Koordinaten

Anstatt die Kugeloberfläche in die Ebene zu projizieren, können die Punkte auch in kartesische Raumkoordinaten umgerechnet werden. Daraufhin kann die euklidische Norm in drei Dimensionen berechnet werden. Dadurch wird zwar die Erdkrümmung ignoriert, für kleine Gebiete ist diese jedoch vernachlässigbar.

In Abbildung 3.4 ist der relative Fehler der Entfernung zweier Punkte unter der Zylinderprojektion (blau) bzw. bei Verwendung der 3D-Koordinaten (grün) gegenüber der realen Entfernung auf der Kugeloberfläche nach Gleichung (3.3) eingezeichnet. Da die Entfernung breitengradunabhängig ist, wurden die Punkte auf demselben Breitengrad mit einer Entfernung von $\Delta\lambda = 1°$ (ca. 111 Kilometer am Äquator) gewählt.
Die Zylinderprojektion verursacht mit zunehmendem Abstand der Punkte vom Äquator einen immer größeren Fehler, wohingegen der Fehler, der durch die Verwendung der 3D-Koordinaten entsteht, in Äquatornähe maximal wird, da er vom Radius des Breitenkreises abhängt.

Da beide Verfahren in den für uns relevanten Breitengraden ähnlich gute Ergebnisse liefern, die Verwendung der 3D-Koordinaten durch die zusätzliche Dimension aber eine Laufzeitverschlechterung im Algorithmus von Brandenberg & Roth verursacht, verwenden wir im Folgenden die durch die Zylinderprojektion festgelegte Distanzfunktion.

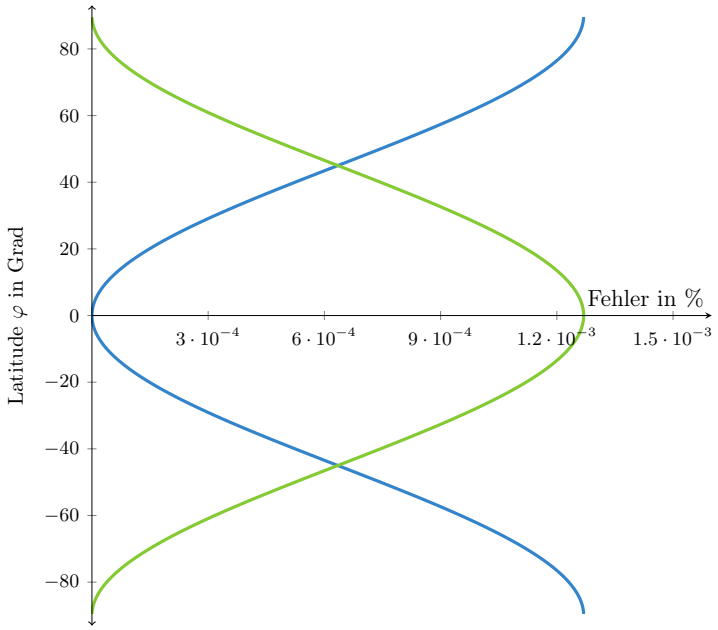

Abbildung 3.4: Relativer Fehler der Zylinderprojektion (blau) und der euklidischen Norm in drei Dimensionen (grün).

Laufzeit des Branch-and-Bound Algorithmus

Brandenberg und Roth haben in [3] gezeigt, dass der Branch-and-Bound Algorithmus eine Laufzeit von $\mathcal{O}(k^h nd)$ hat. Dabei bezeichnet h die maximale Tiefe des Branch-and-Bound Baumes, n die Anzahl der betrachteten Punkte und d die Dimension des Raumes. Für eine ε-Approximation des euklidischen k-Centers kann die Tiefe des Baumes durch

$$h = \mathcal{O}(\frac{k}{\varepsilon^2})$$

abgeschätzt werden. Bei Verwendung der Implementierung von Arnold sind in Abbildung 3.5 die Laufzeit (blau) und die Anzahl der Verzweigungen im Branch-and-Bound-Verfahren von Brandenberg & Roth (grün) für zunehmendes k eingezeichnet. Als Basis wurde der Datensatz *Apotheken* (siehe Abschnitt 5.2) mit einer Toleranz von 1% verwendet. An der logarithmischen Skala sieht man deutlich das für zunehmendes k exponentielle Verhalten.

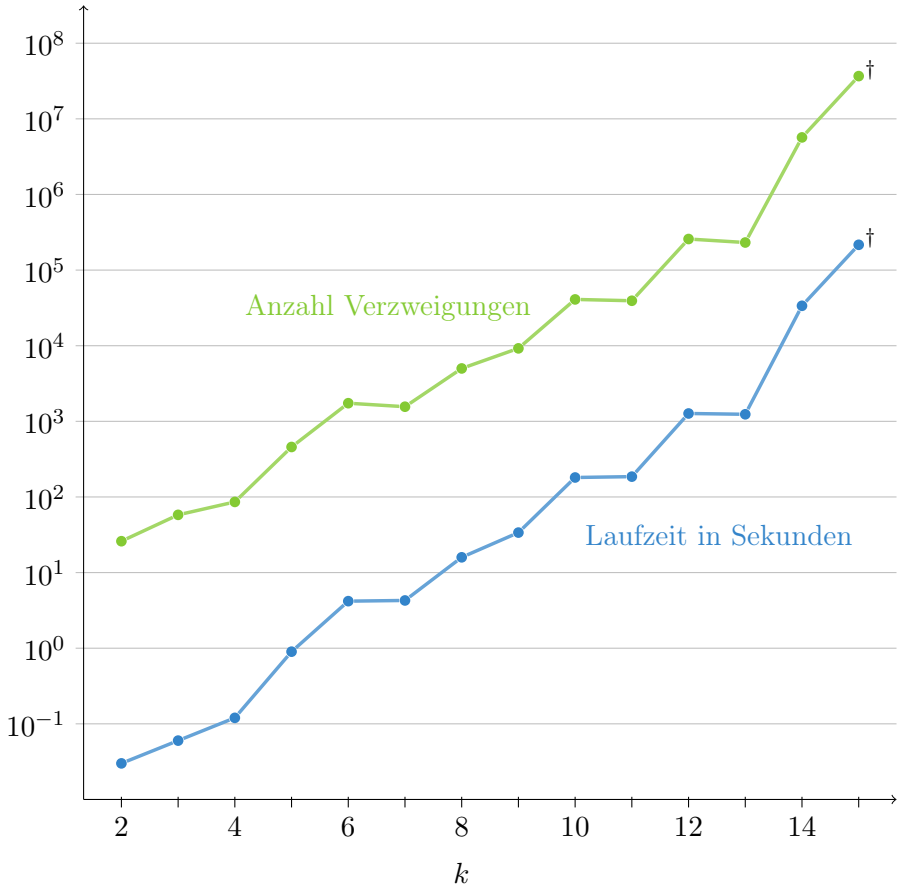

Abbildung 3.5: Die Laufzeit (in Sekunden) und die Anzahl expandierter Knoten des Algorithmus von Brandenberg & Roth auf dem Datensatz *Apotheken*.

[†]Die Berechnung für das 15-Center wurde nach 60 Stunden abgebrochen. Die bis dahin expandierten Knoten und die verstrichene Laufzeit wurden eingezeichnet.

4 Graphentheoretisches k-Center

Beim graphentheoretischen k-Center sind die Standorte im Gegensatz zur geometrischen Variante keine Punkte in einem Minkowski-Raum sondern Knoten in einem gewichteten Graphen $G = (V, E, \delta)$. Dadurch können natürlich die Zentren nicht mehr beliebig verteilt werden, sondern müssen ein Teil des Graphen sein. Auch wenn in einigen Definitionen des graphentheoretischen k-Centers (siehe z.B. [15]) die Zentren auch „auf Kanten“ liegen können, betrachten wir zunächst die klassische Variante, in der die Zentren eine Teilmenge der Knotenmenge sind, bevor wir in Abschnitt 4.3 auf eine Verallgemeinerung eingehen.

Problem 4.1 *(Graphentheoretisches* k*-Center)*
Gegeben sei ein vollständiger, gewichteter Graph $G = (V, E, \delta)$ mit einer nicht-negativen Gewichtsfunktion $\delta : V \times V \to \mathbb{R}^+$, sowie $k \in \mathbb{N}$. Das Problem, eine höchstens k-elementige Menge $C \subseteq V$ zu finden, sodass

$$\max_{v \in V} \min_{c \in C} \delta(c, v) \tag{4.1}$$

minimiert wird, heißt GRAPHENTHEORETISCHES k-CENTER.

In Abbildung 4.1 ist die Lösung eines graphentheoretischen 5-Centers in der OpenStreetMap eingezeichnet. Die Zentren sind schwarz markiert, während die Standorte zum jeweils gleichen nächsten Zentrum in einer Farbe dargestellt sind.

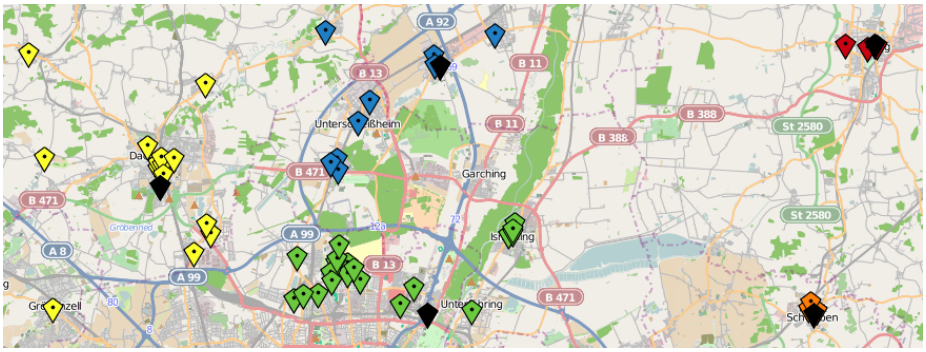

Abbildung 4.1: Beispiel eines graphentheoretischen 5-Centers.

4.1 Komplexität des graphentheoretischen k-Centers

Für beliebige Gewichtsfunktionen lässt sich das GRAPHENTHEORETISCHE k-CENTER nicht polynomiell approximieren. Der Beweis verwendet das $\mathbb{NP}$-vollständige Problem MINIMALE DOMINIERENDE MENGE.

Problem 4.2 *(Minimale dominierende Menge)*
Das Problem `Minimale Dominierende Menge` entscheidet zur Eingabe $(G, k \in \mathbb{N})$, ob im Graphen $G = (V, E)$ eine dominierende Menge mit Kardinalität k existiert.

Lemma 4.3
Das Problem `Minimale Dominierende Menge` ist $\mathbb{NP}$-vollständig.

Beweis Das Problem ist in $\mathbb{NP}$, da durch die Angabe einer höchstens k-elementigen Knotenmenge in polynomieller Zeit überprüft werden kann, ob sie den Graphen dominiert. Wir wollen eine Instanz der Entscheidungsvariante von `3-SAT` mit Klauseln K_j, $j = 1 \dots m$ über jeweils paarweise verschiedenen Literalen aus $\{l_1, \bar{l}_1, \dots, l_n, \bar{l}_n\}$ mit Hilfe von Problem 4.2 lösen. Dazu generieren wir einen Graphen $G = (V, E)$, indem wir jede Variable l_i durch einen Kreis der kombinatorischen Länge 3 mit Knoten $V(l_i) := \{l_i =: l_i^1, l_i^2, l_i^3 := \bar{l}_i\}$ und Kanten $E(l_i)$, sowie jede Klausel K_j durch einen Kreis der kombinatorischen Länge 6 mit Knoten $V(K_j) := \{c_j^1, c_j^2, c_j^3, c_j^4, c_j^5, c_j^6\}$ und Kanten $E(K_j)$ ersetzen.
Außerdem definieren wir für die Klausel K_j mit den drei Literalen x, y und z zusätzliche Kanten

$$E'(K_j) := \{\{c_j^1, x\}, \{c_j^3, y\}, \{c_j^5, z\}\}$$

um die Belegung der Klauseln zu berücksichtigen.
Damit kann der Graph $G = (V, E)$

$$V = \bigcup_{i=1}^{n} V(l_i) \cup \bigcup_{j=1}^{m} V(K_j)$$
$$E = \bigcup_{i=1}^{n} E(l_i) \cup \bigcup_{j=1}^{m} E(K_j) \cup \bigcup_{j=1}^{m} E'(K_j)$$

angegeben werden.

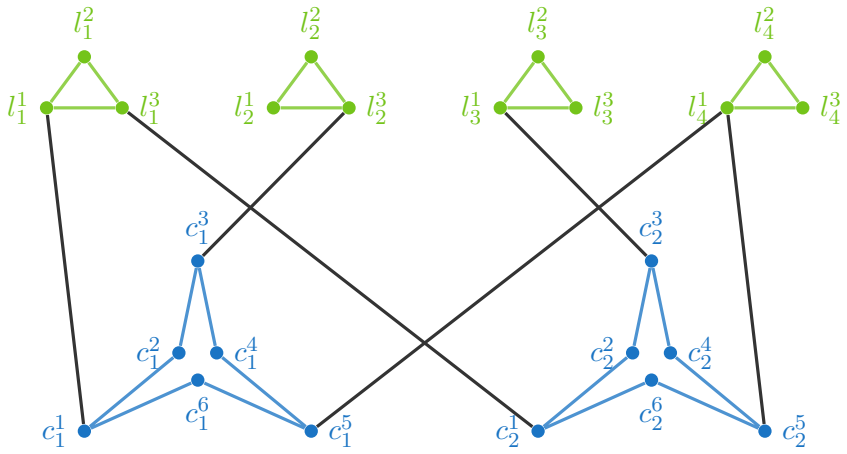

Abbildung 4.2: Beispiel eines aus der 3-SAT-Instanz $K_1 = \{l_1, \bar{l}_2, l_4\}$, $K_2 = \{\bar{l}_1, l_3, l_4\}$ generierten Graphen. In grün sind die Kreise $V(l_i)$, in blau die Kreise $V(K_j)$ markiert und in schwarz sind die zusätzlichen Kanten $E'(K_j)$ eingezeichnet.

Da die zusätzlichen Kanten $E'(K_j)$ jeweils nur ungerade Knoten aus den Kreisen verbinden, stellen wir fest, dass in einer den Graphen G dominierenden Menge jeweils mindestens ein Knoten aus $V(l_i)$ liegen muss, da ansonsten l_i^2 nicht überdeckt werden könnte. Mit demselben Argument müssen auch jeweils mindestens 2 Knoten aus $V(K_j)$ gewählt werden, um die Knoten c_j^2, c_j^4 und c_j^6 zu überdecken. Wir zeigen nun, dass für die Wahl $k = n + 2m$ aus einer Lösung des Problems MINIMALE DOMINIERENDE MENGE eine Lösung von 3-SAT gewonnen werden kann.

Angenommen, es existiert eine minimale dominierende Menge D mit maximal $n+2m$ Knoten, so muss diese mit den vorherigen Überlegungen genau $n + 2m$ Knoten haben, um dominierend sein zu können. Insbesondere wurde für jede Variable l_i genau ein Knoten aus $V(l_i)$ ausgewählt. Wir belegen nun die Variable l_i mit wahr $\Leftrightarrow l_i^1 \in D$. Da auch für jede Klausel K_j genau zwei Knoten aus $V(K_j)$ ausgewählt wurden, muss mindestens einer der Knoten aus $\{c_j^1, c_j^3, c_j^5\}$ durch eine Kante aus $E'(K_j)$ mit D verbunden sein. Dies ist genau dann der Fall, wenn das zugehörige Literal wahr ist und die Klausel damit erfüllt. Da dies für alle Klauseln gilt, haben wir eine Wahrheitsbelegung der Klauseln gefunden.

Betrachten wir nun umgekehrt den Fall, dass eine Wahrheitsbelegung der Klauseln existiert. Wir bestimmen eine dominierende Menge D, indem alle wahren Literale zu D hinzugefügt werden. Da wir eine Wahrheitsbelegung haben, existiert für jede Klausel K_j ein Knoten $x \in \{c_j^1, c_j^3, c_j^5\}$, der durch eine Kante aus $E'(K_j)$ mit D verbunden ist. Es ist also möglich, durch jeweils zwei zusätzliche Knoten alle anderen Knoten aus $V(K_j)$ zu dominieren. Insgesamt enthält die Menge D damit $n + 2m$ Knoten und ist eine Lösung von Problem 4.2. □

Lemma 4.4 *(Nicht-Approximierbarkeit des graphentheoretischen* k*-Centers)*
Falls $\mathbb{P} \neq \mathbb{NP}$, lässt sich das GRAPHENTHEORETISCHE k-CENTER zu keinem polynomiell berechenbaren Faktor $\alpha(n)$ approximieren.

Beweis Als Beweis führen wir eine Reduktion des Problems MINIMALE DOMINIERENDE MENGE auf ein GRAPHENTHEORETISCHES k-CENTER durch. Sei dazu $(G = (V, E), k)$ eine Instanz von MINIMALE DOMINIERENDE MENGE. Daraus konstruieren wir einen vollständigen, gewichteten Graphen $G' = (V, E', \delta)$ mit

$$\delta(u, v) = \begin{cases} 1, & \text{wenn } \{u, v\} \in E \\ \alpha(n) + \varepsilon, & \text{sonst} \end{cases}$$

für ein $\varepsilon > 0$. Offensichtlich gilt:

- Falls eine minimale dominierende Menge mit Kardinalität k existiert, so hat G' ein k-Center mit Radius 1.
- Falls keine minimale dominierende Menge mit Kardinalität k existiert, muss ein k-Center in G' einen Radius $\alpha(n) + \varepsilon$ haben.

Mit einer $\alpha(n)$-Approximation ließe sich damit das Problem MINIMALE DOMINIERENDE MENGE in polynomieller Zeit entscheiden, was $\mathbb{P} = \mathbb{NP}$ impliziert. □

Die starke Aussage von Lemma 4.4 ist möglich, weil in Problem 4.1 keine Einschränkungen an die Gewichtsfunktion gestellt wurden. Insbesondere wird die Dreiecksungleichung von der

im Beweis von Lemma 4.4 verwendeten Gewichtsfunktion verletzt. Im Folgenden betrachten wir daher nur noch spezielle Gewichtsfunktionen.

4.2 Approximation des graphentheoretischen k-Centers

Für gewichtete Graphen, deren Gewichtsfunktionen symmetrisch sind und die Dreiecksungleichung erfüllen, das heißt für die

$$\delta(u,v) = \delta(v,u) \qquad \forall u,v \in V \tag{4.2}$$

$$\delta(u,v) + \delta(v,w) \geq \delta(u,w) \qquad \forall u,v,w \in V \tag{4.3}$$

gilt, existiert eine einfache 2-Approximation. Betrachten wir beispielsweise den von Gonzalez in [7] vorgestellten Greedy-Algorithmus: Im ersten Schritt wird ein zufälliger Knoten als Zentrum ausgewählt. In den nächsten $k-1$ Schritten wird der Knoten mit dem größten Abstand zu allen bis dahin ausgewählten Zentren als neues Zentrum hinzugefügt.

Lemma 4.5
Der Greedy-Algorithmus von Gonzalez für das symmetrische k-Center mit Dreiecksungleichung erreicht bei einer Laufzeit von $\mathcal{O}(kn)$ eine Approximationsgüte von 2.

Beweis Der Algorithmus terminiert nach k Schritten. Im i-ten Schritt wird unter allen n Knoten derjenige mit maximalem Abstand zu den Zentren c_1 bis c_{i-1} gesucht. Durch Speicherung des maximalen Abstands zu allen bisherigen Zentren, kann dieser in konstanter Zeit berechnet werden. Dies führt zu einer Laufzeit von $\mathcal{O}(kn)$.
Für den Beweis der Approximationsgüte sei ρ^* der Radius einer optimalen Lösung des graphentheoretischen k-Centers. Angenommen, der Greedy-Algorithmus würde eine Lösung mit Radius $\rho > 2\rho^*$ zurückliefern, so würden $k+1$ Punkte existieren, die jeweils paarweise eine größere Entfernung als ρ hätten (die k Zentren plus einen weiteren Punkt). Um diese $k+1$ Punkte durch Kreise mit Radius ρ^* zu überdecken, müssten auch $k+1$ Zentren ausgewählt werden. □

Hsu und Nemhauser haben in [11] gezeigt, dass die Approximationsgüte von diesem einfachen Algorithmus sogar optimal ist.

Lemma 4.6
Falls $\mathbb{P} \neq \mathbb{NP}$, dann existiert für kein $\varepsilon > 0$ ein polynomieller Algorithmus mit einem Approximationsfaktor von $2-\varepsilon$ für das symmetrische graphentheoretische k-Center mit Dreiecksungleichung.

Beweis Der Beweis läuft analog zum Beweis von Lemma 4.4 mit $\alpha(n) = 2-\varepsilon$. Zu einer Instanz $(G=(V,E),\, k)$ des Problems Minimale Dominierende Menge wird ein vollständiger, gewichteter Graph $G' = (V, E', \delta)$ mit

$$\delta(u,v) = \begin{cases} 1, & \text{wenn } \{u,v\} \in E \\ 2, & \text{sonst} \end{cases}$$

konstruiert. Offensichtlich erfüllt dabei δ die Dreiecksungleichung. Mit einer $(2-\varepsilon)$-Approximation für das graphentheoretische k-Center ließe sich damit wiederum das Problem Minimale Dominierende Menge in polynomieller Zeit entscheiden. □

Anstatt ein k-Center direkt auf dem Graphen zu suchen, wollen wir nun einen anderen Ansatz betrachten. Dabei sortieren wir zuerst die Kantenmenge E so, dass mit $|E| = m$ für die Kanten

$$\delta(e_1) \leq \delta(e_2) \leq \cdots \leq \delta(e_m)$$

gilt. Anschließend entscheiden wir der Reihe nach für jede Kante e_i, ob ein k-Center mit Radius $\delta(e_i)$ existieren kann. Im Folgenden verwenden wir die Bezeichnung:

$$E_i := \{e_1, e_2, \ldots, e_i\}$$

Lemma 4.7
Problem 4.1 ist äquivalent dazu, einen minimalen Index i^* zu finden, sodass eine minimale dominierende Menge D_{i^*} mit $|D_{i^*}| \leq k$ im Graphen $G_{i^*} = (V, E_{i^*})$ existiert. D_{i^*} ist dann auch eine optimale Zentrenmenge des graphentheoretischen k-Centers.

Beweis Es gilt $E_i \subseteq E_{i+1}$. Damit ist jede dominierende Menge auf G_i auch eine dominierende Menge auf G_{i+1}. Da $E_m = E$ gilt, ist $G_m = (V, E_m)$ vollständig und in G_m existiert eine minimale dominierende Menge mit Kardinalität 1. Insbesondere existiert also immer ein kleinster Index i^* mit $|D_{i^*}| \leq k$.
Nach Konstruktion ist D_{i^*} eine zulässige Lösung für das graphentheoretische k-Center. Würde eine bessere Lösung C existieren, müsste diese alle Knoten aus V überdecken, ohne Kanten mit Länge $\geq \delta(e_{i^*})$ zu verwenden. C wäre dann aber eine minimale dominierende Menge mit $|C| \leq k$ in einem Graphen G_i mit $i < i^*$. Damit wäre i^* kein minimaler Index. □

Da das Problem eine minimale dominierende Menge zu finden $\mathbb{NP}$-vollständig ist, lässt sich der Ansatz aus Lemma 4.7 nicht direkt anwenden.
Hochbaum und Shmoys schlagen in [10] vor, mit Hilfe maximaler stabiler Mengen eine untere Schranke für minimale dominierende Mengen zu bestimmen.

Problem 4.8 *(Maximale stabile Menge)*
Sei $G = (V, E)$ ein Graph. Das Problem Maximale Stabile Menge liefert eine inklusionsmaximale stabile Menge $M \subseteq V$ zurück.

Lemma 4.9
Sei $G = (V, E)$ ein Graph. Betrachten wir folgenden Algorithmus für das Problem Maximale Stabile Menge: Zur leeren Menge M füge in jedem Schritt einen beliebigen Knoten aus $V \setminus (M \cup \mathcal{N}(M))$ hinzu, solange diese Menge nicht leer ist. Dieser Greedy-Algorithmus liefert in einer Laufzeit von $\mathcal{O}(|V|)$ eine maximale stabile Menge im Graphen G.

Beweis In jedem Schritt wird ein Knoten hinzugefügt, der noch zu keinem bisherigen Knoten benachbart ist. Da der Algorithmus auch erst terminiert, wenn kein Knoten mehr hinzugefügt werden könnte, ohne die Stabilität zu verletzen, ist M eine maximale stabile Menge in G.
Speichert man alle Knoten in einer Liste und markiert in jedem Schritt die Nachbarn des neu zu M hinzugefügten Knotens, wird jeder Knoten entweder zu M hinzugefügt oder als zu M benachbart markiert. Daraus folgt, dass die Laufzeit linear in $|V|$ ist. □

Definition 4.10 *(Quadratgraph)*
Zu $G = (V, E)$ ist $G^2 = (V, E')$ mit

$$\{u, v\} \in E' \Leftrightarrow \exists\ u\text{-}v\text{-Weg mit kombinatorischer Länge} \leq 2 \text{ in } G$$

der *Quadratgraph* von G.

Lemma 4.11
Sei $G = (V, E)$ ein Graph. Außerdem sei D eine minimale dominierende Menge in G und M eine maximale stabile Menge im Quadratgraphen G^2. Dann gilt $|M| \leq |D|$.

Beweis Der Graph G enthält maximal $|D|$ Sterne, die die Knotenmenge V überdecken. In G^2 wird jeder dieser Sterne zu einer Clique und es kann offensichtlich maximal ein Punkt jeder Clique in M liegen. □

Algorithmus 4.12 *(Hochbaum & Shmoys)*
Input: $G = (V, E, \delta), k \in \mathbb{N}$
Output: $C \subseteq V$

(1) Sortiere $E = \{e_1, \ldots, e_m\}$ aufsteigend nach Kantengewichten
(2) $E_0 = \emptyset$
(3) **for** $i = 1, \ldots, m$ **do**
(4) $\quad E_i = E_{i-1} \cup e_i$
(5) $\quad$ Berechne eine maximale stabile Menge M_i in $G_i^2 = (V, E_i)^2$
(6) $\quad$ **if** $|M_i| \leq k$ **do**
(7) $\quad\quad$ **return** M_i
(8) $\quad$ **end-if**
(9) **end-for**

Lemma 4.13
Mit den Bezeichnungen aus Algorithmus 4.12 sei i^* der minimale Index, sodass die maximale stabile Menge M_{i^*} die Bedingung $|M_{i^*}| \leq k$ erfüllt. Dann ist durch $\delta(e_{i^*})$ eine untere Schranke für das graphentheoretische k-Center gegeben.

Beweis Für alle $i < i^*$ gilt $|M_i| > k$. Mit Lemma 4.11 gilt somit $|D_i| > k$ für alle minimalen dominierenden Mengen auf dem Graphen $G_i = (V, E_i)$. □

Satz 4.14
Sei $G = (V, E, \delta)$ ein vollständiger, gewichteter Graph mit symmetrischer Gewichtsfunktion, die die Dreiecksungleichung erfüllt. Dann ist Algorithmus 4.12 mit einer Laufzeit von $\mathcal{O}(m \log m + mn)$ eine 2-Approximation für das graphentheoretische k-Center auf G.

Beweis Eine maximale stabile Menge M auf einem Graphen ist auch eine dominierende Menge. Ansonsten könnte ein nicht überdeckter Knoten $\bar{v} \in V \setminus (M \cup \mathcal{N}(M))$ zu M hinzugefügt werden und M wäre damit nicht maximal.
Im Quadratgraphen existieren $|M| \leq k$ Sterne, die alle Knoten überdecken und aufgrund der Dreiecksungleichung maximal einen Radius von $2 \cdot \delta(e_{i^*})$ haben. Mit Lemma 4.13 folgt

die Approximationsgüte.

Hinsichtlich der Laufzeit stellen wir fest, dass die Kantensortierung in $\mathcal{O}(m \log(m))$ durchgeführt werden kann.
Die einzige für die Laufzeit relevante Berechnung innerhalb der Schleife ist die Bestimmung einer maximalen stabilen Menge im Quadratgraphen. Ist der Graph G_i in Form einer Adjazenzmatrix A_i gegeben, entsprechen die verbundenen Knoten im Quadratgraphen den Nicht-Null-Einträgen in A_i^2. Die Berechnung von A_i und A_i^2 kann sukzessive in $\mathcal{O}(n)$ durchgeführt werden, da sich in jedem Durchgang nur zwei Einträge in A_i und damit $2n - 1$ Einträge in A_i^2 ändern können.
Die Berechnung einer maximalen stabilen Menge kann nach Lemma 4.9 ebenfalls in $\mathcal{O}(n)$ durchgeführt werden. Damit erhalten wir für die Schleife eine Laufzeit von $\mathcal{O}(mn)$ und insgesamt folgt die Behauptung. □

Wir führen nun eine Verallgemeinerung des graphentheoretischen k-Centers aus Problem 4.1 ein, damit die Lösung besser mit einer geometrischen k-Center Lösung verglichen werden kann.

4.3 Erweiterung der Knotenmenge

Bisher mussten die Zentren aus der Menge der Standorte ausgewählt werden. Diese Einschränkung ist in praktischen Anwendungen jedoch oft zu restriktiv. In Verkehrsnetzwerken macht es beispielsweise Sinn die Zentren beliebig entlang von Straßen platzieren zu können. Das verallgemeinerte graphentheoretische k-Center ermöglicht es, zusätzliche Knoten als Positionen für mögliche Zentren in die Knotenmenge einzufügen, ohne dass diese überdeckt werden müssen.

Problem 4.15 *(verallgemeinertes graphentheoretisches* k*-Center)*
Gegeben seien ein vollständiger, gewichteter Graph $G = (V, E, \delta)$ mit einer nicht-negativen Gewichtsfunktion $\delta : V \times V \to \mathbb{R}^+$, eine Menge $S \subseteq V$, sowie $k \in \mathbb{N}$. Das Problem, eine höchstens k-elementige Menge $C \subseteq V$ zu finden, sodass

$$\max_{s \in S} \min_{c \in C} \delta(\{c, s\})$$

minimiert wird, heißt VERALLGEMEINERTES GRAPHENTHEORETISCHES k-CENTER.

Die einzige Änderung in Problem 4.15 gegenüber dem graphentheoretischen k-Center aus Problem 4.1 ist die Möglichkeit, die Standortmenge einzuschränken. Mit der Wahl $S = V$ sind die beiden Probleme äquivalent.

Das verallgemeinerte graphentheoretische k-Center kann folgendermaßen als ILP formuliert werden:

$$\begin{aligned}
&\text{Minimiere} && z && \\
&\sum_{v\in V} x_{sv} && = && 1 \quad \forall s \in S && (4.4)\\
&x_{sv} && \leq && y_v \quad \forall s \in S,\ v \in V && (4.5)\\
&\sum_{v\in V} y_v && \leq && k && (4.6)\\
&\sum_{v\in V} \delta(s,v) x_{sv} && \leq && z \quad \forall s \in S && (4.7)\\
&x_{sv},\ y_v && \in && \{0,1\} \quad \forall s \in S,\ v \in V && (4.8)
\end{aligned}$$

Dabei wird die Variable x_{sv} genau dann eins, wenn Standort s dem Zentrum v zugeordnet wird. Die Variable y_v wird eins, wenn v als Zentrum ausgewählt wird. Bedingung (4.4) drückt dabei aus, dass jeder Standort von genau einem Zentrum beliefert werden muss. Bedingung (4.5) legt fest, dass die Standorte nur von Zentren aus beliefert werden dürfen. Mit Bedingung (4.6) wird die Anzahl der Zentren beschränkt und zusammen mit der Gleichheit in Bedingung (4.4) wird in Gleichung (4.7) eine zu minimierende obere Schranke z für den Radius des k-Centers gefordert.
In [16] wurde gezeigt, dass schon für kleine Instanzen die Lösung der ILP-Formulierung sehr lange Berechnungszeiten hat und die LP-Relaxation schwache untere Schranken liefert. Daher betrachten wir nun einen Ansatz, der die Idee, mit Hilfe einer sortierten Kantenliste für verschiedene Radien zu entscheiden, ob ein k-Center mit diesem Radius existieren kann, verfeinert.

4.4 Lösung des verallgemeinerten graphentheoretischen k-Centers

Wir wollen nun einen Algorithmus für eine exakte Lösung des verallgemeinerten graphentheoretischen k-Centers vorstellen. Dieser beruht auf einem von Ilhan und Pinar in [12] vorgeschlagenen Ansatz. Die Grundidee dabei ist, mit Hilfe eines Bisektionsverfahrens den minimalen Radius ρ^* zu finden, für den sich alle Standorte durch maximal k Zentren überdecken lassen.

Die Frage, ob alle Standorte durch maximal k Zentren überdeckt werden können, ohne dabei einen Radius ρ zu überschreiten, ist ein Überdeckungsproblem. Mit der Funktion

$$\begin{aligned}
b_\rho \ :\ & S \times V \to \{0,1\} \\
& (s,v) \mapsto \begin{cases} 1, & \text{wenn } \delta(s,v) \leq \rho \\ 0, & \text{sonst} \end{cases}
\end{aligned}$$

lässt sich dieses als ILP (`SC-ILP(`ρ`)`) formulieren:

$$\begin{aligned}
&\text{Minimiere} && \sum_{v\in V} y_v && \\
&\sum_{v\in V} b_\rho(s,v) y_v && \geq && 1 \quad \forall s \in S && (4.9)\\
&y_v && \in && \{0,1\} \quad \forall v \in V && (4.10)
\end{aligned}$$

Mit Bedingung (4.9) wird gefordert, dass jeder Standort von mindestens einem Zentrum überdeckt wird, während gleichzeitig die Anzahl der Zentren minimiert werden soll.
Bei der LP-Relaxation (SC-LP(ρ)) wird die Bedingung (4.10) durch

$$0 \leq y_v \leq 1 \quad \forall v \in V$$

ersetzt.

Lemma 4.16
Sei ρ^* der Radius einer optimalen Lösung des VERALLGEMEINERTEN GRAPHENTHEORETISCHEN k-CENTERS. Dann gilt für eine Lösung C von SC-ILP(ρ)

$$|C| > k \Longleftrightarrow \rho < \rho^*$$

für beliebiges $\rho > 0$.

Beweis Sei $\rho > 0$ beliebig.
Für die Hinrichtung gelte $|C| > k$ für eine Lösung C von SC-ILP(ρ). Wäre nun $\rho \geq \rho^*$, so wäre insbesondere $b_\rho(s, v) = 1\ \forall (s, v)$ mit $\delta(s, v) \leq \rho^*$. Damit würde auch eine Lösung des ILP mit Kardinalität k – im Widerspruch zur Voraussetzung – existieren.
Gelte nun für die Rückrichtung $\rho < \rho^*$. Würde nun SC-ILP(ρ) eine Überdeckung mit maximal k Zentren finden, so wäre diese eine zulässige Lösung des VERALLGEMEINERTEN GRAPHENTHEORETISCHEN k-CENTERS mit geringerem Radius und stünde somit im Widerspruch zur Optimalität von ρ^*. □

Algorithmus 4.17 *(Ilhan & Pinar)*
Input: $G = (V, E, \delta),\ S \subseteq V,\ k \in \mathbb{N}_0,\ \varepsilon > 0$
Output: $C \subseteq V$

(1) $lb := \min\{\delta(s, v) : s \in S,\ v \in V\}$
(2) $ub := \max\{\delta(s, v) : s \in S,\ v \in V\}$
(3) **while** $ub > lb \cdot (1 + \varepsilon)$ **do**
(4) $\quad \rho := \frac{ub+lb}{2}$
(5) $\quad$ **if** SC-LP(ρ) $\leq k$ **then** $ub := \rho$
(6) $\quad$ **else** $lb := \rho$
(7) **end-while**
(8) $\rho := lb$
(9) **while** $\rho < \max\{\delta(s, v) : s \in S,\ v \in V\}$ **do**
(10) $\quad$ **if** Lösung C von SC-ILP(ρ) $\leq k$ **then**
(11) $\quad\quad$ **return** C
(12) $\quad$ **end-if**
(13) $\quad \rho := \min\{\delta(s, v) : s \in S,\ v \in V,\ \delta(s, v) > \rho\}$
(14) **end-while**

Algorithmus 4.17 besteht aus drei Abschnitten. In der Initialisierungsphase (Zeile 1 und 2) wird eine (triviale) untere und obere Schranke bestimmt. In der LP-Schleife (Zeile 3 bis 7) wird mit LP-Relaxationen eine möglichst gute untere Schranke berechnet, um anschließend

in der ILP-Schleife (Zeile 9 bis 14) für immer größere Radien die Zulässigkeit des ILP-Problems zu überprüfen.
Mit dem Kontrollparameter ε kann gesteuert werden, wie schnell von der LP-Schleife in die ILP-Schleife gewechselt wird.

Lemma 4.18 *(Korrektheit von Algorithmus 4.17)*
Für vollständige Graphen $G = (V, E, \delta)$, $S \subseteq V$ und $k \in \mathbb{N}$ findet Algorithmus 4.17 zu jedem $\varepsilon > 0$ eine optimale Lösung für das Problem VERALLGEMEINERTES GRAPHENTHEORETISCHES k-CENTER.

Beweis Sei ρ^* der Radius einer optimalen Lösung des VERALLGEMEINERTEN GRAPHENTHEORETISCHEN k-CENTERS. Es reicht zu zeigen, dass für den Radius $\bar{\rho}$ der gefundenen Lösung

$$\bar{\rho} \leq \rho^*$$

gilt.
Nach Wahl gilt $lb \leq \rho^*$ in der Initialisierungsphase.
In der LP-Schleife wird lb genau dann auf den Wert ρ erhöht, wenn SC-LP(ρ) $> k$ gilt. Dies bedeutet aber, dass keine LP-Lösung – und daher auch insbesondere keine ILP-Lösung – mit Radius ρ existieren kann. Damit bleibt die Ungleichung $lb \leq \rho^*$ in der LP-Schleife gültig.
Es bezeichne ρ^i den Radius im i-ten Durchgang der ILP-Schleife. Mit den bisherigen Überlegungen gilt zu Beginn der Schleife $\rho^0 \leq \rho^*$. Falls $\rho^0 = \rho^*$ gilt, so wird im ersten Durchgang eine optimale Lösung zurückgegeben.
Mit Lemma 4.16 gilt $\rho^* > \rho^i$, solange der Algorithmus nicht terminiert. Da der Graph vollständig ist, existiert ein minimaler Index $\bar{i}$, für den SC-ILP($\rho^{\bar{i}}$) $\leq k$ gilt und der Algorithmus terminiert.
Angenommen $\bar{\rho} = \rho^{\bar{i}} > \rho^*$. Dann existiert $s^* \in S$ und $v^* \in V$ mit $\delta(s^*, v^*) = \rho^* > \rho^{\bar{i}-1}$. Daraus folgt nach Wahl von ρ in Zeile 13:

$$\begin{aligned} \rho^{\bar{i}} &= \min\{\delta(s,v) : s \in S,\ v \in V,\ \delta(s,v) > \rho^{\bar{i}-1}\} \\ &\leq \delta(s^*, v^*) \\ &< \rho^{\bar{i}} \end{aligned}$$

Ein Widerspruch. Daher muss $\bar{\rho} \leq \rho^*$ gelten. □

Bemerkung 4.19
Die beiden Schranken in Zeile 1 und 2 in Algorithmus 4.17 können oft auch enger gewählt werden. So ist beispielsweise der Radius einer geometrischen Optimallösung eine zulässige untere Schranke und der Radius einer beliebigen k-elementigen Zentrenmenge eine zulässige obere Schranke des graphentheoretischen k-Centers.

Bemerkung 4.20
Wird zu $\tau > 0$ in Algorithmus 4.17 Zeile 13 in

$$(13') \qquad \rho := \min\{\delta(s,v) : s \in S,\ v \in V,\ \delta(s,v) > \rho \cdot (1+\tau)\}$$

verändert, so berechnet dieser eine $(1+\tau)$-Approximationslösung.

Wahl des Parameters ε

Als Wahl für den Parameter ε in Algorithmus 4.17 hat sich ein Wert von 5% bewährt. In Tabelle 4.1 sind für verschiedene Werte von ε die Anzahl von LP-Iterationen, ILP-Iterationen und die Laufzeit in Sekunden als Durchschnitt über jeweils 10 Messungen auf dem Datensatz *Hochsitze* (siehe Abschnitt 5.2) eingezeichnet. Während die Anzahl der gelösten LP-Relaxationen `SC-LP(`ρ`)` gleichmäßig abnimmt, wird die Anzahl der gelösten ILP-Probleme `SC-ILP(`ρ`)` für $\varepsilon = 2\%$ minimal. Für $\varepsilon = 5\%$ ist die Laufzeit sogar noch ein wenig schneller.

ε	LP-Iterationen	ILP-Iterationen	Laufzeit
0,1%	13,4	11,0	57,4 s
0,2%	12,4	10,7	58,0 s
0,5%	11,0	10,4	54,6 s
1%	10,0	9,7	53,6 s
2%	9,0	6,5	42,0 s
5%	7,8	7,4	41,0 s
10%	6,8	9,4	64,2 s

Tabelle 4.1: Wahl des Kontrollparameters ε in Algorithmus 4.17.

Laufzeit des Algorithmus von Ilhan & Pinar

Da Algorithmus 4.17 das graphentheoretische k-Center exakt löst, können wir im Allgemeinen keine polynomielle Laufzeit erwarten.
In Abbildung 4.3 ist die Laufzeit in Sekunden für zunehmendes k in den Datensätzen *Apotheken* (blau) und *Hochsitze* (grün) eingezeichnet. Dabei fällt auf, dass die Laufzeit für zunehmendes k tendenziell sinkt. Dies erklärt sich vermutlich dadurch, dass dann – durch den kleiner werdenden Radius – die dem Problem `SC-ILP(`ρ`)` zugrunde liegende Matrix dünner besetzt ist und dadurch die linearen Programme schneller gelöst werden können.

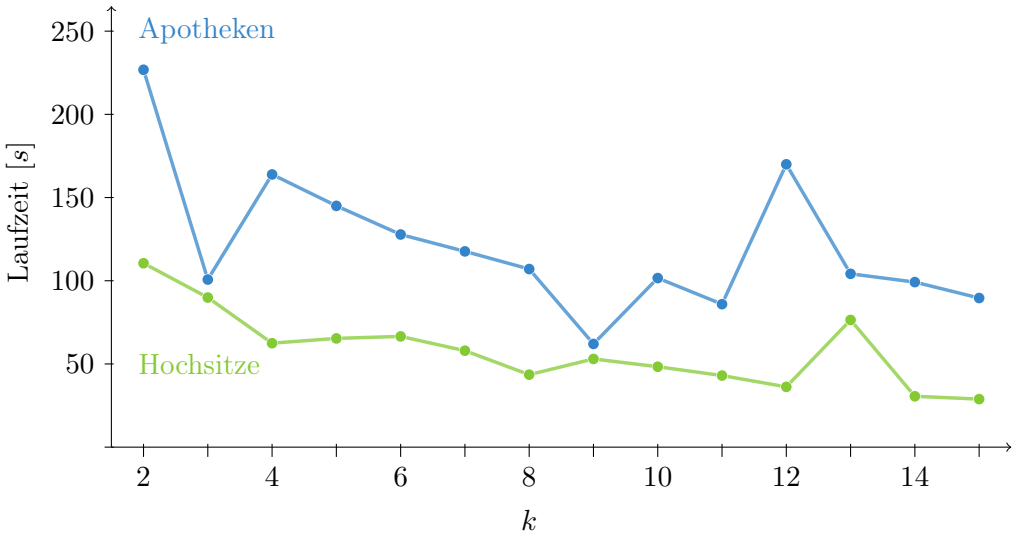

Abbildung 4.3: Laufzeit von Algorithmus 4.17 in Abhängigkeit von k.

In Abbildung 4.4 ist die Laufzeit von Algorithmus 4.17 für die Berechnung eines 6-Centers – diesmal in Abhängigkeit von der Anzahl der Standorte – eingezeichnet. Dabei wurden aus der OpenStreetMap gewonnene Graphen mit zufälliger Standortmenge ohne zusätzliche Zentren verwendet.

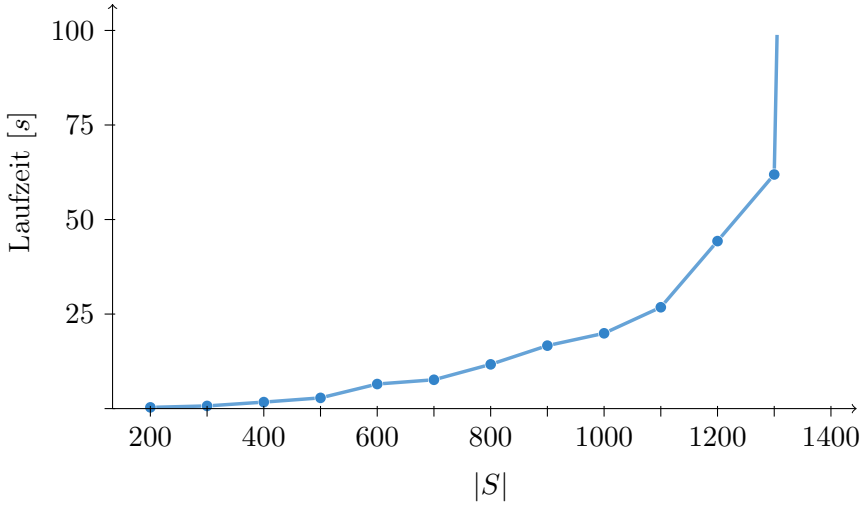

Abbildung 4.4: Laufzeit von Algorithmus 4.17 in Abhängigkeit von $|S|$.

Anhand der Abbildung lässt sich eine exponentielle Steigerung der Laufzeit vermuten. Die Berechnung für 1400 Standorte dauerte 800 Sekunden und für 1500 Standorte wurde nach 24 Stunden immer noch kein Ergebnis gefunden.

5 Vergleich der geometrischen und graphentheoretischen Algorithmen

In den Kapiteln 3 und 4 haben wir Algorithmen kennengelernt, die verschiedene Varianten des Problems k-CENTER lösen. In diesem Kapitel verwenden wir für die Lösung des geometrischen k-Centers den Algorithmus von Brandenberg & Roth, für das graphentheoretische wird der Algorithmus von Ilhan & Pinar (Algorithmus 4.17) verwendet. Unser Ziel ist nun, diese beiden Algorithmen und ihre Lösungen zu vergleichen.
Für den geometrischen Vergleich nutzen wir die Tatsache aus, dass die Knoten im OSM-Graphen Punkte in der Ebene sind und damit auch als Lösung für das geometrische k-Center aufgefasst werden können. Als Maß für die Güte der Lösungen verwenden wir den geometrischen Radius.

Definition 5.1 *(geometrischer Radius)*
Zu einer Instanz des GEOMETRISCHEN k-CENTERS mit Minkowski-Raum $\mathbb{M} = (\mathbb{X}, \|\cdot\|)$ und Standortmenge $S \subseteq \mathbb{M}$ ist für eine beliebige Menge $C \subseteq \mathbb{M}$ mit

$$\rho_{\text{Geo}}(C) = \min\{\rho \geq 0 : S \subseteq \bigcup_{c \in C} c + \rho\mathbb{B}\} \tag{5.1}$$

der *geometrische Radius von C* definiert.

Um aus einer Lösung des geometrischen k-Centers eine zulässige Lösung des graphentheoretischen k-Centers auf einem OSM-Graphen zu erhalten, ersetzen wir jedes geometrische Zentrum c_i durch den dazu nächsten Knoten

$$v(c_i) = \operatorname*{argmin}_{v \in V} \|c_i - v\|_2 \tag{5.2}$$

im Graphen $G = (V, E, \delta)$. Dadurch erhalten wir eine zulässige Lösung des graphentheoretischen k-Centers. Damit die Distanz von c_i und $v(c_i)$ möglichst gering – und damit die Verfahren besser vergleichbar werden – haben wir das verallgemeinerte graphentheoretische k-Center in Abschnitt 4.3 eingeführt. Als Maß für die Güte der Lösungen verwenden wir dann den graphentheoretischen Radius.

Definition 5.2 *(graphentheoretischer Radius)*
Zu einer Instanz des VERALLGEMEINERTEN GRAPHENTHEORETISCHEN k-CENTERS mit vollständigem, gewichteten Graphen $G = (V, E, \delta)$ und einer Standortmenge $S \subseteq V$ ist für eine beliebige Menge $C \subseteq V$ mit

$$\rho_{\text{Graph}}(C) = \max_{s \in S} \min_{c \in C} \delta(c, s) \tag{5.3}$$

der *graphentheoretische Radius von C* definiert.

5.1 Theoretische Ergebnisse

Bevor wir die beiden Verfahren an realen Datensätzen vergleichen, werden wir anhand einfacher Graphen zeigen, dass für den Vergleich beider Radien theoretisch beliebig schlechte Resultate erzielt werden können.
Dazu generieren wir aus einer Instanz des verallgemeinerten graphentheoretischen k-Centers auf dem OSM-Graphen $G = (V, E, \delta)$ mit $S \subseteq V$ und $k \in \mathbb{N}$ ein geometrisches k-Center im Minkowski-Raum $\mathbb{R}^2$ mit Punktemenge S und k aus der graphentheoretischen Variante.

Graphentheoretischer Radius

Für den Fall, dass die Distanzfunktion im Graphen G der euklidischen Norm entspricht, erhalten wir aus einer Optimallösung des geometrischen k-Centers eine 2-Approximation des graphentheoretischen k-Centers.

Lemma 5.3
Wird eine optimale Lösung C_{Geo} des euklidischen k-Centers auf die jeweils nächsten Punkte im OSM-Graphen $G = (V, E, \delta)$ abgebildet, so erhalten wir eine zulässige 2-Approximation für das graphentheoretische k-Center mit Distanzfunktion $\delta(u, v) = \|u - v\|_2$.

Beweis Nach Konstruktion gilt für $\rho^* = \rho_{\text{Geo}}(C_{\text{Geo}})$

$$V \subseteq \bigcup_{c \in C_{\text{Geo}}} (c + \rho^* \mathbb{B}_2^2)$$

das heißt für alle $v \in V$ existiert ein $c_v \in C_{\text{Geo}}$

$$\|v - c_v\|_2 \leq \rho^*.$$

Wird nun jeder Punkt $c \in C_{\text{Geo}}$ auf den dazu nächsten Punkt $v(c) \in V$ abgebildet, so gilt offensichtlich $\|c - v(c)\|_2 \leq \rho^*$, da sich $v(c)$ in der Kreisscheibe $c + \rho^* \mathbb{B}_2^2$ befinden muss. Aus der Dreiecksungleichung folgt für alle v

$$\|v - v(c_v)\|_2 \leq \|v - c_v\|_2 + \|c_v - v(c_v)\|_2 \leq 2\rho^*.$$

Daher gilt auch

$$V \subseteq \bigcup_{c \in C_{\text{Geo}}} (v(c) + 2 \cdot \rho^* \mathbb{B}_2^2).$$

□

Bemerkung 5.4
Die Aussage von Lemma 5.3 lässt sich noch etwas verallgemeinern: Steht keine optimale Lösung für das geometrische k-Center zur Verfügung, sondern nur eine $(1+\varepsilon)$-Approximation, so kann entsprechend eine Schranke von $2 + 2\varepsilon$ garantiert werden.

Bemerkung 5.5
Zwar wurde in Lemma 5.3 nicht das verallgemeinerte graphentheoretische k-Center betrachtet, jedoch lässt sich auch in diesem Fall ohne zusätzliche Anforderungen an die Verteilung der möglichen Zentren keine schärfere Schranke angeben.

Obwohl die, durch die Distanzmatrix D im OSM-Graphen $G = (V, E, \delta)$ induzierte Gewichtsfunktion δ_D nach Bemerkung 2.5 symmetrisch ist und die Dreiecksungleichung erfüllt, kann für beliebige OSM-Graphen keine zu Lemma 5.3 analoge Aussage getroffen werden, sondern sogar jede beliebige Schranke übertroffen werden.

Lemma 5.6
Zu jedem $\xi > 1$ existiert ein vollständiger, gewichteter Graph $G = (V, E, \delta)$ mit symmetrischer Gewichtsfunktion, die die Dreiecksungleichung erfüllt, sodass für die Optimallösung C_{Graph} der graphentheoretischen Variante und die gerundete Zentrenmenge $v(C_{\text{Geo}})$ einer optimalen Lösung C_{Geo} des zugehörigen euklidischen k-Centers

$$\frac{\rho_{\text{Graph}}(v(C_{\text{Geo}}))}{\rho_{\text{Graph}}(C_{\text{Graph}})} \geq \xi$$

gilt. Dabei bezeichnet ρ_{Graph} den graphentheoretischen Radius aus Gleichung (5.3).

Beweis Sei $\xi > 1$ beliebig. Wir betrachten den Graphen aus Abbildung 5.1. Die Distanzfunktion δ ist definiert durch die jeweils kürzeste Verbindung auf dem, durch die blauen Kanten angegebenen Graphen. Nach Bemerkung 2.5 ist sie symmetrisch und erfüllt die Dreiecksungleichung. Eine optimale Lösung des geometrischen 2-Centers mit den Zentren c_1 und c_2 ist ebenfalls eingezeichnet. Für die Abbildung v aus Gleichung (5.2), die die Zentren auf den jeweils nächsten Knoten aus V abbildet, gilt $v(c_1) = v_2$ und $v(c_2) = v_5$.

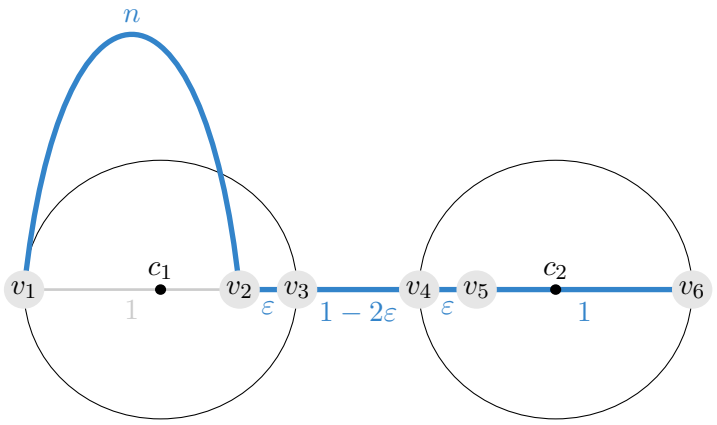

Abbildung 5.1: Problematische geometrische 2-Center-Instanz.

Für $n \geq \xi$ hat die gerundete geometrische Lösung einen graphentheoretischen Radius von n. Der Optimalwert eines `GRAPHENTHEORETISCHEN 2-CENTERS` beträgt hingegen 1 mit Zentren in v_1 und v_5 und somit gilt

$$\frac{\rho_{\text{Graph}}(C_{\text{Geo}})}{\rho_{\text{Graph}}(C_{\text{Graph}})} = \frac{\rho_{\text{Graph}}(\{v_2, v_5\})}{\rho_{\text{Graph}}(\{v_1, v_5\})} = \frac{n}{1} \geq \xi.$$

□

Der graphentheoretische Radius einer gerundeten Optimallösung des geometrischen k-Centers kann also beliebig von der tatsächlichen Optimallösung der graphentheoretischen Variante abweichen.

Geometrischer Radius

Auch umgekehrt, wenn wir die Knoten einer Optimallösung des graphentheoretischen k-Centers als Punkte in $\mathbb{R}^2$ auffassen und mit einer Lösung des geometrischen k-Centers vergleichen, können wir beliebig schlechte Ergebnisse erzielen.

Lemma 5.7
Es existieren Instanzen des verallgemeinerten graphentheoretischen k-Centers, sodass für jedes $\xi > 1$ und für Optimallösungen C_{Graph} des graphentheoretischen k-Centers und Optimallösungen C_{Geo} der daraus gewonnenen geometrischen Variante

$$\frac{\rho_{\text{Geo}}(C_{\text{Graph}})}{\rho_{\text{Geo}}(C_{\text{Geo}})} \geq \xi \tag{5.4}$$

gilt. Dabei bezeichnet ρ_{Geo} den geometrischen Radius aus Gleichung (5.1).

Beweis Sei $\xi > 1$ beliebig. Wir betrachten den Graphen aus Abbildung 5.2. Die Distanzfunktion δ ist definiert durch die jeweils kürzeste Verbindung auf dem, durch die blauen Kanten angegebenen Graphen. Eine optimale graphentheoretische Lösung des 2-Centers verwendet die Zentren v_1 und v_3. Mit c_1 und c_2 ist auch eine optimale Lösung der geometrischen Variante eingezeichnet.

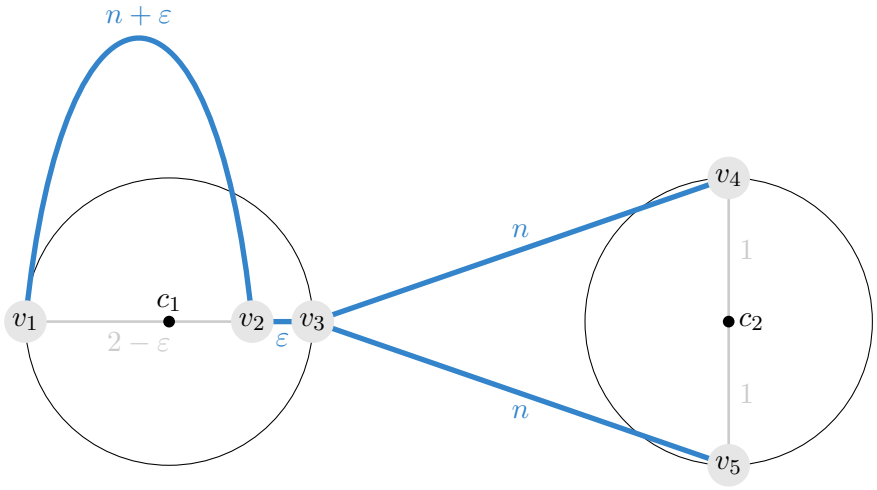

Abbildung 5.2: Problematische graphentheoretische 2-Center-Instanz.

Für $n \geq \xi$ gilt für die in Gleichung (5.4) angegebenen geometrischen Radien:

$$\frac{\rho_{\text{Geo}}(C_{\text{Graph}})}{\rho_{\text{Geo}}(C_{\text{Geo}})} = \frac{\rho_{\text{Geo}}(\{v_1, v_3\})}{\rho_{\text{Geo}}(\{c_1, c_2\})} = \frac{n}{1} \geq \xi$$

□

Analog zum graphentheoretischen Radius sind also auch beim geometrischen Radius theoretisch beliebig große Abweichungen möglich, wenn ein graphentheoretisches k-Center als Lösung der geometrischen Variante aufgefasst wird.

5.2 Datensätze

Im Zuge dieser Diplomarbeit entstand ein Programm, welches die vorgestellten Algorithmen implementiert. Damit können Datensätze auf Basis der OpenStreetMap erzeugt und anschließend als Grundlage für den Vergleich der geometrischen und graphentheoretischen Algorithmen verwendet werden.
Im Folgenden werden erst die verwendeten Datensätze vorgestellt, bevor die daraus erzielten Ergebnisse präsentiert werden.

Apotheken Der Datensatz *Apotheken* besteht aus 766 Apotheken in Österreich. Zusätzlich wurden für 11.918 mögliche Zentren die Entfernungen mitberechnet.

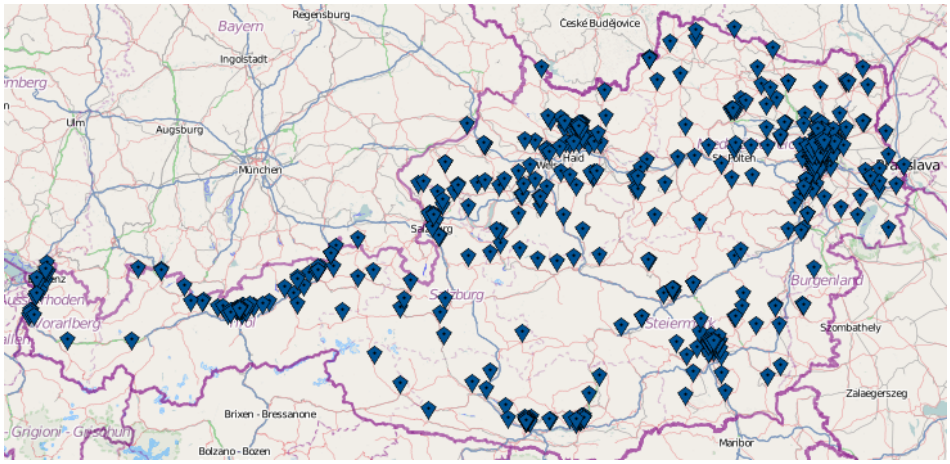

Abbildung 5.3: Die 766 Standorte vom Datensatz *Apotheken*.

Touristenattraktionen Der Datensatz *Touristenattraktionen* besteht aus 376 Touristenattraktionen in Österreich. Zusätzlich wurden für 12.228 mögliche Zentren die Entfernungen mitberechnet.

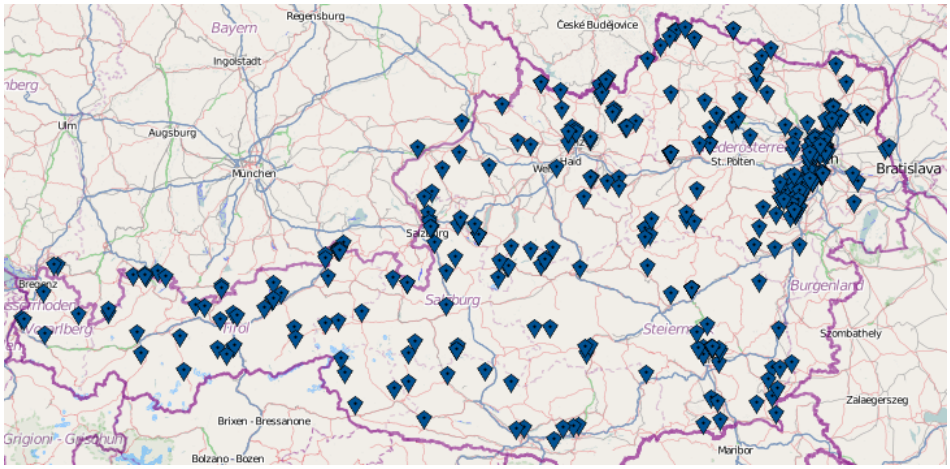

Abbildung 5.4: Die 376 Standorte vom Datensatz *Touristenattraktionen*.

Hochsitze Der Datensatz *Hochsitze* besteht aus 687 Hochsitzen in Bayern. Zusätzlich wurden für 6.341 mögliche Zentren die Entfernungen mitberechnet. Da sich Hochsitze im Allgemeinen abseits der Hauptverkehrswege befinden und sich jeweils in Wäldern gruppieren, ergibt sich eine besonders inhomogene Verteilung.

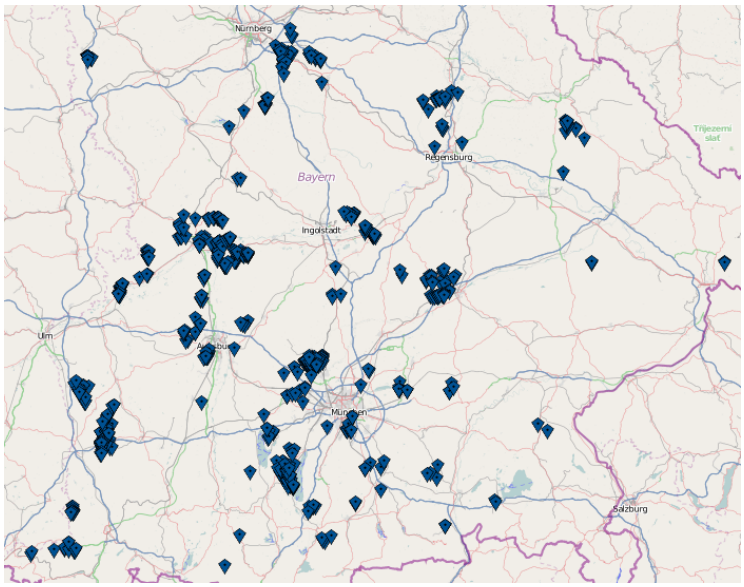

Abbildung 5.5: Die 687 Standorte vom Datensatz *Hochsitze*.

Krankenhäuser Der Datensatz *Krankenhäuser* besteht aus 303 Krankenhäusern in Bayern. Zusätzlich wurden für 5.070 mögliche Zentren die Entfernungen mitberechnet. Im Gegensatz zum Datensatz *Hochsitze* sind die Krankenhäuser wesentlich gleichmäßiger verteilt.

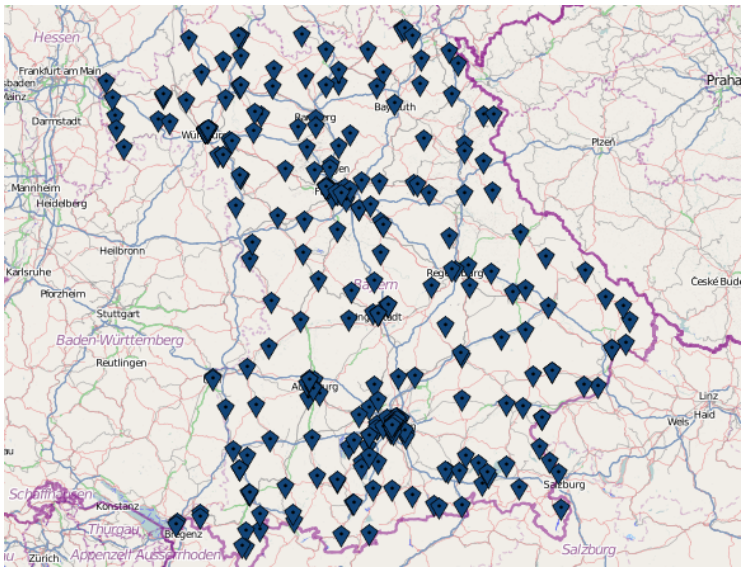

Abbildung 5.6: Die 303 Standorte vom Datensatz *Krankenhäuser*.

Aldis Der Datensatz *Aldis* besteht aus allen 221 in der OpenStreetMap eingetragenen Aldi-Filialen in Bayern. Zusätzlich wurden für 4.814 mögliche Zentren die Entfernungen mitberechnet.

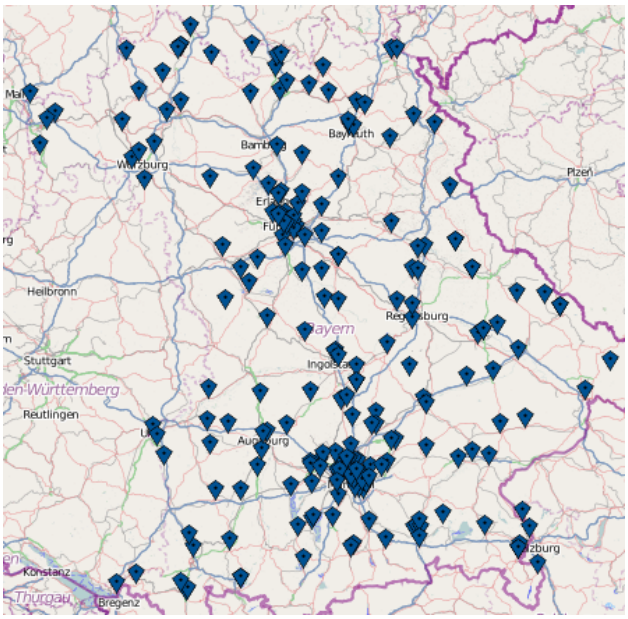

Abbildung 5.7: Die 221 Standorte vom Datensatz *Aldis*.

Biergärten Der Datensatz *Biergärten* besteht aus 743 Biergärten in Bayern. Dabei wurden zusätzlich für 10.127 mögliche Zentren die Entfernungen mitberechnet.

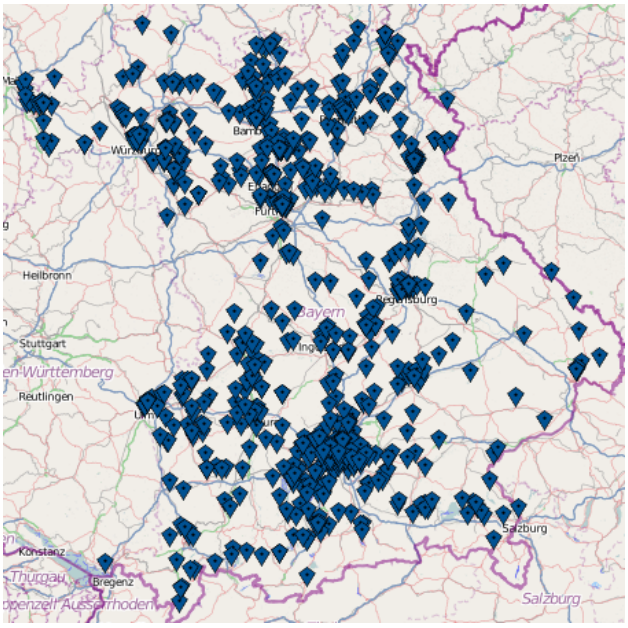

Abbildung 5.8: Die 743 Standorte vom Datensatz *Biergärten*.

5.3 Vergleich der Radien

In Abschnitt 5.1 haben wir gesehen, dass die Abweichungen, wenn wir eine Optimallösung einer Variante als Lösung der anderen Variante von k-Center auffassen, verglichen mit dem Radius der tatsächlich optimalen Lösung, beliebig groß werden können. Nun wollen wir diese Fragestellung auf die betrachteten Datensätze übertragen. Dabei werden wir feststellen, dass sich die Abweichungen in realen Problemstellungen durchaus in Grenzen halten.

5.3.1 Geometrischer Radius

In Tabelle 5.1 sind die geometrischen Radien (in Kilometern) der Zentren einer geometrischen und einer graphentheoretischen Lösung, sowie die Abweichung, die durch die graphentheoretische Lösung verursacht wird, für die unterschiedlichen Datensätze eingetragen.

Datensatz	Geometrisch	Graphentheoretisch	Abweichung
Aldis ($k = 3$)	113,96	116,86	1,7%
Aldis ($k = 5$)	82,69	87,80	6,2%
Aldis ($k = 7$)	64,09	72,09	12,5%
Aldis ($k = 10$)	50,27	54,31	8,0%
Apotheken ($k = 3$)	122,08	138,28	13,3%
Apotheken ($k = 5$)	91,41	103,03	12,7%
Apotheken ($k = 7$)	68,17	83,23	22,1%
Apotheken ($k = 10$)	55,47	64,52	16,3%
Biergärten ($k = 3$)	123,25	128,54	4,3%
Biergärten ($k = 5$)	81,50	86,89	6,6%
Biergärten ($k = 7$)	66,82	72,83	9,0%
Biergärten ($k = 10$)	-	56,45	‡
Krankenhäuser ($k = 3$)	125,97	126,15	0,1%
Krankenhäuser ($k = 5$)	86,27	92,99	7,8%
Krankenhäuser ($k = 7$)	66,21	71,87	8,5%
Krankenhäuser ($k = 10$)	51,73	58,76	13,6%
Hochsitze ($k = 3$)	81,82	83,80	2,4%
Hochsitze ($k = 5$)	59,72	61,86	3,6%
Hochsitze ($k = 7$)	45,29	46,21	2,0%
Hochsitze ($k = 10$)	34,95	39,54	13,1%
Touristenattraktionen ($k = 3$)	126,12	133,48	5,8%
Touristenattraktionen ($k = 5$)	84,71	117,28	38,5%
Touristenattraktionen ($k = 7$)	67,03	87,03	29,8%
Touristenattraktionen ($k = 10$)	55,02	71,52	30,0%

Tabelle 5.1: Vergleich des geometrischen Radius.

In Tabelle 5.1 fällt auf, dass die österreichischen Datensätze vergleichsweise schlechte Ergebnisse erzielen. Bei der Analyse von über 100 Beispielen erhielten wir eine durchschnittliche Abweichung des geometrischen Radius von 12.1%. Die Abweichung von 38.5% beim 5-Center auf dem Datensatz *Touristenattraktionen* wurde dabei nie übertroffen.

‡Die Berechnung für das geometrische 10-Center wurde nach 60 Stunden ohne Ergebnis abgebrochen.

5.3.2 Graphentheoretischer Radius

In Tabelle 5.2 sind die graphentheoretischen Radien (in Kilometern) der Zentren einer graphentheoretischen und einer geometrischen Lösung, sowie die Abweichung, die durch die geometrische Lösung verursacht wird, für die unterschiedlichen Datensätze eingetragen.

Datensatz	Graphentheoretisch	Geometrisch	Abweichung
Aldis ($k = 3$)	141,80	146,97	3,6%
Aldis ($k = 5$)	99,42	108,75	9,4%
Aldis ($k = 7$)	80,33	117,37	46,1%
Aldis ($k = 10$)	62,23	71,08	14,1%
Apotheken ($k = 3$)	174,44	202,92	16,3%
Apotheken ($k = 5$)	119,07	141,87	19,2%
Apotheken ($k = 7$)	92,84	128,79	38,7%
Apotheken ($k = 10$)	75,05	85,66	14,1%
Biergärten ($k = 3$)	146,74	156,97	7,0%
Biergärten ($k = 5$)	102,97	108,86	5,7%
Biergärten ($k = 7$)	84,53	102,99	21,8%
Biergärten ($k = 10$)	65,66	-	‡
Krankenhäuser ($k = 3$)	145,28	151,86	4,5%
Krankenhäuser ($k = 5$)	105,04	120,21	14,5%
Krankenhäuser ($k = 7$)	83,08	118,47	42,6%
Krankenhäuser ($k = 10$)	67,14	76,25	13,6
Hochsitze ($k = 3$)	98,19	102,19	4,1%
Hochsitze ($k = 5$)	70,99	74,48	4,9%
Hochsitze ($k = 7$)	53,72	61,80	15,0%
Hochsitze ($k = 10$)	45,06	61,80	37,1%
Touristenattraktionen ($k = 3$)	180,33	208,25	15,5%
Touristenattraktionen ($k = 5$)	130,00	165,01	26,9%
Touristenattraktionen ($k = 7$)	101,19	117,88	16,5%
Touristenattraktionen ($k = 10$)	84,93	107,58	26,7%

Tabelle 5.2: Vergleich des graphentheoretischen Radius.

Dabei fällt auf, dass die Abweichungen bei den österreichischen Datensätzen *Apotheken* und *Touristenattraktionen* um einiges höher sind als bei den Datensätzen aus Bayern. Dies liegt vermutlich daran, dass die Abweichungen der Weglänge zur Luftlinie aufgrund der gebirgigen Landschaft stärker variieren.
In Abbildung 5.9 haben wir dies näher untersucht. Dort ist eingezeichnet um wieviel Prozent die Weglänge zwischen zwei Punkten im OSM-Graphen länger als die Luftlinienentfernung ist, wobei die y-Achse auf die Gesamtanzahl der Kanten normiert wurde. Links sind die vier Datensätze aus Bayern eingezeichnet und rechts die beiden aus Österreich. Während die bayerischen Wegstrecken relativ konstant knapp 20% länger als die Luftliniendistanzen sind, variiert dieser Wert bei den österreichischen Datensätzen viel stärker.

Bei der Analyse von über 100 Datensätzen stellten wir beim graphentheoretischen Radius eine durchschnittliche Abweichung von 17.5% fest. Die größte Abweichung betrug 72.2% beim 9-Center auf dem Datensatz *Touristenattraktionen*.
In den Abbildungen 5.10 und 5.11 sind diese beiden Lösungen gegenübergestellt. In weiß ist

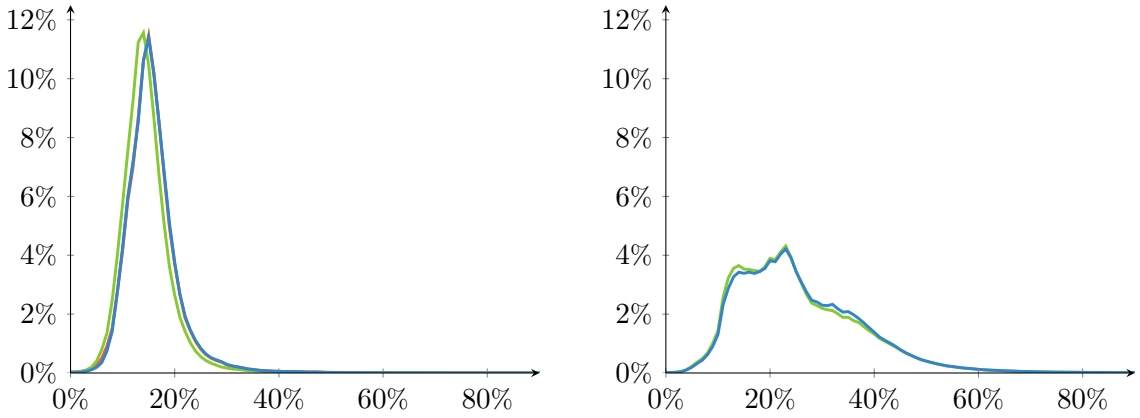

Abbildung 5.9: Dargestellt ist der Anteil der Kanten in einem Datensatz über dem Prozentsatz, um den die Weglänge einer Kante länger ist als die Luftlinienentfernung. Links sind die vier bayerischen Datensätze, rechts die beiden österreichischen eingezeichnet.

jeweils der Standort mit der größten Entfernung zum Zentrum markiert. Der kürzeste Weg zum Zentrum ist blau eingezeichnet. In Abbildung 5.11 sind in grau die vom geometrischen Algorithmus gefundenen Zentren eingezeichnet, während die gerundeten Zentren schwarz markiert sind.

Der Radius der graphentheoretischen Lösung beträgt 88,57 Kilometer, der der geometrischen Lösung hingegen 152,52 Kilometer und hat damit eine Abweichung von $72,2\%$.

Selbst wenn man zusätzlich Straßen über Italien betrachtet, ändert sich der Radius nicht, da sich die Ursache der Abweichung darin begründet, dass die geometrische Lösung ein Zentrum in das Zillertal setzt, aus welchem der weiß markierte Standort nur über einen großen Umweg erreicht werden kann.

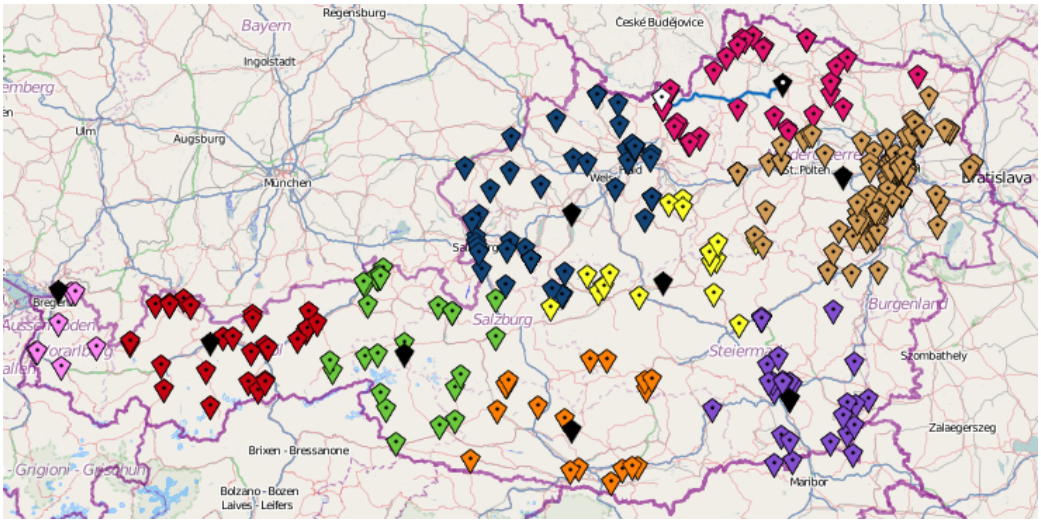

Abbildung 5.10: Graphentheoretisches 9-Center auf dem Datensatz *Touristenattraktionen*.

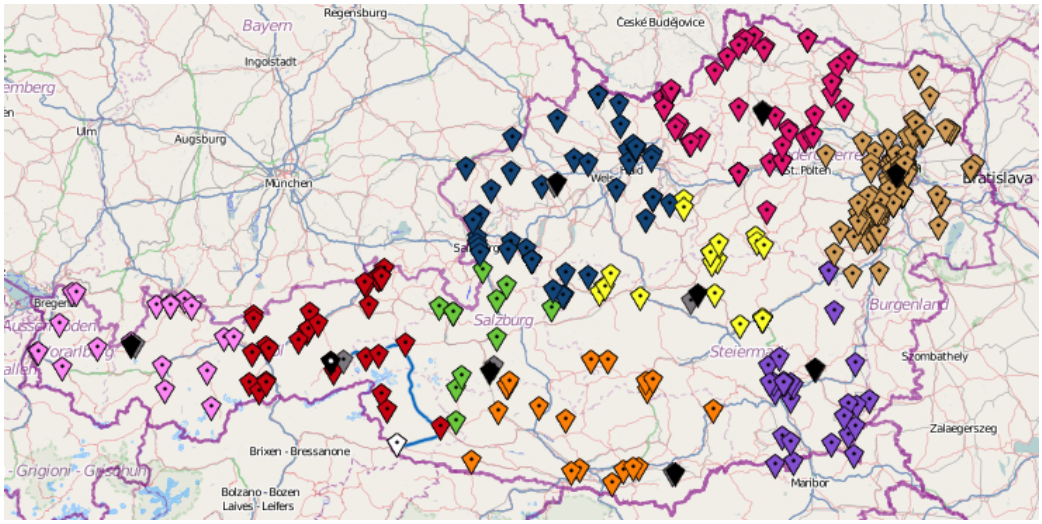

Abbildung 5.11: Geometrisches 9-Center auf dem Datensatz *Touristenattraktionen*.

5.4 Vergleich der Laufzeit

In den Tabellen 5.3 und 5.4 sind die Laufzeiten in Sekunden des geometrischen und des graphentheoretischen Algorithmus für eine steigende Anzahl an Zentren in den verschiedenen Datensätzen eingetragen.

Datensatz	$k = 2$	$k = 4$	$k = 6$	$k = 8$	$k = 10$
Aldis (221 Standorte)	0,01	0,13	2,41	28,73	1.970,44
Apotheken (766 Standorte)	0,02	0,13	4,11	15,62	169,49
Biergärten (743 Standorte)	0,03	0,32	3,77	8.323,55	‡
Krankenhäuser (303 Standorte)	0,01	0,17	2,94	261,64	1.568,66
Hochsitze (687 Standorte)	0,02	0,22	13,54	9,81	15,76
Touristenattraktionen (376 Standorte)	0,02	0,26	6,11	21,41	126,94

Tabelle 5.3: Laufzeit des geometrischen Algorithmus in Abhängigkeit von k.

Datensatz	$k = 2$	$k = 4$	$k = 6$	$k = 8$	$k = 10$
Aldis (221 Standorte)	20,99	15,28	14,03	12,8	11,54
Apotheken (766 Standorte)	254,46	192,99	157,10	135,85	128,84
Biergärten (743 Standorte)	208,09	369,29	328,6	116,47	171,46
Krankenhäuser (303 Standorte)	17,5	13,79	15,42	11,87	12,8
Hochsitze (687 Standorte)	110,52	68,95	66,55	58,63	50,03
Touristenattraktionen (376 Standorte)	103,93	78,5	72,91	60,78	61,38

Tabelle 5.4: Laufzeit des graphentheoretischen Algorithmus in Abhängigkeit von k.

Zum Vergleich sind in Abbildung 5.12 die Laufzeiten der beiden Algorithmen für den Datensatz *Apotheken* (766 Standorte, 11.918 Zentren) bei einer zunehmenden Anzahl an Zentren auf einem Bild gegenübergestellt. Für $k = 12$ beträgt die geometrische Laufzeit schon über 1.200 Sekunden, während die graphentheoretische Laufzeit für zunehmende k weiterhin in

etwa konstant bleibt.

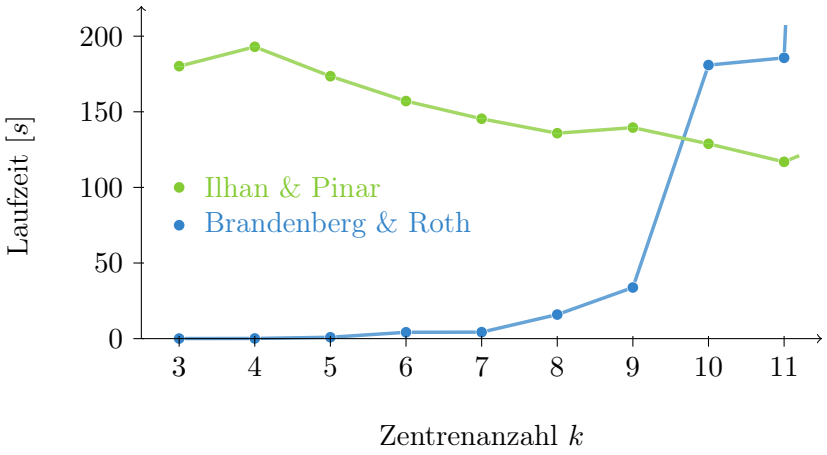

Abbildung 5.12: Vergleich der Laufzeiten in Abhängigkeit von k.

Betrachten wir nun die Laufzeit in Abhängigkeit von der Anzahl der Standorte, so haben wir in Abschnitt 4.4 gesehen, dass der graphentheoretische Algorithmus eine exponentielle Laufzeit aufweist. Der geometrische Algorithmus von Brandenberg & Roth skaliert hier wesentlich besser.

Für den Vergleich verwenden wir einen Basisdatensatz und erhöhen die Standortanzahl, indem wir zufällige, mögliche Zentren als Standorte auffassen. In Abbildung 5.13 sind die Laufzeiten des geometrischen und graphentheoretischen Algorithmus für ein 8-Center auf dem Datensatz *Aldis* für eine zunehmende Anzahl an Standorten gegenübergestellt. Für $|S| > 1500$ konnte die Laufzeit des graphentheoretischen Algorithmus nicht mehr bestimmt werden.

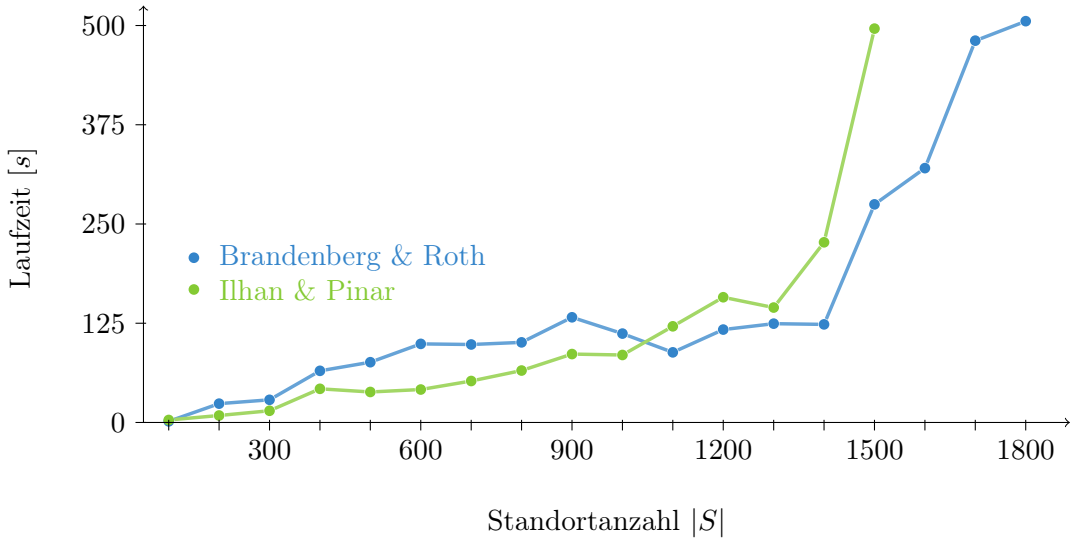

Abbildung 5.13: Vergleich der Laufzeiten in Abhängigkeit von $|S|$.

5.5 Geometrisches k-Center für große k

Wie wir zuvor gesehen haben, ist der geometrische Algorithmus für große k wesentlich langsamer als der graphentheoretische Algorithmus. Daher bietet es sich an, stattdessen ein graphentheoretisches k-Center mit der Luftlinienentfernung als Kantengewicht zu betrachten. Die Distanzmatrix kann dabei offensichtlich schnell in $\mathcal{O}(n^2 d)$ bestimmt werden.
Ohne zusätzliche Zentren haben wir in Lemma 5.3 gesehen, dass wir aus einer graphentheoretischen Lösung nur eine 2-Approximation für das geometrische k-Center erhalten. Jedoch können wir mit dieser Aussage durch Verwendung äquidistant verteilter, zusätzlicher Zentren auf einem Gitter mit Maschenbreite λ eine verbesserte Approximationsgüte in Abhängigkeit von λ beweisen.

Satz 5.8
Betrachte den d-dimensionalen Minkowski-Raum $\mathbb{R}^d$ mit Norm $\|\cdot\|_2$ und $S \subseteq \mathbb{R}^d$. Dadurch wird ein geometrisches k-Center mit einer Lösung C_{Geo} und Radius ρ_{Geo} festgelegt. Zu $\lambda > 0$ sei außerdem ein verallgemeinertes graphentheoretisches k-Center auf dem vollständigen, gewichteten Graphen $G = (V, E, \delta)$ mit Knotenmenge

$$V = S \cup \lambda \mathbb{Z}^d$$

und Gewichtsfunktion $\delta : V \times V \to \mathbb{R}^+$, $(u, v) \mapsto \|u - v\|_2$ mit einer Lösung C_{Graph} und Radius ρ_{Graph} gegeben.
Dann gilt für eine untere Schranke $\bar{\rho}$ von ρ_{Geo}:

$$\frac{\rho_{\text{Graph}}}{\rho_{\text{Geo}}} \leq 1 + \frac{\sqrt{d}\lambda}{2\bar{\rho}} \tag{5.5}$$

Beweis Nach Wahl der Distanzfunktion stimmen für jede Menge $C \subseteq V$ der geometrische und der graphentheoretische Radius überein. Diesen bezeichnen wir mit $\rho(C)$.
Mit den Bezeichnungen wie in Satz 5.8 existiert durch die Wahl der Gitterweite zu jedem $c_i \in C_{\text{Geo}}$ ein $v(c_i) \in V$ mit

$$\|c_i - v(c_i)\|_2 \leq \sqrt{d}\frac{\lambda}{2}.$$

Daher kann für die Menge $C' := \{v(c_i) : c_i \in C_{\text{Geo}}\}$ mit der Dreiecksungleichung der Radius durch

$$\rho(C') \leq \rho_{\text{Geo}} + \sqrt{d}\frac{\lambda}{2}$$

abgeschätzt werden. Mit der Optimalität von ρ_{Graph} folgt

$$\begin{aligned}
\rho_{\text{Graph}} &\leq \rho(C') \\
&\leq \rho_{\text{Geo}} + \sqrt{d}\frac{\lambda}{2} \\
&= (1 + \frac{\sqrt{d}\lambda}{2\rho_{\text{Geo}}})\rho_{\text{Geo}} \\
&\leq (1 + \frac{\sqrt{d}\lambda}{2\bar{\rho}})\rho_{\text{Geo}}
\end{aligned} \tag{5.6}$$

für eine beliebige untere Schranke $\bar{\rho}$ von ρ_{Geo}. □

Bemerkung 5.9
Für eine Lösung C des euklidischen k-Centers zur Standortmenge S gilt $C \subseteq \text{conv}(S)$ (Beweis siehe [9]). Daher reicht es aus in Lemma 5.8 die endliche Knotenmenge

$$V = (S \cup \lambda\mathbb{Z}^d) \cap (\text{conv}(S) + \sqrt{d}\frac{\lambda}{2}\mathbb{B}_2^2)$$

zu verwenden. In der Implementierung wurde statt $\text{conv}(S)$ ein S umfassendes Rechteck verwendet.

Bemerkung 5.10
Als untere Schranke für ρ_{Geo} kann ein beliebiges Verfahren mit bekannter Approximationsgüte verwendet werden. Mit Lemma 5.3 gilt beispielsweise für den graphentheoretischen Radius $\rho_{\text{Graph}} \leq 2\rho_{\text{Geo}}$ und somit erhalten wir

$$\frac{\rho_{\text{Graph}}}{\rho_{\text{Geo}}} \leq 1 + \frac{\sqrt{d}\lambda}{\rho_{\text{Graph}}} \tag{5.7}$$

als Approximationsgüte.

Bemerkung 5.11
Die Approximationsgüte kann oft noch verschärft werden. Dazu stellen wir fest, dass wir in (5.7) die Hälfte des graphentheoretischen Radius ρ_{Graph} als untere Schranke für ρ_{Geo} in (5.6) verwendet haben und dies meist eine schwache Abschätzung des geometrischen Radius darstellt.
Verwenden wir stattdessen rekursiv (5.7) als neue untere Schranke für den geometrischen Radius, so erhalten wir eine Folge

$$a_0 = 2$$
$$a_{n+1} = 1 + \frac{\sqrt{d}\lambda}{2\rho_{\text{Graph}}} a_n$$

mit $\frac{\rho_{\text{Graph}}}{\rho_{\text{Geo}}} \leq a_n \ \forall n$, die für $\sqrt{d}\lambda < \rho_{\text{Graph}}$ gegen den Grenzwert

$$\lim_{n\to\infty} a_n = \frac{1}{1 - \frac{\sqrt{d}\lambda}{2\rho_{\text{Graph}}}}$$

konvergiert. Dieser liefert uns die verschärfte Approximationsgüte

$$\frac{\rho_{\text{Graph}}}{\rho_{\text{Geo}}} \leq \frac{1}{1 - \frac{\sqrt{d}\lambda}{2\rho_{\text{Graph}}}}. \tag{5.8}$$

Wie wir in Abschnitt 4.4 gesehen haben, kann der Algorithmus von Ilhan & Pinar nur auf Graphen mit einer kleinen Anzahl von Standorten angewandt werden. Um im obigen Ansatz eine höhere Genauigkeit zu erreichen, wird jedoch nur $|V|$ und nicht $|S|$ erhöht.

In den Abbildungen 5.14 und 5.15 sind für die Datensätze *Aldis* und *Hochsitze* die Approximationsgüten nach Gleichung (5.8), sowie die Laufzeiten in Abhängigkeit von der Anzahl der betrachteten Gitterpunkte eingezeichnet. Offensichtlich nimmt die Approximationsgüte

umgekehrt proportional zur Anzahl der Gitterpunkte ab, während die Laufzeit in dem betrachteten Intervall linear ansteigt. Während auf dem Datensatz *Aldis* mit nur 221 Standorten auch eine große Anzahl an Gitterpunkten verwendet werden kann, reicht auf dem Datensatz *Hochsitze* mit 687 Standorten ab 43.681 Gitterpunkten der Arbeitsspeicher auf einem handelsüblichen Laptop[6] nicht mehr aus.

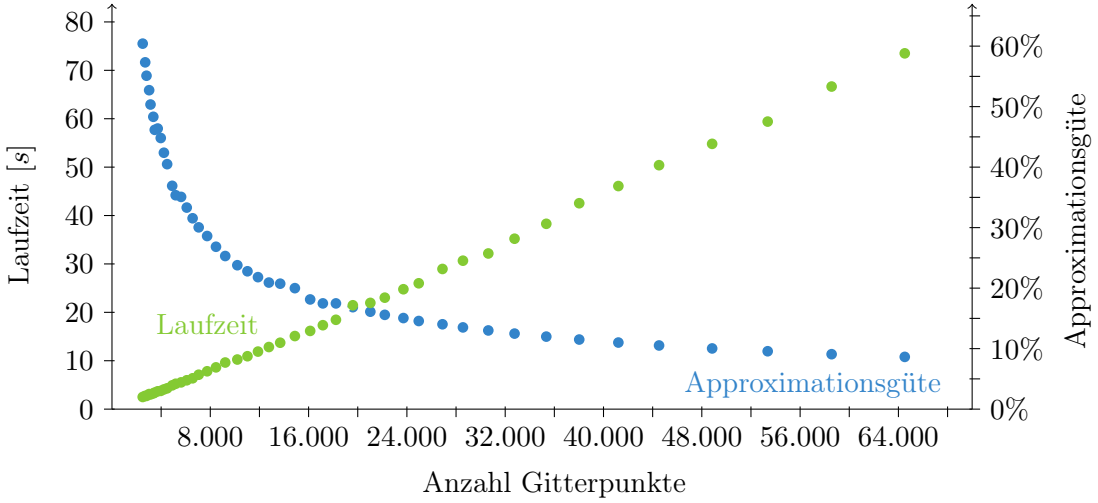

Abbildung 5.14: Approximationsgüte und Laufzeit von *Aldis* für $k = 15$.

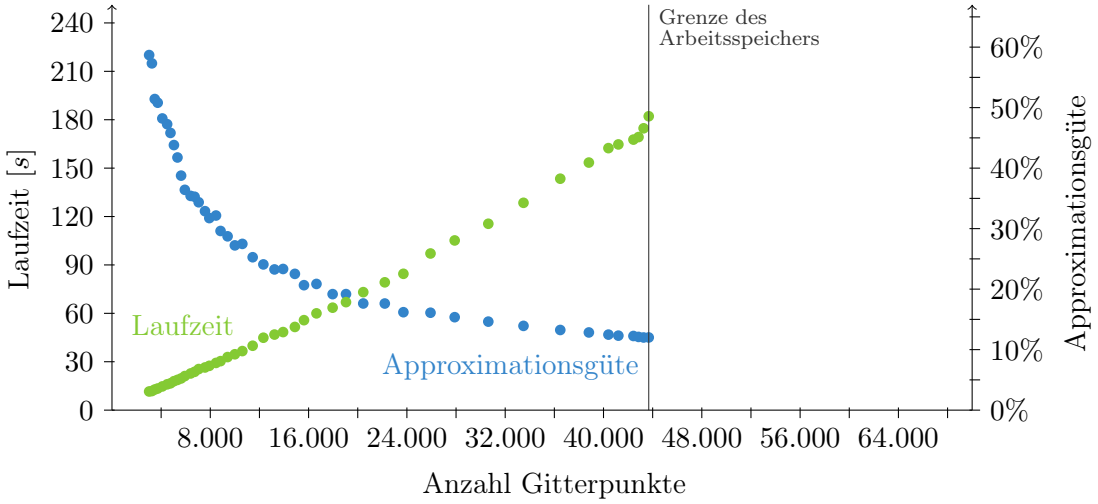

Abbildung 5.15: Approximationsgüte und Laufzeit von *Hochsitze* für $k = 15$.

In Abbildung 5.16 wurde die Laufzeit dieses Verfahrens mit dem geometrischen Algorithmus von Brandenberg & Roth verglichen. Dabei wurde der Datensatz *Aldis* mit einer Approximationsgüte von 10% verwendet. Der Algorithmus von Brandenberg & Roth ist für $k > 17$ wesentlich langsamer als die geometrische Variante des Verfahrens von Ilhan & Pinar.

[6] Intel Pentium T3200 (2.0 GHz), 2GB DDR2

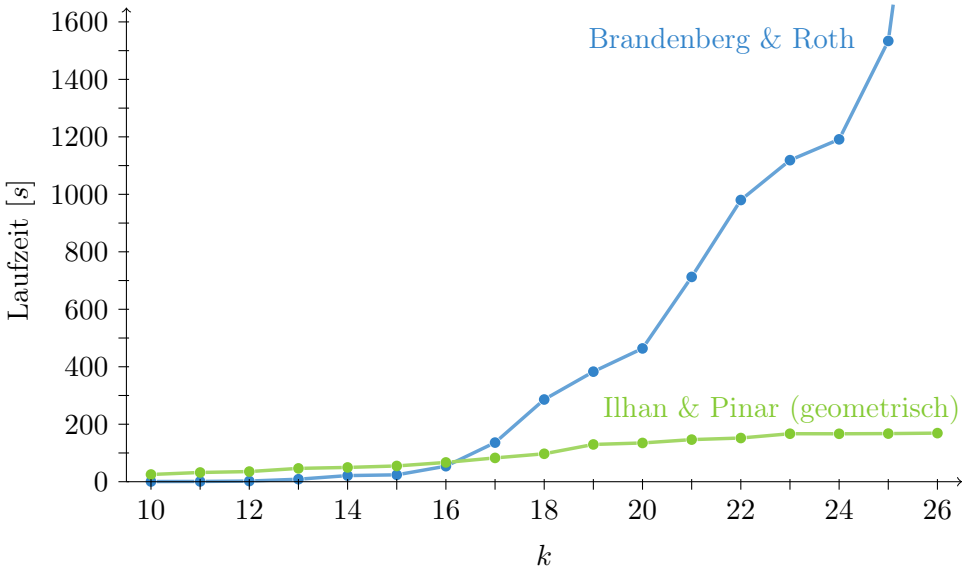

Abbildung 5.16: Laufzeiten der geometrischen k-Center-Algorithmen auf dem Datensatz *Aldis* bei 10% Approximationsgüte.

Da der Radius mit zunehmendem k abnimmt, muss auch die Gitterweite im Algorithmus von Ilhan & Pinar immer weiter reduziert werden, um eine konstante Approximationsgüte von 10% zu erhalten. Dadurch nimmt die Anzahl der betrachteten Gitterpunkte von 23.716 bei $k = 10$ auf 129.600 bei $k = 25$ zu. Aufgrund des begrenzten Arbeitsspeichers kann daher die Zentrenanzahl nicht beliebig weiter erhöht werden.

In Abbildung 5.17 ist die Länge des gefundenen Radius bei beiden Verfahren eingezeichnet. Es fällt auf, dass die Abschätzung bei der geometrischen Variante von Ilhan & Pinar sehr pessimistisch ausfällt, da bei gleicher geforderter Toleranz der gefundene Radius im Durchschnitt nur 93,9% des Radius einer vom Algorithmus von Brandenberg & Roth gefundenen Lösung beträgt.

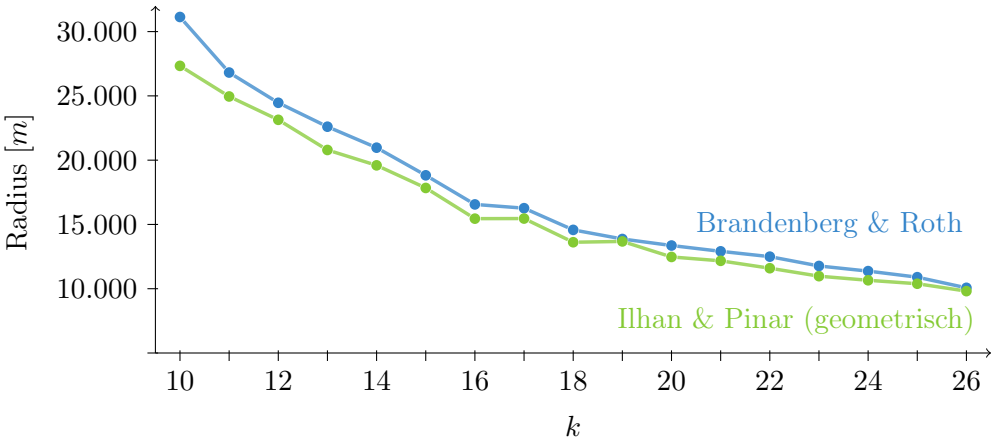

Abbildung 5.17: Radien der geometrischen k-Center-Algorithmen auf dem Datensatz *Aldis* bei 10% Approximationsgüte.

Der Datensatz den wir eben verwendet haben, besitzt nur wenige Standorte. Für größere Standortmengen schneidet der Algorithmus von Ilhan & Pinar im Vergleich wesentlich schlechter ab. In Abbildung 5.18 wurden wieder die Laufzeiten der beiden Algorithmen verglichen. Dabei wurde der Datensatz *Hochsitze* mit einer Approximationsgüte von 20% verwendet.

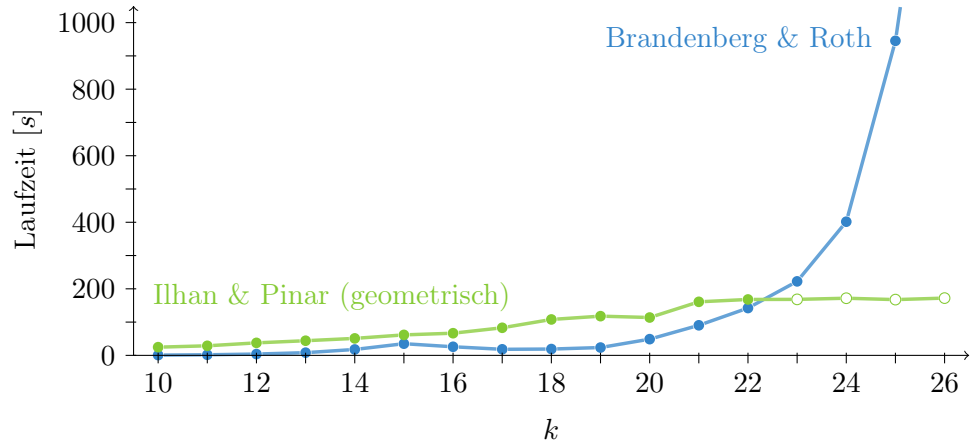

Abbildung 5.18: Laufzeiten der geometrischen k-Center-Algorithmen auf dem Datensatz *Hochsitze* bei 20% Approximationsgüte.

Obwohl nur 20% Approximationsgüte gefordert wurden, kann der Algorithmus von Ilhan & Pinar diese ab $k = 23$ nicht mehr garantieren, da sonst über 44.000 Gitterpunkte betrachtet würden. Wie wir in Abbildung 5.15 gesehen haben, wird dann die Grenze des Arbeitsspeichers erreicht.
In Abbildung 5.19 ist wiederum die Länge des gefundenen Radius bei beiden Verfahren eingezeichnet. Dabei sehen wir, dass die Lösungen für $k \geq 23$ trotz nicht erreichter Approximationsgüte einen geringeren Radius haben als die, die vom Algorithmus von Brandenberg & Roth gefunden wurden.

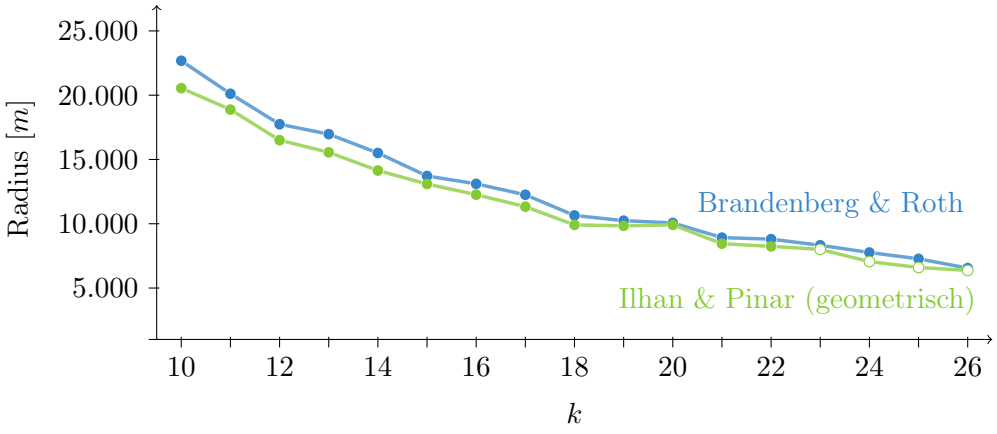

Abbildung 5.19: Radien der geometrischen k-Center-Algorithmen auf dem Datensatz *Hochsitze* bei 20% Approximationsgüte.

6 Abschließende Bemerkungen und Ausblick

Das Ziel dieser Diplomarbeit war der Vergleich der geometrischen und graphentheoretischen Varianten des Standortproblems k-Center in Verkehrsnetzwerken. Für den Vergleich wurden Datensätze verwendet, die mit einem in Kapitel 2 vorgestellten Verfahren aus der OpenStreetMap gewonnen wurden.
Für beide Varianten wurde in den Kapiteln 3 und 4 jeweils ein Algorithmus vorgestellt, mit dem sich das Problem beliebig genau approximieren lässt. Anschließend wurden in Kapitel 5 sowohl die Laufzeiten der beiden Algorithmen verglichen, als auch die Güte der erhaltenen Zentrenmengen als Lösungen der jeweils anderen k-Center Variante bestimmt.

Fazit: Approximation der graphentheoretischen Lösung

Für die Lösung des graphentheoretischen k-Centers benötigt man eine Matrix mit den kürzesten Weglängen im Straßengraphen. In Kapitel 2 haben wir eine Modifikation des Dijkstra-Algorithmus vorgestellt, um diese aus den OSM-Daten zu gewinnen. Jedoch ist dieses Verfahren sehr aufwändig. In Tabelle 6.1 ist die Größe des Graphen (Anzahl der Knoten, der Kanten, der möglichen Zentren und der Standorte) sowie die Berechnungsdauer und der verbrauchte Speicherplatz (in Megabyte) der TSPLib-Datei[7] für die in Abschnitt 5.2 vorgestellten Datensätze eingetragen. Man sieht, dass diese Berechnung einen großen Anteil an der gesamten Berechnungszeit für das graphentheoretische k-Center einnimmt. Zu beachten ist außerdem, dass alle Datensätze auf kleine Gebiete (Bayern bzw. Österreich) eingeschränkt wurden.

Datensatz	Knoten	Kanten	$\|V\|$	$\|S\|$	Laufzeit	Speicherplatz
Aldis	1.359.270	1.479.310	4.814	224	3:07:49	166,2 MB
Apotheken	1.260.221	1.349.276	11.918	766	6:27:45	1.053,7 MB
Biergärten	1.478.205	1.621.494	10.127	743	7:21:44	772,8 MB
Krankenhäuser	1.359.349	1.479.310	4.767	303	3:01:41	168,3 MB
Hochsitze	2.220.578	2.463.474	6.341	687	7:33:01	319,4 MB
Touristenattraktionen	1.259.843	1.349.276	12.228	376	6:32:36	1.041,3 MB

Tabelle 6.1: Aufwand zur Berechnung der Distanzmatrix.

Verzichtet man auf die aufwändige Berechnung einer Distanzmatrix und löst stattdessen ein euklidisches k-Center um anschließend die Zentren auf die jeweils nächsten Knoten im OSM-Graphen zu runden, so beträgt die durchschnittliche Abweichung 17.5%, kann jedoch bei einzelnen Instanzen auch über 70% erreichen.

[7] TSPLib ist eine Sammlung von Datensätzen für das Travelling Salesman Problem. Darin wird ein vollständiger, gewichteter Graph in einem festgelegten Dateiformat gespeichert. Um die vorhandenen TSPLib-Parser verwenden zu können, speichern wir unsere Distanzmatrix ebenfalls in diesem Format.

Dieses Vorgehen kann also nur bei Anwendungen mit entsprechend niedrigen Genauigkeitsanforderungen angewendet werden. Andernfalls führt kein Weg an einer Abkehr von einer normbasierten Abstandsmodellierung, hin zu einer genaueren Beschreibung der Weglängen als Kantengewichte eines Graphen, vorbei.

Fazit: Approximation der geometrischen Lösung

Der Algorithmus von Brandenberg & Roth weist eine in k exponentielle Laufzeit auf. In Abschnitt 3.2 haben wir dies exemplarisch auf dem Datensatz *Apotheken* durchgerechnet. Während ein 9-Center in einer halben Minute gefunden werden konnte, benötigte man für ein 12-Center schon 20 Minuten und ein 15-Center konnte selbst in 60 Stunden nicht gefunden werden.

Daher haben wir in Abschnitt 5.5 ein Verfahren untersucht, bei dem nicht das geometrische k-Center direkt gelöst, sondern stattdessen die Ebene diskretisiert und aus den Gitterpunkten ein Graph mit Luftlinienentfernungen als Kantengewichten generiert wird. Die aufwändige Berechnung einer Distanzmatrix auf dem OSM-Graphen ist hierbei natürlich nicht nötig.

Auf diesem Graphen kann dann mit dem graphentheoretischen Algorithmus von Ilhan & Pinar ein k-Center berechnet werden. Wir haben gezeigt, dass dabei durch Variation der Gitterweite jede beliebige Approximationsgüte erreicht werden kann.

Bei der Implementierung dieses Verfahrens haben wir festgestellt, dass die Anzahl der Gitterpunkte nicht beliebig erhöht werden kann. Da die zu sortierenden Kantengewichte in einer Liste gespeichert werden und die Anzahl der Einträge der Anzahl der Gitterpunkten mal der Anzahl der Standorte entspricht, wird ab einigen zehntausend Gitterpunkten die Grenze des Arbeitsspeichers erreicht. Daher können dann die ILPs nicht mehr aufgestellt werden. Zusätzlich ist das Verfahren nur für einige hundert Standorte geeignet, da der ILP-Löser empfindlich auf zu viele Nebenbedingungen reagiert.

Aufgrund dessen können nur Approximationsgüten von einigen Prozent erreicht werden. Die zusätzliche Einschränkung, dass die Approximationsgüte nur a posteriori bestimmt werden kann, lässt sich durch ein iteratives Verfahren zur Bestimmung der Gitterweite oder durch eine Abschätzung des geometrischen Radius umgehen.

Mit zunehmendem k nimmt, durch den kleiner werdenden Radius, entweder die Approximationsgüte ab oder die Anzahl der Gitterpunkte muss zunehmen. Für konstante Approximationsgüten lässt sich daher die Zentrenanzahl k nicht beliebig erhöhen.

In Abschnitt 5.5 haben wir gesehen, dass die Abschätzung des Fehlers sehr pessimistisch ausfällt. Es bleibt daher zu untersuchen, ob eine bessere Abschätzung gefunden werden kann, um die Randbedingungen, unter denen ein sinnvoller Einsatz dieses Verfahrens möglich ist, zu erweitern.

Ein Vorteil der geometrischen Variante des Algorithmus von Ilhan & Pinar besteht darin, dass in etwa die Hälfte der Laufzeit für die Sortierung der Kantenliste benötigt wird. Sollen nun zu einer Gitterweite mehrere k-Center gelöst werden, lässt sich die sortierte Kantenliste auch mehrfach verwenden und damit die Laufzeit wesentlich reduzieren.

6.1 Weiterführende Fragestellungen

Zum Schluss wollen wir noch einige interessante Erweiterungen der in dieser Diplomarbeit betrachteten Problemstellung anführen.

Gerichtete Graphen

Um die Distanzmatrix als TSPLib-Datei speichern zu können, wurden Richtungsinformationen der Wege in Einbahnstraßen oder auf Autobahnen ignoriert. Dadurch können sich die tatsächlich zurückzulegenden Wegstrecken zwischen Zentren und Standorten beim graphentheoretischen k-Center verlängern.
Beim Vergleich der beiden Lösungen wurde der graphentheoretische Radius auf demselben Graphen berechnet. Daher sind keine großen Veränderungen durch gerichtete Graphen zu erwarten und dieser Fall wurde nicht näher untersucht. Außerdem setzt der Algorithmus von Ilhan & Pinar in seiner hier dargestellten Form einen ungerichteten Graphen voraus.

Fahrtdauer anstelle der Weglänge

Wie in der Einführung angedeutet, ist oft nicht die Weglänge interessant, sondern die Zeit, die für eine bestimmte Strecke benötigt wird. Um diesem Umstand Rechnung zu tragen, könnte die Maximalgeschwindigkeit auf den Straßen, die ebenfalls in den OSM-Daten gespeichert ist, verwendet werden, um eine bessere Modellierung der Fahrtdauer zu berechnen. Dadurch würden die Kanten in unterschiedlichen Leveln im k-Levelgraphen aus Abschnitt 2.2.2 auch unterschiedlich gewichtet werden.

Äquidistant verteilte zusätzliche Zentren

Die zusätzlichen Zentren wurden in dieser Diplomarbeit auf den Kreuzungspunkten eines Graphen mit hohem Level eingetragen. Die Idee dabei war, dass die Standorte eine möglichst gute Verkehrsanbindung haben sollten.
In Abschnitt 5.5 haben wir gesehen, dass für das geometrische k-Center eine äquidistante Verteilung der zusätzlichen Knoten sinnvoll ist. Es bleibt zu untersuchen, ob eine ähnliche Verteilung der zusätzlichen Knoten im graphentheoretischen k-Center Vorteile gegenüber einer Verteilung auf Straßen hohen Levels bringt.

Andere Standortprobleme

Die Fragestellung, wie sich geometrische und graphentheoretische Lösungen eines Standortproblems unterscheiden, ist nicht auf das k-Center beschränkt. Auch für andere Probleme, wie zum Beispiel das k-Median oder das Facility-Location Problem, existieren geometrische und graphentheoretische Varianten, deren Lösungen unterschiedlich schwierig zu berechnen sind.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Arnold, Andreas: *Approximationsalgorithmen zur Lösung von allgemeinen k-Containment Problemen*, Technische Universität München, Diplomarbeit, 2009

[2] Brandenberg, René: *Computational Convexity Optimale Containment Probleme*, Technische Universität München, Vorlesungsskript, 2009

[3] Brandenberg, René ; Roth, Lucia: New Algorithms for k-Center and Extensions. In: *Journal of Combinatorial Optimization* 18 (2009), Nr. 4, S. 376–392

[4] Fredman, Michael L. ; Tarjan, Robert Endre: Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. In: *Journal of the ACM* 34 (1987), Nr. 3, S. 596–615

[5] Garey, Michael R. ; Johnson, David S.: *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman, 1979. – ISBN 978–0716710455

[6] Goblor grübelt: *OpenStreetMap-Projekt: Teil2 – Von den Rohdaten zum gerichteten Graphen*. `http://goblor.de/wp/2009/10/17/`, 17. Oktober 2009

[7] Gonzalez, Teofilo F.: Clustering to Minimize the Maximum Intercluster Distance. In: *Theoretical Computer Science* 38 (1985), S. 293–306

[8] Gritzmann, Peter: *Optimierung 1,2*, Technische Universität München, Vorlesungsskript, 2009

[9] Gritzmann, Peter ; Klee, Victor: Inner and Outer j-Radii of Convex Bodies in Finite-Dimensional Normed Spaces. In: *Discrete & Computational Geometry* 7 (1992), S. 255–280

[10] Hochbaum, Dorit S. ; Shmoys, David B.: A Unified Approach to Approximation Algorithms for Bottleneck Problems. In: *Journal of the ACM* 33 (1986), Nr. 3, S. 533–550

[11] Hsu, Wen-Lian ; Nemhauser, George L.: Easy and Hard Bottleneck Location Problems. In: *Discrete Applied Mathematics* 1 (1979), Nr. 3, S. 209–215

[12] Ilhan, Taylan ; Pinar, Mustafa Ç.: *An Efficient Exact Algorithm for the Vertex p-Center Problem*. `http://www.optimization-online.org/DB_HTML/2001/09/376.html`, 27. September 2001

[13] Kowoma.de: *Kartenprojektionen*. `http://www.kowoma.de/gps/geo/Projektionen.htm`, 03. Februar 2007

[14] Megiddo, Nimrod: On the Complexity of Some Geometric Problems in Unbounded Dimension. In: *Journal of Symbolic Computation* 10 (1990), Nr. 3-4, S. 327–334

[15] Minieka, Edward: The m-Center Problem. In: *Siam Review* 12 (1970), Nr. 1, S. 138–139

[16] Mladenović, Nenad ; Labbé, Martine ; Hansen, Pierre: Solving the p-Center Problem with Tabu Search and Variable Neighborhood Search. In: *Networks* 42 (2003), Nr. 1, S. 48–64

[17] National Imagery and Mapping Agency: World Geodetic System 1984 / Department of Defence. 2000 (8350.2). – Forschungsbericht. – Third Edition

[18] OpenStreetMap: *OpenStreetMap*. `http://www.openstreetmap.org`, 1. Juni 2010

[19] Russell, Stuart ; Norvig, Peter: *Künstliche Intelligenz – Ein moderner Ansatz*. Pearson Studium, 2004. – ISBN 978–3827370891