Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3700

Wohin mit der Arbeit? Fehlertoleranz durch gezielte Workflow-Replikation

Chris Geiger

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Kurt Rothermel
Betreuer/in:	Dipl.-Inf. David Richard Schäfer, Dr. Muhammad Adnan Tariq
Beginn am:	17. November 2014
Beendet am:	19. Mai 2015
CR-Nummer:	C.2.1, C.2.4, C.4

Kurzfassung

In den letzten zehn Jahren hat die Anzahl der mobilen Endgeräte ein enormes Wachstum erlebt. Jeder zweite Mensch in Deutschland verwendet ein mobiles Endgerät, sei es um E-Mails zu schreiben, im Internet zu surfen, oder gar um die eigens entwickelten Apps auszuführen. Um irgendeine Art von Anwendung erfolgreich ausführen zu können, muss diese Prozesse ausführen. Meist ist das nicht nur ein Prozess sondern tausende, oder gar zehn tausende. Die Anwendung muss also einen Workflow, eine Verkettung von Prozessen, ausführen. Die einzelnen Prozesse des Workflows rufen meistens einen Service auf, um ihre Funktion zu erfüllen und die gewünschten Daten zu erhalten. Dieser Service kann über das Internet, direkt über Bluetooth, oder über ein anderes Netzwerk erreichbar sein. Da heutzutage die meisten Anwendungen auch von unterwegs ausgeführt werden, ergeben sich neue Probleme. Das mobile Endgerät könnte die Verbindung zum Netz verlieren, wodurch der gerade auszuführende Prozess keine Verbindung mehr zum erforderlichen Service herstellen könnte. Auch ein leerer Akku würde die weitere Ausführung des Workflows unmöglich machen. Um die weitere Ausführung eines Workflows dennoch zu gewährleisten, werden Replikate des Workflows auf andere mobile Endgeräte, sowie feste Instanzen, zum Beispiel Server, verteilt. Da eine optimale Verteilung der Replikate unter realen Bedingungen nicht in einer akzeptablen Zeit zu berechnen ist, werden andere Lösungsansätze gesucht. Diese Arbeit soll Heuristiken einführen, die sich dieses Problems annehmen und somit zur hohen Verfügbarkeit von Workflows beitragen. Diese Heuristiken sollen eine möglichst effiziente Verteilung der Replikate in einer kurzen Zeit erzielen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

1 Einleitung

In der heutigen Zeit sind Mobiltelefone nicht mehr weg zu denken. Vor zehn bis fünfzehn Jahren sah das noch ganz anders aus.

Unsere Welt ist ständig im Wandel. Sei es in der Natur oder in der vom Menschen erschaffenen Welt. Alles entwickelt sich kontinuierlich weiter. Aus statisch wird dynamisch, aus stationär wird mobil.

In den frühen 90er Jahren gab es noch kaum Computer in privaten Haushalten, heute gibt es kaum noch einen Haushalt ohne einen oder gar mehrerer Computer. Zu dieser Zeit gab es noch keine Smartphones. Das erste Smartphone wurde 1992 von IBM und der damaligen Telefongesellschaft BellSouth entwickelt und kam 1994 auf den Markt. Der Verkauf des ersten Smartphones names Simon war nicht von Erfolg geprägt, lediglich 50.000 Stück wurden verkauft. Da es noch kein mobiles Internet gab, war das Smartphone auch nur sehr begrenzt einsetzbar. Es wurde lediglich in den USA vertrieben und kostete die stolze Summe von 899 US Dollar. Des weiteren hielt der Akku des ersten Smartphones ungefähr für eine Stunde. [Kau15] [Ste15]

Von 1995 bis 2015 machte die Entwicklung einen enormen Sprung, der eine ganz neue Gestaltung und Nutzung von Smartphones zuließ.

Das Smartphone ist nun kein riesiger Klotz mehr, der 500 Gramm wiegt, sondern ein handlicher kleiner Minicomputer mit außergewöhnlicher Leistung. Ein derzeitig mittelpreisiges Smartphone hat ein Prozessor mit vier Kernen und 32 GB Speicherplatz. Das ist im Vergleich zum ersten Smartphone mit 16 MHz und 1 MB Speicherplatz nicht mehr zu vergleichen. [Wik15]

Nicht nur die Technik hat sich zu damals drastisch verändert, auch die Anzahl der Mobiltelefone ist in den vergangenen Jahren exponentiell gewachsen.

Laut einer Pressemitteilung der BITKOM, dem Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V., nutzen derzeit über 44 Millionen Deutsche ein Smartphone. Bei einer Einwohnerzahl von etwas über 80 Millionen Menschen innerhalb Deutschlands sind das über 0,5 Smartphones pro Einwohner. [BIT15] [Sta15a] [Sta15b]

Jeder zweite Mensch nutzt folglich ein Smartphone. Dabei sind die Menschen die gar zwei oder mehr Smartphones benutzen nicht mit eingerechnet. Viele Menschen benutzen mehr als ein Smartphone im Alltag. Arbeitnehmer und Arbeitgeber haben für ihren Beruf und ihr Privatleben oft jeweils ein eigenes Smartphone. Einige Manager haben sogar ein Handy für jede Zeitzone in der sie tätig sind.

Mit diesen Veränderungen der Technik und der Anzahl von Mobiltelefonen, entstehen neue Möglichkeiten, aber auch neue Anforderungen.

Eine der neuen Möglichkeiten ist zum Beispiel das mobile Bezahlen (eng. mobile payment). Diese neue Anwendungsmöglichkeit ist zu diesem Zeitpunkt noch nicht so sehr verbreitet, wird in Zukunft jedoch eine deutlich größere Rolle spielen. Hier nur einige Beispiele des mobilen Bezahlens:

- Bitcoins
- Google Wallet
- Apple Pay
- PayPal Here
- Visa: PayWave
- MasterCard: PayPass

Mit dieser neuen Möglichkeit kommen natürlich auch neue Anforderungen. Sicherheit ist hier zum Beispiel einer der wichtigsten neuen Asperkte.

Dies sind nur einige wenige Lösungen des mobilen Bezahlens. Es gibt bereits schon jetzt viel viel mehr solcher Methoden, obwohl das mobile Bezahlen noch nicht so richtig beim Kunden angekommen ist.

Die Möglichkeit des mobilen Bezahlens sollte nur eins der Beispiel für die Bereiche sein, die sich durch die neue Technik ergeben haben.

1.1 Motivation

Die grundsätzliche Motivation dieser Arbeit lautet “pervasive computing“, dessen wörtliche Übersetzung “durchdringendes Rechnen“ beziehungsweise “um sich greifendes Rechnen“ bedeutet. Sinngemäß ist hier die Rechnerallgegenwart gemeint. Auch bekannt unter “ubiquitous computing“. [Ven06] [Sat01] [VLC10]

Pervasive computing ist die Vorstellung, dass die Technologie über den private Computer zu Hause hinaus geht. Es behandelt die Idee, Rechenleistung überall und jederzeit, in einer einheitlichen Art und Weise, verfügbar zu machen, sodass es den Ansprüchen der Gesellschaft gerecht wird und für jedermann zugänglich ist. Ein naheliegender Vergleich ist das Stromnetz. [Ven06] Es gibt so gut wie überall eine Möglichkeit an Strom zu kommen, sei es in Cafe’s, Restaurants, Universitäten, sogar in Zügen und Bussen gibt es Steckdosen. Ähnlich wie das Stromnetz stellt man sich auch die Rechnerallgegenwart vor. Pervasive computing ist das Resultat der exponentiell schnellen Weiterentwicklung unserer Technologie. Immer kleiner werdende Prozessoren mit steigender Leistung ermöglichen den Einbau eben solcher Prozessoren in alltägliche Dinge. [Sat01] [VLC10] Einige derzeitige Beispiele für pervasive computing sind:

- Kreditkarten
- Autos
- Fernseher (Smart TVs)

- Uhren (Smart Watches)
- Waschmaschinen
- Häuser (Smart Homes)

In all diesen Dingen sind Prozessoren verbaut, die Rechenleistung bereitstellen.

Ein sehr wichtiger Begriff im Zusammenhang mit pervasive computing ist “location based services“. Diese sogenannten location based services, was so viel bedeutet wie positionsbezogene Services, sind eben diese Services beziehungsweise Dinge, die überall um uns herum existieren und mit einem Chip versehen sind. Man spricht oft auch von einem “pervasive environment“, einer “um sich greifenden Umgebung“. In der folgenden Arbeit wird der Begriff “pervasive Umgebung“ für diese Art der Umgebung benutzt. Die Vorstellung einer solchen Umgebung geht weit über das hinaus, was bis jetzt vorhanden und möglich ist. Die Idee ist es, dass nahezu jeder Gegenstand der uns umgibt mit einem Chip versehen wird, um ihm eigener Rechenleistung zu geben. Das geht über die Kleidung, die Kaffeetasse, den Tacker, eben alles was einen Nutzen haben könnte. [Ven06] [Sat01] [VLC10]

Wie in der Einleitung schon erwähnt, bringen neue Möglichkeiten auch neue Anforderungen mit sich. So wie die Rechenleistung immer allgegenwärtiger wird, so muss sich auch die Architektur der Anwendungen, die auf diese Rechenleistung zugreifen wollen, den neuen Anforderungen anpassen.

Die Anzahl der Interaktionen einer solchen Anwendung ist deutlich größer als bei herkömmlichen Anwendungen. Durch nur einen einzigen Meter in eine beliebige Richtung, könnten zahlreiche neue positionsbezogene Services hinzukommen, mit denen die Anwendung kommunizieren kann.

Auch die Firmen passen ihre Anwendungen immer mehr auf die Wünsche ihrer Kunden an. Früher war es beispielsweise nicht möglich sich das gewünschte Automodell, extra in einer gewünschten Farbe zu bestellen. Heutzutage ist es nicht ungewöhnlich sich sein Wunschauto von zu Hause über das Internet so zu konfigurieren, dass es genau den eigenen Anforderungen entspricht. Das gilt selbstverständlich auch für die Anwendungen, die auf positionsbezogene Services zugreifen.

Diese neuen Inventionen, Innovationen und Anpassungen sind aber keinesfalls unfehlbar. Es entstehen hierdurch auch neue Probleme und Fehlerquellen die es zu beseitigen gilt. Sei es beim mobilen Beahlen, die Sicherheit zu gewährleisten, oder ein eher grundlegendes Problem, zum Beispiel das der Fehlertoleranz an sich.

In einem hoch mobilen Netzwerk ist die Wahrscheinlichkeit von ausfallenden Endgeräten ständig present. Die Gründe für den Ausfall der Endgeräte können viele Ursachen haben. Das Netz des Anbieters könnte überlastet sein oder komplett ausfallen. Durch Gewitter oder Stürme könnten Sendemasten beschädigt werden, was den Ausfall des Netzes aller Anbieter bedeuten würde. Durch unzureichende Gebietsabdeckung der Masten könnte es blinde Flecke geben, in denen man kein Empfang hat. In einem stark isolierten Gebäude oder Tunnel reicht oft die Netzstärke nicht aus, um diese starke Abschottung zu durchdringen. Selbst so einfache Gründe wie ein leerer Akku oder gar der Verlust des Endgerätes, führen unweigerlich zu einem Ausfall und somit auch zum Ausfall des auf dem Endgerät ausgeführten Workflows.

Ein Workflow ist eine Verkettung von Prozessen beziehungsweise Aktivitäten, die nacheinander ausgeführt werden müssen, um ein bestimmtes Ziel zu erreichen.

Um Fehlertoleranz bei einem Netzwerk mit vielen mobilen Knoten, die jederzeit ausfallen können, zu erreichen, bietet sich die Replikation der Workflows an. [SSB+14] [DRS15] Sie dient dazu, dass die Replikate des Workflows die einzelnen Prozesse weiterhin ausführen können, während ein Endgerät zeitweise ausfällt. Replikation soll folglich eine hohe Verfügbarkeit der Workflows gewährleisten. Eine spätere Wiederherstellung der Verbindung zum ursprünglichen Endgerät wird hier vorausgesetzt. Eine einfache Replikation an beliebige andere Knoten in einem Netzwerk wäre allerdings nicht wirklich sinnvoll. Diese Arbeit befasst sich mit der genauen Platzierung der Replikate, um eine möglichst hohe Fehlertoleranz zu erreichen. Die Ausarbeitung und Evaluation von Heuristiken für diese Platzierung beziehungsweise Verteilung, von Replikaten, ist der Hauptbestandteil dieser Arbeit.

1.2 Gliederung

Diese Arbeit ist in folgende Kapitel eingeteilt:

Kapitel 2 – Hintergrund beschreibt die Grundlagen der Replikation. In Kapitel 3 – Das Systemmodell wird das Systemmodel vorgestellt. Es definiert die Entitäten an sich und die Kommunikation zwischen den Entitäten, beziehungsweise den Zusammenhang. Die Problemstellung, die sich aus der Motivation ableitet, wird in Kapitel 4 – Problemstellung erklärt. Kapitel 5 – Technischer Teil - Statisch beschreibt den technischen Teil der Ausarbeitung. Hier wird das Problem auf ein bekanntes Problem reduziert, und entsprechende Heuristiken zur Lösung dieses Problems vorgestellt. Der dynamische Teil der Arbeit wird in Kapitel 6 – Technischer Teil - Dynamisch beschrieben. Es wird unter anderem auf die Anpassungen während der Laufzeit einer Simulation eingegangen. Anschließend werden in Kapitel 7 – Evaluation die Ergebnisse der Messungen verglichen und ausgewertet. Kapitel 8 – Verwandte Arbeiten gibt einen kurzen Überblick über Arbeiten, die ähnlichen Themen behandeln. Abschließend werden in Kapitel 9 – Zusammenfassung & Ausblick, die beobachteten Ergebnisse zusammengefasst und ein Ausblick für zukünftige Arbeiten in diesem Bereich gegeben.

2 Hintergrund

In diesem Kapitel werden die Grundlagen beziehungsweise der Hintergrund des Themas dieser Arbeit beschrieben.

Diese Diplomarbeit befasst sich mit einem Thema, welches Teil der Forschung des IPVS (Institut für Parallele und Verteilte Systeme) der Universität Stuttgart-Vaihingen ist.

Zu dieser Forschung gibt es schon mehrere Ausarbeitungen auf die in Kapitel **??** – **??** noch näher eingegangen wird.

Wie in Kapitel 1.1 – Motivation beschrieben, erreichen wir Fehlertoleranz mit Hilfe der Replikation von Workflows. Die Replikation bringt allerdings auch Probleme mit sich. Die Ausführung der Replikate ist folglich die Ausführung mehrerer Instanzen des Workflows. Wir müssen also berücksichtigen, dass die Ausführung mehrerer Instanzen des selben Workflows zu unterschiedlichen Ergebnissen führen könnte beziehungsweise eine Ausführung eine andere Ausführung beeinflussen könnte. Es muss folglich sichergestellt werden, dass bei der Ausführung einer einzelnen Instanz des Workflows, dasselbe Ergebnis wie bei der Ausführung mehrerer Instanzen des Workflow liefert. Man nennt dies auch Konsistenz. Wenn die Ausführung einer einzelnen Instanz eines Workflows das gleichen Ergebnis liefert wie Ausführung mehrerer Instanzen des selben Workflows, so sind diese zwei Ausführungen konsistent. [DRS15]

Die Vorraussetzungen um eine solche Konsistenz prüfen zu können, wurden in [DRS15] erhoben und definiert.

Kurz zusammengefasst wird zwischen den einzelnen Aktivitäten eines Workflows in folgenden Punkten differenziert.

- Interaktion
- Idempotenz
- Determinanz
- Kompensation

Die Interaktion einer Aktivität bezeichnet ihren Einfluss nach Außen. Eine nicht interaktive Aktivität hat keine Auswirkung auf andere Aktivitäten. Wird eine solche Aktivität ausgeführt, so werden keine anderen Aktivitäten beeinflusst. Eine interaktive Aktivität hingegen beeinflusst den Außenzustand, der für anderen Aktivitäten sichtbar ist und gegebenenfalls auch von diesen genutzt wird.

Bei idempotenten Aktivitäten macht es keinen Unterschied, ob sie nur einmal oder mehrmals ausgeführt werden. Das Ergebnis einer einzelnen Ausführung ist hier dasselbe, wie das Ergebnis mehrerer

Ausführungen. Eine nicht idempotente Aktivität hingegen, kann nicht beliebig oft ausgeführt werden, da das Ergebnis nicht dasselbe wäre.

Die Determinanz einer Aktivität gibt an, ob die Ausführung einer Aktivität unter exakt denselben Gegebenheiten immer dasselbe Ergebnis liefert. Eine nicht deterministische Aktivität würde folglich, bei der Ausführung mit immer denselben Gegebenheiten, nicht immer dasselbe Ergebnis liefern.

Als vierter Differenzierungspunkt wurde die Kompensation aufgeführt. Eine kompensierbare Aktivität bietet die Möglichkeit die Ausführung dieser Aktivität rückgängig zu machen. Wenn man es ganz genau nimmt, müsste es hier eine weitere Unterscheidung geben. Eine Aktivität könnte teilweise beziehungsweise zu einem gewissen Grade kompensierbar sein. Die Buchung eines Flugtickets beispielsweise ist zum einem gewissen Teil kompensierbar. Falls man von seiner Reise zurück tritt, zahlt man lediglich einen geringen Teil der eigentlich Kosten. Die Kompensation eines Flugtickets, wäre auch ein gutes Beispiel für die Determinanz. Der Ausgang dieser Aktivität ist ne nach Gegebenheiten (Parametern) unterschiedlich. Falls der Fluggast eine Reiserücktrittsversicherung abgeschlossen hat, so wird er einen größeren Teil erstattet bekommen, als ohne eine solche Versicherung.

In [DRS15] wurde als Beispiel-Workflow die Bestellung eines Taxis bis hin zur Abholung des Fahrgastes genommen. Ähnlich wie in Abbildung 2.1.

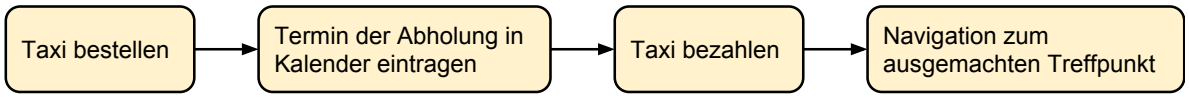

Abbildung 2.1: Beispiel Workflow: Taxibestellung

Dieser Beispiel-Workflow besteht aus vier unterschiedlichen Aktivitäten. Die erste Aktivität, Taxi bestellen, ist eine nicht interaktive, idempotente, nicht deterministische Aktivität. Da es sich hier um eine nicht interaktive Aktivität handelt, kommt die Anwendung der Kompensation hier nicht vor. Kompensation ist nur für interaktive Aktivitäten interessant, da eine nicht interaktive Aktivität keinen Einfluss nach Außen hat und somit kein Grund zur Kompensation bestehen kann.

Den Termin der Abholung in den Kalender eintragen ist hingegen interaktiv, da der Termin in den Kalender eingetragen wird und somit nach Außen sichtbar ist. Zudem ist die Aktivität idempotent, deterministisch und kompensierbar. Hierbei wurde davon ausgegangen, dass die mehrfache Eintragung desselben Termins, durch den Kalender selbst verhindert wird, um Dopplungen zu vermeiden. Daher idempotent. Und deterministisch, da die Eintragung eines Termins mit exakt denselben Parametern, immer zum selben Ergebnis führt.

Die nächste Aktivität ist ebenfalls eine interaktive Aktivität. Sie ist zusätzlich nicht idempotent, deterministisch und je nach Gegebenheit kompensierbar. Im Beispiel von [DRS15] ist sie kompensierbar. Interaktiv, da sie sowohl den eigenen Geldbeutel schwächt, als auch den des Taxifahrers stärkt. Nicht idempotent, da die mehrmalige Ausführung zu einem größeren Geldverlust führt, als die einmalige Ausführung. Kompensierbar, da man nur eine Fahrt in Anspruch nimmt und, falls die Fahrt mehrmals bezahlt wurde, das Geld wieder zurückerstattet bekommt. Hier könnte es auch sein, dass der Taxiservice keine Rückerstattung von bereits gezahlten Fahrten anbieten würde. Somit wäre die Aktivität nicht kompensierbar.

Weiter werden in [DRS15] anhand dieser Differenzierungen beziehungsweise Klassifikationen, Kategorien erstellt. Die Kategorien geben an ob die einzelnen Aktivitäten des Workflows mehrmals ausgeführt werden dürfen oder nicht.

Dies ist von großer Bedeutung für das endgültige Resultat des Werkflows. Aktivitäten die nicht mehrmals ausgeführt werden dürfen, würden bei Mehrfachausführung das Ergebnis des Workflows verändern. Nehmen wir an die Aktivität "Taxi bezahlen" wäre nicht kompensierbar, so dürfte diese Aktivität nicht mehrmals ausgeführt werden, da sonst die Person, die das Taxi geordert hat, mehr Geld bezahlen müsste als eigentlich nötig.

In den folgenden zwei Sektionen werden die zwei Arten der Konsistenz, die in [DRS15] beschrieben werden, kurz wiedergegeben.

2.1 Starke Konsistenz - Fehlertoleranz

Die erste Art der Konsistenz ist die sogenannte "starke Konsitenz".

> **Definition 2.1.1**
> *A replicated execution is consistent if and only if it is equivalent to a correct non-replicated cell execution. [DRS15, p. 27, Definition 1]*

Im Bezug auf diese Ausarbeitung bedeutet starke Konsistenz, dass die Ausführung mehrerer Instanzen einer Aktivität identisch ist mit der Ausführung nur einer korrekten Instanz dieser Aktivität. Der Workflow und dessen Ergebnisse werden folglich nach jeder Aktivität beziehungsweise nach jedem Schritt auf Konsistenz geprüft. Dies ist nur eine vereinfachte Erklärung und spiegelt nicht den Umfang wieder, der in [DRS15] beschrieben wird.

Das bedeutet, dass das Ergebnis einer Aktivität mit dem Ergebnis der Aktivität der anderen Replikate verglichen wird. Falls nun die absolute Mehrheit der Ergebnisse aller Replikate übereinstimmen, so wird an die anderen beziehungsweise fehlenden Ausführungs-Entitäten ein Update mit dem richtigen Ergebnis gesendet. Was genau eine Ausführungs-Entität ist wird in Kapitel 3 – Das Systemmodell erklärt.

2.2 Schwache Konsistenz - Hohe Verfügbarkeit

Die zweite Art, der in [DRS15] beschriebenen Konsistenzen, nennt sich "schwache Konsistenz".

In einer pervasiven Umgebung gibt es etliche mobile Entitäten beziehungsweise mobile Knoten. Mobile Endgeräte und Wearables umgeben uns schon jetzt überall und jederzeit. In der Zukunft wird die Anzahl dieser mobilen Entitäten noch drastisch steigen.

In einer solchen Umgebung beziehungsweise in einem solchen System it die Wahrscheinlichkeit sehr hoch, dass eine Partitionierung [Bre12] [WL02] [HCP03] der Entitäten vorliegt. Es gibt also Gruppen von Knoten, die nicht miteinander kommunizieren können. Veranschaulicht wird das in Abbildung 2.2.

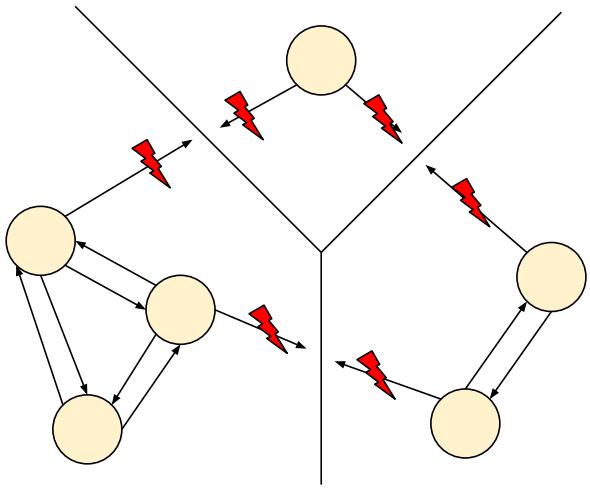

Abbildung 2.2: Partitionierung von Knoten

Durch eine solche Partitionierung beziehungsweise Abschottung der einzelnen Gruppen, kann es bei der starken Konsistenz zu einer Verzögerung beziehungsweise zum Stillstand der Ausführung des Workflows kommen. Wir nehmen an, dass in jeder Partition die Menge der enthaltenen Knoten kleiner ist als die Anzahl der absoluten Mehrheit aller im System vorhandenen Knoten. Im Verlauf dieser Arbeit sprechen wir von $f + 1$ für die absolute Mehrheit einer Menge. Weiter nehmen wir an, dass keine Veränderung der Partitionen stattfinden würde. In diesem Fall könnten die Ergebnisse der Ausführungen von Aktivitäten, bei starker Konsistenz, nicht an $f + 1$ Knoten kommuniziert werden, was zur Folge haben würde, dass der Workflow schon bei der ersten Aktivität zum Stillstand kommen würde.

Dies ist am Beispiel in Abblidung 2.2 gut zu sehen. Die größte Partition umfasst lediglich drei Knoten. Es können als maximal drei Knoten miteinander kommunizieren. Um Konsistenz zu erreichen, müssten $f + 1$, in diesem Falle vier, Knoten miteinander kommunizieren können.

Diese Annahme ist nur rein theoretisch und statischer Natur. Ein System mit vielen mobilen Entitäten ist der realen Welt hoch dynamisch und verändert sich ständig. Es kommt also mit an Sicherheit grenzender Wahrscheinlichkeit nie zu einem kompletten Stillstand des Workflows, lediglich zu einer temporären Verzögerung. Während dieser Verzögerungen ist der Workflow sozusagen nicht verfügbar. Dies wird auch ausführlich im CAP-Theorme [Bre12] beschrieben. Es ist folglich nicht möglich starke Konsistenz und hohe Verfügbarkeit zu vereinen. Mit Hilfe der schwachen Konsistenz, kann die Verfügbarkeit eines Workflows verbessert werden. [SSB+14]

Die zweite Art der Konsistenz versucht also eine hohe Verfügbarkeit des Workflows zu gewährleisten. Sie lässt eine temporäre Inkonsistenz während der Ausführung zu. Lediglich das endgültige Ergebnis des Workflows beziehungsweise der endgültige Status muss konsistent sein. Es können also Aktivitäten mehrfach ausgeführt werden, unabhängig davon, ob diese Aktivitäten andere Resultate erzielen. Lediglich nach der Ausführung der letzten Aktivität eines Workflows wird auf Konsistenz geprüft. Hierbei ist die Konsistenzbedingung gleich wie bei der starken Konsistenz. Es müssen $f + 1$ Knoten denselben Endstatus erzielen, um den Workflow erfolgreich abschließen zu können. Hier müssen

wir noch die Annahme treffen, dass alle während der Ausführung getätigten Aktivitäten, die nicht replizierbar, sprich nicht idempotent oder nicht deterministisch sind, erfolgreich kompensiert wurden. [DRS15]

3 Das Systemmodell

Die in dieser Diplomarbeit erarbeiteten Heuristiken benötigen zu ihrer Validierung und Evaluation ein vereinfachtes Modell der realen Gegebenheiten. Spezifischer gesagt, benötigen die Heuristiken ein abstrahiertes Systemmodell der realen Welt. Das Systemmodell stellt die komplexen Beziehungen der realen Welt vereinfacht dar.

Dieses Kapitel gibt einen Überblick über diese Abstraktion der realen Welt. Es werden sämtliche Entitäten definiert, die in Zusammenhang mit der Replikation von Workflows wichtig sind.

3.1 Netzwerkmodell

Das Netzwerkmodell beschreibt die Abstraktion der realen Welt in Hinsicht auf Entitäten in einem Netzwerk, wie mobile Endgeräte, Wearables [WLL+15], Server, etc.

In dieser Arbeit abstrahieren wir die reale Welt und dessen komplexen Gegebenheiten auf ein sehr simple Form. Wir definieren einen vollständigen gerichteten Graphen G mit Knoten V und Kanten E. Hierbei unterscheiden wir bei der Menge V zwischen ausführbaren Knoten und Services.

Ein ausführbarer Knoten stellt in dem Modell eine Entität dar, die in der Lage ist ein Replikat eines Workflows zu speichern und auszuführen. Ausführbare Knoten werden später für die Auswahl einer Selektion an Knoten, um Fehlertoleranz zu gewährleisten, herangezogen. Weiter muss noch unterschieden werden zwischen mobilen und stationären ausführbaren Knoten. Ein mobiler ausführbarer Knoten kann zum Beispiel ein Smartphone oder ein Wearable sein. Unter einem stationären ausführbaren Knoten verstehen wir ein Knoten, der fest in der Infrastruktur verankert ist und sich nicht bewegt. Beispiele hierfür wären ein Desktop-PC oder ein Server.

Ein Service ist auch ein Knoten des Graphen, jedoch ist er nicht in der Lage ein Replikat eines Workflows zu speichern und folglich auch nicht in der Lage diesen auszuführen. Services dienen Aktivitäten als Interface, um bestimmte Informationen zu erlangen. Auch hier gibt es mobile und stationäre Services.

Die Unterschiede zwischen mobilen und stationären Knoten sind zum einen die Mobilität und zum anderen die Anbindung. Stationäre Knoten haben eine bessere Anbindung als mobile Knoten.

Die Unterscheidung ob ein Knoten mobil oder stationär ist, ist lediglich für spätere Berechnungen und Evaluationen interessant. Um das Netzwerkmodell möglichst einfach zu halten, wird für die Veranschaulichung in Abbildung 3.1, der gegebenen Entitäten des Netzwerkmodells, nicht auf diese Unterscheidung eingegangen.

Die Verbindung zwischen zwei Knoten werden durch Kanten repräsentiert. Da das Routing in einem Netzwerk zwei unterschiedliche Pfade zwischen zwei Knoten ergeben könnte, definieren wir immer zwei gegenläufige Kanten zwischen zwei Knoten.

Es sind also folgende Entitäten gegeben:

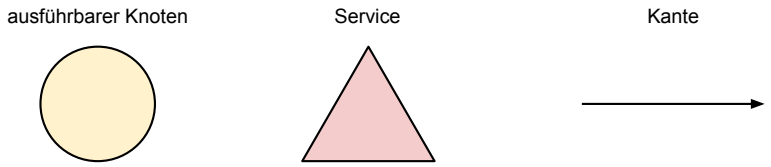

Abbildung 3.1: Entitäten des Netzwerkmodells

Aus den definierten Entitäten und der Abstraktion der realen Welt auf einen vollständigen gerichteten Graphen G, ergibt sich nun der in Abbildung 3.2 zu sehende Beispiel-Graph.

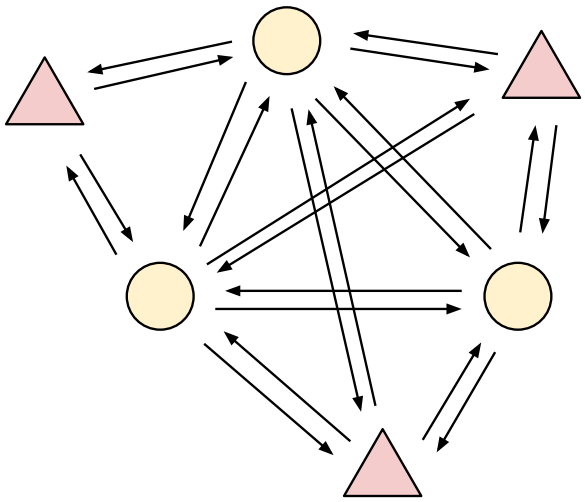

Abbildung 3.2: Beispiel-Graph mit sechs Knoten (drei ausführbare Knoten und drei Services)

Der Graph hat insgesamt sechs Knoten, drei ausführbare Knoten und drei Services. Alle Knoten im Graphen sind miteinander verbunden. Jeder Knoten und jede Kante hat zudem noch einen Wert r.

Zudem definieren wir einen Parameter r, der die Qualität der jeweiligen Entität angibt.

$$r = [0, 1]$$

Was genau dieser Parameter angibt wird an dieser Stelle noch nicht festgelegt. Denkbar wären Einheiten wie zum Beispiel:

- Latenz

- Bandbreite
- Rechenleistung
- Speicherplatz
- Ausfallsicherheit

Natürlich passen nicht alle dieser Einheiten zu jeder der definierten Entitäten. Speicherplatz auf eine Verbindung zwischen zwei Knoten anzuwenden wird keinen Sinn ergeben. Man wird auch keinen Nutzen daraus ziehen können die Einheit der Bandbreite auf Knoten anzuwenden.

Für die spätere Erstellung einer Metrik, zur Berechnung der Qualität einer Selektion, definieren wir an dieser Stelle die wichtigen Variablen und Funktionen für das Netzwerkmodell.

Der gerichtete Graph G wird definiert durch:

$$G = (V, E, f_n, f_e)$$

Die Menge der Knoten V ist die Vereinigungsmenge der Mengen der ausführbaren Knoten V_N und der Menge an Services V_S.

$$V = V_N \cup V_S$$

$$V = \{v_1, ..., v_i\}$$

$$V_N = \{n_1, ..., n_i\}$$

$$V_S = \{s_1, ..., s_j\}$$

Es gibt keinen Knoten, der sowohl Element der Menge V_N sowie der Menge V_S ist.

$$V_N \cap V_S = \emptyset$$

Parameter r eines Knoten:

$$f_n(v_i) = [0, 1]$$

Parameter r einer Kante:

$$f_e(v_i, v_j) = [0, 1]$$

3.2 Erweitertes Netzwerkmodell

Für die spätere Simulation muss das soeben definierte Systemmodell noch etwas angepasst werden.

Die wesentlichen Erweiterungen sind im Folgenden aufgezählt.

- mobile oder stationäre Knoten
- Cluster
- Heatmaps für Mobilnetz- und Infrastrukturnetzabdeckung

Um das Systemmodell der realen Welt, für die Simulation etwas ähnlicher zu gestalten, werden die Knoten des Graphen noch weiter unterteilt. Dies verändert jedoch keine bereit getroffenen Aussagen zu dem Graphen an sich. Es wird definiert, dass es Knoten gibt die sich bewegen und Knoten die sich nicht bewegen. Die Knoten die sich bewegen, nennt man auch mobile Knoten, die die sich nicht bewegen stationäre Knoten. Diese stationären Knoten bezeichnet man als Infrastrukturknoten. Sowohl ausführbare Knoten als auch Services können beides, mobil und stationär sein. Ein Knoten kann jedoch nicht beides gleichzeitig sein.

Zusätzlich werden auch Cluster eingeführt. Cluster werden anhand Knoten erstellt, die zu Beginn der Simulation nah beieinander sind. Die Position für Knoten ist auch unabhängig von den Clustern sehr wichtig, da Knoten die in einem bestimmten Radius zueinander liegen, eine bessere Verbindung haben. Dies soll eine Verbindung über ein LAN-Netzwerk oder gar Bluetooth simulieren. Eine solche “Direktverbindung“ wird mit einer besseren Ausfallsicherheit berechnet, hierzu wird ein Gauss-Funktion mit einem Durchschnitt von 0.999 und einer Standardabweichung von 0.1 verwendet.

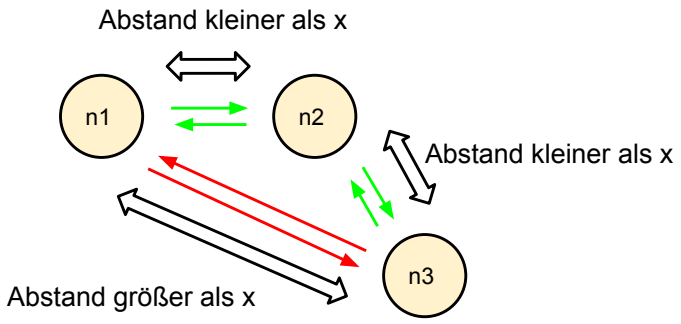

Abbildung 3.3: Simulation von Direktverbindungen durch Abstandsmessung

Als dritter Erweiterung kommen sogenannte Heatmaps hinzu. Diese Heatmaps definieren die Ausfallsicherheiten an einer bestimmten Position.

Diese Heatmaps geben an, wie die Verbindung zum restlichen Netzwerk an einer bestimmten Position ist. Die Berechnung einer Verbindung in der späteren Simulation wird folglich nicht mehr durch ein Wert definiert, sondern streng genommen durch zwei. Die Verbindung wird zusätzlich noch mit einem Gauss-Wert multipliziert, damit die Hin- und Rückverbindung zweier Knoten nicht denselben Wert haben. Zum einen der Wert, der entsprechenden Heatmap (Mobil oder Infrastruktur) an der Position

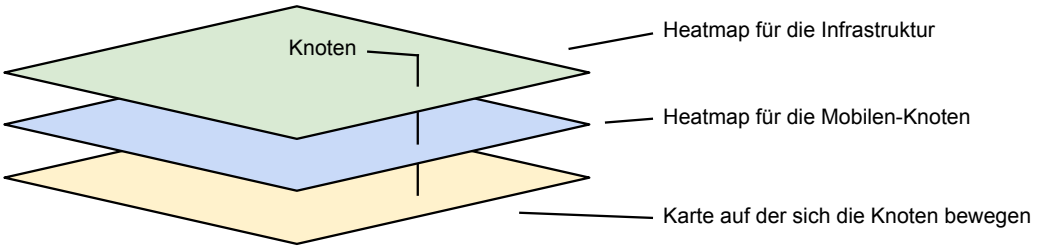

Abbildung 3.4: Heatmaps für Mobil- und Infrastrukturnetz

des ersten Knotens, multipliziert mit dem Wert, der entsprechenden Heatmap an der Position des zweiten Knotens. Hier ist zu beachten, dass beide Heatmaps für eine Berechnung herangezogen werden. Beispiel hierfür ist die Verbindung zwischen einem ausführbaren Knoten, der auf der Infrastruktur liegt, und einem mobilen Knoten. Hier wird der Wert für den ersten Knoten aus der Heatmap für die Infrastruktur abgerufen, der Wert für den zweiten Knoten aus der für Mobilknoten. Die Werte der Heatmaps werden wie folgt erstellt. Für die Heatmap für mobile Knoten, werden zufällige Werte als Ankerpunkte genommen. Die Ankerpunkte der Heatmap für Infrastrukturknoten stammen von einer Gauss-Funktion. Da diese Ankerpunkte nicht für eine Karten-Abdeckung ausreichen, werden die Zwischenwerte mit einer Rauschfunktion interpoliert. Perlin-Noise wurde hier zur Hand genommen.

Die Werte der Infrastrukturkarte sind nun sehr gut, da die verwendete Gauss-Funktion einen Durchschnittswert von 0.998 und eine Standardabweichung von 0.025 verwendet.

Die Werte der Heatmap für Mobil-Knoten hingegen ist sehr schlecht, da sie mit zufälligen Ankerpunkten befüllt wurde. Bei zufälligen Werten ist ein Durchschnitt von 50% anzunehmen, was nicht dem realen Durchschnittswert einer Netzabdeckung entspricht. Wir modifizieren daher alle Werte der Heatmap mit einer weiteren Funktion.

$$f(x) = x/(x + 0.075)$$

Diese Funktion lässt den Durchschnitt der Heatmap auf 80% ansteigen, der Median liegt bei 83%. Für ein Mobilfunknetz sind das eventuell noch keine Werte die man so in der Realität finden würde, da es aber in der Simulation keine Signalabschwächung gibt, sind diese Werte in Ordnung.

Für die Berechnung der Ausfallsicherheit, der Knoten selbst, wurde eine Gauss-Funktion mit einem Durchschnitt von 0.9 und einer Standardabweichung von 0.1 benutzt.

3.3 Workflowmodell

Das Workflowmodell definiert den für die Ausführung vorliegenden Workflow.

Ein Workflow ist eine Menge von Aktivitäten.

$$W = \{a_1, ..., a_i\}$$

In Abbildung 3.5 sieht man einen Workflow, bestehend aus vier Aktivitäten. Wie man anhand der Pfeile sehen kann, ist die Reihenfolge der Ausführung der einzelnen Aktivitäten vorgegeben. Aktivität a2 darf erst ausgeführt werden, nachdem die Aktivität a1 ausgeführt wurde.

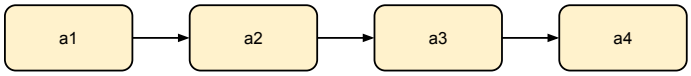

Abbildung 3.5: Beispiel Workflow mit vier Aktivitäten

Eine Aktivität ist gleichzusetzen mit einem Prozess. Sie kann sich aus einer Menge an Services bedienen, um ihre Aufgabe zu erfüllen. Nehmen wir als Beispiel eine Aktivität, die das Wetter für den nächsten Tag ausgeben soll. Es gibt mehrere Wetterstationen in der Umgebung, bei denen die gewünschte Information erfragt werden kann. Die Aktivität kann also aus einer Menge an Wetterstationen auswählen, von welcher sie die Information beziehen möchte. Einige dieser Wetterstationen haben eine schlechtere Anbindung an das Netzwerk als die anderen und brauchen daher länger um auf die Anfrage zu reagieren. Folglich wäre es logisch, dass die Aktivität eher eine der Wetterstationen wählt, die in unmittelbarer Entfernung ist und eine gute Anbindung an das Netzwerk hat, um die Antwort in der schnellst möglichen Zeit zu erhalten.

Eine Aktivität kann aus einer Menge an Services auswählen, um an die gewünschte Information zu kommen (Abbildung 3.6). Sie besitzt sozusagen eine Menge von Services.

$$S_A = \{s_1, ..., s_i\}$$

Die Funktion $BS_A(a_i)$ liefert für eine Aktivität den besten zugehörigen Service zurück.

$$BS_A(a_i) \rightarrow max(S_A)$$

Wie die Berechnung des besten Services zur Stande kommt, ist abhängig von der Einheit von r und wird in **??** genauer erklärt.

Für die spätere Simulation gehen wir davon aus, dass eine Aktivität nur eine Anfrage gleichzeitig senden kann. Eine Aktivität kann also zu einem Zeitpunkt t nur mit einem der wählbaren Services kommunizieren.

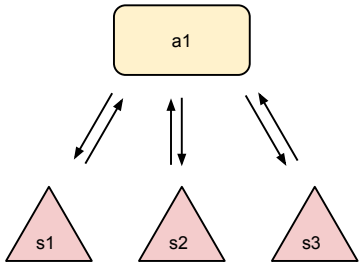

Abbildung 3.6: Eine Aktivität, der drei Services zur Verfügung stehen.

3.4 Ausführungsmodell

Das Ausführungsmodell beschreibt die vier verschiedenen simulierten Ausführung der replizierten Workflows. Die Ausführungen werden später für die Evaluation der erstellten Heuristiken benötigt. Näheres dazu steht in Kapitel 7 – Evaluation.

- Statische Ausführung mit starker Konsistenz
- Statische Ausführung mit schwacher Konsistenz
- Dynamische Ausführung mit starker Konsistenz
- Dynamische Ausführung mit schwacher Konsistenz

Eine statische Ausführung verläuft genauso wie eine dynamischen Ausführungen. Der einzige Unterschied ist, dass die statischen Ausführungen vor dem eigentlichen Beginn der Simulation durchgeführt werden und die dynamischen, während der Simulation.

3.4.1 Ausführungen mit starker Konsistenz

Ausführungen mit starker Konsistenz, lassen für den gesamten Zeitraum der Ausführung keine Inkonsistenz der einzelnen Aktivitäten und somit des Workflows, zu. Wie auch schon in Kapitel 2 beschrieben, wird nach jeder Aktivität der Zustand aller Replikate des Workflows auf Konsistenz geprüft. Für den Fall, dass ein Replikat eine Aktivität erfolgreich ausführen und den Fortschritt an f andere Replikate senden konnte, so wird im nächsten Schritt mit der nächsten Aktivität fortgefahren.

3.4.2 Ausführungen mit schwacher Konsistenz

Ausführungen mit schwacher Konsistenz, lassen während der Ausführung Inkonsistenzen innerhalb des Workflows zu. Lediglich bei der letzten Aktivität gilt hier wieder eine starke Konsistenz. Dies hat den Vorteil, dass Knoten die eine Aktivität erfolgreich ausgeführt haben, nicht auf f weitere Knoten warten müssen, bis diese ihr Update erhalten haben. Wie auch schon in Kapitel 2 beschrieben, wird nach lediglich bei der letzen Aktivität der Zustand aller Replikate des Workflows auf Konsistenz geprüft. Falls dies der Fall ist, so ist der Workflow beendet.

4 Problemstellung

Anhand der Motivation dieser Arbeit und des in Kapitel 3 beschriebenen Systemmodells, kann nun die exakte Problemstellung formuliert werden.

Die Idee war es durch gezielte Replikation von Workflows, Fehlertoleranz zu gewährleisten. Gezielte Replikation heißt so viel wie die durchdachte Verteilung von Replikaten an Knoten in einem Netzwerk, sodass diese Verteilung die optimale Verteilung ist. Es gibt folglich keine andere Verteilung die ein besseres Ergebnis liefert, als die gewählte Verteilung. Um diese optimale Verteilung berechnen zu können, gehen wir von einer festen und gegebenen Anzahl an Knoten aus, auf die der Workflow repliziert werden soll.

Die Aufgabe ist es nun, n Knoten aus allen k ausführbaren Knoten auszuwählen, sodass die optimale Auswahl von n aus k getroffen wurde. Die optimale Auswahl bedeutet soviel wie die Auswahl von n aus k, sodass keine andere Auswahl von n aus k existiert, die besser ist.

5 Technischer Teil - Statisch

Durch die Definition der Problemstellung im vorherigen Kapitel, ist einen vollständigen gerichteten Graphen G, mit gewichteten Knoten V und Kanten gegeben, aus dem die optimale Auswahl von n aus k Knoten gefunden werden muss. Wobei die Menge k nur die Menge der ausführbaren Knoten ist.

Was ist die optimale Lösung?

Im Folgenden bezeichnen wir die Auswahl von Knoten als Selektion. Die optimale Lösung ist eine Selektion n aus k, ohne dass es eine andere Selektion n gibt, die besser ist. Um zwei Selektionen miteinander vergleichen zu können wird eine Metrik benötigt.

5.1 Metrik zur Berechnung der Qualität einer Selektion

Da das Augenmerk der Diplomarbeit auf der Fehlertoleranz durch Replikation liegt, wird die Einheit von r, der im Kapitel 3 – Das Systemmodell definierte Konstante, als Ausfallsicherheit definiert. Es wird innerhalb des Graphen folglich mit Wahrscheinlichkeiten gerechnet.

Die Berechnung der Wahrscheinlichkeit, mit der ein Workflow von einer Selektion erfolgreich ausgeführt wird, wird in mehrere Schritte unterteilt. Damit ein einzelner Knoten eine Aktivität erfolgreich ausführt, muss zum einen der Aufruf eines der zur Aktivität gehörenden Services funktionieren, zum anderen muss der Knoten anschließend sicherstellen, dass die restlichen Knoten der Selektion die abgerufenen Informationen ebenfalls erhalten. Dies beinhaltet, dass das Senden und das Empfangen der Informationen funktioniert. Veranschaulicht wird dieser Schritt anhand der Abbildung 5.1.

Um die Wahrscheinlichkeit mathematisch berechnen zu können, werden folgende Variablen und Funktionen definiert. Die Folgenden Variablen haben kein Bezug zu den Variablen n und k, die für die Selektion verwendet werden.

Wahrscheinlichkeit der Ausführung eines Services:

$$Rel_{N,S}(n_i, a_j)) = f_n(n_i) * f_n(BS_A(a_j)) * f_e(n_i, BS_A(a_j)) * f_e(BS_A(a_j), n_i)$$

Wahrscheinlichkeit der Ausführung einer Aktivität durch einen Knoten:

$$Rel_{N,A}(n_i, a_j) = Rel_{N,S}(n_i, a_j) * \prod_{Knoten der Selektion, n_i \neq n_j}^{k} (f_n(n_k) * f_e(n_i, n_k))$$

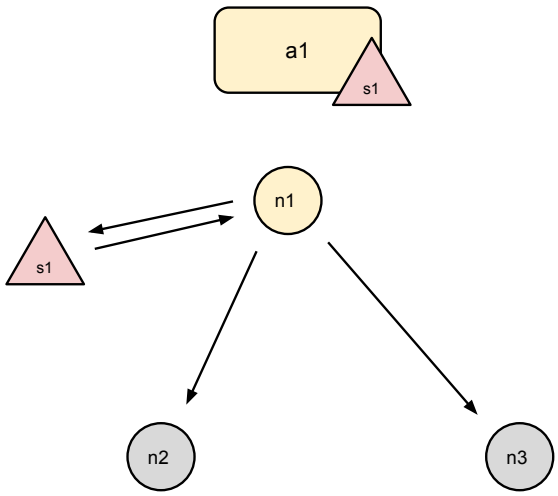

Abbildung 5.1: Ausführung einer Aktivität durch einen Knoten.

Um die Wahrscheinlichkeit zu errechnen, mit der mindestens ein Knoten, der in der Selektion enthaltenen Knoten, die Aktivität ausführen kann, so wie es oben definiert wurde, wird folgende Formel definiert.

$$Rel_{G,A}(a_i) = 1 - \prod_{Knoten der Selektion}^{j} (1 - Rel_{N,A}(n_j, a_i) * Rel_{Conn})$$

Um diese Berechnung nun noch auf den gesamten Workflow anzuwenden, definieren wir noch folgende Funktion:

$$Rel_{G,W}(w) = \prod_{Menge der Aktivitäten}^{i} (1 - Rel_{G,A}(a_i))$$

Die Variable w definiert den Workflow und somit die Menge an Aktivitäten, die es auszuführen gilt.

Diese Formel definiert unsere Metrik der theoretischen Qualität eine Selektion. Die Werte, die hier erwartet werden liegen zwischen 0 und 1.

$$Rel_{G,W}(w) \rightarrow [0, 1]$$

5.2 Weitere Metriken

Um die optimale Lösung und die Heuristiken später nicht nur anhand ihrer theoretischen Qualität bewerten und vergleichen zu können, werden weitere Metriken eingeführt.

Folgende Metrik werden zusätzlich definiert:

- Berechnungszeit der Selektion einer Heuristik

- Statische Ausführungszeit der Selektion
  - mit starker Konsistenz
  - mit schwacher Konsistenz
- Dynamische Ausführungszeit der Selektion
  - mit starker Konsistenz
  - mit schwacher Konsistenz
- Anzahl der Knotenwechsel während der Laufzeit

Die theoretische Qualität der Selektion gibt die Wahrscheinlichkeit an, mit der eine Selektion den Workflow erfolgreich ausführen wird. Um diese Qualität mit den errechneten Qualitäten der anderen Heuristiken besser vergleichen zu können, haben wir zusätzlich die Metrik der Berechnungszeit einer Selektion definiert. Wenn man diese zwei Werte in Zusammenhang setzt, kann sozusagen das Preis-Leistungsverhältnis beziehungsweise das Nutzen-Zeitverhältnis errechnet werden.

Zudem führen wir noch die Metriken der Ausführungszeiten ein. Anhand dieser Metriken, kann überprüft werden, ob die theoretische Qualität wirklich aussagekräftig ist. Für jeder Art der Ausführung definieren wir eine eigene Metrik.

Die letzte Metrik, die wir definieren um die Heuristiken besser evaluieren zu können, ist die Anzahl der Knotenwechsel während der Laufzeit. Hier muss zusätzlich definiert werden, ab wann ein solcher Knotenwechsel beziehungsweise eine solche Neuplanung der Selektion stattfinden soll.

Für den Fall, dass die theoretische Qualität der Selektion während der Laufzeit nur noch 80% des ursprünglichen Wertes beträgt, so wird eine Neuplanung der Selektion durchgeführt.

5.3 Optimale Lösung

Um die optimale Selektion zu ermitteln, muss das Problem genauer betrachtet werden. Zuerst wurde rein logisch betrachtet, wie viele mögliche Selektionen es gibt, sprich wie viele mögliche Mengen n aus k existieren. Hierzu wurde der Beispiel-Graph in Abbildung 3.2 herangezogen.

k ist uns nun durch die Menge der ausführbaren Knoten im besagten Graphen gegeben. Wir definieren nun $n = 2$.

Es müssen folglich zwei der drei ausführbaren Knoten gewählt werden. Um die Kombinationen fest zu halten, versehen wir jeden Knoten mit einem Namen. Wir haben also $k = \{n_1, n_2, n_3\}$ gegeben. Daraus ergeben sich folgenden Kombinationen:

- n_1, n_2
- n_1, n_3
- n_2, n_3

Wenn wir nun den Graphen um einen ausführbaren Knoten erweitern, $k = \{n_1, n_2, n_3, n_4\}$, so ergeben sich folgende Kombinationen:

- n_1, n_2
- n_1, n_3
- n_1, n_4
- n_2, n_3
- n_2, n_4
- n_3, n_4

Bei den Kombinationen wird die Reihenfolge der Knoten nicht beachtet. Der Grund hierfür ist, dass die Auswahlen n_1, n_2 und n_2, n_1 ein und dieselbe sind.

Für die erhaltenen Kombinationen gibt es ein sehr bekanntes Beispiel der Kombinatorik, die des Lottos. Wie haben hier mit 2 aus 3, eine vereinfachte Form des Lottos, 6 aus 49.

Wie viele Möglichkeiten es nun für die Auswahl von n Objekten aus einer Menge k gibt (ohne Zurücklegen und ohne Beachtung der Reihenfolge), lässt sich mit Hilfe des Binomialkoeffizient berechnen.

$$\binom{k}{n}$$

Für relativ kleine Werte für n aus k ist die optimale Lösung sehr leicht zu errechnen. Bei den Dimensionen beim Lotto wird dies schon schwieriger. Hier gäbe es bei 6 aus 49 exakt 13.983.816 mögliche Selektionen, die es zu vergleichen gäbe. Der Vergleich aller Selektionen miteinander würde folglich $13.983.816^{13.983.816} = 1.3983816 * 10^7$ Vergleiche bedeuten. Da dies keine praktikable Lösung ist, wird nach einer Funktion zur Berechnung gesucht. Wenn man bedenkt, dass eine Selektion von 6 Knoten aus einer Menge von 49 Knoten noch weit unter dem Maßstab der realen Welt liegt, gehen wir davon aus, dass dieses Problem nicht in Polynomialzeit gelöst werden kann.

Ein Versuch den Graphen so weit zu vereinfach, dass der Graph lediglich aus ausführbaren Knoten und Kanten bestehen würde und nur die Kanten mit einem Gewicht belegt wären, zeigte, dass die Annahme, das Problem sei nicht in Polynomialzeit zu lösen, sehr wahrscheinlich ist.

Es wurde andere Literatur herangezogen [TLX+06] [BR01] [LLS08] [KL05] [RDR10] [LC12], um verwandte Themen zu finden, die sich mit der Lösung eines ähnlichen Problems befassen. Die Recherche ließ vermuten, dass es sich hierbei um ein NP-Vollständiges Problem handelt. Daher wurde nun ein Problem gesucht, auf das sich die in Kapitel 4 definierte Problemstellung reduzieren lässt. [Sch08]

In [AGKW07] und [Sch08] wurde ein Problem beschrieben, das der definierte Problemstellung ähnlich ist. Das NP-Vollständige Problem Clique. [AGKW07] kommt noch etwas näher an die Problemstellung heran, da dort ein gerichteter Graph mit Kantengewichten gegeben ist.

Die Definition des dort gegebenen Clique-Problems lautet wie folgt:

Definition 5.3.1
Given a complete graph G = (V,E) with n nodes and unrestricted edge weights $c_{i,j}$, find a subclique of G with b or fewer nodes such that the sum of the weights in the subclique is maximized. [AGKW07, p. 593, 2. Problem definition]

Da in unserem Fall mit der Ausfallsicherheit gerechnet wird und das "maximum edge weight clique problem" mit der Summe der Kantengewichte rechnet, sei an dieser Stelle zu sagen, dass durch Anwendung des Logarithmus eine Multiplikation durch eine Addition ersetzt werden kann.

$$\log_b(x \cdot y) = \log_b(x) + \log_b(y)$$

Nehmen wir nun an, dass der Fall eintritt, bei dem alle Knoten des Graphen eine Ausfallsicherheit von 1 und alle Kanten eine Ausfallsicherheit von 0.9 besäßen, so würde durch das Hinzufügen eines beliebigen Knotens, immer genau den gleichen Faktor ausmachen. So wäre jeder neu hinzugefügte Knoten zu einer Selektion eine neue maximale Clique.

Rechnen wir die Services in die Knoten mit ein und anschließend die Knoten auf die Kanten, so haben wir einen exakt gleichen Graphen der in der obigen Definition beschrieben wird.

Das Problem, welches in dieser Arbeit behandelt wird, ist folglich mindestens ein NP-vollständiges Problem. Wir suchen daher nach Heuristiken, um der optimalen Lösung so nah wie möglich zu kommen.

Um die optimale Lösung zu erhalten, müssen also tatsächlich alle möglichen Selektionen miteinander verglichen werden.

5.4 Simulierte Abkühlung

Die Simulierte Abkühlung ist eine heuristisches Optimierungsverfahren um eine Approximation, genauer gesagt eine Annäherung an die optimale Lösung. [Egl90]

Die Grundidee der Simulierten Abkühlung (eng. simulated annealing), ist die Nachbildung eines Abkühlungsprozesses aus der Werkstoffkunde. Erhitztes Metall wird langsam abgekühlt. Umso heißer das Metall ist, umso leichter lässt es sich formen. Je niedriger die Temperatur des Metalls wird, desto schwieriger wird es das Metall noch zu bearbeiten.

Auf unseren Graphen angewandt, bedeutet dies, dass die gewählte Selektion mit einer hohen Wahrscheinlichkeit durch die neue Selektion ausgetauscht wird, auch wenn die neue Selektion schlechter ist. So können lokale Maxima übersprungen werden. Je weiter die Simulation voran schreitet, desto niedriger wird auch die Wahrscheinlichkeit, dass eine Selektion durch eine schlechtere Selektion ersetzt wird.

Dieser heuristische Algorithmus hat zur Folge, dass zu Anfang der Simulation locale Maxima sehr einfach übersprungen werden und eine Lösung, näher am optimalen Maximum, gefunden werden können.

Der grobe Ablauf der Simulierten Abkühlung ist wie folgt. Pro Simulationsschritt wird ein beliebiger Knoten der Selektion ausgetauscht und geschaut ob diese Selektion besser ist. Falls die Selektion

besser ist, so wird diese Selektion gespeichert und es wird der nächste Schritt eingeleitet. Falls die Selektion nicht besser ist, so wird mit einer Wahrscheinlichkeit x die Selektion trotzdem als neue Selektion gespeichert. Die Wahrscheinlichkeit x ist hierbei abhängig von der Temperatur. Nach dieser Entscheidung über eine neue Selektion, wird die Temperatur reduziert und der Ablauf fängt von vorne an.

5.5 Zufällige Selektion

Die zufällige Selektion von Knoten ist nicht wirklich eine Heursitk. Sie dient zur Veranschaulichung und zum Vergleich für die anderen Heuristiken. Falls eine Heuristik schlechtere Werte liefert als die zufällige Selektion von Knoten, so ist diese nicht praktikabel. Diese Heuristik dient sozusagen als Ausschlusskriterium für andere Heuristiken.

5.6 Heuristik: Knoten-Service

Die optimale Lösung für das Problem in Kapitel 4 – Problemstellung beschrieben zu finden, ist wie schon erwähnt nicht praktikabel. Das heuristische Optimierungsverfahren liefert zwar schnellere Lösungen, ist aber bei einem sehr großen Graphen und einer großen Selektion sehr wahrscheinlich auch nicht praktikabel.

Die Heuristik die hier beschrieben wird, versucht anhand von sehr wenig Informationen eine geeignete Selektion zu finden. Wie der Name schon sagt werden hier nur die Informationen von den Knoten und den zugehörigen Services benötigt.

Die Heuristik verfolgt folgende Grundidee. Jeder Knoten wird einzeln betrachtet. Es wird ausgerechnet, wie hoch die Wahrscheinlichkeit ist, dass der Knoten jeweils den besten Service einer Aktivität ausführen kann. Und das für alle Aktivitäten. Dieser Wert gibt einen Anhaltspunkt wie gut dieser Knoten zur alleinigen Ausführung des gesamten Workflow geeignet ist. Abbildung 5.5 stellt die Grundidee bildlich dar.

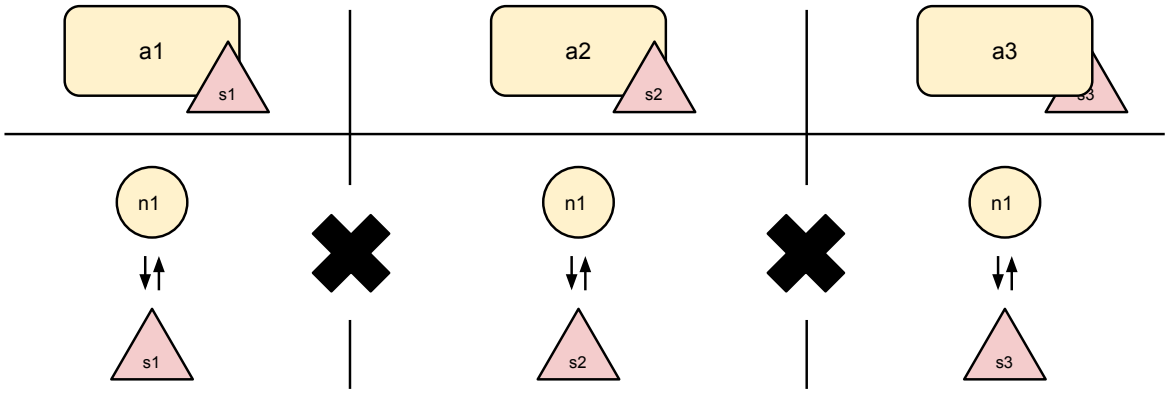

Abbildung 5.2: Grundidee der Heuristik: Knoten-Service

Mathematisch betrachtet sieht die Berechnung wie folgt aus:

$$Bewertung_N(n_i) = \prod_{Mengeder Aktivitäten}^{j} (Rel_{N,S}(n_i, a_j))$$

Wobei hier für die Funktion $Rel_{N,S}$ gilt:

$$Rel_{N,S}(n_i, a_j) = f_n(n_i) * f_n(BS_A(a_j)) * f_e(n_i, BS_A(a_j)) * f_e(BS_A(a_j), n_i)$$

Anhand dieser Bewertung eines einzelnen Knoten, wir eine Liste erstellt. Für die Selektion nimmt die Heuristik nun die obersten beziehungsweise besten Knoten aus der Liste und fügt sie der Selektion hinzu.

Die Überlegung die hinter dieser Heuristik steckt, ist, dass Knoten, genauer gesagt Netzwerk-Entitäten, die eine gute Anbindung an die Services eines Workflows haben, sehr wahrscheinlich auch eine gute Anbindung an den Rest des Netzwerkes haben. Es wäre unlogisch viel Geld in eine Netzwerk-Entität und dessen Anbindung an nur wenige ausgewählte andere Netzwerk-Entitäten zu investieren. In der Regel haben gute Knoten, die zu ausgewählten Services eine gute Verbindung haben, auch zum Rest des Netzwerkes eine gute Verbindung.

Diese Heuristik benötigt am wenigsten Information aller in dieser Bearbeiteten Heuristiken und müsste folglich die Schnellste sein. Wie gut sie im Vergleich zu den Anderen ist, wird sich in Kapitel 7 – Evaluation zeigen.

5.7 Heuristik: Knoten-Service-Kanten

Die zweite Heuristik, fasst den Grundgedanken der Heuristik, Knoten-Service, wieder auf und verfeinert ihn noch ein wenig.

In der vorher vorgestellten Heuristik, wurde angenommen, dass ein Knoten, der eine gute Anbindung zu einigen ausgewählten Services, auch eine gute Anbindung zum Rest des Netzwerkes hat. Diese Annahme wird in dieser Heuristik nun in die Bewertung eines Knoten mit aufgenommen.

Sie nutzt zur Berechnung die Ausführwahrscheinlichkeit eines Services, mit Es wird ausgerechnet, wie hoch die Wahrscheinlichkeit ist, dass der Knoten jeweils den besten Service einer Aktivität ausführen kann und zusätzlich wie gut die Anbindung an den Rest des Netzwerkes ist. Auch hier gilt diese Teilberechnung für alle Aktivitäten. Die Teilberechnungen werden multipliziert um die endgültige Bewertung des Knoten zu erhalten.

Bei dieser Art Berechnung werden die ausgehenden Kanten eines Knoten so oft in die Berechnung mit aufgenommen wie es Aktivitäten gibt. Je größer der Workflow bei dieser Heuristik ist, umso mehr wird die Position des Knotens, sprich die Anbindung an den Rest des Netzwerkes, gewichtet. Dies hat zum einen den Hintergrundgedanken, dass mehr auf das Versenden der Updates eingegangen wird, zum anderen wird bei der späteren Ausführung die Berechnung genauso stattfinden. Da sich während der späteren Simulation die Ausfallsicherheiten der Kanten ändern, da sich die Knoten bewegen, ist dies zudem näher an der Realität, wie wenn die Kanten nur ein mal statisch in die Berechnung

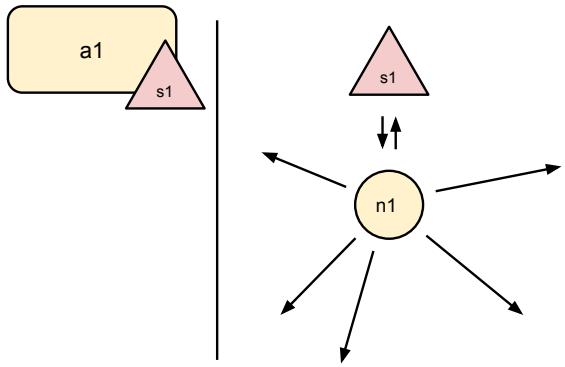

Abbildung 5.3: Grundidee der Heuristik: Knoten-Service-Kanten

mit einfließen. Dennoch kann es nicht passieren, dass ein Knoten in die Selektion genommen wird, der eine außerordentlich schlechte Verbindung zu den ausgewählten Services hat, da die Werte im Bereich von 0 bis 1 liegen und miteinander multipliziert werden.

Mathematisch betrachtet sieht die Berechnung der Heuristik wie folgt aus:

$$Rel_{N,Conn}(n_i) = \prod_{Alle ausführbaren Knoten, i \neq j}^{j} f_l(n_i, n_j)$$

$$Rel_{N,S}(n_i, a_j) = f_n(n_i) * f_n(BS_A(a_j)) * f_e(n_i, BS_A(a_j)) * f_e(BS_A(a_j), n_i)$$

$$Bewertung_N(n_i) = \prod_{Menge der Aktivitäten}^{j} (Rel_{N,S}(n_i, a_j * Rel_{N,Conn}(n_i)))$$

5.8 Heuristik: Knoten-Service-n-Kanten

Die dritte erstellte Heuristik, ist eine Modifizierung der zweiten Heuristik, Knoten-Service-Kanten.

Bei der vorherigen Heuristik, wurde die Anbindung eines Knoten, an das gesamte Netzwerk in die Bewertung eingebunden. Die Berechnung wird nun insofern modifiziert, dass nicht mehr alle Kanten des Netzwerkes in die Berechnung mit einfließen, sondern nur noch so viele Kanten, wie es Knoten in der Selektion gibt. Der Name dieser Heuristik enthält zwar ein n welches die Größe der Selektion widerspiegeln soll, für die Berechnung werden jedoch nur $n - 1$ Kanten mit einbezogen da aus Sicht eine Knotens nur $k - 1$ und nicht k Kanten existieren. Bei einer Größe der Selektion von drei, würde pro Knoten zwei Kanten in die Berechnung mit einbezogen werden.

Die restliche Berechnung in Hinsicht auf die Ausführung einer Aktivität und die Zusammenfassung der Bewertung eines Knotens bezüglich einer Aktivität, mit den Teilbewertungen des Knotens für die anderen Aktivitäten im Workflow, bleiben gleich.

Bei dieser Heuristik werden die ausgehenden Kanten eines Knoten ebenfalls so oft in die Berechnung mit aufgenommen wie es Aktivitäten gibt. Der Unterschied hier ist jedoch, dass nur $n-1$ ausgehende Kanten mit einbezogen werden. Es ist außerdem anzumerken, dass diese $n-1$ Kanten nicht zufällig gewählt werden, sondern die Kanten ausgewählt werden mit der höchsten Ausfallsicherheit beziehungsweise Wahrscheinlichkeit.

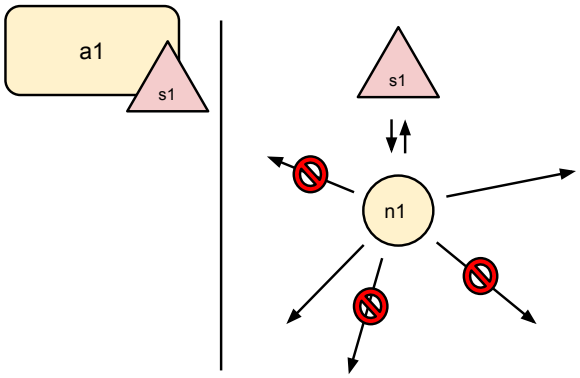

Abbildung 5.4: Grundidee der Heuristik: Knoten-Service-n-Kanten, bei einer Selektion $n = 3$

Mathematisch betrachtet sieht die Berechnung dieser Heuristik wie folgt aus: Die Funktion $Rel_{N,Conn(n-1)}(n_i)$ liefert das Produkt der Ausfallsicherheiten für die $n-1$ besten Kanten für n_i zurück.

$$Rel_{N,S}(n_i, a_j) = f_n(n_i) * f_n(BS_A(a_j)) * f_e(n_i, BS_A(a_j)) * f_e(BS_A(a_j), n_i)$$

$$Bewertung_N(n_i) = \prod_{Mengeder Aktivitäten}^{j} (Rel_{N,S}(n_i, a_j * Rel_{N,Conn(n-1)}(n_i)))$$

5.9 Heuristik: Cluster

Die Cluster Heuristik greift, im Vergleich zu den drei vorherigen Heuristiken, einen ganz neuen Aspekt mit auf.

Wie der Name schon sagt bezieht diese Heuristik Cluster mit ein. Hierzu muss zuerst ein mal definiert werden, was genau ein Cluster ist.

In der Informatik versteht man unter einem Cluster eine Menge miteinander vernetzter autonomer Computer, die sich wie eine virtueller Entität verhalten. Es besteht folglich aus mindestens zwei Knoten, die miteinander verbunden sind. In der Regel stellt sich für den Anwender ein solches Cluster als einzelner Computer dar.

Laut dieser Definition wäre unser kompletter Graph ein einziges Cluster, da alle Knoten autonom und jeweils miteinander verbunden sind. Für diese Heuristik muss daher die gängige Definition eines Clusters angepasst werden.

Wir definieren ein Cluster wie folgt:

Definition 5.9.1
Ein Cluster ist eine Menge miteinander vernetzter autonomer und zugleich mobiler Entitäten, die in eine Maximalabstand m zueinander stehen und diesen weder im statischen noch im dynamischen Zustand überschreiten.

Die Inspiration der Idee dieser Heuristik stammt aus [HCP03]. Dort werden Knoten die ein ähnliches, oder gar gleiches Bewegungsbild haben, zu einem Cluster zusammengefügt. Diese Knoten bewegen sich mit derselben Geschwindigkeit in dieselbe Richtung. Sie bilden dadurch einen Kreis mit einem Radius und einem Mittelpunkt, den sie für die Zeit in der sie dem Cluster angehören nicht verlassen.

Um nun die Idee der Cluster in die Heuristik mit einzubauen, werden vorhandene Cluster innerhalb des Graphen stärker gewichtet. Besser gesagt, werden die Verbindungen zu Knoten außerhalb des Clusters, in dem sich der aktuell zu berechnende Knoten befindet, abgeschwächt. Folglich wird jede Kante, die zwei Knoten, die nicht innerhalb desselben Clusters liegen, durch Multiplikation mit einer Konstanten abgeschwächt.

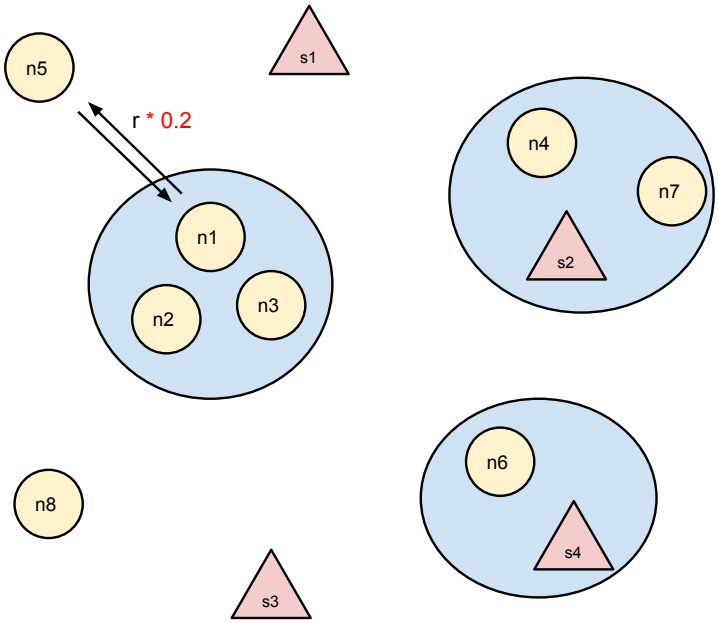

Abbildung 5.5: Berücksichtigung der Cluster, bei der Bewertung von Knoten.

Diese Abschwächung soll bewirken, dass eher Knoten in einem Cluster gewählt werden, als Knoten die keinem Cluster angehören. Es wird angenommen, dass durch diese Selektion, die Neuplanungsschritte während der Laufzeit, deutlich gesenkt werden. Zudem müsste die theoretische Qualität dieser Selektion während der Laufzeit konstanter bleiben als die Selektion der anderen Heuristiken, da die Verbindung innerhalb eines Clusters deutlich besser ist und nicht so sehr schwankt.

5.10 Andere Heuristiken

Es wurden auch andere mögliche Heuristiken in Erwägung gezogen, die allerdings nicht in der Simulation und somit auch nicht in der Evaluation enthalten sind.

5.10.1 Dezentraler Ansatz mit Koordinator

Bei diesem Ansatz wurde überlegt, wie man von der zentralen Berechnung der Selektion wegkommt.

Die Überlegung dieser Metrik besteht darin, dass nicht ein Knoten das Wissen aller anderer Knoten sammelt und dann über eine Selektion entscheidet, sonder, dass jeder Knoten selbst die Berechnung seiner Qualität übernimmt.

Man könnte beispielsweise eine Nachricht mit k-Einträgen definieren. Nun würde ein Startknoten beziehungsweise der Koordinator seine Qualität im Netzwerk berechnen und in die Nachricht schreiben. Anschließend sendet er diese Nachricht einem seiner Nachbarknoten. In einem vollständigen Graphen sind alle Knoten Nachbarknoten, es muss also drauf geachtet werden ,dass ein Knoten die Nachricht nicht zweimal bekommt. Der zeite Knoten berechnet nun ebenfalls seine Qualität, schreibt diese in die Nachricht und schickt sie weiter.

Nachdem die Nachricht ein mal bei allen Knoten im Netzwerk war, wird sie an den Koordinator zurückgeschickt. Dieser kann nun anhand der in der Nachricht gespeicherten Qualitäten entscheiden, welche Knoten der Selektion hinzugefügt werden.

6 Technischer Teil - Dynamisch

Alle im statischen Teil beschriebenen Heuristiken könnten theoretisch auch auf den dynamischen Fall angewandt werden. Was in der Theorie möglich ist, ist in diesem Falle unpraktikabel und nicht zu empfehlen. Gründe hierfür, sind die sehr langen Berechnungszeiten der optimalen Lösung und der Simulierten Abkühlung.

Dieses Kapitel befasst sich mit den Möglichkeiten, eine Selektion von Knoten während der Laufzeit zu ändern.

Diese Neuplanung, oder auch Replanning genannt, ist nötig, um Knoten zu ersetzen die keine gute Verbindung mehr zu andren Knoten haben, oder ganz ausfallen.

6.1 Neuplanung

Die Erstellung und Evaluierung von mehreren dynamischen Methoden zur Neuplanung beziehungsweise Replanning der Selektion während der Laufzeit, hätte den zeitlichen Rahmen dieser Diplomarbeit überschritten, daher wurde nur eine relativ schnelle Methode zur dynamischen Anpassung der Selektion erarbeitet.

Die Bedingung, wann eine Neuplanung stattfindet, wurde auf 80% des ursprünglichen Wertes festgelegt.

Sollte sich eine Selektion während der Ausführungszeit so weit verschlechtern, dass sie nur noch 80% der Qualität ihrer Ausgangsqualität besitzt, so wird der schlechteste Knoten dieser Selektion ermittelt und durch einen besseren ausgetauscht.

Die Ermittlung des schlechtesten Knoten einer Selektion, geschieht mit Hilfe der erstellten Metrik zur Berechnung der Qualität einer Selektion.

Sollte nun ein Knoten ausgetauscht werden, so wird dieser erst von der alten Selektion entfernt, sobald er mindestens einem weiteren Knoten erfolgreich sein Update gesendet hat.

Dadurch wird verhindert, dass die Information beziehungsweise der Fortschritt des Knoten verloren geht.

Da wir in der gesamten Simulation von einer eventuellen Wiederverbindung von Knoten ausgehen, sprich, dass die Knoten zu einem Zeitpunkt t miteinander kommunizieren können, auch wenn zwischendurch die Verbindung gestört war, kann durch dieses Update keine Information verloren gehen.

6.2 Andere Heuristiken

6.2.1 Heuristik mit Übergangsgraph und Koordinator

Für die dynamische Selektion wäre diese Heuristik in Betracht zu ziehen.

Die Idee dieser Heuristik, ist die komplette Neuselektion während der Laufzeit.

Diese Neuselektion wird jedoch nicht gleich übernommen, sondern wird der zu erreichende Teilgraph. Pro Ausführungsschritt wird nun versucht ein Knoten der alten Selektion mit einem Knoten der neuen Selektion auszutauschen.

Hierbei muss berücksichtigt werden, dass Knoten wenn sie ausgetauscht werden, nicht sofort weiter an der Ausführung des Workflows teilnehmen können. Dies hat zum Beispiel den Grund, da der neue Knoten noch kein Update von den anderen Knoten der Selektion bekommen hat.

Wie genau man nun vom alten zum neuen Graphen kommt wird an dieser Stelle nicht definiert.

6.2.2 Broadcast, alle die besser sind antworten und dann Abgabe an den besten

Die Broadcast-Heuristik ist die Überlegung einer komplett dezentralen Lösung.

Mit dieser Heuristik kommt man zu keiner Grundselektion, sie ist ausschließlich für die dynamische Ausführung und der Neuselektion gedacht.

Das Prinzip dieser Heuristik, ist es, dass Knoten die zu schlecht werden einen Broadcast senden. Die Berechnung der Qualität und des Schwellenwertes könnten ebenfalls dezentral auf dem Knoten stattfinden.

Falls also nun ein Knoten seine Qualität berechnet und merkt, dass es im Vergleich zu einem Ausgangswert nur noch beispielsweise eine Qualität von 80% hat, so sendet er einen Broadcast mit seiner derzeitigen Qualität.

Alle anderen Knoten die besser sind und den Broadcast erhalten, senden ihm eine Nachricht mit ihrer Qualität zurück.

Der Knoten der ausgewechselt werden will, kann nun anhand der Nachrichten auswählen, welcher Knoten am besten ist, und ihm das Replikat des Workflows schicken.

Der Rechenaufwand würde hier um ein Vielfaches reduziert werden. Auch der Message-Overhead wäre geringer wie bei einer zentralisierten Lösung.

7 Evaluation

Für die Evaluation wurde der Simulator The ONE [KOK09] verwendet.

Es wurden Simulation auf zwei Systemen ausgeführt um die in diesem Kapitel vorgestellten Messwerte zu erhalten.

Das erste System war das Folgende. Es ist ein unverändertes System, das es von einem bekannten Anbieter zu kaufen gibt.

- MacBook Air
- OS X Yosemite - Version 10.10.3
- Prozessor: 1,3 GHz Intel Core i5
- Speicher: 8 GB 1600 MHz DDR3

Das zweite System war ein unbekannter Desktop-PC. Unbekannt in dem Sinne, da er ein Eigenbau ist.

- Windows - Version 8.1 N
- Prozessor: AMD Phenom II X4 (3,4 GHz)
- Speicher: 24 GB 1600 MHz DDR3

Für die Evaluation wurden insgesamt mehr als 200 Simulationen durchgeführt.

Das Grundgerüst der Simulation ist eine Fläche von 1000m^2, auf der sich 50 mobile ausführbare Knoten, 25 Infrastrukturknoten und 100 Service-Knoten bewegen. Die Menge der Services die einer Aktivität zur Verfügung stehen, ist begrenzt auf maximal 10.

Für die Direktverbindung zwischen zwei Knoten wird ein Abstand von 10m gewählt.

Es wurder bei der Implementierung darauf geachtet, die Knoten des Simulators so wenig wie möglich zu verändern, um so nah wie möglich an der gegebenen Abstrahierung der realen Welt von The ONE festzuhalten.

Auch die Cluster werden anhand dieser Distanz ausgewählt. Es gibt insgesamt 5 Cluster. Ungefähr 25% der Knoten sind während der Simulation in einem Cluster. Die Abschwächung der Verbindungen eines Knoten innerhalb eines Cluster nach draußen, beträgt den Faktor 0,2.

Die Simulierte Abkühlung wurde ohne Zeitparameter implementiert. Für die Starttemperatur wurde 1, für die Endtemperatur wurde 0.001 festgelegt. Die Abkühlung pro Zeitschritt beträgt 0.001.

Alle Schaubilder, die in diesem Kapitel der Evaluation dienen, wurden mit gnuplot [WK+10] erstellt.

Wenn im Folgenden von allen Heuristiken gesprochen wird, so ist die Zufällige Selektion, die Simulierte Abkühlung, und alle eigenen Heuristiken gemeint. Die optimale Lösung gehört nicht zu diesem Begriff, da sie keine Heuristik darstellt, sondern eine systematischer Vergleich jeder validen Möglichkeit ist.

Vorweg ist zu sagen, dass einige Messreihen die durchgeführt wurden, nicht in die Evaluation mit aufgenommen werden konnte, da die optimale Lösung zu lange gedauert hat um diese Messreihe zu beenden. Wie unpraktikabel die optimale Lösung ist, zeigte sich schon bei der ersten Messung die durchgeführt wurde.

Für alle Evaluations-Messungen gibt die Variable k die Größe der Selektion und die Variable w die Größe des Workflows an. Falls in den Evaluations-Messungen das Wort Replanning benutzt wird, so ist die Neuplanung der Selektion gemeint. Wenn in diesem Kapitel von einer Messung die Rede ist, so ist eine Vielzahl von Simulationen mit gleichbleibenden Parametern gemeint.

7.1 Die erste Messung ($1 \leq k \leq 5$ und $w = 10$)

Die erste Messung die durchgeführt wurde, war eine Messung mit allen Heuristiken inklusive der optimalen Lösung.

Für diese Messung wurde ein Workflow der Größe 10 definiert. $w = 10$

7.1.1 Theoretische Qualität

Als erstes wurde die theoretische Qualität der Selektion verglichen.

Wie man in Abbildung 7.1 sehen kann liegen einige der theoretischen Qualitäten sehr nah beieinander beziehungsweise aufeinander. Hier bekommt man allerdings eine Vorstellung davon, in welchem Bereich sich eine Heuristik befindet und wie gut sie im Vergleich zur optimalen Lösung ist. Um einen Besseren Überblick über die Ergebnisse zu bekommen schauen wir uns die Lösungen in Abbildung 7.2 und Abbildung 7.3 separat an. Die Einheiten der x- und y-Achse sind dieselben wie in Abbildung 7.1.

Wie man sehen kann, sind die optimale Lösung, Simulated Annealing und die Heuristik Knoten-Service in etwa identisch. Die Heuristik Knoten-Service-n-Kanten hat für eine Selektion der Größe fünf einen leichte Verschlechterung der theoretischen Qualität.

Die Verschlechterung der Werte bei höherem k geben vorerst zu denken. Wenn man sich die Berechnung der Metrik der theoretischen Qualität jedoch nochmals anschaut ist dies ganz logisch. Im Bezug auf diese Metrik wird eine Selektion nicht zwangsläufig besser, je mehr Knoten man hinzufügt. Die Knoten einer Selektion müssen um eine Aktivität erfolgreich auszuführen miteinander kommunizieren. Je mehr Knoten in einer Selektion vorhanden sind, umso mehr muss kommuniziert werden. So sind die Verschlechterungen der Werte einiger Heuristiken für eine größere Selektion zu erklären.

Auf die Werte der Cluster-Heuristik wollen wir an dieser Stelle dennoch kurz genauer eingehen. Hier ist interessant zu sehen, dass eine Selektion der Größe 2 und 3 die besten Werte liefert. Wie man zudem erkennen kann, ist die Standardabweichung für diese k am größten. Das lässt daran erklären, dass für eine Selektion der Größe 2 und 3 wohl ein Cluster gefunden wurde. Für $k = 4$

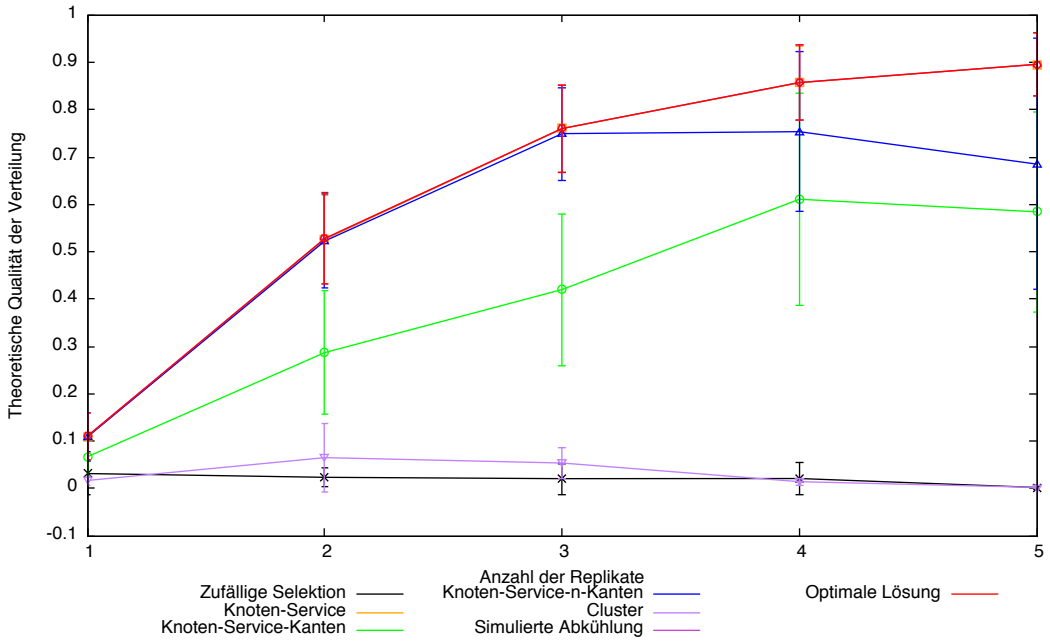

Abbildung 7.1: Messung der theoretischen Qualität für $1 \leq k \leq 5$ und $w = 10$

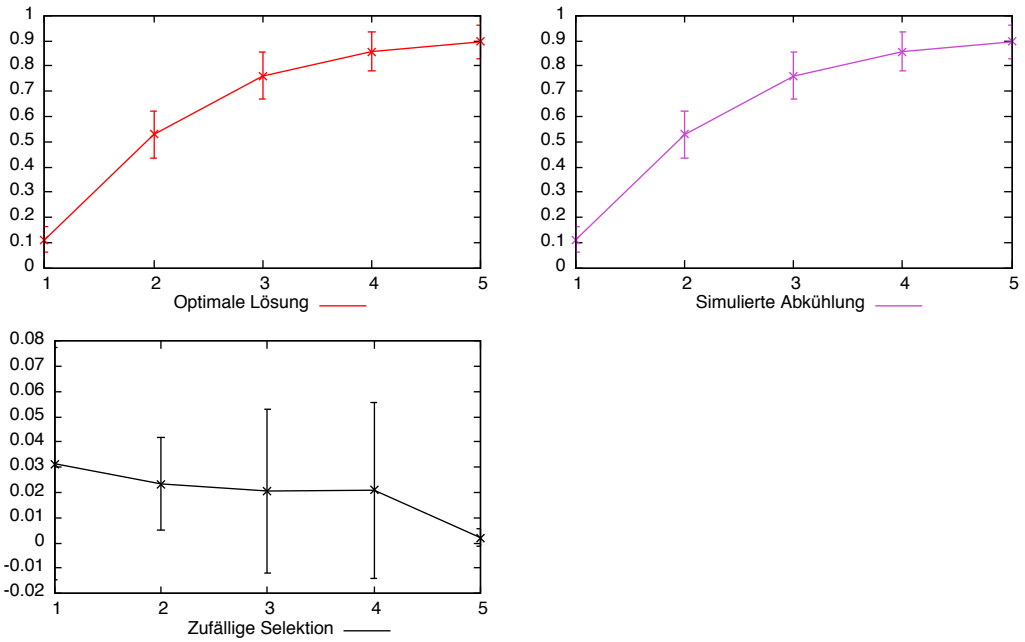

Abbildung 7.2: Messung der theoretischen Qualität für $1 \leq k \leq 5$ und $w = 10$

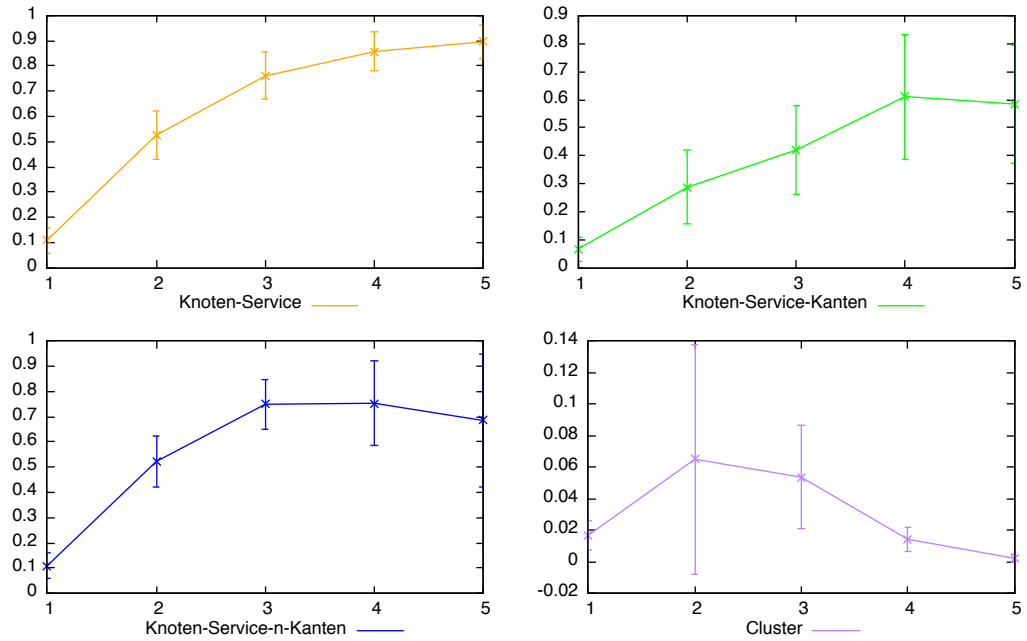

Abbildung 7.3: Messung der theoretischen Qualität für $1 \leq k \leq 5$ und $w = 10$

wurde wahrscheinlich kein Cluster mehr gefunden und somit nimmt die theoretische Qualität wieder stark ab.

Der erste Vergleich hat gezeigt, dass die Heuristiken teilweise ähnlich gute Werte wie die optimale Lösung erzeugen, zumindest im Bezug auf die theoretische Qualität der Selektion.

7.1.2 Berechnungszeit der Selektion

Wir betrachten nun die gemessenen Werte für die Berechnungszeit der Selektion.

Das wie angenommen die optimale Lösung sehr rechenaufwändig sein wird ist, in Abbildung 7.4, auf den ersten Blick zu erkennen. Eine Steigerung der Berechnungszeit von 25000 Millisekunden bei $k = 4$ auf 400000 Millisekunden bei $k = 5$ ist enorm. Umgerechnet benötigt die optimale Lösung 25 Sekunden für die Berechnung einer Selektion der Größe 4 und und knappe 7 Minuten für eine Selektion der Größe 5. Wenn man diesen Steigerungsfaktor nimmt und für $k = 6$ anwendet, so kommt man auf eine Berechnungszeit von fast zwei Stunden. Wie angenommen ist die optimale Lösung für einen Graphen mit 75 ausführbaren Knoten, einem Workflow der Größe 10 ab einer Selektion der Größe 4 nicht mehr praktikabel. Aufgrund dieser Erkenntnis, wird die optimale Lösung für alle weiteren Messungen nicht mit einbezogen.

Die Werte der anderen Heuristiken werden in Abbildung 7.5 ohne die optimale Lösung nochmals dargestellt. Wir betrachten nun die gemessenen Werte für die Berechnungszeit der Selektion. Hier

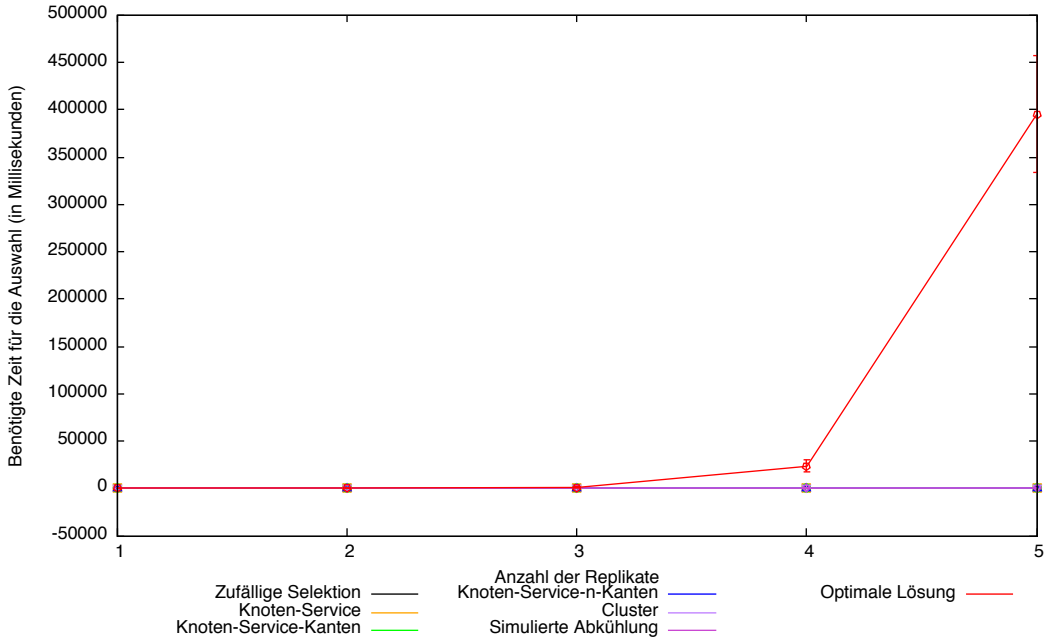

Abbildung 7.4: Berechnungszeit der Selektion für $1 \leq k \leq 5$ und $w = 10$

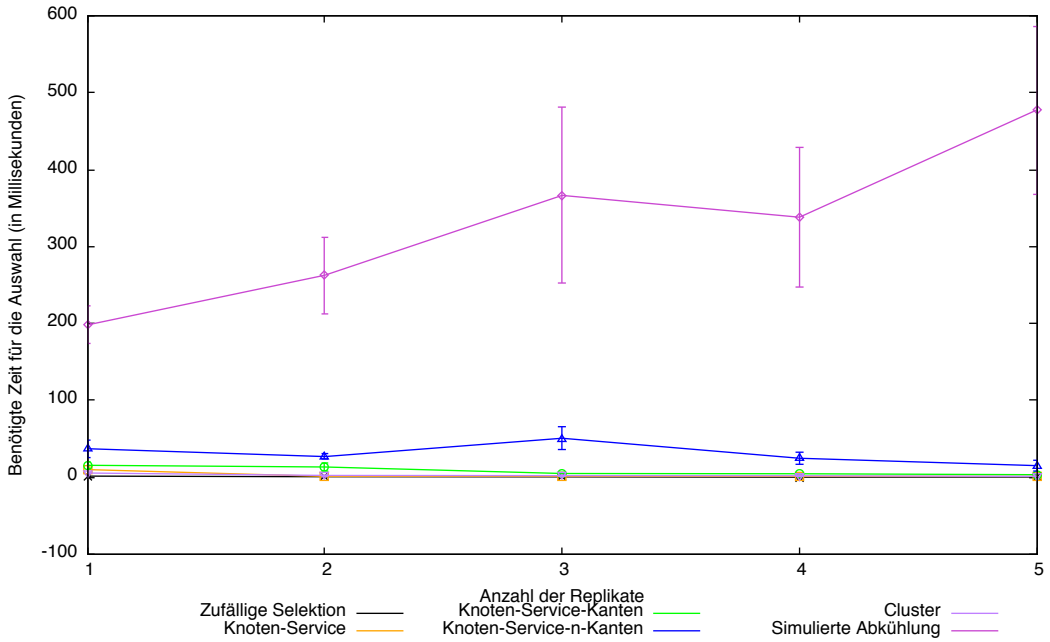

Abbildung 7.5: Berechnungszeit der Selektion für $1 \leq k \leq 5$ und $w = 10$ (ohne optimale Lösung)

ist zu erkennen, dass auch die Simulierte Abkühlung deutlich länger braucht als die in dieser Diplomarbeit entwickelten Heuristiken. Es ist jedoch besser ersichtlich in welchem Bereich sich die Berechnungszeiten befinden.

In Abbildung 7.6 werden die eigenen Heuristiken separiert betrachtet. Die Einheiten der x- und y-Achse sind dieselben wie in Abbildung 7.5. Es fällt auf, dass die Berechnungszeit für größere Selektionen, niedirger sind als für kleinere. Dies war so nicht zu erwarten. Eine mögliche Erklärung hierfür könnte das System sein, auf dem die Simulation durchgeführt wird, oder die Ausführungsumgebung von Java selbst. Auffällig ist hier, dass die Anfangswerte immer am höchsten sind. Im Laufe der Evaluation wird dies weiter beobachtet.

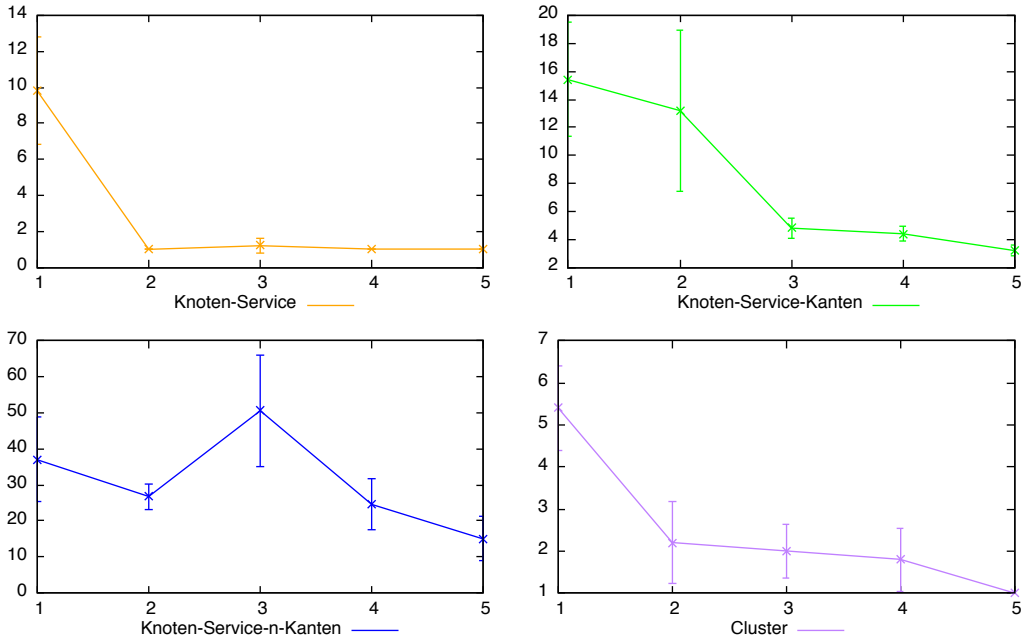

Abbildung 7.6: Berechnungszeit der Selektion für $1 \leq k \leq 5$ und $w = 10$ (für die eigenen Heuristiken)

7.1.3 Schritte der statischen Ausführungen

Die nächste Metrik die betrachtet worden ist, ist die der statischen Ausführungszeit. Die statische Ausführung ist wie in 3.4 beschrieben, eine Ausführung die vor der eigentlichen Laufzeit der Simulation stattfindet. Die Knoten bewegen sich also nicht.

mit starker Konsistenz

Die benötigten Schritte dieser Ausführung für die optimale Lösung und allen Heuristiken, kann man in Abbildung 7.7 sehen.

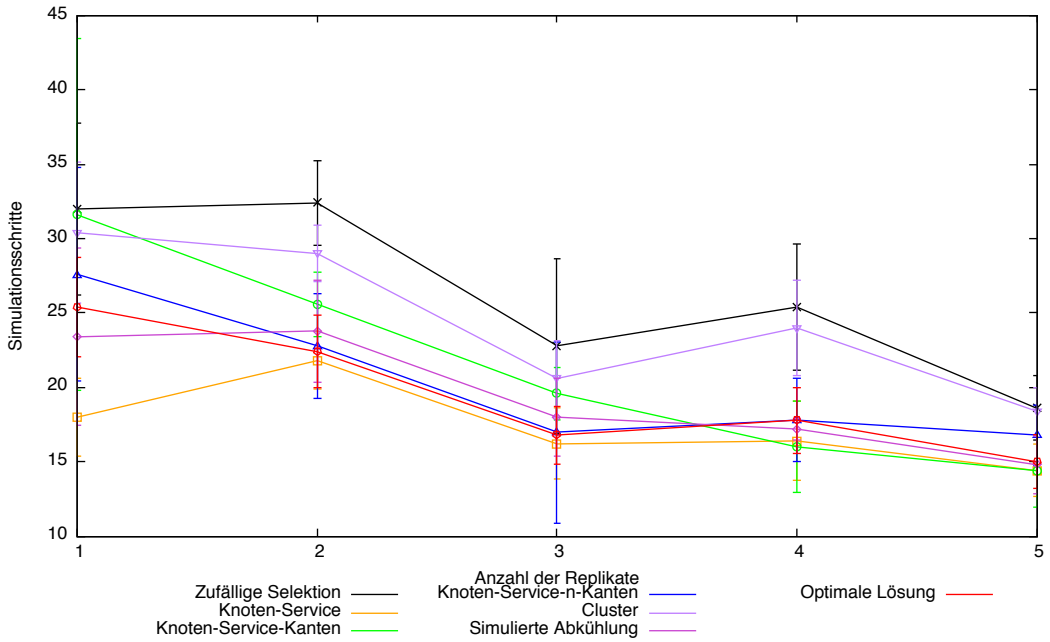

Abbildung 7.7: Schritte der statischen Ausführung für $w = 10$ mit starker Konsistenz

Hier ist ganz wichtig zu erwähnen, dass der Workflow 10 Aktivitäten umfasste.

Man kann erkennen, dass die Standardabweichung bei allen Heuristiken und auch der optimalen Lösung sehr groß ist. Die Streuung der Ausführungsschritte für eine Workflow der Größe 10 ist zu groß um definitiv sagen zu können, welche Heuristik hier der optimalen Lösung am nächsten kommt. Wie man sieht hat selbst die optimale Lösung eine höhere Anzahl an Schritten gebraucht, wie einige Heuristiken. Da es sich um Wahrscheinlichkeiten handelt ist so ein Ergebnis zwar möglich, aber für eine Evaluierung nicht als Beweis gültig.

mit schwacher Konsistenz

Die Ausführung unter schwacher Konsistenz ist geringfügig besser als die vorherige.

Ein großer Unterschied ist allerdings nicht zu erkennen. Die Vermutung liegt nahe, dass es einen deutlich größeren Workflow benötigt, um hier eine genaue Aussage treffen zu können.

7.1.4 Schritte der dynamischen Ausführungen

Bei der dynamischen Ausführung bewegen sich die Knoten und dadurch verändern sich ihre Ausfallsicherheiten anderen Knoten gegenüber. Außerdem wird die Neuplanung der Selektion zur Laufzeit ausgeführt.

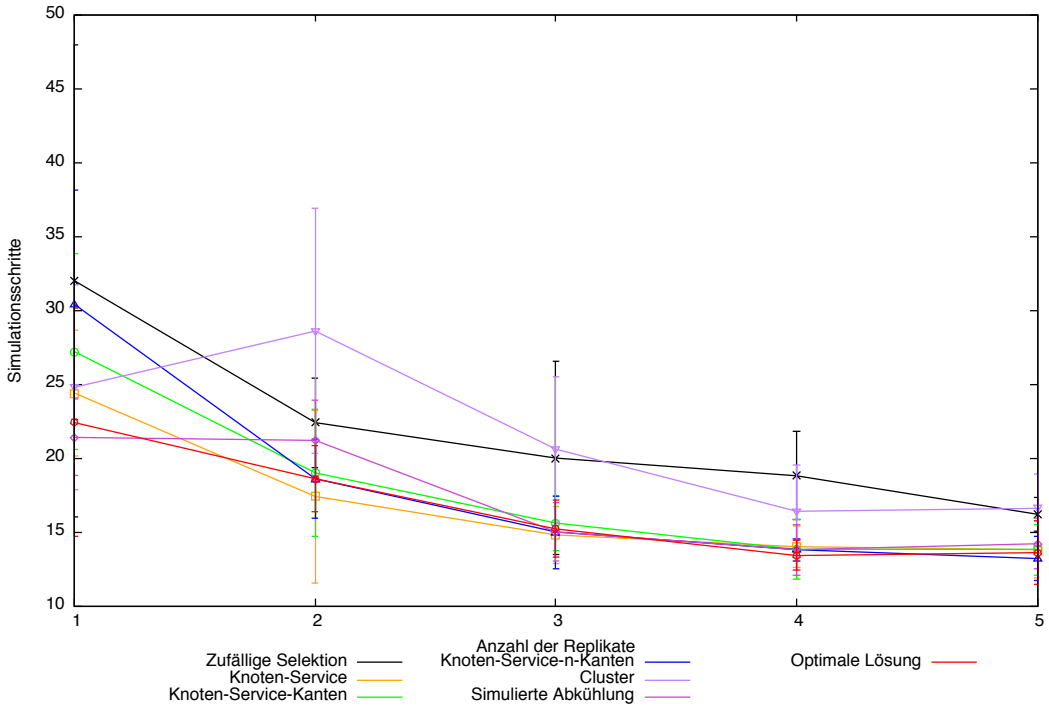

Abbildung 7.8: Schritte der statischen Ausführung für $w = 10$ mit schwacher Konsistenz

mit starker Konsistenz

mit schwacher Konsistenz

Auch bei der dynamischen Ausführung ist kein sichtbarer Unterschied zu erkennen, für einen Workflow der Größe 10.

7.1.5 Neuplanungen der Selektion

Bei dieser ersten Messung war es nicht möglich über die Anzahl der Knotenwechsel während der Laufzeit eine Aussage treffen zu können. Die Selektion war zu klein um viele Knoten wechseln zu können, zudem waren die Ausführungszeiten beziehungsweise die Schritte, die für die Ausführung benötigt wurde, zu gering. Viele Zählungen ergaben 0. Hierfür gibt es zwei mögliche Gründe. Zum einen, dass die Ausführung beendet war, bevor die Selektion eine theoretische Qualität von 80% ihrer Ausgangsqualität hatte und zum anderen, dass die theoretische Qualität während der Laufzeit besser geworden ist.

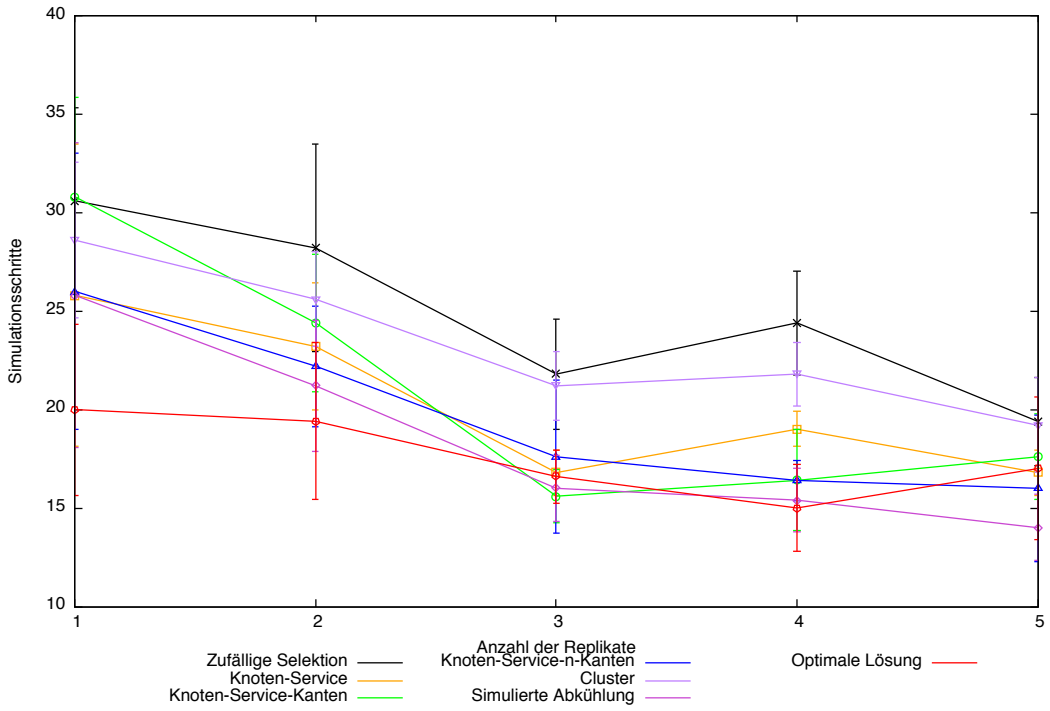

Abbildung 7.9: Schritte der dynamischen Ausführung für $w = 10$ mit starker Konsistenz

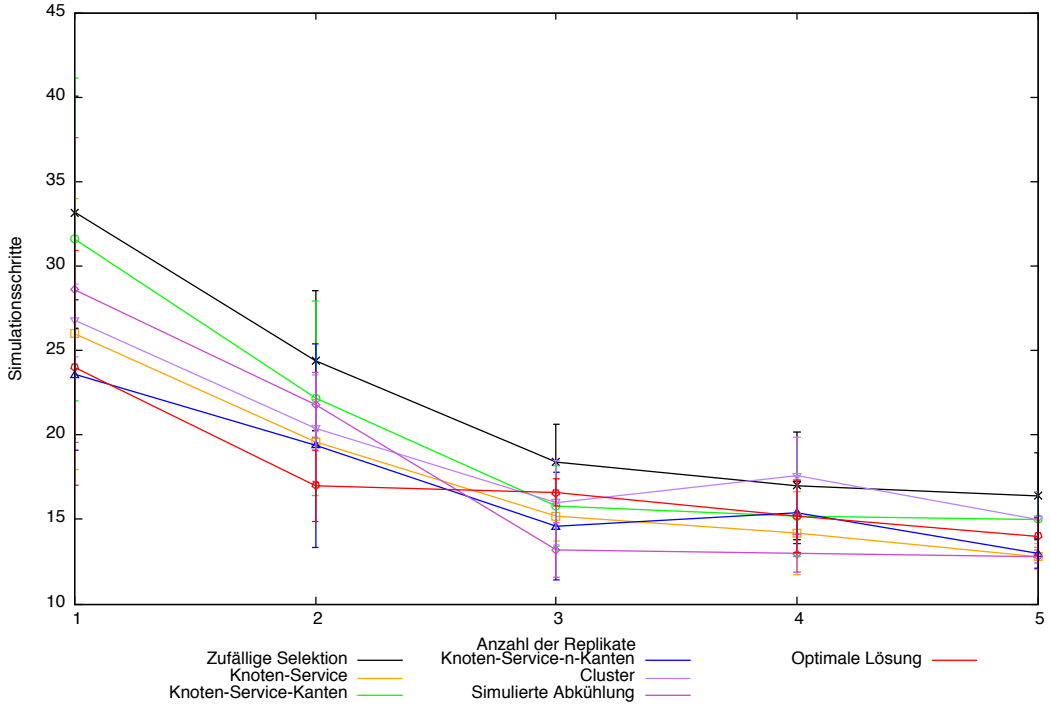

Abbildung 7.10: Schritte der dynamischen Ausführung für $w = 10$ mit schwacher Konsistenz

7.2 Messung mit $1 \leq k \leq 10$ und $w = 50$

Für die zweite Messung wurde die Größe des Workflows auf 50 erhöht. Des weiteren wurde die Selektion auf bis zu 10 erhöht.

7.2.1 Theoretische Qualität

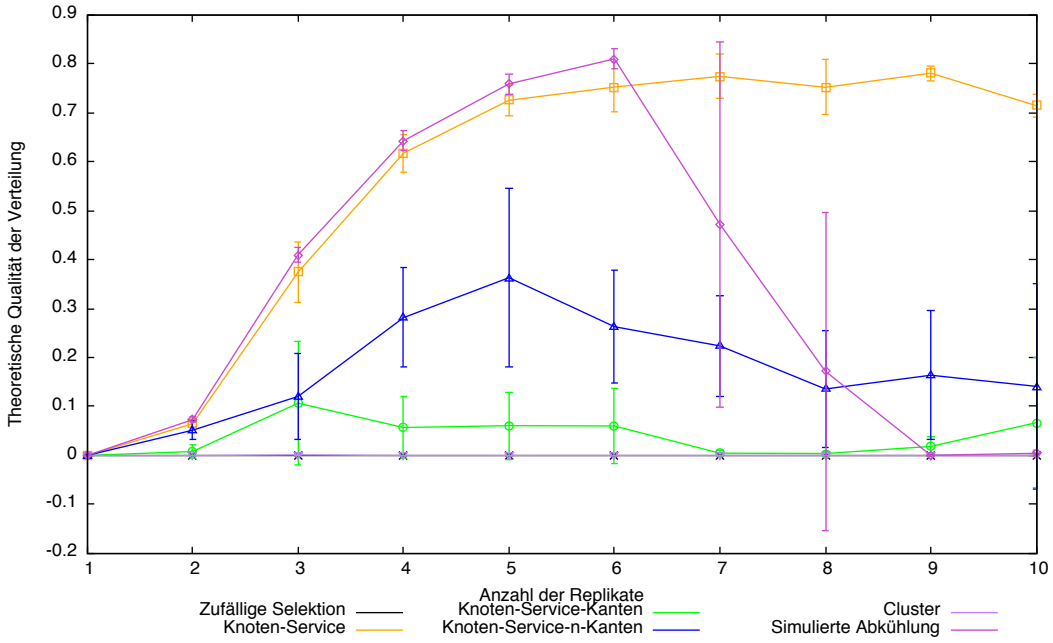

Abbildung 7.11: Messung der theoretischen Qualität für $1 \leq k \leq 10$ und $w = 50$

Bei der Darstellung der theoretischen Qualität fällt auf, dass die Simulierte Abkühlung ab einer Selektion der Größe 7, keine guten Werte mehr erzielt. Dies könnte daran liegen, dass es mehr mögliche Selektionen gibt als die Simulierte Abkühlung Schritte vornimmt. Gegebenenfalls müsste hier die Starttemperatur der Simulierten Abkühlung erhöht werden. Weitere Möglichkeiten wären, die Endtemperatur zu senken, oder die Abkühlungsrate zu verkleinern.

Da in dieser Diplomarbeit nicht evaluiert werden soll, wie gut die Simulierten Abkühlung geeignet ist um eine Selektion nahe der optimalen Lösung zu finden, werden die Parameter vorerst nicht verändert.

7.2.2 Berechnungszeit der Selektion

Die gemessenen Werte für die Berechnungszeit der Selektion sind in Abbildung 7.12 zu sehen.

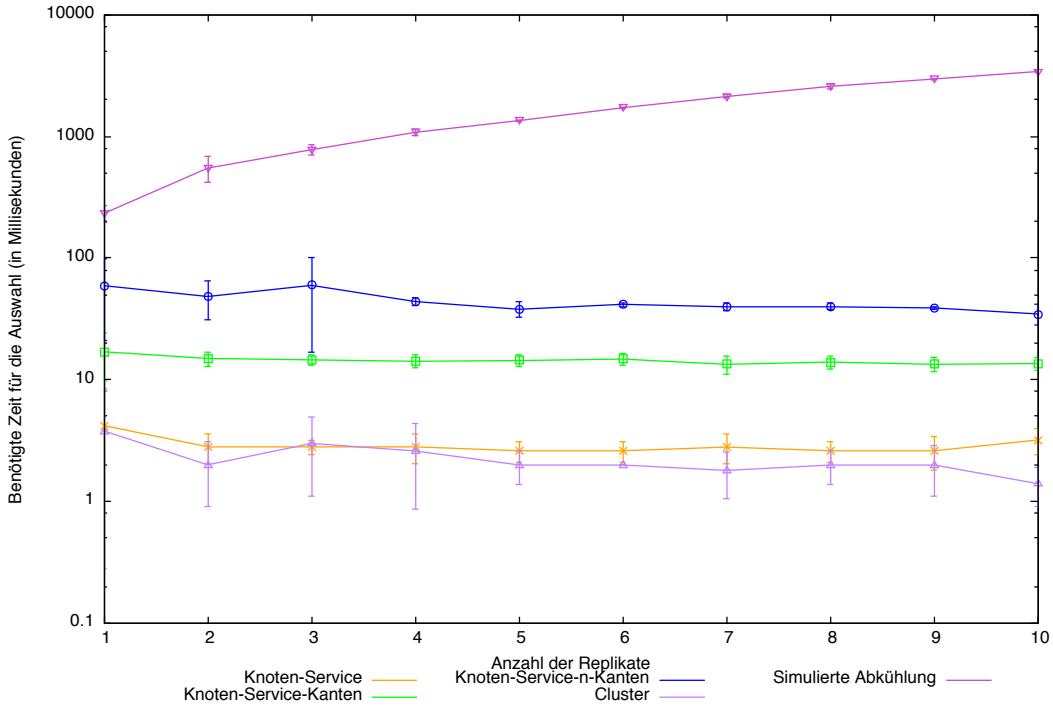

Abbildung 7.12: Berechnungszeit der Selektion für $1 \leq k \leq 10$ und $w = 50$

In dieser Abbildung ist relativ gut zu erkennen, wie schnell die eigenen Heuristiken sind. Vergleich man diese Abbildung mit Abbildung 7.11, so sieht man, dass die Heuristik Knoten-Service eine sehr gute theoretische Qualität erzielt und das in einer sehr geringen Zeit.

Die Simulierte Abkühlung hingegen braucht schon jetzt sehr viel Zeit. Bearbeitet man nun auch noch ihre Parameter, damit sie eine bessere theoretische Qualität erzielt, so wird die sowieso schon hohe Berechnungsdauer noch größer.

Es ist also abzusehen, dass die Simulierte Abkühlung für große Graphen mit einem Workflow größer 100, nicht mehr praktikabel ist.

7.2.3 Schritte der statischen Ausführungen

mit starker Konsistenz

mit schwacher Konsistenz

Bis zu dieser Stelle scheint es so, dass die Heuristik Knoten-Service die wenigstens Ausführungsschritte benötigt. Dies gilt allerdings nur für die Ausführung mit schwacher Konsistenz. Für die Ausführung mit starker Konsistenz sind die Werte immer noch zu sprunghaft.

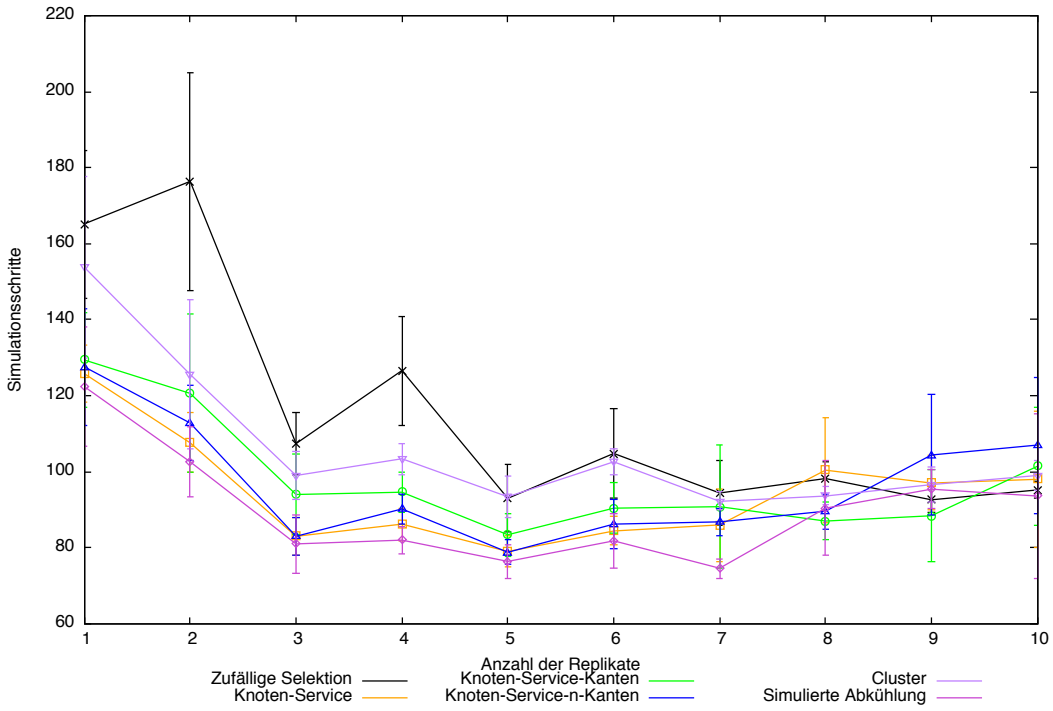

Abbildung 7.13: Schritte der statischen Ausführung für $w = 50$ mit starker Konsistenz

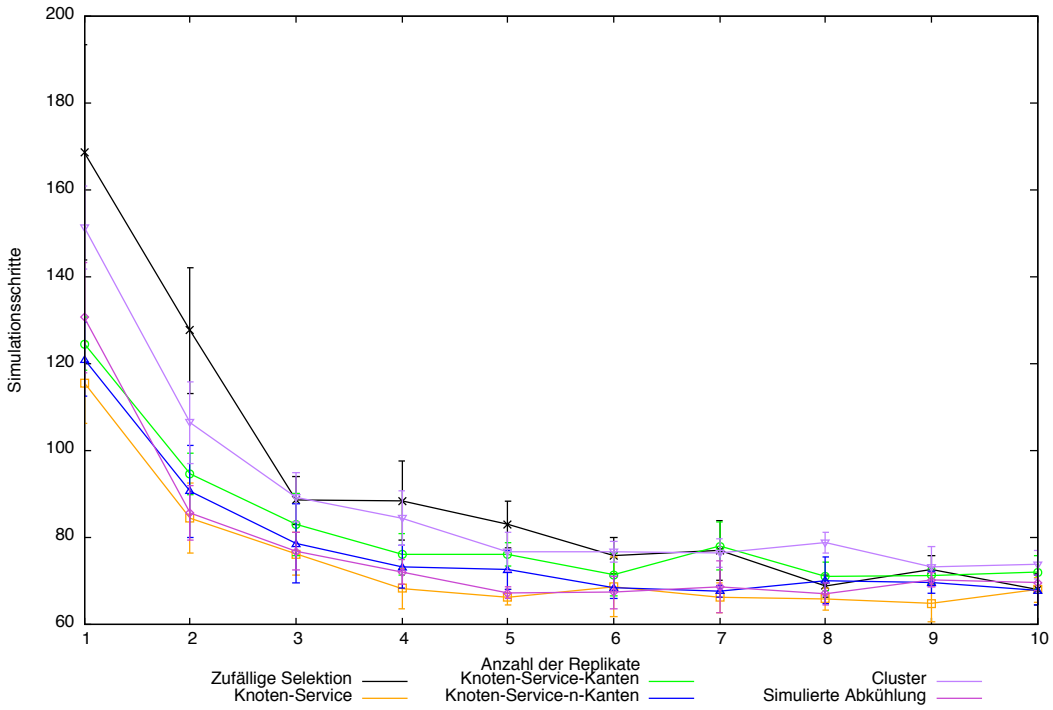

Abbildung 7.14: Schritte der statischen Ausführung für $w = 50$ mit schwacher Konsistenz

Es ist auch zu sehen, dass allgemein die Ausführung mit schwacher Konsistenz sichtbar schneller ist, als die Ausführung mit starker Konsistenz.

7.2.4 Schritte der dynamischen Ausführungen

mit starker Konsistenz

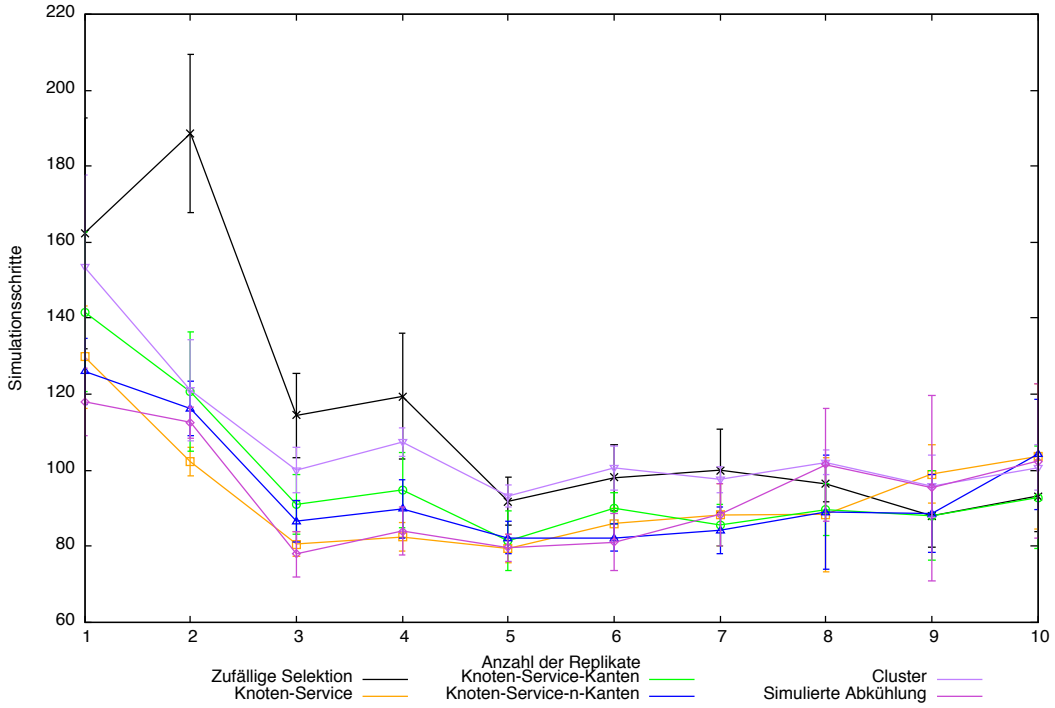

Abbildung 7.15: Schritte der dynamischen Ausführung für $w = 50$ mit starker Konsistenz

mit schwacher Konsistenz

Auch bei der dynamischen Ausführung scheint die Heuristic Knoten-Service eins der besten Resultate, für die Ausführung mit schwacher Konsistenz, zu liefern. Die Cluster-Heuristik scheint, zu diesem Zeitpunkt, die schlechteste der eigenen Heuristik zu sein. Zu Beginn der tieferen Recherche, wurde diese Heuristik als sehr vielversprechend angesehen.

7.2.5 Neuplanungen der Selektion

Die Neuplanung der Selektion, für die Ausführung eines Workflows der Größe 50, sieht wie folgt aus. Es wurde lediglich die größte Selektion für das Replanning herangezogen, da kleinere Selektionen

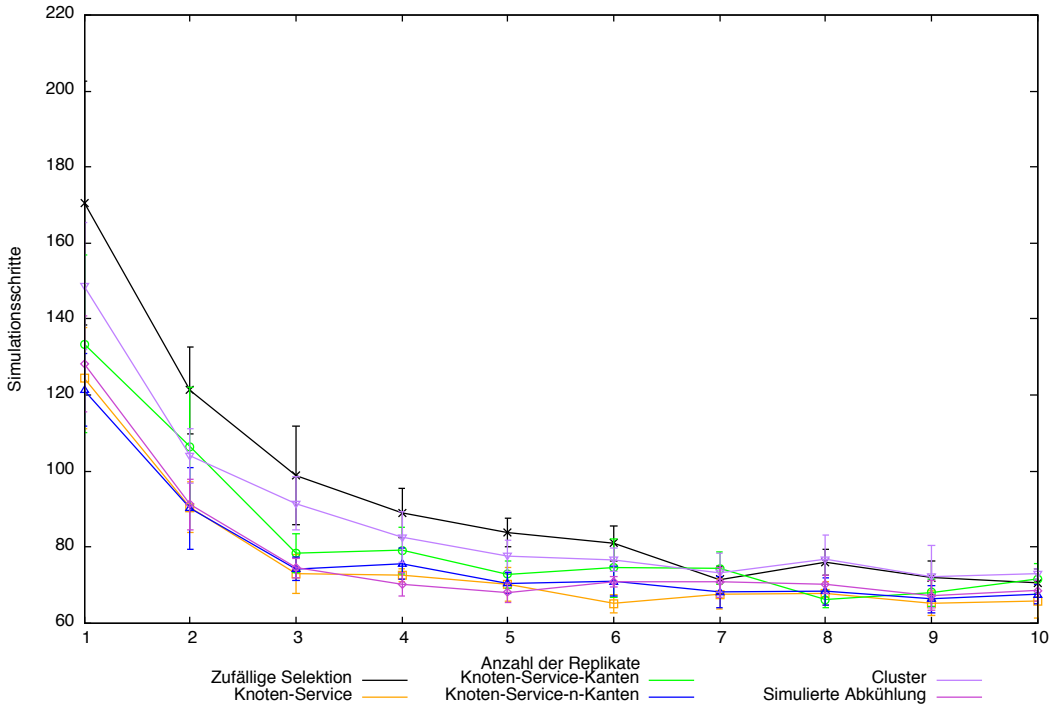

Abbildung 7.16: Schritte der dynamischen Ausführung für $w = 50$ mit schwacher Konsistenz

Heuristike	Anzahl
Knoten-Service	45
Knoten-Service-Kanten	14
Knoten-Service-n-Kanten	22
Cluster	2

Tabelle 7.1: Replanning für $k = 10$ und $w = 50$

durch die mangelnde Anzahl an Knoten nicht aussagekräftig sind. Außerdem wurden nur eigene Heuristiken betrachtet.

Beim Replanning zeigt sich, dass die Cluster Heuristik während der Laufzeit kaum an theoretischer Qualität verliert, da bevorzugt Knoten gewählt werden, die sich in einem Cluster befinden. Diese Clusterknoten bleiben während der gesamten Laufzeit in ein und demselben Cluster.

7.3 Messung mit $3 \leq k \leq 15$ und $w = 100$

Für die dritte Messung wurde die Größe des Workflows auf 100 erhöht. Des weiteren wurde die Selektion auf bis zu 15 erhöht. Die kleinste Selektion dieser Messung hatte die Größe 3.

7.3.1 Theoretische Qualität

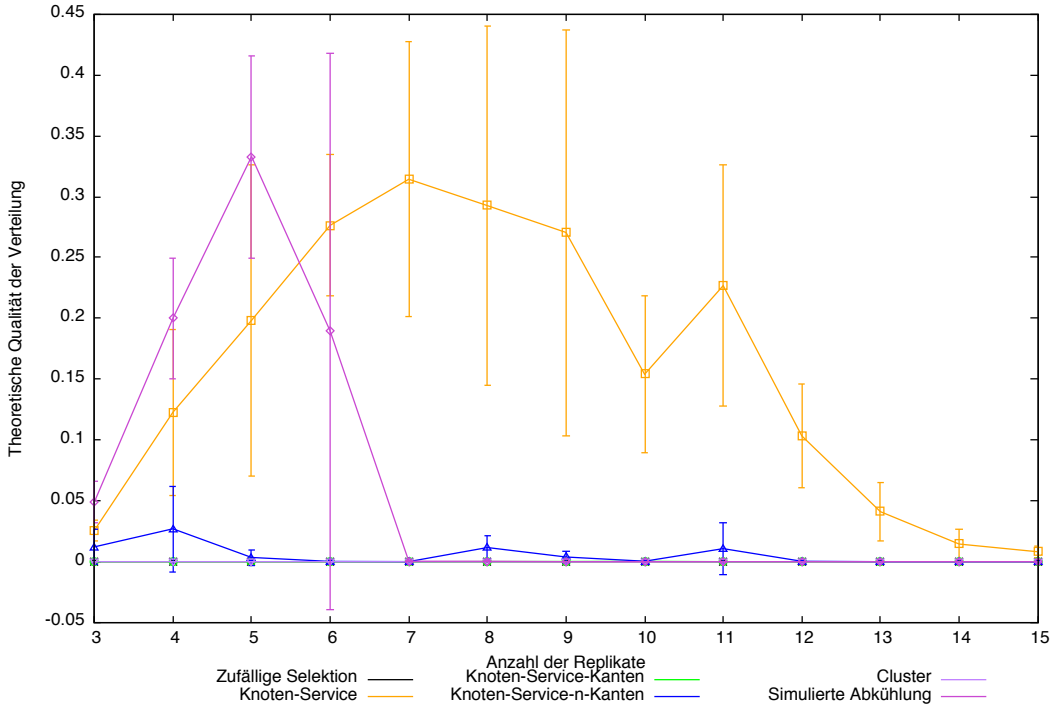

Abbildung 7.17: Messung der theoretischen Qualität für $1 \leq k \leq 15$ und $w = 100$

Wie bei der vorherigen Messung fällt die Simulierte Abkühlung ab einer gewissen Größe der Selektion stark ab. Wie erwartet gibt es deutlich mehr mögliche Selektionen, als diese Heuristik Optimierungsschritte durchführt. Es wurde absichtlich Anpassung vorgenommen, da sonst die Berechnungsdauer deutlich länger werden würde. So kann man zudem die eigenen Heuristiken mit einer Simulierten Abkühlung, die sehr schnell Werte liefert, vergleichen.

Des weiteren ist zu sehen, dass die Heuristik Knoten-Service mit Abstand die beste theoretische Qualität erreicht. Der Einbruch bei $k = 10$ ist so zu erklären, dass ein Knoten anhand eines guten Services gewählt wurde, der offensichtlich eine schlechte Netzanbindung hatte.

Der Abfall aller Kurven der theoretischen Qualität für größer werdenen k, ist dadurch zu erklären, dass die Metrik in Kapitel 3 – Das Systemmodell beschrieben, anhand der Ausfallsicherheiten berechnet wird. Diese Ausfallsicherheiten sind Wahrscheinlichkeiten und daher beruht die Berechnung auf eine Multiplikation. Eine Multiplikation von Zahlen zwischen 0 und 1 wird immer gegen 0 gehen.

7.3.2 Berechnungszeit der Selektion

In Abbildung 7.18 ist die Berechnungszeit der Heuristiken sehr gut erkennbar. Hier benötigt die Cluster-Heuristik die kürzeste Zeit. Die Heuristik Knoten-Service-n-Kanten benötigt am längsten um eine Selektion zu finden. Dennoch sind alle Zeiten unter 100 Millisekunden, was praktikabel für die Berechnung während der Laufzeit ist.

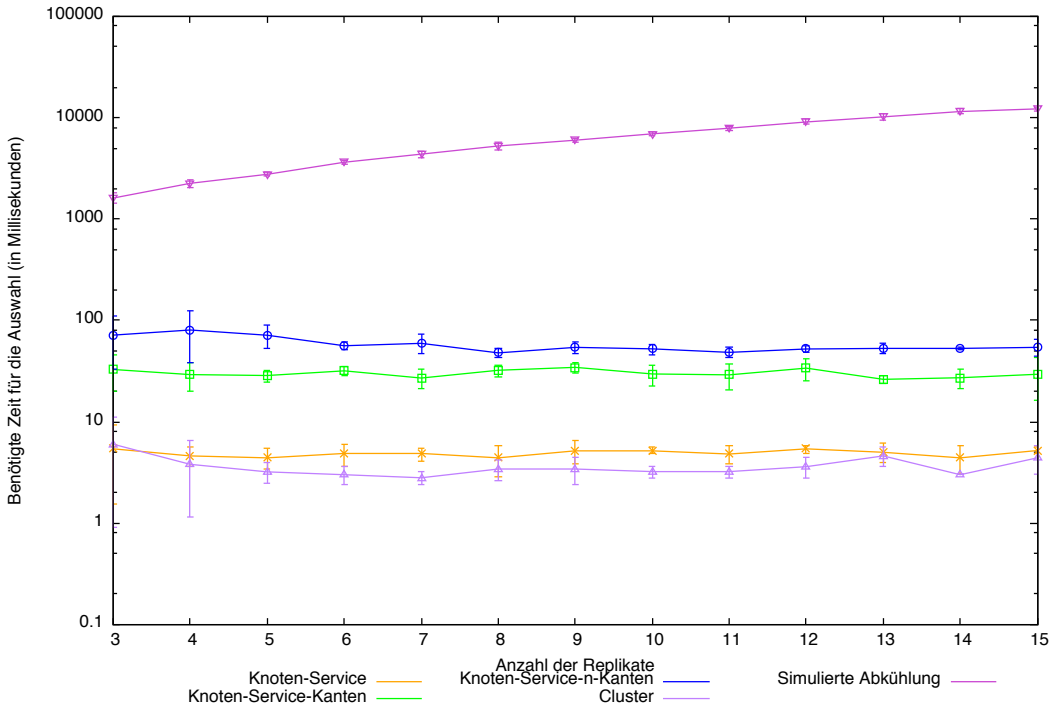

Abbildung 7.18: Berechnungszeit der Selektion für $1 \leq k \leq 15$ und $w = 100$

7.3.3 Schritte der statischen Ausführungen

Für die Ausführungen ist die Größe des Workflows immer noch zu gering, womöglich sieht man hier erst eindeutig welche Heuristik besser ist, wenn der Workflow eine Größe von 10000 hat.

Aber man sieht trotzdem einen Trend für die Ausführung mit hoher Konsistenz und für die mit niedriger Konsistenz.

mit starker Konsistenz

Für die Ausführung mit starker Konsistenz, liefert die Cluster-Heuristk durchgängig gute Werte. Auch in den vorangegangenen Messungen, hat man dasselbe beobachten können.

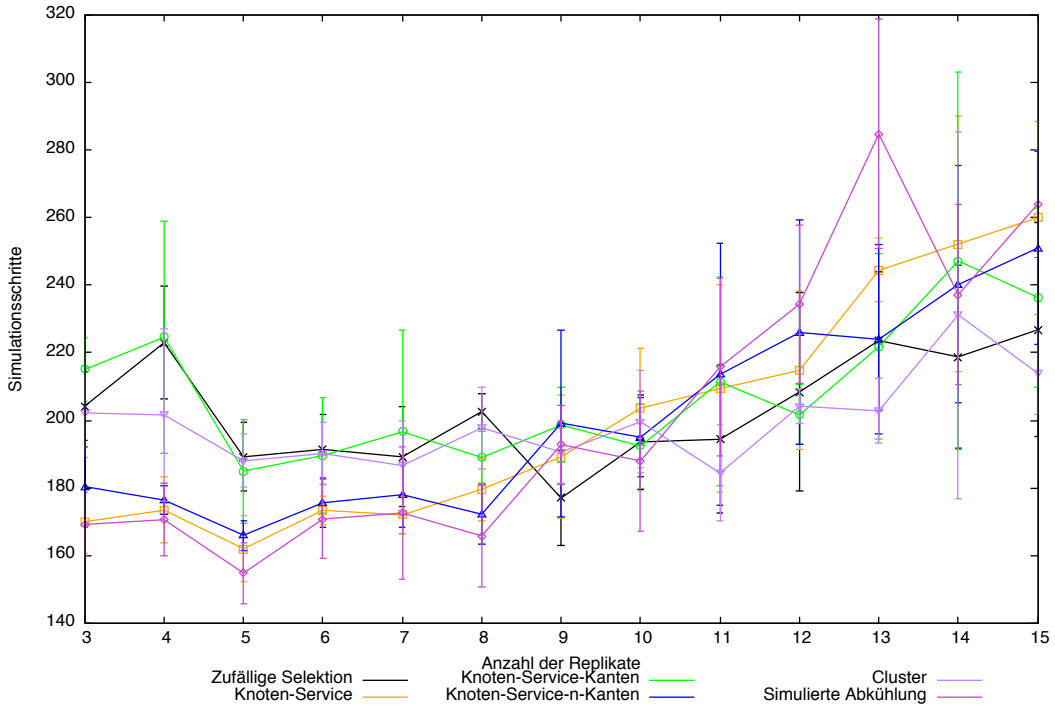

Abbildung 7.19: Schritte der statischen Ausführung für $w = 100$ mit starker Konsistenz

mit schwacher Konsistenz

Für die Ausführung mit schwacher Konsistenz, ist die Knoten-Service Heuristik im Durchschnitt die Beste. Bei allen drei Messungen lieferte sie durchgängig gute Resultate für die Ausführung mit schwacher Konsistenz.

7.3.4 Schritte der dynamischen Ausführungen

Zur dynamischen Ausführung kann gesagt werden, dass das gleiche zu beobachten ist wie für die statische Ausführung. Die dynamische Ausführung ist zudem im Mittel schneller als die statische Ausführung. Diese Beobachtungen beziehen sich jeweils auf den Vergleich der Ausführungen mit starker und schwacher Konsistenz.

mit starker Konsistenz

Die gleichen Beobachtungen wie bei der statischen Ausführung mit starker Konsistenz sind hier auch zu sehen. Die Cluster-Heuristik liefert die besten Werte im Bezug auf Ausführungen mit starker Konsistenz.

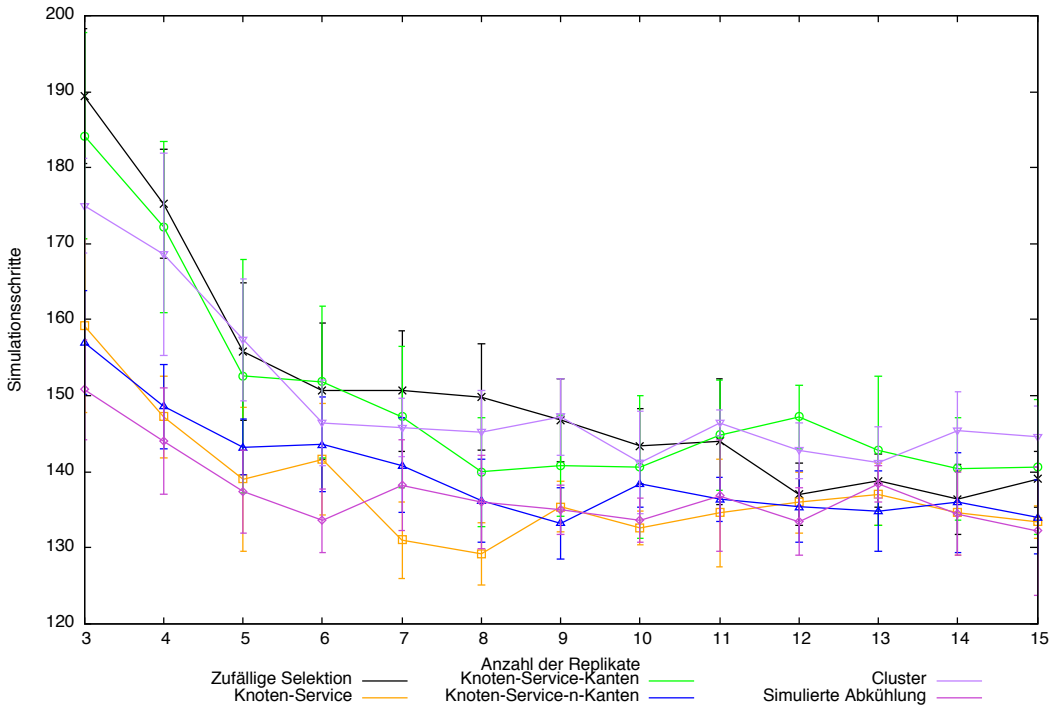

Abbildung 7.20: Schritte der statischen Ausführung für $w = 100$ mit schwacher Konsistenz

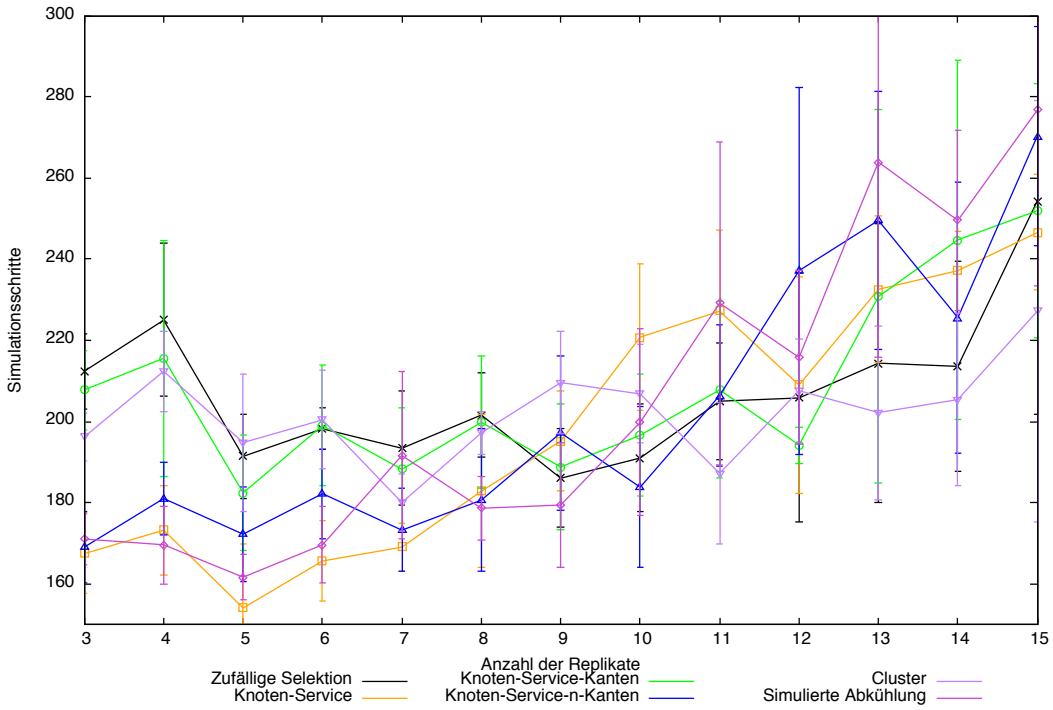

Abbildung 7.21: Schritte der dynamischen Ausführung für $w = 100$ mit starker Konsistenz

mit schwacher Konsistenz

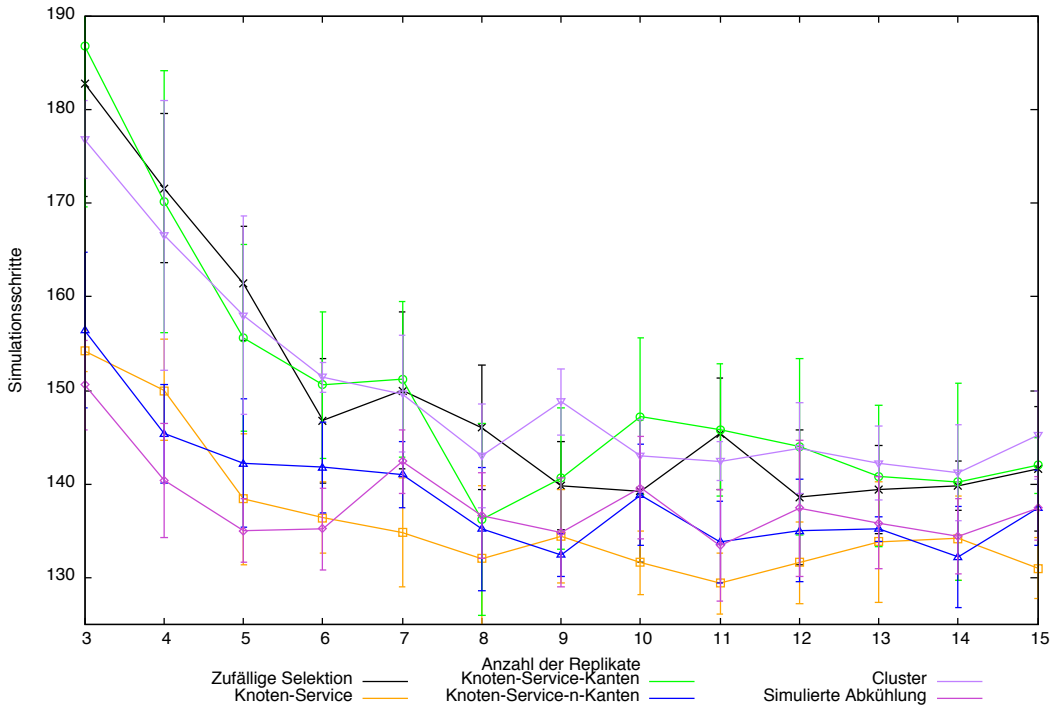

Abbildung 7.22: Schritte der dynamischen Ausführung für $w = 100$ mit schwacher Konsistenz

7.3.5 Neuplanungen der Selektion

Die Anzahl der Neuplanungen bestätigt teilweise die Beobachtungen aus der vorherigen Messung. Hier muss zusätzlich erwähnt werden, dass eine Heuristik, die eine schlechte Ausgangsselektion berechnet hat, weniger Replanning betreiben muss, als eine Heuristik die eine gute Selektion zu Beginn der Ausführung getroffen hat. Das kommt daher, dass ein schlechter Ausgangswert leichter gehalten werden kann, als ein guter.

Heuristike	Anzahl
Knoten-Service	79
Knoten-Service-Kanten	108
Knoten-Service-n-Kanten	34
Cluster	17

Tabelle 7.2: Replanning für $k = 15$ und $w = 100$

8 Verwandte Arbeiten

Es existieren bereits verwandte Arbeiten auf dem Gebiet der optimierten Platzierung von Workflow-Replikaten. Diese Arbeiten werden in diesem Kapitel vorgestellt.

8.1 Increasing Availability of Workflows Executing in a Pervasive Environment

[SBTR14] beschäftigt sich mit der Verfügbarkeit, der Ausführung eines Workflows in großen verteilten Systemen sowie pervasiven Umgebungen. Da diese Umgebungen eine starke Heterogenität und eine hohe Dynamik aufweisen, sind sie anfällig für Kommunikations- und Hardwarefehler. Dies stellt neue Anforderungen an die Ausführung von Workflows. Um die Verfügbarkeit zu erhöhen werden mehrere Replikate des Workflows auf verschiedene Knoten simultan ausgeführt. Hierfür wurden Techniken entwickelt, wodurch verschieden strukturierte Workflowkopien mit identischer Funktionalität generiert werden. Durch einen speziell entwickelten Algorithmus wird die simultane Ausführung der Workflows auf den Kopien koordiniert. Die Kopien werden dann wiederum durch eine Metrik evaluiert, welche die Menge an Kopien mit der höchsten Verfügbarkeit während der Ausführung findet. Die Metrik wird “availability metric“ genannt und analysiert den Grad der Verfügbarkeit während der simultanen Ausführung mehrere Workflow-Instanzen. Die Metrik beurteilt Lösungen im Bezug auf drei Anforderungen:

- der Zeitabstand zwischen zwei Ausführungen einer Aktivität in verschiedenen Kopien sollte so groß wie möglich sein
- alternative Aktivitäten sollten so häufig wie möglich verwendet werden
- Kopien mit wenigen Aktivitäten sollten bevorzugt werden

Um die Workflowkopien zu erstellen werden sogenannte “model-checking“ Verfahren verwendet, um eine LTL-Spezifikation durch schrittweises Ausdehnen der LTL-Formel in einen Automation um zu wandeln. Die Automation enthält die Information um sich alle möglichen Replikate des Workflows zu erschließen, die der Workflowspezifikation entsprechen.

Während der Ausführung müssen die verschiedenen Kopien koordiniert werden. Hierzu wird vor Beginn der Ausführung ein sogenannter Koordinator bestimmt. Der Koordinator verwaltet eine Datenstruktur, die folgende Informationen enthält:

- eine Liste alle Aktivitäten der Workflow-Spezifikation und ihrer Kardinalität
- die Aktivitäten, die gerade ausgeführt werden

- die Anzahl der Ausführungen jeder Aktivität zusammen mit deren Ergebnissen

Diese Datenstruktur wird während der Ausführung aktuell gehalten und wird dazu verwendet die weitere Ausführung zu koordinieren.

8.2 Deliverable 6.2 - Robustness models and algorithms

[DRS15] ist eine unveröffentlichte Arbeit und behandelt ebenfalls ein ähnliches Thema. Hier wird vor allem auf Konsistenzen der Ausführungen von Workflow-Instanzen eingegangen.

Es werden Kriterien erstellt, anhand welcher es möglich ist zwischen replizierbaren und nicht replizierbaren Aktivitäten zu unterscheiden.

Ist eine Aktivität nicht replizierbar, heißt das, dass nicht mehrere Instanzen dieser Aktivität ausgeführt werden dürfen.

Es wird eine Unterscheidung getroffen zu starker und schwacher Konsistenz. Die starke Konsistenz, lässt währende der Ausführung eines Workflows zu keinem Zeitpunkt eine Inkonsistenz zu.

Die schwache Konsistenz hingegen, lasst Während der Ausführung eines Workflows Inkonsistenzen zu, fordert aber die starke Konsistenz für die letzte Aktivität.

Dies wurde in Kapitel 2 – Hintergrund ausführlich beschrieben.

8.3 Abgrenzung zu dieser Arbeit

Diese Arbeit greift die Themen der verwandten Arbeiten auf und wendet diese zur Lösung ihrer Problemstellung an. Sie befasst sich mit der Findung von Heuristiken, die zu einer gute und hoch verfügbaren Ausführung von mehreren Workflow-Instanzen führt.

Mit Hilfe der Kenntnisse aus den verwandten Arbeiten, erschafft diese Arbeit ein abstrahiertes Model der realen Welt und versucht anhand dieses Modells eine optimale Verteilung von Workflowreplikaten zu finden, sodass eine möglichst hohe Verfügbarkeit der Ausführung dieser Worklfow-Instanzen erreicht wird.

9 Zusammenfassung & Ausblick

Zusammenfassend sei zu sagen, dass keine, der in dieser Arbeit vorgestellten Heuristiken, die Beste ist. Jede dieser Heuristiken, hat eine Idee zur Grunde liegen, die versucht die Problemstellung best möglich zu lösen. Sie betrachten hierzu fast immer dieselben Parameter. Der Unterschied jedoch ist die Gewichtung, mit der eine Heuristik diesen Parameter mit einbezieht.

Welche Parameter mit welcher Gewichtung mit einbezogen werden, wurde im technischen Teil dieser Arbeit ausführlich erläutert.

Abschließend zu den Heuristiken sei zu sagen, dass die Evaluation gezeigt hat, dass alle der eigenen Heuristiken praktikabel sind. Dennoch gibt es zwei Heuristiken die besonders herausgestochen sind.

Zum einen ist es die Knoten-Service Heuristik, zum anderen die Cluster Heuristik.

Es konnte beobachtet werden, dass die Knoten-Service Heuristik eine Selektion berechnet, die durchschnittlich die schnellste Ausführungszeit bei Ausführungen mit schwacher Konsistenz, besitzt. Auch die theoretische Qualität dieser Heuristik überzeugt sehr. Dennoch wird hier das Augenmerk auf die Ausführungszeit gerichtet.

Für die Cluster Heuristik konnte man sehen, dass sie eine Selektion berechnet, die durchschnittlich die schnellste Ausführungszeit bei Ausführungen mit starker Konsistenz, besitzt. Des weiteren muss während der Laufzeit, im Vergleich zu den anderen Selektionen der anderen Heuristiken, nur selten eine Neuplanung der Selektion erfolgen. Dadurch, dass jedoch nur ein Algorithmus zur Neuplanung erstellt und angewandt wurde, kann man nur sagen, dass die Selektion der Cluster Heuristik länger ausgeführt werden kann, bis sie die erste Neuplanung benötigt. Ab der ersten Neuplanung einer Selektion, ist der ursprüngliche Hintergedanke, warum gerade diese Selektion getroffen wurde, nicht mehr gegeben.

Es war auch zu beobachten, dass das Hinzufügen eines Knotens zu einer Selektion, nicht unbedingt einen Vorteil bringt. Man bedenke, dass pro Knoten der hinzukommt, neue Updates gesendet werden müssen. Außerdem erhöht sich die benötigte Menge der absoluten Mehrheit.

Ausblick

Es wurden nur ein paar Heuristiken entwickelt und evaluiert. Unendlich viele andere Heursitiken könnten hier rein theoretisch noch betrachtet werden.

Auch die vorgestellten Heuristiken die nicht in die Simulation mit eingeflossen sind, bieten Ansatz für weitere Forschungsarbeiten auf diesem Gebiet.

Es waren außerdem ursprünglich noch weitere Messungen bis zu einer Workflowgröße von 10000 geplant.

Das Systemmodell verhindert dies jedoch, da die Werte der theoretischen Qualitäten einer Selektion für einen solch großen Workflow zu schlecht werden. Mit zu schlecht ist gemeint, dass der Datentyp *double* nicht ausreicht um so kleine werte festzuhalten. Man müsste für diesen Fall also das Systemmodell anpassen, oder die Metrik anders berechnen.

Eine Andere Berechnung der Metrik kann auch beinhalten, dass die Einheit mit der gerechnet wird, anders definiert wird. Auch die in dieser Arbeit erhobenen Daten könnten noch genauer betrachtet werden.

Zudem könnte man auch die Simulation auf realen Mobiltelefonen und Computern durchführen, was wohl die naheliegendste und beste Evaluation sämtlicher möglicher Heuristiken wäre.

10 Danksagung

An dieser Stelle möchte ich mich gerne bei den Menschen bedanken, die mich bei dieser Diplomarbeit unterstützt haben.

In erster Linie möchte ich mich aber vor allem bei meinem Betreuer, Herrn Dipl.-Inf. David Richard Schäfer, für seine Unterstützung bedanken. Er hat mir mit seinem fachlichen und auch persönlichen Ratschlägen maßgeblich bei der Anfertigung dieser Diplomarbeit geholfen.

Des weiteren möchte ich mich bei meinen Freunden bedanken, die mich während der Anfertigung dieser Diplomarbeit motiviert und unterstützt haben, die mir geholfen haben Korrektur zu lesen, oder mir anderweitig zur Seite gestanden sind.

Zu guter Letzt möchte ich mich bei meinen Eltern bedanken. Sie haben mir das Studium überhaupt erst ermöglicht, zum einen in finanzieller Hinsicht, zum anderen in persönlicher Hinsicht. Besonderer Dank gilt hier meinem Vater.

Literaturverzeichnis

[AGKW07] B. Alidaee, F. Glover, G. Kochenberger, H. Wang. Solving the maximum edge weight clique problem via unconstrained quadratic programming. *European Journal of Operational Research*, 181(2):592–597, 2007. (Zitiert auf den Seiten 32 und 33)

[BIT15] BITKOM. *44 Millionen Deutsche nutzen ein Smartphone*, 2015 (zuletzt besucht 14.05.2015). URL http://www.bitkom.org/files/documents/BITKOM-Presseinfo_Smartphone_Nutzung_25_03_2015_final.pdf. (Zitiert auf Seite 9)

[BR01] I. D. Baev, R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, S. 661–670. Society for Industrial and Applied Mathematics, 2001. (Zitiert auf Seite 32)

[Bre12] E. Brewer. CAP twelve years later: How the "rules"have changed. *Computer*, 45(2):23–29, 2012. doi:10.1109/MC.2012.37. (Zitiert auf den Seiten 15 und 16)

[DRS15] M. A. T. David Richard Schäfer, Thomas Bach. Deliverable 6.2 - Robustness models and algorithms, 2015. Deliverable of the ALLOW Ensembles (European Union's. Seventh Framework Programme - project 600792). (Zitiert auf den Seiten 12, 13, 14, 15, 17 und 64)

[Egl90] R. Eglese. Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46(3):271 – 281, 1990. doi:http://dx.doi.org/10.1016/0377-2217(90)90001-R. URL http://www.sciencedirect.com/science/article/pii/037722179090001R. (Zitiert auf Seite 33)

[HCP03] J.-L. Huang, M.-S. Chen, W.-C. Peng. Exploring Group Mobility for Replica Data Allocation in a Mobile Environment. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, S. 161–168. ACM, New York, NY, USA, 2003. doi:10.1145/956863.956894. URL http://doi.acm.org/10.1145/956863.956894. (Zitiert auf den Seiten 15 und 38)

[Kau15] S. Kaulfuss. *Simon: Das erste Smartphone, lange vor iPhone und Co.*, 2013 (zuletzt besucht 14.05.2015). URL http://www.giga.de/unternehmen/ibm/news/simon-das-erste-smartphone-lange-vor-iphone-und-co-video-of-the-day/. (Zitiert auf Seite 9)

[KL05] T. Kosar, M. Livny. A framework for reliable and efficient data placement in distributed computing systems. *Journal of Parallel and Distributed Computing*, 65(10):1146–1157, 2005. (Zitiert auf Seite 32)

[KOK09] A. Keränen, J. Ott, T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, New York, NY, USA, 2009. (Zitiert auf Seite 43)

[LC12] B. Liskov, J. Cowling. Viewstamped replication revisited. 2012. (Zitiert auf Seite 32)

[LLS08] G. T. Lakshmanan, Y. Li, R. Strom. Placement strategies for internet-scale data stream systems. *Internet Computing, IEEE*, 12(6):50–60, 2008. (Zitiert auf Seite 32)

[RDR10] S. Rizou, F. Durr, K. Rothermel. Solving the multi-operator placement problem in large-scale operator networks. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, S. 1–6. IEEE, 2010. (Zitiert auf Seite 32)

[Sat01] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001. doi:10.1109/98.943998. (Zitiert auf den Seiten 10 und 11)

[SBTR14] D. R. Schafer, T. Bach, M. A. Tariq, K. Rothermel. Increasing Availability of Workflows Executing in a Pervasive Environment. In *Services Computing (SCC), 2014 IEEE International Conference on*, S. 717–724. IEEE, 2014. (Zitiert auf Seite 63)

[Sch08] U. Schöning. *Theoretische Informatik - kurz gefasst*. Spektrum Hochschultaschenbücher. Spektrum Akademischer Verlag, 2008. URL https://books.google.de/books?id=eFqeJAAACAAJ. (Zitiert auf Seite 32)

[SSB+14] D. R. Schäfer, S. G. Sáez, T. Bach, V. Andrikopoulos, M. A. Tariq. Towards Ensuring High Availability in Collective Adaptive Systems. In *Proc. 1st Int. Workshop of Business Processes in Collective Adaptive Systems: BPCAS*, Band 14. 2014. (Zitiert auf den Seiten 12 und 16)

[Sta15a] Statista. *Bevölkerung - Entwicklung der Einwohnerzahl von Deutschland von 1990 bis 2014 (in Millionen)*, 2014 (zuletzt besucht 14.05.2015). URL http://de.statista.com/statistik/daten/studie/2861/umfrage/entwicklung-der-gesamtbevoelkerung-deutschlands/. (Zitiert auf Seite 9)

[Sta15b] Statista. *Anzahl der Mobilfunkanschlüsse in Deutschland von 1993 bis 2014 (in Millionen)*, 2015 (zuletzt besucht 14.05.2015). URL http://de.statista.com/statistik/daten/studie/3907/umfrage/mobilfunkanschluesse-in-deutschland/. (Zitiert auf Seite 9)

[Ste15] D. Steimels. *Wie alles begann: Die Geschichte des Smartphones*, 2012 (zuletzt besucht 14.05.2015). URL http://www.pcwelt.de/ratgeber/Handy-Historie-Wie-alles-begann-Die-Geschichte-des-Smartphones-5882848.html. (Zitiert auf Seite 9)

[TLX+06] M. Tu, P. Li, L. Xiao, I.-L. Yen, F. Bastani. Replica placement algorithms for mobile transaction systems. *Knowledge and Data Engineering, IEEE Transactions on*, 18(7):954–970, 2006. doi:10.1109/TKDE.2006.114. (Zitiert auf Seite 32)

[Ven06] Y. Venkataramana. Pervasive Computing: Implications, Opportunities and Challenges for the Society. In *Pervasive Computing and Applications, 2006 1st International Symposium on*, S. 5–5. 2006. doi:10.1109/SPCA.2006.297455. (Zitiert auf den Seiten 10 und 11)

[VLC10] G. Vanderhulst, K. Luyten, K. Coninx. Pervasive maps: Explore and interact with pervasive environments. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, S. 227–234. 2010. doi:10.1109/PERCOM.2010.5466973. (Zitiert auf den Seiten 10 und 11)

[Wik15] *Simon: Das erste Smartphone, lange vor iPhone und Co.*, 2013 (zuletzt besucht 14.05.2015). URL http://en.wikipedia.org/wiki/IBM_Simon. (Zitiert auf Seite 9)

[WK+10] T. Williams, C. Kelley, et al. Gnuplot 4.4: an interactive plotting program. *Official gnuplot documentation, http://sourceforge. net/projects/gnuplot*, 2010. (Zitiert auf Seite 43)

[WL02] K. Wang, B. Li. Group mobility and partition prediction in wireless ad-hoc networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, Band 2, S. 1017–1021 vol.2. 2002. doi:10.1109/ICC.2002.997008. (Zitiert auf Seite 15)

[WLL+15] J. Williamson, Q. Liu, F. Lu, W. Mohrman, K. Li, R. Dick, L. Shang. Data sensing and analysis: Challenges for wearables. In *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, S. 136–141. 2015. doi:10.1109/ASPDAC.2015.7058994. (Zitiert auf Seite 19)

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift