Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institut für Informatik
Lehrstuhl VII
Robotik und Telematik
Prof. Dr. Klaus Schilling

Clusteralgorithmen für Multisatellitensysteme

Diplomarbeit im Fach Informatik
vorgelegt von

Thomas Niebler

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institut für Informatik
Lehrstuhl VII
Robotik und Telematik
Prof. Dr. Klaus Schilling

Clusteralgorithmen für Multisatellitensysteme

Diplomarbeit im Fach Informatik
vorgelegt von

Thomas Niebler
geboren am 21.02.1987 in Bayreuth

Angefertigt am
Lehrstuhl für Robotik und Telematik
Bayerische Julius–Maximilians–Universität Würzburg

Betreuer:
Prof. Dr. K. Schilling
Dr. rer. nat. M. Schmidt

Abgabe der Arbeit:
30.09.2011

Erklärung

Ich versichere, die vorliegende Diplomarbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur angefertigt zu haben.

Würzburg, den 30.09.2011

(Thomas Niebler)

Zusammenfassung

Auf dem Gebiet der Satellitentechnik verschiebt sich der Trend von der Entwicklung einzelner multifunktionaler und insbesondere großer Satelliten zum Entwurf mehrerer kleiner Satelliten, die einzelne Aufgaben übernehmen und miteinander kommunizieren. Der Einsatz mehrerer solcher Kleinsatelliten ermöglicht es außerdem, Aufgaben zu erfüllen, die mit dem Einsatz großer Satelliten vorher nicht möglich waren, sei es wegen technischer oder finanzieller Restriktionen. Durch die Beschränkung der Leistungsfähigkeit der einzelnen Satelliten ist es allerdings auch erforderlich, eine funktionierende Kommunikationsstruktur aufzubauen, um die Handlungen der einzelnen Satelliten untereinander abzustimmen und so eine Leistungsfähigkeit zur Verfügung stellen zu können, die mit einzelnen, größeren Satelliten nicht möglich wäre. Bei kleinen Satellitennetzwerken, die zudem oft in einer Formation oder Konstellation fliegen, ist diese Aufgabe noch berechenbar. Mit einer steigenden Zahl an Netzwerkmitgliedern, die darüber hinaus keine festen relativen Positionen einhalten, gestaltet sich diese Aufgabe weitaus komplizierter. Durch die große Anzahl an Kommunikationslinks können außerdem Interferenzen in der Kommunikation entstehen, wodurch das Netzwerk ganz empfindlich gestört werden kann. Als Beispiel für ein solches Ad-Hoc-Netzwerk kann das QB50-Projekt angesehen werden, das plant, 50 Double Cubesats auf einen gemeinsamen Orbit zu schicken, um In-Situ-Messungen in der Thermosphäre durchführen zu können. In diesem Projekt verteilen sich die einzelnen Satelliten während der Projektlaufzeit von 3 Monaten über den gesamten Orbit und bauen so häufig Kommunikationslinks auf und unterbrechen diese ebenso häufig wieder. Um solche Links zu koordinieren, wird in dieser Arbeit der Ansatz der Aufteilung in Cluster betrachtet und untersucht, wie man diesen auf Satellitennetzwerke anwenden kann. Hierzu werden verschiedene Algorithmen vorgestellt und einige ausgewählte in mehreren Simulationen getestet. Die Ergebnisse werden interpretiert und vorgestellt.

Danksagung

Ich möchte mich hiermit bei allen Leuten bedanken, die zu einem erfolgreichen Abschluss dieser Arbeit beigetragen haben, allen voran meiner Freundin Julia Kwasny sowie meinen Eltern, die mich durch viel Verständnis und Geduld über die gesamte Arbeitszeit unterstützt haben. Ein besonderer Dank gilt meinem Diplomarbeitsbetreuer Dr. Marco Schmidt, der mich während dieser außergewöhnlichen und interessanten Phase meines Studiums immer bestens beraten und betreut hat. Ein ebenso großer Dank richtet sich an Professor Klaus Schilling für das interessante Thema sowie viele aufschlussreiche Denkanstöße.

Inhaltsverzeichnis

Kapitel 1.

Einleitung

1.1. Motivation

Durch den schnellen Fortschritt in der Satellitenforschung ist es möglich geworden, durch Einsatz von Pico- und Nanosatelliten (wie beispielsweise die Experimentalsatelliten der Universität Würzburg UWE-1 und UWE-2) große Mengen an Kleinstsatelliten gleichzeitig in einen Orbit zu bringen. Ein prominentes Projekt, das eine Realisierung dieser Idee anstrebt, ist das QB50-Projekt, in dem 50 Pico- und Nanosatelliten in die Umlaufbahn gebracht werden sollen. Pico- und Nanosatelliten sind jedoch nur wenige Kubikdezimeter groß und deshalb in ihrer Leistungs- und Manövrierfähigkeit sehr beschränkt. Außerdem zieht die große Stückzahl eine entsprechend hohe Anzahl an Kommunikationslinks nach sich, die koordiniert werden müssen und es kann vorkommen, dass bei einer entsprechend hohen Zahl von Links Interferenzen entstehen, wodurch Übertragungen mit Fehlern behaftet werden.

Ein weiteres Problem stellt zudem die hohe Mobilität von Satelliten im Low Earth Orbit, kurz LEO, bei der Kommunikation mit Bodenstationen dar. Durch die niedrige Umlaufbahn umrunden solche Satelliten die Erde in etwa 15 mal pro Tag und haben somit eine eher kurze Kontaktzeit zu einer Bodenstation. Sind darüber hinaus alle Satelliten auf nicht sehr ähnlichen Orbits, entstehen und verschwinden Kommunikationslinks zwischen Satelliten sehr schnell, da sich durch verschiedene Umlaufgeschwindigkeiten die Nachbarschaft eines Satelliten oft verändert.

Für dieses Kommunikationsproblem gibt es mehrere Ansätze: Es ist zum einen möglich, allen Satelliten durch Scheduling eine Kommunikationszeit einzuräumen. Dies ist allerdings rechen- sowie timingtechnisch sehr aufwändig. Eine weitere Möglichkeit besteht darin, über das bestehende Satellitennetzwerk direkt zu routen, woran im Moment intensiv geforscht wird.

1.2. Aufgabenstellung

In der vorliegenden Arbeit soll untersucht werden, wie solche Kommunikationsprobleme gelöst werden können, wenn man das Satellitennetzwerk in Organisationseinheiten aufteilt, die *Cluster* heißen sollen. Diese Cluster partitionieren das Satellitennetzwerk in verschiedene Untergruppen, die dann ein eigenes kleines Netzwerk bilden. Um solche Cluster zu bilden, existieren verschiedene Algorithmen, die mit unterschiedlichen Kriterien bei der Clustererstellung arbeiten.

Um den Einsatz von Clusteralgorithmen auf Satellitennetzwerken zu testen, sollen mehrere ausgewählte Algorithmen miteinander verglichen werden. Zu diesem Zweck müssen diese Algorithmen implementiert und simuliert werden. Es müssen also diverse Vergleichskriterien aufgestellt sowie einige Simulationsszenarien definiert werden, sodass alle Simulationen auf einer gemeinsamen Grundlage durchgeführt werden.

1.3. Überblick über die Arbeit

In Kapitel 2 werden einige Grundlagen, die zum Verständnis der Diplomarbeit hilfreich sind, näher erklärt und mit Beispielen veranschaulicht. Kapitel 3 beschäftigt sich mit dem aktuellen Stand der Forschung auf dem Gebiet der Satellitennetzwerke. Dort werden außerdem verschiedene Clusteralgorithmen für mobile Ad-Hoc Netzwerks (MANETs) vorgestellt und diskutiert. Um die durchgeführten Simulationen auf ein theoretisches Fundament zu stellen, werden die Randbedingungen für die Simulationsszenarios in Kapitel 4 definiert und deren Motivation erläutert. Eine Erläuterung der Implementierung eines Basisframeworks sowie eine Übersicht der Anpassungen der zu simulierenden Algorithmen findet sich in Kapitel 5. Die Resultate sowie kurze Analysen der Simulationen sind schließlich in Kapitel 6 zu sehen und werden anschließend in Kapitel 7 diskutiert. Kapitel 8 fasst die Arbeit kurz zusammen und gibt einige Ausblicke auf weitere mögliche Forschungsarbeit.

Kapitel 2.

Theoretische Grundlagen

Bevor wir einige Clusteringalgorithmen betrachten, sollten zunächst notwendige theoretische Grundlagen erklärt und verdeutlicht werden. Dazu wird in Abschnitt 2.1 definiert, was mobile Ad-Hoc-Netzwerke sind und was diese auszeichnet. Anschließend werden in Abschnitt 2.2 einige Überlegungen darüber angestellt, wie man solche Netzwerke als ungerichtete Graphen darstellt. In Abschnitt 2.3 wird auf den Begriff des Clusterings eingegangen und schließlich in Abschnitt 2.4 das Keplersche Satellitenmodell erklärt, das zur Bahnberechnung in ihrer ursprünglichsten Form verwendet wird. In Abschnitt 2.5 werden zuletzt das SGP4-Modell, der Begriff eines J2-Perturbators und das TLE-Format kurz erläutert.

2.1. Mobile Ad-Hoc-Netzwerke

Ein *Ad-Hoc-Netzwerk* ist ein in sich geschlossenes Netzwerk, das sich selbst organisiert und keine Hierarchie besitzt. Ein weiteres Merkmal von Ad-Hoc-Netzwerken ist die Abwesenheit eines zentralen Managements, es gibt also für die Organisation des Netzwerks selbst keine Basisstation. Es ist jedoch ohne weiteres möglich, Kommunikation mit einer Basisstation zu betreiben. Die Basisstation hat aber keinen direkten Einfluss auf die Topologie des Ad-Hoc-Netzwerks.

Ein *drahtloses Sensornetzwerk* oder *Wireless Sensor Network* (WSN) besteht aus einer Menge von Sensorknoten, die per Funk kommunizieren. Diese Sensorknoten besitzen nur eine beschränkte Energiekapazität, was die Lebensdauer des Netzwerks beschränkt. In [1] wird die Lebensdauer eines Netzwerks als die Zeitspanne zwischen dem Funktionsbeginn des Netzwerks bis hin zum ersten Ausfall eines Knotens aufgrund unzureichender Energievorräte definiert. Ein Wireless Sensor Network besteht aus folgenden Elementen (nach [2]):

- **Sensorknoten** Ein *Sensorknoten* ist die Grundeinheit eines WSN. Ein WSN besteht aus einer Menge von mehreren Sensorknoten, die miteinander in Kontakt stehen. Ein solcher Sensorknoten bekommt, je nach Anwendung des WSN, eine bestimmte Aufgabe zugewiesen, die sich im Laufe der Zeit ändern kann. Beispiele für solche Aufgaben sind Routing, Datenerfassung, Datenspeicherung oder Datenverarbeitung. Sensoren besitzen eine maximale Übertragungsreichweite, die *transmission range*, die sich bei Bedarf proportional zur Sendeleistung ändern kann.

- **Cluster** *Cluster* sind organisatorische Einheiten von WSN. Ein Cluster besteht aus einem Clusterhead und eventuell mehreren zugewiesenen Knoten.

- **Clusterhead** *Clusterheads* sind die organisatorischen Leiter von Clustern. Clusterheads organisieren den Datenaustausch in Clustern sowie das Routing innerhalb dieser Cluster. Clusteralgorithmen beschäftigen sich mit der je nach Anforderung optimalen Auswahl von Clusterheads und der Zuweisung von Knoten. Es ist möglich, entsprechend ausgestattete Knoten in das WSN einzubinden, die für die Rolle eines Clusterheads prädestiniert sind. Viele Clusteralgorithmen gehen allerdings von einem WSN aus, das aus gleichen Knoten besteht, sodass jeder Knoten für die Rolle des Clusterheads in Frage kommt.

- **Gateway** *Gateways* sind zu einem Cluster gehörige Knoten, die Kommunikation zwischen zwei Clustern ermöglichen. Gateway-Knoten werden nur im LCA-Algorithmus angesprochen (siehe Abschnitt 3.2.1).

- **Basisstation** Eine *Basisstation* bildet den obersten Hierarchielevel in einem WSN. Die Basisstationen sammeln alle Informationen und besitzen sinnvollerweise eine bei weitem umfassendere Ausstattung als die Sensorknoten und bilden den zentralen Datensammel- und Verarbeitungspunkt. In den meisten Clusteringalgorithmen werden Basisstationen nicht in den Clusteringprozess miteinbezogen.

Ein *mobiles Ad-Hoc-Netzwerk* (MANET) ist ein drahtloses Ad-Hoc-Netzwerk mit mobilen Knoten [3]. MANETs werden in vielen Bereichen eingesetzt, wie zum Beispiel im militärischen Bereich, lebensgefährlichen Umgebungen oder in lokalen Netzwerken. Der große Vorteil von MANETs gegenüber Kabelnetzwerken besteht darin, Kommunikation ohne gegebene physikalische Infrastruktur zu ermöglichen und so an viele Situationen anpassbar zu sein. Allerdings sind MANETs auf einen bestimmten Sendebereich beschränkt und bieten niedrige Übertragungsraten. Außerdem ist mit Interferenzen zu rechnen, falls zu viele Knoten existieren. Darüber hinaus besitzen mobile Knoten meist nur einen beschränkten Energievorrat und daher beschränkte Aktionsmöglichkeiten [4].

2.2. Darstellung durch Graphen

Um ein MANET einfach analysieren zu können, besteht die Möglichkeit, es als ungerichteten Graphen $G = (V, E)$ mit V als Knotenmenge und $E \subseteq V \times V$ als Kantenmenge zu interpretieren. Hierbei stellen die Sensorknoten des Netzwerks die Knoten im Graphen dar. Eine Kante zwischen zwei Knoten besteht genau dann, wenn die beiden zugehörigen Sensorknoten direkt miteinander in Kontakt stehen. Für einen beliebigen Knoten $v \in V$ ist dann $N(v) \subseteq V$ die Anzahl aller Nachbarn von v, wobei $u \in V$ genau dann ein Nachbar von v ist, wenn beide Knoten mit genau einer Kante direkt miteinander verbunden sind.

Einen Spezialfall bilden hierbei Unit Disk Graphs (UDGs). Bei UDGs werden um jeden Knoten Kreise mit Einheitsradius gelegt. Falls ein Knoten v im Kreis von u

liegt, wird zwischen diese beiden Knoten eine Kante gelegt. In manchen Algorithmen genügt für eine Kantenbildung bereits, dass die Kreise überlappen, ohne dass ein Knoten im Kreis eines anderen liegen muss. Solche UDGs finden bei manchen Clusteringalgorithmen Anwendung wie beispielsweise in [5].

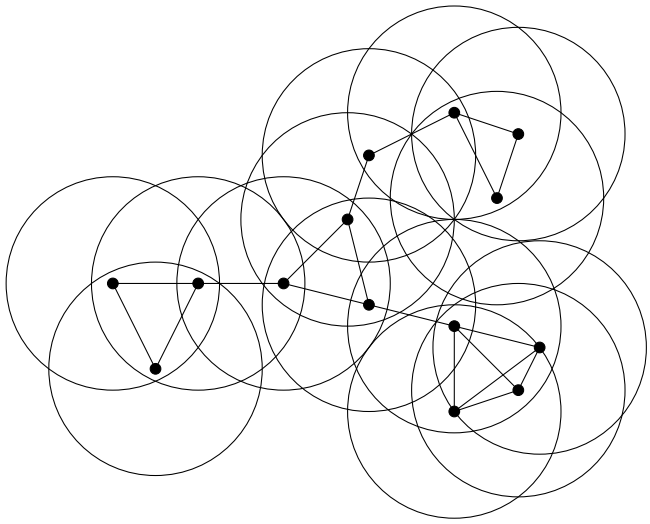


Abb. 2.1.: Eine Beispielkonfiguration von Knoten mit eingezeichneten UDGs. Der resultierende dominierende Teilgraph findet sich in 2.2

Man kann einem Knoten v auch ein Gewicht $w_v \in \mathbb{R}$ zuweisen. Dieses Gewicht kann vorher definiert sein, kann sich aber auch dynamisch ändern. Knotengewichte werden in verschiedenen Clusteringalgorithmen benutzt, um Clusterheads zu finden oder auch Knoten einem Clusterhead zuzuweisen. Diese Clusteringalgorithmen heißen "gewichtete Algorithmen". Ebenso ist es möglich, Kanten e ein Gewicht w_e zuzuweisen, wie zum Beispiel die euklidische Entfernung zweier Punkte u und v. Im Falle von MANETs ändert sich dieses Gewicht w_e im Allgemeinen über die Zeit.

Die meisten Algorithmen konstruieren einen *dominierenden Teilgraphen* oder ein *dominating set*, dessen Knoten dann als Clusterheads fungieren können. Darunter versteht man eine Teilmenge $D \subseteq V$ der Knotenmenge, sodass jeder Knoten $v \in V$ mindestens eine direkte Verbindung zu einem Knoten $w \in D$ besitzt oder selbst in D liegt [6]. Manche Algorithmen versuchen, diese Menge zu minimieren. Dieses Minimierungsproblem ist allerdings NP-hart, also werden nur Approximationen an das minimum dominating set angegeben, siehe auch [5].

2.2.1. Adjazenz- und Distanzmatrizen

Einen Graphen kann man nun mit Kantengewichten mittels einer Adjazenzmatrix kompakt darstellen. Eine Adjazenzmatrix A ist eine $|V| \times |V|$-Matrix über $\mathbb{R}$, in deren Zellen a_{ij} das Kantengewicht der Kante zwischen v_i und v_j steht. Falls keine Kante zwischen diesen Knoten existiert, wird der Wert der Zelle auf 0 gesetzt, ebenso auch bei $i = j$, wenn also Anfangs- und Endknoten der Kante gleich wären, da wir keine Kanten von einem Knoten zu sich selbst betrachten. Eine Adjazenzmatrix

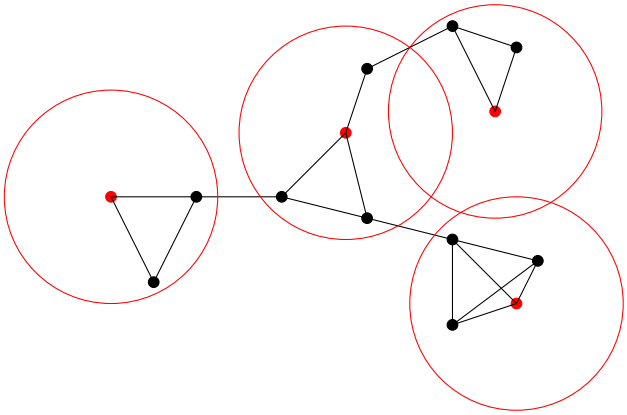
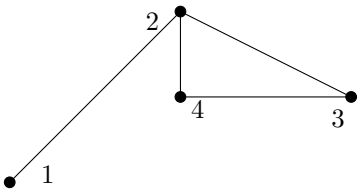


Abb. 2.2.: Das gleiche Ad-Hoc-Netzwerk wie in 2.1 mit rot gekennzeichneten Knoten des dominierenden Teilgraphen und seinen Clustern

A zeigt also an, zwischen welchen Knoten direkte Wege existieren. Setzt man die Gewichte für die Kanten auf einen Distanzwert d_{ij} zwischen den verbundenen Knoten, so bezeichnen wir eine solche Adjazenzmatrix als Distanzmatrix. Insbesondere ist eine Adjazenzmatrix symmetrisch, da die Existenz und auch Länge eines Weges zwischen zwei Punkten unabhängig von der Wahl des Startpunktes ist, wenn wir von einem bidirektionalen Graphen ausgehen. In Abbildung 2.3 ist dieser Sachverhalt an einem Beispiel dargestellt.



$$\leftrightarrow A = \begin{pmatrix} 0 & 11.2 & 0 & 0 \\ 11.2 & 0 & 4.47 & 2 \\ 0 & 4.47 & 0 & 4 \\ 0 & 2 & 4 & 0 \end{pmatrix}$$

Abb. 2.3.: Ein bidirektionaler Graph mit seiner Adjazenzmatrix A. Die Zellen beinhalten die Distanzen der einzelnen Knoten 1 bis 4, A ist also eine Distanzmatrix.

Aufgrund der dynamischen Natur von MANETs bietet es sich an, diese Matrizen in Abhängigkeit von der Zeit aufzustellen. So kann man für zwei Zeitpunkte t_1 und t_2 mit $t_1 < t_2$ jeweils die Adjazenzmatrizen $A(t_1)$ und $A(t_2)$ berechnen. Die Differenz $A(t_2) - A(t_1)$ gibt dann an, ob sich zwei Knoten voneinander weg- oder aufeinander zubewegen, je nachdem, ob die jeweilige Zelle respektive negativen oder positiven Inhalt hat. Eine Variante dieser Technik mit Sendestärken wird in mehreren Algorithmen benutzt (siehe Abschnitte 3.2.9 und3.2.10).

2.3. Clustering in Ad-Hoc-Netzen

Der Begriff Clustering beschreibt den Vorgang, eine Menge von Knoten zu gruppieren, also in Cluster aufzuteilen. Sei nun V die Menge aller Knoten. Ein Clusteringvorgang

teilt die Menge V nun in eine Anzahl $\{V_1, \ldots, V_n\}$ auf, die nicht notwendigerweise disjunkt sein müssen. Für diese Mengen gilt nun $\bigcup_{k=1}^{n} V_k = V$. Jedes V_k definiert hierbei einen zusammenhängenden Teilgraphen von V (siehe [6]). Eine solche Aufteilung kann ausschließlich abhängig von der momentanen Topologie geschehen (siehe z.B. 3.2.2), es können aber auch weitere Parameter eingebunden werden, wie zum Beispiel die verbleibende Energie eines Knotens (siehe z.B. 3.2.10).

Man kann den Clusteringvorgang in zwei Phasen aufteilen: Zu Beginn steht das Clustering Setup, um die Clusterstruktur zu erstellen. Dies geschieht durch Wahl von geeigneten Clusterheads und Zuweisung der restlichen Knoten. Die zweite Phase heißt Clustering Maintenance, in der versucht wird, die bestehenden Strukturen beizubehalten oder Änderungen an der Netzwerktopologie in die Clusterstruktur einzuflechten, wie z.B. das Einbinden eines neuen Knotens oder Überprüfung und gegebenfalls Neuerstellung der Clusterstruktur bei Ausfall eines oder mehrerer Knoten.

Der Sinn des Clusterings liegt darin, einer ungeordneten Menge von Knoten eine Struktur zu geben, auf der dann weitergehende Operationen stattfinden können. Außerdem kann so die Kommunikation zwischen den Knoten besser koordiniert und eventuelle Interferenzen vermieden werden. Zusätzlich wird durch diese Koordination Energie gespart. Mit einer gegebenen Clusterstruktur ist es außerdem einfacher, einzelne Knoten zu adressieren und hierdurch eine hierarchische Routingstruktur aufzubauen [7]. Das Clustering verteilt auch Verantwortlichkeiten und Rechen- sowie Kommunikationslasten (siehe [8]). Ein gutes Beispiel für ein geclustertes Netzwerk ist das Internet, in dem über TCP/IP kommuniziert wird.

2.4. Satellitenorbits

Um die Bewegung von Satelliten um die Erde zu beschreiben, fand Johannes Kepler in seinen Werken Astronomia Nova (1609) und Harmonice Mundi (1618) drei Gesetze, die eine ideale Orbitbahn beschreiben:

- **Erstes Keplersches Gesetz:** Die Planeten bewegen sich auf elliptischen Bahnen, in deren einem Brennpunkt die Sonne steht.
- **Zweites Keplersches Gesetz:** Eine von der Sonne zum Planeten gezogene "Kette" überstreicht in gleichen Zeiten gleich große Flächen.
- **Drittes Keplersches Gesetz:** Die Quadrate der Umlaufzeiten zweier Planeten verhalten sich wie die dritten Potenzen der großen Bahnhalbachsen.

Später wurden folgende sechs Parameter für ideale Orbitbahnen um die Erde als Zentralkörper eingeführt (siehe auch Abbildung 2.4), die den drei Keplerschen Gesetzen genügen:

ϵ Die *numerische Exzentrizität* beschreibt die Form einer elliptischen Orbitbahn, wobei ein Wert von $\epsilon = 0$ einer perfekten Kreisbahn entspricht. Bis 1 wird die

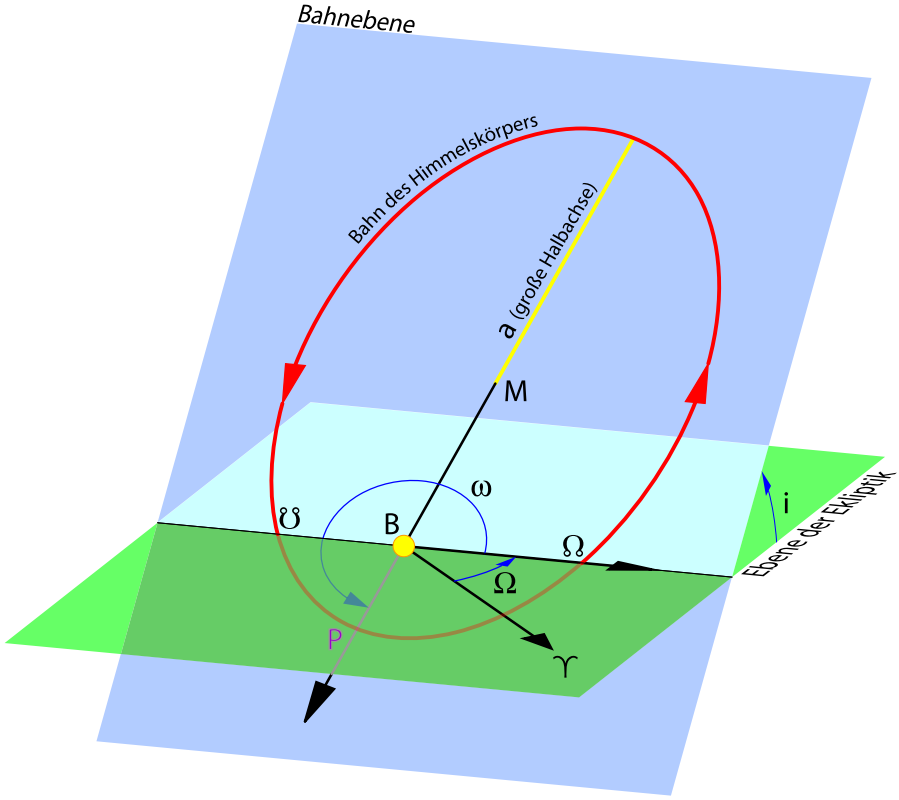


Abb. 2.4.: Alle Bahnelemente in einer Zeichnung. Im Brennpunkt B der Ellipse liegt der Zentralkörper, um den sich die Orbitbahn bewegt. M bezeichnet den Mittelpunkt der Orbitbahn. ♈ bezeichnet den Frühlingspunkt, ☊ den aufsteigenden Knoten und ☋ den absteigenden Knoten.

Ellipse entsprechend langgestreckter, je mehr ϵ wächst. Werte ab einschließlich 1 sind für unsere Anwendung nicht interessant, da sonst parabolische oder hyperbolische Bahnen erzeugt würden.

a Durch a wird die *Länge der großen Halbachse* festgelegt, die die Größe der Ellipse beeinflusst.

Durch diese beiden Elemente, die Gestaltelemente, wurde die Form der Ellipse festgelegt. Die nachfolgenden drei Elemente, die Lageelemente, beschreiben die Orientierung der Ellipse im Raum in Relation zur Äquatorialebene der Erde (oder allgemein zu einem Referenzsystem):

i Mit i wird die *Inklination* oder Neigung der Bahnebene zum Referenzsystem beschrieben.

Ω Das *Argument des Knotens*, welches mit Ω bezeichnet ist, beschreibt den Winkel des aufsteigenden Knotens ☊ zum Frühlingspunkt ♈. Der aufsteigende Knoten

bezeichnet den Schnittpunkt der Orbitbahn mit der Referenzebene, an dem die Orbitbahn die Referenzebene von unten durchstößt.

ω Durch ω wird das *Argument des Perigäums*[1] bezeichnet, das die Orientierung der Ellipse in der Bahnebene beschreibt. Das Perigäum (in Abb. 2.4 mit P bezeichnet) ist der näheste Punkt einer Orbitbahn zur Erde.

Als letztes wird ein Element angegeben, das die tatsächliche Position eines Objektes auf der Orbitbahn angibt. Dieses Element wird als Zeitparameter bezeichnet.

ν Der letzte Parameter ν beschreibt die *wahre Anomalie*, also den Winkel zwischen dem Perigäum und der Position eines Objektes auf der Orbitbahn.

Durch diese Bahnelemente werden verschiedene Differentialgleichungen definiert. Die Lösungen solcher Differentialgleichungen des Keplermodells heißen Propagators.

Als Spezialfall einer Orbitbahn soll kurz auf sonnensynchrone Orbits eingegangen werden, da solche Orbits in dieser Arbeit oft benutzt werden. Die herausstechende Eigenschaft dieser Orbits ist, dass die Orbitebene im Laufe eines Jahres immer einen festen Winkelwert zur Erde-Sonnen-Linie besitzt und somit einen Punkt auf der Erde alle 24 Stunden überfliegt. Um diese Eigenschaft zu erreichen, wurde die Präzession[2] der Orbitbahn auf den gleichen Wert wie den der Erde gesetzt.

2.5. TLE-Daten, SGP4-Propagatoren und J2-Perturbation

Im oben beschriebenen Keplermodell wird nur eine idealisierte Bahnform angegeben, die keine Rücksicht auf äußere Störungen wie z.B. die Anziehungskraft großer Körper oder die Reibung im Orbit nimmt. Der Einfluss der atmosphärischen Reibung ist beispielsweise so groß, dass ein Satellit in einem erdnahen Orbit innerhalb einiger Jahre in einer Spirale auf die Erde stürzen kann oder in der Atmosphäre verglüht. Um solche Orbitstörungen möglichst vorauszuberechnen, um sie berücksichtigen zu können, wurden verschiedene Modelle eingeführt.

Das *Two-Line Element Set*-Format beschreibt in zwei Zeilen die Orbitbahn eines Satelliten. Mit einem Modell (wie dem SGP4-Modell) kann dann anhand dieser Daten die Position eines Satelliten im Orbit zu einem gewissen Zeitpunkt berechnet werden. Diese TLE-Daten werden regelmäßig von NORAD bereitgestellt und frei im Internet auf `http://celestrak.com/` veröffentlicht. Für eine genaue Erklärung der einzelnen Komponenten eines TLE-Datensatzes siehe [9]. Die TLE-Daten, auf deren Grundlage Orbitberechnungen durchgeführt werden, besitzen eine Epoche, für diese Daten mit einem Fehler von $1km$ gelten. Dieser Fehler wächst pro Tag um ungefähr ein bis drei Kilometer, weswegen die TLE-Daten für alle Satelliten häufig erneuert werden müssen.

[1] oder Periapsis, wenn es sich um einen beliebigen Zentralkörper handelt

[2] Die Präzession beschreibt die Rotationsänderung der Orbitalebene über die Zeit aufgrund der ungleichen Masseverteilung der Erde

Das SGP4-Modell[3] wird von der NASA für alle erdnahen Satelliten mit einer Umlaufbahn von weniger als 225 Minuten benutzt und ist eins der am häufigsten benutzten Modelle, um die Position und Geschwindigkeit eines Satelliten im Orbit vorauszuberechnen [10]. Das SGP4-Modell benutzt die von NORAD[4] bereitgestellten TLE-Daten, um die Orbitbahn eines Satelliten berechnen zu können. Durch das SGP4-Modell werden verschiedene Bahnstörungen in die Berechnung einbezogen, wie beispielsweise die Reibung im niedrigen Erdorbit, die einen Satelliten abbremst und so seine Bahn verändert.

Während das SGP4-Modell für real existierende Satelliten mit TLEs verwendet werden kann, wird für simulierte, nichtexistente Satelliten auf einen J2-Perturbator zurückgegriffen, da TLE-Daten von NORAD nur für existierende Satelliten im Orbit berechnet und veröffentlicht werden. Der J2-Perturbator ist ein Ansatz, Bahnstörungen in eine Keplerbahn mit einzuberechnen, die durch die Einflüsse der Erdabflachung und ungleichmäßige Massenverteilung im Erdinneren entstehen. Nicht betrachtet werden hingegen atmosphärische Reibungen sowie Anziehungskräfte dritter Körper wie Sonne oder Mond.

[3]SGP steht für "Simplified General Perturbations"
[4]NORAD steht für das "North American Aerospace Defense Command"

Kapitel 3.

Forschungskontext

Im folgenden Kapitel wird ein Überblick über den aktuellen Forschungskontext gegeben. Im ersten Abschnitt werden Satellitennetzwerke mit Nano- und Picosatelliten betrachtet. Im zweiten Abschnitt werden einige Clusteringalgorithmen für MANETs vorgestellt sowie die Auswahl der im Zuge dieser Arbeit implementierten Algorithmen begründet.

3.1. Netzwerke von Kleinstsatelliten

Dieser Abschnitt gibt einen Überblick über aktuelle Technologien und Forschungsgebiete rund um Satellitennetzwerke von Kleinstsatelliten. Im ersten Unterabschnitt wird das CubeSat-Projekt vorgestellt, das 1999 von der California Polytechnic State University vorgestellt wurde, um es auch kleineren Unternehmen sowie Universitäten zu ermöglichen, Satellitentechnik kostengünstig zu erforschen und in der Lehre einzusetzen. Es werden weiterhin verschiedene Pico- und Nanosatellitenmissionen vorgestellt. Im zweiten Unterabschnitt wird auf Satellitenmissionen eingegangen, die auf Satellitennetzwerken aufbauen, so wie das CanX-4/-5-Projekt, in dem der Formationsflug mit zwei Satelliten erprobt wird. Der dritte Unterabschnitt fokussiert sich auf die Anforderungen von MANETs auf Satellitenbasis, während der vierte Unterabschnitt einige größtenteils noch geplante Anwendungsgebiete von Satellitennetzwerken vorstellt.

3.1.1. Das CubeSat-Projekt und verschiedene Satellitenmissionen

Das CubeSat-Projekt (siehe [11]) wurde 1999 an der California Polytechnic State University (kurz: Cal Poly) und an der Stanford University entwickelt. Mittlerweile arbeiten über 60 verschiedene Universitäten und private Firmen an eigenen Picosatelliten im Rahmen des CubeSat-Programms, die für wissenschaftliche oder kommerzielle Zwecke eingesetzt werden. Das CubeSat-Programm ermöglicht es nun, diese Satellitenprojekte kostengünstig ins All zu schicken, wo es vorher für die einzelnen Universitäten und Unternehmen alleine nur schwer möglich war. Die Cal Poly stellt hierbei ein Deployment System bereit, das einfach in bereits bestehende Luftfahrzeuge eingebaut werden kann und einen kleinen, zuverlässigen Mechanismus zum Auswurf von CubeSats enthält.

Um die Satellitengröße für ein Launch zu vereinheitlichen, wurde der CubeSat-Standard in [12] definiert, nachdem ein würfelförmiger Satellit eine Kantenlänge von

nicht mehr als $10cm$ und nicht mehr als ein Gesamtgewicht von $1.33kg$ vorweisen darf (für eine Beispielgrafik siehe 3.1). Solche CubeSats sind durch Verdopplung oder Verdreifachung erweiterbar, indem ein quaderförmiger Satellit mit Seitenlängen $20cm \times 10cm \times 10cm$ oder $30cm \times 10cm \times 10cm$ erstellt wird, der respektive maximal $2kg$ oder $3kg$ wiegen darf.

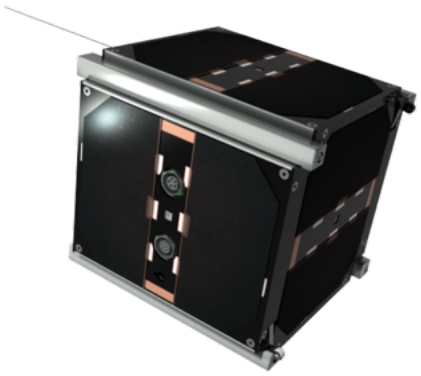


Abb. 3.1.: CAD-Zeichnung des Picosatelliten UWE-3 der Universität Würzburg als Beispiel für einen CubeSat

Ein weiterer Vorteil des CubeSat-Projekts liegt in den enormen Forschungs- und Ausbildungsmöglichkeiten. Studenten können direkt an realen Satellitenprojekten wertvolle Erfahrung sammeln, die sie in späteren Berufen in der Luft- und Raumfahrtindustrie gewinnbringend einsetzen können.

Im Folgenden sollen einige CubeSat-Missionen kurz vorgestellt werden.

Das UWE-Programm Das UWE-Programm der Universität Würzburg ist Teil des CubeSat-Programms. Das Kürzel UWE steht hierbei für "Universität Würzburg Experimental-Satellit". Derzeit befindet sich der dritte Picosatellit UWE-3 (zu sehen in Abbildung 3.1) in Entwicklung, während UWE-1 im Jahre 2005 und UWE-2 im Jahre 2009 bereits erfolgreich in ihre Umlaufbahnen geschossen wurden. Bemerkenswert ist hierbei, dass UWE-1 der erste deutsche Picosatellit im All war.

UWE-1 diente dazu, Telekommunikationsexperimente im Orbit durchzuführen, indem mehrere IP-basierte Protokolle wie TCP oder UDP im Weltraum getestet und optimiert werden, wobei typische Probleme wie Delays, relativ kleine Bandbreiten, hohe Paketverlustraten und kurze Kommunikationsfenster betrachtet wurden [13].

UWE-2 wurde mit dem Ziel konstruiert, verschiedene Methoden zur Lagebestimmung zu testen, unter anderem auch ein eigens entwickeltes System namens ADS [14]. Außerdem wurden mit UWE-2 die bei UWE-1 bereits begonnen Tests von IP-Protokollen weitergeführt. In Abbildung 3.2 ist der UWE-2-Satellit zu sehen.

UWE-1 und UWE-2 sind beides OSCAR-Satelliten [15]. OSCAR steht für Orbiting Satellite Carrying Amateur Radio, also für Satelliten, über die Amateurfunk für lizenzierte Funker über Satelliten möglich ist.

UWE-3 soll schließlich um ein Lageregelungssystem erweitert und mit ihm die bereits bestehenden Lagebestimmungssysteme überarbeitet werden [16]. Dies hat

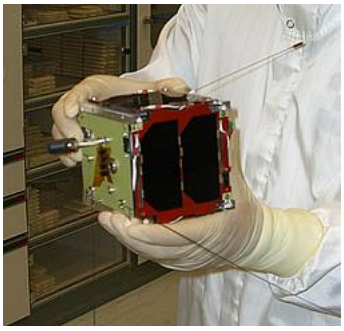


Abb. 3.2.: Flight Model des UWE-2-CubeSats.

den Zweck, die technischen Grundlagen für die Realisierung robuster Picosatellitenschwärme zu schaffen. Außerdem soll die Einsetzbarkeit von COTS[1]-Produkten im realen Weltraumumfeld getestet werden [17].

Nachfolgende Generationen von UWE-Satelliten befinden sich bereits in Planung [18].

Das CanX-Programm Das CanX-Programm[2] wurde vom Space Flight Laboratory am Institute for Aerospace Studies der University of Toronto in Kanada initiiert und soll einen kostengünstigen Zugang ins All für verschiedene Forschungs- und Entwicklungszwecke sowohl in Kanada als auch international schaffen. Die CanX-Satelliten besitzen eine Masse von ungefähr $10kg$ und benutzen moderne Technologien und Subsysteme und besitzen Lageregelungssysteme, -aktuatoren und Hochgeschwindigkeitskommunikationssysteme [19].

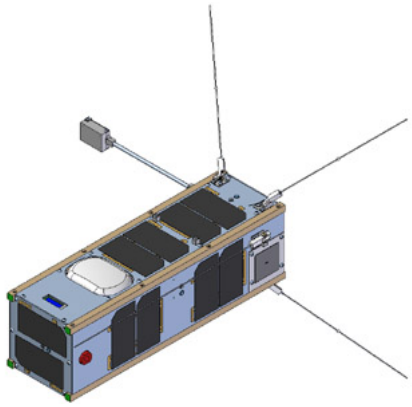


Abb. 3.3.: Der CanX-2-Satellit.

Im Rahmen dieses Programms wurden mehrere Satellitenmissionen entworfen und in eine Umlaufbahn gebracht. Der Beginn wurde mit dem Launch von CanX-1 im Ju-

[1]COTS steht für "Commercial Off The Shelf", was Massenherstellungsprodukte bezeichnet, die man aus dem Regal kaufen kann

[2]kurz für "Canadian Advanced Nanospace eXperiment"

ni 2003 gemacht. CanX-1, der zugleich der erste Nanosatellit aus Kanada war, sollte mehrere neuartige Technologien im Weltraum testen, wie zum Beispiel einen Sternsensor und ein GPS-basiertes Lagebestimmungssystem und wurde mit dem CubeSat-Programm in seine Umlaufbahn gebracht. CanX-2 (in Abbildung 3.3 zu sehen) wurde 2008 auf seinen Orbit gebracht und hatte den Zweck, neue Technologien zu erproben, die später in der CanX-4-CanX-5-Mission im Formationsflug eingesetzt werden sollten. CanX-3, auch BRITE[3] genannt, ist eine Mission, die photometrische Untersuchungen bei hellen Sternen durchführen und die Veränderlichkeit der Sterne beobachten soll. Auch dieser Satellit erhält diverse Designvorlagen, die in CanX-2 erprobt wurden.

BEESat Von der Technischen Universität Berlin wurde ein CubeSat entwickelt, der zum Ziel hat, bereits existierende Techniken zu miniaturisieren und in Kleinstsatelliten einzubauen. Hauptsächlich sollen hierbei neu entwickelte Mini-Reaktionsräder zur Lageregelung von Picosatelliten getestet werden. In weiterer Zukunft sollen so Grundlagen für Schwärme von Pico- und Nanosatelliten gelegt werden, um verschiedene Anwendungsfelder auf diesem Gebiet erschließen zu können [20].

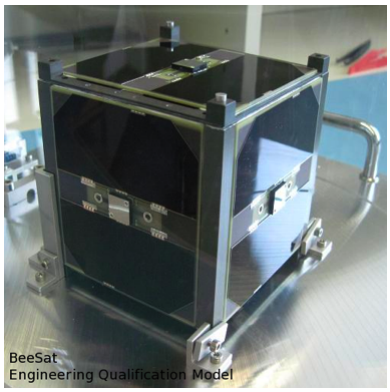


Abb. 3.4.: BEESat-1

BEESat-1 wurde 2009 zusammen mit UWE-2 in eine sonnensynchrone Umlaufbahn gebracht. Der Hauptzweck von BEESat-1 war es, die neu entwickelten Mini-Reaktionsräder in der Lageregelung praktisch zu erproben. Nach einem Jahr und 10 Monaten sendet BEESat-1 immer noch Signale und erst jetzt musste der redundante Bordcomputer aktiviert werden, da auf dem Hauptbordcomputer Anomalien in der Telemetrie aufgetreten waren. Das Missionsziel von einem Jahr Lebensdauer im Orbit wurde also erreicht und übertroffen [21].

BEESat-2 wird momentan entwickelt und besitzt das gleiche Grundkonzept wie BEESat-1, allerdings mit verbesserter Hardware, wie beispielsweise einer neuen Kamera und einem besseren Nutzlastdaten-Computer [22]. Die Forschungsziele des Vorhabens bestehen primär in der Implementierung eines innovativen Lageregelungssystems für Picosatelliten und dessen experimentelle technische Erprobung unter Welt-

[3]kurz für "BRIght-star Target Explorer"

raumbedingungen. Ein sekundäres Ziel ist die Einbindung des Satellitenbetriebs in den Lehrbetrieb der Universität.

BEESat-3 befindet sich ebenfalls noch in der Entwicklung und soll primär in der Lehre eingesetzt werden. Sekundäre Ziele sind der Test des S-Band-Senders HISPICO, die Erprobung einer Erdbeobachtungskamera sowie weiterer Picosatellitentechnologie [23].

3.1.2. Formationen und Konstellationen von Kleinstsatelliten

In diesem Unterabschnitt sollen verschiedene Satellitenmissionen vorgestellt werden, die der Kommunikation zwischen mindestens zwei Satelliten bedürfen und somit bereits ein kleines Netzwerk bilden.

CanX-4 & CanX-5 Die Zwillingssatelliten CanX-4 und CanX-5 entstammen ebenfalls dem oben genannten CanX-Programm. CanX-4 und CanX-5 sind 2008 in einen gemeinsamen Orbit gebracht worden [24, 25]. Der Hauptzweck dieser Mission liegt darin, Formationsflug im All zu erforschen, wobei das Projekt nach eigenen Angaben hierbei eine Vorreiterrolle einnimmt[4]. Hierbei fliegen CanX-4 und CanX-5 in einer autonomen Leader-Follower-Formation. Beide Satelliten werden identisch aufgebaut sein und dabei Hardware benutzen, die schon mit CanX-2 erfolgreich getestet wurde. Außerdem wurde eigens für die Mission ein neues Antriebssystem entwickelt [27].

Die Mission ist hauptsächlich zu Demonstrationszwecken gedacht: Im Orbit sollen mehrere Satellitenformationen für zwei Satelliten erreicht und beibehalten werden, diverse relative Lagebestimmungsmessungen mit einer Genauigkeit bis maximal 10cm und absolute Lagebestimmungsmessungen mit Genauigkeiten bis maximal 1m durchgeführt werden und schließlich ein Kommunikationssystem zwischen Satelliten getestet werden. Außerdem sollen effiziente Formationsflugalgorithmen entwickelt und validiert werden, die den Treibstoffverbrauch optimieren [27].

BRITE-Constellation Das BRITE-Projekt, kurz für "BRIght Target Explorer", hat den Zweck, visuell schwache Oszillationen und Temperaturveränderungen bei Sternen heller als $4.0 mag$[5] zu beobachten und aufzuzeichnen, und zwar mit einer so hohen Präzision und zeitlichen Abdeckung, wie es mit Mitteln auf der Erdoberfläche nicht möglich wäre. Dafür sollen 6 Nanosatelliten mit jeweils optischen Teleskopen mit einer Apertur von 3cm in einer Konstellation fliegen und per Interferometrie hochauflösende und farbige Bilder von hellen Sternen aufnehmen [24].

HUMSAT Das HUMSAT-Projekt [28, 29] ist eine internationale Initiative, initiiert von drei Universitäten: Der University of Vigo (Spanien), der Universidad Nacional Autónoma de México und der bereits bekannten California Polytechnic University, die auch das CubeSat-Projekt begründet hat. Ziel des Projekts ist es, eine Konstellation von Nanosatelliten zu konstruieren, die Kommunikationsmöglichkeiten für Gegenden

[4] Tatsächlich war das SPHERES-Projekt schneller und hat bereits 2006 Formationsflüge mit Picosatelliten vorgenommen [26]

[5] mag ist eine logarithmische Einheit für die Helligkeit von Planeten.

bereitstellen soll, die eine schlechte bis keine Kommunikationsinfrastruktur besitzen. Das HUMSAT-Projekt soll hierbei diese fehlende Infrastruktur ersetzen. Das Projekt wurde außerdem bewusst für die Lehre initiiert, sodass Studenten entsprechender Fachrichtungen direkte Erfahrungen mit einem Weltraumprojekt sammeln können, indem sie von Beginn bis Ende an allen Teilen des Projektes mitwirken können. Sie lernen darüberhinaus das Projektmanagement der ESA kennen. Durch die Mitarbeit vieler Universitäten, Forschungseinrichtungen und Raumfahrtagenturen soll außerdem die internationale Zusammenarbeit verbessert und gestärkt werden, insbesondere mit ärmeren und Entwicklungsländern.

Das HUMSAT-Projekt ist in drei Bereiche aufgeteilt: In das Space Segment, in dem sich 9 Nanosatelliten in einer Konstellation befinden und ein Ad-Hoc-Netzwerk bilden, in das Ground Segment, das auf dem GENSO-Netzwerk[6] aufbaut und in das User Segment, das auf billigen Bodensensoren basiert und die Benutzerbasis des Netzwerks darstellt, die über HUMSAT kommuniziert.

Laut den Initiatoren des Projekts soll das Netzwerk für verschiedenste Anwendungen verfügbar sein, wie die Beobachtung und Prävention von Naturkatastrophen, Umweltverschmutzungen oder Klimabeobachtungen. Durch das GENSO-Netzwerk soll durchgehender Kontakt zu allen Satelliten möglich sein.

QB50 QB50 bezeichnet ein Netzwerk von 50 internationalen Satelliten, das das wissenschaftliche Ziel hat, in der niederen Thermosphäre in situ-Messungen zeitlicher und örtlicher Phänomene durchzuführen. Hierbei besteht das Netzwerk aus 50 Double CubeSats, die jeweils einen Abstand von 100km zueinander besitzen und identische Sensoren besitzen. Außerdem soll der Wiedereintrittsvorgang in die niedrige Atmosphäre erforscht werden, indem unter anderem die berechnete und tatsächliche Flugbahn der Satelliten verglichen wird. Der Launch des gesamten Projekts soll 2014 erfolgen [30].

QB50 unternimmt nun den Versuch, ein Netzwerk von 50 Satelliten in der niederen Thermosphäre aufzubauen, bewusst mit CubeSats, da diese sehr billig herstellbar und schnell entwickelbar sind. Gerade dieser Kostenpunkt hat verschiedene Raumfahrtagenturen bisher abgeschreckt, eine derartige Mission zu planen oder gar durchzuführen, da die hohen Kosten von 50 Satelliten auf Industriestandard die kurze Lebenszeit im Orbit von geplanten 3 Monaten nicht rechtfertigen würden [31, 32]. Der Downlink erfolgt hierbei über das GENSO-Netzwerk [33].

Bei einem so großen Netzwerk ist es wichtig, die Kommunikation unter den Satelliten zu koordinieren. Diese Diplomarbeit benutzt das QB50-Setting, um die Methode des Clusterings zur Strukturierung des Netzwerks zu benutzen, um so mögliche Kommunikationspfade für die Kommunikation zwischen Satelliten vorzugeben.

[6] Das GENSO-Netzwerk hat zum Ziel, ein weltweites Netz von Bodenstationen von Universitäten oder Amateuren aufzubauen, das über das Internet kommuniziert, um die Operationen von Universitäten zu unterstützen, siehe auch `http://www.genso.org/`

3.1.3. Anforderungen an Satelliten-MANETs

In [34] wurde im Rahmen der NaKoFo-Studie[7] analysiert, welche Vorteile man aus dem Einsatz von MANETs auf Satelliten ziehen könnte und welche Probleme damit einhergehen. Der große Vorteil von MANETs gegenüber Formationen oder Konstellationen im All liegt darin, dass auf der einen Seite keine festen relativen Positionen eingehalten werden müssen und auf der anderen Seite keine Koordinierung seitens einer Bodenstation notwendig ist. Auch sind MANETs fehlertoleranter als Formationen und Konstellationen, da nicht durch Ausfall weniger Satelliten das Gesamtsystem kompromittiert werden würde. Bisher wurden MANETs allerdings nur auf der Erde eingesetzt und erprobt, nicht jedoch im Orbit. Satellitennetzwerke im Orbit beschränken sich bis jetzt erst auf so wenige Satelliten, dass MANETs ungeeignet wären.

Verschiedene Forschungsbereiche auf diesem Gebiet sind beispielweise das Routing in Satelliten-MANETs sowie der Aufbau und die Sicherheit von Links zwischen Satelliten. Im Bereich des Routings gibt es verschiedene Ansätze, die Netzwerktopologie festzustellen und sinnvoll zu nutzen, unter anderem auch das Cluster Based Routing, bei dem hierarchisch über Clusterheads und Gateways geroutet wird. Auch existieren manche Routing-Algorithmen, die auf kleinere MANETs ausgelegt sind und entsprechend Probleme mit einer großen Anzahl an Knoten haben und entsprechend anders herum. Ein weiteres Problem ist die Adressierung von Knoten in einem Netzwerk sowie Interferenzvermeidung. Wie in allen Netzwerken mit begrenzter Sendereichweite der einzelnen Knoten existiert auch hier das Hidden-Node-/Exposed-Node-Problem. Auch wurden in [34] verschiedene Methoden zum Verbindungsaufbau besprochen sowie zum Sendescheduling.

3.1.4. Anwendungsgebiete von Satellitennetzwerken

Die Anwendungsmöglichkeiten von Satellitennetzwerken sind mannigfaltig. Viele Projekte beschäftigen sich mit Erdbeobachtungen [27, 29, 35–37], sei es zu humanitären [29,35] oder militärischen Zwecken [36]. Für Erdbeobachtungen werden normalerweise erdsynchrone Orbits benutzt, d.h. Orbits, bei denen sich nach einer gewissen Zeit die Bahn des Satelliten auf der Erde wiederholt [38]. Bekannte und bereits seit Jahren aus dem täglichen Leben kaum mehr wegzudenken sind zum Beispiel das Global Positioning System [39], das im Begriff ist, durch das europäische GALILEO-Projekt erweitert zu werden [40].

Andere Projekte wiederum betrachten höhere Schichten der Atmosphäre [32, 41]. Der Einsatz von Satelliten zu Telekommunikationszwecken ist an sich nicht neu, die Idee, diese Aufgabe von einzelnen wenigen Satelliten, die meistens nur als Zwischenstation fungierten, auf ein Satellitennetzwerk zu verteilen, allerdings schon [29]. Allgemein gesehen ist es gerade auch durch Formationsflug im All möglich, größere Satelliten durch mehrere kleine Satelliten zu ersetzen, die die gleiche Funktionalität erfüllen können oder sogar Aufgaben übernehmen können, die vorher gar nicht möglich waren, wie beispielsweise Interferometric Imaging [24,27]. Interferometric Imaging bezeichnet

[7] NaKoFo steht für Navigations- und Kommunikationstechnologien für Kleinsatelliten-Formationen

ein Bildgebungsverfahren, das aus der Überlagerung verschiedener Wellen ein höher auflösendes Bild erzeugt, als es mit einem Empfänger möglich wäre. Ein aktuelles Beispiel für Interferometrie ist das LISA-Projekt, das mit 3 Satelliten Gravitationswellen misst, um Informationen über den Ursprung des Universums herauszufinden [42, 43].

3.2. Clusteringalgorithmen für MANETs

Im Bereich von MANETs wurden schon seit mehreren Jahrzehnten verschiedene Heuristiken und Algorithmen entworfen, die Clusteringprobleme lösen sollen. In diesem Abschnitt werden einige Algorithmen betrachtet und ihre Vor- und Nachteile diskutiert. In jedem Unterabschnitt werden die Systemanforderungen diskutiert, also die Anforderungen an das Netzwerkmodell, die dazu notwendig sind, um den Algorithmus auszuführen. Anschließend wird das Ziel des Algorithmus angegeben, gefolgt von einer kurzen Beschreibung des Ablaufs. Als letztes werden Eigenschaften des Algorithmus diskutiert, die vor- oder nachteilhaft für eine Adaption auf Satellitennetzwerke sind.

3.2.1. LCA: Linked-Cluster-Algorithm

Einer der ersten Clustering-Algorithmen wurde 1984 von Baker, Ephremides und Flynn vorgestellt (siehe [44]). Der LCA zielt darauf ab, eine Linked Cluster Architecture zu erstellen. In dieser Architektur werden Knoten in Clustern zusammengefasst und zusätzliches Augenmerk auf Kommunikation zwischen Clustern gelegt.

Für den Algorithmus genügt es laut den Autoren, dass Knoten nur partielles Wissen über das Netzwerk besitzen, genauer gesagt, ihre Nachbarn kennen. Jeder Knoten besitzt eine beschränkte Sendereichweite r. Diese ist bei allen Knoten gleich. Der Algorithmus kategorisiert alle Knoten in drei Klassen: Clusterheads und -members wie gehabt, außerdem zusätzlich in Gatewayknoten, die zur Kommunikation zwischen zwei Clusterheads dienen. Durch die Gatewayknoten und Clusterheads wird das "Backbone Network" definiert, das Kommunikation zwischen verschiedenen Clustern ermöglicht.

Der Algorithmus läuft sehr intuitiv ab: Als erster Clusterhead wird ein beliebiger Knoten A gewählt. Alle Knoten innerhalb seiner Reichweite werden zu seinem Cluster hinzugefügt. Als nächstes wird ein beliebiger Knoten B außerhalb des Clusters von A gewählt, zum Clusterhead erklärt und wiederum alle Nachbarn zu seinem Cluster hinzugefügt. Dieser Vorgang wiederholt sich so lange, bis alle Knoten entweder Clusterheads sind oder einem Cluster zugewiesen wurden.

Durch diese Vorgehensweise wird sichergestellt, dass kein Clusterhead direkten Kontakt mit anderen Clusterheads besitzt, wodurch eine ausreichende Verteilung von Clusterheads über das Netzwerk gewährleistet wird.

Anschließend werden Gatewayknoten bestimmt. Als Gatewayknoten kann jeder Knoten fungieren, der Kontakt zu mindestens einem anderen Knoten außerhalb seines Clusters besitzt. Für Gatewayknoten gibt es zwei Möglichkeiten: Im ersten Fall besitzen sie direkten Kontakt zu zwei Clusterheads, im zweiten Fall nur zu einem anderen Gateway.

Die Kommunikation wird hierbei näher definiert. Laut dem Autor soll eine Time Division Multiple Access-Struktur eingeführt werden, sodass jedem Knoten für jeden Nachbarn ein Sendeslot eingerichtet wird. Hierbei ist eine Transmission Schedule 2^k Slots lang, wobei k für die Anzahl der Knoten im Netzwerk steht. Die Länge soll so angepasst sein, dass keine Kollisionen beim Nachrichtenaustausch auftreten. Die Schedulelänge kann außerdem durch Implementierungen begrenzt werden. Damit ein Knoten weiß, welche Knoten sich in seiner Nachbarschaft befinden, werden nur $2k$ Slots benötigt. Für kleine Netzwerke ist diese Zahl noch nicht zu groß, wenn allerdings Ad-Hoc-Netzwerke mit 1000 oder mehr Knoten auftreten, wird dieser Algorithmus zu langsam.

Der Algorithmus an sich ist sehr abstrakt, was ihn einfach anpassbar und implementierbar macht. Von den einzelnen Knoten wird keine spezielle Funktionalität gefordert. Allerdings würde in einer getreuen Umsetzung von jedem Knoten gefordert werden, dass dieser komplettes Wissen über das Netzwerk besitzt, also weiß, wieviele Knoten sich im Netzwerk befinden, welche davon seine Nachbarn sind und was welche Aktion welcher Knoten gerade durchführt. Ansonsten wäre es ohne eine zentrale Stelle nicht möglich, den Clusteringvorgang durchzuführen Nachteilig ist auch, dass durch die Zufallsauswahl des Clusterheads nicht vorhersehbar ist, wie groß die zugehörigen Cluster werden.

Anpassbar an unser Szenario wäre er sehr einfach, da Bodenstationen keine besondere Rolle einnehmen könnten außer der eines Clusterheads. Ein Vorteil bei der Implementierung wäre durch die Gatewayknoten gegeben, da durch diese nicht ein Satellitenclusterhead direkten Kontakt zu einer Bodenstation haben müsste, sondern die Kommunikation eben auch über einen Gateway laufen könnte.

3.2.2. Lowest-ID und Highest-Degree

Mario Gerla und Jack Tzu-Chieh Tsai haben 1995 in einem Paper [45] zwei einfache Clusteringalgorithmen vorgestellt, deren primäre Aufgabe es ist, Cluster zu bestimmen, ohne dabei auf andere Parameter wie verbleibende Energie und relative Mobilität zu achten. Bei beiden Algorithmen wird vorausgesetzt, dass alle Knoten gleich sind und eine eindeutige ID zugewiesen bekommen haben. Für alle Knoten wird außerdem angenommen, dass sie nur eine begrenzte Übertragungsreichweite besitzen, die auf einen Wert festgelegt ist.

Der Lowest-ID-Algorithmus (LID) funktioniert denkbar einfach: In regelmäßigen Abständen broadcastet jeder Knoten eine Liste aller Knoten in seiner Reichweite. Falls ein Knoten nur von anderen Knoten hört, deren ID höher ist als seine eigene, so definiert er sich selbst als Clusterhead. Falls ein Knoten von einem anderen Knoten hört, der eine niedrigere ID hat als er selbst, nimmt er diesen Knoten als Clusterhead an, außer dieser hat die Rolle als Clusterhead explizit nicht angenommen. Ein Knoten, der mehr als zwei Clusterheads hört, ist ein Gateway. Alle Knoten, die weder Gateway noch Clusterheads sind, heißen ordinary nodes.

Der Highest-Degree-Algorithmus (HD) ist ähnlich zu LID. Auch hier broadcastet jeder Knoten eine Liste aller Knoten in seiner Reichweite. Ein Knoten, der noch keinen Clusterhead gewählt hat, heißt *nicht bedeckt*, ansonsten *bedeckt*. Ein Knoten

wird dann zum Clusterhead gewählt, falls er den höchsten Grad von allen seinen nicht bedeckten Nachbarn besitzt. Falls zwei Knoten in Reichweite liegen und den selben Grad besitzen, gewinnt der Knoten mit der niedrigsten ID. Ein Knoten, der bereits einen anderen Knoten als Clusterhead gewählt hat, kann nicht selbst Clusterhead werden.

Bei beiden Algorithmen besitzt jeder Knoten entweder einen Clusterhead als Nachbarn oder ist selbst ein Clusterhead. Falls ein Knoten sich aus einem Cluster herausbewegt, wird von ihm ein Reclustering initiiert.

Der LID-Algorithmus überlastet Knoten mit einer niedrigen ID, während er Knoten mit einer hohen ID quasi nicht beansprucht. Falls dieser Algorithmus also auf Satelliten implementiert wird, kann man voraussagen, dass der Satellit mit ID 1 mit hoher Wahrscheinlichkeit sehr viel mehr Energie benötigt als die restlichen Knoten und deshalb einen größeren Energiespeicher benötigt, im Gegensatz zu Satelliten mit sehr hohen IDs, bei denen an der Energie gespart werden kann. Dies erfordert, dass nicht gleiche Satelliten in ein Netzwerk gebracht werden, da durch die IDs eine vorherige Priorisierung der Knoten vorgenommen wird. Laut Aussage der Autoren ist der LID-Algorithmus stabiler als der HD-Algorithmus, da der Auswahlparameter für Clusterheads, die ID des Knoten, konstant bleibt. Der HD-Algorithmus hingegen stößt häufigere Rollenwechsel an, da durch die Knotenbewegung der Grad eines Knotens schnell wechseln kann. Dadurch werden zwar nicht bestimmte Knoten überbeansprucht, die Clusterheads aber werden umso stärker ausgelastet, je höher ihr Grad ist. Dies ist vor allem bei schwach mobilen Netzwerken ein Problem, da sich der Knotengrad nur langsam ändern kann und somit die Clusterheads sehr schnell an Energie verlieren.

Beide Algorithmen sind sehr einfach anpassbar. Durch die oben genannten Nachteile sind beide Algorithmen in ihrer eigenen Weise nicht praktisch für unser gegebenes Szenario, da der LID-Algorithmus einzelne Knoten überbeansprucht und der HD-Algorithmus sehr häufig Reclusterings anstößt, wodurch beide Algorithmen von vornherein sehr energieaufwändig sind. Ein verbesserter Algorithmus wurde 1997 in [46] veröffentlicht und wird in 3.2.3 vorgestellt.

3.2.3. Routing in clustered multihop, mobile wireless networks with fading channel

Chiang, Wu, Liu, und Gerla stellten im Jahr 1997 eine Verbesserung für LID und HD (siehe 3.2.2) vor [46]. Der LCC-Algorithmus ist aber letztendlich nur eine verbesserte Cluster maintenance-Phase. Für das Cluster Setup benutzt LCC entweder LID oder HD, je nach Implementierung. Im Gegensatz zu LID und HD wird nun allerdings nicht jedes Mal dann neu geclustert, wenn sich irgendetwas an der Clusterstruktur ändert, wie der Ausfall oder das Hinzukommen eines Knotens, sondern nur noch, wenn ein Knoten aus dem Cluster ausscheidet und sich an kein anderes anschließen kann. Das andere Ereignis, das zu einem Reclustering führt, ist ein Verbindungsaufbau zwischen zwei Clusterheads. Letzteres Reclustering ist allerdings ein eher “schwaches” Reclustering, da sich ein Clusterhead dem anderen entsprechend des zum Setup gewählten Algorithmus unterordnet.

Im Paper wurden Testergebnisse in Grafiken vorgestellt, die zeigen, dass die LCC-Varianten von LID und HD beinahe gleich abschneiden, allerdings eine leichte Reduktion gegenüber LID und eine immense Reduktion gegenüber HD in Bezug auf Clusterhead-Wechsel erzielen. Die Verbesserung in der Maintenance-Phase ist zwar ein Fortschritt im Algorithmendesign gegenüber 3.2.2, die bereits existierenden Probleme, wie die Überlastung des Knotens mit der höchsten ID beim LID-Algorithmus bleiben aber bestehen. Ein Lösungsansatz, der die Knoten-IDs durch Gewichte verallgemeinert, findet sich in 3.2.4.

3.2.4. DMAC: Distributed Mobility-Adaptive Clustering in Ad Hoc Networks

In [47] hat Stefano Basagni zwei Clusteringalgorithmen beschrieben. Der erste Algorithmus (DCA) wurde speziell für quasistatische Netzwerke entworfen, also für Netzwerke, in denen die Knoten fixiert sind oder sich kaum bewegen. Der zweite Algorithmus (DMAC) liefert die selben Cluster wie DCA, ist allerdings auf mobile Netzwerke ausgelegt. Als Voraussetzungen für beide Algorithmen wird angegeben, dass jeder Knoten den Algorithmus ausführt, aber nur wissen muss, wie seine Nachbarschaft beschaffen ist. Außerdem bekommt jeder Knoten v ein festes Gewicht w_v zugewiesen, das seine Qualifikation als Clusterhead repräsentiert. DCA und DMAC sind also beide gewichtete Algorithmen.

Beide Algorithmen besitzen drei Designvorgaben, die in ähnlicher Form bei allen Algorithmen zu finden sind. Als erstes soll jeder Knoten, der kein Clusterhead ist, mindestens einen Clusterhead als Nachbarn besitzen. So wird sichergestellt, dass alle Knoten in die Clusterstruktur eingefügt sind. Sollte ein Knoten nun Kontakt zu mehreren Clusterheads haben, so wird der beste Clusterhead ausgewählt. Dadurch soll eine bessere Funktionalität des Netzwerks gewährleistet werden, die sich beispielsweise durch eine höhere Lebensdauer ausdrückt. Die letzte Forderung besteht darin, dass zwei Clusterheads nicht direkt benachbart sein können, wodurch eine gewisse Verteilung der Clusterheads über das Netzwerk geschaffen werden soll.

Für den DCA-Algorithmus wird vorausgesetzt, dass sich die Netzwerktopologie während der Ausführung des Algorithmus nicht ändert. Außerdem wird angenommen, dass eine gesendete Nachricht nach einer bestimmten Zeit alle Nachbarn korrekt erreicht und jeder Knoten seine eigene ID und Gewicht sowie die ID und die Gewichte seiner Nachbarn kennt. Der Algorithmus wird auf jedem Knoten einzeln ausgeführt und die Handlungen der Knoten hängen allein davon ab, welche Entscheidungen Knoten in ihrer Nachbarschaft getroffen haben, die ein größeres Gewicht besitzen. Wenn ein Knoten also keine Nachricht von seinen Nachbarn mit einem höheren Gewicht empfängt, dass er einem Cluster beitreten soll, deklariert sich der Knoten selbst als Clusterhead und sendet diese Information an alle Knoten in seiner Nachbarschaft, die ein geringeres Gewicht als er besitzen. Der DCA-Algorithmus hängt in seiner Ausführung davon ab, welche Nachrichten er empfängt (oder nicht empfängt) und läuft deshalb vollkommen verteilt ohne eine zentrale Steuerungsstelle ab. Im Paper werden weiterhin explizit Routinen in Pseudocode angegeben, die ausgeführt werden sollen, falls ein Knoten eine bestimmte Nachricht empfängt.

Der DMAC-Algorithmus besitzt unter anderem die gleichen Voraussetzungen wie der DCA-Algorithmus, abgesehen davon, dass Knoten nun während der Ausführung des Algorithmus mobil sein können, d.h. Verbindungen aufgebaut und abgebrochen werden können. Daraus folgen dann zusätzliche Voraussetzungen: Falls eine Verbindung aufgebaut werden kann oder abgebrochen wird, wird jeder Knoten benachrichtigt, die Aktionen des DMAC-Algorithmus können nicht unterbrochen werden und zu Beginn des Algorithmus ist jeder Knoten als Nicht-Clusterhead definiert. Auch hier werden verschiedene Prozeduren in Pseudocode angegeben, die bei entsprechenden Events aufgerufen werden.

Für beide Algorithmen wird ihre Korrektheit anhand der Designvorgaben gezeigt. Vergleiche zu anderen Algorithmen werden allerdings nicht gezogen, ebensowenig wurde der Algorithmus simuliert oder anderweitig praktisch implementiert, sondern nur in der Theorie vorgestellt. Ein weiterer Nachteil ist, dass alle Knoten feste Gewichte zugewiesen bekommen. So verbraucht der Knoten mit dem höchsten Gewicht am meisten Energie, da er durchgehend Clusterhead sein muss und fällt somit früher aus, was für die Cluster- und Netzwerklebensdauer sehr nachteilig ist.

3.2.5. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks

In einem im Jahre 2002 von Chatterjee, Das und Turgut veröffentlichten Paper [48] wird ein Algorithmus zur Clusterauswahl beschrieben, der wie DMAC auf Gewichten von Knoten aufbaut. Diese Gewichte werden allerdings bei jedem Aufruf des Algorithmus neu berechnet, was eine entscheidende Neuerung zu den DCA- und DMAC-Algorithmen (3.2.4) darstellt, da nun nicht mehr gewisse Knoten von vornherein stärker belastet werden, sondern sich diese Belastung besser auf das gesamte Netzwerk aufteilt. Der Algorithmus wird ereignisorientiert aufgerufen, d.h. jedes Mal, wenn der Kern des Graphen nicht mehr Kontakt zu allen anderen Knoten besitzt. Jeder Knoten besitzt außerdem a priori als Parameter eine Zahl δ, die angibt, wieviele Nachbarn der Knoten im Idealfall haben sollte.

Jeder Knoten v erstellt eine Liste aller Nachbarn und berechnet seinen Grad d_v, also die Anzahl aller Knoten, die direkte Nachbarn des Knotens sind. Außerdem wird die Graddifferenz $\Delta_v = |d_v - \delta|$, die Summe der Distanzen zu allen Nachbarknoten D_v, seine mittlere Bewegung M_v und die als Clusterhead zugebrachte Zeit P_v berechnet. Die Graddifferenz ist notwendig zur Einschätzung der Kommunikationseffizienz, da zu wenige oder zu viele Knoten jeweils schlechter als eine Zahl nahe der idealen Anzahl δ ist. Leider sind keine näheren Informationen zur Berechnung dieser Graddifferenz angegeben. Aus der Summe der Distanzen zu den Nachbarknoten lässt sich der Schluss ziehen, welche Knoten in der nächsten Zeit aus dem Cluster ausscheiden könnten. Wenn also entsprechend diese Summe sehr groß ist, liegen entweder mehrere weit entfernte Knoten in der Nachbarschaft oder aber sehr viele. Die mittlere Bewegung des Satelliten definiert, wie schnell oder langsam sich der Knoten bewegt. Eine hohe Geschwindigkeit ist gerade für Clusterheads eher nachteilhaft, da die Nachbarschaft sehr oft wechseln kann. Allerdings hängt die Nachteilhaftigkeit immer vom Netzwerk, also den umliegenden Knoten ab; die Geschwindigkeit sollte natürlich angepasst sein,

da ein immobiler Knoten in einem hochmobilen Netzwerk ebenso schlecht als Clusterhead geeignet ist wie ein hochmobiler Knoten in einem stationären Netzwerk. Die als Clusterhead zugebrachte Zeit kann man als Pendant zur verbrauchten Energie sehen und ist ein wichtiger Parameter für Clusteralgorithmen in Ad-Hoc-Netzwerken. Falls ein normaler Knoten ausfällt, hat dies keine so weitreichenden Konsequenzen wie der Ausfall eines Clusterheads.

All diese Größen werden schließlich miteinander verrechnet und bilden das kombinierte Gewicht $W_v = c_1\Delta_v + c_2D_v + c_3M_v + c_4P_v$. Die Parameter c_i, $i = 1..4$ sind von vornherein definiert und sollen insgesamt normiert sein, d.h. $\sum_{i=1}^{4} c_i = 1$. Welche Größen wie gewichtet werden, hängt von den Prioriäten der Anwendung ab. Jeder Knoten sendet nun sein kombiniertes Gewicht an alle seine Nachbarn. Stellt ein Knoten fest, dass er das höchste Gewicht in seiner Nachbarschaft besitzt, deklariert er sich als Clusterhead und teilt dies wiederum seinen Nachbarn mit, die sich ihm dann anschließen und dies wiederum allen ihren Nachbarn mitteilen.

Es wurde außerdem darauf eingegangen, wie die Clusterstruktur möglichst beibehalten kann und bei Bedarf einzelne Satelliten ihre Clusterzugehörigkeit wechseln, ohne einen Reclusteringprozess anstoßen zu müssen. Dafür beobachtet jeder Knoten die Signalstärke des Clusterheads und benachrichtigt den Clusterhead rechtzeitig, falls abzusehen ist, dass der Kontakt bald abgebrochen werden muss, um so ein neues Cluster zu finden oder komplett neu zu clustern.

Es wurden Simulationen mit verschiedenen Knotenzahlen durchgeführt. Getestet wurden unter anderem die Anzahl der erzeugten Cluster, die Neuzuweisungen pro Zeiteinheit und die Zahl an Dominant-Set-Updates, jeweils in Relation zur Übertragungsreichweite. Zusätzlich wurde der Algorithmus mit anderen Algorithmen, namentlich Lowest-ID, Highest-degree (beide 3.2.2) und Node-weight (3.2.4) in puncto Neuzuweisungen pro Zeiteinheit verglichen, in denen der WCA-Algorithmus stellenweise deutlich besser abschnitt. Ein angenehmer Punkt dieses Algorithmus liegt darin, dass man auch selbst Größen definieren kann, die besser an eine spezielle Anwendung angepasst sind.

Eine verbesserte Version mit genetischen Algorithmen wurde in [49] veröffentlicht.

3.2.6. Max-Min d-Cluster Formation in Wireless Ad Hoc Networks

Amis, Prakash, Vuong und Huynh stellen 2000 eine neue Heuristik zum Clustern von MANETs vor, die d-Hop-Cluster erzeugt [50], also Cluster, in denen Members bis zu d Hops vom Clusterhead entfernt sein können. Die vorgestellte Heuristik ist laut den Autoren für mehrere tausend Knoten in einem MANET gedacht, wo die Idee eines d-Hop-Clusters auch Sinn macht, um nicht Unmengen kleiner Cluster zu erzeugen, sondern einige wenige, die dafür größer sind. Die Konstruktion eines d-Hop-dominierenden Teilgraphs, also eines dominierenden Teilgraphs, dessen Knoten alle anderen Knoten des Graphen über maximal d Hops erreichbar sind, ist allerdings NP-vollständig. Zielvorgaben für den Entwurf des Algorithmus waren hauptsächlich

die Begrenzung der Laufzeit abhängig allein von der Anzahl der Hops d, Minimierung der Anzahl an Clusterheads als Funktion von d, gleichmäßige Verteilung der Verantwortungen, das Cluster stabil zu halten und außerdem Stabilität der Cluster. Außerdem soll die Ausführung des Algorithmus verteilt ablaufen, d.h. eventbasiert anstelle von synchronisiert.

Alle Knoten sollen für die Cluster Setup-Phase zwei Arrays der Länge $2d$ besitzen, namentlich WINNER und SENDER, wobei an einer Position in WINNER die ID des gewinnenden Knotens einer Runde steht. Das WINNER-Array wird später zur Bestimmung des Clusterheads benutzt. Das SENDER-Array speichert die ID des Knotens, der die ID des Winners dieser Runde gesendet hat. Wenn ein Clusterhead ausgewählt wurde, wird dieses Array benutzt, um den kürzesten Pfad zum Clusterhead zu finden.

Die Clusterheadsuche wird über zwei Phasen gesteuert. Die erste Phase ist die Floodmax-Phase, in der jeder Knoten seinen WINNER-Wert an alle Nachbarn schickt. Nachdem die WINNER-Werte von allen Nachbarn empfangen wurden, sucht jeder Knoten den höchsten der empfangenen WINNER-Werte und speichert diesen. Dieser Prozess wird für d Runden wiederholt. Die zweite Phase heißt Floodmin und läuft essentiell ab wie die Floodmax-Phase, nur werden jetzt die kleinsten Werte verbreitet. Auch diese Phase wird d mal wiederholt. Nach den beiden Phasen überprüft jeder Knoten, ob er in der Floodmin-Phase seine eigene ID gesendet bekommen hat. Falls ja, definiert er sich selbst als Clusterhead und wartet bis zur Clusterzuweisung. Die restlichen Knoten betrachten die WINNER-Liste und suchen nach Node Pairs. Ein Node Pair ist eine ID, die bei einem Knoten sowohl in der Floodmax- und in der Floodmin-Phase mindestens einmal als WINNER aufgetreten ist. Aus allen Node Pairs, die ein Knoten in seiner WINNER-Liste findet, sucht er das niedrigste Node Pair aus und fügt sich diesem als Clustermember hinzu. Falls ein Knoten keine Node Pairs findet, fügt er sich dem Knoten mit der maximalen ID der Floodmax-Runde hinzu. In manchen Szenarien kann es passieren, dass ein Knoten zur Kommunikation mit seinem ausgewählten Clusterhead über einen anderen Clusterhead kommunizieren muss. In diesem Fall bestimmt der nähere Clusterhead, dass der kommunizierende Knoten seinem Cluster hinzugefügt wird und teilt das diesem Knoten mit. Nachdem die Clusterheadsuche beendet ist, sendet jeder Knoten seine Clusterheadwahl an alle seine Nachbarn. Ein Knoten wird zum Gatewayknoten, wenn in seiner Nachbarschaft mindestens ein Knoten ist, der zu einem anderen Cluster gehört.

Der Algorithmus besitzt selbst keine Maintenance-Phase. Allerdings ist die Wahrscheinlichkeit laut den Autoren hoch, dass bereits gewählte Clusterheads wiedergewählt werden, wenn der Algorithmus neu aufgerufen wird. In Simulationsergebnissen wurde gezeigt, dass die MaxMin-Heuristik in gewissen Punkten wie der Wiederwahl von Clusterheads oder Aufenthaltszeit von Membern in ihrem Cluster gleich gut oder sogar besser abschneidet als der LCA-Algorithmus (3.2.1), der LCA2 -Algorithmus oder die HD-Heuristik (3.2.2).

3.2.7. An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks

Der hier vorgestellte Algorithmus wurde 2003 von Bandyopadhyay und Coyle entworfen. Er arbeitet mit einer prädefinierten Anzahl P von Clustern. Der Algorithmus benötigt keine Synchronisation der Satelliten, kann also zu jeder Zeit ausgeführt werden.

Jeder Knoten definiert sich mit Wahrscheinlichkeit $\frac{1}{P}$ selbst als Clusterhead und gibt diese Information an alle erreichbaren Knoten weiter. Diese Clusterheads nennen wir *volunteer clusterheads*. Jeder Knoten, der kein Clusterhead ist und eine solche Nachricht erhält, schließt sich dem nächsten Clusterhead an. Jeder Knoten, der kein Clusterhead ist und keinem Cluster beigetreten ist, definiert sich nach einer gewissen Zeit t selbst als Clusterhead. Diese Clusterheads heißen *forced clusterheads*. Die Wartezeit t hängt von der Übertragungsreichweite der einzelnen Knoten und der Anzahl an maximal erlaubten Hops k ab.

Die Optimierungsparameter sind also die Anzahl der Cluster P, die Wartezeit t und die Anzahl der maximal erlaubten Hops k. Im Paper werden optimale Parameter errechnet. Im selben Paper wurde außerdem ein weiterer Algorithmus mit mehreren Layern vorgestellt, der allerdings für einen Layer genau den beschriebenen Algorithmus wiedergibt. Im Paper wird der Algorithmus mit dem Max-Min d-Cluster Formation-Algorithmus [50] verglichen. Simulationsresultate im Paper haben ergeben, dass der vorgestellte Algorithmus für verschiedene Simulationen weniger Energie verbraucht als [50]. Die hierbei benutzten Parameter sind vorher explizit berechnet worden.

Ein Vorteil dieses Algorithmus liegt darin, dass die gewünschte Clusteranzahl von vornherein bekannt ist und die tatsächliche Clusterzahl sich im Mittel auch um die gewünschte Clusterzahl bewegt. Die Bestimmung der Clusterheads erfordert außerdem keinen großen Rechenaufwand, sondern hängt nur von einer Zufallskonstante ab. Außerdem kann der Algorithmus auf jeder Zelle autonom aufgerufen werden. Die Optimierung des Algorithmus ist auch vorher bis zu einem gewissen Punkt berechenbar. Ein Nachteil dieses Algorithmus liegt darin, dass durch die zufällige Auswahl des Clusterheads nicht vorausgesagt werden kann, welcher Satellit nun tatsächlich zum Clusterhead wird. Auch ist nicht vollständig gesichert, dass die tatsächliche Anzahl der Cluster stabil um die beabsichtigte Clusterzahl liegt. Für den Algorithmus ist außerdem kein Maintenance-Teil angegeben, das Paper konzentriert sich nur auf den Konstruktionsteil.

3.2.8. A Clustering Algorithm applied to the Satellite Networks Management

Ein früher Clusteralgorithmus für Satellitennetzwerke wurde 2003 von Yueqiu Jiang, Yongbing Liu, yingyou Wen und Guangxing Wang in [51] vorgestellt. Das Paper betrachtet hauptsächlich militärische Zwecke als Anwendung von Satellitennetzwerken. Das Paper beschreibt einen Algorithmus ähnlich zum WCA-Algorithmus (3.2.5), fügt allerdings noch einen Token-Mechanismus hinzu: Hält ein Satellit ein Token, so be-

sitzt er eine höhere Priorität, Clusterhead zu sein, als Satelliten ohne Token. Haben zwei benachbarte Satelliten jeweils ein Token, so wird der Satellit Clusterhead, der als letztes ein Token zugewiesen bekommen hat. Falls sich keine Satelliten mit einem Token in der Nachbarschaft befinden, so entscheidet das höchste Gewicht die Wahl des Clusterheads. Das Gewicht wird wie in 3.2.5 als kombiniertes Gewicht aus verschiedenen Größen berechnet, namentlich aus einer standardisierten Repräsentation von Kommunikations- und Verarbeitungsressourcen[8] c, der maximalen Anzahl an Hops r eines Clustermembers zum Clusterhead, einem Sicherheitsfaktor s und der maximal angestrebten Lebensdauer t des Satelliten.

Im Paper sind weiterhin Anweisungen angegeben, wie die Clusterbeibehaltung verfahren soll. Das Protokoll ist ereignisorientiert, eine Aktion wird also nur dann aufgerufen, wenn Satelliten ihr Cluster verlassen, ausfallen, einem neuen Cluster beitreten möchten oder Token neu vergeben werden. Ein interessanter Teil dieses Algorithmus besteht darin, dass Satelliten einem Cluster nur dann beitreten können, wenn ihr privater Key bekannt ist, um so Einbindungen von fremden oder feindlichen Satelliten zu unterbinden. Nicht definiert hingegen ist, was mit einem Satelliten passiert, der sein Cluster verlässt.

Dieser Algorithmus ist zwar direkt anpassbar, indem Satelliten genau dann Token bekommen, wenn sie Kontakt zu einer Bodenstation besitzen, allerdings ist nicht näher definiert, was im Falle eines Satellitenaustritts aus einem Cluster mit dem ausgetretenem Satelliten passieren soll. Außerdem wurden weder Vergleiche mit anderen Algorithmen noch Simulationsergebnisse vorgelegt und nicht näher definiert, was mit den Größen, aus denen das kombinierte Gewicht berechnet werden soll, gemeint ist.

3.2.9. A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks

Basu, Khan und Little verfolgen in ihrem 2001 veröffentlichten Paper [8] einen fundamental anderen Ansatz, den manche Algorithmen nur ansatzweise oder oftmals gar nicht betrachten, um Cluster stabil zu halten. Das Hauptaugenmerk des im Paper vorgestellten Clusteralgorithmus MOBIC liegt auf der Mobilität der Knoten im Netzwerk. Dazu werden verschiedene Mobilitätsmetriken betrachtet und eine eigene Metrik vorgestellt, die auf MANETs ausgerichtet ist. Die vorgestellte relative Mobilitätsmetrik lautet wie folgt:

$$M_Y^{rel}(X) = 10 \log_{10} \frac{RxPr_{X \to Y}^{new}}{RxPr_{X \to Y}^{old}} \tag{3.1}$$

Hierbei steht $RxPr$ für die Signalstärke eines empfangenen Signals, entsprechend $RxPr_{X \to Y}$ für die Signalstärke eines von Y empfangenen Signals von X. Die Metrik vergleicht effektiv also zwei Signalstärken, um zu erkennen, ob sich ein fremder Knoten her- (positiver Wert) oder wegbewegt (negativer Wert). Dazu werden zwei konsekutive HELLO-Nachrichten von allen Satelliten gesendet, um zwei Signalstärken zu empfangen. Die lokale Mobilitätsmetrik M_Y bildet sich nun aus der Varianz der relativen Mobilitätsmetriken $M_Y = var_0\left(M_Y^{rel}(X_1), \ldots, M_Y^{rel}(X_n)\right)$ mit X_i Nachbar

[8] Es wurde leider nicht näher definiert, was damit gemeint sein soll

von Y. var_0 bezeichnet die Varianz gegenüber 0, obiger Wert für M_Y ließe sich also auch als $M_Y = E\left[\left(M_Y^{rel}\right)^2\right]$ schreiben.

Der Algorithmus selbst läuft nun wie folgt ab. Jeder Knoten sendet zu Beginn zwei konsekutive HELLO-Nachrichten ab. Jeder empfangende Knoten berechnet nun alle relativen Mobilitätsmetriken für die Knoten, von denen er zwei HELLO-Nachrichten empfangen hat. Sollte nur eine HELLO-Nachricht empfangen worden sein, so wird der Knoten ignoriert. Nun sendet jeder Knoten in regelmäßigen Abständen eine HELLO oder I'M ALIVE-Nachricht mitsamt dem lokalen M. Der Knoten in der Nachbarschaft, der die niedrigste lokale Mobilitätsmetrik besitzt, wird nun Clusterhead. Knoten in Reichweite von zwei Clusterheads werden Gateway-Knoten. Im unwahrscheinlichen Fall, dass 2 benachbarte Knoten den gleichen Wert für M besitzen, entscheidet die niedrigere ID. Wenn ein Member-Knoten mit einer niedrigen Mobilität in die Reichweite eines Clusterheads mit einer höheren Mobilität kommt, wird kein Reclustering eingeleitet. Falls zwei Clusterheads in gegenseitige Reichweite gelangen, so wird für einen bestimmten Zeitraum kein Reclustering durchgeführt, um durch kurzzeitigen Kontakt nicht unnötig Energie zu verbrauchen. Sind beide Clusterheads nach diesem Zeitraum immer noch in Reichweite, so wird trotzdem neu geclustert.

In Simulationen wurde MOBIC mit der LCC-Variante des LowestID-Algorithmus verglichen und erzielte 33% bessere Ergebnisse in Bezug auf Clusterhead-Wechsel, sofern die Übertragungsreichweite groß genug gewählt wird. Bei kleinen Übertragungsreichweiten liefert LCC bessere Ergebnisse.

3.2.10. A Robust Clustering Algorithm for Mobile Ad Hoc Networks

Der 2008 von Zhaowen Xing, Le Gruenwald und K. K. Phang veröffentlichte PMW-Algorithmus versucht, die drei Hauptmetriken Power, Mobility und Workload in einen Algorithmus zu vereinen [52]. Es werden verschiedene Algorithmen erwähnt, die jeweils nur eine der drei Metriken beschreiben: Die Power-Metrik wird im Stable Cluster Algorithm [53] von Sheu und Wang behandelt. Dort definiert sich das Auswahlkriterium für Clusterheads über die Restenergie der Nachbarn insofern, dass ein Knoten bei Unterschreiten einer Energiegrenze als Bottleneck gilt. Ein Knoten wird dann Clusterhead, wenn sich viele Bottlenecks in seiner Umgebung befinden. Dieser Algorithmus besitzt allerdings keine Maintenance-Phase. An diesem Algorithmus wird kritisiert, dass durch hochmobile Knoten viele Reclusterings aufgerufen werden können. Die Mobility-Metrik wird unter anderem in WCA (siehe 3.2.5) und MOBIC (siehe 3.2.9) benutzt. Der Kritikpunkt hier liegt darin, dass auch Knoten mit einem niedrigen Energiestand als Clusterheads gewählt werden können. Diese fallen dann um so schneller aus, was wiederum Reclusterings zur Folge hat. Außerdem wurden diverse Kombinationsalgorithmen betrachtet, die mehrere Größen in Betracht ziehen, wie z.B. WCA und DMAC (siehe 3.2.4). Allerdings treffen auch hier ähnliche Kritikpunkte zu, da eine oder mehrere Größen nicht betrachtet werden, die häufige Reclusterings zur Folge haben.

Der PMW-Algorithmus betrachtet also die Größen Power, Mobility und Workload. Die Power des Knotens j wird hierbei durch die Größe RP_j (Remaining Power)

dargestellt, die die Restenergie in Prozent als eine Zahl im Intervall $[0; 1]$ beschreibt. Der Mobility-Faktor wird durch MP_j ausgedrückt. MP_j steht hierbei für Mobility Prediction und wird folgendermaßen berechnet: Zuerst wird für jeden Knoten j wie in MOBIC (3.2.9) der Wert $RM_{ij} = \frac{rss_{ij_1}}{rss_{ij_2}}$ berechnet, was die relative Mobilität in Bezug auf die Nachbarknoten i beschreibt. Abweichend zu MOBIC wird hier kein Logarithmus angewandt, sondern nur betrachtet, ob der errechnete Wert größer oder kleiner als 1 ist. MP_j ist in diesem Fall die Standardabweichung der Zufallsvariable RM_{ij}, wobei $E[RM_{ij}] = 1$ angenommen wird. Es gilt also

$$MP_j = \sqrt{\frac{\sum_{i=1}^{n} (RM_{ij} - E[RM_{ij}])^2}{n}} \tag{3.2}$$

Als dritte Größe wird noch die Power Decreasing Rate PDR_j eingeführt, die indirekt den Workload repräsentiert: Je mehr der Knoten belastet wird, desto schneller verbraucht er Energie. Für die Power Decreasing Rate gilt $PDR_j = \frac{RP_{j_1} - RP_{j_2}}{t_1 - t_2}$, wobei RP_{j_1} die Restenergie zum Zeitpunkt t_1 darstellt, entsprechend RP_{j_2}. Das letztendliche Gewicht W_j ergibt sich nun als

$$W_j = f_1 \cdot \exp(-MP_j) + f_2 \cdot RP_j + \cdot f_3 \cdot \exp(-PDR_j) \tag{3.3}$$

Wie in WCA sind die f_i Gewichtungsfaktoren für die einzelnen Größen mit $f_1 + f_2 + f_3 = 1$, die je nach Anwendung verteilt werden.

Der Ablauf des Algorithmus ist sehr ähnlich zum Ablauf von MOBIC: Jeder Knoten sendet beständig eine HELLO-Nachricht. Nachdem er 2 HELLO-Nachrichten von einem Knoten empfangen hat, berechnet er sein Gewicht und sendet das Gewicht an alle Nachbarn. Nachdem alle Knoten das Gewicht von jedem Knoten in der Nachbarschaft empfangen haben, definieren sich Knoten mit dem höchsten Gewicht als Clusterheads und informieren alle Nachbarknoten darüber, die sich nun dem Clusterhead in ihrer Nachbarschaft mit dem größten Gewicht anschließen.

Die Maintenance-Phase besteht darin, möglichst wenige Reclusterings durchzuführen. Falls ein Knoten den Kontakt zu seinem Clusterhead verliert, wartet er 3 HELLO-Broadcasts ab. Hat sich der Clusterhead nach dem dritten Broadcast nicht gemeldet, versucht der Knoten, sich zu dem Clusterhead in der Nachbarschaft zu verbinden, der das höchste Gewicht besitzt. Falls kein Clusterhead verfügbar ist, deklariert der Knoten sich selbst als Clusterhead. Falls zwei Clusterheads direkte Nachbarn werden, wird wie bei MOBIC eine geraume Zeit gewartet. Falls die Clusterheads danach immer noch direkte Nachbarn sind, ordnet sich der Clusterhead mit dem niedrigeren Gewicht dem mit dem größeren Gewicht unter. Die Member des leichteren Clusterhead verbinden sich nun mit dem nächsten Clusterhead mit dem höchsten Gewicht. Falls das nicht möglich ist, deklarieren sie sich selbst als Clusterhead.

Ein interessanter Punkt in der Maintenance Phase ist, dass Clusterheads ihre Rolle freiwillig aufgeben können, falls ihre Energie unter ein gewisses Limit fällt. Der Clusterhead sucht dann rechtzeitig nach einem Nachbarn, der keinen weiteren Clusterhead als Nachbarn besitzt, aber eine höhere Energiestand als der Clusterhead selbst hat. Er stößt dann ein Reclustering an, das allerdings auf das lokale Cluster begrenzt ist.

Der alte Clusterhead fügt sich dem neuen Clusterhead nun als Member hinzu. Knoten, die keinen Kontakt zum neuen Clusterhead aufbauen können, suchen sich einen anderen Clusterhead oder deklarieren sich selbst als Clusterhead.

Dieser Algorithmus ist sehr umfangreich und detailliert beschrieben. Im weiteren Verlauf des Papers wurde der PMW-Algorithmus mit MOBIC verglichen. Die Vergleichspunkte waren die Lebensdauer des Netzwerks, also das Zeitintervall vom Beginn des Netzwerks bis zum ersten Ausfall eines Knotens aufgrund von Energieverlust, die Rate, mit der Clusterheads wechseln und Neuzuweisungen pro Sekunde. Der PMW-Algorithmus schnitt in allen drei Kriterien besser ab als MOBIC, wenn meistens auch nur sehr leicht.

3.3. Begründung der Algorithmenauswahl

Es wurden drei Algorithmen ausgewählt, die implementiert und auf mehrere Szenarien mit Satellitennetzwerken angewandt wurden. Details zur Implementierung finden sich in Kapitel 5, die Bewertungskriterien ebenfalls in diesem Kapitel und die tatsächlichen Ergebnisse in Kapitel 6. Eine Diskussion der Ergebnisse findet anschließend in 7 statt.

Der erste ausgewählte Algorithmus ist der WCA-Algorithmus (3.2.5). Dieser Algorithmus stammt von der Idee her von LID ab, wurde aber (mit einem Zwischenschritt über DLA-DC) verallgemeinert, dass nicht mehr feste IDs ausschlaggebend für die Wahl zum Clusterhead sind, sondern dynamische Gewichte. Außerdem wurde eine stabile Cluster maintenance-Phase angegeben (siehe Abschnitt 2.3). Ein weiterer Vorteil der dynamischen Gewichte liegt darin, dass es keine Probleme bereitet, eigene Größen einzubinden. Dadurch wird dieser Algorithmus sehr gut anpassbar an verschiedenste Szenarien.

Der zweite Algorithmus, der implementiert wurde, ist der BC-Algorithmus (3.2.7). Die Clusterheadwahl erfolgt hier zufällig, was ungünstige, aber auch optimale Ergebnisse liefern kann. Ein weiterer interessanter Gesichtspunkt liegt darin, dass frei gewählt werden kann, wieviele Cluster bei der Erzeugung angestrebt werden sollen. Die Auswahl dieses Algorithmus wurde deswegen getroffen, da er eine Verbesserung des LCA-Algorithmus darstellt, allerdings auf die Gateway-Knoten verzichtet. Durch seine Abstraktheit ist auch dieser Algorithmus sehr einfach an unterschiedliche Szenarien anpassbar.

Der PMW-Algorithmus wurde schließlich als dritter Algorithmus ausgewählt. PMW ist zwar eine Spezialisierung des WCA-Algorithmus, benutzt aber drei Größen, die auch für Satellitennetzwerke sehr interessant und wichtig sind. Zum einen ist für Picosatelliten die verbleibende Energie und deren Abnahmegeschwindigkeit immens überlebenswichtig, zum anderen ist gerade die Mobilitätsmetrik, die hier benutzt wird, sehr interessant und auch hilfreich, den Energieverbrauch der Satelliten zu vermindern, indem nur Satelliten ausgewählt werden, deren Cluster voraussichtlich eine höhere Stabilität besitzen als die anderer potentieller Clusterheads.

Als vierter Algorithmus wurde eine Heuristik implementiert, die einen neuen Ansatz erprobt. Die Heuristik soll mit NH bezeichnet werden und wählt Clusterheads zufällig aus, begrenzt aber das erzeugte Cluster in seiner Größe. Außerdem werden

nur die nächsten Satelliten in ein Cluster aufgenommen. Eine genaue Beschreibung der Heuristik findet sich im Abschnitt 5.3.4. Die Heuristik wurde deswegen entworfen, da bis jetzt kein anderer Algorithmus eine ähnliche Idee verwirklicht. Auf eng gepackten Netzwerken führt die von den anderen Algorithmen benutzte Herangehensweise, alle freien Nachbarn in ein Cluster aufzunehmen, zu extrem großen Clustern und starker Überlastung einzelner Knoten. Diese Probleme sollen mit einer statischen Begrenzung der Clustergröße umgangen werden.

Es wurde vereinbart, nur Algorithmen zu betrachten, die nur die direkte Nachbarschaft eines Satelliten betrachten. Ausgeschlossen wurden also explizit Algorithmen, die Kommunikation über mehrere Hops ermöglichen sowie Algorithmen, die mit Gateway-Satelliten arbeiten.

Kapitel 4.

Theoretisches Modell

In diesem Kapitel sollen zuerst einige grundlegende Annahmen für das Simulationsmodell getroffen werden, um dieses Modell in seinen Grundzügen zu definieren und zu beschreiben. Im zweiten Abschnitt werden die unterschiedlichen Satellitenszenarien und die Bodenstationskonfiguration beschrieben und erläutert. Der dritte Abschnitt befasst sich mit den beiden Testzeiträumen, in denen die Algorithmen simuliert werden. Schließlich beschreibt der letzte Abschnitt die verschiedenen Vergleichskriterien, die aufgestellt wurden, um die simulierten Algorithmen zu vergleichen und zu bewerten.

4.1. Annahmen

Für das Simulationsmodell sollen verschiedene Annahmen getroffen werden, um eine Basis für die Simulationen zu schaffen. Diese Annahmen werden erklärt und begründet. Zuerst wird beschrieben, wie ein Satellitennetzwerk auf Graphen abgebildet werden kann sowie die Eigenschaften der Satellitenobjekte, die für die Simulationen von Belang sind. Anschließend folgt eine Erklärung über Annahmen zur Bandbreite der Satelliten. Zuletzt werden Grundlagen der Orbiterzeugung der Satellitenobjekte betrachtet.

4.1.1. Abbildung des Satellitennetzwerks auf Graphen

Prinzipiell besitzt jeder Satellit die genauen Informationen über die Positionen der anderen Satelliten und somit auch die Distanz zwischen ihm und seinen Nachbarn. Zwischen einem Satelliten und einem Nachbarn besteht per Definition stets Kontakt (siehe Abschnitt 2.2), was in diesem Zusammenhang heißt, dass es prinzipiell möglich wäre, Daten zu senden. Wie diese Kommunikation letztendlich stattfindet, wird nicht beachtet. Durch programmiertechnische Eingriffe kann trotzdem für jeden Algorithmus durchgesetzt werden, dass beispielsweise ein Satellit nur die Informationen über die Satelliten in seiner Nachbarschaft erhält und keinerlei Information über die restlichen Satelliten, zu denen er keinen Kontakt besitzt, vermittelt wird. Die Distanzberechnung zwischen einzelnen Satelliten und Bodenstationen erfolgt intern im STK. Zur Erzeugung dieser Distanzen wurden im STK zwischen allen möglichen 2-Tupeln von Satelliten und Bodenstationen sogenannte *Chains* erzeugt. Über diese Chains ist es möglich, in einem eigens definierten *Report* deren Länge, also die Distanz zwischen den Tupelkomponenten, zu berechnen und über das Connect-Framework an

das Simulations-Framework weiterzugeben. Siehe hierzu auch Abschnitt 5.1.2. Zur Speicherung dieser Distanzen für einen bestimmten Zeitpunkt t wird eine Distanzmatrix $A_{total}(t)$ benutzt, ein Spezialfall der Adjazenzmatrix, bei der in den einzelnen Zellen die Distanz zwischen zwei Satelliten oder einem Satellit und einer Bodenstation steht (siehe Abschnitt 2.2.1). Diese Distanzmatrix $A_{total}(t)$ ist folgendermaßen aufgebaut:

$$A_{total}(t) = \left[\begin{array}{c|c} A_{Sat \to Sat}(t) & A_{Fac \to Sat}(t) \\ \hline A_{Sat \to Fac}(t) & 0 \end{array}\right] \tag{4.1}$$

Die Teilmatrizen $A_{Sat \to Sat}(t)$ und $A_{Sat \to Fac}(t)$ bezeichnen respektive die Distanzmatrizen nur von Satellit zu Satellit und von Satellit nur zu Bodenstationen. Wie in Abschnitt 2.2.1 bereits erwähnt, ist $A_{total}(t)$ symmetrisch, also gilt natürlich auch $A_{Sat \to Fac}(t) = A_{Fac \to Sat}(t)^T = A_{Sat \to Fac}(t)$, was auch intuitiv dadurch klar ist, dass die Distanz zwischen einer Bodenstation und einem Satelliten die gleiche ist wie zwischen dem Satelliten und der Bodenstation. Diese Matrix wird zu jedem Messzeitpunkt neu erzeugt. Alternativ formuliert ergibt sich nun, dass ein Satellit i also dann Kontakt zu einem anderen Satelliten j besitzt , wenn in der Distanzmatrix für die entsprechende Zelle $a_{ij} = a_{ji} \neq 0$ gilt. Dies korrespondiert mit der Anzeige einer Chain in der graphischen Simulationsdarstellung im STK, die als Linie zwischen zwei Satelliten eingezeichnet wird.

4.1.2. Satellitenkommunikation

Für die Satellitenkommunikation werden außerdem folgende Annahmen getroffen: Die maximale Bandbreite für die Kommunikation zwischen Satelliten und Bodenstation wird wie im UWE-Projekt auf $1200 bit/s = 150 byte/s$ FSK beschränkt [54]. Die Kommunkationsdauer zwischen Satellit und Bodenstation für einen Überflug wurde auf der einen Seite im Mittel auf 8-10 Minuten mithilfe des STK berechnet, auf der anderen Seite wird dieser Wert gestützt durch [55], wo die Überflugsdauer des MOST-Satelliten berechnet und gemessen wurde. Für die Berechnung des Datendurchsatzes wurden 10 Minuten Kontaktdauer angenommen, da es mit diskreten Punktaufnahmen[1], nicht möglich ist, eine tatsächliche Kontaktdauer zu messen.

Es wird zusätzlich angenommen, dass alle Satelliten im Netzwerk gemeinsam an einer beliebigen Mission arbeiten. Jeder Satellit führt gewisse Messungen durch und gibt diese Informationen an den Clusterhead weiter. Damit ein Mindestmaß an Missionsinformationen gesendet werden kann, wird die Mindestbandbreite für Missionsnachrichten von unten auf mindestens $150 bit/s$ beschränkt, um zu garantieren, dass diese Nachrichten auch vollständig gesendet werden können. Die Maximalbandbreite wird allerdings auf $400 bit/s$ beschränkt, damit neben Missionsnachrichten außerdem genügend Bandbreite vorhanden ist, um die Clusterstruktur aufrecht zu erhalten. Diese Werte gelten also nur für die Weitergabe von gesammelten Informationen, nicht

[1] Also Betrachtungen einer Momentaufnahme eines einzelnen Zeitpunktes statt eines Zeitintervalls

für die Kommunikationskoordination. Im tatsächlichen Anwendungsfall sollten diese Werte auf jeden Fall explizit berechnet werden, da hier nur von einer hypothetischen Mission ausgegangen wurde.

4.1.3. Orbiterzeugung

Bei der Satellitenerzeugung wurden zwei Propagators eingesetzt: Falls die Satelliten über ihre TLE-Daten eingebunden wurden, wurde ein SGP4-Propagator benutzt. Falls zufällige Keplerelemente zur Bahnberechnung eingesetzt wurden, wurde ein J2-Perturbator benutzt. Propagators sind Lösungskurven der durch die Bahnelemente gegebenen Differentialgleichungen, die verschiedene Orbitstörungen einberechnen, wie zum Beispiel die Erdanziehungskraft oder die Abbremsung eines Satelliten durch athmosphärische Reibung. Ein Perturbator rechnet ebenfalls Störungen zu einer Satellitenbahn hinzu, die sich aus einer Lösung der Keplerelemente ergibt. Für weitere Informationen siehe Abschnitt 2.5.

In den Tabellen 4.1 und 4.2 werden die Funktionen *random_int*(int k) und *random_double*(double d) verwendet, durch die gleichverteilte Zufallszahlen erzeugt werden. *random_int*(int k) liefert eine ganzzahlige Zufallszahl zwischen 0 und k, *random_double*(double d) verhält sich analog, nur wird eine Gleitkommazahl zurückgeliefert.

4.2. Orbital- und Bodenstationsszenarien

Um die Clusteringalgorithmen zu testen, wurden verschiedene Satellitenszenarien erstellt, die verschiedene Situationen im Erdorbit simulieren sollen. Außerdem wurde eine Menge an Bodenstationen ausgewählt, um die Fähigkeit der Clusteralgorithmen zur passenden Aufteilung der Satellitenmenge zu testen.

Satellitenszenario 1 - Ähnliche Orbits, 13 Satelliten Das erste Szenario besteht aus 13 real existierenden Satelliten, die auf 3 verschiedenen Orbitbahnen liegen, siehe auch Abbildung 4.1. Diese Satelliten sind entsprechend ausgewählt, um die Algorithmen an einem Szenario mit ähnlichen, aber nicht gleichen Orbits zu testen. Für die Orbitdaten der verwendeten Satelliten siehe Anhang A.

Satellitenszenario 2 - Annähernd gleiche Orbits, 7 Satelliten Das zweite Szenario besteht aus 7 Satelliten, die von einer einzigen Rakete in den Orbit gebracht wurden. Die Orbitbahnen der Satelliten liegen also sehr nah beieinander. Eine Beispielkonfiguration ist in Abbildung 4.2 zu sehen. Die Satelliten wurden so ausgewählt, dass mit wenigen Satelliten das vierte Szenario angenähert. Für die Orbitdaten der verwendeten Satelliten siehe auch hier Anhang A. Die Satelliten in diesem Szenario wurden gemeinsam am 17. April 2007 gestartet.

Satellitenszenario 3 - Zufällige, annähernd gleiche Orbits, 50 Satelliten direkt nach Abwurf Um das QB50-Szenario (siehe Abschnitt 3.1.2) mit 50 Pico- und Nanosatelliten vom Prinzip her zu simulieren, wurden 50 Satellitenorbits zufällig erzeugt

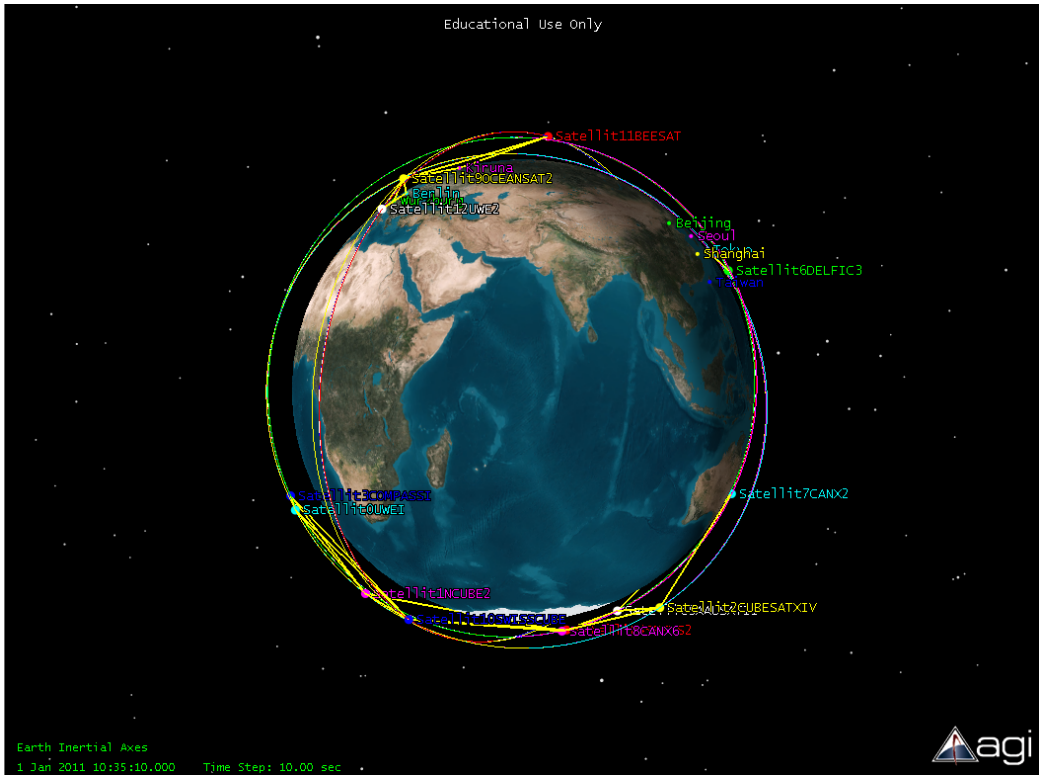


Abb. 4.1.: Beispielkonfiguration von 13 real existierenden Satelliten auf 3 Orbits

(siehe Abbildung 4.3 für eine Beispielkonfiguration). Das erste Zufallsszenario soll die Situation der Satelliten kurz nach dem Abwurf aus der Rakete simulieren. Zu diesem Zeitpunkt liegen alle Satelliten sehr nahe beieinander und haben mit allen anderen Satelliten direkten Kontakt.

Die Problemstellung liegt hierbei darin, dass auf der einen Seite alle Satelliten miteinander direkten Kontakt haben, also zwingend aufgeteilt werden müssen, um Interferenzen zu vermeiden und einzelne Clusterheads nicht zu überlasten. Auf der anderen Seite sind die Orbitdaten der einzelnen Satelliten noch nicht bekannt, also kann man die absoluten Positionen der einzelnen Satelliten noch nicht berechnen und daraus eine optimale Clusteraufteilung ableiten.

Die Orbits wurden mit den Parametern und Abweichungen aus Tabelle 4.1 erzeugt. Durch die Variation in der großen Halbachse zusammen mit der Exzentrizität wird die Bahnhöhe und -form geändert Die Bahnform ist hier beinahe kreisförmig, da die Exzentrizität immer nahe bei 0 liegt. Die Variation in der Inklination neigt manche Bahnen stärker als andere, bleibt aber in der Abweichung noch gering genug, um einen sonnensynchronen Orbit zu erzeugen. Die mittlere Anomalie bewegt sich zwischen 3 und 7 Grad, was alle erzeugten Satelliten relativ nahe beieinander liegen lässt. Das Argument des Perigäums und die Rektaszension bleiben fest, da zu viele Variationen in den Lageparametern die Lage der Bahnen zu komplex verändern würden.

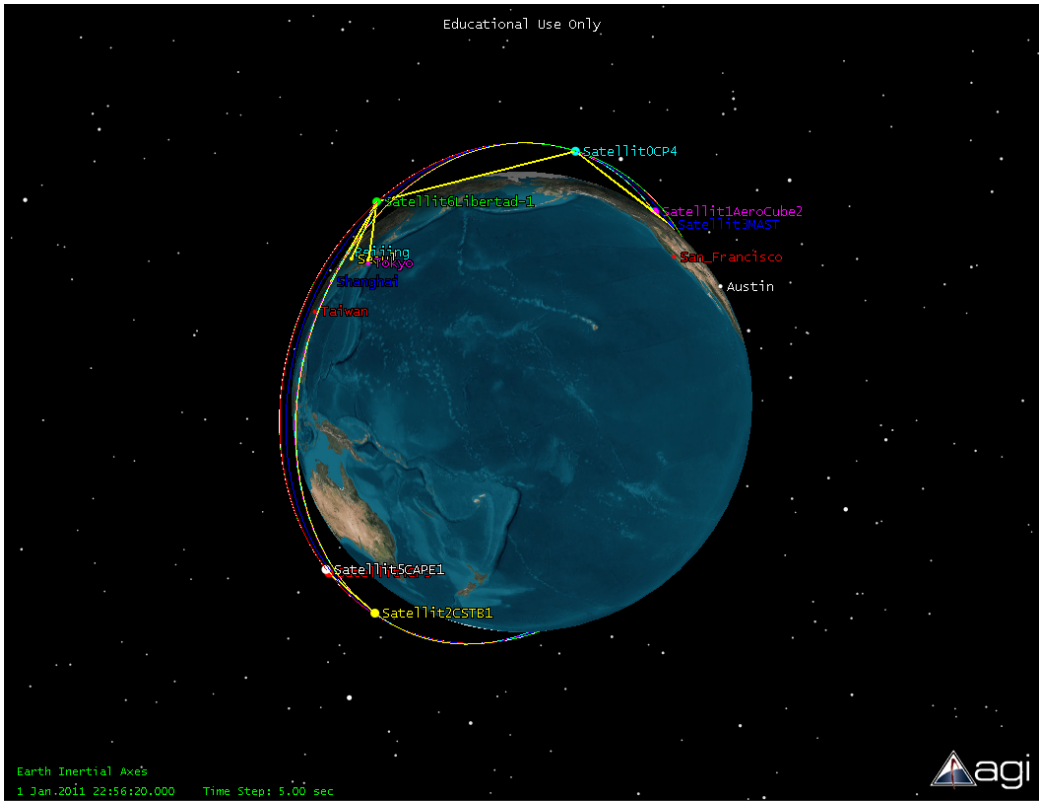


Abb. 4.2.: Beispielkonfiguration von 7 real existierenden Satelliten auf sehr ähnlichen Orbits

Parameter	**Mittelwert**	**Abweichung**
Länge der großen Halbachse	$6778,14\text{km}$	$random\_int(1000m)$
Exzentrizität	0	$0.002 \cdot random\_double(1)$
Inklination	$98.5°$	$random\_double(1.5)$
Argument des Perigäums	5	0
Rektaszension des aufsteigenden Knotens	0	0
Mittlere Anomalie	5	$random\_double(4) - 2$

Tab. 4.1.: Orbitelemente der Satelliten aus Szenario 3

Satellitenszenario 4 - Zufällige, annähernd gleiche Orbits, 50 Satelliten auf einer "Perlenschnur" In diesem Szenario soll die Situation des QB50-Szenarios nach einer längeren Zeitspanne nachgestellt werden, wenn sich die zusammenhängende Menge von Satelliten voneinander entfernt und sich auf der gemeinsamen Orbitbahn wie eine Perlenschnur verteilt hat. Dieser Situation wird durch die Veränderung der Mittleren Anomalie Rechnung getragen, bei der auf jeden Satelliten mit Nummer i nochmals $5 \cdot i$ Grad aufaddiert werden, bevor ein zufälliger Abweichungswert miteinberechnet wird.

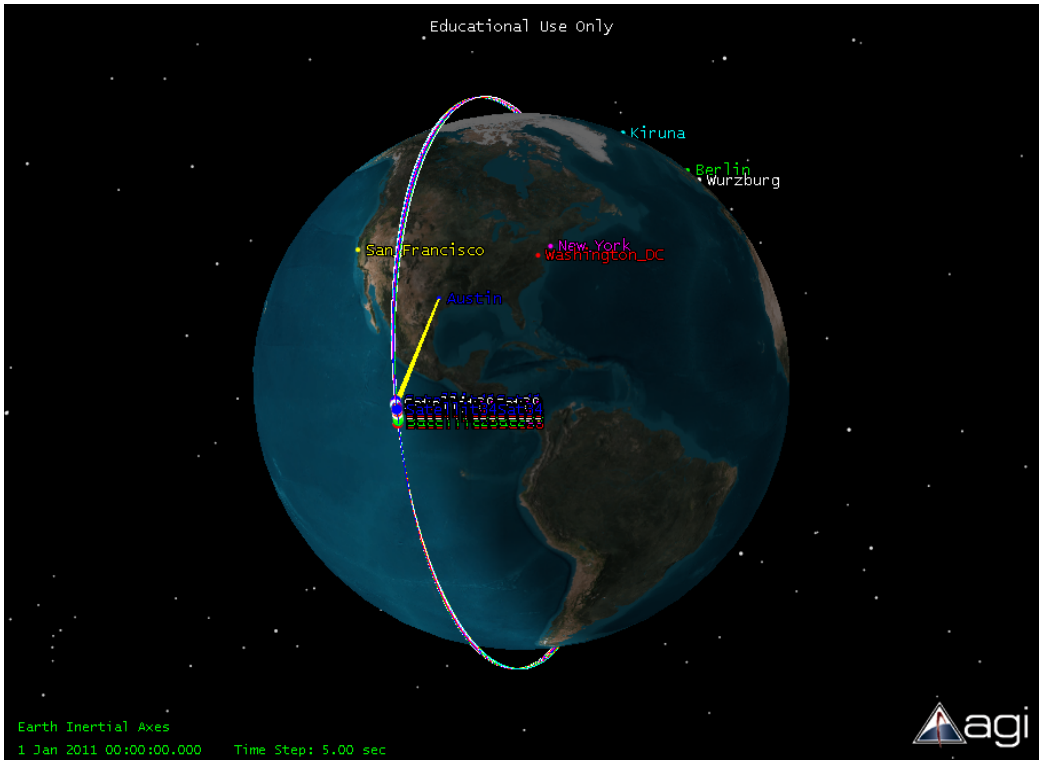


Abb. 4.3.: Beispielkonfiguration von 50 Satelliten direkt nach dem Auswurf. Alle Satelliten befinden sich sehr nahe beieinander.

Die Orbits wurden mit den Parametern und Abweichungen aus Tabelle 4.2 erzeugt. Die Erzeugungsparameter sind bis auf die eben beschriebene Variation in der Mittleren Anomalie identisch denen im Satellitenszenario 3. Eine Beispielkonfiguration ist in Abbildung 4.4 zu sehen.

Parameter	**Mittelwert**	**Abweichung**
Länge der großen Halbachse	$6778,14\text{km}$	$random\_int(1000m)$
Exzentrizität	0	$0.002 \cdot random\_double(1)$
Inklination	$98.5°$	$random\_double(1.5)$
Argument des Perigäums	5	0
Rektaszension des aufsteigenden Knotens	0	0
Mittlere Anomalie	$5 + 5i$	$random\_double(4) - 2$

Tab. 4.2.: Orbitelemente der Satelliten aus Szenario 4

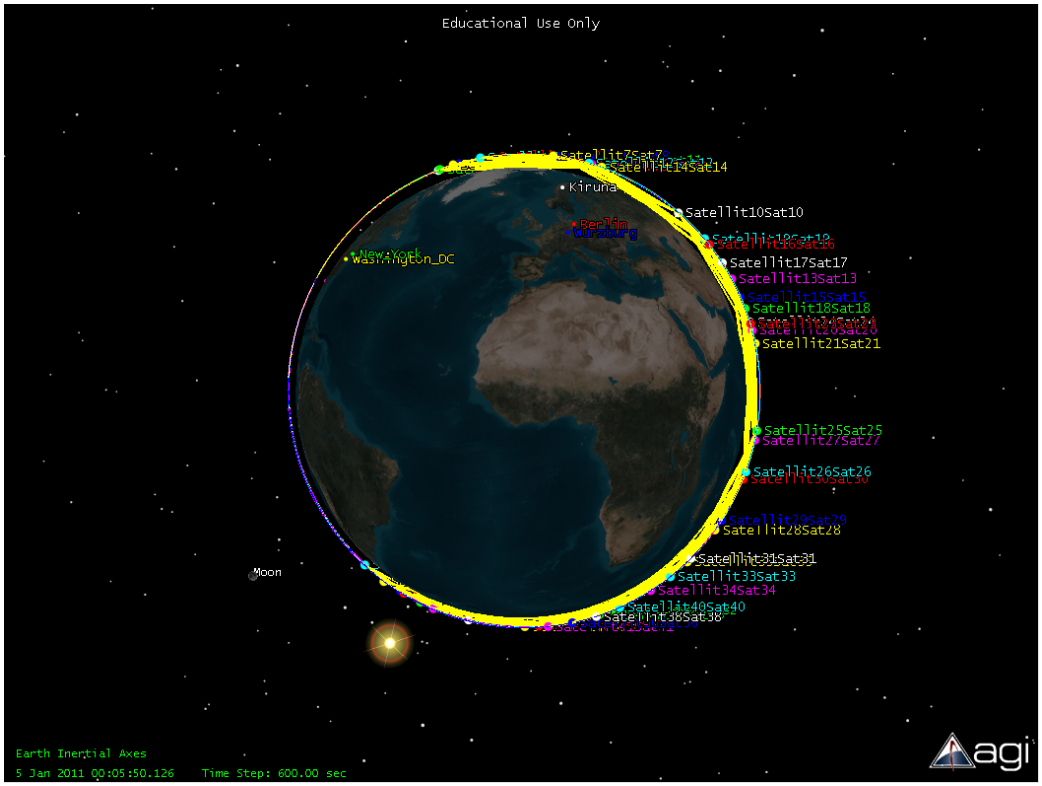


Abb. 4.4.: Beispielkonfiguration von 50 Satelliten geraume Zeit nach dem Auswurf. Die Satelliten haben sich über zwei Drittel des Orbits verteilt.

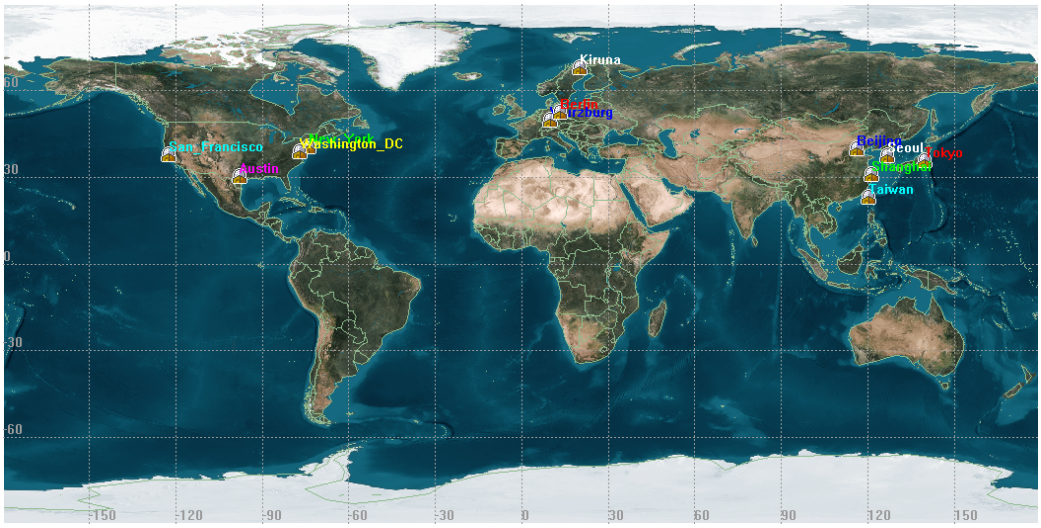


Abb. 4.5.: Positionen der Bodenstationen auf der Erdkugel in 2D-Ansicht

Bodenstationen Um die Fähigkeit der Clusteringalgorithmen zur Aufteilung von größeren Satellitengruppen zu testen, wurden drei Punkte auf der Erdkugel mit jeweils mehreren Bodenstationen besetzt. In Tabelle 4.3 sind die europäischen Boden-

stationen Würzburg, Kiruna und Berlin mit ihren Breiten- und Längengraden angegeben, in 4.4 liegen die Koordinaten der US-amerikanischen Bodenstationen New York, Austin und Washington, D.C. Schließlich sind in 4.5 die Koordinaten der ostasiatischen Bodenstationen Beijing, Tokyo, Seoul, Shanghai und Taiwan aufgezählt. Die Positionen der Bodenstationen sind in Abbildung 4.5 grafisch dargestellt.

Die Bodenstationsmengen und -anzahlen wurden so gewählt, dass die einzelnen Bodenstationen nah bis sehr nah beieinander liegen und so testen, ob die Algorithmen fähig sind, die Satellitenmenge entsprechend aufzuteilen. Die Auswahl an Bodenstationen ist für jedes Szenario gleich, um so eine gemeinsame Vergleichsgrundlage zu liefern.

Die Längengrade sind nach östlicher Breite angegeben.

Name	**Breitengrad**	**Längengrad**
Würzburg	49.7878	9.9361
Berlin	52.5167	13.4
Kiruna	67.85	20.2167

Tab. 4.3.: Gruppe 1 - Europa

Name	**Breitengrad**	**Längengrad**
New York	40.7143	-74.006
San Francisco	37.7749	-122.4194
Austin	30.2672	-97.7431
Washington, D.C.	38.8951	-77.0364

Tab. 4.4.: Gruppe 2 - USA

Name	**Breitengrad**	**Längengrad**
Beijing	39.9829	116.3883
Tokyo	35.6850	139.7514
Seoul	37.5985	126.9783
Shanghai	31.1094	121.3681
Taiwan	23.2092	120.1872

Tab. 4.5.: Gruppe 2 - Ostasien

4.3. Testzeiträume

Die Tests wurden über verschiedene Zeiträume und -intervalle geführt. Die Langzeittests erstrecken sich über den Zeitraum von zwei Monaten, um die Langzeitentwicklung von Satellitenbewegungen in die Clusterbildung miteinzubeziehen. Dieser Zeitraum wurde deswegen gewählt, weil das QB50-Szenario selbst nicht länger als 3 Monate dauern soll. Hierbei wird alle 8 Stunden eine Momentaufnahme der aktuellen

Satellitenpositionen und -kontakte betrachtet. Die Kurzzeittests erstrecken sich über einen Tag, wobei bei hier alle 8 Minuten Informationen über die Netzwerktopologie, also die räumliche Lage und der Kontakt der Satelliten zueinander, abgerufen werden. Mit den Kurzzeittests soll beobachtet werden, wie sich Änderungen innerhalb eines kurzen Zeitraums auf das Algorithmenverhalten auswirken. Die Menge aller Messzeitpunkte soll mit $\mathbb{T} = \{t_1, t_2, \ldots, t_{180}\}$ bezeichnet werden.

Testzeitraum 1: Kurz, fein gekörnt Der erste Testzeitraum ist insbesondere für die Beobachtung des Clusterverhaltens und die grobe Verbindungsdauer eines Clusters zu einer Bodenstation gedacht. Der Zeitraum umfasst einen Tag und 180 Beobachtungszeitpunkte, die Clusterkonfiguration wird also alle acht Minuten überprüft. Für die Menge aller Messzeitpunkte $\mathbb{T} = \{t_1, t_2, \ldots, t_{180}\}$ gilt also hier $\mathbb{T}_1 = \{0min, 8min, \ldots, 1440min = 24h\}$, also $t_i = 8min \cdot i$ für einen Messzeitpunkt $t_i \in \mathbb{T}_1$.

Testzeitraum 2: Sehr lang, sehr grob gekörnt Im letzten Testzeitraum wird ausschließlich die Clusterstabilität betrachtet. Über zwei Monate soll alle acht Stunden gemessen werden, wie sich die Clusterstrukturen geändert haben und gegebenenfalls neu geclustert werden. Der Kontakt zu Bodenstationen ist in diesem Testzeitraum nicht so wichtig wie im ersten Testzeitraum, da bei nur einem Snapshot pro acht Stunden aus der Kontaktinformation kein Nutzen gezogen werden kann. Für die Menge der Messzeitpunkte $\mathbb{T}$ gilt hier $\mathbb{T}_2 = \{0h, 8h, 1440h = 60d\}$, also $t_i = 8h \cdot i$ für einen Messzeitpunkt $t_i \in \mathbb{T}_2$

4.4. Vergleichskriterien

Um verschiedene Algorithmen in ihrer Güte vergleichen zu können, werden nachfolgend verschiedene Vergleichskriterien definiert und beschrieben.

4.4.1. Mittlere Clustergröße

Um einzelne Satelliten als Clusterheads nicht zu überlasten, sowohl durch den überhöhten Nachrichtenverkehr als auch durch den erhöhten Energieverbrauch, sollte die Anzahl der Satelliten in einem Cluster insgesamt nicht zu groß werden. Gleichzeitig sollten Satelliten möglichst gleichmäßig auf Cluster aufgeteilt werden. Als Maßzahl wird die mittlere Clustergröße betrachtet, die pro Messzeitpunkt durch den Algorithmus erzeugt wurde. Eine stabile mittlere Clustergröße, also mit geringer Standardabweichung, kann außerdem auch von einer gewissen Clusterstabilität zeugen. Die mittlere Clustergröße $\mu_c(t)$ pro Messzeitpunkt $t \in \mathbb{T}$ errechnet sich aus der Summe aller Clustergrößen $|C_i(t)|$ geteilt durch die Anzahl der Cluster $k(t)$:

$$\mu_c(t) = E\left[|C_i(t)|\right] = \frac{\sum_{i=1}^{k(t)} |C_i(t)|}{k(t)} \tag{4.2}$$

Als Maßzahl für eine gleichmäßige Verteilung der Satelliten auf die Cluster bietet sich die Standardabweichung $\sigma_c(t)$ der Clustergrößen an. Sie errechnet sich nach folgender Formel:

$$\begin{aligned}\sigma_c(t) &:= \sqrt{E\left[\left(|C_i(t)| - \mu_c(t)\right)^2\right]} = \sqrt{E\left[|C_i(t)|^2\right] - \mu_c(t)^2} \\ &= \sqrt{\frac{\sum_{i=1}^{k(t)} |C_i(t)|^2}{k(t)} - \mu_c(t)^2} \end{aligned} \tag{4.3}$$

Die mittlere Clustergröße $\mu_c(t)$ gibt eine erste Beschreibung der allgemeinen Tendenz der Clustergrößen. Ist $\mu_c(t)$ recht groß in Relation zur insgesamten Anzahl aller Satelliten im Szenario, gibt es im Gegenzug wenige Cluster, die aber jeweils viele Satelliten beinhalten. Ist $\mu_c(t)$ eher klein in Relation, gibt es viele Cluster mit wenigen Satelliten. Gerade bei mittelgroßen Werten von $\mu_c(t)$ wird die Standardabweichung $\sigma_c(t)$ interessant, die angibt, wie viel die einzelnen Clustergrößen im Mittel von $\mu_c(t)$ abweichen. Ist diese Standardabweichung sehr groß, gibt es hauptsächlich große und kleine Cluster, allerdings eher wenig mittelgroße, die Satelliten sind also nicht gleichmäßig auf die Cluster verteilt. Die sehr vagen Größenangaben sind hier immer in Relation zur Gesamtzahl von Satelliten oder zu den anderen Clustergrößen zu sehen. Beispielsweise ist ein Cluster mit 25 Satelliten in einem Satellitennetzwerk mit insgesamt 30 Satelliten groß, während zwei weitere gebildete Cluster mit jeweils 2 und 3 Satelliten dagegen sehr klein sind. Besteht ein Szenario aus 50 Satelliten, kann man 3 Cluster mit respektive 11, 14 und 15 Satelliten jeweils als mittelgroß bewerten.

4.4.2. Anzahl der Cluster über Gruppen von Bodenstationen

Um möglichst optimal Daten zwischen Satelliten und Bodenstationen austauschen zu können, sollten pro Bodenstationsmenge k Cluster mit jeweils $\lfloor \frac{n}{k} \rfloor [+1]$ Satelliten erzeugt werden, mit n als Anzahl der erreichbaren Satelliten pro Bodenstationsmenge und k Anzahl der Bodenstationen in der Menge. Als Maßzahl soll hier die Differenz zwischen optimaler und tatsächlicher Clusteranzahl betrachtet werden. Die optimale Clusterzahl über einer Menge von Bodenstationen ist genau gleich der Kardinalität dieser Bodenstationsmenge. Ob ein Cluster erreichbar ist, hängt davon ab, ob ein Clusterhead die Bodenstation in seiner Reichweite hat.

$$facsAsia(t) := |5 - clusterHeadsOverAsia(t)| \tag{4.4}$$
$$facsUSA(t) := |4 - clusterHeadsOverUSA(t)| \tag{4.5}$$
$$facsEurope(t) := |3 - clusterHeadsOverEurope(t)| \tag{4.6}$$

Für einen Zeitpunkt $t \in \mathbb{T}$ bezeichnen die einzelnen Funktionen $facsXXX(t)$[2] auf der linken Seite die Differenz zwischen der tatsächlichen Anzahl an Clustern,

[2] $facs$ steht hierbei für "Facilities", womit im STK Bodenstationen bezeichnet werden

die Kontakt zu der entsprechenden Bodenstationsgruppe besitzen (bezeichnet durch $clusterHeadsOverXXX(t)$), und der idealen Anzahl an Clustern, die Kontakt besitzen sollten, also der Anzahl an Bodenstationen dieser Bodenstationsmenge. Je niedriger also die Maßzahl, desto besser der Bodenstationskontakt. Es kann auch sein, dass zu viele Cluster über einer Bodenstationsmenge erscheinen, was aber ebenso wenig erstrebenswert ist wie zu wenige Cluster.

4.4.3. Clusterstabilität

Um möglichst viel Energie zu sparen, sollten wenig Neuzuweisungen oder Reclusteringprozesse angestoßen werden. Allerdings sind solche Aktivitäten nicht immer vermeidbar. Beispielsweise kann ein Satellit aus einem Cluster ausscheiden und in ein anderes Cluster übergehen oder gar ein eigenes Cluster bilden, wenn er keinen Kontakt zu anderen Clustern besitzt, wodurch ein Reclusteringprozess angestoßen werden muss. Es soll also im Mittel betrachtet werden, wie oft auf der einen Seite insgesamt Reclusteringprozesse durchgeführt werden und auf der anderen Seite Satelliten die Cluster wechseln. Für die Algorithmen WCA, BC und NH wird dann von einem Reclusteringprozess gesprochen, wenn sich die Menge der Clusterheads von einem Messzeitpunkt zum vorherigen unterscheidet. Bei PMW wird nur dann ein Reclustering angestoßen, wenn ein Satellit nach 3 Runden Warten ohne Clusterhead zu wenig Energie besaß, um selbst Clusterhead zu werden und keinen Clusterhead in der Nähe hatte. Reassignments werden nur dann gezählt, wenn ein Satellit seinen Clusterhead gewechselt hat, ohne dass ein Reclustering stattfand.

4.4.4. Datendurchsatz

Um zu schätzen, wieviele Daten durch die erzeugten Cluster an Bodenstationen übertragen werden können, soll der mittlere Datendurchsatz pro Messzeitpunkt $t \in \mathbb{T}$ betrachtet werden. Hier kann und soll nur eine mittlere Größe angegeben werden, da der tatsächliche Datendurchsatz unter anderem von Datenverlustraten und Schedulingverfahren zur Übertragungskoordinierung abhängt, die nur schwer simuliert werden können, wenn nur diskrete Zeitpunkte betrachtet werden.

Für die Schätzung wurden verschiedene Größen eingeführt: Für jeden Messzeitpunkt $t \in \mathbb{T}$ wird der momentane Gesamtdurchsatz $TP(t)$ berechnet, der aus den Durchsätzen $TP_{C_i(t)}(t)$ aller Cluster besteht, die Kontakt zu Bodenstationen besitzen. Diese Menge soll mit $C_{fac}(t)$ bezeichnet werden. Gleichzeitig werden der bisherige kumulierte Datendurchsatz $\Sigma_{TP}(t)$ sowie der bisherige mittlere Durchsatz $\mu_{TP}(t)$ berechnet. Der Gesamtdurchsatz $TP_{C_i(t)}(t)$ hängt davon ab, wie groß das Cluster $C_i(t)$ ist und wie lange der Clusterhead Kontakt zu einer Bodenstation besitzt. Für die Kontaktdauer zwischen Clusterhead und Bodenstation wird ein Mittelwert von 10 Minuten pro Überflug angenommen, der einerseits aus mehreren Versuchen im STK entnommen wurde, andererseits aus [55] übernommen. Dort wurde die mittlere Überflugszeit mit 5 bis 15 Minuten angegeben, allerdings hat der betrachtete MOST-Satellit eine höhere Orbithöhe als die in den Satellitenszenarien 3 und 4 verwendeten Satelliten. Die Bandbreite der Kommunikationslinks zwischen Satelliten und Boden-

stationen wird wie bei UWE-I als $1200bit/s = 150byte/s$ angenommen [54].

Der momentane Gesamtdurchsatz $TP_{C_i(t)}(t)$ zu einem Messzeitpunkt $t \in \mathbb{T}$ für ein einzelnes Cluster $C_i(t)$ berechnet sich wie folgt: Für eine angenommene Kontaktdauer von 10 Minuten, also 600 Sekunden, wird die Anzahl der Clustermember betrachtet. Die Bandbreite der Member variiert zwischen $150bit/s$ bis zu maximal $400bit/s$. Die benutzte Bandbreite ist invers proportional zur Clustergröße. Befinden sich also 8 oder mehr Member im Cluster, so wird für jeden Satelliten eine Bandbreite von $150bit/s$ angenommen. Für eine kleinere Anzahl an Satelliten berechnet sich die Bandbreite abhängig von der Clustergröße $|C_i(t)|$ durch $bw_{C_i(t)}(t) = \frac{1200bit/s}{|C_i(t)|}$ bis zu einer maximalen Bandbreite von $400bit/s$. Befinden sich also 2 oder weniger Member in einem Cluster, so wird die maximal mögliche Bandbreite von $1200bit/s$ nicht ausgenutzt. Mathematisch ausgedrückt ergibt sich folgende Formel für die Bandbreite $bw_{C_i(t)}(t)$ für die Kommunikation zwischen Satelliten in einem Cluster:

$$bw_{C_i(t)}(t) = \begin{cases} 400bit/s & |C_i(t)| \leq 2 \\ \frac{1200bit/s}{|C_i(t)|} & 3 \leq |C_i(t)| \leq 8 \\ 150bit/s & |C_i(t)| \geq 9 \end{cases} \tag{4.7}$$

Für den momentanen Gesamtdurchsatz $TP_{C_i(t)}(t)$ für ein Cluster ergibt sich:

$$TP_{C_i(t)}(t) = 600s \cdot \min\left(1200bit/s, |C_i(t)| \cdot bw_{C_i(t)}(t)\right) \tag{4.8}$$

Der momentane Gesamtdurchsatz für alle Cluster berechnet sich als Summe aller $TP_{C_i(t)}(t)$:

$$TP(t) = \sum_{C_i(t) \in C_{fac}(t)} TP_{C_i(t)}(t) \tag{4.9}$$

Für den bisherigen kumulierten Datendurchsatz $\Sigma_{TP}(t_k)$ bis zu einem Zeitpunkt $t_k \in \mathbb{T}$ gilt:

$$\Sigma_{TP}(t_k) = \begin{cases} \Sigma_{TP}(t_{k-1}) + TP(t_k) & k \neq 0 \\ TP(t_0) & k = 0 \end{cases} \tag{4.10}$$

Für den bisherigen mittleren Datendurchsatz $\mu_{TP}(t_k)$ bis zu einem Zeitpunkt $t_k \in \mathbb{T}$ gilt schließlich:

$$\mu_{TP}(t_k) = \frac{\Sigma_{TP}(t_k)}{k+1} \tag{4.11}$$

Kapitel 5.

Implementierung

Für die Implementierung der Algorithmen wurde ein Framework implementiert, das grundlegende Basisfunktionalitäten für alle Algorithmen bereitstellt, das aber ebenso anpassbar ist, um verschiedene Eigenheiten zu berücksichtigen. Dieses Framework kommuniziert mit dem Satellite Tool Kit (STK) von Analytical Graphics Inc. (AGI), um Satellitenpositionen und Kontaktinformationen zu gegebenen Zeitpunkten zu bestimmen und diese zu verarbeiten.

5.1. Übersicht über den Simulationsablauf

Im Folgenden soll eine Übersicht über den Simulationsablauf gegeben werden. Im ersten Abschnitt wird der gesamte Ablauf kurz umrissen, während er in den nachfolgenden Abschnitten nochmals detaillierter beschrieben wird.

5.1.1. Kurzübersicht

Eine der Voraussetzungen für einen erfolgreichen Simulationsdurchlauf ist, dass eine Instanz des STK bereits gestartet ist und die Szenariodaten, die eingelesen werden, bereits existieren. Im STK selbst muss außerdem manuell ein Report-Modell für Chains erstellt worden sein. Im STK repräsentieren Chains die möglichen Verbindungen zwischen Satelliten untereinander und Bodenstationen und sind für die Distanzberechnung notwendig. Im STK können automatisiert Reports für Objekte des STK erstellt werden, die mehrere Informationen umfassen können. Welche Informationen ein Report enthält, wird in Report-Modellen definiert. Für weitere Informationen über Reports siehe [56] unter "Generate Reports & Graphs". Schließlich muss auch eine lokale Installation von Octave vorhanden sein. Für einen Simulationsdurchlauf wird eine Octave-Instanz gestartet, die die Matrizen speichert und deshalb eingebunden wurde, um sehr rechenaufwändige Operationen an großen Matrizen schnell durchzuführen, wie Multiplikationen oder Subtraktionen.

In Abbildung 5.1 wird der Ablauf eines Simulationsdurchlaufs schematisch dargestellt.

Eine Simulation wird durch den Aufruf der Main-Methode der Simulationsklasse gestartet. Die Simulationsklasse besitzt den Start- und Endzeitpunkt t_{start} bzw. t_{end} des Szenarios als globale Variablen, außerdem auch das Simulationsintervall $\overline{t}$. Zudem wird bei Beginn definiert, welches Satellitenszenario benutzt werden soll. Zuerst werden die Satelliten- und Bodenstationsdaten aus einer XML-Datei geladen (Für die

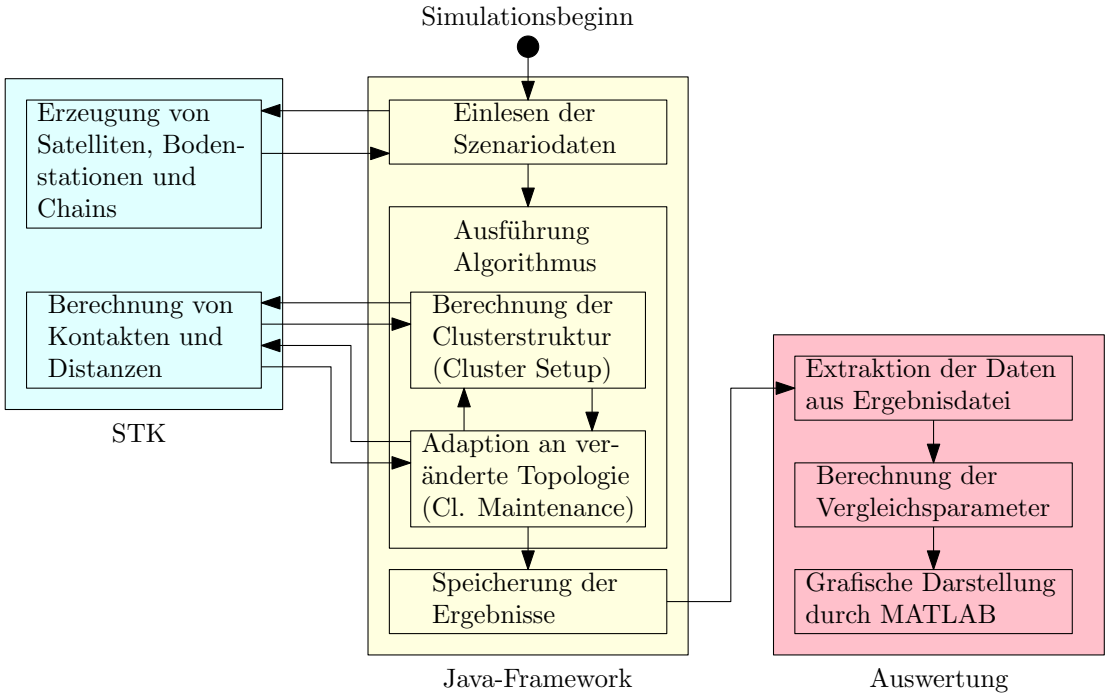


Abb. 5.1.: Schematisierter Simulationsablauf. Die drei Hauptkomponenten sind in 3 Kästen dargestellt. Die Pfeile zwischen verschiedenen Ablaufkomponenten stellen den Datenfluss dar. Der Datenfluss zwischen dem Java-Framework und dem STK wird durch das Connect-Framework verwirklicht. Der Datenfluss zwischen Java-Framework und Auswertung erfolgt über XML-Dateien.

Struktur der XML-Datei siehe Listing 5.1). Mit den gegebenen Informationen wird ein Szenario im STK erzeugt, die geladenen Satelliten und Bodenstationen eingefügt und zwischen allen Objekten Chains erstellt. Sobald dies geschehen ist, werden alle Kontakte zwischen Objekten über den gesamten Simulationszeitraum berechnet und anschließend der eigentliche Algorithmus ausgeführt, der in Abbildung 5.1 durch einen extra Kasten gekennzeichnet ist.

Vor jedem Durchlauf des Algorithmus für einen Zeitpunkt t_i wird die zu diesem Datum gehörige globale Distanzmatrix berechnet, die die Distanzen zwischen allen Satelliten untereinander und außerdem zwischen Satelliten und Bodenstationen in ihren Zellen beinhaltet. Aufbauend darauf erzeugt der angewandte Algorithmus eine Menge von Clustern. Diese Menge wird gespeichert und an den erneuten Aufruf des Algorithmus für den nächsten Zeitpunkt $t_{i+1} := t_i + \bar{t}$ übergeben, um etwaige Änderungen an der Topologie des Satellitennetzwerks zu beurteilen und entsprechend die Clusterstruktur beizubehalten oder neu zu clustern.

Sind alle Messzeitpunkte abgearbeitet, werden die erzeugten Cluster in einer XML-Datei (siehe Listing 5.3) gespeichert und das Szenario im STK beendet. Aus dieser XML-Datei werden nun mit einem anderen Java-Skript alle notwendigen Informationen extrahiert, um die Vergleichskriterien aus Abschnitt 4.4 berechnen zu können. Diese Daten werden in einer Plaintext-Datei gespeichert, wo sie von einem

Matlabskript ausgelesen werden und schließlich daraus die Ergebnisgrafiken aus Kapitel 6 erzeugt werden.

Im Folgenden soll näher auf die einzelnen Vorgänge eingegangen werden.

5.1.2. Kommunikation mit dem STK über Connect

Grundlage für die Kommunikation zwischen Framework und AGI STK bildet das Connect-Framework, das von AGI Software bereitgestellt wird und Kommunikation zwischen dem STK und eigener Software über TCP/IP ermöglicht. Die zugehörige Klasse im Java-Framework findet sich im Paket StkConnection (Abschnitt 5.2.2) und heißt `StkCon` und wurde mit dem STK mitgeliefert. Die Steuerung des STKs durch das Connect-Framework erfolgt über Plaintextbefehle, deren Antwort ebenfalls als Plaintext zurückgegeben wird. Ein Befehl über Connect ist nach dem Schema `Kommando [STK-Pfad zum manipulierten Objekt] [Optionen]` aufgebaut und liefert entweder die gewünschten Daten oder eine Fehlermeldung zurück. Für eine genaue Befehlsreferenz siehe [56] unter "External Control using Connect/Connect Command Library". Durch die Kommunikation über TCP/IP ist es außerdem möglich, eine entfernte Instanz von STK zu steuern.

5.1.3. Einlesen der Szenariodaten und Erzeugung von Satelliten, Bodenstationen und Chains

Bevor etwas eingelesen wird, muss eine Verbindung zum STK hergestellt werden. Dies geschieht mittels eines `StkCon`-Objektes, das standardmäßig versucht, eine Verbindung zu einer STK-Instanz auf dem lokalen Rechner herzustellen. Durch einen zweiten Konstruktor wäre es auch möglich, eine Verbindung zu einem entfernten Rechner aufzubauen. Anschließend werden ein `SatelliteFactory`-Objekt spezifisch für den zu simulierenden Algorithmus und ein `FacilityFactory`-Objekt erstellt, über die die Satelliten und Bodenstationen in ein Szenario eingefügt werden. `SatelliteFactory` wurde als Interface entworfen, das für jeden Algorithmus neu implementiert werden muss, da zwar manche Methoden immer vorhanden sein müssen, aber viele Funktionen auch vom implementierten Algorithmus abhängen und nicht generisch programmierbar waren.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2    <cussMainDeclaration id="1">
3      <participiants id="2">
4        <groundStationList id="3" serialization="custom">
5          <int>12</int>
6          <linked-list>
7            <default/>
8            <groundStation id="4">
9              <name>Wurzburg</name>
10             <latitude>49.7878</latitude>
11             <longitude>9.9361</longitude>
12           </groundStation>
13           ...
14         </linked-list>
```

```
15        </groundStationList>
16        <satelliteList id="16" serialization="custom">
17          <int>7</int>
18          <linked-list>
19            <default/>
20            <satellite>
21              <name>Satellit0CP4</name>
22              <ssc>31132</ssc>
23              <tle>1 31132U 07012Q   11025.04034972  .00000235  00000-0
                    61783-4 0    998
24                  2 31132 097.9109 064.1389 0084584 244.7998 114.4378
                       14.55415361200386</tle>
25            </satellite>
26            ...
27          </linked-list>
28        </satelliteList>
29      </participiants>
30    </cussMainDeclaration>
```

Listing 5.1: scenario_ x _ y.xml

```
1 <satellite>
2   <name>Satellit0Sat0</name>
3   <orbitalData>
4     <semimajorAxis>6778511.0</semimajorAxis>
5     <eccentricity>0.0017323051872690748</eccentricity>
6     <inclination>99.84166284678739</inclination>
7     <argOfPerigee>5.0</argOfPerigee>
8     <raan>0.0</raan>
9     <meanAnomaly>5.293942762455972</meanAnomaly>
10    <orbitEpoch>1 Jan 2011 00:00:00.000</orbitEpoch>
11  </orbitalData>
12 </satellite>
```

Listing 5.2: keplerSatellite.xml

Die Daten für die zu erstellenden Satelliten und Bodenstationen müssen in einer XML-Datei vorliegen, wie es in Listing 5.1 für über TLE-Daten eingebundene Satelliten und in Listing 5.2 für über Kepler-Elemente eingebundene Satelliten beschrieben wird. Die Szenario-Dateien sind nach dem Muster `scenario_[time]_[satellites].xml` benannt. Hierbei bezeichnet `[time]` den Simulationszeitraum (0 für den Kurzzeittest, 1 für den Langzeittest) und `[satellites]` das jeweilig benutzte Satellitenszenario (siehe auch Abschnitt 4.2). Für zufällig erzeugte Satelliten werden die Kepler-Elemente anstelle der SSC und TLE-Daten angegeben. Diese Dateien sind unabhängig vom ausgeführten Algorithmus und bieten ein gemeinsames Szenario für alle Algorithmen, auf denen diese verglichen werden können. Über die Klasse `ScenarioLoader` im Paket `persistence` wird eine solche XML-Datei eingelesen und ein `Scenario`-Objekt mit den in der XML-Datei beschriebenen Satelliten und Bodenstationen zurückgegeben. Durch das Erstellen dieses `Scenario`-Objektes wird gleichzeitig im STK ein Szenario geöffnet und entsprechend mit Satelliten, Bodenstationen und Chains befüllt. Falls die Satellitenszenarien 1 oder 2 verwendet werden, erhalten die Satelliten einen internen Namen

nach dem Schema "`Satellit[`k`][designierter Name]`", wobei `[`k`]` eine fortlaufende Nummer bezeichnet und `[designierter Name]` den offiziellen Namen des Satelliten bezeichnet, also beispielsweise `Satellit0UWEI`. Da keine Sonderzeichen oder Leerzeichen im Namen enthalten sein dürfen, erhalten Bodenstationen ihren englischen Namen, zum Beispiel `Wurzburg` und Leerzeichen werden durch Unterstriche ersetzt, wie geschehen bei `New_York`. Die Chainbenennung erfolgt nach dem Schema `[Startobjekt][Endobjekt]`. Eine beispielhafte Benennung einer Chain könnte also `Satellit0UWEISatellit2CUBESATXIV` lauten. Es wurden außerdem die umgekehrten Chains erzeugt, es existiert entsprechend auch eine Chain mit dem Namen `Satellit2CUBESATXIVSatellit0UWEI`. Für Chains zwischen Bodenstationen und Satelliten lautet die Benennung grundsätzlich `[Bodenstation][Zielsatellit]`.

Falls Satelliten über ihre TLE-Daten eingefügt werden, wie es in den Satellitenszenarien 1 und 2 der Fall ist, werden die Informationen über deren Orbitbahnen aus dem Internet von einem AGI-Server heruntergeladen und anschließend mittels eines SGP4-Propagators lokal berechnet. Es ist auch möglich, eine lokale Datenbank zu verwenden, allerdings ist diese dann nicht unbedingt aktuell. Die Satelliten aus den Szenarien 3 und 4 werden über ihre Kepler-Elemente eingebunden. Im STK selbst wird dann zur Bahnberechnung ein J2-Propagator verwendet. Für Informationen über SGP4- und J2-Propagatoren siehe Abschnitt 2.5.

Wurden alle Objekte im STK erzeugt, werden schließlich noch die Kontaktzeiten aller Objekte untereinander berechnet. Nachdem auch diese Berechnung abgeschlossen wurde, können die Algorithmen ausgeführt werden.

5.1.4. Ausführung der Algorithmen

Vor jeder Ausführung eines Algorithmus für einen gewissen Zeitpunkt t wird die gesamte globale Distanzmatrix erstellt und in einer Octave-Instanz gespeichert. Eine Distanzmatrix ist eine Adjazenzmatrix, deren Zellen die Distanzen zwischen den Knoten enthalten (siehe Abschnitt 2.2.1). Als globale Distanzmatrix soll die Distanzmatrix nicht nur zwischen allen Satelliten, sondern zusätzlich zwischen allen Satelliten und Bodenstationen bezeichnet werden. Da dieser Schritt normalerweise mehrmals notwendig ist, wird die erzeugte Matrix für einen Zeitpunkt gespeichert und bei Bedarf erneut abgerufen, was erheblich Zeit bei der Ausführung spart, da die Distanzabfrage aus dem STK sehr zeitintensiv ist und ohne Speicherung signifikant mehr Abfragen ans STK gestellt werden müssen als bei einer einfachen Berechnung und anschließenden Speicherung. Die Erzeugung der Distanzmatrix erfolgt durch einmalige Abfrage der Distanzen zwischen allen 2-Mengen[1] von Satelliten und Bodenstationen. Dies bedeutet, dass zwar die Distanz zwischen Satellit x und Satellit y abgefragt wird, nicht jedoch zwischen Satellit y und Satellit x. Hierbei wird die Symmetrie der Distanzmatrix ausgenutzt, was die Hälfte der Rechenschritte spart. Außerdem gilt für die Distanzmatrix zwischen Bodenstationen, dass diese gleich null ist, da die Distanzen zwischen den Bodenstationen untereinander nicht interessant für die Ausführung der ausgewählten Algorithmen sind. Es wäre aber durchaus denkbar, eine Heuristik zu entwerfen, die unter anderem solche Distanzinformationen ausnutzen könnte.

[1] Eine 2-Menge bezeichnet eine Menge mit 2 Elementen

Für jeden Algorithmus wurde eine eigene Klasse erstellt, die jeweils die statischen Methoden `run(StkDate now, ...)` und `runWithStabilization(StkDate now, ClusterList oldClusterList...)` besitzen. Durch beide Methoden wird ein `ClusterList`-Objekt zurückgegeben, das die erzeugten `Cluster`-Objekte beinhaltet. Die `run`-Methode stellt die Cluster Setup-Phase dar, in der eine Clusterstruktur errichtet wird, während die `runWithStabilization`-Methode die Cluster Maintenance-Phase darstellt, in der versucht wird, eine bestehende Clusterstruktur beizubehalten. Falls `runWithStabilization` mit `oldClusterList = null` aufgerufen wird, wird der Funktionsaufruf direkt an `run` weitergegeben. Weitere Aufrufe für `run` ergeben sich durch spezifische Eigenheiten der Algorithmen.

Nach einem einzelnen Durchlauf des Algorithmus für einen Zeitpunkt t_i wird ein ClusterList-Objekt zurückgegeben, das die für den Zeitpunkt t_i erzeugten Cluster enthält. Diese Cluster werden durch `Cluster`-Objekte repräsentiert, die den Clusterhead, die Clustermember und den Erstellungszeitpunkt gespeichert haben. Dieses zurückgegebene ClusterList-Objekt wird einerseits in eine Speicherliste eingetragen, die ebenfalls vom Typ ClusterList ist, andererseits wird es so gespeichert, dass es im Aufruf des Algorithmus für Zeitpunkt $t_{i+1} = t_i + \bar{t}$ als `oldClusterList` erscheint.

5.1.5. Speicherung und Auswertung der Ergebnisse

```
<?xml version="1.0" encoding="UTF-8" standalone="no">
<clusterData>
  <scenarioData>
    <scenarioFile>scenario_x_y.xml</scenarioFile>
    <clusterInterval>0 interval 0 00:08:00.000</clusterInterval>
  </scenarioData>
  <clusterList id="1" serialization="custom">
    <int>872</int>
    <linked-list>
      <default/>
      <date string="1 Jan 2011 00:00:00.000">
        <cluster id="2">
          <clusterhead>
            <satellite>
              <name>Satellit7CANX2</name>
              <ssc>32790</ssc>
            </satellite>
            <nearestFacility>
              <name>Tokyo</name>
              <latitude>35.685</latitude>
              <longitude>139.7514</longitude>
              <reachableSatellites>
                <int>1</int>
                <satellite>
                  <name>Satellit7CANX2</name>
                  <ssc>32790</ssc>
                </satellite>
              </reachableSatellites>
            </nearestFacility>
          </clusterhead>
```

```
31            <members>
32              ...
33            </members>
34          </cluster>
35          ...
36        </date>
37      </linked-list>
38    </clusterList>
39    <reachableSatellites>
40      <date string="1 Jan 2011 00:00:00.000">
41        <groundStation id="1">
42          <name>Tokyo</name>
43          <latitude>35.685</latitude>
44          <longitude>139.7514</longitude>
45          <reachableSatellites>
46            <satellite>
47              <name>Satellit7CANX2</name>
48              <ssc>32790</ssc>
49            </satellite>
50            ...
51          </reachableSatellites>
52        </groundStation>
53        ...
54      </date>
55    </reachableSatellites>
56  </clusterData>
```

Listing 5.3: clusterDataABC_ x_ y.xml

Nach allen Durchläufen des Algorithmus werden die Clusterdaten in einer XML-Datei gespeichert, die nach dem Schema in Listing 5.3 aufgebaut ist. Die Resultats-Dateien erhalten ihren Namen nach dem Muster `clusterData[Algorithmus]_[time]_[satellites].xml`, wobei auch hier `[time]` für den Simulationszeitraum und `[satellites]` für das verwendete Satellitenszenario stehen. `[Algorithmus]` steht für das Kürzel des Algorithmus, der die Daten erzeugt hat. Die Erzeugung der einzelnen Subelemente ist in den Klassen selbst in der Methode `createNode` gespeichert, die eine XML-Node erzeugt, die dann anschließend direkt in einen XML-Baum eingehängt werden kann.

In der XML-Datei mit den Clusterdaten wird zuerst in `<scenarioData>` ein Verweis auf das zugrundeliegende Satellitenszenario gegeben sowie das Intervall zwischen zwei Simulationsschritten in einem `StkDate`-Format angegeben. Anschließend folgt in `<clusterList>` die Liste aller erzeugten Cluster, nach Datum der Erzeugung geordnet.

Das erste Kind von `<clusterList>` beschreibt die Anzahl seiner Kinder, die unter dem `<linked-list>`-Tag gespeichert sind. Unter `<linked-list>` folgt schließlich die Auflistung aller Simulationszeitpunkte `<date>`, in deren Attribut `string` das explizite Datum in `StkDate`-Format gespeichert ist. Die Kinder von `<date>` schließlich sind die zu diesem Zeitpunkt erzeugten Cluster, für die jeweils ein `<cluster>`-Element angelegt wurde.

Das erste Kind eines `<cluster>` ist der Clusterhead des Clusters, der entweder durch seine SSC und seinen qualifizierten Namen angegeben wird oder durch seine

Keplerelemente. Falls der Clusterhead Kontakt zu Bodenstationen besitzt, werden diese ebenfalls aufgeführt sowie eine Liste der von dieser Bodenstation aus erreichbaren Satelliten. Das zweite Kind eines Cluster-Knoten beinhaltet die verschiedenen Clustermember, die wie der Clusterhead selbst durch ihre SSC und qualifizierten Namen oder ihre Kepler-Elemente angegeben sind. Für die Clustermember werden allerdings keine erreichbaren Bodenstationen angegeben.

Wurde der XML-Baum dann vollständig erzeugt und gespeichert, ist der Simulationsschritt an sich abgeschlossen und das Szenario im STK wird geschlossen und alle Verbindungen des Java-Frameworks zum STK sowie zu Octave beendet.

```
1  <Anzahl Satelliten>
2  <Anzahl Bodenstationen>
3  <Anzahl Messzeitpunkte>
4  ---
5  <Größe der an diesem Messzeitpunkt erzeugten Cluster>
6  1.0 5.0 1.0 5.0 1.0
7  5.0 5.0 2.0 1.0
8  ...
9  2.0 4.0 4.0 1.0 2.0
10 2.0 3.0 4.0 3.0 1.0
11 ---
12 <Anzahl der Bodenstationen mit Clusterkontakt>
13 <und die Größe der Cluster mit Kontakt,
14 <aufgeschlüsselt nach Kontinenten>
15 EU 0 0 AS 2 4 AM 0 0
16 EU 0 0 AS 1 3 AM 0 0
17 ...
18 EU 0 0 AS 1 3 AM 0 0
19 EU 0 0 AS 0 0 AM 0 0
20 ---
21 <Anzahl der Reassignments pro Messzeitpunkt>
22 ---
23 <Anzahl der Reclusterings pro Messzeitpunkt>
24 ---
```

Listing 5.4: outputABC_ x_ y.txt

Um eine solche XML-Datei auszuwerten, wurde in Java ein weiteres Skript geschrieben, das über den XML-Baum traversiert und eine Textdatei nach dem Schema von 5.4 ausgibt. Alle Abschnitte dieser Datei werden durch --- getrennt. In der Präambel dieser Datei findet sich die Anzahl der Satelliten, der Bodenstationen und die Anzahl der Messzeitpunkte. Der zweite Abschnitt listet die einzelnen Clustergrößen zu allen Messzeitpunkten auf. Im dritten Abschnitt stehen pro Zeile die Anzahl der Cluster, die Kontakt zu den jeweiligen kontinental unterteilten Bodenstationen besitzen sowie die summierte Anzahl der Satelliten dieser Cluster. Der vierte Abschnitt beschreibt die Anzahl von Neuzuweisungen von Satelliten an andere Cluster pro Messzeitpunkt. Falls zu diesem Messzeitpunkt ein Reclustering stattfand, wurde eine 0 geschrieben. Der fünfte Abschnitt schließlich gibt Auskunft über die Reclusterings zu den einzelnen Messzeitpunkten. Falls ein Reclustering stattfand, steht hier eine 1, ansonsten eine 0.

5.2. UML-Darstellung des Frameworks

Für einen Gesamtüberblick über das Framework kann man die nachfolgenden UML-Darstellungen betrachten, die alle wichtigen Klassen und Relationen zeigen. In Abbildung 5.2 sind außerdem die Paketabhängigkeiten kompakt aufgeführt. In den folgenden Unterabschnitten werden die verschiedenen Pakete im Java-Framework vorgestellt und die wichtigsten Klassen und Interfaces kurz vorgestellt. Wo es nötig war, wurde auf die tatsächliche Implementierung eingegangen. Eine detaillierte Dokumentation des gesamten Frameworks würde allerdings den Rahmen der Diplomarbeit sprengen und ist auf der CD im Anhang enthalten.

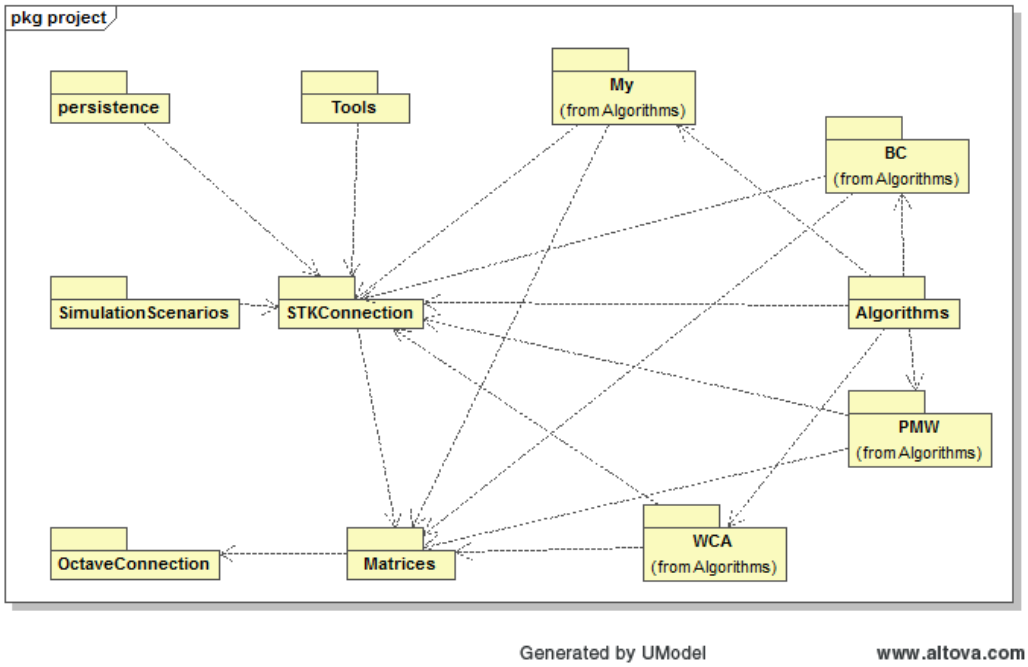


Abb. 5.2.: Die Abhängigkeiten der Pakete untereinander. Ein Pfeil von einem Paket zu einem anderen symbolisiert eine Abhängigkeit.

5.2.1. Basisverzeichnis

Im Basisverzeichnis des Frameworks liegt für jeden Algorithmus eine ausführbare Klasse, die den in Abschnitt 5.1 beschriebenen Simulationsablauf durchführt, indem die jeweilge `runWithStabilization`-Methode der Algorithmusklasse im Paket 5.2.3 aufgerufen wird. Diese ausführbaren Klassen besitzen jeweils die globalen Variablen `time` und `satellites`, die definieren, welches Szenario simuliert werden soll, wobei `time` auf 0 für das Kurzzeitszenario oder 1 für das Langzeitszenario gesetzt werden kann und `satellites` einen Wert von 1 bis 4 zugewiesen bekommen muss und dadurch das zu simulierende Satellitenszenario auswählt. Abhängig von `time` wird die Variable `interval` auf den entsprechenden Intervallwert von 8 Minuten oder 8 Stunden gesetzt. In der Variable `scenarioFile` steht dann der Name der XML-Datei, die die entsprechenden Satelliten- und Bodenstationsdaten beinhaltet. `outputString`

beinhaltet den Namen der Ausgabedatei, in der die berechneten Cluster gespeichert werden.

Die ausführbare Klasse `CreateScenarios` erzeugt die XML-Dateien mit den Satelliten- und Bodenstationsauflistungen, die als Grundlage für die Simulationen dienen.

5.2.2. Paket STKConnection

Im Paket STKConnection liegt das Grundgerüst des Frameworks in Interfaces und generischen Klassen vor, die nachfolgend beschrieben werden. Damit kann ein Algorithmus komfortabel implementiert werden.

StkDate Das STK benutzt ein von Java abweichendes Datumsformat. Deswegen wurde mit `StkDate` eine Schnittstelle zwischen beiden Formaten erstellt, die so ineinander konvertiert werden können. Die `StkDate`-Klasse wird für alle Datumsangaben im Framework benutzt und arbeitet intern mit einem `Integer`-Array, kann aber auch `String`s parsen. Ein `StkDate`-Objekt kann außerdem als Intervall benutzt werden, da Methoden zum Addieren und Subtrahieren zweier `StkDate`-Objekte existieren.

StkCon Die Klasse `StkCon` stammt aus dem STK-Framework und wurde von Qui Lui entworfen und von einigen weiteren Autoren verbessert. Die Klasse stellt Wrapper-Funktionen für einen Verbindungsauf- und abbau sowie Nachrichtenaustausch über TCP/IP zum STK bereit. Von dieser Klasse existiert während eines Simulationsdurchlaufs nur eine Instanz, die im `Scenario`-Objekt gespeichert wird.

Scenario Die `Scenario`-Klasse ist das Herzstück der gesamten Simulation. Über sie werden Befehle mithilfe einer singulären `StkCon`-Instanz an das STK geschickt und von dort empfangen, außerdem stellt sie Verknüpfungen zur `SatelliteFactory`, der `FacilityFactory` und der `MatrixFactory` (im Paket `Matrices`, Abschnitt 5.2.5) des Szenarios bereit und bietet außerdem Zugriff auf die zu den verschiedenen Messzeitpunkten erzeugten Matrizen. Ein `Scenario`-Objekt kann entweder direkt erstellt werden oder über den `ScenarioLoader` aus dem Paket `Persistence` (Abschnitt 5.2.7) geladen werden. Durch die Erstellung eines `Scenario`-Objektes wird gleichzeitig ein Szenario im STK geöffnet sowie eine Instanz von Octave gestartet, in der die Matrizen verwaltet werden (siehe hierzu Abschnitt 5.2.6).

Satellite `Satellite` ist ein Interface, das grundlegende Operationen für Satelliten bereitstellt wie z.B. die Abfrage der Distanz zu einem anderen Satelliten oder die Berechnung der Nachbarschaft, jeweils zu einem festgelegten Zeitpunkt. Außerdem kann abgefragt werden, ob der jeweilige Satellit momentan ein Clusterhead ist. Wie die verschiedenen Operationen implementiert werden, hängt vom Algorithmus ab. In allen Implementierungen kontaktieren die Konstruktoren das STK und legen dort ebenfalls ein neues Satelliten-Objekt an. Außerdem werden die Erzeugungsdaten wie Name, SSC oder Keplerelemente in einer `Satellite`-Klasse gespeichert. Die Methode `createNode` liefert eine XML-Repräsentation des Satellitenobjekts für die

Speicherung zurück. Siehe hierzu auch den Abschnitt 5.3, wo die verschiedenen Algorithmusimplementierungen näher betrachtet werden.

SatelliteFactory Das generische Interface `SatelliteFactory` ist von einer Parameterklasse, die von Satellite erbt, abhängig, die den Satellitentyp beschreibt, der in der jeweiligen Simulation benutzt wird. Eine SatelliteFactory verwaltet die Satelliten im Szenario. Sie kann neue Satelliten erstellen, die entweder durch ihre SSC beschrieben werden oder durch ihre Kepler-Elemente, indem neue `Satellite`-Objekte erstellt werden, die ihrerseits über die `StkCon`-Instanz des `Scenario`-Objekts entsprechende Anfragen an das STK senden. Die `SatelliteFactory` besitzt außerdem eine Methode, mit der die Distanzmatrix aller Satelliten zu einem bestimmten Zeitpunkt berechnet werden kann. Darüber hinaus werden Methoden zur gezielten Selektion einzelner Satelliten bereitgestellt. Die Methode `createNode` liefert hier eine XML-Repräsentation aller erstellten Satelliten zurück, indem deren `createNode`-Methoden aufgerufen werden.

Facility Die Klasse Facility dient hauptsächlich zur Darstellung einer Bodenstation durch Name, Längen- und Breitengrad. Sie stellt aber zusätzlich Funktionen bereit, mit einem Satelliten eine Verbindung herzustellen und wieder abzubrechen, sodass sie während dem Verbindungszeitraum keine anderen Verbindungen mit Satelliten aufnehmen kann.

FacilityFactory Die generische Klasse FacilityFactory dient zur Verwaltung der Facilities (Bodenstationen) im Szenario. Durch eine FacilityFactory können neue Bodenstationen erstellt werden, die durch ihren Längen- und Breitengrad charakterisiert werden. Sie stellt außerdem Methoden bereit, um die komplette Distanzmatrix des Szenarios zu berechnen, also die Distanzen zwischen den einzelnen Satelliten untereinander sowie der Bodenstationen zu den Satelliten. Darauf aufbauend kann auch bestimmt werden, welche Satelliten zu einem gewissen Zeitpunkt von einer Bodenstation aus erreichbar waren.

Cluster Instanzen dieser Klasse werden dazu benutzt, ein erstelltes Cluster zu repräsentieren. Ein Cluster wird charakterisiert durch seinen ClusterHead, das Datum, an dem dieses Cluster erzeugt wurde und seine nächstgelegene Bodenstation. Letzteres wird für Analysezwecke eingesetzt. Ein Cluster kann auch nur aus einem Clusterhead bestehen. Ein Clusterhead zählt außerdem nicht als Clustermember. Ein Cluster besitzt außerdem eine `createNode`-Methode, mit der eine XML-Repräsentation des Clusters erzeugt wird. Hierbei wird von jedem `Satellite`-Objekt dessen `createNode`-Methode aufgerufen, um eine XML-Ausgabe wie in Listing 5.3 zu erzeugen.

ClusterList Eine `ClusterList` ist eine Liste von `Cluster`-Objekten. In der Implementierung wird für jeden Durchlauf des Algorithmus eine `ClusterList` zurückgegeben, in der die erzeugten Cluster gespeichert sind. `ClusterList`-Objekte werden hauptsächlich in den Algorithmusimplementierungen im Paket Algorithms (Abschnitt 5.2.3) verwendet. Ein `ClusterList`-Objekt besitzt außerdem ebenfalls eine

createNode-Methode, die die einzelnen createNode-Methoden der gespeicherten Cluster-Objekte aufruft und eine Repräsentation wie in Listing 5.3 erzeugt.

5.2.3. Paket Algorithms

Im Paket Algorithms liegen die Klassen, die die Algorithmusimplementierungen enthalten, namentlich BCAlgorithm, PMWAlgorithm, WCAAlgorithm und MyAlgorithm. Die Subpakete enthalten algorithmenspezifische Implementierungen der Interfaces Satellite und SatelliteFactory aus dem Paket StkConnection (Abschnitte 5.2.2 und 5.3). Die Algorithmenklassen sind alle gleich aufgebaut und enthalten jeweils eine Methode runWithStabilization, die die Cluster Maintenance-Phase des jeweiligen Algorithmus darstellt sowie eine Methode run, die die Cluster Setup-Phase implementiert. Nähere Beschreibungen der jeweiligen Implementierungen der Algorithmen finden sich im Abschnitt 5.3.

Die Methode runWithStabilization nimmt als Argumente ein SatelliteFactory-Objekt mit der Liste aller Satelliten, ein FacilityFactory-Objekt mit der Liste aller Bodenstationen, ein durch ein StkDate-Objekt repräsentiertes Datum, zu dem der Algorithmus ausgeführt wird sowie eine Vergleichs-ClusterList, für die versucht wird, deren Struktur beizubehalten, entgegen. Je nach Algorithmus können auch spezifische Parameter wie die Gewichtung der einzelnen Messgrößen (wie bei der Klasse WCAAlgorithm) als zusätzliche Parameter vorkommen. Ist die Vergleichs-ClusterList gleich dem null-Objekt, so wird die Methode run aufgerufen, die die genannten Objekte weitergereicht bekommt und eine neue Clusterstruktur errichtet. Die run-Methode wird außerdem innerhalb von runWithStabilization aufgerufen, falls ein Reclustering initiiert wurde.

Alle Methoden liefern ein ClusterList-Objekt zurück, in dem die während eines Algorithmusdurchlaufs erzeugten Cluster-Objekte gespeichert sind.

5.2.4. Paket SimulationScenarios

Das Paket SimulationScenarios stellt zwei Klassen mit einfachen Methoden bereit, die vordefinierte Bodenstations- und Satellitenmengen in ein Szenario über jeweils eine FacilityFactory oder eine SatelliteFactory einfügen können. Diese Klassen werden nur zur Erzeugung von Simulations-XML-Dateien in der Klasse CreateScenarios im Grundverzeichnis (Abschnitt 5.2.1) benutzt.

Die Klasse PredefinedGroundStations erzeugt die in Abschnitt 4.2 beschriebenen Bodenstationen, während die einzelnen Methoden der Klasse PredefinedSatellites jeweils ein im gleichen Abschnitt beschriebenes Satellitenszenario erzeugt.

5.2.5. Paket Matrices

Im Paket Matrices liegen die Klassen MatrixFactory und Matrix. Das Paket dient zur Verwaltung von Matrizen im Framework, insbesondere den in Abschnitt 5.1.4 erzeugten Distanzmatrizen. Die Klasse MatrixFactory bietet eine Anbindung an Octave, die über das Paket OctaveConnection hergestellt wird (siehe Abschnitt 5.2.6) und

verwaltet alle im Szenario erstellten Matrizen. Die `MatrixFactory` bietet außerdem zwei Methoden, um mit Matrizen zu rechnen.

Ein `Matrix`-Objekt kann über explizite Zellenwerte in einem zweidimensionalen `double`-Array oder über eine Dimensionsangabe erstellt werden. Bei der Erstellung ist es außerdem notwendig, neben einer Matrixbezeichnung Spaltenbezeichnungen in einem `String`-Array zu definieren. Diese Bezeichnungen werden später bei Bedarf sowohl für Spalten als auch Zeilen angezeigt, wenn eine Matrix auf der Konsole ausgegeben wird. Es werden nur quadratische und symmetrische Matrizen unterstützt, da nichtquadratische und nichtsymmetrische Matrizen im Framework nicht notwendig sind.

5.2.6. Paket OctaveConnection

Um die Matrizen aus dem Paket `Matrices` (Abschnitt 5.2.5) schnell einigen Rechenoperationen wie Addition, Subtraktion oder Multiplikation zu unterziehen, wurde im Paket `OctaveConnection` eine Anbindung an Octave implementiert. Die Klasse `OctaveCon` wurde als Singleton implementiert, um so zu garantieren, dass beim Zugriff auf die gespeicherten Matrizen auch ein Pendant in der laufenden Octave-Instanz existiert. Ein Singleton ist ein Entwurfsmuster der objektorienterten Programmierung, das garantieren soll, während der Laufzeit eines Programms maximal eine Instanz dieses Objektes erstellen zu können. Die tatsächliche Anbindung erfolgt hierbei über ein `OctaveEngine`-Objekt, das von einer fremden Quelle (siehe [57]) eingebunden wurde.

Diese Anbindung wurde deswegen vorgenommen, da Matrizenoperationen für große Matrizen schnell sehr rechenaufwändig werden und Octave auf eine schnelle Ausführung solcher Operationen ausgerichtet ist.

5.2.7. Paket Persistence

Das Paket `persistence` wird sowohl im Ablaufschritt 5.1.3 als auch in 5.1.5 eingesetzt.

Um in einer Simulation schnell ein gegebenes Szenario laden und speichern zu können, wurden im Paket `Persistence` zwei Klassen entworfen.

Die Klasse `ScenarioLoader` erhält als Klassenparameter den Satellitentyp, den Typ der zugehörigen `SatelliteFactory` sowie den Typ der in dieser Implementierung benutzten `FacilityFactory`. Die Methode `load` erhält als Parameter jeweils eine leere `SatelliteFactory` und eine leere `FacilityFactory`, die anschließend mit den Daten aus der Szenariobeschreibung im XML-Format befüllt werden und gibt ein fertiges `Scenario` zurück, auf dem Algorithmen ausgeführt werden können. Der Anfangs- und Endzeitpunkt dieser Simulation wurden gleichzeitig mit dem Pfad zur Szenariobeschreibung übergeben.

Die Klasse `ScenarioSaver` erhält die selben Klassenparameter wie `ScenarioLoader`. Bei der Erstellung eines `ScenarioSaver`-Objekt erhält der Konstruktor Informationen über den Speicherort der Szenariobeschreibung und der Clusterdaten, das `Scenario`-Objekt, aus dem alle Informationen über Satelliten und Bo-

denstationen extrahiert werden, das Simulationsintervall sowie die erzeugten Clusterdaten selbst.

5.2.8. Paket Tools

Die Klassen im Paket `Tools` werden im Ablaufschritt 5.1.5 eingesetzt.

Im Paket `Tools` sind schließlich zwei ausführbare Klassen enthalten, die die Clusterdaten, die ein Algorithmus erzeugt hat, auswertet und in ein für Matlab lesbares Format bringen. Die Klasse `ClusterAnalysis` übernimmt diese Aufgabe für den BC-, den WCA- und den NH-Algorithmus. Für den PMW-Algorithmus wurde die Klasse leicht verändert, da im PMW-Algorithmus Reclusterings signifikant anders als in den anderen Algorithmen gehandhabt werden.

In beiden Klassen wird durch die `main`-Methode das Ausgabedokument, das durch `outFile` definiert ist, vorbereitet sowie das Eingabedokument, also die XML-Datei mit den Clusterbeschreibungen, eingelesen. Anschließend wird die Eingabedatei mit der `analyser`-Methode analysiert sowie die Vergleichsparameter aus Abschnitt 4.4 berechnet und die Analyseergebnisse in eine Ausgabedatei geschrieben, die im nächsten Schritt von Matlab gelesen wird.

5.3. Implementierungsdetails der einzelnen Algorithmen

Im folgenden werden die Implementierungen der Algorithmen WCA, BC, PMW und einer eigenen Heuristik näher beschrieben sowie auf die notwendigen Anpassungen eingegangen.

5.3.1. Der WCA-Algorithmus

Die für den WCA-Algorithmus spezifischen Dateien liegen im Java-Framework sowohl im Paket `Algorithms` (Abschnitt 5.2.3), in dem die Algorithmenimplementierung liegt, als auch im Subpaket `Algorithms.WCA`. Die Klassen `WCASatellite` und `WCASatelliteFactory` sind Implementierungen der Interfaces `Satellite` und `SatelliteFactory` aus dem Paket `STKConnection` (Abschnitt 5.2.2). Die Klasse `WCASatellite` speichert zusätzlich die Parameter, mit denen das Gewicht das Satelliten vom Algorithmus in `WCAAlgorithm` (Paket `Algorithms`, Abschnitt5.2.3) berechnet wird, sowie den ihm zugewiesenen Clusterhead.

Am WCA-Algorithmus wurden einige Veränderungen gegenüber der eigentlich vorgeschlagenen Variante in Abschnitt 3.2.5 vorgenommen. Der eigentliche Algorithmus läuft wie folgt ab: Für jeden Satelliten v wird ein Gewicht nach folgender Formel berechnet, abhängig von vorher zu treffenden Normierungsfaktoren c_1 bis c_4 mit $\sum_{i=1}^{4} c_i = 1$ und $c_i > 0 \forall i \in \{1, 2, 3, 4\}$:

$$w_v(t) := c_1\Delta_v(t) + c_2 D_v(t) + c_3 M_v(t) + c_4 P_v(t) \tag{5.1}$$

$\Delta_v(t) = |deg(v)(t) - \delta|$ steht für die Graddifferenz des Knotens, abhängig von einer Konstante δ, die den Idealgrad des Knotens angibt. $D_v(t)$ bezeichnet die Summe der Distanzen zu den Nachbarknoten, $M_v(t)$ die mittlere Bewegung des Knotens und $P_v(t)$ steht für die als Clusterhead zugebrachte Zeit. Nach der Berechnung dieser Gewichte wird die Clusterheadsuche gestartet, wobei ein Knoten, der in seiner Nachbarschaft das niedrigste Gewicht besitzt, zum Clusterhead wird, wenn er nicht in der direkten Nachbarschaft eines anderen Clusterheads liegt.

Nachteilhaft an dieser Wahl der mittleren Bewegung als Gewicht ist, dass die Messung der eigenen Geschwindigkeit unabhängig von der Geschwindigkeit der Nachbarknoten nicht viel aussagt. Wenn sich ein Knoten v sehr langsam bewegt und somit eine niedrige mittlere Bewegung aufweist, alle anderen Knoten im Szenario sich allerdings sehr schnell bewegen, hat v zwar gute Chancen, Clusterhead zu werden, die entstehende Clusterstruktur ist allerdings sehr instabil. Der Clusteraufbau besitzt ebenfalls eine erhebliche Schwäche: Liegen alle Knoten wie im Satellitenszenario 3 sehr eng beieinander, wird nur ein Clusterhead ausgewählt, da nach seiner Wahl alle anderen Knoten als Clusterheads ausscheiden, da sie ja in seiner direkten Nachbarschaft liegen.

In der Implementierung des Algorithmus wurde das Gewicht $M_v(t)$ weggelassen und stattdessen ein weiteres Gewicht $F_v(t)$ eingeführt, das die Distanz zur nächsten Bodenstation in Einheiten von Kilometern beschreibt. Falls keine Bodenstation in der Nähe ist, wird dieses Gewicht auf 100 gesetzt, wodurch Satelliten mit Bodenstationskontakt einen erheblichen Vorteil bei der Clusterheadwahl erhalten. Die Summe der Distanzen wird aus Gründen der Skalierbarkeit ebenfalls in Einheiten von Kilometern angegeben. Die Berechnung der Graddifferenz $\Delta_{v,new}(t)$ wurde so verschärft, dass eine größere Abweichung vom Idealgrad δ sehr viel stärker ins Gewicht fällt als beim eigentlichen Algorithmus. Die Graddifferenz berechnet sich nun nach der Formel $\Delta_{v,new}(t) = \Delta_v(t)^2 \cdot \exp(\Delta_v(t) - 2)$. Eine Zeichnung des Funktionsgraphen findet sich in Abbildung 5.3.

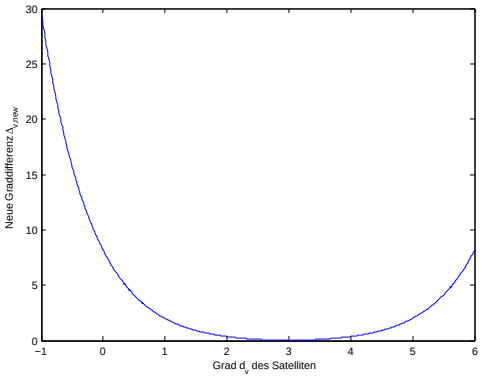


Abb. 5.3.: Funktionsgraph von $\Delta_{v,new}(t)$ für $\delta = 3$

Die Formel wurde so entworfen, dass ein Minimum (nämlich 0) bei $deg(v)(t) = \delta$

auftritt, aber Werte von $deg(v)(t)$, die nahe bei δ liegen, ebenfalls kein allzu hohes Gewicht verursachen. Werte, die weiter von δ entfernt liegen, sollen dann allerdings sehr schnell höhere Gewichte verursachen, was mit der Forderung, die Cluster möglichst gleich zu verteilen, vereinbar sein soll. Die Größe $P_v(t)$, die die als Clusterhead verbrachte Zeit ausdrückt, wurde in ihrer ursprünglichen Weise implementiert. Indirekt sagt $P_v(t)$ etwas über den Energieverbrauch aus, ist allerdings auch zu ungenau, um daraus etwas Handfestes zu schließen.

Als weitere Adaption an ein Szenario mit Bodenstationen wurde zusätzlich der Idealgrad[2] eines Knotens $\delta(t)$ dynamisch anpassbar gestaltet, sodass nun die optimale Anzahl an Nachbarn unter anderem auch von der Anzahl der Bodenstationen in der Nähe abhängt. Falls keine Bodenstationen in der Nähe liegen, so ändert sich δ auf ein Drittel aller Nachbarn $\lfloor \frac{n}{3} \rfloor$ mit n als Anzahl der Nachbarn bis zu einer Mindestgrenze von einem Drittel aller Satelliten im Netzwerk. Falls Bodenstationen in der Nähe sind, wird δ auf das Verhältnis zwischen der Zahl der Nachbarsatelliten und der Zahl der erreichbaren Bodenstationen k gesetzt, also auf den Wert $\delta = \lfloor \frac{n}{k} \rfloor$. So soll zusätzlich versucht werden, möglichst gleichmäßig verteilte Cluster zu erzeugen. In einer Formel ausgedrückt heißt das:

$$\delta(t) = \begin{cases} \lfloor \frac{n}{3} \rfloor & \text{falls keine Bodenstationen erreichbar sind} \\ \lfloor \frac{n}{3} \rfloor & \text{falls keine Bodenstationen erreichbar sind} \\ \lfloor \frac{n}{k} \rfloor & \text{falls } k \text{ Bodenstationen erreichbar sind} \end{cases} \tag{5.2}$$

Der Algorithmus berechnet nun also für jeden Satelliten das neue Gewicht nach folgender Formel:

$$w_v(t) = c_1\Delta_{v,new}(t) + c_2D_v(t) + c_3F_v(t) + c_4P_v(t) \tag{5.3}$$

Die Gewichte wurden folgendermaßen gesetzt: Als wichtigstes Kriterium wurde die als Clusterhead verbrachte Zeit mit $c_4 = 0.45$ bewertet, um so Überlastungen einzelner Satelliten möglichst zu vermeiden. Als zweitwichtigstes wurde die Graddifferenz mit $c_1 = 0.3$ bewertet, um so zu ermöglichen, nicht zu große Cluster zu bilden. Als nicht ganz so wichtig wurde der Abstand zu einer Bodenstation mit $c_3 = 0.2$ bewertet, da alleine das Kriterium selbst schon ausreicht, um Satelliten mit Bodenstationskontakt von solchen ohne deutlich zu unterscheiden. Schließlich wurde die summierte Distanz zu den restlichen Satelliten als am wenigsten wichtig mit $c_2 = 0.05$ bewertet, da gerade in Satellitenszenarien sehr große Distanzen zwischen Satelliten auftreten und somit ein sehr großer Parameter entsteht, der dadurch die anderen Parameter beinahe bedeutungslos werden lassen kann. Also wurde die Distanzsumme mit dem niedrigsten und sehr kleinen Wert gewichtet.

Die weitere Vorgehensweise wie die Wahl des Clusterheads läuft immer noch gleich ab, sodass der Satellit mit dem niedrigsten Gewicht in seiner Nachbarschaft Clusterhead wird, falls sich kein Clusterhead in seiner Nachbarschaft befindet. Die Cluster Maintenance-Phase wurde einfacher als in Abschnitt 3.2.5 implementiert, aber funktioniert nach dem selben Prinzip. So versucht nun ein Satellit, der sein Cluster

[2]der Idealgrad eines Knotens bezeichnet die ideale Anzahl seiner Nachbarn, siehe auch Abschnitt 3.2.5

verloren hat, sich einem neuen Cluster anzuschließen und stößt, falls dies misslingt, ein Reclustering an.

5.3.2. Der BC-Algorithmus

Im Paket `Algorithms.BC` sind die Implementierungen der Klassen `BCSatellite` und `BCSatelliteFactory` angegeben. Diese implementieren ebenfalls die Interfaces `Satellite` und `SatelliteFactory` aus dem Paket `STKConnection` (Abschnitt 5.2.2). Für den BC-Algorithmus wurden keine Veränderungen am Klassenlayout vorgenommen, da der algorithmenspezifische Teil ausschließlich in der zugehörigen Algorithmusklasse `BCAlgorithm` (im Paket `Algorithms`, Abschnitt 5.2.3) implementiert wurde. Trotzdem sind eigene Klassen nötig, da Interfaces keinen Code enthalten und deswegen implementiert werden müssen.

Der BC-Algorithmus erfuhr ebenfalls eine Überarbeitung und Anpassung an das Satellitennetzwerkszenario. Der in Abschnitt 3.2.7 vorgestellte Algorithmus besitzt als Parameter eine erwartete Clustergröße P, die dann die Auswahlwahrscheinlichkeit für einen Clusterhead durch $\frac{1}{P}$ definiert, eine Wartezeit t, nach der Satelliten sich zum Clusterhead proklamieren, sollten sie noch keine Nachricht von anderen Clusterheads erhalten haben, und eine Anzahl von maximalen Hops k, die ein Clusterhead entfernt sein darf, um einen Satelliten in sein Cluster aufzunehmen.

Der BC-Algorithmus, der in 3.2.7 beschrieben wurde, lässt auf jedem Satelliten einen Zufallswert (der, wie in 4.1 angemerkt wurde, mit dem Java-Zufallsgenerator erzeugt wird und eine gleichverteilte Zufallsgröße zurückgibt) berechnen, durch den manche Satelliten sich zu Clusterheads proklamieren, falls dieser Zufallswert kleiner als $1/P$ ist. Diese sogenannten "freiwilligen" Clusterheads geben diese Information an alle Satelliten weiter, die sie über k Hops erreichen können, die sich dann dem nächstgelegenen Clusterhead anschließen. Nach einem Wartezeitraum t proklamieren sich dann die Satelliten, die keine Nachricht von anderen Clusterheads erhalten haben, zu "gezwungenen" Clusterheads, und fügen die restlichen Satelliten in ihrer Nähe ihrem Cluster hinzu. In der Implementierung wird zwischen freiwilligen und gezwungenen Clusterheads bei der Clusterkonstruktion kein Unterschied gemacht, die Bezeichnung ist rein deskriptiver Natur.

In der vorliegenden Implementierung wird k auf 1 gesetzt, sodass zwischen einem potentiellen Clustermember und einem Clusterhead direkter Kontakt existieren muss, Clustermember und Clusterheads müssen also Nachbarn sein. Auf die Wartezeit t in ihrer eigentlichen Form für ein Szenario mit kontinuierlicher Zeitleiste wird nicht direkt eingegangen, da zeitdiskrete Simulationen durchgeführt werden. Stattdessen geht die Implementierung bei einem Algorithmusaufruf die Liste aller Satelliten durch, in der sich manche Satelliten wie oben beschrieben als freiwillige Clusterheads proklamieren. In einem anschließenden zweiten Listendurchlauf wird ermittelt, welche Satelliten nicht in direkter Nachbarschaft zu einem Clusterhead liegen. Diese freien Satelliten proklamieren sich dann als gezwungene Clusterheads, falls durch den zweiten Listendurchlauf keine gezwungenen Clusterheads entstanden sein sollten (ansonsten fügen sie sich dem nächstgelegenen Clusterhead zu). So "warten" also Satelliten auf Nachrichten von Clusterheads, allerdings eben keinen festen Zeitraum.

Die größte Anpassung dieses Algorithmus erfolgte durch die dynamische Änderung von P. Falls Bodenstationen in der Nähe liegen, wird P auf $P = \frac{n}{r}$ gesetzt, wobei n die Anzahl der Satelliten über der Bodenstationssammlung bezeichnet und r die Anzahl der Bodenstationen in einer solchen Sammlung. Hierdurch soll eine passende Aufteilung der Satelliten über Bodenstationen motiviert werden. Besitzt ein Satellit keinen Kontakt zu Bodenstationen, so wird P auf $P = \max\left(8, \frac{n}{2}\right)$ gesetzt. Dadurch soll möglichst eine Clustergröße erreicht werden, die den Datendurchsatz wie in 4.4.4 definiert maximiert und dabei so viele Satelliten wie möglich einbindet. Falls nicht sehr viele Satelliten im Netzwerk vorhanden sind, wie im Satellitenszenario 1 oder 2, wird diese Zahl so gesetzt, dass genügend große Cluster entstehen, die ebenfalls den Datendurchsatz maximieren und dabei keine Bandbreite verschwenden.

Da im zugehörigen Paper, das den Algorithmus vorstellt, keine Cluster Maintenance-Phase angegeben ist, wurde der Algorithmus an dieser Stelle erweitert. In der Cluster Maintenance-Phase wird zuerst pro Cluster betrachtet, welche Satelliten noch Kontakt zu ihrem Clusterhead besitzen. Falls ein Clusterhead nicht mehr alle Members erreicht, so wird das Cluster vorläufig verkleinert und eine clusterübergreifende Liste aller fehlenden Satelliten erstellt. Nachdem alle Cluster durchlaufen wurden, wird für jeden Satelliten in dieser Liste ermittelt, ob dieser Satellit Kontakt zu einem beliebigen anderen Clusterhead besitzt, dem er zugewiesen werden kann. Falls ein Clustermember keinen Kontakt zu irgendwelchen Clusterheads mehr besitzt, wird ein Reclustering eingeleitet.

5.3.3. Der PMW-Algorithmus

Die Klasse `PMWSatellite` aus dem Paket `Algorithms.PMW` besitzt für jede Größe, aus der später ein Gewicht gebildet wird, ein Attribut, sowie außerdem eine Aufzeichnung aller Ladezustände zu den Simulationszeitpunkten. Eine Besonderheit an der Implementierung dieser Satellitenklasse liegt darin, dass ein Satellit die Proklamation zum Clusterhead verweigert, wenn sein Ladezustand unter ein gewisses Level fällt, das durch das statische Attribut `powerThreshold` festgelegt ist. Sollte der Ladezustand sogar auf 0 fallen, deklariert sich der Satellit als "tot" (abgefragt durch `isDead()`) und verweigert jegliche Aktionen wie das Hinzufügen zu einem Cluster. Außerdem implementiert der Algorithmus eine gewisse Toleranzzeit, falls ein Satellit den Kontakt zu seinem Clusterhead verliert oder sich zwei Clusterheads in direkter Nachbarschaft befinden, bis ein Reclustering angestoßen wird. Diese Toleranzzeit wird in `waitsWithoutCluster` gespeichert und wird für beide Konfliktsituationen verwendet.

Die ebenfalls im Paket `Algorithms.PMW` liegende Klasse `PMWSatelliteComparator` implementiert das `Comparator`-Interface von Java und dient dazu, zwei `PMWSatellite`-Objekte bezüglich des Gewichts des repräsentierten Satelliten vergleichen zu können. Dieser Comparator wird dazu benutzt, Satellitenlisten absteigend nach dem aktuellen Gewicht auszugeben, um so leichter auf den Satelliten mit dem höchsten Gewicht beispielsweise in der Nachbarschaft zugreifen zu können. Diese Klasse dient aber einzig und allein zur Programmiererleichterung und hat keine Auswirkungen auf den Ablauf des Algorithmus selbst.

Der PMW-Algorithmus wurde hauptsächlich so implementiert, wie er in Abschnitt 3.2.10 beschrieben ist. Der PMW-Algorithmus ist ebenfalls ein gewichteter Algorithmus, der seine Prioritäten allerdings eher in die Stabilität der Cluster sowie die Langlebigkeit des Netzwerks gesetzt hat, was sich in der Wahl seiner Parameter als auch im Entwurf der Cluster Maintenance-Phase zeigt.

Der Ablauf der Cluster Setup-Phase ist ähnlich zu der des WCA-Algorithmus (siehe Abschnitt 5.3.1). Jeder Satellit j berechnet für jeden Algorithmusdurchlauf zu einem Zeitpunkt t zuerst seinen aktuellen prozentualen Ladestand $RP_j(t) \in [0;1]$, dann die Rate $PDR_j(t) = RP_j(t) - RP_j(t-\bar{t})$, mit der dieser Ladestand abnimmt und anschließend die Mobilitätsmetrik aus Formel 3.2 $MP_j(t) = \sqrt{\frac{\sum_{i=1}^{n}(RM_{ij}(t)-E[RM_{ij}(t)])^2}{n}}$. $RM_{ij}(t)$ bezeichnet hierbei die relative Mobilität eines Satelliten i im Vergleich zu einem Nachbarn j. Für diese relative Mobilität gilt immer $E[RM_{ij}(t)] = 1$. Die drei Größen $RP_j(t)$, $PDR_j(t)$ und $MP_j(t)$ werden wie bei WCA mit 3 Gewichtungsfaktoren f_1, f_2 und f_3 verrechnet und bilden so das Gesamtgewicht $W_j(t)$ des Satelliten j:

$$W_j(t) = f_1 \cdot \exp(-MP_j) + f_2 \cdot RP_j + \cdot f_3 \cdot \exp(-PDR_j) \tag{5.4}$$

Die Exponentialfunktionen dienen dem Zweck, die entsprechenden Parameter zu normieren und so mit dem verbleibenden Ladezustand besser vergleichbar zu gestalten, da so jeder Parameter vor seiner Gewichtung einen Wert aus dem Intervall $[0;1]$ annimmt.

Der Satellit mit dem höchsten Gewicht in seiner Nachbarschaft (anders als bei WCA, wo die Wahl auf den Satelliten mit dem niedrigsten Gewicht fällt) wird Clusterhead, sofern er nicht bereits einen anderen Clusterhead als Nachbarn besitzt und seine Energie ausreichend ist. Sobald alle Clusterheads ihre Rolle proklamiert haben, teilen sich die restlichen Satelliten auf diese Clusterheads auf, wobei sich ein Satellit immer dem Clusterhead mit dem höchsten Gewicht in der Nachbarschaft zuweist.

Die vergleichsweise komplexe Cluster Maintenance-Phase versucht, Reclusterings möglichst zu vermeiden, da diese sehr energieaufwändig sind. Falls ein Satellit den Kontakt zu seinem Clusterhead verliert, wartet er 3 Algorithmusdurchläufe ab. Falls er bis dahin den Kontakt zu seinem alten Clusterhead nicht aufbauen konnte, versucht er, einen anderen Clusterhead in seiner Nähe zu finden, ansonsten deklariert er sich selbst als Clusterhead, sofern seine Energie dafür ausreicht. Tut sie es nicht, wird ein Reclustering angestoßen. Falls zwei Clusterheads miteinander in direkten Kontakt kommen, warten beide Clusterheads ebenfalls für 3 Runden auf einen Kontaktabbruch. Falls danach immer noch Kontakt besteht, ordnet sich der Clusterhead mit dem niedrigeren Gewicht dem mit dem höheren Gewicht unter und tritt seinem Cluster bei. Die Satelliten aus dem nun aufgelösten Cluster suchen sich den Clusterhead in ihrer Nähe mit dem niedrigsten Gewicht, ansonsten deklarieren sie sich der Reihe nach selbst zu Clusterheads. Falls danach immer noch Satelliten ohne Clusterhead existieren, wird ebenfalls ein Reclustering angestoßen

In der Implementierung wurden verschiedene Größen festgelegt. Die Gewichtungen der Gewichtsparameter wurden auf $f_1 = 0.3$, $f_2 = 0.4$ und $f_3 = 0.3$ festgelegt, um alle Parameter möglichst gleich zu bewerten, aber ein Hauptaugenmerk auf die

Restenergie zu legen, da diese auch ausschlaggebend bei einer Wahl zum Clusterhead ist. Außerdem wurden verschiedene Größen für den Energievorrat und -verbrauch eines Satelliten festgelegt, um einen abnehmenden Ladezustand zu simulieren. Diese Größen sind rein hypothetisch. Zu Beginn der Simulation besitzt jeder Satellit einen abstrakten Energievorrat von 10 Einheiten. Falls ein Satellit ein einfacher Clustermember ist, so verbraucht er 0.05 Einheiten pro Algorithmusdurchlauf. Nimmt ein Satellit die Rolle eines Clusterheads an und verwaltet ein Cluster mit k Membern, so werden pro Algorithmusdurchlauf $0.15 + 0.05k$ Energieeinheiten verbraucht. Wird ein Reclustering durchgeführt, verbraucht jeder Satellit 0.1 Energieeinheiten. Die Ladezustandsgrenze, unter der ein Satellit die Rolle eines Clusterheads aufgibt und nicht mehr annehmen darf, wurde auf 1.5 Einheiten gesetzt. Die Wahl dieser Größen ist erst einmal rein hypothetisch, allerdings wurde versucht, auf der einen Seite möglichst niedrige Abnahmeraten für den Ladezustand zu finden, auf der anderen Seite sollte getestet werden, was passiert, wenn tatsächlich ein Clusterhead wegen Überlastung "stirbt", wie es zum Beispiel im Satellitenszenario 3 der Fall sein wird.

Die Cluster Setup- und Maintenance-Phasen wurden beinahe entwurfsgetreu umgesetzt, bis auf den Unterschied, dass in der Maintenance-Phase bereits während der Wartezeit eines clusterlosen Satelliten versucht wird, diesen einem anderen Cluster zuzuweisen.

5.3.4. Eigene Heuristik

Es soll außerdem eine eigene Heuristik "NH"[3] angegeben werden, die versucht, das Clusterproblem stabil zu lösen. Als Parameter erhält die Heuristik eine gewünschte Anzahl n an Clustern. Nun wird ähnlich zum LCA-Algorithmus (siehe Unterabschnitt 3.2.1) ein beliebiger Satellit, wenn möglich mit Kontakt zu einer Bodenstation ausgewählt, der zum Clusterhead erklärt wird. Dieser Clusterhead markiert nun die nächststehenden $\frac{k}{n}$ Satelliten als "erste Wahl" (FC), wobei k für die Gesamtanzahl an Satelliten im Netzwerk steht. Diese als FC markierten Satelliten können nicht mehr Clusterhead werden. Alle restlichen Satelliten, zu denen der Clusterhead Kontakt hat, werden als "Second Choice" (SC) markiert und können trotzdem noch Clusterheads werden. Nicht-Clusterheads können mehrfach sowohl als FC als auch als SC markiert werden. Dieser Vorgang wiederholt sich so lange, bis alle Satelliten entweder Clusterheads sind oder als FC markiert wurden. Nun versuchen alle FC-Satelliten, sich dem nähesten Clusterhead hinzuzufügen. Sobald ein Clusterhead $\frac{k}{n}$ Satelliten in seinem Cluster hat, ist er voll und nimmt keine weiteren Satelliten auf. Falls ein FC-Satellit versucht, sich einem vollen Cluster hinzuzufügen, wird er abgewiesen und sucht den nächsten Clusterhead, der am nähesten zu ihm liegt, bis er einem Cluster hinzugefügt wird. Findet ein Satellit kein Cluster, nachdem alle CHs angefragt wurden, so beginnt er die Suchrunde von neuem und sucht nach dem nächsten SC-CH. Sollte immer noch kein Cluster gefunden worden sein, so fügt sich der Satellit mit einem "Override"-Befehl dem nächsten Clusterhead hinzu.

Die Maintenance-Phase besteht ausschließlich daraus, einen Satelliten bei Kontaktverlust zum Clusterhead entweder einem neuen Clusterhead zuzuordnen, oder

[3] NH steht für *Nearest-Heuristik*

im Falle eines Kontaktverlusts zu allen Clusterheads neu zu clustern.

Die Klasse `MySatellite` im Paket `Algorithms.My` implementiert die oben angegebenen Parameter sowie die Beschränkungen für die Clustermember, die einem Clusterhead zugewiesen werden können. Das Besondere an der Implementierung der `MySatellite`-Klasse ist die Einführung einer Clustergrößenbeschränkung, die es in dieser Form bei keinen anderen Algorithmen gibt. Dadurch werden Clustermember auch direkt an einen Clusterhead zugewiesen anstelle in einem Clusterobjekt als Member behandelt zuwerden.

Kapitel 6.

Analyse der ausgewählten Algorithmen

Auf den nachfolgenden Seiten sind die Ergebnisgrafiken für jede Szenariokonfiguration angegeben. Die Erklärungen und die zugehörige Diskussion finden sich immer am Ende eines Unterabschnitts. Die Legende ist rechts neben den Graphen zu sehen.

In den Simulationen im Kurzzeitszenario (Abschnitte 6.1.1 bis 6.1.4) wurde über den Zeitraum eines Tages alle acht Minuten eine Momentaufnahme betrachtet, während bei den Simulationen im Langzeitszenario (Abschnitte 6.2.1 bis 6.2.4) über den Zeitraum von zwei Monaten alle 8 Stunden Momentaufnahmen betrachtet wurden.

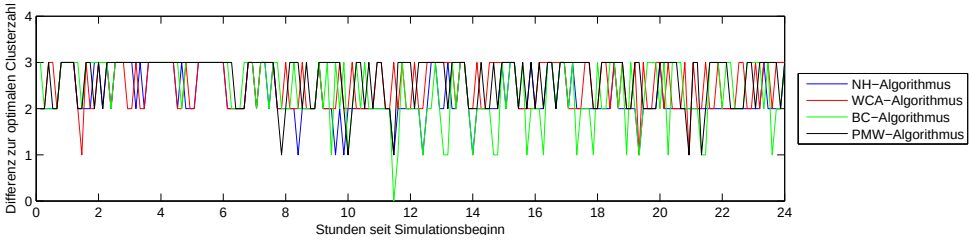
Falls ein Ergebnisgraph eines Algorithmus nicht zu sehen ist, liegt das daran, dass dieser Algorithmus an den entsprechenden Messzeitpunkten die gleichen Ergebnisse geliefert hat wie ein anderer Algorithmus und von dessen Graphen überzeichnet wurde.

Der erste Teil jedes Unterabschnitts beinhaltet die Diagramme für die Clusterkontakte zu den einzelnen Bodenstationen sowie die insgesamte Anzahl an gebildeten Clustern. Im jeweiligen zweiten Teil sind schließlich die Diagramme mit der mittleren Clustergröße, der Standardabweichung der Clustergrößen von der mittleren Clustergröße, den kumulierten Reclusterings (durchgezogene Linie) und kumulierten Reassignments (gestrichelte Linie), dem kumulierten Datendurchsatz und dem mittleren Datendurchsatz zu sehen.

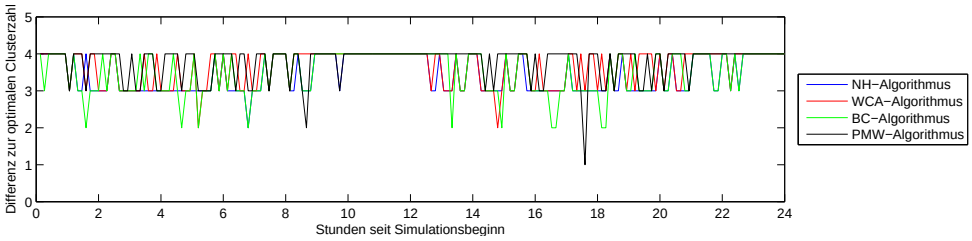
Die Formeln, mit denen die Ergebnisse in den einzelnen Diagrammen berechnet wurden, finden sich alle in Abschnitt 4.4.

6.1. Kurzzeitszenario

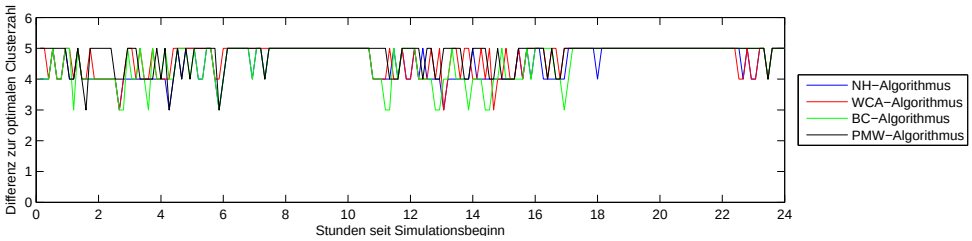
6.1.1. Satellitenszenario 1



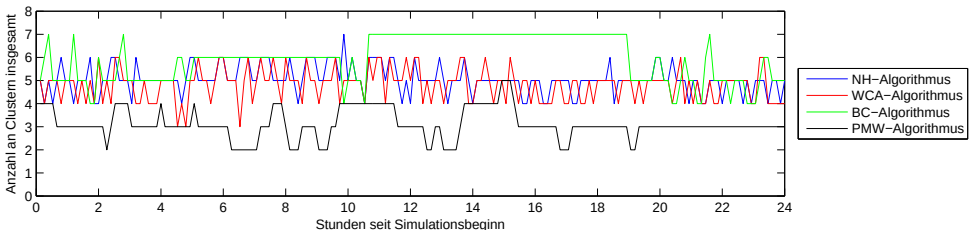
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



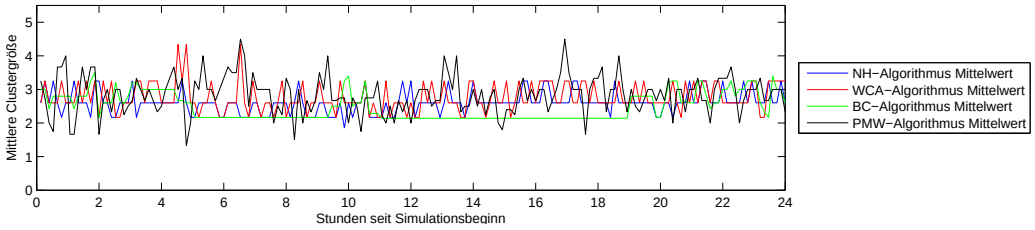
(c) Differenz zur optimalen Clusterzahl über Asien



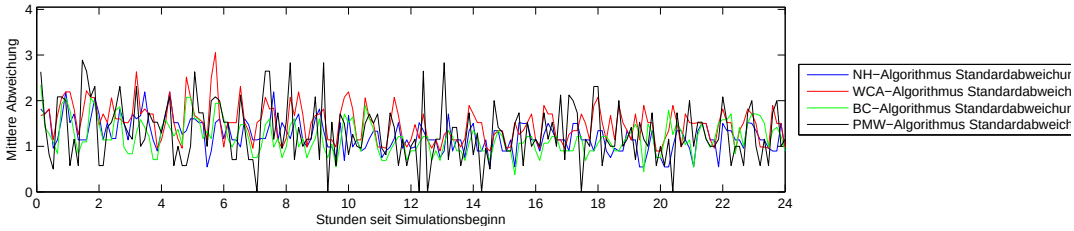
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.1.: Auswertungsgraphen des Kurzzeitszenarios mit 13 Satelliten

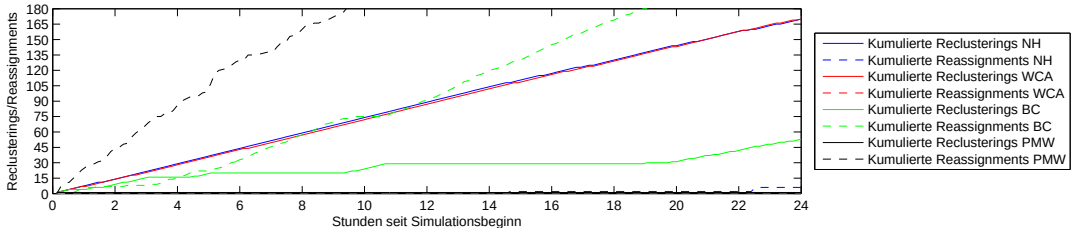
Die mittlere Clustergröße (Abb. 6.2a) liegt bei allen Algorithmen einigermaßen stabil zwischen 2 und 3 Satelliten, wobei PMW und WCA größere Ausschläge nach oben verzeichnen. Die mittlere Abweichung der Clustergrößen liegt nicht höher als maximal 3 Satelliten (Abb. 6.2b), der Trend geht sogar dahin, dass sich die mittlere Abweichung mit der Zeit um den Wert 1 bewegt. Durch den besseren Bodenstationskontakt (gerade in Europa, Abb. 6.1a bis 6.1c) und die größere Anzahl an ausreichend großen Clustern (Abb. 6.1d) kann der NH-Algorithmus einen besseren Datendurchsatz erreichen als andere Algorithmen (Abb. 6.2d, 6.2e). PMW schneidet beim Durchsatz



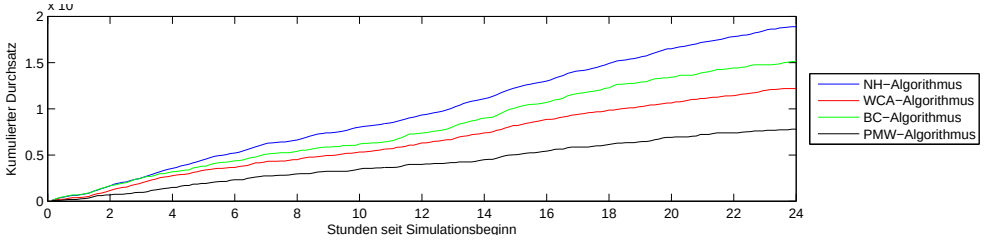
(a) Mittlere Clustergröße



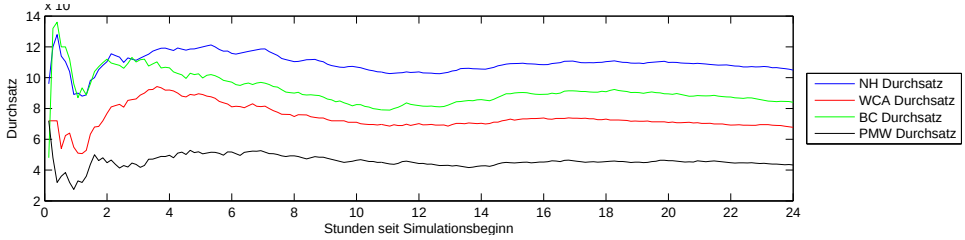
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



(d) Kumulierter Datendurchsatz

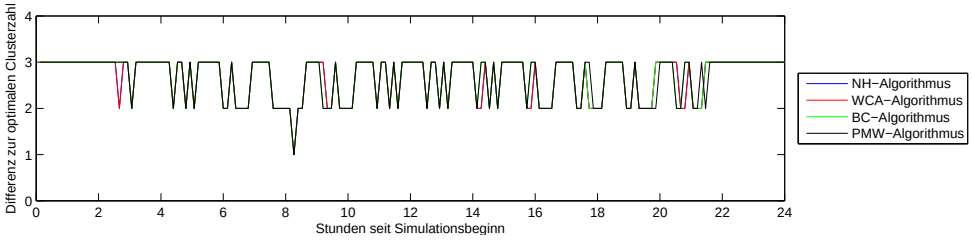


(e) Mittlerer Durchsatz

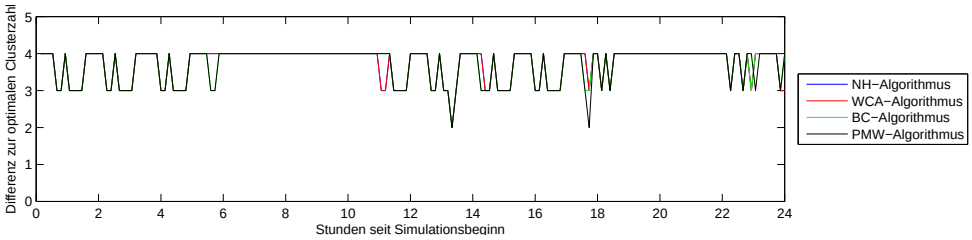
Abb. 6.2.: Auswertungsgraphen des Kurzzeitszenarios mit 13 Satelliten

deutlich am schlechtesten ab. Abb. 6.2c zeigt, dass höchstens der BC-Algorithmus in der Lage ist, eine stabile Clusterstruktur aufzubauen.

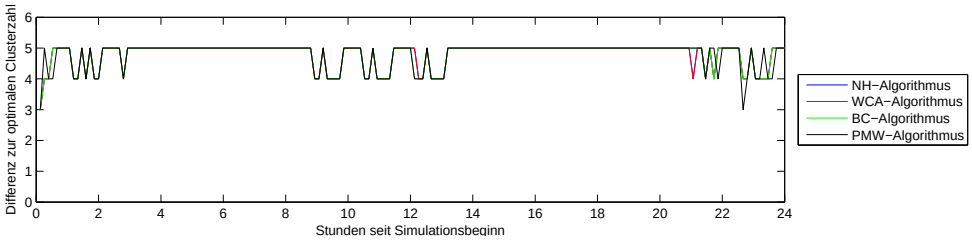
6.1.2. Satellitenszenario 2



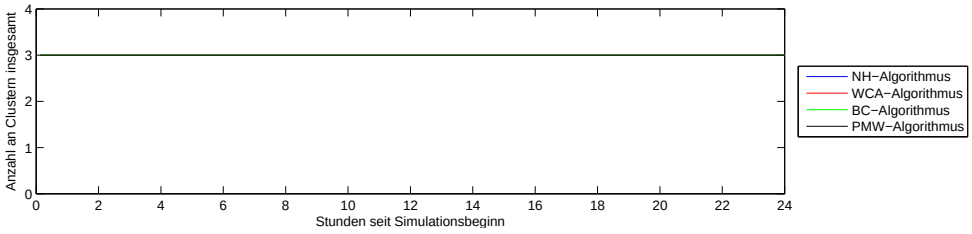
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



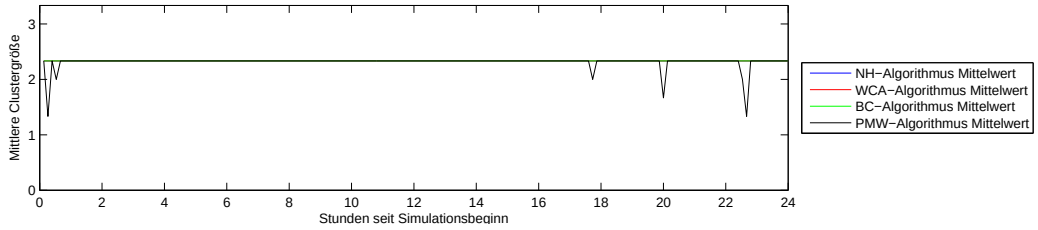
(c) Differenz zur optimalen Clusterzahl über Asien



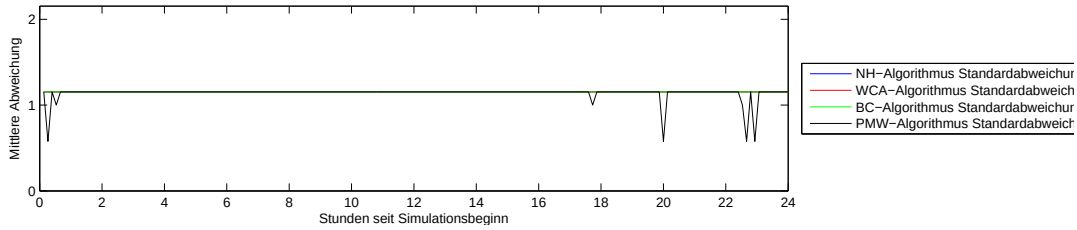
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.3.: Auswertungsgraphen des Kurzzeitszenarios mit 7 Satelliten

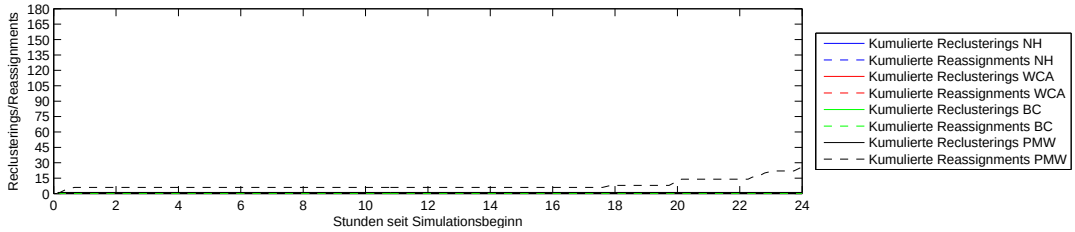
Aufgrund der schwachen Mobilität im kurzen Zeitraum dieses Szenarios ist es nicht verwunderlich, dass die Clustergröße sowie die Clusterzahl stabil bleiben (Abb. 6.3d, 6.4a, 6.4b). Die Einbrüche des PMW-Algorithmus liegen an der Wartezeit, die ein Satellit ohne Clusterhead verbringen kann. Tatsächlich verhalten sich BC- und NH-Algorithmus in diesem Szenario im Mittel vollkommen gleich. Der einzige Unterschied, der in den Diagrammen nicht erkennbar ist, besteht in der Clusteraufteilung. Auch wurde kein einziges Mal neu geclustert, einzig der PMW-Algorithmus hat Reassignments durchgeführt (6.4c). Da sich alle Algorithmen hier beinahe gleich verhalten, unterscheiden sich die verschiedenen Durchsatzraten auch nur minimal (Abb. 6.4d und 6.4e). Sichtbare Unterschiede in der Clusteraufteilung zeigen sich in den Boden-



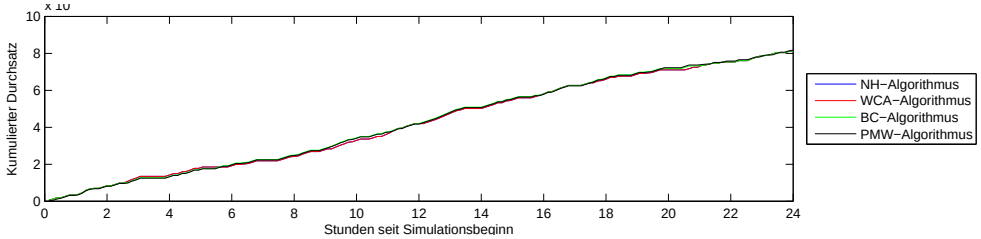
(a) Mittlere Clustergröße



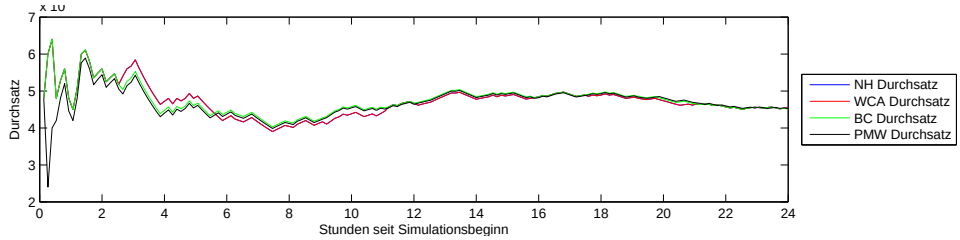
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



(d) Kumulierter Datendurchsatz

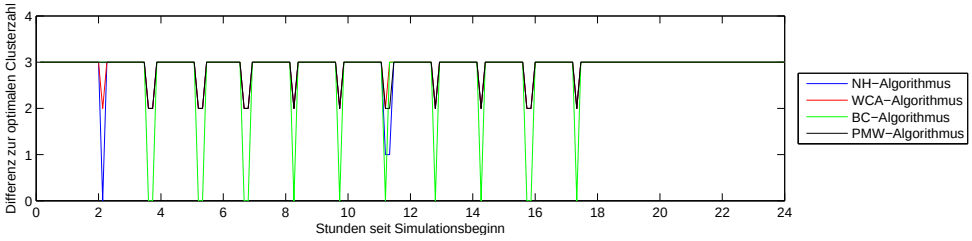


(e) Mittlerer Durchsatz

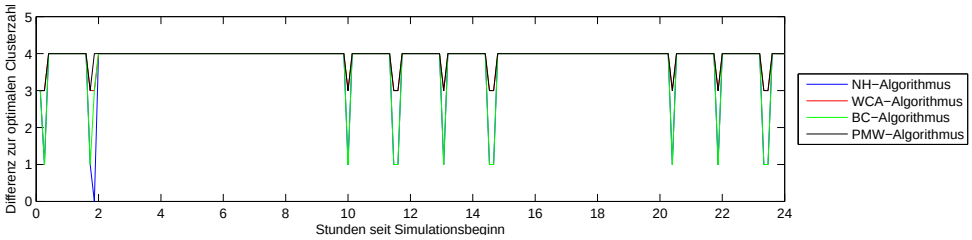
Abb. 6.4.: Auswertungsgraphen des Kurzzeitszenarios mit 7 Satelliten

stationsdiagrammen: Die Abb. 6.3a, 6.3b und 6.3c zeigen, dass der WCA-Algorithmus an manchen Stellen ein Cluster erzeugt, das Kontakt zu Bodenstationen besitzt. Entsprechend besitzt der WCA an späteren Stellen keinen Bodenstationskontakt, wo dann der BC-, PMW- und NH-Algorithmus wiederum Kontakt aufbauen können.

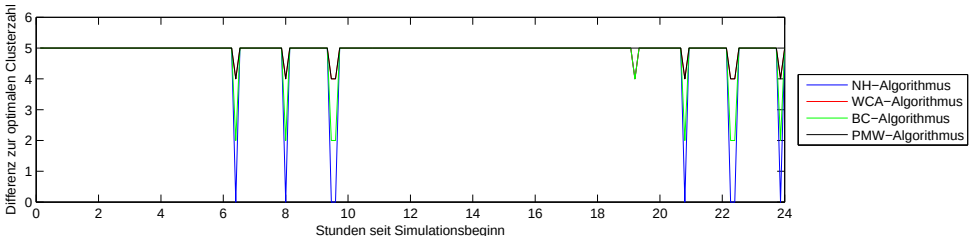
6.1.3. Satellitenszenario 3



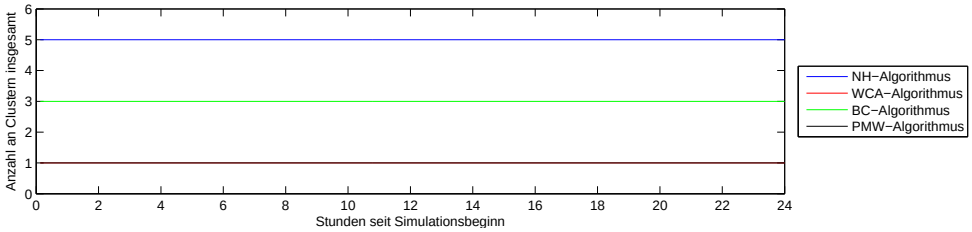
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



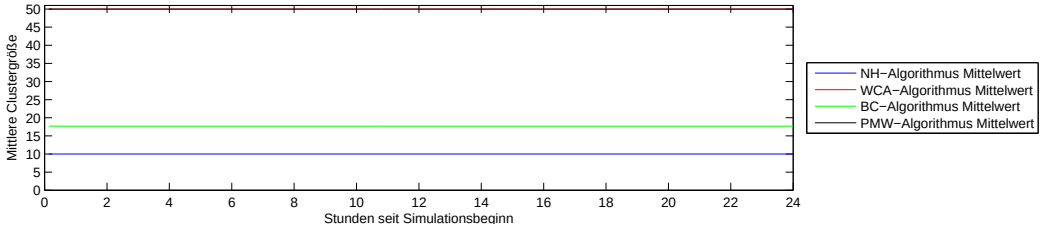
(c) Differenz zur optimalen Clusterzahl über Asien



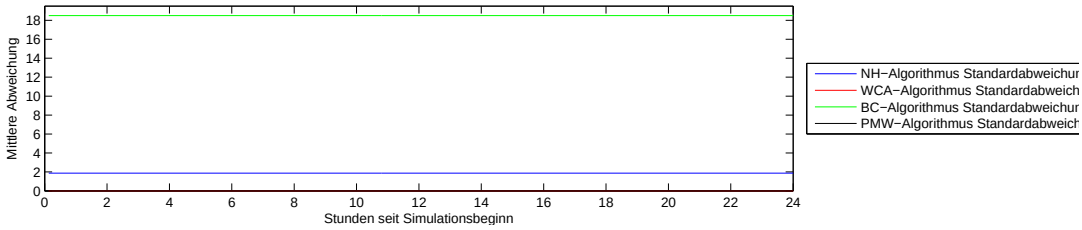
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.5.: Auswertungsgraphen des Kurzzeitszenarios mit 50 Satelliten in der Abwurfkonfiguration

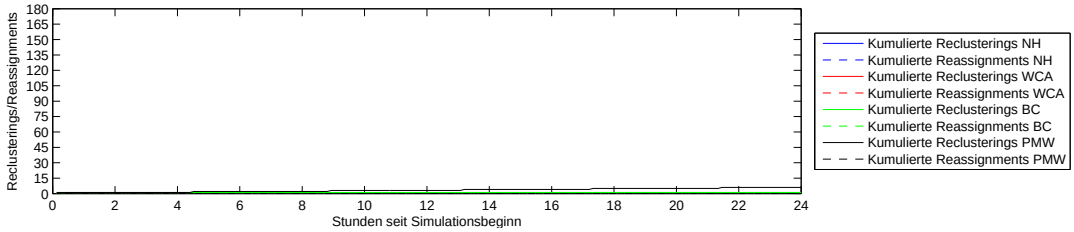
Gerade bei dieser Simulation mit geringer Dynamik und hoher Satellitendichte schneidet der NH-Algorithmus in allen Bereichen bei weitem am besten ab. Durch die Beschränkung der Clustergröße auf 10 Satelliten (Abb. 6.5d) werden 5 in etwa gleich große Cluster gebildet (Abb. 6.6a, 6.6b), die so auch sehr viel besseren Bodenstationskontakt aufbauen können (Abb. 6.5a bis 6.5c), was zu einem erheblich erhöhten Durchsatz führt (Abb. 6.6d, 6.6e). Der BC-Algorithmus bietet hier den Vorteil, dass seine Clusterheads willkürlich ausgewählt werden und so auch in direktem Kontakt stehen, wodurch er das Cluster von 50 Satelliten ebenfalls aufteilen kann, allerdings nicht so gut wie der NH-Algorithmus. Außerdem besitzt der BC-Algorithmus stark



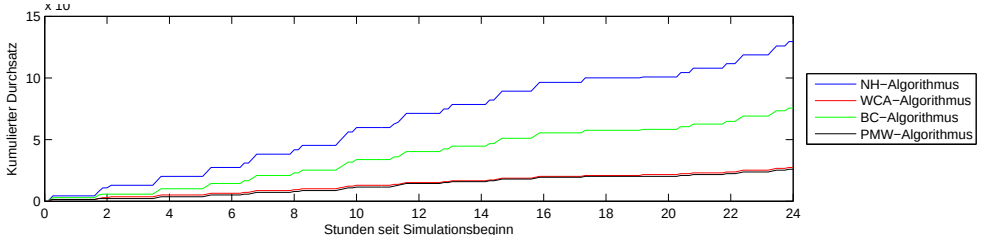
(a) Mittlere Clustergröße



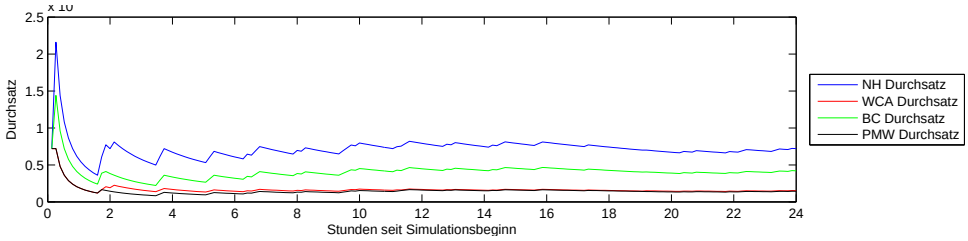
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



(d) Kumulierter Datendurchsatz

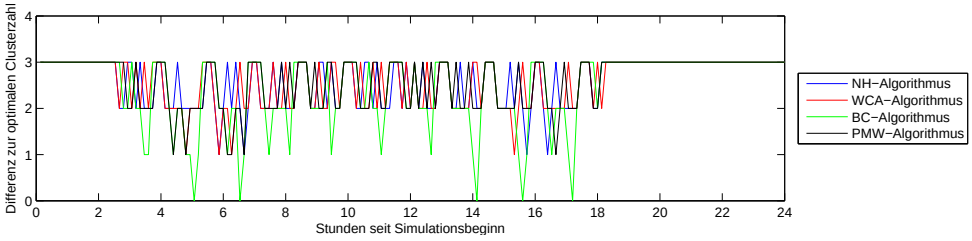


(e) Mittlerer Durchsatz

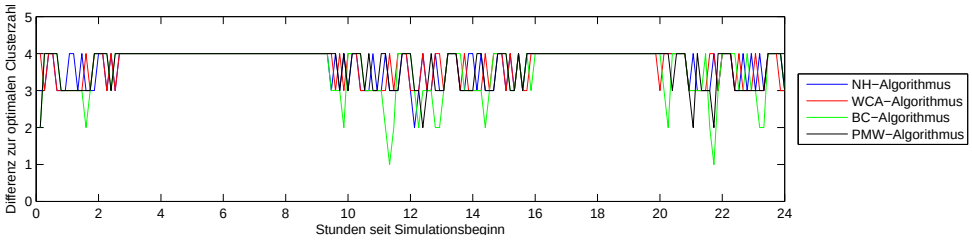
Abb. 6.6.: Auswertungsgraphen des Kurzzeitszenarios mit 50 Satelliten in der Abwurfkonfiguration

in der Größe variierende Cluster (Abb. 6.6b), teilt die Satelliten also sehr ungleich auf die Cluster auf.

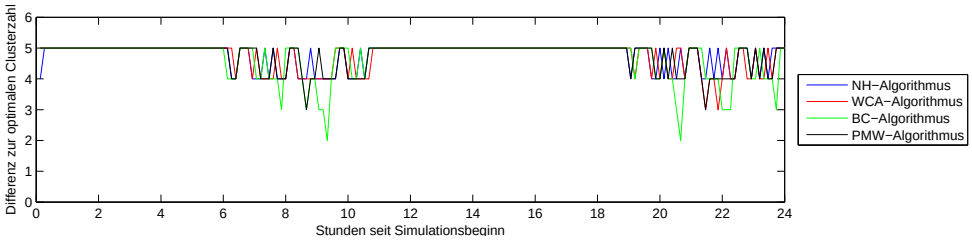
6.1.4. Satellitenszenario 4



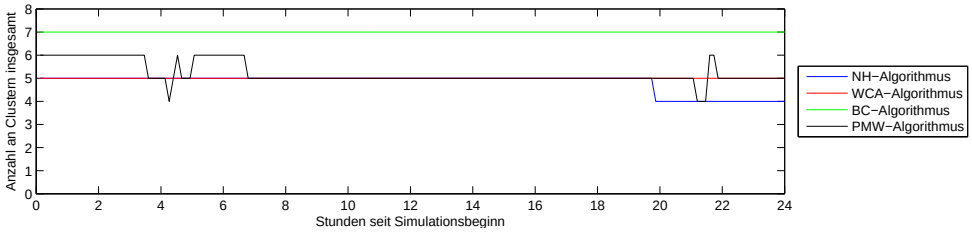
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



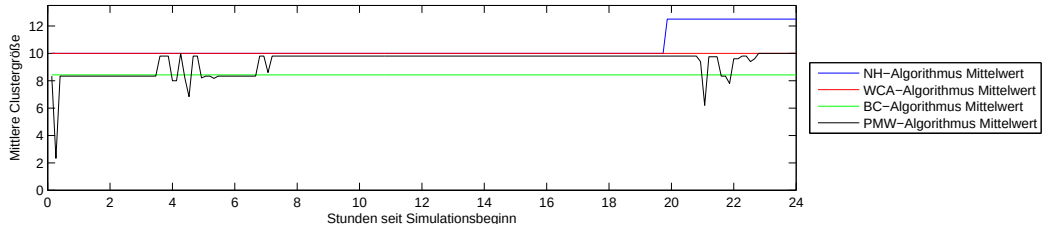
(c) Differenz zur optimalen Clusterzahl über Asien



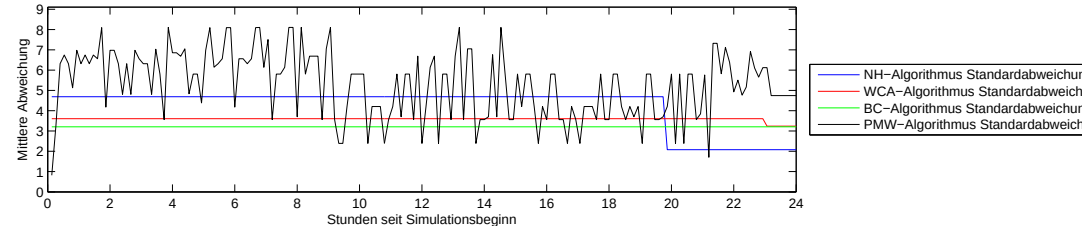
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.7.: Auswertungsgraphen des Kurzzeitszenarios mit 50 Satelliten im Perlenschnurszenario

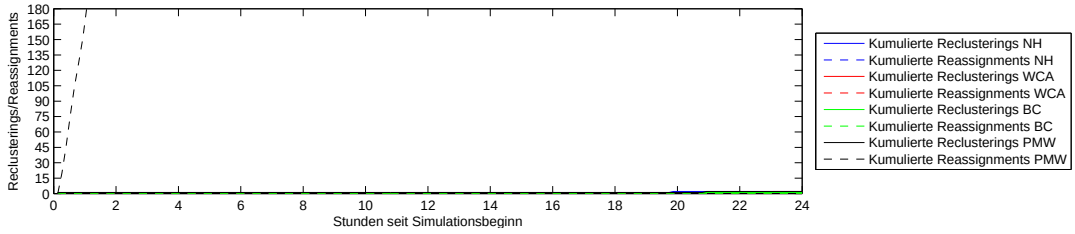
In diesem Szenario wurde beinahe nicht neu geclustert (Abb. 6.8c), die mittlere Clustergröße bleibt also sehr stabil (Abb. 6.8a), ebenso die Anzahl der Cluster (Abb. 6.7d), was sehr bemerkenswert ist, da das Perlenschnurszenario sehr mobil ist. Einzig der PMW-Algorithmus hat sehr instabile Cluster (Abb. 6.8b und 6.8c). Durch die stabilen Cluster (Abb. 6.8a, 6.8b, die beim BC-Algorithmus gebildet werden, erzielt der BC-Algorithmus in diesem Szenario eine weitaus bessere Bodenstationsabdeckung (Abb. 6.7a bis 6.7c) als die anderen Algorithmen und liegt dadurch beim Datendurchsatz klar vorne (Abb. 6.8d, 6.8e). WCA- und NH-Algorithmus verhalten sich in diesem Szenario relativ ähnlich, der NH-Algorithmus führt an einer Stelle ein



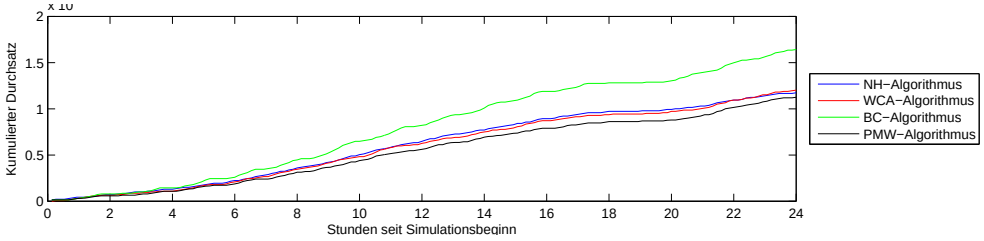
(a) Mittlere Clustergröße



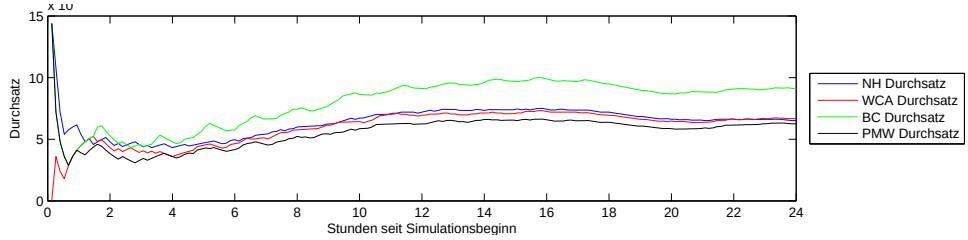
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



(d) Kumulierter Datendurchsatz



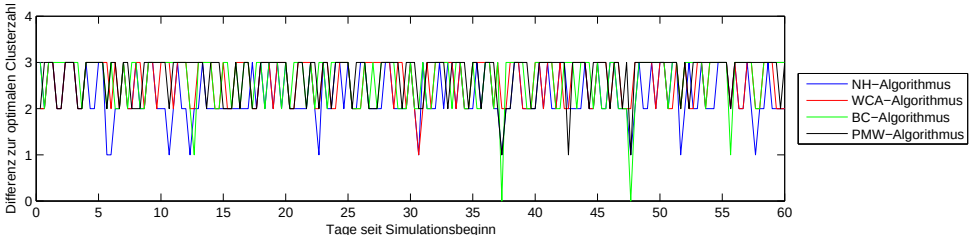
(e) Mittlerer Durchsatz

Abb. 6.8.: Auswertungsgraphen des Kurzzeitszenarios mit 50 Satelliten im Perlenschnurszenario

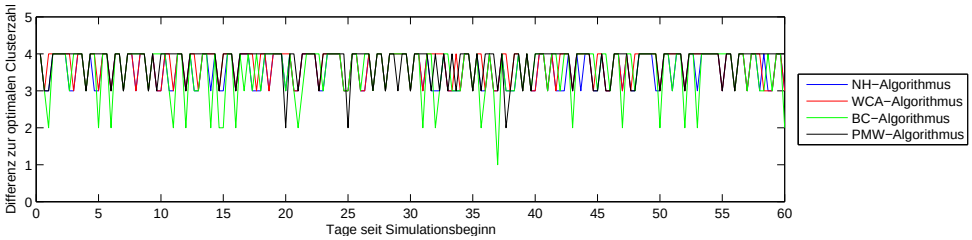
Reclustering durch, ansonsten sind die erzeugten Cluster ähnlich stabil wie die des WCA- und des BC-Algorithmus.

6.2. Langzeitszenario

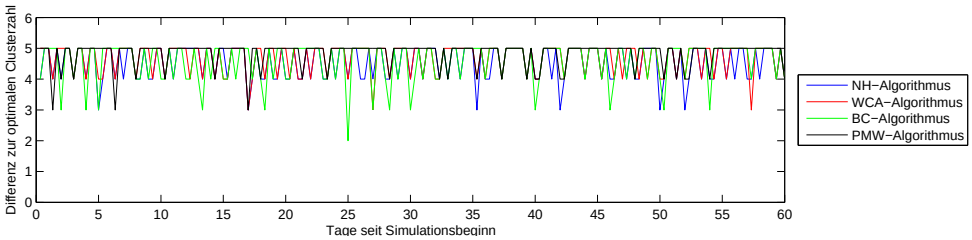
6.2.1. Satellitenszenario 1



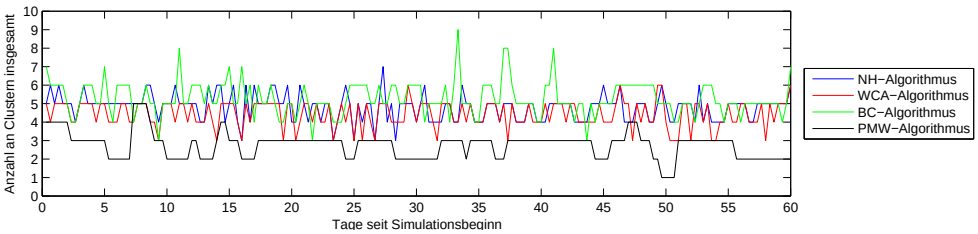
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



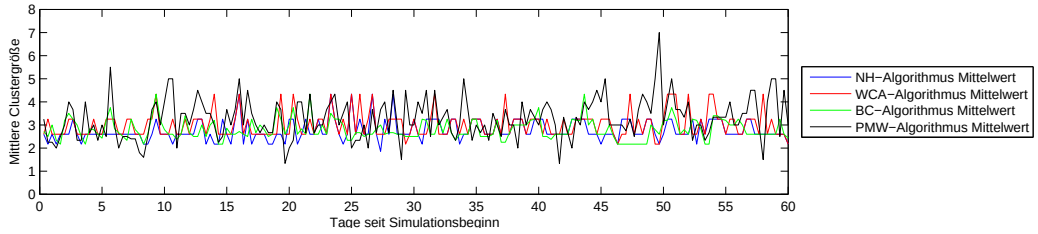
(c) Differenz zur optimalen Clusterzahl über Asien



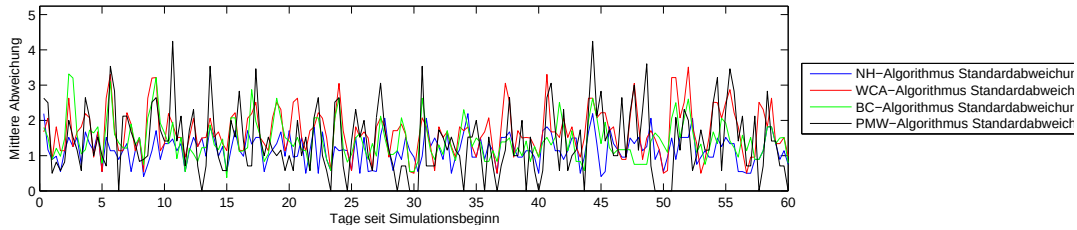
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.9.: Auswertungsgraphen des Langzeitszenarios mit 13 Satelliten

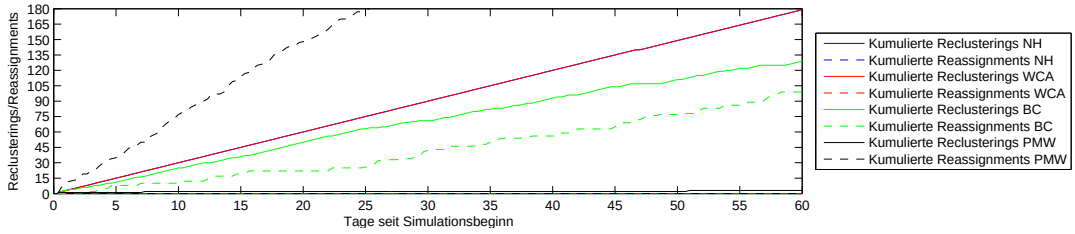
An den Ergebnisgrafiken ist nun deutlich bemerkbar, dass das Satellitenszenario 1 einen sehr hohen Mobilitätsgrad aufweist. Während der PMW-Algorithmus damit nur schwer zurechtkommt und nur wenige Cluster bildet (Abb. 6.9d), die recht instabil sind (Abb. 6.10b), verhält sich der WCA-Algorithmus trotzdem noch sehr stabil, indem er zwischen 3 und 5 Cluster bildet (Abb. 6.9d), deren mittlere Clustergröße auch sehr beschränkt bleibt (Abb. 6.10a), obwohl durchgängig neu geclustert wird (Abb. 6.10c). Der NH-Algorithmus bildet auch hauptsächlich zwischen 4 und 6 Clustern, deren Größe im Allgemeinen ebenfalls recht stabil bleibt. Der BC-Algorithmus



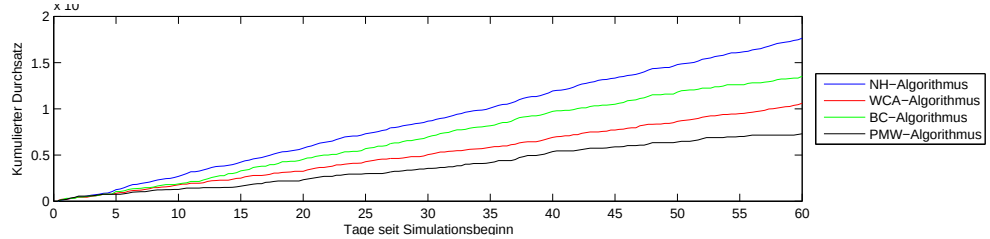
(a) Mittlere Clustergröße



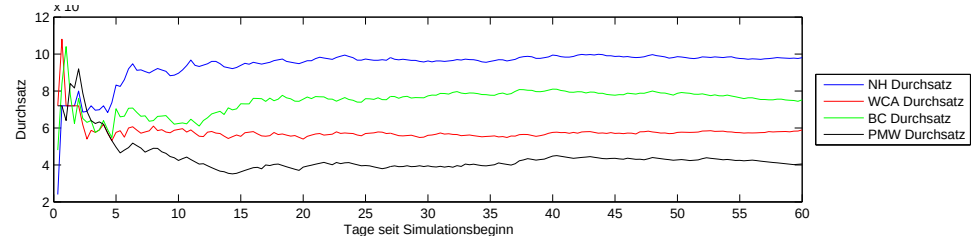
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



(d) Kumulierter Datendurchsatz

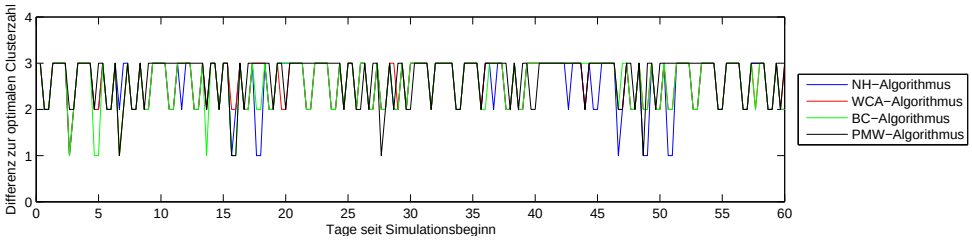


(e) Mittlerer Durchsatz

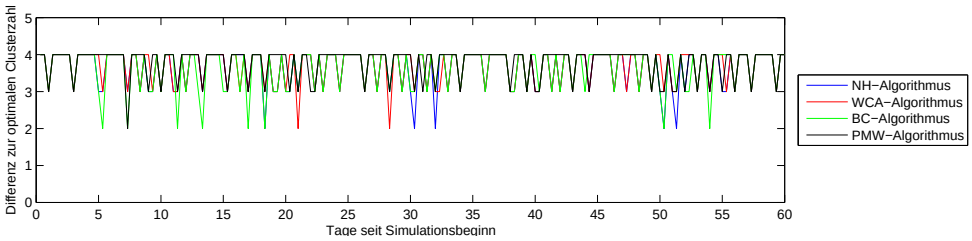
Abb. 6.10.: Auswertungsgraphen des Langzeitszenarios mit 13 Satelliten

hingegen bildet sehr viele kleine Cluster, wodurch zwar vergleichsweise viele Cluster Bodenstationskontakt besitzen (Abb. 6.9a bis 6.9c), durch die geringe Clustergröße kann er allerdings beim Datendurchsatz nicht gegen den NH-Algorithmus bestehen (Abb. 6.10d, 6.10e), obwohl weniger neugeclustert (Abb. 6.10c) wird.

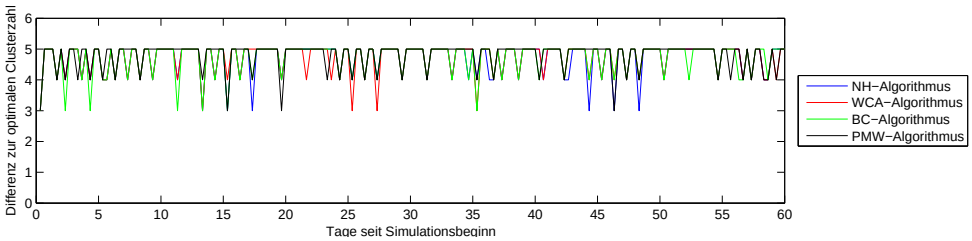
6.2.2. Satellitenszenario 2



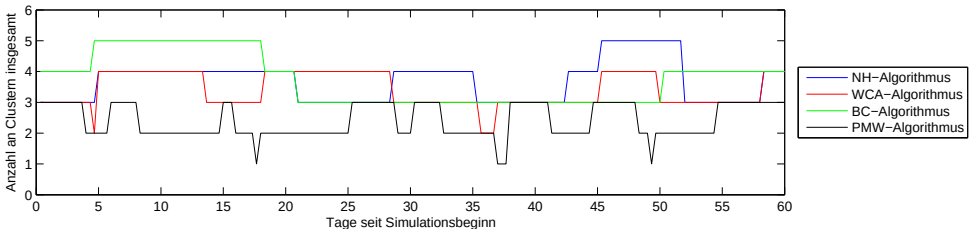
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



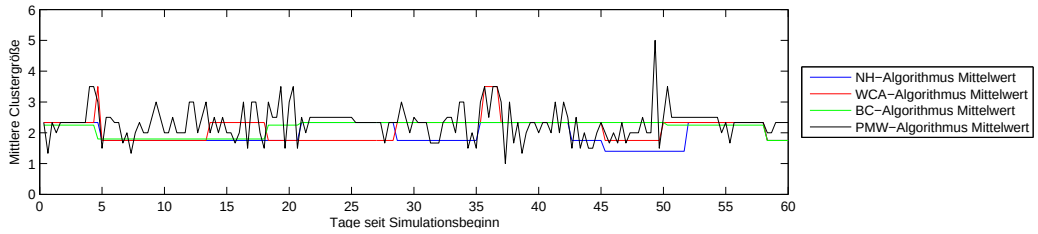
(c) Differenz zur optimalen Clusterzahl über Asien



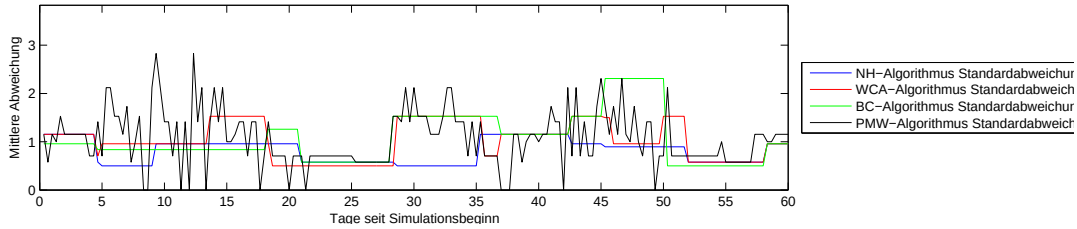
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.11.: Auswertungsgraphen des Langzeitszenarios mit 7 Satelliten

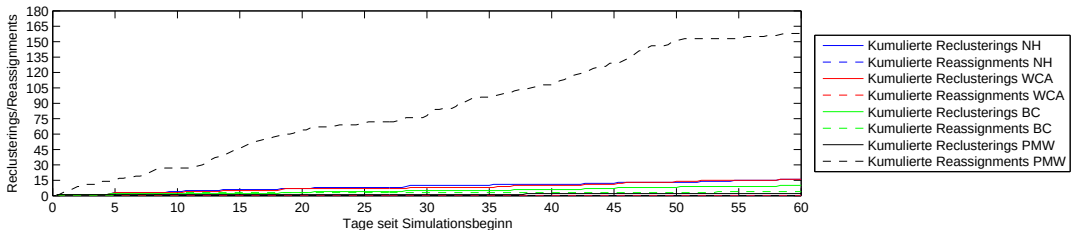
Die mittlere Clustergröße für alle Algorithmen hält sich in einem Bereich von 2 bis 3 Satelliten (Abb. 6.12a), wobei der PMW-Algorithmus auch Cluster mit bis zu 5 Satelliten erstellt. Das Mittel der Clustergrößen des PMW-Algorithmus ist ebenso instabil wie die zugehörige mittlere Abweichung (Abb. 6.12b), was von den häufigen Reassignments kommt, die der Algorithmus durchführt (Abb. 6.12c). Der Bodenstationskontakt ist besonders beim BC-Algorithmus gut (Abb. 6.11a bis 6.11c), der recht oft mit 2 Bodenstationen in einer Bodenstationsmenge Kontakt herstellen kann und zu Beginn des Szenarios bis zu 5 Cluster erzeugt (Abb. 6.11d). Entsprechend erreicht der BC-Algorithmus auch den höchsten Durchsatz, obwohl der NH-Algorithmus gegen Ende aufholen kann (Abb. 6.12d, 6.12e).



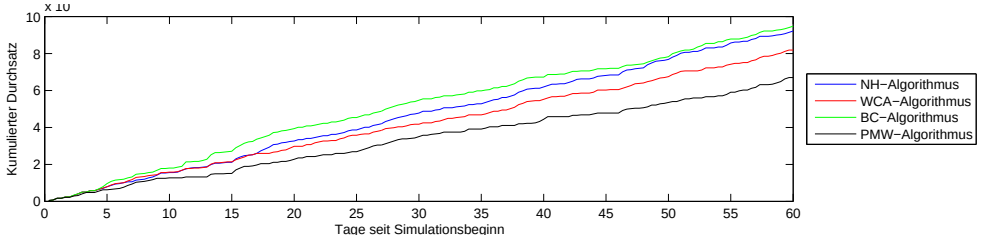
(a) Mittlere Clustergröße



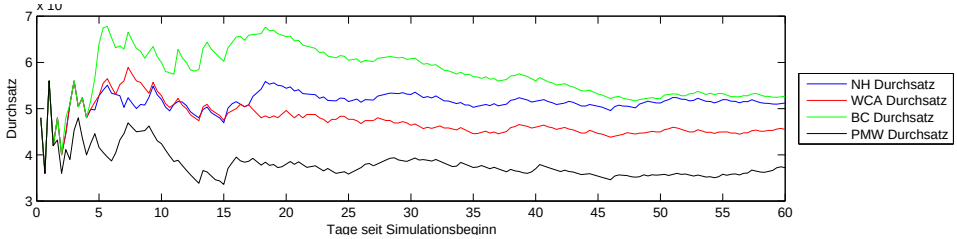
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



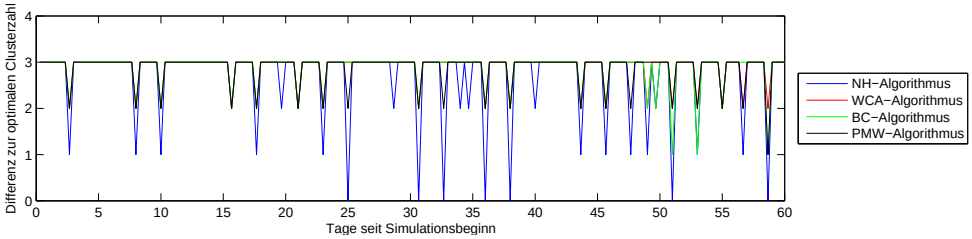
(d) Kumulierter Datendurchsatz



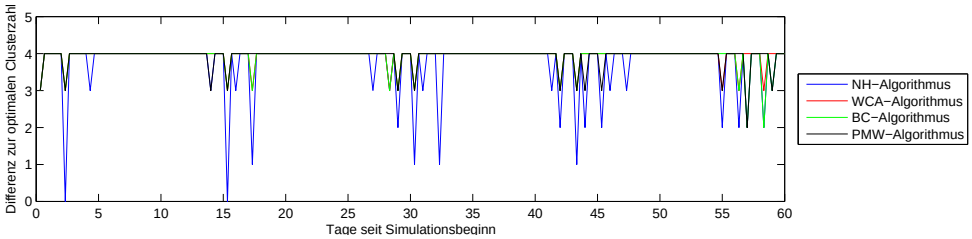
(e) Mittlerer Durchsatz

Abb. 6.12.: Auswertungsgraphen des Langzeitszenarios mit 7 Satelliten

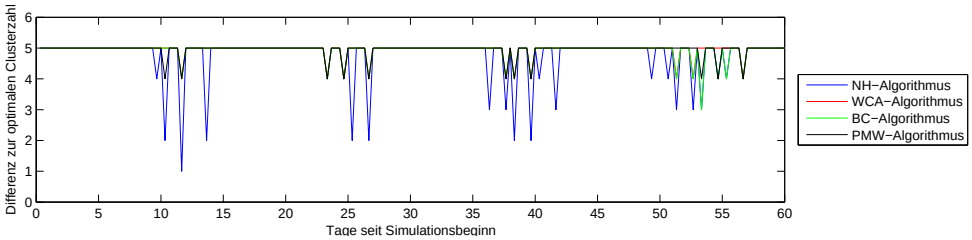
6.2.3. Satellitenszenario 3



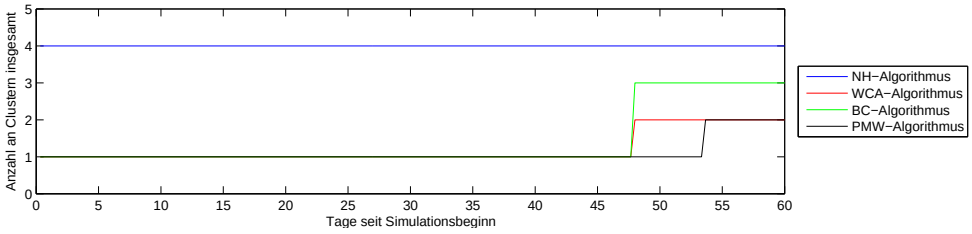
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



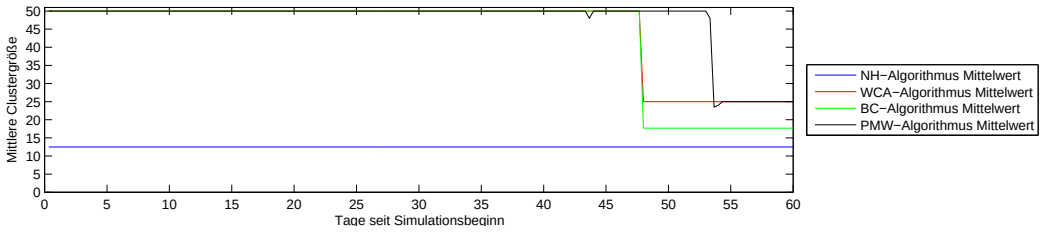
(c) Differenz zur optimalen Clusterzahl über Asien



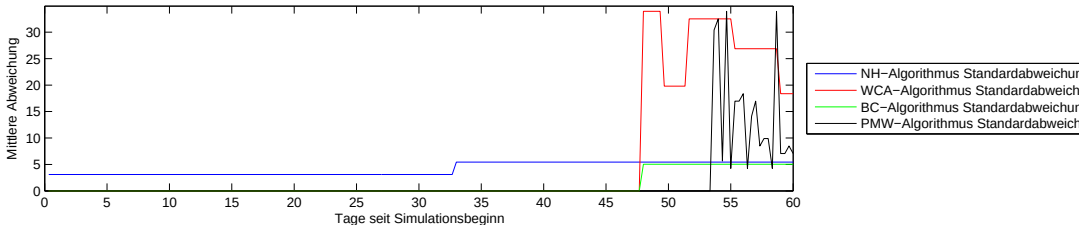
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.13.: Auswertungsgraphen des Langzeitszenarios mit 50 Satelliten in der Auswurfkonfiguration

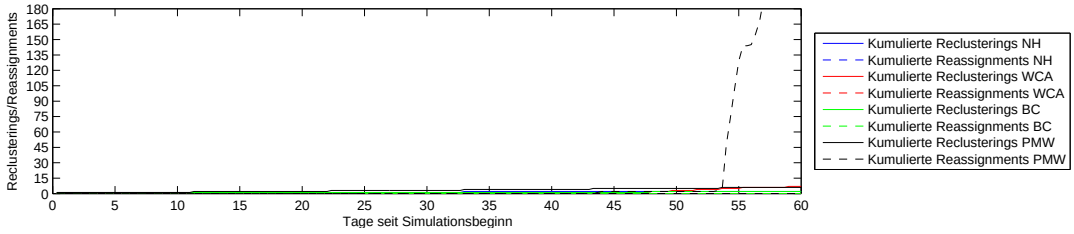
Wie im Kurzzeitszenario 6.1.3 schneidet der NH-Algorithmus hier mit Abstand am besten ab. Durch die höhere Clusterzahl (Abb. 6.13d) und kleine Clustergrößen (Abb. 6.14a, 6.14b) wird ein immens hoher Datendurchsatz erzeugt, durch den der NH-Algorithmus seine Konkurrenten weit hinter sich lässt, wogegen auch das Auffächern des Satellitenhaufens und das damit verbundene Neuclustering aller Algorithmen nicht hilft (Abb. 6.14c). Der kleine Zacken in Abb. 6.14a beim PMW-Algorithmus zeigt an, dass der Clusterhead nicht mehr genügend Energie hatte, um seine Rolle beizubehalten, woraufhin ein neuer Clusterhead ausgewählt wurde, dem sich der alte dann angeschlossen hat. Auch benötigt der PMW-Algorithmus einige Zeit (Abb.



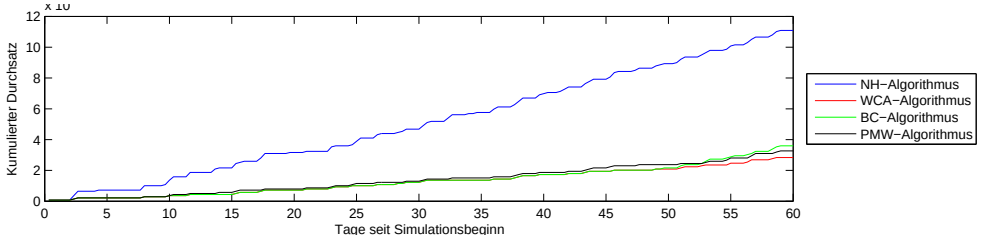
(a) Mittlere Clustergröße



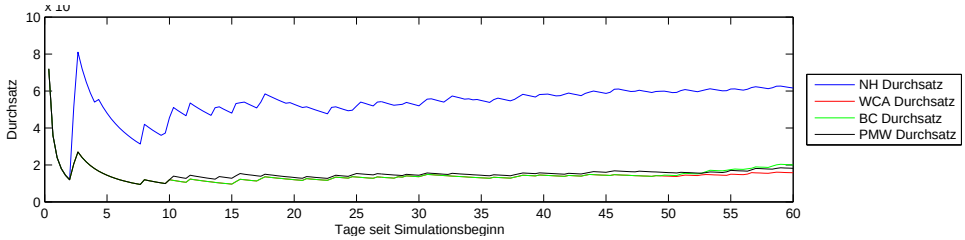
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



(d) Kumulierter Datendurchsatz

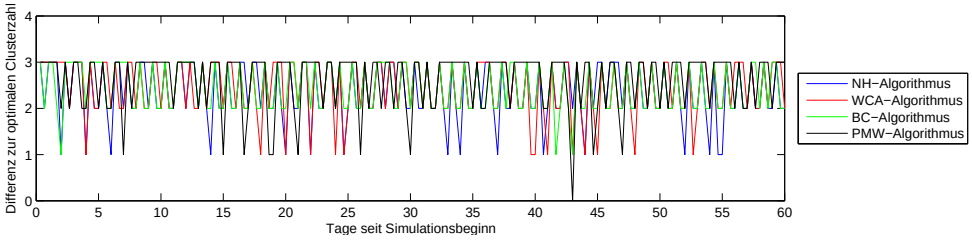


(e) Mittlerer Durchsatz

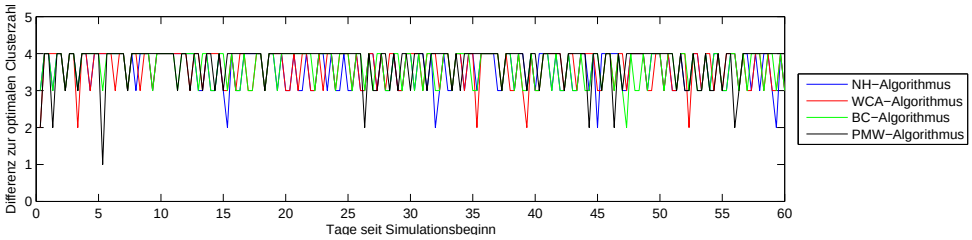
Abb. 6.14.: Auswertungsgraphen des Langzeitszenarios mit 50 Satelliten in der Auswurfkonfiguration

6.14b), um alle Satelliten gleichmäßig auf die 2 entstandendenen Cluster (Abb. 6.14a) aufzuteilen.

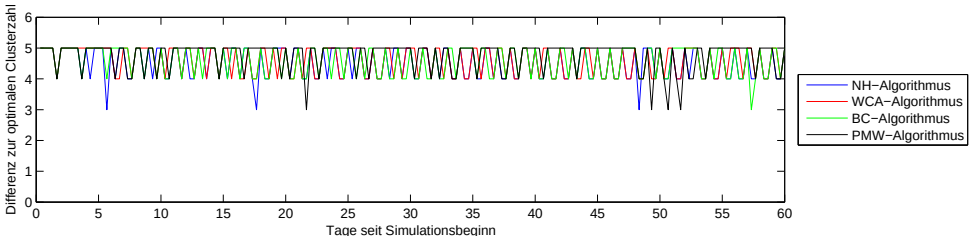
6.2.4. Satellitenszenario 4



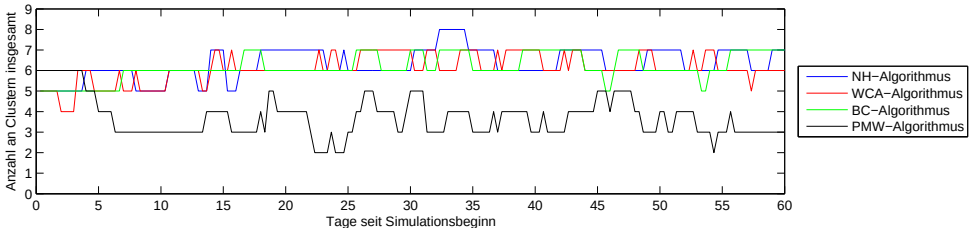
(a) Differenz zur optimalen Clusterzahl über Europa



(b) Differenz zur optimalen Clusterzahl über den USA



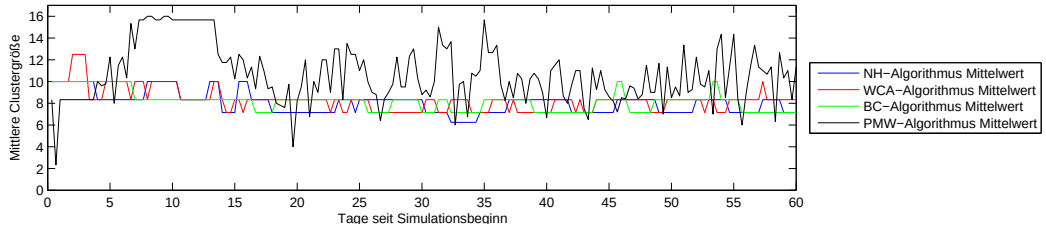
(c) Differenz zur optimalen Clusterzahl über Asien



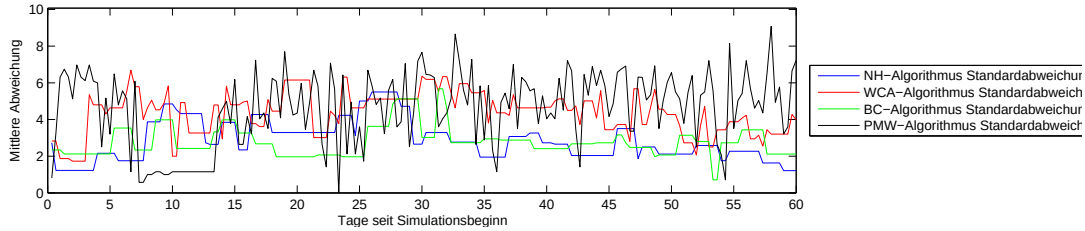
(d) Insgesamte Anzahl an gebildeten Clustern

Abb. 6.15.: Auswertungsgraphen des Langzeitszenarios mit 50 Satelliten in der Perlenschnurkonfiguration

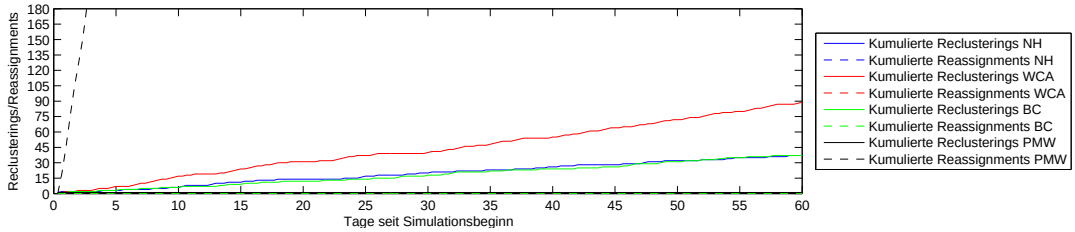
Alle Algorithmen bis auf den PMW-Algorithmus erzeugen in diesem Szenario zwischen 5 und 7 Cluster (Abb. 6.15d), deren mittlere Größe recht stabil bei 8 Satelliten liegt (Abb. 6.16a), wobei gerade beim WCA-Algorithmus die Clustergröße um bis zu 6 Satelliten variieren kann (Abb. 6.16b). Die Bodenstationsabdeckung aller Algorithmen ist ziemlich mäßig (Abb. 6.15a bis 6.15c), wobei trotzdem durchgehend Bodenstationskontakt von mindestens einem Cluster besteht. BC- und NH-Algorithmus führen recht wenige Reclusterings aus, WCA doppelt so viele, während PMW extrem viele Reassignments durchführt. Durch die Clustergrößen und ihre in der zweiten Hälfte des Simulationsintervalls geringe Varianz besitzen NH-, BC- und WCA-Algorithmus



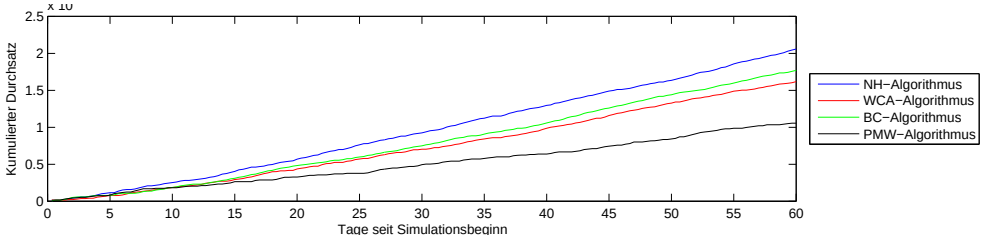
(a) Mittlere Clustergröße



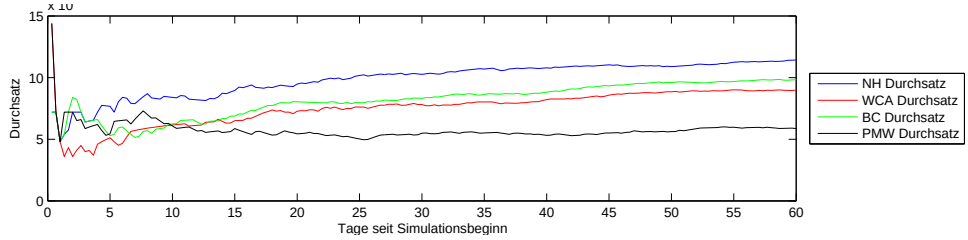
(b) Standardabweichung der Clustergröße



(c) Kumulierte Reclusterings und Reassignments



(d) Kumulierter Datendurchsatz



(e) Mittlerer Durchsatz

Abb. 6.16.: Auswertungsgraphen des Langzeitszenarios mit 50 Satelliten in der Perlenschnurkonfiguration

einen guten Datendurchsatz (Abb. 6.16e, 6.16d), der PMW-Algorithmus schneidet allerdings wieder schlecht ab.

Kapitel 7.

Diskussion

Abschließend sollen alle Algorithmen in ihrer Performance über die verschiedenen Satellitenszenarien betrachtet und mittels der in Abschnitt 4.4 definierten Vergleichsparameter verglichen und bewertet werden.

7.1. PMW-Algorithmus

Bis auf drei Zweitpunkte (einmal in Abb. 6.1b, einmal in Abb. 6.15a und einmal in Abb. 6.15b) ist es dem PMW-Algorithmus unmöglich, mehr als zwei Cluster Kontakt zu Bodenstationen aufbauen zu lassen, wobei meistens sogar nur ein einziges Cluster Kontakt aufbauen kann und oft auch gar kein Kontakt möglich ist. In den Kurzzeitmessungen ist allerdings zu sehen, dass der Kontakt meist über längere Zeit gehalten werden kann oder kurz nach einem Abbruch wieder möglich ist, wie man in den Abbildungen 6.1, 6.3 und 6.7 (jeweils Teilgrafiken (a) bis (c)) sehen kann. Die Kontaktzeiten in den Langzeitszenarien sind von häufigem Kontaktaufbau und -abbruch geprägt (Abb. 6.9, 6.11, 6.15, jeweils Teilgrafiken (a) bis (c)). Der Bodenstationskontakt im Satellitenszenario 3 ist im Kurz- und Langzeitszenario (Abb. 6.5 und 6.13) ähnlich, da der Algorithmus in beiden Szenarien nur ein einziges Cluster mit 50 Satelliten erzeugt und Bodenstationskontakt nur dann aufgebaut werden kann, wenn der Clusterhead Kontakt zu einer beliebigen Bodenstation besitzt. Lediglich gegen Ende des Simulationszeitraums kann man in Abb. 6.13b beobachten, dass durch diverse Reclusterings zwei Cluster Kontakt zu den amerikanischen Bodenstationen aufbauen konnten.

Der PMW-Algorithmus erzeugt meistens deutlich weniger Cluster als andere Algorithmen. In Abbildung 6.1d sieht man, dass PMW für das Satellitenszenario 1 in der Kurzzeitsimulation meistens drei, manchmal vier und nur einmal fünf Cluster erzeugt, während alle anderen Algorithmen mindestens vier und hauptsächlich fünf Cluster erstellen. Dieses Verhalten wird auch in der Langzeitsimulation (Abb. 6.9d) beibehalten. Vergleicht man Abbildung 6.3d und 6.11d miteinander, so sieht man, dass sich der Algorithmus im Kurzzeitverhalten zwar beinahe gleich den anderen Algorithmen verhält, auf lange Sicht allerdings wiederum sehr wenige Cluster erzeugt, was hauptsächlich daran liegt, dass Satelliten in einem Wartezustand verharren können, in dem sie keinem Cluster zugewiesen sind, bevor sie sich danach selbst zu einem Clusterhead proklamieren. Für das Satellitenszenario 3 verhält sich der Algorithmus sogar fast vollständig gleich, indem sowohl im Kurzzeitszenario (Abb. 6.5d) als auch im Langzeitszenario (Abb. 6.13d) nur ein einziges Cluster erzeugt wird. Nur gegen

Ende des Langzeitszenarios wird ein zweites Cluster erzeugt, da sich die Satelliten nach mehr als eineinhalb Monaten genügend ausgebreitet haben, sodass nicht mehr jeder Satellit Kontakt zu einem ausgewählten Clusterhead aufrechterhalten kann. Das Verhalten des Algorithmus für das Satellitenszenario 4 deutet im Kurzzeitverhalten (Abb. 6.7d) bereits an, dass die Clusterzahl auch im Langzeitszenario (Abb. 6.15d) nicht stabil bleibt, sondern ähnlich wie in 6.1d weit weniger Cluster erzeugt als die anderen Algorithmen, was wiederum daran liegt, dass viele Satelliten nach Kontaktverlust zu ihrem Clusterhead in einem Wartezustand verharren.

Entsprechend instabil wie bei der Anzahl der Cluster verhält sich der PMW-Algorithmus auch bei der mittleren Clustergröße. In Abb. 6.2a und 6.2b kann man sehen, dass der PMW-Algorithmus die stärksten Ausschläge von allen Algorithmen besitzt, die Clustergröße also sehr instabil ist. Im entsprechenden Langzeitszenario (Abb. 6.10a und 6.10b) ist das gleiche Verhalten zu beobachten, diesmal sogar noch stärker als im Kurzzeitszenario. In der Kurzzeitsimulation des wenig mobilen Satellitenszenario 2 erklären sich die Ausschläge in Abb. 6.4a und 6.4b durch kurzzeitige Kontaktverluste und anschließende Reassignments einzelner Satelliten. Im zugehörigen Langzeitszenario hingegen verhalten sich die Cluster extrem instabil, was insbesondere im Vergleich zum Verhalten der von anderen Algorithmen erzeugten Cluster sehr negativ auffällt. Das Clusterverhalten im Kurzzeitszenario zum Satellitenszenario 3 (Abb. 6.6a, 6.6b) ist wie erwartet sehr stabil, da auch nur ein einziges Cluster existiert und sich dieses in seiner Größe nicht ändert. Im Langzeitverhalten (Abb. 6.14a, 6.14b) hingegen ist es dem Algorithmus nach der Aufteilung in 2 Cluster nicht möglich, innerhalb kurzer Zeit zwei ungefähr gleich große Cluster zu erzeugen, da die Standardabweichung sehr stark schwankt. Im Satellitenszenario 4 kann schließlich keine Rede mehr von Clusterstabilität mehr sein, da sowohl die mittlere Clustergröße als auch die Standardabweichung der Clustergrößen extrem schwanken (Abb. 6.8a, 6.8b, 6.16a, 6.16b).

Der PMW-Algorithmus führt insgesamt so gut wie keine Reclusterings durch. Dafür wechseln Satelliten allerdings sehr oft ihre Clusterzugehörigkeit. In den Satellitenszenarien 1 und 4 sowie im Langzeitszenario für das Satellitenszenario 3 wächst die Zahl der Reassignments über alle Grenzen, was auch sehr gut mit der Instabilität der Clustergrößen in diesen Szenarien korreliert. In den übrigen Szenarien hingegen halten sich die Reassignments deutlich in Grenzen, sind allerdings nicht zwingend notwendig, da die restlichen Algorithmen, wenn überhaupt, dann nur sehr wenige Reclusterings durchführen und somit nur einen geringen Teil der Kommunikation zwischen den Satelliten auf Strukturinformationen aufwenden.

Der Datendurchsatz des PMW-Algorithmus liegt beinahe immer unter dem anderer Algorithmen. Einzig in der Langzeitsimulation des Satellitenszenario 3 liegt der PMW-Algorithmus mit seinem Datendurchsatz leicht über dem WCA-Algorithmus. In den meisten Fällen kann der PMW-Algorithmus nur halb so viele Daten wie der NH-Algorithmus übertragen. Die Ergebnisse der Satellitenszenarien 2 und 4 sollen besonders betrachtet werden. Im Kurzzeitszenario des Satellitenszenario 2 verhalten sich alle Algorithmen beinahe exakt gleich, so auch der PMW-Algorithmus. Im entsprechenden Langzeitszenario hingegen fällt der PMW-Algorithmus nach ca. 10 Tagen in seinem mittleren Durchsatz (Abb. 6.12e) sowie dann auch im kumulierten

Durchsatz so ab, dass er sich zusätzlich nicht mehr von diesem Defizit erholt. Allerdings ist der Datendurchsatz im Vergleich zu anderen Algorithmen in gerade diesem Szenario nicht so schlecht wie beispielsweise im Satellitenszenario 1 in der Langzeitsimulation, wo PMW um das dreifache von NH beim mittleren Durchsatz übertroffen wird (Abb. 6.10e) und im kumulierten Datendurchsatz (Abb. 6.10d) nur halb so gut wie BC und sogar weniger als halb so gut wie NH abschneidet.

7.2. WCA-Algorithmus

Der WCA-Algorithmus besitzt ebenfalls nur sehr mäßigen Bodenstationskontakt. In allen vier Kurzzeitsimulationen beträgt die Zahl der Cluster mit Bodenstationskontakt grundsätzlich weniger oder gleich 2. Im Satellitenszenario 3 (Abb. 6.5a bis 6.5c) ist dies auch nicht verwunderlich, da der WCA-Algorithmus ähnlich wie PMW nur ein einziges großes Cluster erzeugt. Im Satellitenszenario 2 (Abb. 6.3a bis 6.3c) verhalten sich alle Algorithmen beinahe gleich. WCA verschiebt nur teilweise seine Kontaktzeiten ein wenig, kann allerdings teilweise sogar längeren Kontakt aufbauen als andere Algorithmen. Im Satellitenszenario 4 (Abb. 6.7a bis 6.7c) verhält sich der Bodenstationskontakt des WCA-Algorithmus eher unauffällig, besitzt aber fast immer auch dann Kontakt, wenn andere Algorithmen ebenfalls Kontakt aufbauen können. Das Verhalten im Satellitenszenario 1 (Abb. 6.1a bis 6.1c) ist ähnlich. Im Langzeitszenario verhält sich WCA in allen Satellitenszenarien ähnlich zu den entsprechenden Kurzzeitszenarien, da meistens nur ein Cluster Kontakt aufbauen kann, aber dafür meistens immer dann, wenn die anderen Algorithmen ebenfalls Kontakt aufbauen konnten. Speziell im Satellitenszenario 4 (Abb. 6.15a und 6.15b) sticht der WCA-Algorithmus mit einigen Kontaktspitzen in seiner Performance heraus.

Im Satellitenszenario 1 erzeugt der WCA-Algorithmus in beiden Simulationszeiträumen (Abb. 6.1d und 6.9d) vier bis fünf Cluster, mit gelegentlichen geringen Abweichungen nach oben und unten. Auch im Satellitenszenario 2 bleibt die Anzahl der Cluster nach oben und unten beschränkt, da im Normalfall drei bis vier Cluster erzeugt werden (Abb. 6.3d und 6.11d). Im Satellitenszenario 3 (Abb. 6.5d und 6.13d) verhält sich der WCA-Algorithmus genauso wie der PMW-Algorithmus und teilt das in beiden Simulationszeiträumen erzeugte einzelne Cluster gegen Ende der Langzeitsimulation in zwei Cluster auf. In der Kurzzeitsimulation des Satellitenszenario 4 (Abb. 6.7d) ist der WCA-Algorithmus der einzige Algorithmus, der durchgehend fünf Cluster erzeugt. Diese Stabilität um einen Wert setzt sich, wenn auch nicht so stark, im Langzeitszenario fort (Abb. 6.15d), wo im Mittel ungefähr 6 Cluster erzeugt werden, mit gelegentlichen leichten Abweichungen nach oben und unten.

Die mittlere Clustergröße der erzeugten Cluster bleibt meistens recht stabil, was entsprechend zur Stabilität der erzeugten Clusterzahl korrespondiert. In Abb. 6.2a ist sogar gegen Ende des Simulationszeitraums eine gewisse Beschränkung der Ausschläge der mittleren Clustergröße zu beobachten. Im Satellitenszenario 4 zeigt sich sogar, dass sowohl die mittlere Clustergröße als auch die Standardabweichung bis auf eine kleine Veränderung durchgängig konstant bleiben (Abb. 6.8a, 6.8b). Die Kehrseite der Medaille offenbart sich allerdings dann, wenn man einen näheren Blick auf die Entwicklung der Standardabweichungen der Clustergrößen während der Langzeitsze-

narios wirft. Gerade in Abbildung 6.10b treten meistens beim WCA-Algorithmus die höchsten Abweichungen in der Clustergröße auf, ebenso in Abbildung 6.12b als auch in Abbildung 6.16b, wobei die Stärke der Abweichungen in der letzten Abbildung nur vom PMW-Algorithmus übertroffen wird. Ebenso benötigt der WCA-Algorithmus noch länger, um nach der Aufteilung in zwei Cluster im Satellitenszenario 3 in der Langzeitsimulation eine gleichmäßige Aufteilung der Satelliten auf die zwei Cluster herzustellen.

Betrachtet man die Grafiken 6.2c bis 6.16c in jeder Simulation, so sieht man, dass der WCA-Algorithmus die meisten Reclusterings anstößt und so gut wie nie Reassignments durchführt. Bemerkenswert ist allerdings, dass in der Kurzzeitsimulation des Satellitenszenarios 4 nur ein einziges Mal gegen Schluss neu geclustert werden musste, während im korrespondierenden Langzeitszenario im Mittel an jedem zweiten Messzeitpunkt ein Reclusteringprozess angestoßen wurde.

Der Datendurchsatz des WCA-Algorithmus ist zwar beinahe immer merklich höher als der des PMW-Algorithmus (außer in Abbildungen 6.14d, 6.14e), aber doch auch merklich niedriger als der des BC- und NH-Algorithmus. In der Langzeitsimulation des Satellitenszenarios 4 (Abb. 6.16d, 6.16e) ist dieser Unterschied zum BC- und NH-Algorithmus allerdings vergleichsweise gering.

7.3. BC-Algorithmus

Der Bodenstationskontakt des BC-Algorithmus ist im Allgemeinen besser als der anderer Algorithmen. Im Satellitenszenario 2 in der Kurzzeitsimulation (Abb. 6.3a bis 6.3c) verhalten sich alle Algorithmen, wie bereits erwähnt, beinahe gleich. Sichtbar wird der verbesserte Bodenstationskontakt vor allem in den Kurzzeitsimulationen der Satellitenszenarios 1 (Abb. 6.1a bis 6.1c), 3 (Abb. 6.5a bis 6.5c) und 4 (Abb. 6.7a bis 6.7c). In den Langzeitsimulationen setzt sich dieses Verhalten in den Satellitenszenarios 1 (Abb. 6.9a bis 6.9c), 2 (Abb. 6.11a bis 6.11c) und 4 (Abb. 6.15a bis 6.15c) analog fort. Einzig in beiden Simulationszeiträumen für das Satellitenszenario 3 schneidet der NH-Algorithmus deutlich besser ab, was aber auch daran liegt, dass der BC-Algorithmus im Langzeitraum zu Beginn nur ein einziges Cluster erzeugt hat (Abb. 6.13d).

Der BC-Algorithmus erzeugt in den Kurzzeitsimulationen fast überall die meisten Cluster (Abb. 6.1d, 6.3d und 6.7d). In Abbildung 6.5d ist zu sehen, dass nur der NH-Algorithmus im Satellitenszenario 3 mehr Cluster erzeugt als der BC-Algorithmus, während WCA und PMW jeweils nur ein einziges Cluster erzeugen. Dieses Verhalten setzt sich bedingt in den Langzeitsimulationen fort. Im Satellitenszenario 1 (Abb. 6.9d) erzeugt der BC-Algorithmus merklich mehr Cluster als die anderen Algorithmen, nur selten übertroffen vom NH-Algorithmus. Im Satellitenszenario 2 (Abb. 6.11d) erzeugen BC und NH im Mittel gleich viele Cluster, allerdings ist es dort auch nicht sehr gut, vier oder mehr Cluster zu erzeugen, da das gesamte Szenario nur aus 7 Satelliten besteht. Im Satellitenszenario 3 (Abb. 6.13d) wird zu Beginn nur ein einziges Cluster erzeugt, was an der Zufallsauswahl von Clusterheads liegt und hier zu einem erheblichen Nachteil gereicht. Letztlich können dann doch noch 2 zusätzliche Cluster erzeugt werden. Im Satellitenszenario 4 schließlich (Abb. 6.15d)

erzeugen WCA, BC und NH ungefähr gleich viele Cluster. Die Zahl der erzeugten Cluster hält sich hier sehr stabil bei sechs Clustern, mit geringen Abweichungen nach oben und unten.

In der Kurzzeitsimulation des Satellitenszenarios 1 kann man sehen, dass der BC-Algorithmus die Clustergröße sehr stabil hält (Abb. 6.2a), auch die Standardabweichung ist durchgängig sehr gering (Abb. 6.2b). Im korrespondierenden Langzeitszenario hingegen variiert die Standardabweichung sehr stark (Abb. 6.10b). Ein ähnliches Verhalten zeigt sich auch für das Satellitenszenario 2: Im Kurzzeitszenario ändern sich mittlere Clustergröße und Standardabweichung gar nicht (Abb. 6.4a, 6.4b), in der Langzeitsimulation variiert die Standardabweichung hingegen sehr stark (Abb. 6.12b). Ebenso zeigt sich dieses Verhalten im Satellitenszenario 4 (Abb. 6.8a, 6.8b sowie Abb. 6.16a, 6.16b). Im Kurzzeitszenario des Satellitenszenario 3 ändert sich wie erwartet gar nichts an der mittleren Clustergröße und Standardabweichung der Clustergrößen (Abb. 6.6a, 6.6b), im Langzeitszenario erfreulicherweise ebenfalls fast nichts (Abb. 6.14a, 6.14b). Sehr bemerkenswert ist die Tatsache, dass alle Satelliten sehr gleichmäßig auf die nach einem Reclusteringprozess erzeugten 3 Cluster aufgeilt wurden, da die Standardabweichung der Clustergrößen hier sehr gering ist (Abb. 6.14b).

Der BC-Algorithmus besitzt im Satellitenszenario 1 sowohl im Kurz- als auch im Langzeitraum ein ausgewogenes Verhältnis zwischen Reassignments und Reclusterings (Abb. 6.2c, 6.10c). In den übrigen 3 Simulationen im Kurzzeitraum wurden keine bis nur sehr wenige Reclusterings und Reassignments durchgeführt. Im Langzeitszenario des Satellitenszenarios 2 (Abb. 6.12c) wurden ebenfalls nur sehr spärlich Reclusterings und Reassignments angestoßen, allerdings deutlich mehr als in den Kurzzeitszenarios. Im Langzeitszenario für das Satellitenszenario 4 schließlich wurde nur neu geclustert und keinerlei Reassignments durchgeführt (Abb. 6.16c), womit sich der BC-Algorithmus hier genauso verhält wie der NH-Algorithmus.

Der Datendurchsatz des BC-Algorithmus ist meistens merklich höher als der des WCA- und PMW-Algorithmus, wie man in den Abbildungen 6.2d, 6.6d, 6.8d, 6.10d, 6.12d und 6.16d sehen kann (analog auch die zugehörigen Abbildungen zum mittleren Durchsatz). Im Satellitenszenario 2 im Kurzzeitraum (Abb. 6.4d, 6.4e) ist kein Unterschied zwischen den verschiedenen Algorithmen bemerkbar, da sich in diesem Szenario alle Algorithmen annähernd gleich verhalten. Die gleiche Begründung kann man auch für das Satellitenszenario 3 in der Langzeitsimulation (Abb. 6.14d, 6.14e) verwenden, da der BC-Algorithmus durch das Erzeugen eines einzigen Clusters zu Beginn des Zeitraums das gleiche Verhalten vorweist wie die WCA- und PMW-Algorithmen. Im Langzeitszenario zum Satellitenszenario 2 (Abb. 6.12d, 6.12e) erreicht der BC-Algorithmus sogar die höchste Durchsatzrate von allen Algorithmen.

7.4. NH-Algorithmus

Der NH-Algorithmus besitzt keinen herausstechend guten Bodenstationskontakt wie der BC-Algorithmus, kann allerdings trotzdem eine merklich bessere Abdeckung als der WCA- und der PMW-Algorithmus herstellen. Die größte Stärke des NH-Algorithmus zeigt sich im Satellitenszenario 3 (Abb. 6.5a bis 6.5c, 6.13a bis 6.13c):

Dort übertrifft der NH-Algorithmus alle anderen Algorithmen mit dem Clusterkontakt zu Bodenstationen.

Die erstellte Anzahl an Clustern ist meistens recht ähnlich der des WCA-Algorithmus (Abb. 6.1d, 6.3d, 6.7d, 6.9d, 6.11d, 6.15d). Im Satellitenszenario 3 erzeugt der NH-Algorithmus im Kurzzeitraum fünf Cluster (Abb. 6.5d), im Langzeitraum vier Cluster (Abb. 6.13d), was aber jeweils deutlich mehr Cluster als bei allen anderen Algorithmen sind.

Die mittlere Clustergröße bleibt im Satellitenszenario 1 sowohl im Kurz- als auch im Langzeitszenario sehr beschränkt (Abb. 6.2a, 6.10a), ebenso die Standardabweichung der Clustergrößen (Abb. 6.2b, 6.2b). Im Kurzzeitraum verändert sich wie auch bei WCA und BC im Satellitenszenario 2 nichts (Abb. 6.4a, 6.4b). Im Langzeitraum hingegen variieren sowohl die mittlere Clustergröße als auch die Standardabweichung, allerdings nur sehr wenig (Abb. 6.12a, 6.12b). Im Satellitenszenario 3 passiert im Kurzzeitraum wie erwartet ebenfalls keine Änderung (Abb. 6.6a, 6.6b), im Langzeitraum erfolgt an einer Stelle ein Reclustering, so dass die Standardabweichung steigt, ansonsten bleiben beide Größen auch hier sehr stabil (Abb. 6.14a, 6.14b). Im Kurzzeitraum für das Satellitenszenario 4 (Abb. 6.8a, 6.8b) ändern sich beide Größen ebenfalls nur an einer Stelle, da auch hier ein Reclustering durchgeführt wurde. Im Langzeitszenario hingegen variieren beide Größen merklich, die mittlere Clustergröße hält sich allerdings stabil zwischen 7 und 8, während die Standardabweichung gerade zu Beginn des Szenarios bis in die Mitte des Testzeitraums sehr stark schwankt. In der zweiten Hälfte der Simulation verringert sich die Standardabweichung aber schließlich und bleibt mit geringen Schwankungen bei einem Wert nahe 2 (Abb. 6.16a, 6.16b).

Insgesamt tendiert der NH-Algorithmus dazu, entweder sehr oft neu zu clustern (Abb. 6.2c, 6.10c) oder nur sehr wenige bis gar keine Reclusterings anzustoßen (Abb. 6.4c, 6.6c, 6.8c, 6.12c, 6.14c, 6.16c). Reassignments werden grundsätzlich fast gar nicht durchgeführt.

Der Datendurchsatz ist beim NH-Algorithmus beinahe überall am höchsten und oft mehr als doppelt so hoch als der des PMW-Algorithmus (Abb. 6.2d, 6.6d, 6.10d, 6.14d). Er liegt meistens knapp über dem des BC-Algorithmus, wird von diesem allerdings in Abb. 6.8d und 6.12d übertroffen. Im Satellitenszenario 2 in der Kurzzeitsimulation zeigt sich kein Unterschied zum Durchsatz der anderen Algorithmen, da sich hier alle Algorithmen sehr ähnlich verhalten.

7.5. Bewertung

Der PMW-Algorithmus schnitt als schlechtester der vier betrachteten Algorithmen ab. Ihm war es nicht möglich, stabile Cluster zu erzeugen, sobald ein Szenario nicht mehr quasistatisch war. Sein Datendurchsatz liegt meistens mit einem signifikanten Abstand unter dem der anderen Algorithmen. Durch die Wartezeit, bis sich ein Satellit einem anderen Cluster anschließt, passierte es ständig, dass ein großer Teil der Satelliten keinem Cluster zugewiesen war und somit auch nicht fähig war, Daten zu senden. Dieser Fehler kann eventuell dadurch behoben werden, dass der Algorithmus in einem kontinuierlichen Szenario simuliert wird, wo er eventuell besser abschneiden würde. Auch wirkte sich die hohe Anzahl von Reassignments schlecht auf die Clus-

terstabilität aus. Vom theoretischen Konzept her ist die Cluster Maintenance-Phase allerdings die beste im Vergleich zu den anderen 3 Algorithmen, da sehr viele Fälle betrachtet werden und versucht wird, eine stabile Lösung dafür zu finden.

Der WCA-Algorithmus, der ebenfalls ein gewichteter Algorithmus wie PMW ist, lieferte stellenweise merklich bessere Ergebnisse als PMW, gerade auch beim Datendurchsatz. Er krankt aber daran, dass verboten wird, 2 Clusterheads als direkte Nachbarn zu erzeugen, was ihn insbesondere bei den beiden Abwurfszenarien sehr schlecht abschneiden lässt. Der WCA-Algorithmus stieß außerdem sehr oft Reclusterings an, mindestens genauso viele wie der BC- und der NH-Algorithmus, teilweise sogar doppelt so viele. Die Wahl der Gewichtsparameter ist zusätzlich nicht sehr ausgereift; so stellt beispielsweise die als Clusterhead verbrachte Zeit $P_v(t)$ keine greifbare Aussage über den Energieverbrauch auf, da es auch darauf ankommt, wieviele Satelliten sich im zugehörigen Cluster befanden. Auch die Graddifferenz ist zwar ein Schritt in die richtige Richtung, indem ein Satellit ausgewählt wird, der eine möglichst passende Anzahl an Nachbaren besitzt, allerdings wäre es sinnvoll, die Zahl der maximalen Clustermember dann auch tatsächlich zu begrenzen oder zumindest die Aufnahme in ein Cluster immer mehr zu erschweren. Weiterhin könnte die Cluster Maintenance-Phase deutlich verbessert werden.

Der BC-Algorithmus liefert überraschend gute Ergebnisse, obwohl die Clusterheads nur rein zufällig erzeugt werden, wodurch auch benachbarte Clusterheads entstehen können. Insbesondere der Bodenstationskontakt konnte in manchen Szenarien den der anderen Algorithmen übertreffen. Auch eignet sich dieser Algorithmus für die kritischen Auswurfszenarien. Ein schwieriger Punkt des Algorithmus liegt allerdings auch gerade in der zufälligen Auswahl der Clusterheads. Sollten beispielsweise zu wenige Clusterheads ausgewählt werden, werden die erzeugten Cluster sehr groß, was die wenigen Clusterheads überlasten würde. An diesem Punkt könnte eine verbesserte Problembehandlung entworfen werden, die dann die Auswahlwahrscheinlichkeit für einen Clusterhead dynamisch erhöht, um so sicherzustellen, dass die nächste Auswahlrunde mehr Clusterheads findet. Außerdem wäre es sicherlich sinnvoll, die Clusterheads nicht ganz zufällig auszuwählen, sondern gewisse Gütekriterien einzuführen, die die Wahrscheinlichkeit erhöhen, einen Satelliten zum Clusterhead zu proklamieren.

Der NH-Algorithmus schließlich wählt Clusterheads ebenfalls zufällig aus, beschränkt sich aber selbst in der Zahl der Clustermember, wodurch in allen Simulationsszenarien durchgängig ein hoher, wenn nicht der höchste Durchsatz erreicht werden konnte. Ebenso war der Bodenstationskontakt der durch den Algorithmus erzeugten Cluster stets gut oder merklich besser als der durch andere Algorithmen erzeugter Cluster. Das Distanzkriterium führt auch dazu, dass selten Reclustering-Prozesse stattfinden, wenn das Szenario nicht allzu mobil ist, da bei nahen Algorithmen die Chance höher ist, dass diese auch in der Nähe des Clusterheads bleiben. Obwohl der NH-Algorithmus als bester Algorithmus abschneidet, gibt es noch einige Verbesserungspunkte zu benennen. Zum einen wäre es durchaus sinnvoll, auch hier die Clusterheads nicht rein zufällig auszuwählen, sondern beispielsweise durch Gewichte eine grobe Vorauswahl zu treffen. Ebenso fehlt eine ausgeklügelte Cluster Maintenance-Phase.

Kapitel 8.

Zusammenfassung

Zum Abschluss soll eine Zusammenfassung des Inhalts der vorliegenden Arbeit gegeben werden. Anschließend wird das Ergebnis der Arbeit formuliert sowie zuletzt ein Ausblick auf weitere Forschungsarbeit gegeben werden.

8.1. Kurzüberblick über die Arbeit

In der vorliegenden Arbeit wurden verschiedene Clusteringalgorithmen für Satellitennetzwerke untersucht. Zuerst wurden in Kapitel 2 theoretische Grundlagen und Basisvokabular vorgestellt. Anschließend wurde in Kapitel 3 ein Überblick über aktuelle Projekte in der Pico- und Nanosatellitenforschung gegeben, wobei insbesondere Projekte betrachtet wurden, die ein Augenmerk auf Satellitennetzwerke legten. Im gleichen Kapitel wurden außerdem verschiedene Clusteralgorithmen betrachtet und schließlich eine Auswahl von Algorithmen getroffen, die implementiert und auf mehrere Szenarien im Orbit angewandt werden sollten.

Für diese Implementierungen wurde in Kapitel 4 ein theoretisches Grundmodell definiert sowie die Simulationsszenarien vorgestellt, auf denen die Algorithmen getestet werden sollten. Zuletzt wurden vier Vergleichskriterien festgelegt, mit denen die implementierten Algorithmen verglichen und bewertet wurden. Kapitel 5 befasste sich mit der tatsächlichen Implementierung eines Simulationsframeworks und stellte kurz den Ablauf einer exemplarischen Simulation vor. Es wurden alle Pakete des Frameworks betrachtet und kurz erläutert sowie schließlich eine detaillierte Betrachtung zu den Implementierungen der Algorithmen gegeben.

Kapitel 6 analysierte die Ergebnisse der implementierten Algorithmen über vier Kurz- und vier Langzeittests und gab kurze Beschreibungen zu den Ergebnisgrafiken. Kapitel 7 bewertete diese Ergebnisse und definierte eine Rangliste der getesteten Algorithmen aufgrund ihrer Gesamtperformance über alle acht Tests.

8.2. Ergebnis der Arbeit

Als Ergebnis der Arbeit ergibt sich nun, dass Clusteringalgorithmen einen sinnvollen Weg darstellen, Strukturen auf Satellitennetzwerken zu errichten und zur Kommunikation zu nutzen, wenn die Netzwerktopologie nicht allzu chaotischen Änderungen unterliegt. Es wurden verschiedene Ansätze zum Clustering getestet, wobei der Ansatz, die nächsten k Satelliten zu einem Cluster zusammenzufügen, wobei k dynamisch

bestimmt werden muss, merklich am besten abschnitt. Insbesondere im Satellitenszenario 3, das eine Satellitenkonstellation von 50 Satelliten direkt nach dem Abwurf betrachtet, hatte der NH-Algorithmus deutliche Vorteile gegenüber seinen Konkurrenten.

Empfehlenswert für eine Implementierung sind also insbesondere der NH-Algorithmus, der BC-Algorithmus sowie eingeschränkt der WCA-Algorithmus. Alle drei Algorithmen sind auf Satellitennetzwerken gut einsetzbar und würden sich durch diverse Korrekturen und direkte Anpassungen an die technischen Gegebenheiten zu sehr mächtigen Werkzeugen zur Strukturierung eines Satellitennetzwerks entwickeln.

Für die Situation direkt nach dem Abwurf aller 50 Satelliten (wie im Satellitenszenario 3 annähernd dargestellt) empfiehlt sich ohne Frage der NH-Algorithmus, da dieser die große Menge aller 50 Satelliten in angemessen große Cluster aufteilen und somit den Datendurchsatz optimieren kann. Für einen späteren Verlauf des Experiments wäre es möglich, eine Kombination aus NH- und WCA-Algorithmus anzuwenden, indem Clusterheads zwar intelligent ausgewählt werden, aber nicht grundsätzlich jeder Nachbar einem Cluster hinzugefügt wird, sondern nur die nähesten Nachbarn, was einen zusätzlichen stabilisierenden Faktor für die Clusterlebensdauer darstellt.

8.3. Ausblick

Als Ausblick auf weitere Forschungs- und Entwicklungsarbeit ist folgendes zu sagen: Die Auswahl der implementierten Algorithmen hat Algorithmen ausgeschlossen, die Kommunikation über mehrere Hops führen und somit hierarchische Cluster bilden könnten. Ebenfalls nicht betrachtet wurden Algorithmen, mit Gateway-Satelliten arbeiten. Eine weitere Möglichkeit wäre es gewesen, explizit Token zu vergeben, um so die Clusterheadwahl direkter beeinflussen zu können.

Weitergehende Ideen wären beispielsweise der Gebrauch eines Bodenstationsnetzwerks wie GENSO, um die Clusterbildung progressiv zu gestalten, in der Form, dass Satelliten von vornherein wissen, mit wem sie bald in Kontakt treten werden. Außerdem anzudenken wäre die Möglichkeit, die Sendeleistung dynamisch an die Anzahl und Entfernung der Nachbarn anzupassen, was dem Energieverbrauch zugute käme. Die Idee der Mitgliederbeschränkung wurde bereits experimentell implementiert. Ebenfalls wäre es interessant, die Entfernungsveränderung von Satelliten zu Bodenstationen über die Zeit zu betrachten, um interpolieren zu können, wann der Kontakt wieder abbrechen wird. In Kapitel 7 wurden für die vier betrachteten Algorithmen außerdem Vorschläge gegeben, wie man die Algorithmen jeweils verbessern könnte.

Anhang A.

Verwendete Orbitdaten

Dies ist eine Auflistung aller real existierenden Satelliten mit ihren TLEs.

A.1. Szenario 1 - Ähnliche Orbits

UWE-I	1 28892U 05043C 11025.15965966 .00000138 00000-0 38156-4 0 8399 2 28892 097.9832 266.0393 0016709 312.1614 047.8170 14.59976977279480
NCUBE2	1 28897U 05043H 11025.16070514 .00000142 00000-0 38758-4 0 3557 2 28897 097.9852 266.7952 0016732 312.8581 047.1209 14.60191117271677
CUBESAT X-IV	1 28895U 05043F 11025.17703811 .00000186 00000-0 48141-4 0 7840 2 28895 097.9867 266.9231 0017197 312.1900 047.7833 14.59998582279370
COMPASS I	1 32787U 08021E 11024.76201212 .00000563 00000-0 77283-4 0 8913 2 32787 097.8666 091.1589 0014772 285.6347 074.3256 14.82119723148312
SEEDS2	1 32791U 08021J 11025.08908587 .00000643 00000-0 87841-4 0 8812 2 32791 097.8669 091.4593 0015022 282.8823 077.0680 14.81865303148317
AAUSAT II	1 32788U 08021F 11025.75508789 .00000714 00000-0 95942-4 0 8812 2 32788 097.8648 092.3154 0014272 282.0140 077.9516 14.82282667148474
DELFIC 3	1 32789U 08021G 11025.80025399 .00000242 00000-0 36726-4 0 8989 2 32789 097.8751 093.1232 0014431 280.0971 079.8621 14.82557911148489
CAN-X 2	1 32790U 08021H 11025.21548402 .00000371 00000-0 53616-4 0 8879 2 32790 097.8651 091.5990 0014498 284.6732 075.2892 14.81782976148346
CAN-X 6	1 32784U 08021B 11025.84078733 .00000041 00000-0 12001-4 0 9010 2 32784 097.8684 091.8172 0014834 276.8512 083.0938 14.81330860148410
OCEANSAT 2	1 35931U 09051A 11025.34637221 .00006032 00000-0 15388-2 0 9249 2 35931 098.2789 124.0434 0002182 150.7139 209.3873 14.50883851 70915
SWISSCUBE	1 35932U 09051B 11025.30211946 .00005241 00000-0 12986-2 0 7956 2 35932 098.3302 126.3937 0006128 216.7544 143.2336 14.52323671 70975
BEESAT	1 35933U 09051C 11025.25313138 .00007828 00000-0 19109-2 0 8595 2 35933 098.3267 126.5531 0005697 236.3456 123.6591 14.52875572 71000
UWE-2	1 35934U 09051D 11025.31145966 .00003314 00000-0 81574-3 0 6719 2 35934 098.3228 126.4547 0005668 240.1865 119.8219 14.52878777 71014

Tab. A.1.: Liste der in Szenario 1 verwendeten Satelliten mit TLEs

A.2. Szenario 2 - Annähernd gleiche Orbits

CP-4	1 31132U 07012Q 11025.04034972 .00000235 00000-0 61783-4 0 998 2 31132 097.9109 064.1389 0084584 244.7998 114.4378 14.55415361200386
AeroCube 2	1 31133U 07012R 11025.04618409 .00000157 00000-0 44866-4 0 606 2 31133 097.9122 064.1439 0084700 244.9044 114.3329 14.55403368200381
CSTB-1	1 31122U 07012F 11025.00963068 .00000556 00000-0 13192-3 0 1267 2 31122 097.9160 064.4553 0084439 245.0299 114.2105 14.55523201200519
MAST	1 31126U 07012K 11025.14343586 .00000077 00000-0 28059-4 0 685 2 31126 097.9097 060.3480 0093523 255.8206 103.2608 14.53666880200309
CP-3	1 31129U 07012N 11025.54440186 .00000077 00000-0 28602-4 0 7812 2 31129 097.9133 057.9106 0101210 263.7751 095.1899 14.52367422200137
CAPE 1	1 31130U 07012P 11024.58140183 .00000060 00000-0 24670-4 0 884 2 31130 097.9149 056.8111 0101621 267.3410 091.6124 14.52282445199790
Libertad-1	1 31128U 07012M 11025.08061661 -.00000047 00000-0 00000+0 0 1322 2 31128 097.9129 057.2335 0101598 265.7233 093.2370 14.52256999200076

Tab. A.2.: Liste der in Szenario 2 verwendeten Satelliten mit TLEs

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Carla-Fabiana Chiasserini, Imrich Chlamtac, Paolo Monti, and Antonio Nucci. **Energy efficient design of wireless ad hoc networks**. In Enrico Gregori, Marco Conti, Andrew Campbell, Guy Omidyar, and Moshe Zukerman, editors, *NETWORKING 2002: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, volume 2345 of *Lecture Notes in Computer Science*, pages 376–386. Springer Berlin / Heidelberg, 2006. URL: `http://dx.doi.org/10.1007/3-540-47906-6_30`.

[2] D.J. Dechene et al. **A survey of clustering algorithms for wireless sensor networks**. 2006.

[3] T. Johansson and L. Carr-Motycková. **On clustering in ad hoc networks**. In *Proc. Vehicular Tech. Conf. Fall, Swedish National Computer Networking Workshop*, volume 2003. Citeseer, 2003.

[4] P. Goyal, V. Parmar, and R. Rishi. **MANET: Vulnerabilities, Challenges, Attacks, Application**.

[5] Javad Akbari Torkestani and Mohammad Reza Meybodi. **Clustering the wireless ad hoc networks: A distributed learning automata approach**. *Journal of Parallel and Distributed Computing*, 70(4):394 – 405, 2010. URL: `http://www.sciencedirect.com/science/article/B6WKJ-4XH5MHY-1/2/639405d1243002ab12f95c6e4b6dc57b`.

[6] Y.P. Chen, A.L. Liestman, and J. Liu. **Clustering algorithms for ad hoc wireless networks**. *Ad Hoc and Sensor Networks*, 28, 2004.

[7] E.M. Belding-Royer. **Multi-level hierarchies for scalable ad hoc routing**. *Wireless Networks*, 9(5):461–478, 2003.

[8] P. Basu, N. Khan, and T.D.C. Little. **A mobility based metric for clustering in mobile ad hoc networks**. *icdcsw*, page 0413, 2001.

[9] **Celestrak, bahnelemente**. `http://celestrak.com/NORAD/documentation/tle-fmt.asp`.

[10] Martin Schnieders. **Diplomarbeit: Attitude determination control for the uwe satellite platform**, 2008.

[11] **Cubesat project**. `http://www.cubesat.org/project.php`.

[12] B. Twiggs and J. Puig-Suari. **Cubesat design specifications document**, 2003.

[13] Y. Aoki, R. Barza, F. Zeiger, B. Herbst, and K. Schilling. **The cubesat project at the university of wuerzburg-the mission and system design**. In *STEC Conference*, 2005.

[14] M. Schmidt, K. Ravandoor, O. Kurz, S. Busch, and K. Schilling. **Attitude determination for the pico-satellite uwe-2**. In *Proceedings IFAC World Congress, Seoul*, 2008.

[15] **Wikipedia-eintrag zu oscar, abgerufen am 25.08.2011**. `http://en.wikipedia.org/wiki/OSCAR`.

[16] S. Ghuffar. **Design and implementation of attitude determination algorithm for the cubesat uwe-3**. 2010.

[17] **Homepage des uwe3-projekts**. `http://www7.informatik.uni-wuerzburg.de/forschung/research_groups/space_exploration/projects/cubesat/uwe_3/`.

[18] Marco Schmidt and Klaus Schilling. *Formation Flying Techniques for Pico Satellites*. NSPO, 2010.

[19] **Homepage des canx-projekts**. `http://www.utias-sfl.net/nanosatellites/CanXProgram.html`.

[20] H. Kayal, F. Baumann, K. Briess, and S. Montenegro. **Beesat: A pico satellite for the on orbit verification of micro wheels**. In *Recent Advances in Space Technologies, 2007. RAST'07. 3rd International Conference on*, pages 497–502. IEEE.

[21] **News über beesat-1 auf der homepage der tu berlin, abgerufen am 22.08.2011**. `http://www.raumfahrttechnik.tu-berlin.de/beesat/v-menue2/beesat-1/news/`.

[22] **Astronews-artikel "berliner kleinsatellit seit einem jahr im all", abgerufen am 13.08.2011**. `http://www.astronews.com/news/artikel/2010/09/1009-035.shtml`.

[23] **Homepage des beesat-3-projekts**. `http://www.raumfahrttechnik.tu-berlin.de/beesat/v-menue2/beesat-3/`.

[24] A. et al. Schwarzenberg-Czerny. **The brite nano-satellite constellation mission**. 2010. `http://www.utias-sfl.net/docs/LivePapersAsOfJan2011/BRITE-COSPAR2010-PaperSR-WW-REZ-SM-AS-TM.pdf`.

[25] **Präsentation "launches and on-orbit performance"**, 2010. `http://www.cubesat.org/images/cubesat/presentations/SummerWorkshop2010/kekez-utias_sfl.pdf`.

[26] **Homepage des spheres-projekts**. `http://ssl.mit.edu/spheres/`.

[27] N.G. Orr, J.K. Eyer, B.P. Larouche, and R.E. Zee. **Precision formation flight: the canx-4 and canx-5 dual nanosatellite mission**. *Space Flight Laboratory, University of Toronto Institute For Aerospace Studies*, 4925.

[28] **Homepage des humsat-projekts**. `http://www.humsat.org/`.

[29] Camacho Vivas Castro Balogh Reglero Aguado, Puig-Suari. **Präsentation vom iac 2010**, 2010.

[30] **Homepage des qb50-projekts, unterseite "launch"**. `https://www.qb50.eu/launch.php`.

[31] E. Gill et al. **Formation flying within a constellation of nano-satellites: The qb50 mission**. 2010. 6th International Workshop on Satellite Constellation and Formation Flying, Taipei, Taiwan.

[32] **Qb50 project**. `http://www.vki.ac.be/QB50`.

[33] J. Muylaert. **Qb50, an international network of 50 double cubesats for multi-point, in-situ measurements in the lower thermosphere and for re-entry research**, 2009. `https://www.qb50.eu/download/workshop/papers_17nov/muylaert.pdf`.

[34] Maximilian Drentschew, Michael Marszalek, Florian Zeiger, Markus Sauer, Marco Schmidt, and Klaus Schilling. **Pico- and Nano-Satellite Based Mobile Ad-hoc Networks - A Requirements Study**. In *Accepted for 1st International SPACE World Conference*, 3.-5. October, Frankfurt a.M., Germany, 2010.

[35] **Homepage von quakesat**. `http://www.quakefinder.com/services/quakesat-ssite/`.

[36] Rainer Sandau. **Distributed satellite systems for earth observation and surveillance**. In *Small Satellite Formations For Distributed Surveillance: System Design and Optimal Control Considerations*, 2009.

[37] K. Schilling. **Earth observation by distributed networks of small satellites**. In *Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2009 International Conference on*, pages 1–3. IEEE, 2009.

[38] K. Schilling. **Mission analyses for low-earth-observation missions with spacecraft formations**. In *Small Satellite Formations For Distributed Surveillance: System Design And Optimal Control Considerations*, 2009.

[39] **Global positioning system standard positioning service performance standard**. 2008. `http://www.navcen.uscg.gov/pdf/gps/geninfo/2008SPSPerformanceStandardFINAL.pdf`.

[40] Gasparini G. Lindström, G. **The galileo satellite system and its security implications**. *Occasional Papers*, 2003. http://www.navcen.uscg.gov/pdf/gps/geninfo/2008SPSPerformanceStandardFINAL.pdf.

[41] C.P. Escoubet, R. Schmidt, and M.L. Goldstein. **Cluster ? science and mission overview**. *Space Science Reviews*, 79:11–32, 1997. 10.1023/A:1004923124586. URL: http://dx.doi.org/10.1023/A:1004923124586.

[42] ESA. *LISA assessment study report*. 2011. http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=48364#.

[43] T.A. Prince, P. Binetruy, J. Centrella, LS Finn, C. Hogan, G. Nelemans, ES Phinney, and B. Schutz. **Lisa: probing the universe with gravitational waves**. In *Bulletin of the American Astronomical Society*, volume 38, page 990, 2006.

[44] D. Baker, A. Ephremides, and J. Flynn. **The design and simulation of a mobile radio network with distributed control**. *Selected Areas in Communications, IEEE Journal on*, 2(1):226–237, 1984.

[45] M. Gerla and J. Tzu-Chieh Tsai. **Multicluster, mobile, multimedia radio network**. *Wireless networks*, 1(3):255–265, 1995.

[46] C.C. Chiang, H.K. Wu, W. Liu, and M. Gerla. **Routing in clustered multihop, mobile wireless networks with fading channel**. In *proceedings of IEEE SICON*, volume 97, pages 197–211. Citeseer, 1997.

[47] S. Basagni. **Distributed clustering for ad hoc networks**. In A. Y. Zomaya, D. F. Hsu, O. Ibarra, S. Origuchi, D. Nassimi, and M. Palis, editors, *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, pages 310–315, Perth/Fremantle, Australia, June 23–25 1999. IEEE Computer Society.

[48] Mainak Chatterjee, Sajal K. Das, and Damla Turgut. **Wca: A weighted clustering algorithm for mobile ad hoc networks**. *Cluster Computing*, 5:193–204, 2002. 10.1023/A:1013941929408. URL: http://dx.doi.org/10.1023/A:1013941929408.

[49] D. Turgut, S.K. Das, R. Elmasri, and B. Turgut. **Optimizing clustering algorithm in mobile ad hoc networks using genetic algorithmic approach**. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 1, pages 62–66. IEEE, 2002.

[50] A.D. Amis, R. Prakash, T.H.P. Vuong, and D.T. Huynh. **Max-min d-cluster formation in wireless ad hoc networks**. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 32–41. IEEE, 2000.

[51] Y. Jiang, Y. Liu, Y. Wen, and G. Wang. **A clustering algorithm applied to the satellite networks management**. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 396–399. IEEE, 2003.

[52] Z. Xing, L. Gruenwald, and KK Phang. **A Robust Clustering Algorithm for Mobile Ad-hoc Networks**. *Handbook of Research on Next Generation Networks and Ubiquitous Computing*, 2010.

[53] P. Sheu and C. Wang. **A stable clustering algorithm based on battery power for mobile ad hoc networks**. *Tamkang Journal of Science and Engineering*, 9(3):233, 2006.

[54] K. Schilling. **Networked distributed pico-satellite systems for earth observation and telecommunication applications**, 2009.

[55] S. Cakaj, W. Keim, and K. Malaric. **Communications duration with low earth orbiting satellites**. In *Proc. IEEE, IASTED, 4th International Conference on Antennas, Radar and Wave Propagation*, pages 85–88.

[56] **Handbuch des agi stk, abgerufen am 19.09.2011**. `http://www.agi.com/resources/help/online/stk/index.html`.

[57] **Homepage des javaoctave-projekts**. `http://kenai.com/projects/javaoctave/pages/Home`.