Projektarbeit
im Rahmen des Masterstudiengangs
Systemtechnik

Räumliche Komplexitätsreduktion in eTraGo - Entwicklung eines k-medoids Dijkstra Clusterings

vorgelegt von
Katharina Esterl

Eidestattliche Erklärung

Ich versichere, dass ich die vorliegende Thesis ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen benutzt habe.

Ort, Datum

Katharina Esterl

Abstract

The open-source *Electricity Transmission Grid Optimization - Tool* (*eTraGo*) is part of the transparent, inter-grid-level operating grid planning tool *eGo* (*electrical grid optimization*) which at the moment is extended by more flexibilities and sectors. *eTraGo* optimizes grid and storage expansion on the German high and extra-high voltage level for future scenarios and is characterized by a high temporal and spatial resolution. To ensure simulations of high accuracy despite suitable calculation times, approaches for adequate reduction of complexity are needed.

At the moment, a k-means clustering is used to reduce spatial resolution in *eTraGo*. This is a common complexity reduction approach clustering the buses of the original network considering their geographical locations and representing the clusters by aggregated buses at the cluster centers. In some cases, the spatially reduced networks show lines between cluster centers which are not directly connected in the original grid topology. Hence, the original network does not get represented accurate enough in those cases.

Therefore, within this work a new clustering approach gets introduced, implemented and validated which considers the topology of the electrical grid. This approach is called k-medoids Dijkstra Clustering. The methodology is comprised of the following steps:

1. k-means Clusterings considering the geographical locations of the original buses
2. identification of the medoids of the clusters
3. Dijkstra's algortihm assigning the original buses to the clusters with the nearest medoids considering the electrical distance

The examination of the spatially reduced networks' topologies shows that the k-medoids Dijkstra Clustering reduced network does not show the prior explained problems occuring when using a k-means Clustering. Instead, the k-medoids Dijkstra Clustering reduced grid topologies show more equalities to the original grid topology, especially concerning the meshing of the grid.

The examination of the effects on network calculations using the two different complexity reduction approaches does not show relevant differences in the overall results like overall network and storage expansion. It does however point out differences in network expansion requirements in special regions where a lot of the original buses get assigned differently using the k-medoids Dijkstra Clustering than using the k-means Clustering and where the k-means Clustering reduced network therefore show an extra highly meshed grid topology. For those cases an underestimation of the grid expansion requirement in k-means Clustering reduced networks has to be assumed.

Concerning the caclulation time, k-means Clustering and k-medoids Dijkstra Clustering are to grade equally, because the extra time needed to conduct the Dijkstra's algorithms only adds a a fraction to the time needed to conduct a k-means Clustering and can be neglected.

In summary, the k-medoids Dijkstra Clustering is identified as adequate complexity reduction approach for *eTraGo* and as the better alternative compared to the currently used k-means Clustering. Therefore, the future usage of the k-medoids Dijkstra Clustering for spatial complexity reduction in *eTraGo* is recommended.

Inhaltsverzeichnis

Abkürzungsverzeichnis

eTraGo *Electricity Transmission Grid Optimization - Tool*

PyPSA *Python for Power System Analysis - Tool*

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Im Rahmen des Forschungsprojektes *open_eGo* wurde ein transparentes, netzebenenübergreifendes Netzplanungstool entwickelt, mithilfe dessen die Netzentwicklung im Hinblick auf die Energiewende untersucht werden kann. Aktuell werden zur Abbildung der voranschreitenden Elektrifizierung die weiteren Sektoren Gas, Wärme und Mobilität integriert, um Herausforderungen und Chancen der Sektorkopplung zu untersuchen.

eTraGo stellt das Tool zur Optimierung von Netz- und Speicherausbau auf Hoch- und Höchstspannungebene des elektrischen Netzes in Deutschland dar. Das aus *open source*-Daten erstellte Modell weist eine hohe räumliche und zeitliche Auflösung auf. Die hohe Komplexität des Systems hat einen hohen Rechenaufwand für Simulationen zur Folge, sodass Komplexitätsreduktionsmethoden angewandt werden. Ziel dieser Komplexitätsreduktionsmethoden ist es, eine Netzberechnung unter möglichst geringem Informationsverlust bei gleichzeitig akzeptabler Rechenzeit zu ermöglichen.

Zur räumlichen Komplexitätsreduktion dient in *eTraGo* aktuell ein k-means Clustering, das derzeit in mehreren Arbeiten auf elektrische Netze angewandt wird (siehe z. B. [1]). Dabei werden die Originalknoten des Netzes unter Berücksichtigung derer geografischen Positionen in Cluster zusammengefasst und durch repräsentative Knoten ersetzt, sodass die räumliche Auflösung reduziert wird. An einigen Stellen der mithilfe k-means Clustering reduzierten Netze ergeben sich Leitungen zwischen Clustern, die im Originalnetz keine direkte elektrische Verbindung aufweisen. Das Originalnetz wird an diesen Stellen nicht genügend genau dargestellt, sodass die Gefahr der Verfälschung von Simulationsergebnissen besteht.

Aus diesem Grund soll im Rahmen dieser Arbeit eine neue Methodik zur räumlichen Komplexitätsreduktion vorgestellt, implementiert und validiert werden, die die Topologie des elektrischen Netzes bei der Komplexitätsreduktion berücksichtigt. Sie soll im Folgenden als k-medoids Dijkstra Clustering bezeichnet werden. Es soll untersucht werden, ob das k-medoids Dijkstra Clustering eine adequate räumliche Komplexitätsreduktionsmethode zur Anwendung in *eTraGo* und unter Berücksichtigung der Kriterien Genauigkeit und Rechenaufwand eine bessere Alternative zum k-means Clustering darstellt.

Im Folgenden werden zunächst die Methodik des k-means Clusterings sowie die sich durch Anwendung dieser Methodik ergebende Problematik dargestellt (siehe Kapitel 2). In Kapitel 3 wird das k-medoids Dijkstra Clustering sowie dessen Implementierung in *eTraGo* vorgestellt. Es folgt die Festlegung der Methodik zur Untersuchung der Clustermethoden in Kapitel 4 sowie weiterhin die Darstellung der Ergebnisse der durchgeführten Rechnungen in Kapitel 5. Anschließend soll das k-medoids Dijkstra Clustering hinsichtlich seiner Anwendbarkeit in *eTraGo* und im Vergleich zum aktuell verwendeten k-means Clustering bewertet werden (siehe Kapitel 6) bevor in Kapitel 7 ein Fazit aus den Erkenntnissen der Untersuchungen geschlossen wird.

2 Aktuell verwendete Methodik: k-means Clustering

Im Folgenden soll die in *eTraGo* aktuell zur räumlichen Komplexitätsreduktion verwendete Methodik vorgestellt werden. Dazu wird zunächst die prinzipielle Funktionsweise des k-means Clusterings erläutert sowie die Implementierung in *eTraGo* beschrieben. Anschließend wird auf Problematiken, die durch die Verwendung dieses Ansatzes beobachtet werden, hingewiesen. In diesem Zuge wird die Funktionalität *remove_stubs* dargestellt, die in einigen Fällen Abhilfe schafft. Abschließend wird Verbesserungspotential abgeleitet.

2.1 Prinzipielle Funktionsweise und Implementierung in eTraGo

Ziel eines k-means-Algorithmus ist es, einen Datensatz nach bestimmten Kriterien in eine gewissen Anzahl von Gruppen, die sogenannten Cluster, zu unterteilen. So kann dieser Algorithmus dazu genutzt werden, die Knoten eines Stromnetzes in eine vorgegebene Anzahl von Clustern zu unterteilen, um die Cluster durch neue Knoten mit bestimmten Eigenschaften zu repräsentieren. Die Clustereinteilung erfolgt in diesem Fall nach der geografischen Position der Knoten. Die repräsentativen Knoten stellen beim k-means-Algorithmus geografisch die durchschnittlichen Knoten pro Cluster dar. So wird das Stromnetz mit seiner originalen Knotenanzahl durch ein Stromnetz mit einer geringeren Knotenanzahl ersetzt und die räumliche Komplexität dadurch reduziert.

Diese prinzipielle Vorgehensweise wird auch in *eTraGo* verwendet. Die Methodik orientiert sich an der in [2] dargestellten Methodik und nutzt das in *PyPSA* implementierte k-means Clustering. Hier wird der *expectation-maximization*-Algorithmus des Python Pakets *scikit-learn* ([3]) verwendet.

Zunächst werden die Knoten des Originalnetzes proportional zur angeschlossenen konventionellen Erzeugung und zum angeschlossenen Verbrauch im heutigen Energiesystem (*Status Quo*-Szenario) gewichtet. Die Gewichtung wird berücksichtigt, indem die Knoten des Originalnetzes entsprechend ihrer Gewichtung auf ihrer Koordinate vervielfältigt werden, sodass eine Menge aus Punkten entsprechend der Gewichtung auf den geografischen Koordinaten der Originalknoten entsteht. Anschließend kommt der *expectation-maximization*-Algorithmus zum Einsatz, der folgendermaßen arbeitet:

1. Initale Wahl von Clusterzentren im Originalnetz
2. Wiederholung bis zur Erfüllung des Konvergenzkriteriums
    a) Erwartungsschritt: Zuordnung der Originalknoten zu den jeweils geografisch nächsten Clusterzentren
    b) Maximierungsschritt: Neuberechnung der Clusterzentren als geografische Mittelwerte der in einem Cluster befindlichen Knoten

Die Wahl der Clusterzentren erfolgt anhand der folgenden Gleichung 2.1. Hier wird die Summe der quadrierten gewichteten euklidischen Abständen zwischen den Koordinaten der Originalknoten x_n und den Koordinaten der Clusterzentren x_c minimiert.

$$\min_{x_c} \sum_{c=1}^{k} \sum_{n \in N_C} w_n \left\| x_c - x_n \right\|^2 \tag{2.1}$$

mit

- N: Menge der Knoten im Originalnetz
- n: Benennung eines einzelnen Knoten des Originalnetzes
- x_n: geografische Koordinaten des Knotens n im Originalnetzes
- w_n: Gewichtung des Knotens n im Originalnetz
- k: Anzahl der Cluster
- c: Benennung der einzelnen Cluster
- x_c: geografische Koordinaten des Clusterzentrums im Cluster c
- N_c: Menge der Knoten im betrachteten Cluster

Das Ergebnis der beschriebenen Berechnung stellt eine *Busmap* dar, die eine Zuordnung der N Originalknoten zu den k Clustern enthält. Die für die Gewichtung benötigten zusätzlichen Knoten spielen lediglich während der Durchführung des *expectation-maximization*-Algorithmus ein Rolle, in der *Busmap* sind sie nicht zu finden.

Es folgt die Aggregation der Knoten innerhalb eines Clusters, indem die Knoten-Komponenten Erzeugung, Speicher und Last an der geografischen Position des jeweiligen während des *expectation-maximization*-Algorithmus ermittelten Clusterzentrums zusammengefasst werden. Die Leitungen innerhalb eines Clusters werden vernachlässigt, Leitungen zwischen zwei Clustern werden zu abstrakten Leitungen zusammengefasst.

Zu beachten ist, dass durch den beschriebenen Optimierungsalgortihmus nicht gewährleistet ist, dass globale Optima gefunden werden. Es handelt sich bei den Ergebnissen um lokale Optima. Maßgebend für die Güte und die Robustheit der Lösung sind die initiale Wahl der Clusterzentren sowie die Konvergenzkriterien. Diese werden bei Aufruf des Clusterings als Parameter übergeben.
Im Laufe der Entwicklung von *eTraGo* sind einige Testsimulationen durchgeführt worden, die folgende Parameterkombination hat sich dabei als geeignet herausgestellt:

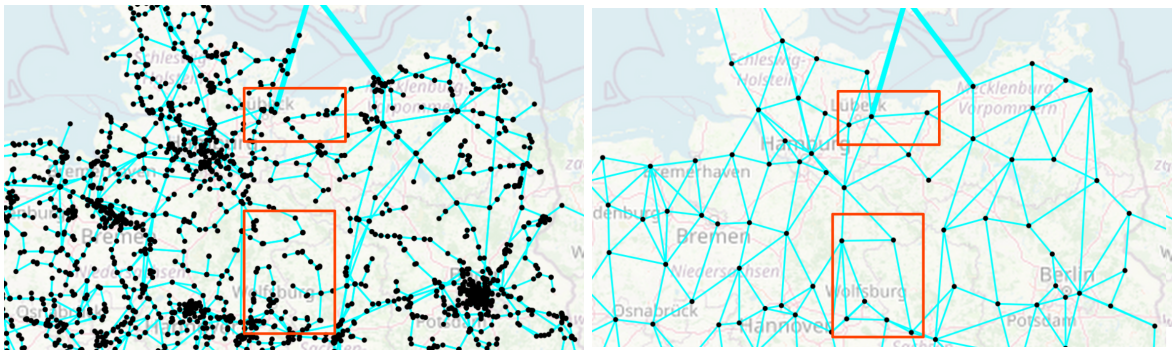
Tabelle 2.1: Wahl der Parameter für k-means Clustering

Parameter	**Wert**	**Bedeutung**
n_init	2500	limitiert die Anzahl der Durchläufe mit verschiedener initialer Wahl der Clusterzentren
max_iter	1000	limitiert die Anzahl der Iterationsschritte je Durchlauf
tol	1e-20	definiert die maximal zulässige Toleranz des Inertias
n_jobs	-1	zur parallelen Berechnung auf n-1 von n verfügbaren Prozessoren

2.2 Verbesserungspotential

Das k-means Clustering berücksichtigt zur Zuordnung der Originalknoten zu verschiedenen Clustern die geografischen Distanzen zwischen den Knoten. Die elektrischen Distanzen zwischen den Knoten und somit die Topologie des Netzes bleiben unberücksichtigt. Dies hat in einigen Fällen zur Folge, dass Knoten einem Clusterzentrum zugeordnet werden, das zwar geografisch nah, aber elektrisch weit entfernt liegt. Dieser Fall tritt insbesondere für Stichleitungen und in einigen Maschen auf. Im Zuge der Aggregation werden dann Leitungen zwischen Clustern gebildet, die eigentlich keine direkte elektrische Verbindung aufweisen. Die Topologie des komplexitätsreduzierten Netzes bildet die Topologie des Originalnetzes in diesen Fällen nur unzureichend ab. Es besteht die Gefahr verfälschter Ergebnisse der Netzberechnung, da der Leistungsfluss verändert wird.

Abbildung 2.1 zeigt einen Ausschnitt des in *eTraGo* modellierten elektrischen Netzes Deutschlands, das das Originalnetz mit 11305 Knoten darstellt, sowie den entsprechenden Ausschnitt eines mithilfe k-means Clusterings auf 300 Knoten räumlich komplexitätsreduzierten Netzes. Die roten Rechtecke markieren zwei Beispiele, für die der beschriebene Fall zutrifft. Das obige Rechteck markiert die Region nahe Lübeck. Das komplexitätsreduzierte Netz enthält eine Querverbindung zwischen zwei Clusterzentren, die im Originalnetz keine direkte Verbindung aufweisen. Diese kommt zustande, weil die Knoten der Spitze der Masche, die von rechts in das Rechteck hineinragt, aufgrund ihrer geografischen Koordinaten dem Clusterzentrum links im Rechteck zugeordnet werden. Ebenso verhält es sich in dem Fall, das das untere rote Rechteck in der Region Wolfsburg markiert. Auch in diesem Beispiel erhält das komplexitätsreduzierte Netz eine Leitung, die die Topologie des Originalnetzes so nicht aufweist.



(a) Originalnetz (b) k-means Clustering reduziertes Netz

Abbildung 2.1: Exemplarische Beispiele für die Problematik des k-means Clusterings

Teilweise kann die als Option integrierte Funktionalität ***remove_stubs*** die beschriebene Problematik verbessern. Bei Anwendung dieser Funktionalität wird das Originalnetz vor der Durchführung des Clusterings um Stichleitungen, die keine weitere Vermaschung aufweisen, reduziert. Die Komponenten der entfernten Knoten werden den angebundenen Knoten zugeordnet. Somit werden die oben beschriebenen Fälle an den wegfallenden Stichleitungen verhindert. Abbildung 2.1 zeigt allerdings, dass die Problematik auch an anderen Stellen auftritt, weshalb diese durch die Funktionalität *remove_stubs* nicht gänzlich behoben werden kann. Außerdem ergeben sich durch Anwendung dieser Funktionalität Einbuße in der Genauigkeit. Während die Anwendung bei der räumlichen Komplexitätsreduktion auf eine relativ kleine Knotenanzahl (z. B. von 11305 Originalknoten auf 100 Knoten) verhältnismäßig genau funktioniert, weil innerhalb der Cluster ohnehin keine Stichleitungen sichtbar sind, stellt die Anwendung dieser Funktionalität insbesondere dann, wenn auf eine relativ hohe Anzahl von Knoten

geclustert wird (z. B. 500 Knoten), eine erhebliche Änderung der Originaltopologie dar, sodass mit Abweichungen der Ergebnisse einer Netzberechnung gerechnet werden muss.

Aus der mangelhaften Darstellung des Originalnetzes an den durch die Beispiele repräsentativ aufgezeigten Stellen sowie aus der Gefahr der Verfälschung der Ergebnisse einer Netzberechnung lässt sich Verbesserungspotential in der räumlichen Komplexitätsreduktion in *eTraGo* ableiten.

3 Methodik des k-medoids Dijkstra Clustering

Der in diesem Kapitel vorgestellte Ansatz berücksichtigt, im Gegensatz zu einem k-means Clustering (Kapitel 2) die Topologie des elektrischen Netzes. Er setzt sich zusammen aus der Kombination eines k-medoids Clustering, das ähnlich wie das k-means Clustering funktioniert, und einem anschließend ausgeführten Dijkstra-Algorithmus, der die Berücksichtigung der elektrischen Distanzen zwischen den Knoten und somit der Netztopologie realisiert.

Im Folgenden soll zunächst die prinzipielle Funktionsweise des Ansatzes erklärt und anschließend die Umsetzung in *eTraGo* dargestellt werden.

3.1 Prinzipielle Funktionsweise

Diese Methodik beinhaltet im ersten Schritt ein k-medoids Clustering, das dem in Kapitel 2 beschriebenen k-means Clustering sehr ähnlich ist. Ziel dieses Arbeitsschrittes ist es, die Knoten des Originalnetzes zunächst nach ihrer geografischen Verteilung in eine vorgegebene Anzahl von Clustern einzuteilen. Die Originalknoten werden dazu wie beim k-means Clustering gewichtet, bevor die Clusterberechnung durchgeführt wird. Im Gegensatz zum k-means Clustering werden die Cluster beim k-medoids Clustering nicht durch den geografisch durchschnittlichen Knoten pro Cluster repräsentiert (daher „mean “für arithmetisches Mittel), sondern durch einen Originalknoten, der in diesem Anwendungsfall den Median des Clusters darstellt. Diesen Punkt nennt man für allgemeine, n-dimensionale Datensätze Medoid des Clusters. Die Cluster werden also durch Knoten repräsentiert, die geografisch unter den Knoten des Originalnetzes zu finden sind und eine minimale geografische Distanz zu den restlichen Knoten innerhalb des Clusters aufweisen.

Im zweiten Schritt wird ein Dijkstra-Algorithmus durchgeführt. Dieser dient allgemein der Berechnung der kürzesten Pfade zwischen einem Startknoten und einem der übrigen oder aller übrigen Knoten in einem Graphen. Folgende Schritte beschreiben den Algorithmus ([4]):

1. Initialisierung
    - alle Knoten erhalten die Attribute „Distanz“und „Vorgänger“
    - Distanz im Startknoten erhält den Wert 0
    - Distanz aller anderen Knoten erhalten den Wert ∞
    - Startknoten wird in Warteschlange eingefügt
2. Wiederholung bis keine Knoten mehr in der Warteschleife
    - Wahl des Knotens aus der Warteschlange mit der geringsten Distanz zum Startknoten
    - Speicherung des Besuchs des ausgewählten Knotens
    - einzelne Überprüfung der Nachbarknoten des ausgewählten Knotens auf folgende Fälle:

a) Wenn der Nachbarknoten bereits in der Warteschleife vermerkt ist und die Distanz zur Erreichung über die neue Kante geringer ist als die bisherige Distanz, wird die neue Distanz gespeichert.

b) Falls a) nicht zutrifft: Wenn der Knoten noch gar nicht besucht worden ist, wird er der Warteschleife hinzugefügt und die Distanz wird auf die Summe der Distanzen des Vorgängerknotens und der Kante zu dem betreffenden Knoten aktualisiert.

Nach Durchführung des k-medoids Clusterings liegen eine Aufteilung des Originalnetzes in eine vorgegebene Anzahl von Clustern sowie die Positionen der repräsentativen Knoten der einzelnen Cluster vor, die Positionen von Originalknoten darstellen. Mithilfe des Dijkstra-Algorithmus soll die Zuordnung der Knoten des Originalnetzes zu den einzelnen Clustern, die bisher auf Grundlage der geografischen Verteilung erfolgt ist, auf Grundlage der elektrischen Distanzen zwischen den Knoten überprüft und angepasst werden. Dazu werden die Pfade von allen Knoten des Originalnetzes zu den einzelnen Clustermedoiden berechnet. Die Zuordnung der Originalknoten zu den durch das k-medoids Clustering identifizierten Clustermedoiden erfolgt letztlich anhand der durch den Dijkstra-Algorithmus ermittelten elektrischen Pfaden, indem die einzelnen Originalknoten jeweils demjenigen Clustermedoid zugeordnet werden, das zur Erreichung den kürzesten elektrischen Pfad aufweist.

Nach Identifizierung der Position der neuen repräsentativen Knoten mithilfe des k-medoids Clusterings und der Zuordnung der Originalknoten zu diesen neuen repäsentativen Knoten mithilfe des Dijkstra-Algorithmus, erfolgt die Aggregation der Komponenten. Die Attribute der repräsentativen Knoten werden wie in Kapitel 2 beschrieben ermittelt, die geografische Position des repräsentativen Knotens wird auf den während des k-medoids Clusterings identifizierten Medoid des Clusters festgelegt.

Folgend sind die Arbeitsschritte des k-medoids Dijkstra Clustering zusammengefasst:

Tabelle 3.1: Übersicht über Arbeitsschritte des k-medoids Dijkstra Clusterings

Arbeitsschritt	**Beschreibung**
k-means Clustering	
1. Gewichtung der Knoten	Gewichtung der Knoten nach Erzeugung und Last im *Status Quo*-Szenario durch Vervielfältigung der Punkte
2. Durchführung des Clusterings	Durchführung des *expectation-maximization*-Algorithmus zum Clustern der Punkte
3. Erstellung der *Busmap*	Zuordnung der Originalknoten zu den Clustern
Identifikation der Medoide	Identifikation derjenigen Originalknoten, die den während des k-means Clusterings idendifizierten geografisch durchschnittlichen Clusterzentren geografisch am nächsten sind, als Medoide der Cluster
Durchführung des Dijkstra-Algorithmus	
1. Dijkstra-Algortihmus	Berechnung der elektrischen Pfade der Originalknoten zu den Clustermedoiden
2. Erstellung der angepassten *Busmap*	Zuordnung der Originalknoten zu denjenigen Clustern, deren Medoide elektrisch am nächsten sind
Aggregation	Zusammenfassung der Originalknoten pro Cluster sowie der Leitungen zwischen den Clustern

3.2 Implementierung in eTraGo

Aufgrund von Schwierigkeiten mit der Kompatibilität der vorhandenen Datensätze und der entsprechenden Methoden wurde davon abgesehen, das k-medoids Clustering mithilfe bereits existierender Methoden (beispielsweise im *scikit-learn-extra*-Paket) zu implementieren. Stattdessen wird das k-means Clustering, wie es in Kapitel 2 beschrieben ist, verwendet und so abgeändert, dass das Ergebnis dem eines k-medoids Clusterings entspricht. Dazu wird zunächst das k-means Clustering unter Berücksichtigung der Gewichtung nach Erzeugung und Last im *Status Quo-szenario* durchgeführt, um die Originalknoten auf Basis ihrer geografischen Verteilung in ein vorgegebene Anzahl von Clustern zu unterteilen und die Clusterzentren als geografisch durchschnittliche Knoten zu identifizieren. Die hierfür verwendete Klasse des Python Pakets *scikit-learn* ([3]) liefert außerdem eine Methode zur Berechnung einer Matrix, die die Distanzen aller Originalknoten zu den Clusterzentren enthält. Mithilfe dieser Matrix können diejenigen Originalknoten identifiziert werden, die den geografisch durchschnittlichen Knoten pro Cluster geografisch am nächsten sind. Diese Knoten stellen die Medoide der einzelnen Cluster und deren Koordinaten die Koordinaten der neuen repräsentativen Knoten dar.

Die Implementierung des Dijkstra-Algorithmus erfolgt mithilfe des Python Pakets *NetworkX* ([5]), das innerhalb der Algorithmen *Shortest Paths* mit der Funktion *single_source_dijkstra_path_length* eine vorgefertigte Funktion zur Durchführung des Algorithmus bereitstellt. Mithilfe dieser Funktion werden jeweils die elektrischen Pfade zwischen den im k-medoids Clustering identifizierten Clustermedoiden und allen weiteren Originalknoten berechnet. Die Zuordnung der Originalknoten zu den neuen, repräsentativen Knoten erfolgt dann auf Basis der errechneten elektrischen Distanz.

Aufgrund der Komplexität dieser Rechenaufgabe wird das Python Paket *multiprocessing* verwendet, ein Modul der *The Python Standard Library*. Unter Verwendung dieses Moduls kann die Berechnung in Abhängigkeit der verfügbaren Kerne parallelisiert werden. Außerdem besteht die Möglichkeit der Nutzung des Parameters *Cutoff*, den die für den Dijkstra-Algortihmus verwendete Methode enthält. Dieser Parameter bewirkt den Abbruch der weiteren Berechnung eines bestimmten Pfades ab einer vorgegebenen Länge, wodurch sich der Rechenaufwand verringert. Mehrere Testsimulationen haben ergeben, dass ein Wert sinnvoll ist, der in etwa der zwei- bis vierfachen Distanz desjenigen Originalknotens zu seinem Clusterzentrum im k-means Clustering entspricht, der am weitesten von seinem Clusterzentrum entfernt ist. Der Parameter wird als Faktor dieser maximalen Distanz angegeben, zum Beispiel: *Cutoff = 3*. Die Wahl des *Cutoff*-Parameters verändert das Ergebnis des k-medoids Dijkstra Clusterings nicht. Wenn er sehr groß gewählt wird, wird der Rechenaufwand entsprechend weniger reduziert. Wird er zu klein gewählt, kann keine Zuordnung durch den Dijkstra-Algorithmus erfolgen. Für diesen Fall ist eine Fehlermeldung implementiert, die den/die Nutzer*in darauf hinweist.

Im Anhang A.1 sind Ausschnitte der relevanten Passagen der Implementierung zu finden.

4 Methodik zur Untersuchung

Ziel der räumlichen Komplexitätsreduktion ist es, das Originalnetz auf ein räumlich komplexitätsreduziertes Netz abzubilden, sodass eine Netzberechnung bei möglichst hoher Genauigkeit und gleichzeitig unter akzeptablem Rechen- und Zeitaufwand möglich ist. Daraus ergeben sich die folgenden Kriterien zur Bewertung: **Genauigkeit** und **Rechenaufwand**.

Im Rahmen dieser Arbeit sollen insbesondere **k-means Clustering** und **k-medoids Dijkstra Clustering** verglichen werden, da sich das k-means Clustering als Standardanwendung zur räumlichen Komplexitätsredutkion in *eTraGo* etabliert hat (siehe z. B. [6]). Da das k-medoids Dijkstra Clustering im ersten Schritt ein k-means Clustering beinhaltet, dem nach der Findung der Medoide der Cluster eine potentielle Korrektur der Zuordnung zu den Clustern unter Anwendung des Dijkstra-Algorithmus folgt, ist ein Vergleich der komplexitätsreduzierten Netze sowie der Clusterbildung sehr gut möglich.

Das **k-means Clustering unter Verwendung von *remove_stubs*** wird weniger häufig genutzt. Der Grund dafür liegt in den Fehlerpotentialen, die diese Funktionalität insbesondere beim Clustern auf hohe Knotenzahlen mit sich bringt. Wie in Kapitel 2.2 beschrieben wirkt diese Funktionalität der auftretenden Problematik außerdem nur bedingt entgegen. Aus diesem Grund soll sich der Vergleich zu der Anwendung der Funktionalität *remove_stubs* auf ein einziges exemplarisches Beispiel beschränken, eine detaillierte Untersuchung und Bewertung wird im Rahmen dieser Arbeit nicht durchgeführt.

4.1 Untersuchung der Genauigkeit

Grundsätzlich soll die Untersuchung der Genauigkeit mithilfe zweier Rechenszenarien durchgeführt werden.

4.1.1 Rechenszenario zur Untersuchung der Netztopologien

Zunächst sollen die Ansätze anhand der Topologien der komplexitätsreduzierten Netze untersucht werden. Bei dieser Analyse geht es darum, wo und wie häufig sich die Cluster durch Verwendung des k-medoids Dijkstra Clustering verändern und welche Auswirkungen die Veränderungen auf die Topologie des komplexitätsreduzierten Netzes hat. Insbesondere soll dabei ein Augenmerk auf die in Kapitel 2.2 beschriebene Problematik gelegt werden, wozu exemplarische Beispiele betrachtet werden. Im Rahmen dieser Untersuchung wird das Originalnetz mithilfe der drei verschiedenen Clustermethoden **k-means Clustering**, **k-means Clustering unter Verwendung der Funktionalität remove_stubs** sowie **k-medoids Dijkstra Clustering**, in seiner Komplexität reduziert. Das Originalnetz, das 11305 Knoten enthält, wird jeweils auf 100, 200, 300, 400 und 500 Knoten geclustert. Die Untersuchung der Topologie des komplexitätsreduzierten Netzes sowie der Cluster wird mithilfe eines auf 300 Knoten reduzierten Netzes durchgeführt, da sich diese Einstellung in Veröffentlichungen als Standardfall

etabliert hat (siehe z. B. [6]). Zur Überprüfung der Gültigkeit der Beobachtungen für weitere Clusterzahlen werden die Häufigkeit der Wirkung des Dijkstra-Algorithmus zur Abänderung der Zuordnung im Vergleich zum k-means Clustering sowie der Vermaschungsgrad der komplexitätsreduzierten Netze in Abhängigkeit der Clusterzahlen untersucht.

Der Vermaschungsgrad ist im Rahmen dieser Untersuchung als Quotient aus Anzahl der Leitungen im Netz pro Knoten im Netz definiert. Damit gibt dieser Quotient nicht für jeden Fall den Grad der Vermaschung eines Netzes an. So kann beispielsweise auch ein Strahlennetz ohne Vermaschung einen Wert von größer als 1 aufweisen. Jedoch handelt es sich bei dem hier betrachteten Netzmodell um das elektrische Übertragungsnetz in Deutschland, das zu großen Teilen von Maschen durchsetzt ist. Auch die Topologien der komplexitätsreduzierten Netze weisen Vermaschung auf. Aus diesem Grund ist davon auszugehen, dass der Quotient aus Leitungen pro Knotenzahl in etwa den Vermaschungsgrad der hier betrachteten Netze widergibt und einen probaten Wert zum Vergleich der Netze darstellt.

Zur Durchführung des k-means Clusterings werden die in Kapitel 2 festgehaltenen Parameter gewählt. Um eine allumfassende Untersuchung zu ermöglichen, kommt der optionale Parameter *Cutoff* (siehe Kapitel 3) im Rahmen dieser Rechnung nicht zum Einsatz.

4.1.2 Rechenszenario zur Untersuchung der Auswirkungen auf die Netzberechnung

Mithilfe dieses Szenarios sollen die Auswirkungen der beobachteten Veränderungen auf die Netzberechnung untersucht werden. Im Rahmen dieser Untersuchung werden insbesondere die Auswirkungen auf den Leitungs- und Speicherausbau analysiert. Es handelt sich hierbei um allgemein relevante Ergebnisse einer Netzberechnung. Da aufgrund der verschiedenen Topologien der komplexitätsreduzierten Netze unterschiedliche Leitungsstrukturen vorliegen, sind die Ergebnisse der Leitungssauslastung, von denen die genannten Ergebnisse maßgeblich abhängen, im Rahmen dieser Betrachtung besonders interessant.

Im Zuge dieser Untersuchung wird das Originalnetz jeweils unter Anwendung des **k-means Clusterings** und des **k-medoids Dijkstra Clusterings** auf 300 Knoten geclustert, da es sich hierbei den in Veröffentlichungen etablierten Standardfall handelt (siehe z. B. [6]. Theoretisch bedarf es zur Untersuchung auf Genauigkeit dem Vergleich mit dem Originalnetz. Dieser ist für die hier beschriebene Analysestrategie nicht möglich, da die Komplexität der Berechnung die vorhandenen Rechenkapazitäten übersteigt. Nichtsdestotrotz lassen die ausgewählten Untersuchungsaspekte einen Vergleich zwischen den vorgestellten Methodiken zu. Es lassen sich hinreichend gute Schlüsse auf die Genauigkeit ziehen.

Zur Durchführung des k-means Clustering werden auch in diesem Rechenszenario die in Kaitel 2 festgehaltenen Parameter gewählt, der optionale Parameter *Cutoff* zur Durchführung des Dijkstra-Algortihmus kommt nicht zum Einsatz. Es wird das Szenario *eGon 100* mit den Optionen Netz- und Speicherausbau für das gesamte Jahr in jedem fünften Zeitschritt gerechnet.

Im Anhang A.7 ist ein Screenshot der verwendeten *eTraGo*-Argumente zu finden, der die Parameter des Rechenszenarios zusammenfasst.

4.2 Untersuchung der Rechenzeit

Zur Untersuchung des Rechenaufwandes soll die Rechenzeit, die zur Durchführung der räumlichen Komplexitätsreduktion benötigt wird, herangezogen werden. Während der Entwicklung sowie Validierung der neuen Clustermethode sowie im Zuge der Durchführung der bereits beschriebenen Rechnungen wurden die Rechenzeiten dokumentiert. Im Rahmen dieser Untersuchung werden die Rechenzeiten des k-means Clusterings sowie des k-medoids Dijkstra Clusterings in Abhängigkeit der Knotenzahlen gegenübergestellt.

Zur Durchführung des k-means Clusterings werden die in Kapitel 2 festgehaltenen Parameter gewählt. Der optionale Parameter *Cutoff* (siehe Kapitel 3) kommt nicht zum Einsatz. Das *multiprocessing* hingegen wird vollumfänglich genutzt.

5 Darstellung und Diskussion der Ergebnisse

Folgend werden die Ergebnisse nach Durchführung der in Kapitel 4 beschriebenen Rechenszenarien zur Untersuchung der Clustermethoden dargestellt und diskutiert.

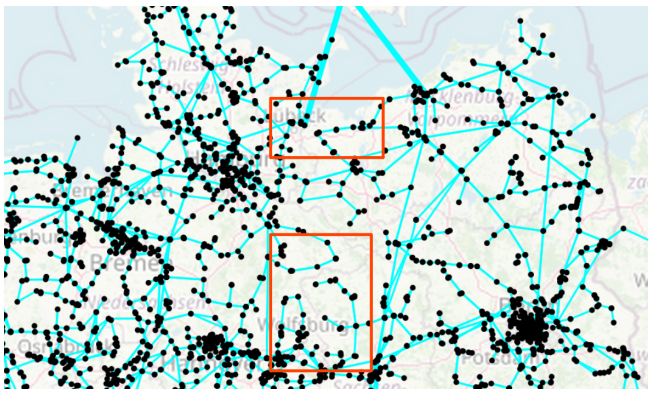
5.1 Untersuchung der Netztopologien

Die Untersuchung der Netztopologien teilt sich in vier Abschnitte. Zur Überprüfung der Wirksamkeit des k-medoids Dijkstra Clusterings soll zunächst das exemplarische Beispiel betrachtet werden, das in Kapitel 2.2 zur Verdeutlichung der Problematik des k-means Clusterings herangezogen worden ist. Anschließend soll ein kurzer Einblick in die Verwendung der Funktionalität *remove_stubs* gegeben werden. Es folgt eine Gegenüberstellung der Topologien der komplexitätsreduzierten Netze, bevor zuletzt ein Vergleich der Bildung der Cluster folgt.

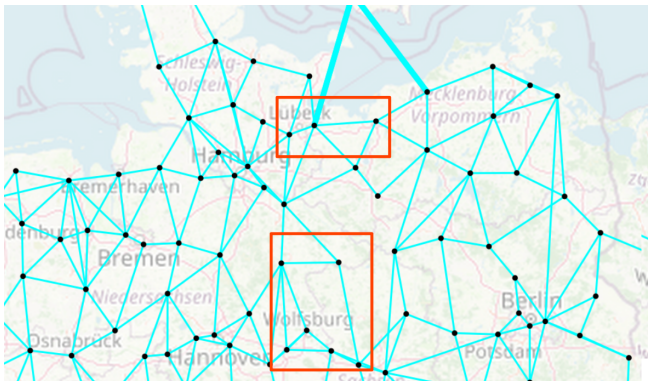
5.1.1 Betrachtung eines exemplarischen Beispiels

Abbildung 5.1 zeigt oben einen Ausschnitt des in *eTraGo* modellierten elektrischen Netzes Deutschlands, das das Originalnetz mit 11305 Knoten darstellt. Mittig befindet sich die Abbildung des entsprechenden Ausschnitt eines mithilfe k-means Clusterings auf 300 Knoten räumlich komplexitätsreduzierten Netzes, unten der entsprechende Ausschnitt eines ebenso mithilfe eines k-medoids Dijkstra Clustering reduzierten Netzes. Die roten Rechtecke markieren zwei Beispiele, für die die in Kapitel 2.2 beschrieben Problematik zutrifft.

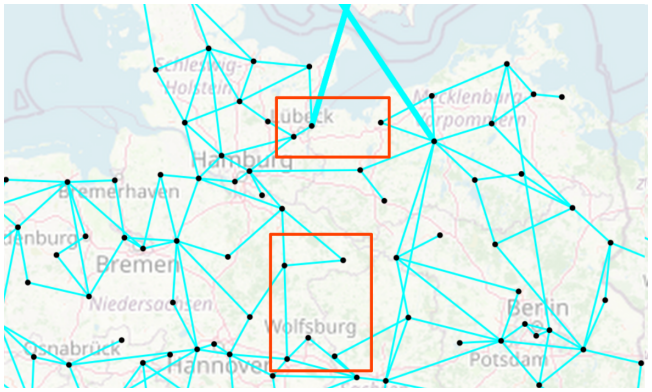
In beiden Fällen tritt die durch das k-means Clustering auftretende problematische Verbindung zwischen Clusterpunkten, die im Originalnetz keine direkte elektrische Verbindung aufweisen, im k-medoids Dijkstra Clustering geclusterten Netz nicht auf. Die Knoten in der Spitze der Masche, die von rechts in das obere Rechteck hineinragt, sowie die Knoten Leitung, die von unten rechts in das untere Rechteck ragt, werden im Falle des k-medoids Dijkstra Clusterings aufgrund ihrer elektrischen Distanzen nicht den Knoten der jeweils anderen Maschen zugeordnet, wie es aufgrund der geografischen Distanzen beim k-means Clustering der Fall ist. Damit wird im Zuge der Aggregation keine Verbindung zwischen den repräsentativen Knoten erstellt. Das k-medoids Dijkstra geclusterte Netz liegt an diesen Stellen somit näher an der Topologie des Originalnetzes.



(a) Originalnetz



(b) k-means Clustering reduziertes Netz



(c) k-medoids Dijkstra Clustering reduziertes Netz

Abbildung 5.1: Exemplarische Beispiele zur Überprüfung der Wirksamkeit des k-medoids Dijkstra Clusterings

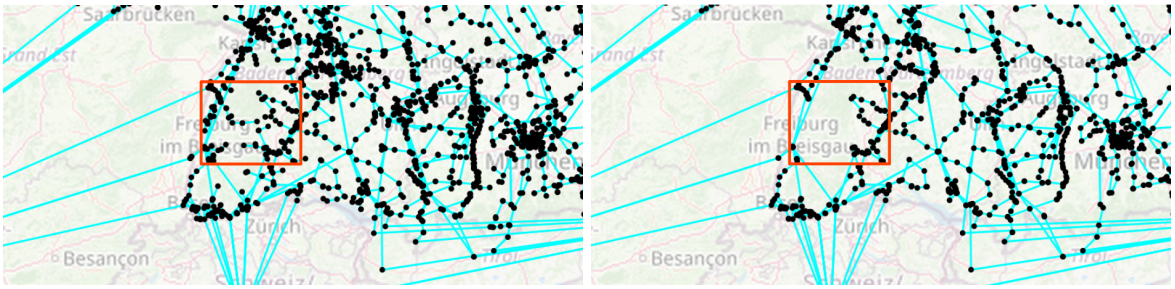
5.1.2 Betrachtung der Funktionalität remove_stubs

Die optionale Funktionalität *remove_stubs* verhindert die in 2.2 erläuterte Problematik in den Fällen, in denen Stichleitungen ohne weitere Vermaschung geografisch nah an anderen Knoten liegen. Dies ist nicht der Fall für die im vorhergehenden Abschnitt betrachteten Beispiele, weil in beiden Fällen Maschen vorliegen, die durch die Funktionalität nicht verändert werden.

Um die Wirkung dieser Funktionalität zu untersuchen und mit der Wirkung des k-medoids Dijkstra Clusterings zu vergleichen, soll das folgende Beispiel betrachtet werden. Abbildung 5.2 zeigt links einen Ausschnitt des betrachteten Originalnetzes, rechts ist das um die Stichleitungen reduzierte Netz zu

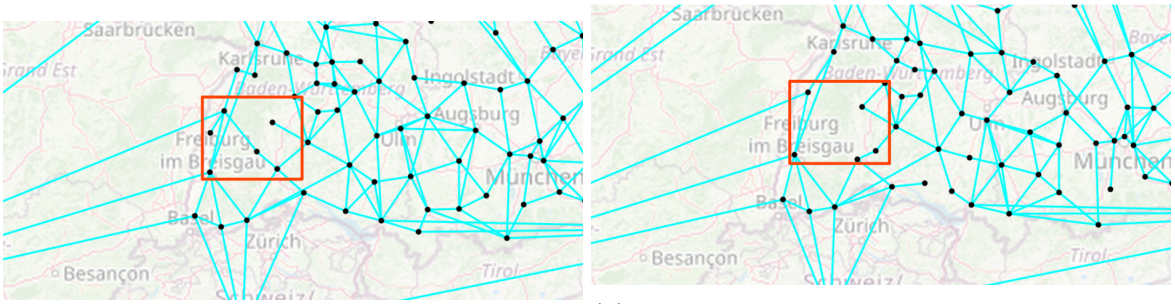
finden. Das rote Rechteck markiert einen Bereich des Netzes, in das besonders viele Stichleitungen hineinragen, sodass die Knoten hier geografisch nah beieinander liegen. Die Funktionalität *remove_stubs* entfernt diese Stichleitungen.

Abbildung 5.3 zeigt die auf 300 Knoten reduzierten Netze nach Anwendung eines k-means Clusterings auf das Originalnetz (oben links), nach Anwendung des k-means Clustering auf das *remove_stubs* reduzierte Netz (oben rechts) sowie nach Anwendung des k-medoids Dijkstra Clusterings auf das Originalnetz (unten). Wie prognostiziert weist das k-means geclusterte Netz in dem markierten Bereich eine Leitung quer durch den markierten Bereich auf, die das durch *remove_stubs* mit anschließendem k-means Clustering reduzierte Netz nicht aufweist. Die Funktionalität wirkt der in 2.2 beschriebenen Problematik in diesem Fall entgegen, das mithilfe *remove_stubs* komplexitätsreduzierte Netz bildet das Originalnetz an dieser Stelle damit genauer ab als das k-means geclusterte Netz ohne Anwendung der Funktionalität *remove_stubs*. Im k-medoids Dijkstra geclusterten Netz liegt die entsprechende Leitung ebenso nicht vor, dieses komplexitätsreduzierte Netze bildet das Originalnetz an dieser Stelle ebenso gut ab.

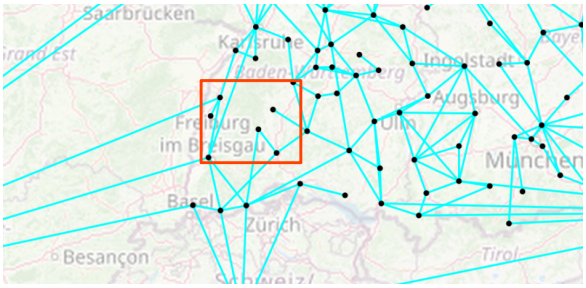


(a) Originalnetz (b) *remove_stubs* reduziertes Netz

Abbildung 5.2: Exemplarisches Beispiel zur Überprüfung der Wirksamkeit der Funktionalität *remove_stubs*



(a) k-means Clustering reduziertes Netz (b) *remove_stubs* und k-means Clustering reduziertes Netz

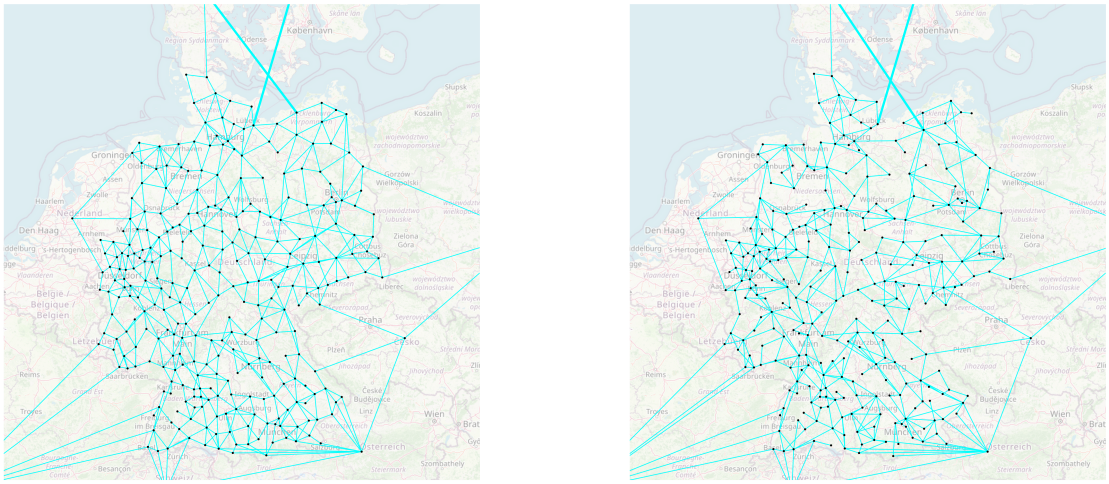


(c) k-medoids Dijkstra Clustering reduziertes Netz

Abbildung 5.3: Exemplarisches Beispiel zur Gegenüberstellung der Wirkung des k-means Clusterings mit der Funktionalität *remove_stubs* und des k-medoids Dijkstra Clusterings

5.1.3 Betrachtung der Topologien der komplexitätsreduzierten Netze

Abbildung 5.4 zeigt die Topologien des auf 300 Knoten geclusterten Originalnetzes. Links ist das k-means geclusterte Netz zu sehen, rechts ist das mithilfe des k-medoids Dijkstra Clusterings reduzierte Netz gegenübergestellt.



(a) k-means Clustering reduziertes Netz (b) k-medoids Dijkstra Clustering reduziertes Netz

Abbildung 5.4: Gegenüberstellung der Topologien der komplexitätsreduzierten Netze

Das k-means Clustering reduzierte Netz weist eine vergleichsweise gleichmäßigere und insgesamt höhere Vermaschung auf als es für das k-medoids Dijkstra geclusterte Netz der Fall ist. Dieses enthält einige „Leerstellen“ ohne Leitungen, insbesondere fällt eine Schneise mit wenigen West-Ost-Leitungen entlang der ehemaligen innerdeutschen Grenze auf. Dies bildet das originale Netz ab, das aufgrund der Historie Deutschlands genau dieses Merkmal aufweist. Weiterhin sind in dem k-medoids Dijkstra geclusterten Netz Stichleitungen zu finden, was für das k-mean Clustering reduzierte Netz nicht der Fall ist. Auch diese lassen sich im Originalnetz finden.

Der folgende Graph 5.5 zeigt den in Kapitel 4.1.1 eingeführten Vermaschungsgrad als Leitungen pro Knoten für die komplexitätsreduzierten Netze bei verschiedenen Knotenzahlen sowie für das Originalnetz. Die Beobachtung eines höheren Vermaschungsgrades der k-means geclusterten Netze bestätigt sich für die dargestellten weiteren Knotenzahlen. Der Vermaschungsgrad liegt für das k-medoids Dijkstra Clustering stets näher an dem Vermaschungsgrad des Originalnetzes, wobei sich dieser für beide Clustermethoden mit größerer Knotenanzahl dem Vermaschungsgrad des Originalnetzes annähert. Die Differenz zwischen den Vermaschungsgraden des k-means Clusterings und des k-medoids Dijkstra Clusterings nimmt mit steigender Knotenzahl geringfügig zu. Bei einer Knotenzahl von 500 Knoten ist der Vermaschungsgrad des k-medoids Dijkstra geclusterten Netzes dem des Originalnetzes nahezu gleich. Die Ursache für die höhere Vermaschung der k-means Clustering reduzierten Netze liegt vermutlich in der in 2.2 beschriebenen Problematik, die Leitungen zwischen Clusterknoten zur Folge hat, die im Originalnetz keine direkte Verbindung aufweisen. Das Netz enthält damit „zusätzliche“ Leitungen, die sowohl das Originalnetz als auch das k-medoids Dijkstra geclusterte Netz nicht aufweisen.

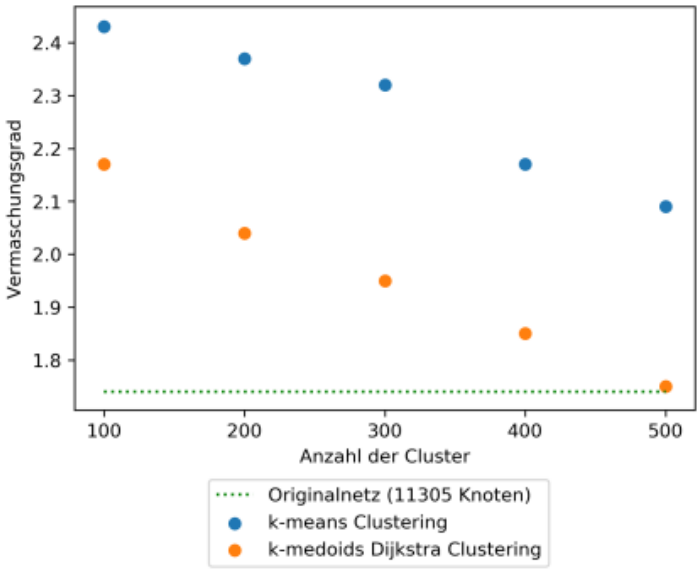
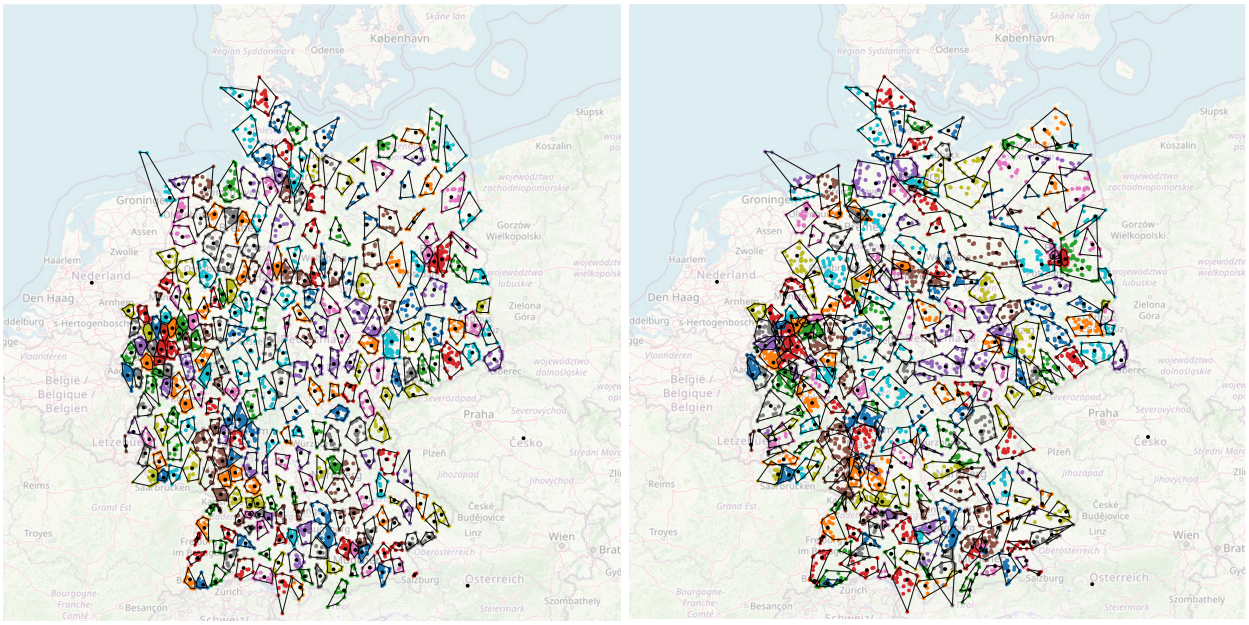


Abbildung 5.5: Darstellung des Vermaschungsgrades des Originalnetzes und der komplexitätsreduzierten Netze bei verschiedenen Knotenzahlen

5.1.4 Betrachtung der Clusterbildung

Abbildung 5.6 zeigt jeweils alle Knoten des Originalnetzes eingefärbt nach deren Zuordnung zu 300 Clustern. Die verschiedenen Cluster sind zusätzlich durch schwarze Umrandungen gekennzeichnet, außerdem ist der repräsentative Knoten pro Cluster schwarz abgebildet. Abbildung 5.6 a) zeigt die k-means Cluster, in Abbildung 5.6 b) sind die k-medoids Dijkstra Cluster dargestellt. Es ist deutlich zu sehen, dass die k-means Cluster verschieden große, einzelne und nicht überlappende Gruppierungen darstellen. Die k-medoids Dijkstra Cluster hingegen sind verschieden große, sich überlappende und ineinander übergehende Gruppierungen. Diese Beobachtung gibt die Funktionsweise der beiden verschiedenen Ansätze wider: Das k-means Clustering berücksichtigt die geografischen Distanzen, sodass die Knoten eines Clusters nebeneinander liegen. Die Größen der verschiedenen Cluster ist durch die Gewichtung bestimmt. Die k-medoids Dijkstra Cluster überlappen einander, da die Zuordnung zu Clusterpunkten durch die elektrischen Distanzen bestimmt wird, die, wie in diesem Bild deutlich wird, nicht immer der Zuordnung mithilfe der geografischen Distanzen entspricht.

Abbildung 5.7 zeigt ein weiteres Bild der Knoten des Originalnetzes. Wie in Abbildung 5.6 b) werden die 300 k-medoids Dijkstra Cluster durch schwarze Umrandungen markiert. In dieser Abbildung sind im Gegensatz zu Abbildung 5.6 nur die Knoten farbig, deren Zuordnung nach dem k-medoids Clustering durch Überprüfung deren elektrischer Distanz zu den Medoiden mithilfe des Dijkstra-Algortihmus abgeändert ist. Das bedeutet, dass alle Knoten, die im Zuge des k-means Clusterings und des k-medoids Dijkstra Clusterings dem gleichen Cluster zugeordnet sind, grau sind. Alle Knoten, die im Zuge der beiden Verfahren verschiedenen Clustern zugeordnet werden, sind entsprechend der k-medoids Dijkstra Cluster eingefärbt. Über das gesamte Netz verteilt sind Knoten zu identifizieren, die aufgrund des Dijkstra-Algortihmus im k-medoids Dijkstra Clustering anders als im k-means Clustering zugeordnet werden. Insgesamt ist ein Großteil der bestehenden Cluster durch eine veränderte Zuordnung betroffen. Meistens handelt es sich um mehrere Knoten pro Cluster, auf die der Dijkstra-Algorithmus wirkt.



(a) k-means Clustering

(b) k-medoids Dijkstra Clustering

Abbildung 5.6: Gegenüberstellung der Bildung der Cluster

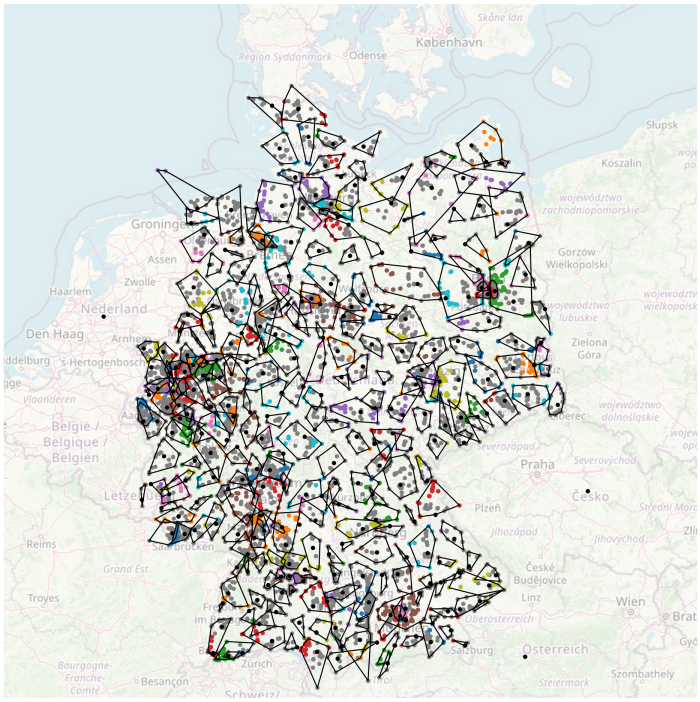


Abbildung 5.7: Markierung der durch Dijkstra-Algortihmus veränderten Zuordnungen

Diese Beobachtungen bestätigen sich mit Blick auf die in Abbildung 5.8 dargestellte Grafik auch für die weiteren untersuchten Clusterzahlen. Die obere Kurve zeigt, dass über alle betrachteten Clusteranzahlen hinweg ein Großteil aller Cluster Knoten enthält, die aufgrund des Dijkstra-Algorithmus zugeordnet werden, wobei der Anteil mit steigender Clusterzahl leicht abnimmt. Der Anteil der Originalknoten, die aufgrund des Dijkstra-Algortihmus im Gegensatz zum k-means Clustering verändert zugeordnet werden, liegt für alle Clusterzahlen bei einem Wert zwischen 20 und 30 Prozent. Daraus ergibt sich, dass auch der Anteil der Knoten pro Cluster, die verändert zugeordnet werden, in einem ähnlichen Bereich liegen.

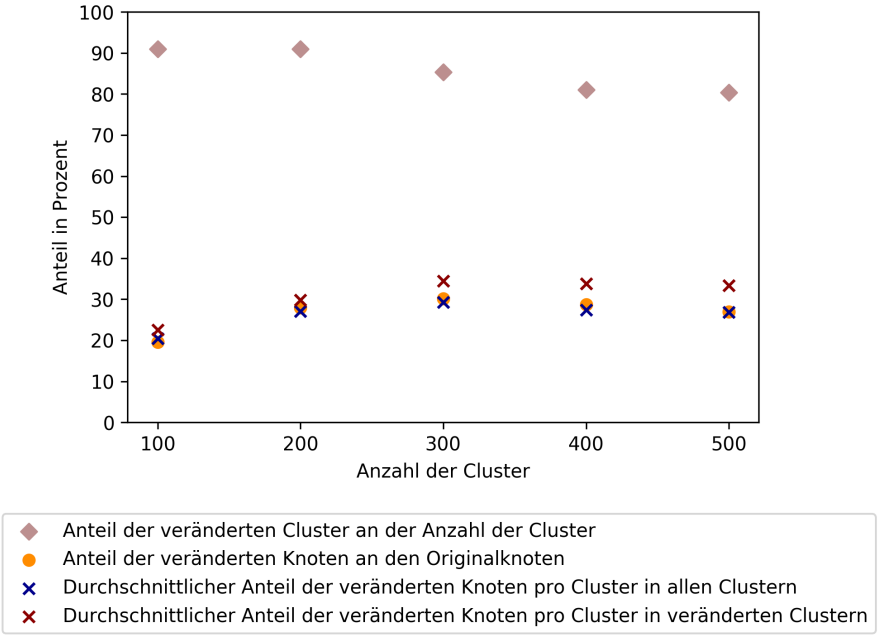


Abbildung 5.8: Darstellung des Anteils der durch Dijkstra-Algortihmus veränderten Zuordnungen für verschiedene Clusterzahlen

5.2 Untersuchung der Auswirkungen auf die Netzberechnung

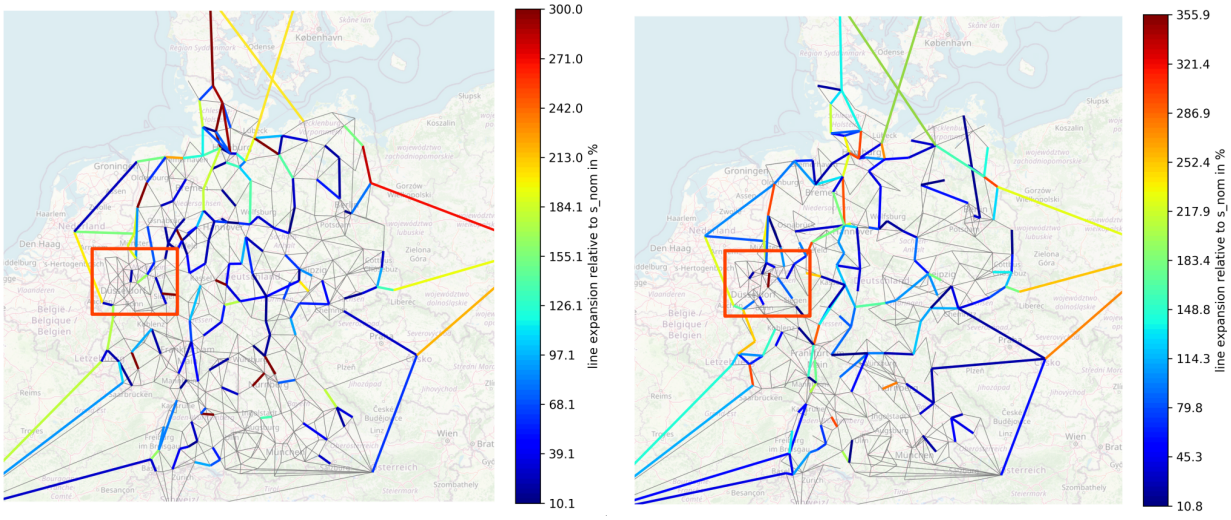
Die Berechnung des in Kapitel 4.2 beschriebenen Szenarios hat für die komplexitätsreduzierten Netze die in Tabelle 5.1 aufgeführten Kosten zum Ergebnis.

Tabelle 5.1: Gegenüberstellung der Ausbaukosten nach Netzberechnung der mithilfe k-means Clustering und k-medoids Dijkstra Clustering komplexitätsreduzierten Netze

	Netzausbau	**Speicherausbau**
k-means Clustering	1,30 Mrd. €	0,02 Mrd. €
k-medoids Dijkstra Clustering	1,28 Mrd. €	0,00 Mrd. €

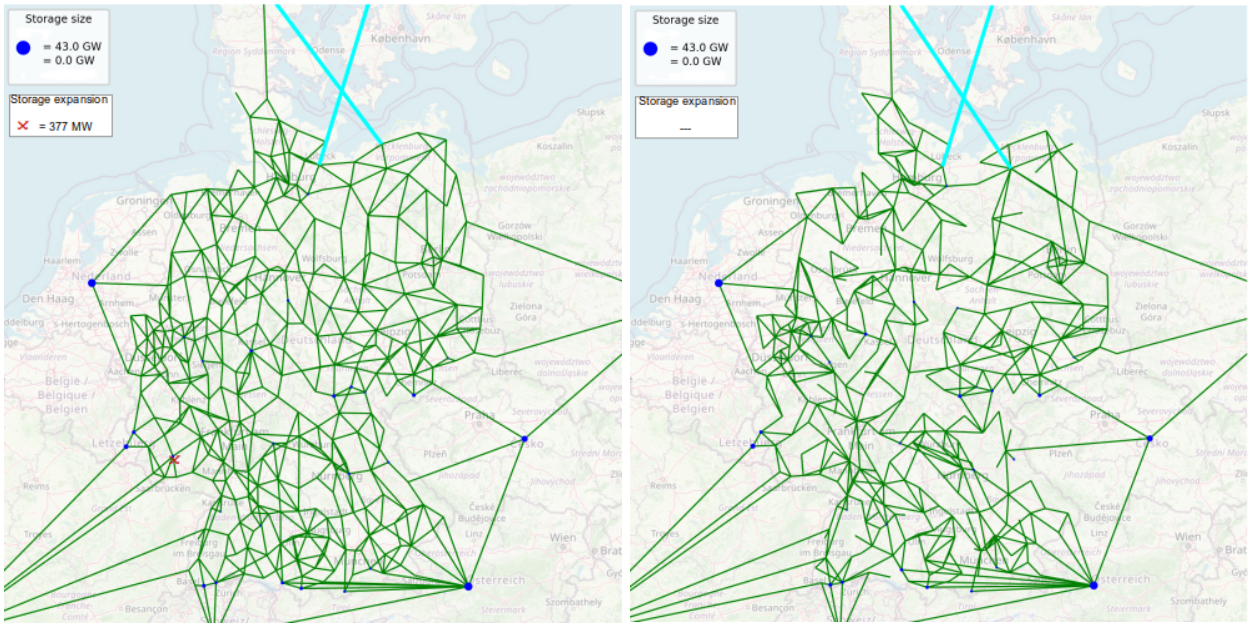
Die Kosten für den Netzausbau liegen im Fall des k-means geclusterten Netzes geringfügig über den Kosten, die für das k-medoids Dijkstra geclusterte Netz angegeben werden. Der Eindruck, dass der Netzausbaubedarf in beiden Netzen etwa gleich stark ausgeprägt ist, bestätigt sich mit Blick auf Abbildung 5.9. So liegen die Leitungsausbauten für beide Netze in einem ähnlichen Bereich relativ zur Nennleistung der Leitungen. Auch die Anzahl der Leitungen, die ausgebaut werden, scheint etwa

vergleichbar hoch zu sein. Nichtsdestotrotz unterscheiden sich der Ausbau einzelner Leitungen in einigen Bereichen deutlich. So ist beispielsweise in dem markierten Raum bei Düsseldorf im k-medoids Dijkstra geclusterten Netz mehr Netzausbau zu verorten als es für das k-means geclusterte Netz der Fall ist. An dieser Stelle sind im Originalnetz besonders viele Knoten pro Fläche zu finden, von denen einige aufgrund der verschiedenen Clustermethoden verschieden zugeordnet werden (siehe Abbildung 5.7). Die Topologie des k-means geclusterten Netzes weist in diesem Bereich eine deutlich höhere Vermaschung auf als das k-medoids Dijkstra geclusterte Netz. Das kann die Ursache dafür sein, dass das k-means Clustering reduzierte Netz an dieser Stelle weniger Netzausbaubedarf aufweist.



(a) k-means Clustering (b) k-medoids Dijkstra Clustering

Abbildung 5.9: Gegenüberstellung des Leitungsausbaus



(a) k-means Clustering (b) k-medoids Dijkstra Clustering

Abbildung 5.10: Gegenüberstellung der Speicherverteilung sowie des Speicherausbaus

Abbildung 5.10 zeigt die Verteilung der Speicher sowie des Speicherausbaus. Wie auch in Tabelle 5.1 zu sehen ist, bedarf es nur im k-means geclusterten Netz des Speicherausbaus. Dieser begrenzt sich auf einen Punkt im Netz. Die Kosten für den Speicherausbau sind im Vergleich zu den Kosten des Leitungsausbaus aus diesem Grund für beide Netze nahezu vernachlässigbar.

5.3 Betrachtung der Rechenzeit

Abbildung 5.11 stellt die zur Durchführung des k-means Clusterings sowie des Dijkstra-Algortihmus benötigte Rechenzeit in Abhängigkeit der Clusterzahl dar. Die Rechenzeit des k-medoids Dijkstra Clustering in Abhängigkeit der Clustezahl ergibt sich jeweils aus der Summe der dargestellten Rechenzeiten, da die aufgeführten Prozesse die maßgeblichen Schritte darstellen. Es wird deutlich, dass die Rechenzeit zur Durchführung des Dijkstra-Algortihmus nur einen Bruchteil zur Gesamtrechenzeit des k-medoids Dijkstra Clusterings beiträgt. Der Großteil der Rechenzeit wird zur Durchführung des k-means Clusterings benötigt. Die Rechenzeiten zur Durchführung des k-means Clusterings und des k-medoids Dijkstra Clusterings sind damit ungefähr gleich groß.

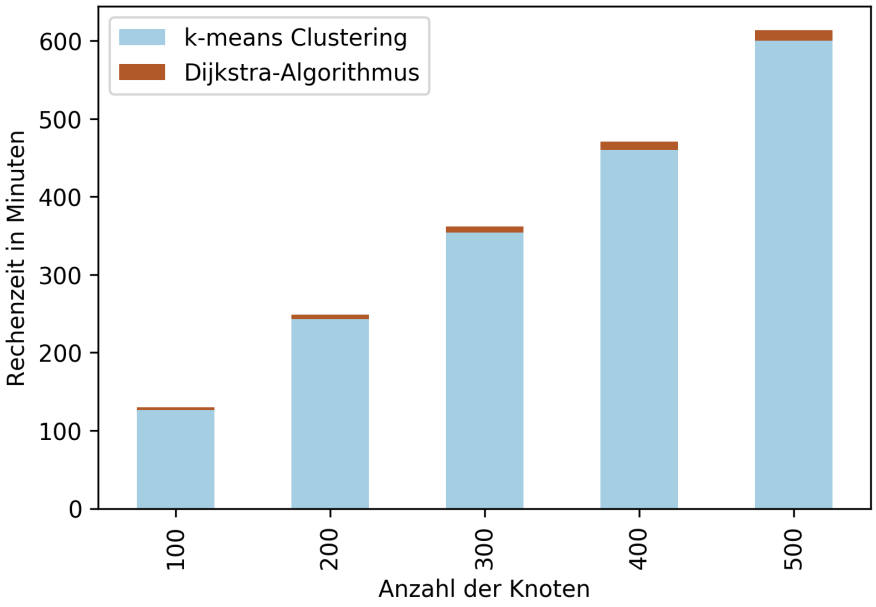


Abbildung 5.11: Darstellung der Rechenzeiten zur Durchführung des k-means Clusterings sowie des Dijkstra-Algorithmus in Abhängigkeit der Clusteranzahl

6 Bewertung

Aus den in Kapitel 5 dargestellten Ergebnissen gilt zu bewerten, ob das k-medoids Dijkstra Clustering eine adequate räumliche Komplexitätsreduktionsmethode zur Anwendung in *eTraGo* und unter Berücksichtigung der Kriterien Genauigkeit und Rechenaufwand eine bessere Alternative zum k-means Clustering darstellt.

Nach Untersuchung der Wirksamkeit des k-medoids Dijkstra Clusterings hinsichtlich der bei der Verwendung des k-means Clusterings auftretenden Problematik, bleibt festzuhalten, dass die mithilfe k-medoids Dijkstra Clustering reduzierten Netze diese Ungenauigkeit nicht aufweisen und das Originalnetz an diesen Stellen genauer abbilden. Die Topologie des mithilfe des k-medoids Dijkstra Clusterings reduzierten Netzes weist an vielen Stellen mehr Ähnlichkeit zum Originalnetz auf als das k-means geclusterte Netz. Dies betrifft insbesondere den nach Kapitel 4.1.1 eingeführten Vermaschungsgrad, der im k-means geclusterten Netz deutlich höher liegt als im Originalnetz und im k-medoids Dijkstra geclusterten Netz besser abgebildet wird.

Hinsichtlich der Auswirkungen auf die Netzberechnung sind bei Betrachtung der Gesamtergebnisse kaum Unterschiede zwischen den komplexitätsreduzierten Netzen auszumachen. Allerdings weisen einige Bereiche der komplexitätsreduzierten Netze angesichts des Netzausbaubedarfs durchaus Unterschiede auf. Es ist davon auszugehen, dass die besonders hohe Vermaschung des k-means Clustering reduzierten Netzes dazu führt, dass der Bedarf an Netzausbau in einigen Regionen unterschätzt wird. Das k-medoids Dijkstra Clustering reduzierte Netz bildet die Vermaschung des Netzes insbesondere an diesen Stellen genauer ab, sodass davon auszugehen ist, dass auch der Netzausbaubedarf genauer berechnet werden kann.

Hinsichtlich der Rechenzeiten der Komplexitätsreduktionsmethoden sind k-means Clustering und k-medoids Dijkstra Clustering in der im Rahmen dieser Arbeit durchgeführten Untersuchung gleichwertig einzustufen, da der für den Dijkstra-Algorithmus benötigte Zeitaufwand den Zeitaufwand zur Durchführung des k-means Clusterings um einen vernachlässigbaren Anteil erhöht.

Insgesamt ist das k-medoids Dijkstra Clustering als geeignet zur Anwendung für die räumliche Komplexitätsreduktion in *eTraGo* einzuordnen und als bessere Alternative im Vergleich zum aktuell verwendeten k-means Clustering zu bewerten.

7 Fazit und Ausblick

Das k-medoids Dijkstra Clustering lässt sich als adequate Methodik zur Durchführung einer räumlichen Komplexitätsreduktion in *eTraGo* einordnen und als bessere Alternative im Vergleich zum k-means Clustering bewerten.

Zwischen den Gesamtergebnissen der Netzberechung der komplexitätsreduzierten Netze lassen sich nur geringfügig Unterschiede ausmachen, sodass davon ausgegangen werde kann, dass die bisher mithilfe des k-means Clusterings erzeugten Ergebnisse als hinreichend genau einzuordnen sind. Jedoch ergeben sich insbesondere im Hinblick auf bestimmte Bereiche des Netzes aufgrund der unterschiedlichen Vermaschung Unterschiede in den Ergebnissen der Netzberechnung, sodass insbesondere bei Betrachtung bestimmter Regionen mit Ungenauigkeiten gerechnet werden muss.

Aus diesem Grund ist die zukünftige Verwendung des k-medoids Dijkstra Clusterings als Standardmethodik zur räumlichen Komplexitätsreduktion in *eTraGo* empfehlenswert.

Literatur

[1] *PyPSA-Eur-Sec: A Sector-Coupled Open Optimisation Model of the European Energy System*. online, accessed 22-June-2021. URL: https://github.com/PyPSA/pypsa-eur.

[2] Jonas Hörsch und Tom Brown. "The role of spatial scale in joint optimisations of generation and transmission for European highly renewable scenarios". In: *2017 14th International Conference on the European Energy Market (EEM)*. 2017. DOI: 10.1109/EEM.2017.7982024. URL: https://ieeexplore.ieee.org/document/7982024/.

[3] F. Pedregosa u. a. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[4] Lisa Velden. *Der Dijkstra-Algorithmus*. online, accessed 22-Feb-2021. 2014. URL: https://www-m9.ma.tum.de/graph-algorithms/spp-dijkstra/index_de.html.

[5] Aric A. Hagberg, Daniel A. Schult und Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Hrsg. von Gaël Varoquaux, Travis Vaught und Jarrod Millman. Pasadena, CA USA, 2008, S. 11–15.

[6] Ulf Philipp Müller u. a. "Integrated Techno-Economic Power System Planning of Transmission and Distribution Grids". In: *Energies* 12.11 (2019). ISSN: 1996-1073. DOI: 10.3390/en12112091. URL: https://www.mdpi.com/1996-1073/12/11/2091.

A Anhang

A.1 Implementierung des k-medoids Dijkstra Clusterings

```
# set bus weightings
bus_weightings=pd.Series(weight)
buses_i=network.buses.index
points = (network.buses.loc[buses_i, ["x","y"]].values
              .repeat(bus_weightings.reindex(buses_i).astype(int), axis=0))

from importlib.util import find_spec
if find_spec('sklearn') is None:
    raise ModuleNotFoundError("sklearn not found")
from sklearn.cluster import KMeans

# optional load of cluster coordinates
if load_cluster != False:
    busmap_array = np.loadtxt(load_cluster)
    kmeans = KMeans(init=busmap_array, n_clusters=n_clusters, \
                n_init=n_init, max_iter=max_iter, tol=tol, n_jobs=n_jobs)
    kmeans.fit(points)

# assignment of points to clusters based on geographical distance
else:
    kmeans = KMeans(init='k-means++', n_clusters=n_clusters, \
                n_init=n_init, max_iter=max_iter, tol=tol, n_jobs=n_jobs)
    kmeans.fit(points)
np.savetxt("cluster_coord_k_%i_result" % (n_clusters), kmeans.cluster_centers_)
print("Inertia of k-means = ", kmeans.inertia_)

# creation of busmap
busmap_kmean = pd.Series(data=kmeans.predict(network.buses.loc[buses_i, ["x", "y"]]),
                     index=buses_i, dtype=object)
```

(a) k-means Clustering mit gewichteten Knoten unter Berücksichtigung der geografischen Distanzen

```
# distances of data points to cluster centers
distances = pd.DataFrame(data=kmeans.transform(network.buses.loc[buses_i, ["x", "y"]].values),
                         index=buses_i, dtype=object)

# get distances between original buses to their mean-buses
dist_mean = pd.Series(data=np.NaN)
for i in range(0,len(distances)):
    index=int(distances.index[i])
    dist_mean[index]=distances.iloc[i].min()
dist_mean.dropna(inplace=True)
dist_mean = 90 * dist_mean # rounded factor to transform distance out of EPSG:4326 to distance in km

# get closest point to each cluster center as new medoid
medoid_idx = pd.Series(data=np.zeros(shape=n_clusters, dtype=int))
for i in range(0, n_clusters):
    index=int(distances[i].idxmin())
    medoid_idx[i]=index
```

(b) Identifikation der Clustermedoide

```
# original data
o_buses = network.buses.index
# k-medoids centers
medoid_idx=medoid_idx.astype('str')
c_buses = medoid_idx.tolist()

# k-medoids assignment
df_kmedoid=pd.DataFrame({'medoid_labels':busmap_kmean.values}, index=busmap_kmean.index)
df_kmedoid['medoid_indices']=df_kmedoid['medoid_labels']
for index, row in df_kmedoid.iterrows():
    label = int(row['medoid_labels'])
    df_kmedoid['medoid_indices'].loc[index] = c_buses[label]

# list of all possible pathways
ppathss = list(product(o_buses, c_buses))

# graph creation
lines = network.lines
edges = [(row.bus0, row.bus1, row.length, ix) for ix, row
         in lines.iterrows()]
M = graph_from_edges(edges)

# cutoff to reduce complexity of Dijkstra's algorithm
cutoff = None

# processor count
cpu_cores = mp.cpu_count()
```

(c) Vorbereitung des Dijkstra-Algortihmus: Erstellung des Graphen, Identifikation der relevanten Pfade, Vorbereitung des *multiprocessing*

```
# calculation of shortest path between original points and k-medoids centers
# using multiprocessing
p = mp.Pool(cpu_cores)
chunksize = ceil(len(ppathss) / cpu_cores)
container = p.starmap(shortest_path, gen(ppathss, chunksize, M, cutoff=cutoff))
df = pd.concat(container)
dump(df, open('df.p', 'wb'))
df.sortlevel(inplace=True)
```

(d) Aufruf der Funktion zur Durchführung des Dijkstra-Algortihmus

```
def shortest_path(paths, graph, cutoff):
    """ Finds the minimum path lengths between node pairs defined in paths.

    Parameters
    ----------
    paths : list
        List of pairs containing a source and a target node

    graph : :class:`networkx.classes.multigraph.MultiGraph
        Graph representation of an electrical grid.

    cutoff : int
        reduction of complexity pf calculation by stopping calculation of paths by this length

    Returns
    -------
    df : pd.DataFrame
        DataFrame holding source and target node and the minimum path length.
    """

    idxnames = ['source', 'target']
    idx = pd.MultiIndex.from_tuples(paths, names=idxnames)
    df = pd.DataFrame(index=idx, columns=['path_length'])
    df.sort_index(inplace=True)

    df_isna = df.isnull()
    for s, t in paths:
        while (df_isna.loc[(s, t), 'path_length'] == True):
            try:
                s_to_other = nx.single_source_dijkstra_path_length(graph, s, cutoff=cutoff)
                for t in idx.levels[1]:
                    if t in s_to_other:
                        df.loc[(s, t), 'path_length'] = s_to_other[t]
                    else:
                        df.loc[(s,t),'path_length'] = np.inf
            except NetworkXNoPath:
                continue
            df_isna = df.isnull()

    return df
```

(e) Funktion zur Durchführung des Dijkstra-Algortihmus

```
# assignment of data points to closest k-medoids centers
mask = df.groupby(level='source')['path_length'].idxmin()
df_dijkstra = df.loc[mask, :]
df_dijkstra.reset_index(inplace=True)

# delete double entries in df due to multiprocessing
duplicated=df_dijkstra.duplicated()
for i in range(len(duplicated)):
    if duplicated[i]==True:
        df_dijkstra = df_dijkstra.drop([i])
df_dijkstra.index=df_dijkstra['source']
```

(f) Identifikation der kürzesten Pfade

```
# creation of new busmap with final assignment (format: medoids indices)
busmap_ind=pd.Series(df_dijkstra['target'], dtype=object).rename("final_assignment", inplace=True)
busmap_ind.index=df_dijkstra['source']
```

(g) Erstellung der neuen Busmap

Abbildung A.6: Ausschnitte der relevanten Abschnitte der Implementierung des k-medoids Dijkstra Clusterings

A.2 eTraGo-Einstellungen für Rechenszenarien

```
args = {
    # Setup and Configuration:
    'db': 'oedb',  # database session
    'gridversion': 'v0.4.6',  # None for model_draft or Version number
    'method': 'lopf',  # lopf or pf
    'pf_post_lopf': False,  # perform a pf after a lopf simulation
    'start_snapshot': 1,
    'end_snapshot': 8760,
    'solver': 'gurobi',  # glpk, cplex or gurobi
    'solver_options': {'BarConvTol': 1.e-5, 'FeasibilityTol': 1.e-5, 'method':2, 'crossover':0,
                       'logFile': 'Testcase22/solver.log'},  # {} for default options
    'model_formulation': 'kirchhoff', # angles or kirchhoff
    'scn_name': 'eGo 100',  # a scenario: Status Quo, NEP 2035, eGo 100
    # Scenario variations:
    'scn_extension': None,  # None or array of extension scenarios
    'scn_decommissioning': None,  # None or decommissioning scenario
    # Export options:
    'lpfile': False,  # save pyomo's lp file: False or /path/tofolder
    'csv_export': 'Testcase22/results',  # save results as csv: False or /path/tofolder
    'db_export': False,  # export the results back to the oedb
    # Settings:
    'extendable': ['network','storage'],  # Array of components to optimize
    'generator_noise': 789456,  # apply generator noise, False or seed number
    'minimize_loading': False,
    'ramp_limits': False,  # Choose if using ramp limit of generators
    'extra_functionality': {},  # Choose function name or {}
    # Clustering:
    'network_clustering_kmeans': False,  # False or the value k for clustering
    'network_clustering_kmedoidDijkstra': 300, # False or the value k for clustering
    'load_cluster': False,  # False or predefined busmap for k-means
    'network_clustering_ehv': False,  # clustering of HV buses to EHV buses.
    'disaggregation': None,  # None, 'mini' or 'uniform'
    'snapshot_clustering': False,  # False or the number of 'periods'
    # Simplifications:
    'parallelisation': False,  # run snapshots parallely.
    'skip_snapshots': 5,
    'line_grouping': False,  # group lines parallel lines
    'branch_capacity_factor': {'HV': 0.5, 'eHV': 0.7},  # p.u. branch derating
    'load_shedding': False,  # meet the demand at value of loss load cost
    'foreign_lines': {'carrier': 'AC', 'capacity': 'osmTGmod'},
    'comments': None}
```

Abbildung A.7: Screenshot der *etrago*-Argumente zur Durchführung des Rechenszenarios zur Untersuchung der Auswirkungen auf die Netzberechnung