



QOS-AWARE CLOUD INFRASTRUCTURE PROVISIONING IN HETEROGENEOUS ENVIRONMENTS

Dem Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
vorgelegte Dissertation

von

DIPLO.-WIRTSCH.-ING. RONNY HANS

Geboren am 28.10.1979 in Wurzen

Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr. Schahram Dustdar

Tag der Einreichung: 20. Juni 2017

Hochschulkennziffer: D17
Darmstadt 2017

Dieses Dokument wird bereitgestellt von / This document is provided by:
tuprints, E-Publishing-Service der Technischen Universität Darmstadt.

<http://tuprints.ulb.tu-darmstadt.de>
tuprints@ulb.tu-darmstadt.de

Bitte zitieren Sie dieses Dokument als / Please cite this document as:
URN: urn:nbn:de:tuda-tuprints-76656
URL: <http://tuprints.ulb.tu-darmstadt.de/id/eprint/7665>

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:
Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International
<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.de>

This publication is licensed under the following Creative Commons License:
Attribution - NonCommercial - NoDerivatives 4.0 International
<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en>



ABSTRACT

Over the last decades Information Technology (IT) has become an enabler for nearly all businesses from industrial production to finance. The IT resources required for these business activities are usually provided by local and remote data centers. Although most resources are still hosted in companies' proprietary data centers, cloud computing initiated a paradigm shift in IT service provisioning from owning to leasing resources and services. Today, over 50% of German companies use cloud services while shifting services into the cloud has become an emerging trend. Cloud computing, which is often referred to as the fifth utility in addition to water, electricity, gas, and telephony, provides commoditized computation resources that are available any time on demand in the required quantity.

However, in contrast to other commodities, a single quality level is insufficient for IT service provisioning. Instead, the required quality for a provided IT service depends on the various functional and non-functional requirements. For example, highly interactive applications such as cloud gaming require a high quality level in terms of latency.

Providers of cloud services have to face a highly competitive market. Cost advantages in cloud computing are primarily achieved by utilizing large centralized data centers at low-cost locations. However, this kind of resource provisioning impacts the quality of service of different types of services such as the aforementioned interactive multimedia services that possess strict quality of service constraints. Hence, infrastructure providers have to face a trade-off between cost reduction and adherence to the required Quality of Service (QoS) attributes.

Apart from how services are provisioned, the way of consuming IT services also changed substantially over the last years. Mobile devices have begun to replace locally installed desktop computers at an accelerated pace. By utilizing these mobile devices, service providers are confronted with two major challenges: (i) a cellular network connection, which potentially causes a higher and more fluctuating latency and (ii) severely limited resources compared to local Personal Computers (PCs). These two aspects restrict the utilization of multimedia services, e. g., cloud gaming.

To address these challenges, we present two novel approaches for (i) resource planning on a global level for multiple services with heterogeneous QoS characteristics and (ii) the augmentation of the centralized cloud infrastructure with locally installed resources to provide viable multimedia services to mobile devices. As the first major contribution, we introduce the Cloud Data Center Selection Problem (CDCSP). This problem describes the data center placement and resource selection on a global scale. We consider the role of a cloud provider, who aims to dimension

resources in a cost-minimal fashion under the consideration of multiple services with different QoS attributes. Based on a mathematical optimization model, we propose the exact solution approach CDCSP-EXA.KOM. Due to the high complexity and the resulting computational effort to find the optimal solution, we propose and analyze four heuristic approaches to identify the most appropriate one for the given problem. As a first heuristic, we propose an approach based on linear program relaxation, CDCSP-REL.KOM. Furthermore, to take the specific structure of the problem into consideration, we develop the custom tailored CDCSP-PBST.KOM approach, which is based on a prioritized processing of demands and supplies. To further improve the results, we combine multiple heuristics to a *Best-of-Breed* approach, named CDCSP-BoB.KOM. Finally, as a metaheuristic improvement procedure, we propose the tabu search approach CDCSP-TS.KOM. To assess the practical applicability and performance of these optimization approaches, we analyze them in detail and compare their performance in a quantitatively.

The second major contribution of this work addresses the augmentation of the centralized cloud infrastructure with local resources to provide services to mobile devices. Therefore, we formulate the Dynamic Cloudlet Placement and Selection Problem (DCPSP), as a multi-period resource planning problem, which includes local characteristics, such as space for hosting resources and available network bandwidth. We focus on a cloud provider who aims to augment the centralized infrastructure using local resources to improve the QoS guarantees for mobile used applications. We formalize the problem as a mathematical optimization model and derive the exact solution approach DCPSP-EXA.KOM. Due to the high complexity that is caused by an optimization over many time slots, we propose the heuristic optimization approach DCPSP-HEU.KOM. We assess the performance of these two approaches by the means of quantitative evaluation.

In summary, the contributions of this thesis provide the means for a cost-efficient and QoS-aware resource selection in cloud infrastructures. We contribute the formalization of the problems and algorithms to support the efficient planning of future cloud infrastructures in environments with a multitude of heterogeneous services on a global scale. Furthermore, to enable mobile users to consume multimedia cloud services, we propose an optimization model and algorithms to augment a global centralized infrastructure by local resource units.

KURZFASSUNG

In den vergangenen Jahrzehnten hat sich die Informationstechnologie (IT) zu einem wesentlichen Produktionsfaktor für nahezu alle Branchen von der industriellen Produktion bis zu Finanzdienstleistungen entwickelt. Obwohl nach wie vor die meisten dafür notwendigen IT-Ressourcen in unternehmensinternen Rechenzentren bereitgehalten werden, hat Cloud Computing in den vergangenen Jahren einen Paradigmenwechsel eingeleitet, bei dem sich ein Wandel vom Besitzen von IT-Ausstattung hin zum Leasen solcher vollzieht. Cloud Computing, bei dem IT-Ressourcen jederzeit standardisiert in beliebiger Menge verfügbar sind, wird dabei oft mit Wasser, Elektrizität, Gas und Telefonie verglichen. Im Gegensatz zu diesen grundlegenden Gebrauchsgütern sind für IT-Dienstleistungen sehr unterschiedliche Qualitätsniveaus erforderlich, welche auf den verschiedenen funktionalen und nicht-funktionalen Anforderungen der bereitgestellten Dienste basieren.

Cloud-Anbieter offerieren ihre Dienste in einem hochgradig kompetitiven Markt, was eine kosteneffiziente Bereitstellung von Cloud-Ressourcen erfordert. Kostenvorteile werden dabei primär durch sehr große und zentralisierte Rechenzentren erzielt. Durch eine geringe Zahl weit entfernter Rechenzentren wird jedoch die Latenz für viele Versorgungsgebiete deutlich erhöht, was sich negativ auf die Nutzbarkeit interaktiver Multimediaanwendungen auswirkt. Aus diesen Gründen muss ein Anbieter von Cloud-Infrastrukturen eine geeignete Abwägung zwischen Kosteneinsparungen und einem angemessenen Qualitätsniveau finden.

Neben der Art wie IT-Dienstleistungen bereitgestellt werden, hat sich auch die Art wie diese konsumiert werden grundlegend in den vergangenen Jahren geändert. Anstelle lokal installierter Desktop Computer werden immer häufiger mobile Endgeräte, wie Smartphones, eingesetzt, was zu weiteren wesentlichen Herausforderungen für die Dienstanbieter führt. Zum einen verursacht die Nutzung des Mobilfunknetzes höhere und stärker fluktuierende Latenzzeiten, was die Nutzung multimedialer Cloud-Anwendungen weiter einschränkt. Ferner, sind diese Geräte durch eine limitierte Ressourcenausstattung, hinsichtlich der Rechenleistung und der Akkukapazität eingeschränkt.

Um die Herausforderungen einer geeigneten Ressourcenbereitstellung zu adressieren, werden im Rahmen dieser Arbeit zwei neuartige Ansätze präsentiert. Im Rahmen des ersten Kernbeitrags wird das sogenannte *Cloud Data Center Selection Problem (CDCSP)* untersucht. Der Schwerpunkt dieses Problems liegt in der Platzierung von Rechenzentren und der Auswahl von geeigneten Ressourcen auf globaler Ebene aus Sicht eines Anbieters von Cloud-Infrastrukturen. Ziel eines Anbieters ist dabei die kosteneffiziente Bereitstellung von Ressourcen unter Be-

rücksichtigung verschiedener Dienstgüteanforderungen. Basieren auf einem in der Arbeit eingeführten mathematischen Optimierungsmodels, wird der exakte Lösungsansatz CDCSP-EXA.KOM hergeleitet. Aufgrund des hohen Rechenaufwands dieses Lösungsverfahrens, werden vier verschiedene heuristische Ansätze untersucht. Das erste Lösungsverfahren, CDCSP-REL.KOM basiert dabei auf dem allgemeinen Prinzip der LP-Relaxation. Um die spezifische Struktur des Problems besser adressieren zu können, werden weitere, speziell zugeschnittene heuristische Verfahren entwickelt und untersucht. Das erste nutzt dabei eine prioritätsbasierte Verarbeitung von angebotenen und nachgefragten Ressourcen. Dieses Verfahren wird um einen sogenannten Best-of-Breed Ansatz ergänzt, der in der Lage ist mehrere einzelne Verfahren zu kombinieren. Abschließend wird das metaheuristische Verbesserungsverfahren Tabu Search zur Lösung des Problems mittels des Lösungsansatzes CDCSP-TS.KOM untersucht. Um die praktische Anwendbarkeit der Verfahren und ihre Leistungsfähigkeit zu bestimmen, werden sie im Rahmen einer quantitativen Evaluation analysiert und miteinander verglichen.

Der zweite Kernbeitrag dieser Arbeit befasst sich mit der Erweiterung der zentralisierten Cloud-Infrastruktur um lokale Ressourcen. Hierfür wurde das *Dynamic Cloudlet Placement and Selection Problem (DCPSP)* formuliert. Dieses Mehrperiodenmodell beinhaltet wesentliche Charakteristika der lokalen Standorte, wie dem Platzangebot zu Ressourcenbereitstellung und der verfügbaren Netzwerkbandbreite. Der Schwerpunkt der Betrachtung liegt auch in diesem Fall auf dem Cloud-Anbieter, dessen Ziel die kostenminimale Erweiterung seine Infrastruktur ist, um die Garantien hinsichtlich der Dienstqualität mobil genutzter Anwendungen zu verbessern. Basierend auf der mathematischen Formalisierung des Problems wird der exakte Lösungsansatz DCPSP-EXA.KOM vorgestellt. Aufgrund dessen hoher Zeitkomplexität durch die genutzte periodenübergreifende Optimierung, wird das heuristische Lösungsverfahren, DCPSP-HEU.KOM, entwickelt. Die Leistungsfähigkeit der Ansätze wird anschließend im Rahmen einer quantitativen Evaluation untersucht.

Zusammenfassend bieten die Beiträge dieser Arbeit die Mittel für eine kosteneffiziente Ressourcenbereitstellung unter Berücksichtigung einer hohen Dienstqualität. Zur Bereitstellung von Ressourcen auf globaler Ebene wird das bestehende Problem formalisiert und es werden geeignete Lösungsverfahren entwickelt um zukünftige Cloud-Infrastrukturen planen zu können. Um ferner auch für mobile Cloud-Nutzer multimediale Dienste mit einer hohen Dienstgüte bereitzustellen zu können, werden ein Optimierungsmodel sowie geeignete Algorithmen zur Erweiterung der globalen Infrastruktur um lokale Einheiten untersucht.

DANKSAGUNG

Allein kommt man schneller voran, zusammen kommt man weiter. Für diese Dissertation, meinem bisher anspruchsvollstes Projekt, gilt dies besonders. Vielen Menschen, die mich auf diesem langen Weg begleitet und tatkräftig unterstützt haben, möchte ich hier meinen Dank aussprechen.

Zu allererst möchte ich mich bei Prof. Dr.-Ing. Ralf Steinmetz bedanken, der mir die Möglichkeit geboten hat, am Lehrstuhl für Multimediakommunikation (KOM) zu arbeiten und diese Dissertation anzufertigen. Vielen Dank für die Betreuung und die Unterstützung in den letzten Jahren. Ebenfalls möchte ich mich bei meinem Korreferenten Prof. Dr. Schahram Dustdar für seine wertvolle Unterstützung beim Anfertigen dieser Arbeit bedanken.

Meine allererste Station am Lehrstuhl war die Forschungsgruppe Service-oriented Computing (SOC). Meinen Kollegen der ersten Stunde: Melanie Holloway, Apostolos Papageorgiou, Dieter Schuller, Olga Wenge und Sebastian Zöller, möchte ich für die sehr gute und inspirierende Zusammenarbeit danken. Ganz besonders möchte ich hierbei Ulrich Lampe erwähnen, der mir sehr geholfen hat in der Forschung Fuß zu fassen. Vielen Dank auch für die enge Zusammenarbeit bei unseren zahlreichen gemeinsamen Publikationen.

Nichts ist so beständig wie der Wandel. Nach langjährigem Bestehen ist unsere SOC-Gruppe 2014 in der Forschungsgruppe Adaptive Overlay Communications (AOC) aufgegangen. Auch bei den vielen neuen Kollegen dieser Gruppe möchte ich mich für die sehr gute Zusammenarbeit und die bereichernden Gespräche bedanken. Besonders danken möchte ich meinem Gruppenleiter Amr Rizk, der neben seinen zahlreichen anderen Aufgaben immer die Zeit gefunden hat, meine Arbeit zu lesen und mir wertvolles Feedback zu geben. Ein großes Dankschön auch an meinen Mitstreiter und Leidensgenossen Björn Richerzhagen, mit dem ich mich gemeinsam durch die heiße Phase des Aufschreibens gekämpft habe. Trotz des eigenen nahenden Abgabetermins hat er immer Zeit gefunden, mir Feedback zu meiner Arbeit zu geben und hat stets eine Lösung parat gehabt, wenn ich an LaTex verzweifelt bin.

Wie ein Fels in der Brandung, der die Gezeiten kommen und gehen sieht, so begleiten unsere ATMs Generationen von Doktoranten von ihren ersten *Gehversuchen* am Lehrstuhl bis zur Promotion. Sie halten durch ihr unermüdliches Engagement den Lehrstuhl am Laufen und unterstützen uns bei unzähligen administrativen, rechtlichen, organisatorischen und technischen Dingen. Sie haben stets ein offenes Ohr, stehen uns mit Rat und Tat zur Seite und sind auch bei den fröhlichen Abenden im KWT nicht wegzudenken. Herzlichen Dank für die viele Unterstützung

und die schöne Zeit an Marion Ehlhardt, Britta Frischmuth-Zenker, Jan Hansen, Monika Jayme, Sabine Kräh, Silvia Rödelsperger, Gisela Scholz-Schmidt, Karola Schork-Jakobi und Stephan Tittel. Ganz besonders möchte ich mich hier bei Frank Jöst für die vielen spannenden und auch lustigen Stunden bedanken, in denen wir uns über unsere IT-Infrastruktur den Kopf zerbrochen haben. Vielen Dank auch, dass du mir in den Monaten vor der Abgabe komplett den Rücken freigehalten hast. Mein Dank geht ebenso an alle Mitglieder des Admin-Teams, mit denen ich die letzten Jahre zusammenarbeiten durfte. Durch Euren unermüdlichen Einsatz habt ihr einen Beitrag zu jeder einzelnen Publikation und zu jedem erfolgreich abgeschlossenen Projekt am Lehrstuhl geleistet.

Immer wieder hatte ich auch die Gelegenheit mit herausragenden Studenten und studentischen Mitarbeitern zu forschen und zu arbeiten. Sie leisten einen wichtigen Beitrag in allen Belangen am Lehrstuhl und tragen auch wesentlich zu unseren Forschungsergebnissen bei. Ganz besonders möchte ich hier Alexander Müller and David Steffen erwähnen. Vielen Dank für Eure großartige Hilfe bei Projekten, der Unterstützung bei unseren Publikationen und die vielen Stunden anregender Diskussionen über heuristische Lösungsverfahren.

KOM besteht aber nicht nur aus Projekten und Forschung. Zu vielen Kollegen hat sich über die Jahre eine Freundschaft aufgebaut, die über die gemeinsamen Konferenzbesuche, spannenden Diskussionen sowie dem wertvollen Feedback zu Publikationen und zur Dissertation hinaus geht. Vielen Dank für die gemeinsamen Urlaube, Festivals, Stadionbesuche und die vielen großartigen Momente an Paul Baumann, Ulrich Lampe, Patrick Lieser, Björn Richerzhagen, Karsten Saller, Steffen Schnitzler, Sophie Schönherr, Dominik Stingl und Viktor Wendel. Ich hoffe, dass noch viele Events folgen werden!

Abschließend möchte ich meiner Familie, namentlich meinen Eltern Siegfried und Hannelore und meinen Schwestern Katja und Kerstin, herzlich danken, die mich schon mein Leben lang bei all meinen Entscheidungen und bei all meinen Vorhaben aus voller Kraft unterstützt haben. Leider konnte unsere Mutter Hannelore die Zeit meiner Promotion nicht mehr miterleben. Neben all den anderen Dingen, hat sie aber durch ihr geduldiges Üben des *kleinen 1x1* schon vor vielen Jahren den Grundstein für diesen Erfolg gelegt.

Vielen Dank Euch allen!

CONTENTS

1	INTRODUCTION	1
1.1	Contributions.....	3
1.2	Thesis Organization	4
2	FUNDAMENTALS	5
2.1	Cloud Computing	5
2.1.1	Definition and Key Characteristics.....	5
2.1.2	Service Models	7
2.1.3	Deployment Models	8
2.1.4	Cloud Market Participants	9
2.2	Costs and Pricing Models in Cloud Computing.....	10
2.2.1	Cost of Physical Infrastructure.....	10
2.2.2	Pricing Schemes for Virtualized Infrastructure.....	13
2.2.3	Pricing Schemes for Software Services	14
2.3	Quality of Service	16
2.3.1	Definition and Categorization of Non-Functional Requirements	16
2.3.2	Service Level Agreements.....	20
3	RELATED WORK	23
3.1	Data Center and Server Placement.....	23
3.2	Cloudlet Placement	26
3.3	Resource Allocation in Cloud Environments.....	27
3.3.1	Market Participants and Location Decisions	27
3.3.2	Cost.....	27
3.3.3	Quality of Services	29
3.4	Conclusion	31
4	STATIC APPROACH FOR A GLOBAL CLOUD INFRASTRUCTURE.....	33
4.1	Problem Statement.....	34
4.2	Optimization Model	34
4.2.1	Formal Notations	35
4.2.2	Mathematical Model.....	36
4.3	Heuristic Optimization Approaches	39
4.3.1	Linear Program Relaxation.....	40
4.3.2	Priority-based Start Heuristics	41
4.3.3	Best-of-Breed Heuristic.....	48

4.3.4 Tabu Search Heuristic	50
4.4 Evaluation	57
4.4.1 Prototypical Implementation	58
4.4.2 Experimental Setup	61
4.4.3 Configurations of the Heuristic Approaches.....	65
4.4.4 Comparison of all Heuristics	80
4.5 Chapter Summary.....	87
5 DYNAMIC APPROACH FOR CLOUDLET ENHANCED INFRASTRUCTURES...	89
5.1 Problem Statement.....	90
5.2 System Architecture	91
5.2.1 Cloudlets	91
5.2.2 Local Integration of Cloudlets.....	94
5.3 Optimization Model	95
5.3.1 Formal Notations	95
5.3.2 Mathematical Model	97
5.4 Heuristic Optimization Approach.....	108
5.5 Evaluation	110
5.5.1 Prototypical Implementation	110
5.5.2 Experimental Setup	113
5.5.3 Comparison of all Heuristics	120
5.6 Chapter Summary.....	124
6 SUMMARY AND OUTLOOK.....	127
6.1 Summary	127
6.2 Outlook.....	129
BIBLIOGRAPHY.....	131
ACRONYMS	149
A APPENDIX	151
A.1 Optimization Model for the Cloud Data Center Selection Problem	151
A.2 Optimization Model for the Dynamic Cloudlet Placement and Selection Problem	152
B AUTHOR'S PUBLICATIONS.....	159
C SUPERVISED THESES	163
D ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG.....	165

INTRODUCTION

CLOUD COMPUTING has enjoyed increasing popularity over the last decade. The basic idea of this paradigm was already announced in 1961 by MIT Professor John McCarthy: "Computing may someday be organized as a public utility just as the telephone system is a public utility [...] Each subscriber needs to pay only for the capacity he actually uses, but he has access to all programming languages characteristic of a very large system" [52].

It required decades and numerous significant developments to come close to the realization of this vision – one of them was the Internet. With the ARPANET (Advanced Research Projects Agency Network), the first packet switching network, communication between computers had its first appearance in 1969 [132]. To manage the increasing size and complexity of the ARPANET, TCP/IP, DNS, and other cornerstone protocols needed to be developed [170]. In 1991, the Centre for Nuclear Research (CERN) launched the *World Wide Web (WWW)* to share linked information in a client-server architecture. With the release of the *Mosaic* browser, the Internet was accessible to the general public [132]. Also in the early 1990 Eric Schmidt, at this time Chief Technology Officer of Sun Microsystems, forecasted the "Computer in the Cloud" [131]. It took until 1999 when Salesforecs became the first company that delivered *Software as a Service (SaaS)* over the Internet. According to later cloud computing definitions, access to their software services was offered as subscription, not as a purchasable license that was a common practice at that time [90].

The development of cloud computing rapidly gained momentum, when big players, such as Amazon, Google, IBM, and Microsoft launched their services about ten years ago. A recent study shows, that more than half of the German companies utilize cloud computing [142] and it still is a major development topic for IT companies [157]. Cisco Systems, Inc., a leading IT company, predicts a 26% Compound Annual Growth Rate for cloud workloads from 2015 to 2020 [32].

Along with the increasing *quantity* of provided cloud services, the significance of non-functional *quality* requirements, i. e., Quality of Service (QoS), grows substantially. For example, over the last years, multimedia applications such as *cloud gaming* and *desktop-as-a-service* have been gaining popularity [9, 33]. Such applications require low latency and high bandwidth. Furthermore, they may require specific hardware accelerated graphics within servers [30, 40]. For future application scenarios, such as the tactile Internet, QoS requirements are predicted to exceed the current ones by far [51].

Along with cloud computing and multimedia applications, increasing user mobility is another trend that has been arising in the past few years. Before the breakthrough of smartphones, which started with the introduction of the first iPhone about 10 years ago, mainly desktop computers were used to access local and remote services. Today, most people rely on smartphones in their daily life. For 2016 the number of used smartphones in Germany has been estimated at 49.2 million [163]. Mobile devices are used – among other things – to access a variety of cloud applications. Because of their characteristics, especially in terms of network connectivity, the provisioning of multimedia cloud services and guaranteeing the adherence of their QoS attributes is a challenging task.

Currently, we are closer to McCarthy's vision of commoditized IT services than ever before. As the fifth utility, computing is available in a similar fashion such as water, electricity, gas, and telephony, i. e., available any time on demand in the required quantity [24]. Because of the wide distribution of cloud services, especially in the business context [32], they have become a crucial part of many supply chains. Hence, providers need to guarantee the adequate service delivery including the corresponding non-functional characteristics.

However, in contrast to the four other commodities, quality requirements for IT services are much more diverse. A single quality level, as would be sufficient for drinking water or electricity in private households, is insufficient for IT service provisioning. Consequently, dimensioning the right amount of appropriate resources is a challenging task for cloud infrastructure providers. They are confronted with various trade-offs regarding costs, capacities, locations, and QoS attributes.

The first trade-off when provisioning cloud resources arises between centralization and decentralization. In cloud computing, cost advantages are achieved through the construction of data centers at low-cost locations and by economies of scale, i. e., consolidating similar resources in huge centralized data centers [7]. This approach saves costs, but leads to poor provisioning of multimedia services, such as cloud gaming [30]. Since latency is directly impacted by the distance and the number of hops between the data center and the users, small regional data centers may be an appropriate solution. Nevertheless, low latency comes at the expense of substantially higher costs because the need to provide distributed data centers near to the users [60]. Hence, a cloud service provider has to balance between centralization, which results in lower cost, and lower latency provided by regionally placed computation resources. Accompanied with this decision, the cloud provider needs to dimension the right amount of provisioned resources. Over-provisioning decreases the revenue the provider. However, an underestimated amount of resources may lead to overload and outages, and, thus, to violations of service level agreements or dissatisfied customers. This may lead to a decreasing number of customers or so called penalty cost.

The resource provisioning for mobile multimedia services is an even more challenging task. Using cellular communication, such as Long-Term Evolution (LTE), the provisioning of IT services may suffer from a high latency compared to Wi-Fi [162]. Further limitations are battery capacity and the cost for using cellular communication techniques [104]. Thus, local resources, connected via Wi-Fi are desirable for mobile cloud computing. Furthermore, all the dimensioning decisions have to be made in an environment, which is characterized by fluctuating demand for numerous applications with various QoS constraints.

1.1 CONTRIBUTIONS

Infrastructure providers are caught between the requirements to reduce costs due to a highly competitive market and the objective to provide an appropriate service quality to different applications and users. In this thesis, we address two major research challenges: (i) the cost-efficient and QoS-aware resource selection on a global scale and (ii) the local resource provisioning to provide multimedia services to customers utilizing mobile devices.

The first contribution addresses the problem of cost-efficient and QoS-aware selection of cloud data centers. Given the cloud definition, resources are available on-demand, in any required quantity, and at any time. Hence, we formulate a static optimization problem that considers a constant amount of provided resource capacities over a given planning period. To analyze the efficiency of the selection of cloud data centers we formalize the underlying *Cloud Data Center Selection Problem (CDCSP)* as a mathematical optimization problem. The resulting model constitutes the exact solution approach for this problem. Due to the intractability of this approach, i. e., the high computational effort required for large problem sizes, we develop suitable heuristic approaches with polynomial complexity. First, we analyze greedy priority-based approaches that are characterized by low computational effort, but reduced solution quality. For an improved solution quality we propose a best-of-breed approach that includes several greedy heuristics. Additionally, we adopt and assess the metaheuristic tabu search, that significantly improves the solution quality while sustaining polynomial computational complexity. Furthermore, we provide a detailed comparison of all solution approaches.

The user behavior regarding the consumption of application services has changed over the last years towards mobile devices, such as smartphones. These devices are characterized by limitations regarding their network connectivity which may result in high battery drain, higher latency, or low transmission rates. Hence, locally provided resources, i. e., cloudlets, provide the means to address these challenges. In this thesis we present our model that augments the global cloud infrastructure using local resources. In contrast to the large-sized cloud data centers

– with their assumption of abundant computing resources, local resources are strongly limited by their computational power and network capacity. Hence, a dynamic optimization approach is required that is able to address fluctuations in demand over multiple time periods, e.g., to capture the different utilization characteristics over a day. To address the challenge of planning and efficiently utilizing such augmented infrastructures, we formulate the underlying *Dynamic Cloudlet Placement and Selection Problem (DCPSP)* as a mathematical optimization model and, thus, provide an exact solution approach. Furthermore, we develop a heuristic optimization approach that calculates a solution with polynomial time complexity. Within the evaluation we assess and compare the provided approaches with respect to their performance.

1.2 THESIS ORGANIZATION

The remainder of the thesis is structured as follows: Chapter 2 provides an overview of the basic concepts of this thesis. First, we introduce the foundation of cloud computing, including service and deployment models, market roles, and enabling concepts and technologies. We give an overview of Quality of Service aspects in the context of multimedia service provisioning and cloud computing. Furthermore, we detail costs and pricing schemes occurring for different service models.

Chapter 3 presents an overview of the related work, on data center placement approaches. Here, we specially focus on Quality of Service issues and therefore we discuss corresponding cloudlet approaches. *Resource allocation*, as closely related topic, addresses the efficient utilization of existing resources. Hence, we give an overview of the major scientific publications in this research area.

Chapter 4 details the first main contribution of this thesis, the formalization of the *Cloud Data Center Selection Problem (CDCSP)* and the development of appropriate solution approaches. Within this chapter we address the challenges of cost minimal and QoS-aware data center selection on a global scale. We provide an optimization model that enables calculating an exact solution for the problem. Furthermore, we present appropriate heuristic solution approaches to solve this problem in reasonable time. We evaluate and compare these approaches.

Chapter 5 provides the second main contribution of this thesis, the formalization and solution of the *Dynamic Cloudlet Placement and Selection Problem (DCPSP)*. Here, we focus on local data centers, i.e., cloudlets, where we consider a time-depended service demand. Therefore, we provide a resource planning approach that addresses the challenges of resource scarcity and time fluctuating workloads. We formulate the corresponding optimization problem and provide an extract solution approach. Additionally, we detail a heuristic approach to address these challenges. Chapter 6 provides a summary of the thesis and outlines directions for future research.

FUNDAMENTALS

In this chapter, we present basic concepts and necessary fundamentals of this thesis. In Section 2.1, we give an overview of cloud computing and its basic concepts. Section 2.2, deals with relevant cost and pricing schemes within the different service models of cloud computing. Section 2.3 introduces the concept of Quality of Service (QoS) and explains the corresponding attributes for cloud computing.

2.1 CLOUD COMPUTING

Cloud computing as an umbrella term for on-demand computing services was initially inspired by the cloud symbol often used to represent the Internet within network illustrations [122]. Over the last years, numerous authors proposed definitions and tried to identify the core characteristics of this new IT paradigm. In the following, we give an overview of the respective definitions and characteristics of cloud computing. First, in Section 2.1.1 we present selected definitions and state their key aspects. Section 2.1.2 describes relevant service models, followed by Section 2.1.3 which explains the commonly used deployment models in cloud computing. Lastly, in Section 2.1.4, we give an overview of the main stakeholders, i. e., market participants in cloud computing.

2.1.1 *Definition and Key Characteristics*

Vaquero et al. [180] are among the first to propose a unified definition for cloud computing. Based on a survey of more than 20 definitions the authors derive the following one: "*Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs.*" The authors state that there are no common features used by all definitions. The most commonly used features are scalability, pay-as-you-go pricing and the virtualization of resources [180].

According to Armbrust et al. [6], cloud computing encompasses hardware resources and application services. Regarding hardware provisioning, which the

authors denote as *cloud*, they find the following novel aspects in cloud computing compared to traditional IT provisioning: (i) the illusion of infinite, on-demand available compute resources, (ii) the avoidance of up-front investments, and (iii) short term availability of resources charged depending on their utilization. Furthermore, if hardware resources are publicly accessible and charged in a pay-as-you-go manner, the authors use the term *utility computing*.

Similar key characteristics are identified by Buyya et al. [24]. In their work, the authors characterize cloud computing by (i) a resource pricing in a pay-as-you-go manner, (ii) resource elasticity and the illusion of an infinite pool of resources, (iii) on-demand self-services, and (iv) virtualized resources. Services are accessible whenever required, independently of the location where they are hosted [24].

According to the National Institute of Standards and Technology (NIST), "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models*" [125]. The authors describe the following key characteristics: (i) on-demand self-service, (ii) service provisioning via the Internet, (iii) pooled resources that are served to multiple consumers, (iv) rapid elasticity to scale resources according to the demand, and (v) a measured and controlled resource usage.

As previously mentioned, publications set different priorities in their definitions [180]. Nevertheless, the most frequently cited publications, also discussed in this section, deliver a rather homogenous picture of cloud computing including the following main characteristics:

- Resources are available in any quantity out of a common resource pool.
- Resources are accessible any time from any place through the Internet.
- The required resources can be scaled up or down on-demand according to the current demand.
- They are provided as virtualized resources.
- Based on measurements of their utilization, resources may be charged in a pay-as-you-go manner.

Furthermore, all definitions include different service and deployment models, as explained in the subsequent sections.

2.1.2 Service Models

In the context of cloud computing, the term Everything as a Service (XaaS) is used frequently. This represents the idea that every required IT component or software application can be provided as a service, which may include Communication as a Service (CaaS), Databases as a Service (DBaaS), Desktop as a Service (DaaS), or Hardware as a Service (HaaS) [103, 149, 154]. Nevertheless, most scientific publications usually refer to three main service models for cloud computing: (i) Infrastructure as a Service (IaaS), (ii) Platform as a Service (PaaS), and (iii) Software as a Service (SaaS) [43, 125, 192, 197].

2.1.2.1 Infrastructure as a Service (IaaS)

This service model forms the lowest and most basic layer, which provides processing, storage, and network capabilities. By the means of these resources users, such higher level software service providers are able to deploy, run, and offer their software products. IaaS users, such as IT administrators, have full control over the virtualized resources and hence over operating systems and deployed applications [125]. Infrastructure resources are usually hosted in large-scale cloud data centers and provided on-demand in the required quantities [180].

Apart from frequently mentioned virtualized resources, some authors divide this service model into additional layers or types. Lenk et al. [114], for example, distinguish between a *physical resource set* and a *virtual resource set*. The physical resource set refers to hardware resources offered by a hardware vendor. These physical resources build the foundation for the virtualized resources upon this layer. Youseff et al. [192] introduce a basic layer named *cloud software infrastructure layer*, which is built upon hardware resources (HaaS) and includes computational resources (IaaS), data storage (DBaaS), and communication (CaaS).

From our point of view, the fundamental idea behind the considered service model is the abstraction from physical hardware resources by the means of virtualization. Hence, an IaaS provider offers scalable, virtualized resources by relying on a sufficient quantity of physical hardware resources hosted in various data centers.

2.1.2.2 Platform as a Service (PaaS)

The next higher layer provides platform services. It offers capabilities and tools for development, testing, deployment, and hosting of applications [43]. Furthermore, platforms often provide services to support marketing, distribution, and operation of applications [15]. By the means of such platforms, services are made accessible to a broad range of clients over the Internet. With an increasing usage of an application, the platform automatically scales the underlying resources without requiring an intervention of the software developer [184]. In contrast to infrastructure services,

developers are not required to maintain the underlying infrastructure, such as updating operating systems or development tools.

2.1.2.3 Software as a Service (SaaS)

On the highest level, software services are provided to the customers [43]. These services are accessible by various devices, such as PCs or smartphones, over the Internet. Therefore, common thin client interfaces, such as web browsers, or specifically developed interfaces, e. g., mobile applications, are used [125]. In contrast to traditional software licensing, users usually pay a usage-independent periodical fee, e. g., on a monthly basis (cf. Section 2.2.3). The applications are hosted within the cloud and might run on top of services offered by PaaS or IaaS providers [8]. Both, providers and users can benefit from using this model. By offering an application using a centralized cloud infrastructure, software installation, maintenance, and versioning are easier to handle for providers. Furthermore, users can easily access an application independent of location, time, or utilized device [6].

2.1.3 Deployment Models

Deployment models refer to different access types of cloud deployments. They specify the group of users that is granted access to services and the relationship between provider and user regarding the organizational affiliation. Four main deployment models can be distinguished: public cloud, private cloud, community cloud, and hybrid cloud [125], as discussed in the following.

2.1.3.1 Public Cloud

Public cloud providers offer their services to everyone willing to pay for it. This openness to the *general public* is the key characteristic of public cloud computing [103]. Here, providers may be business enterprises, academic institutions, or government organizations [125]. Within this deployment model, cloud providers and customers belong to different organizational units, whereby no dedicated contract between these two parties is required. A customer enters into a contract, when he/she accepts the contractual agreements as stated in Service Level Agreements (SLAs) [12]. According to Mell and Grance multiple customers are served by shared resources [125], requiring highly standardized services [80].

2.1.3.2 Private Cloud

This deployment model refers to an exclusive service provisioning within a single organization. Provider and customers may belong to the same organizational unit, but also third parties may provide private cloud services [125]. According to

Zhang et al. [197] private clouds provide the highest degree of control regarding performance, reliability, and security. However, since proprietary server farms are required, this model might not benefit from all advantages associated with cloud computing, such as economies of scale [6] or no up-front capital costs [197].

2.1.3.3 *Community Cloud*

By the means of this model, multiple companies or organizations exclusively share a common cloud infrastructure. Such community clouds are formed based on shared interests or goals, such as security or compliance requirements. In contrast to a private cloud, it can be owned and operated by one or more of the involved companies, a third external party, or a combination of both [125].

2.1.3.4 *Hybrid Cloud*

According to the NIST definition [125], hybrid clouds are compositions of two or more distinct cloud types, i. e., private, public, or community clouds. This deployment model aims to address the limitation of both, public and private clouds. On the one hand, the model offers better control and security over data, since an organization can decide which data is appropriate to be transferred to a public infrastructure or which data requires higher levels of security and privacy. A public cloud, on the other hand, offers a wide range of instantly available and freely scalable resources [197]. Therefore, Baun et al. [12] propose the utilization of public infrastructures for peak loads. Under normal circumstances, a company uses its own private IT infrastructure and only in case of internal resource scarcity, external resources are leased.

2.1.4 *Cloud Market Participants*

Apart from the already stated service and deployment models, cloud computing can also be seen from a business-oriented perspective. Therefore, literature distinguishes between various market participants. To the best of our knowledge and regarding the relations for IT service provisioning, all publications agree on two major roles: (i) cloud infrastructure providers and (ii) users/customers. These two roles are closely tied to the corresponding service models. Cloud providers offer infrastructure services [7, 113, 155] and, thus, can in general be assigned to the IaaS layer. Referring to the service model, these actors can be seen as the first link within the value chain of cloud service delivery. They acquire physical IT infrastructure and hardware to offer scalable infrastructure services (cf. Section 2.2.1). The last link within the cloud value chain are customers that buy and use the applications offered by SaaS Provider [7, 113]. These customers can be private people or companies. However, in literature, there is no consensus about these last market participants. According to

Schubert et al. [155], who refer to *cloud* as infrastructure services only, cloud users are only large companies that outsource their IT infrastructure to the cloud.

Between (IaaS) providers and users literature pose different roles. Based on the three layer service model and the assumption that users are software consumers, an additional role is the SaaS provider [7, 113]. These providers develop and operate applications. Therefore, they might utilize services from platform providers [113, 155]. So far, all market participants can be referred to in corresponding service models. Above these, *aggregators*, *resellers*, or *brokers* combine existing services to create new offers for their customers [113, 155]. For example, they bundle resources originating from different providers to offer greater quantities, increase performance, reduce prices, or ensure SLAs [27, 99, 120, 138]. Lampe [103] summarizes all these additional roles between provider and user as *intermediary*.

2.2 COSTS AND PRICING MODELS IN CLOUD COMPUTING

Regarding the common NIST definition of cloud computing [125], one of its basic characteristics is the pay-as-you-go pricing schema for used resources (cf. Section 2.1.1). Based on a literature review, we identified that this pricing scheme is commonly assumed in the scientific community [106]. In this section we provide deeper insights regarding the cost and pricing schemes arising on different abstraction levels. Here, we focus on three major market participants, i. e., IaaS providers, SaaS providers, and SaaS users as stated in [6]. Generally, IaaS providers have to invest in basic infrastructure, such as data centers, physical hardware components, and software licenses to offer a virtualized infrastructure to their customers, e. g., SaaS providers. Building on this virtualized infrastructure, SaaS providers offer software services to their customers, such as companies or private people. Consequently, in this section we describe the costs and pricing schemes that occur for each of these market players. First, we give an overview of costs regarding the physical environment, which is required to offer virtualized infrastructure (cf. Section 2.2.1). For higher level services – in contrast to the absolute cost – pricing schemes play the most important role when offering services. Therefore, we focus on pricing schemes that are relevant for SaaS providers to use virtualized infrastructure and for customers to use software services (cf. Section 2.2.2).

2.2.1 Cost of Physical Infrastructure

In cloud computing, infrastructure services are mostly provided in terms of on-demand virtualized resources and charged in a pay-as-you-go manner. Virtualized resources are provided on top of physical hardware, such as servers and routers. In context of an IaaS provider, which offers virtualized hardware resources, such

Table 1: Sizes of data centers and the estimated number of racks (based on [189]).

Data center size	Number of racks
Computer room	1-5
Small data center	5-20
Medium data center	20-1,000
Large data center	1,000-10,000
Mega data center	>10,000

as CPU, RAM, and storage, the primary infrastructure components are cloud data centers. Bauer and Adams [11] define a data center as a physical space that is environmentally controlled to enable proper operation of IT hardware. Such a space offers (redundant) components for power supply and cooling. Furthermore, it is physically secured to prevent unauthorized access and it is well connected to the public Internet. To secure the stored information firewalls and security appliances are used. Additionally, load balancers distribute traffic across the servers. The size of a data center varies from a computer room with a few servers to a mega data center with over 10,000 racks [189]. Table 1 illustrates different data center sizes. Over the last years the number of data centers, the used space, and the number of servers have grown rapidly. For example, in Germany the area used for data centers increased by 42% to 1.8 million square meters from 2003 to 2013. Furthermore, in 2013 a total of about 51,100 data center locations were determined in Germany [83].

Operating data centers results in different types of costs, which are described in the subsequent paragraphs. Generally, costs can be divided into two groups: (i) capital expenditures (CapEx) and (ii) operational expenditures (OpEx). CapEx are costs that occur for establishing a new data center location, e.g., cost to acquire property, cost for building a data center, as well as costs for IT equipment such as servers, storage, and network equipment. Expenses for software, maintenance, or warranty also belong to this group [18, 60, 189]. According to Wu and Buyya [189], costs to build up new capabilities, e.g., for research and development to ensure the long-term success of a company, also belong to the capital expenditures. OpEx summarize the costs of all operating activities. While CapEx are costs to acquire inventory, OpEx can be seen as costs to turn inventory into products and services. Examples for such costs are: utility consumption such as electricity, maintenance costs, rental fees, employees' salary, and administrative expenses [60, 189].

For providers, the quantification of total cost is an important factor to calculate profitability. Various publicly available sources of information provide a rough estimation of the total cost and their distribution among different cost types. It

should be noted that the sources do not provide uniform conclusions. A detailed overview of costs is most likely unpublished information only accessible to the provider itself. Nevertheless, based on our literature review we derive an overview of the cost composition for operating data centers in this section. To compare arising expenses, all costs need to be normalized on an annual basis due to different amortization periods. For example, long lived capital cost for facilities are indicated with an amortization period of fifteen years, while servers are indicated with an amortization period of three years [64, 68].

In their work, Greenberg et al. [64] consider a data center with about 50,000 servers. Therefore, the authors identify four major cost components: servers, infrastructure, power usage, and network. In their calculation the authors determine the share of CapEx for servers to be about 45%. The basic data center infrastructure costs which include expanses for power delivery and cooling supplies, are estimated to account for about 25% of the total cost. The energy, which is required for running the infrastructure and servers, is estimated with about 15%. Thus, it is the main contributor to operational cost [198]. The final major cost factor are the network components. It is estimated to account for about 15%, including hardware, link, and transmission cost. Furthermore, the authors estimate that the operational costs for staff are relatively low making up for less than 5% of the total cost [64].

Hamilton [68] uses a slightly different classification, but the values are roughly at the same scale. The author estimates the share of monthly costs as follows: about 53% expenditures for servers, about 23% for power and cooling infrastructure, about 19% for energy cost, and about 5% for other infrastructure expenses.

The study by Koomey [101] results in a different cost allocation. The author estimates the shares for annualized CapEx to three quarters of the total cost and for the annualized OpEx to one quarter. Here, the largest part of annualized CapEx are the costs for site infrastructure which accounts for about 42%, followed by the expenses for IT equipment with about 34%. The energy cost, again the highest single operational cost factor, causes cost of about 11% of the total cost.

The considered publications present different figures for the propositions of various cost types, although all focusing on large scale data centers. The reasons behind these differences are manifold. Two explanations for varying data center costs are given by literature: (i) the *Tier* level and (ii) the data center size. The Uptime Institute defines four different Tier levels [177]. The Tier level is determined by the site availability and thus the overall IT availability in a data center. For example, data centers on the lowest level (Tier I), do not offer redundant power and cooling components, a planned maintenance requires a system shutdown, and failures of site infrastructure cause an interruption in data center operations. In contrast, the highest level (Tier IV), requires, for example, at least two independent electrical systems and multiple distribution paths to simultaneously serve the computing equipment [177]. Since primary drivers for construction cost are power and cooling

capacity [176], a higher Tier level and thus higher availability guarantees, directly correlates with infrastructure capital costs.

The costs also depend on the size of a data center. In general, smaller data centers are relatively more expensive in comparison to large data centers [175, 176]. Providers utilizing large data centers benefit from greater economies of scale in many ways, compared to providers using smaller ones. First, they are able to purchase equipment at high volumes and, thus, they are able to negotiate better prices with suppliers [185]. Furthermore, huge data centers benefit from a high degree of automation. By the means of management software, activities, such as software deployment, monitoring, or dealing with errors, can be highly automated [94]. Using a partial automated structure, such as in well-running companies, Greenberg [64] states an IT staff to server ration of 1:100. A typical ratio in large data centers is 1:1,000.

To summarize, the capital and operational costs of data centers depend on various factors. Publicly available sources give a rough estimate on the cost types and their absolute and relative values. Nevertheless, the sources do not give uniform conclusions regarding these aspects, caused by numerous options when designing and operating data centers. However, literature roughly shows a cost allocation of three quarters for CapEx and one quarter for OpEx.

2.2.2 Pricing Schemes for Virtualized Infrastructure

With cloud computing, consumers are able to shift their CapEx into OpEx, i. e., avoiding investments for physical hardware and instead leasing resources on-demand when required [189]. However, in general, buying hardware resources is always cheaper than leasing them, since a provider adds profit surcharges to earn money. This holds especially true for high and constant resource utilization. Thus, cloud computing is more beneficial for low, fluctuating, or uncertain demand [185].

Since the market for cloud services is highly competitive, IaaS providers have to offer their services at attractive prices and by the means of appropriate pricing schemes. Therefore, in our work [106] we analyzed pricing schemes for virtualized infrastructures, i. e., virtual machines. We found three predominant pricing schemes for virtualized infrastructure offers: (i) pay-as-you-go, (ii) subscription, and (iii) freemium. According to Weinhardt et al. [184], pay-as-you-go is the most frequently used and simplest pricing model in cloud computing. Here, a user pays a fixed price per resource and time unit, which is usually one hour [106].

When using the *subscription* pricing model, a user signs a contract that defines services, prices, and the time period. In contrast to pay-as-you-go, time periods of one or more months are common [151]. The last of the three pricing schemes, *freemium*, is a combination of the words *free* and *premium*. It is often used by

application developers, which offer the basic features of a product free of charge. For richer functionality or the removal of advertisements, a subscription fee is charged [102]. In case of IaaS, VMs with limited resources are offered at no price. The freemium pricing model can be seen as a marketing tool and may serve as an incentive for costumer to get to know a specific product [169].

To gain insights regarding the practical relevance of different cloud pricing schemes, we analyzed publicly available pricing information of 48 IaaS providers. Out of this sample, 81% of the providers offer pay-as-you-go pricing, 50% offer subscription models, and 39% of the providers use freemium as pricing scheme [106]. Note that a single provider may offer multiple pricing schemes.

2.2.3 Pricing Schemes for Software Services

The absolute costs, a consumer has to pay for utilizing software, depend on the used products as well as on the license model, i. e., pricing scheme. Due to the huge amount of available software products and their possible customizations in a business context, no overview regarding absolute costs can be given within this thesis. The relevant question of interest in research and for practical applications comes down again to the choice of appropriate pricing schemes. Thus, in this section we give an overview regarding this topic focusing on cloud computing. The content of this section is primarily based on our recent work [75].

Software, in general, belongs to the group of digital or information goods, which are characterized by high production costs for the initial product, but low expenses for all subsequent copies [67]. In contrast to traditional goods, other types of price differentiation and other pricing schemes are required for selling software as well as providing software as a service via the cloud. Therefore, SaaS providers require appropriate pricing schemes to remain competitive and to ensure revenue [112].

Using these schemes, SaaS providers aim to map the economic value for their customers to their own provisioning cost. Focusing on cloud computing, we classify three different types of pricing schemes based on [112]: (i) usage-independent, (ii) usage-dependent, and (iii) hybrid pricing schemes.

By utilizing *usage-independent pricing schemes*, users have to pay a fee independently from the actual use of the software product. Criteria for pricing are, for example, the number of users or the number of required servers [110]. Here, the major advantage for customers is the planning certainty regarding the arising costs [36]. Furthermore, we distinguish three different types of usage-independent pricing schemes: (i) flatrate, (ii) freemium, and (iii) one-time payment. *Flatrates* offer unlimited access to software services, where all offered functions are included and the user is required to pay a monthly fee. In practice, providers usually offer limited flatrate models. Hence, the number of users, usage time, or the functionality is

restricted [23]. As already described in Section 2.2.2, by the means of *freemium* models, services are offered at no cost with limited functionality or with included advertising. Such offers are usually combined with usage-based offers, such as additional storage. For instance, *Dropbox*¹ offers a freemium service with respect to storage space and charges the enhancement to higher values. Corresponding to *traditional* licensing models for software products, utilizing *one-time payments*, a user pays once for a product and acquires permanent usage permission. As literature shows, SaaS providers use this pricing model only jointly with usage-dependent models in hybrid pricing schemes [23].

By the means of *usage-dependent pricing schemes*, only the consumed units are charged. Thereby, these pricing schemes reflect the cloud payment characteristic *pay-as-you-go*. Here, users with low service usage benefit from lower cost compared to *power users*, which have to pay higher fees. Literature distinguishes between three types [23]: (i) storage-based, (ii) transaction-based, and (iii) time-based models. Using storage-based pricing, the total cost depends on the amount of required storage. As already mentioned, a popular example for this kind of pricing is *Dropbox*. Examples for transaction-based pricing are cost per number of orders or per accounting transaction [23]. This type of pricing might be relevant in business context, but from our point of view, referring to public cloud computing, it is not of relevance. The same holds true for time-based models, where the charged prices differ depending on the time of day or the time of year [23]. Again, to the best of our knowledge, there is no practical relevance for this type of price differentiation regarding the provisioning of software services. While usage-dependent pricing schemes may be beneficial for the consumer with less utilization, providers have to face challenges, such as an unpredictable income stream and a high administrative overhead for measuring the consumption [112].

Hybrid pricing schemes represent a combination of usage-independent and usage-dependent models. For instance, a user acquires a software license pays a one-time fee and is charged with an additional monthly fee for support [112]. By utilizing these pricing schemes, providers are able to ensure a stable stream of revenue.

To assess the practical relevance of the pricing schemes for SaaS providers, Lehmann et al. [111] analyze a sample of 84 providers regarding these three categories. Most providers, 46%, offer usage-independent models, 35% provide hybrid models, and only 7% of them charge their services based on the usage. For the remaining 12% the pricing model could not be specified.

In service provisioning, costs are part of attributes that characterize a service. Apart from these costs, further additional non-functional quality attributes are described in the subsequent section.

¹ www.dropbox.com

2.3 QUALITY OF SERVICE

In general, (cloud) services can be characterized by their functional and non-functional attributes. Functional attributes describe the function of an application, i. e., the defined tasks an application or a service is capable to fulfill [92, 150]. In contrast, non-functional attributes describe the characteristics of a service while it is executed [150]. The compliance with these attributes has a significant influence on the decision to use cloud applications or move locally hosted applications into the cloud [115]. In this section, we focus on non-functional requirements with the purpose to provide a common understanding of relevant terms and to give an overview of relevant Quality of Service (QoS) parameters in the context of cloud computing. Therefore, we present a categorization scheme for QoS attributes in Section 2.3.1. In Section 2.3.2 we give an overview on Service Level Agreements. The content of this section is mainly based on our recent work [75].

2.3.1 *Definition and Categorization of Non-Functional Requirements*

Since the term Quality of Service (QoS) is used in different contexts, no general definition exists. Regarding network operations, the International Telecommunication Union (ITU) [171] defines QoS as: "*Totality of characteristics of a telecommunication service that bear on its ability to satisfy stated and implied needs of the user of the service*". In general, with the goal to "satisfy stated and implied needs of the user", it can also be applied to all kind of services. Nevertheless, Glinz [55] showed that there is no common understanding about the categorization of QoS attributes. Furthermore, due to the heterogeneity of multimedia applications, currently used QoS attributes for this class of services differ from the traditional attributes that describe communication systems [164]. Therefore, based on the categorization schemes of Berbner [17] and Siegel et al. [160], we deduct the categorization schema presented in Figure 1.

At the first stage we distinguish between QoS attributes and costs. Costs are part of non-functional characteristics since the price determines whether a service is used by a customer or not [17]. Because of its importance, we already discussed this topic separately in Section 2.2. According to Berbner [17], we further segment the QoS attributes into quantitative and qualitative attributes.

2.3.1.1 *Quantitative Attributes*

The common characteristic of all quantitative attributes is that they can be measured at least on an ordinal scale. We further group the quantitative attributes in the following three subgroups: (i) performance, (ii) quality assurance, and (iii) agility. Performance is a quality measure that is commonly used in the majority of related publications [55]. Here, we distinguish between two attributes response

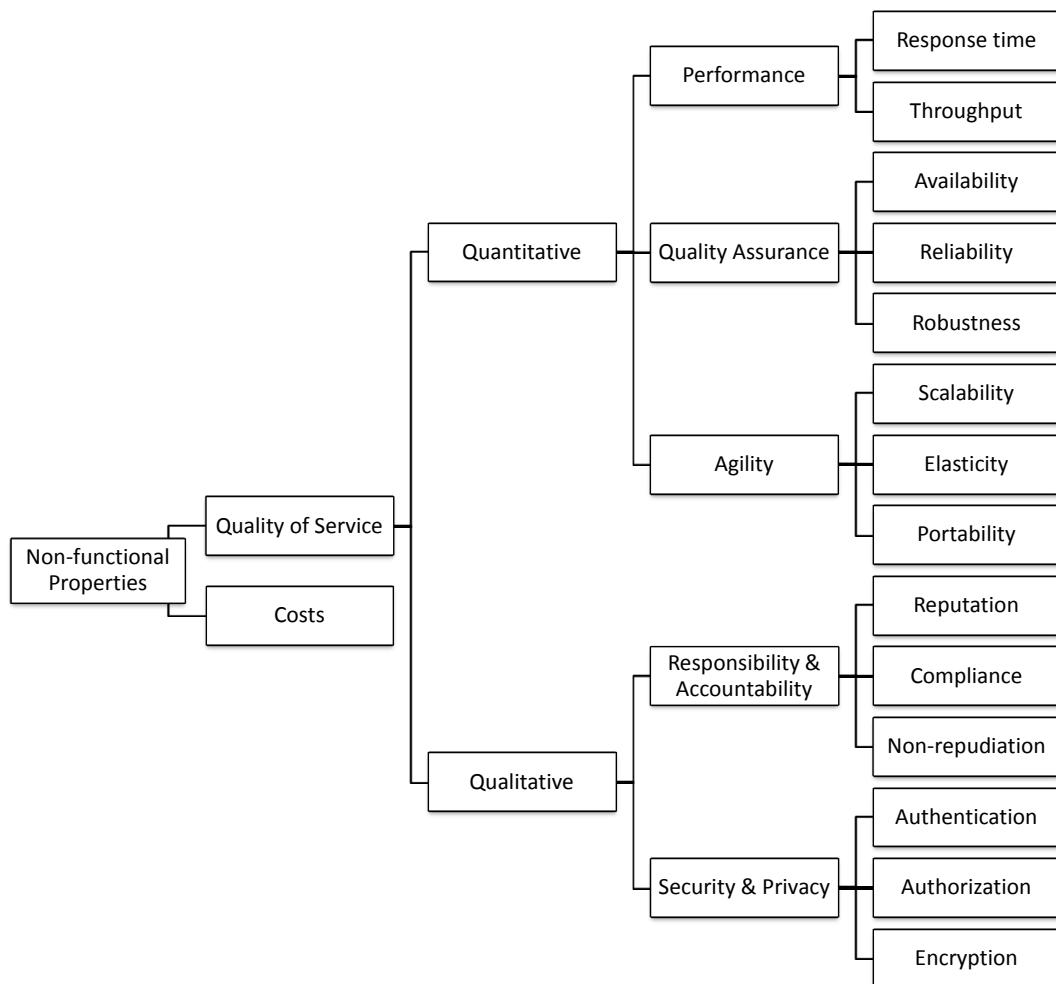


Figure 1: Categorization of non-functional requirements (based on [17, 75, 160])

time and throughput. The response time represents the time between sending a request and receiving the corresponding answer [194]. It is comprised of the time a client needs to process the user input before sending a request to a remote server and the time to process the server response and show the results on the display. Additionally, network latency is also a part of the overall response time. Network latency corresponds to the time required to transfer data to a provider and the response back to the user. Finally, the provider itself requires time to execute or process user requests [30]. The second QoS attribute, *throughput*, is closely related to the execution time and describes the number of requests a system can handle in a given time interval or in average [137].

In the context of multimedia applications, the performance, especially latency, is a crucial factor regarding the quality perceived by users. It directly influences the revenue of market participant, such as online shops, since customers quickly *flee* an online shop for the reason of high latency [93]. Furthermore, the latency requirements directly correlate with the degree of interactivity of an offered application or service [85]. For interactive applications, such as desktop services, literature identified a maximum value of 150 milliseconds for a good usability [173]. For gaming a threshold of 100 milliseconds is proposed [136]. However, this value depends on the played game and even the performed actions within a game. For example, for first person shooter games a value of about 75 milliseconds might influence the gameplay negatively [14]. Recent publications suggest a threshold for latency of 80 millisecond for mail applications and 20 milliseconds for cloud gaming [165]. For future application scenarios even latencies of about one millisecond are expected, which requires computation resources within a distance of 15 kilometers to the user [51].

The category *quality assurance* indicates how likely it is, that a service is provided as specified [160]. We separate between the three different quality measures: (i) availability, (ii) reliability, and (iii) robustness. *Availability* describes the percentage of time a service is instantaneously usable and delivers correct results [53, 63]. It is closely linked to the *reliability* which expresses the number of correct executions in relation to all service invocations in a defined time period [139]. Finally, the *robustness* represents the degree to which a service can provide correct functionality even with invalid, incomplete or conflicting inputs [144].

Based on [160] we use *agility* as a category to summarize the QoS attributes that reflect specific cloud computing characteristics as stated in the definition in Section 2.1.1. Here, we distinguish between three attributes: (i) scalability, (ii) elasticity, and (iii) portability. *Scalability* describes the capability of a provider to increase or decrease the number of provided resources depending on the demand for a specific service. By scaling resources up or down, a provider is able to process more or less requests in a given time period. Thereby, this attribute substantially characterizes the performance of a system [144]. One of the most important, cloud-specific QoS

measures is *elasticity*. This criterion is essentially required in cloud computing since limited resources are offered on-demand in an apparently unlimited amount. Therefore, providers must be able to elastically scale them up and down. Elasticity measures the responsiveness of a system to changes in parameters. Dustdar et al. [47] distinguish three types of elasticity: (i) resource elasticity, (ii) cost elasticity, and (iii) quality elasticity. *Resource elasticity* describes the adjustment of machine characteristics, such as CPU, RAM, or network capabilities. The *price elasticity* represents the responsiveness of the price according to the service demand. Here, a prominent example are spot instances offered by Amazon², which are sold in an auction-based manner and, thus, cause higher cost during peak times. The *quality elasticity* represents the property of adaptable result quality. Here, an example are algorithms in data processing that are able to provide an approximation of a result in short time [47]. Finally, the *portability* represents the ability to move a service from one provider to another. This quantitative attribute measures the costs occurring for such platform changes [119].

2.3.1.2 Qualitative attributes

Qualitative QoS attributes are characterized by the fact that there exists no metric to measure their fulfillment. To assess these QoS attributes, customers – for example – have to be surveyed by the means of questionnaires. We distinguish between *responsibility and accountability* and *security and privacy*.

The first category, *responsibility and accountability*, features the following three subcategories: (i) reputation, (ii) compliance, and (iii) non-repudiation. *Reputation* represents the opinion a customer has about a provider. This QoS criterion is the only one that is exclusively given by the user and cannot be (directly) influenced by the service provider itself [97, 118]. If customers associate positive experiences with a service, the respective provider gains a higher reputation [17].

All private persons and companies are subject to legal regulations. The adherence with legal regulations, as well as with company guidelines is called *compliance* [148]. Rules a service provider has to adhere to are manifold. They depend on the business activity, e. g., financial services, and the kind of processed data, e. g., personal-related data. In the latter case, German and European legal provisions determine for example in which countries such data must be processed. Thus, apart from performance attributes, legal regulations might determine resources that are applicable for specific services. A provider has been held accountable for all actions. Therefore, preconditions are traceability and verifiability of the actions and possible failures [17]. Mechanisms such as logging ensure that a provider cannot deny violations of regulations and agreements. This attribute is called *non-repudiation*.

² <https://aws.amazon.com/ec2/spot/>

The second qualitative category, *security and privacy*, deals with the confidentiality of the information processed and hosted by a provider. To ensure data privacy and avoid unauthorized access by third parties, a provider has to take action to adhere to the following QoS requirements: (i) authentication, (ii) authorization, and (iii) encryption [46]. *Authentication* is the ability of a provider to identify the user. Known users subsequently can be granted with access permissions for specific functions and data, which is called *authorization*. To secure data it should be stored and transmitted using encryption algorithms. Such algorithms modify the data in a way that it is only readable for authorized entities [13]. Both, *responsibility and accountability* and *security and privacy*, are closely related and cannot be considered independently. For example, IT compliance rules impose mechanisms to ensure data security and privacy, such as authentication and authorization. Both are also required to implement non-repudiation.

Such dependencies between single categories apply to many QoS attributes. Generally, as stated in the beginning of this section, no common definition of the term *non-functional requirement* exists. Glinz [55] surveys various existing definitions and finds major differences regarding: (i) definition, (ii) classification, and (iii) representation. The presented classification in the work at hand provides an overview of QoS attributes that are especially important in the context of service provisioning in cloud environments. Nevertheless, because of service diversity and their various technical and legal requirements, we cannot claim completeness regarding the presented schema.

2.3.2 Service Level Agreements

The expected levels of QoS characteristics between a provider and a customer are expressed in the form of Service Level Agreements (SLAs). SLAs are used in nearly all IT-related areas between organizations to define services and their quality level. These contractual agreements are also used between business units within large companies [121]. Traditionally, SLAs are plain natural language documents [121], which consist of (i) involved parties, (ii) service level parameters, and (iii) Service Level Objective (SLO) [156]. Regarding cloud computing, the involved parties are usually a service provider and a service user. These parties agree on service level parameters, i. e., the properties of a service, such as availability. SLOs define the target level of a parameter that has to be achieved [178], e.g., an availability of 99.9%. Furthermore, SLAs determine costs and penalties based on the achieved performance levels [13, 196]. The structure of an SLA is comparable to a traditional contract, whereby SLOs do not use a commonly agreed structure [17]

Utilizing the deployment model of a *public cloud* from Section 2.1.3.1, a customer usually agrees to an SLA, when he/she concludes to the contractual agreements as

stated on the providers website [12]. Thus, in public cloud computing we do not have individually agreed contracts but contracts in form of *general terms and conditions*. Since the involved parties agree on service level parameters and corresponding service level objectives, the achieved values need to be monitored. Because of cloud computing characteristics, such as shared and virtualized resources accessible over the Internet, the determination of these values and the analysis of possible violations are challenging tasks, which can be addressed by reliable monitoring of cloud services by the means of distributed software agents [86]. Furthermore, appropriate monitoring systems need the capability to map low-level resource metrics, such as downtime and uptime to high-level SLA parameters, such as availability [48].

In this chapter we presented the basic concepts required for the understanding of this thesis. We started with an overview of cloud computing including its definition, service models, deployment models, and market participants. Subsequently, we analyzed different costs and pricing models, which occur in cloud computing. Finally, we gave an overview of relevant QoS parameters. In the remainder of this thesis we focus on cloud infrastructure providers with the optimization goal of cost-minimal and QoS-aware service provisioning. Hence, we are focusing on the service model *infrastructure as a service* as presented in Section 2.1.2.1 and the deployment model of *public cloud computing* from Section 2.1.3.1. Since our goal is cost minimization, we consider infrastructure and hardware costs as input parameters for our optimization models. Furthermore, our optimization approaches are open to all kinds of numerically comparable QoS parameters.

RELATED WORK

CLOUD COMPUTING has been a frequently discussed topic for over a decade, both in scientific publications as well as in economical context. Within this broad area, many open research problems are addressed. In this chapter, we focus on related work that is closely associated to the work at hand. Our primary interest lies in the placement and capacity planning of cloud data centers. In addition, we have a closer look at the placement of edge resources, such as cloudlets. Furthermore, we also include the topic of *resource allocation* in existing data centers. Here, we focus on virtual machine placement and the placement of software components in distributed systems. In Section 3.1 we have a closer look at the topic *data center placement*. Here, we focus on optimization problems including exact and heuristic approaches. In Section 3.2, we present cloudlet placement problems and their integration in wireless area networks. Section 3.3 addresses the topic resource allocation by the means of virtual machine (VM) placement. We summarize this chapter in Section 3.4.

3.1 DATA CENTER AND SERVER PLACEMENT

In this section, we address the placement of data centers and the required hardware resources. We include optimization approaches for the selection of appropriate locations and the placement of servers. Although we focus on QoS-aware resource provisioning we do not limit the analysis of the related work to this area and include placement decision problems in general, as they shown a common methodology in problem formalization.

To begin with, Goiri et al. [60] focus on the construction of new data centers. In their work, the authors optimize the choice of appropriate data center locations within the United States of America. With their approach, they minimize the total economic costs for data center providers. These costs are comprised of fixed and variable parts, e. g., costs for buying property, for the construction of facilities, for energy, for cooling, and for labor. Apart from the costs, location decisions are also influenced by other factors such as proximity to power plants and network backbones. By utilizing their proposed optimization algorithms, the authors show a cost reduction amounting to millions of dollars. Furthermore, the authors show that in order to achieve a low end-to-end latency between data centers and customers, infrastructure cost increases substantially.

Among other scientific publications, the approaches in [60] may provide the most practical benefits, since a sophisticated model for data center placement decisions is provided. However, this work is limited when it comes to multiple services with heterogeneous QoS requirements. For example, the authors state that lower latency bounds result in substantially higher cost. Nevertheless, they assume a single latency bound, which may result in a higher number of data centers, even if only a fraction of the considered services require those stringent QoS guarantees. Furthermore, the authors assume an average service demand. Considering different service types, with fluctuating demands may allow more efficient resource provisioning. For example, services with low QoS requirements could be migrated to distant data centers during time periods with high resource utilization in local data centers.

The problem of providing multimedia services through large centralized data centers is addressed by Choy et al. [30]. Regarding the latency constraints of cloud gaming applications, the authors analyze the usability of Amazon's public cloud infrastructure. They find that a small number of centralized cloud data centers and the deployed general-purpose hardware resources result in poor conditions for multimedia service provisioning. As the result of their measurement study, the authors show that only about 70% of the US households are able to reach the nearest cloud data center with a latency of 80 milliseconds or less. Consequently, the authors evaluate different strategies to enhance the current infrastructure by utilizing additional cloud data centers or edge servers. They show that edge resources – compared to relying only on (large) cloud data centers – are the most efficient strategy to enhance the current infrastructure to provide multimedia services. In contrast to our work, the authors simulate a possible distribution of data centers and resources, but they do not use optimization algorithms for their decisions. Further, the authors mention in their work, that different applications pose different latency constraints. Nevertheless, their work focuses on a fixed latency constraint and the heterogeneous landscape of different services is not considered. The authors also do not consider capacity constraints and, thus, assume that the edge servers are able to cover unlimited demand.

Larumbe and Sansò [107–109] address and orchestrate different aspects of data center and software placement in their work such as the location of data centers, locations of software services within the data centers, and related routing problems. As a use case the authors use a web search engine that consists of different software components, i. e., user interfaces, web servers, and index servers. In their publications, Larumbe and Sansò pursue different optimization objectives and solution approaches to address the problems. To begin with, in [108], a mathematical optimization problem was proposed that minimizes the network delay between software components placed in geographically distributed data centers. Depending on a given budget, the authors show the correlation between the number of data centers and latency, which corresponds to the findings in the previously presented publica-

tions. In their subsequent work, Larumbe and Sansò [107] extend their model by considering cost, energy, and environmental constraints. They propose a multi-goal optimization problem that minimizes the total economic cost. As before, an exact optimization approach is used, which suffers from exponentially growing computational requirements. To address this drawback, the authors published a heuristic approach using tabu search for this problem [109]. Depending on the weight of the used coefficients, cost-efficient or delay-optimized solutions are achieved. Although the authors take different software components into consideration, they do not consider multiple services with different QoS constraints. The authors further consider an average demand, which is assumed to be constant over time. Similar to our work, the authors use tabu search as a heuristic approach. However, we determine, analyze, and evaluate problem specific parameters in detail to fit best to the problems described in the work at hand. All previously described approaches focus on (i) reducing the economic cost while ensuring a certain latency level or (ii) reducing the latency itself. The subsequent research publications propose an optimized decision making regarding failure resilience and disaster-aware data center placement.

Chang et al. [28] provide an optimization model for choosing data center locations in a military context. To this end, the authors formulate a linear program with the objective to minimize the total distance between data center facilities and users. To ensure high availability and failure resilience, the authors additionally propose dual homing, i. e., applications are installed in multiple facilities in parallel.

Ferdousi et al. [49] propose a disaster-resilient cloud network to minimize the overall risk of data loss. They provide an approach for disaster-aware data center placement and dynamic content management. Using a static approach, the authors initially place the data centers and assign the content which needs to be provided to the customers. Additionally, they contribute a dynamic algorithm for content management. To this end, they propose a heuristic algorithm which aims to adjust the content placement by limiting the number of content rearrangements, satisfying latency constraints, and reducing the number of replicas within data centers. The authors evaluate their proposed approaches by generating a risk map for various locations. They find that the risk decreases with increasing budget and low latency requirements leads to a higher risk since data centers must be placed in close proximity to each other.

In contrast to our work, the authors do not consider capacity constraints of data centers and they consider latency requirements only within their (dynamic) heuristic approach. Furthermore, latency constraints are only fulfilled if there is a data center available, otherwise a higher latency is tolerated.

All of the reviewed publications propose approaches for data center placements, hardware placements, and some of them for service placements. However, none of them studies optimization approaches for provisioning of QoS-aware multimedia

services. Furthermore, topics such as considering multiple services with different QoS constraints and time variant demands have not been addressed so far.

3.2 CLOUDLET PLACEMENT

After analyzing the placement of data centers, we provide an overview of cloudlet-based placement approaches. Cloudlets are resource-rich compute units that are located in close proximity to the user and can be viewed as small local data centers. Because of their proximity to potential customers, cloudlets enable low latency service provisioning.

Feeehaye et al. [50] analyze the impact of cloudlets on mobile applications. The authors present a cloudlet-based architecture to reduce the propagation delay and increase the throughput for different types of applications. The cloudlets are connected in a peer-to-peer fashion and the mobile nodes, e. g., smartphones, tablets, or notebooks, are affiliated to their nearest cloudlet. To evaluate their results, the authors assess different kinds of applications, such as file editing, video streaming, and collaborative chatting. With their approach the authors achieve a delay reduction if the maximum number of hops between user and cloudlet does not exceed *two*.

Ceselli et al. [26] analyze the placement of cloudlets within 4G access networks in metropolitan areas. The authors place such compute units on base stations and determine the number and capacity of required cloudlets. The authors analyze scenarios for static and dynamic planning, including different methods of VM migration. Their optimization goal is to minimize the cost, which includes installation cost and penalties for a low quality of service. The authors solve the optimization problem using a heuristic approach. Using a dataset containing Internet traffic from a mobile service provider, the authors determine the number of cloudlets that are required to be deployed on base stations to fulfill application requirements with different latency and link usage characteristics.

Since cloudlets mainly offer their services to mobile devices, Wi-Fi coverage plays an important role. To handle the rapidly growing mobile traffic, Dimateo et al. [44] present an architecture to evaluate costs and gains of Wi-Fi offloading. The authors simulate the deployment of Wi-Fi hotspots in San Francisco and analyze how many access points are required to obtain performance improvements.

Our work improves many aspects of the works presented in [26, 44, 50]. In contrast to the approaches presented in [44] and [50], we are realistically tied to a set of fixed locations of Wi-Fi hotspots and cannot choose Wi-Fi or cloudlet locations freely. In contrast to Dimateo et al. [44], where the main goal lies in the reduction of download delay, we focus on highly interactive applications and, thus, on the response time.

3.3 RESOURCE ALLOCATION IN CLOUD ENVIRONMENTS

Many scientific publications focus on resource allocation in cloud environments. These publications cover a broad research area, including placement of virtual machines as one of the basic concepts of cloud computing. Recent surveys found over 150 articles in this research area published over the past years [195]. According to our research focus, we divide this section into the following subsections: *Roles and Locations*, *Costs*, and *Quality of Service*.

3.3.1 Market Participants and Location Decisions

Regarding resource allocation, we can basically distinguish between the market participants as described in Section 2.1.4. In the context of VM placement, the *market participant* denotes the responsible party for placing VMs, which includes decisions concerning the location of resources. In literature, such decisions are mostly made by infrastructure providers and brokers. To begin with, the goal of an infrastructure provider is either an optimized resource allocation in *local* data centers [19, 25, 66, 78, 96, 117, 123, 126, 129, 143, 183, 191] or in *geo-distributed* data centers [1, 98, 116, 141, 146, 158]. Based on these locality aspects different optimization goals can be derived. While in local data centers the server consolidation plays a major role, in geo-distributed data centers energy efficient resource allocation is of great interest. Brokers, as resource resellers, access resources of multiple infrastructure providers [16, 27, 99, 120, 134, 138, 159, 174, 182, 199]. Decisions regarding appropriate resources are often depend on cost and QoS, which are described in detail in the subsequent sections.

3.3.2 Cost

The term cost is used for negative effects, such as higher latency, loss of bandwidth by additional traffic, or monetary cost. In this section we focus on monetary cost, e.g., expenditures for hardware, energy, or leasing VMs. The aspects referring to QoS attributes, e.g., latency or throughput, are analyzed in Section 3.3.3. Optimization goals with respect to cost are closely related to the number of data centers, their distribution, and the considered market participants. For example, a provider who operates multiple geo-distributed data centers is able to reduce cost by increasing data center and server utilization, reducing network traffic, or using locations with cheap energy supply. Within a single data center, cost savings are mainly achieved by efficient server utilization. A broker, in contrast, achieves cost savings by selecting the cheapest provider or the best fitting contract types.

3.3.2.1 Cost Minimization for Infrastructure Providers within Single Data Centers

Considering only single locations, the major aspect for cost savings is efficient server utilization. A higher server utilization, i.e., hosting a multitude of VMs on a single server, results in a small number of required servers and, hence, in lower energy consumption. However, an over-utilization of resources results in an increased response time and, consequently, SLA violations or a degradation of customer's satisfaction. Several publications propose approaches to determine an appropriate server load to adhere to SLAs [78, 191]. One way to achieve this goal is the exploration of multiplexing approaches. Sizing servers according regarding the peak loads of applications results in over-provisioning and additional cost. Applications with different load characteristics can be combined, which results in a lower number of required servers [2, 190]. The consolidation of VMs and, hence, a lower number of required servers result in unused resources. In such cases, energy can be saved by dynamically putting servers into standby or even completely powering them off and on if required [117, 123, 183].

Further, efficiency can be improved by selecting servers depending on the application requirements. Hence, energy cost can be saved by using the most efficient servers for heavy load applications [143].

3.3.2.2 Cost Minimization for Infrastructure Providers in Geo-distributed Environments

A major aspect regarding operational cost of data centers is the energy consumption. A provider that owns multiple geo-distributed data centers can reduce these expenditures by choosing locations with low electricity prices or with better cooling conditions, e.g., areas with lower temperatures [116, 146, 158]. Further aspects considered in literature are the required data center capacities. Reducing the number of required data centers and servers result in increased efficiency and, thus, in lower cost [1, 98]. However, consolidation and centralization of computational resources may cause degradation in service quality. Hence, providers have to find a good trade-off between cost savings and proximity to their customers. Keller et al. [98] investigate resource allocation approaches to meet latency constraints and reduce the number of required servers at the same time. Another aspect which causes additional monetary cost is a high network load. Therefore, corresponding placement approaches which reduce the inter-data center traffic are proposed in [1].

3.3.2.3 Cost-efficient Cloud Broker Approaches

In contrast to infrastructure providers, cloud brokers aim to optimize their cost by selecting providers with reasonable prices and the best suitable pricing schemes. Using placement algorithms, brokers are able to optimize their VM deployment among different clouds to reduce cost, increase performance, or improve QoS guarantees. Regarding the cost, we distinguish between two groups of approaches.

The first group uses a single pricing scheme, i. e., finding the provider that offers the cheapest on-demand prices. Cost can be reduced by efficiently using different predefined VM instance types [174] or finding the lowest on-demand prices for VMs by considering CPU, RAM, and storage prices separately [120]. Another approach by Wenge focuses on profit maximization in cloud collaborations among multiple cloud providers in consideration of regulatory compliance [186]. The second group of approaches aims to find a resource allocation with minimal cost among different pricing schemes, such as on-demand, reservation, or spot pricing. Chaisiri et al. [27] propose an approach to minimize the cost for leasing VMs from different cloud providers by taking on-demand and reservation pricing plans into consideration. Since long-term reservation plans induce less cost than on-demand instances, Wang et al. [182] propose an approach for multiplexing user demands for an efficient utilization of *reserved* VM instances. To ensure both, low latency and low cost, Bellur et al. [16] formulate an optimization approach to select resources from different providers and from different data center locations.

3.3.2.4 Price Negotiation

In contrast to the previously introduced approaches, the following publications propose auction-based mechanisms to determine prices. These approaches aim to increase the profit of providers [103, 193], find the users subjective value for requested bundles of VM [133], or propose multiple-attribute auctions by considering prices and QoS parameters for requested VMs [10]. Further, with a focus on multiple providers Hassan et al. [79] propose different game theoretic approaches to maximize profit or social welfare by cloud providers.

3.3.3 Quality of Services

The adherence to QoS constraints is an important factor for resource allocation, e. g., to fulfill service objective levels agreed upon in SLAs (cf. Section 2.3.2). QoS attributes can either be optimization goals, e. g., minimizing latency, or a constraint for cost reduction approaches. Subsequently, we consider (i) performance parameters, such as response time and throughput and (ii) traffic-aware resource placement.

3.3.3.1 Performance

The main goals of the following publications are the avoidance of congestion and the efficient utilization of resources. Again, we can differentiate between approaches for cloud infrastructure providers and those for brokers.

One aspect that has received much attention is the placement of VMs with respect to server load and network utilization [19, 66, 96]. To improve network scalability using traffic-aware VM placement, Meng et al. [126] analyze the impact of traffic

patterns and network architectures within data centers. Data-intensive applications often need to interact with distributed data storage components. Since this interaction depends on the network between the application server and the storage, Piao et al. [141] propose a network-aware VM placement and migration approach that considers the network characteristics between these components. Sharkh et al. [159] propose an approach to tackle the resource allocation problem for a group of cloud users. They conjointly provision computational and network resources with the aim to minimize delay, minimize cost, maximize resource utilization, and fulfill all user requests. To avoid SLA violations and a decreasing user satisfaction, Morshedlou and Meybodi [129] propose a proactive resource allocation approach which considers additional user characteristics: willingness to pay for service and willingness to pay for certainty. Other approaches propose dynamic placement decisions for continuous streams of deployment requests [25] or consider heterogeneity of servers to reduce the queuing delay [116].

To improve performance and ensure SLA compliance, cloud brokers can choose between various cloud providers, in order to find the provider with the best performance, with the lowest cost associated with a fixed performance level, or with an appropriate trade-off of both properties. For multiple cloud environments, Tordsson et al. [174] propose scheduling algorithms for cross-site deployment of applications. Within their work, the authors enable placements by considering the trade-off between price and performance, e. g., users can control the VM allocation by specifying a maximum budget or minimal performance. Lucas-Simarro et al. [120] analyze optimal service deployments in multi-cloud environments with dynamic pricing. The authors propose several scheduling algorithms to optimize parameters, such as performance or total cost of the infrastructure. Patel and Sarje [138] argue that user requests may exceed resource limits of a single cloud provider. Using multiple data centers in a federated cloud environment the authors aim to reduce SLA violations. For this reason, they propose a load balancing algorithm which considers two types of pricing models, i. e., on-demand and reserved pricing. Kessaci et al. [99] contribute a multi-objective, pareto-based, genetic algorithm to minimize the response time and the cost. With this approach, the authors aim to satisfy the user requirements and to maximize the profit of a cloud broker at the same time.

3.3.3.2 *Latency*

Using geo-distributed clouds for service provisioning, the user satisfaction may suffer from high latency when using highly interactive applications with real-time constraints. Especially for these applications, a short distance between a provider and the users is required [16, 158]. Bellur et al. [16] provide a broker approach to select resources from different cloud providers to meet SLA requirements for highly interactive applications. The proposed approach reduces the total cost by

considering different pricing schemes and meeting the latency constraints of applications, such as virtual desktops and cloud gaming. Zhu et al. [199] propose an optimization algorithm for distributed applications with dependencies in their geo-distributed software components. Their approach focuses on minimizing the user-perceived latency while ensuring low operational cost. Shao et al. [158] focus on the reduction of energy cost by taking queuing delay and transmission delay into account.

In general, the analyzed approaches for resource allocation can be categorized as either provider-based or broker-based. Provider-based approaches mainly aim to save cost (from a customer perspective) by using different cloud providers and/or different pricing schemes. Brokers-based approaches aggregate resources from single cloud providers in order to handle large customer requests. Since our focus lies on a single cloud provider, these approaches are no applicable.

Optimization approaches for single providers focus on single or geo-distributed data centers. For single data centers, efficient VM placement with respect to server utilization and network characteristics plays a major role. With these approaches, for example, the number of required servers can be reduced. In our work, we also focus on required computational resources. Nevertheless, such optimization approaches are not part of this thesis. Therefore, we assume an efficient placement of VMs as a given precondition.

Given our focus on multimedia services, approaches focusing on resource allocation in geo-distributed data centers and the related latency are relevant. Most relevant work regarding this thesis is the work of Shao et al. [158]. The authors pursue the goal to reduce the operational cost by using regions with lower energy cost while adhering to delay constraints. Because they use existing data centers, there is no need for considering fixed cost, which separates it from our work. Furthermore, in our work we consider various QoS attributes for multiple heterogeneous services.

3.4 CONCLUSION

In this chapter we analyzed related work, which considers similar problems and solution methods as the work at hand. We focused on the topics *data center and hardware placement*, *cloudlet placement*, and *resource allocation*.

The most relevant research topic with respect to this work is dealing with *data center and hardware placement*. In contrast to the existing publications, we focus on infrastructure services for QoS-aware multimedia applications. Additionally, we consider multiple heterogeneous services and address the resource planning for service demands that vary over time. The second area on cloudlet placement considers computational resources that are close to potential customers. In this

research area we go beyond the existing literature by considering fixed Wi-Fi locations and focusing on highly interactive applications. Furthermore, we include remote cloud locations within our analysis. The third part, resource allocation, focuses on VM placement. Although the area is closely related and addresses similar problems, the optimization approaches do not take long term hardware planning into consideration.

To the best of our knowledge, our work is the first to addresses resource planning over multiple time periods, including multiple services, with multiple QoS criteria. Further, we analyze resource planning for local data centers, i. e., cloudlets, to augment the cloud infrastructure for mobile devices. In the following chapter we address our first research problem, i. e., the resource planning on a global scale.

4

STATIC APPROACH FOR A GLOBAL CLOUD INFRASTRUCTURE

OVER the last decades Information Technology has become an enabler for nearly all businesses from industrial production to finance. As it has been described in Section 2.2, data centers are the main building block and the major cost factor for this development. For the huge amount of heterogeneous services, cloud infrastructure providers must be able to fulfill individual functional and non-functional requirements of customers, i. e., QoS guarantees. Otherwise, with inadequate QoS guarantees, a cloud provider will not be able to stand its ground in this competitive market. Apart from that, services need to be offered at favorable prices. According to Armbrust et al. [7] the construction and operation of large-scale data centers at low-cost locations is a key enabler for cloud computing. However, such an approach will contradict the goal of QoS guarantees required for multimedia and business applications. To find the right balance between cost minimization and QoS-aware service provisioning for a globally provided cloud infrastructure, we formulate an optimization model for data center selection from the perspective of infrastructure providers by considering the user demand for various services with different QoS requirements. Furthermore, in accordance with common cloud computing characteristics (cf. Section 2.1.1), we assume a provider that is required to fulfill on-demand requests of their customers at any time. Therefore, we propose a static planning model which assumes a sufficiently large amount of resources over a predefined planning horizon.

Considering the huge amount of possible resources, users, services, and QoS requirements, solving such optimization problems can be very costly in terms of computational effort. Therefore, we develop and analyze appropriate heuristic approaches for the described problem.

This chapter proceeds as follows: In the subsequent Section 4.1 we detail the problem statement. In Section 4.2 we present the optimal solution approach including formal notations and the mathematical model. Section 4.3 introduces appropriated heuristic approaches to address the problem in an efficient manner. In Section 4.4 we describe our evaluation procedure. Furthermore, we analyze the best configurations for the heuristic approaches and compare them against each other. Finally, in Section 4.5 we give a summary of the chapter.

4.1 PROBLEM STATEMENT

In our analysis, we assume the role of a cloud provider who aims to offer infrastructure services to its customers, e. g., higher level SaaS providers (cf. Section 2.1). The demand for software services and, thus, the demand for underlying infrastructure services arises from geographically distributed users. We pool these users into so called *user clusters*, to estimate the aggregated demand for a whole area, such as, a city. Each user cluster has a certain demand for particular services. Each service is subject to various QoS constraints, defined as QoS attributes. For example, highly interactive services with low latency requirements, such as cloud gaming, require cloud resources in their vicinity. Financial services with certain compliance constraints may require facilities in a specific legislation, e. g., within the European Union (cf. Section 2.3).

To provide infrastructure services to customers, cloud providers can choose between a set of existing or potential data centers. In this problem, we assume a discrete set of data centers with given locations. Although a continuous model would be feasible for such an optimization, it would not reflect practical requirements, since the location of a data center depends on several factors, such as energy supply or the availability of network access nodes. For the utilization of data centers fixed costs arise, e. g., for construction, long-term leasing, or IT equipment. The operation of resource units, such as servers, causes variable cost, e. g., for maintenance and electricity [64]. Because of the different geographical locations and the particular characteristics of each facility, both, cost and guaranteed QoS, differ between the data centers. The objective of a cloud infrastructure provider is to allocate resources and services to data centers and take decisions regarding the required capacity, i. e., decide on the number of resource units for each data center. The overall goal is to minimize the total cost of the solution. In the following, we refer to this problem as *Cloud Data Center Selection Problem (CDCSP)*.

4.2 OPTIMIZATION MODEL

A model can be seen an abstract representation of a given problem to describe its essence and enable a mathematical analysis. Referring to Operation Research [45], an *optimization model* is a representation of a decision or planning model. To convert the previously described CDCSP into an optimization model, formal notations are required, which are stated in Section 4.2.1. Later on, in Section 4.2.2, we describe the mathematical formalization of our problem in detail. The contents presented in this section were published in [70, 71].

4.2.1 Formal Notations

To formalize the CDSCP as a mathematical optimization problem, several formal notations are required. When we first published this optimization approach in 2013 [70], we used a rather different set of mathematical notations. Due to matters of consistency and readability within this thesis, we use the more elaborate notations required in Chapter 5 for this section as well.

Based on the problem statement, the CDCSP consists of four basic entities. By the means of *data centers*, operated by an infrastructure provider, resources are provided to *user clusters*. These resources are provisioned for different *services* with their individual *QoS attributes*. We express the entities using the following symbols:

- $D = \{d_1, \dots, d_\lambda\}$: Set of (potential) data centers.
- $U = \{u_1, \dots, u_M\}$: Set of user clusters.
- $S = \{s_1, \dots, s_N\}$: Set of services.
- $Q = \{q_1, \dots, q_\Xi\}$: Set of QoS attributes.

All of these basic entities can be characterized by different parameters. First, a data center has certain capacity constraints, i. e., a minimal and a maximal capacity. The maximal capacity is determined, for example, by the available area and technology restrictions. The minimal capacity refers to economic considerations since data centers can only be operated in a cost-efficient manner when a minimal number of resources are provided.

- $K_{d_\lambda}^{\min} \in \mathbb{N}$: Minimal capacity of data center d_λ .
- $K_{d_\lambda}^{\max} \in \mathbb{N}$: Maximal capacity of data center d_λ .

The demand is determined by user clusters, which may request particular services, and consequently hardware resources, in a certain amount.

- $V_{u_\mu, s_v} \in \mathbb{N}$: Resource demand of user cluster u_μ for service s_v .

To provide the corresponding infrastructure services, for each data center, fixed and variable costs arise. A detailed explanation of cost related to data center placement and operation is provided in Section 2.2. For our model, we use the two basic cost types, fixed and variable cost:

- $C_{d_\lambda}^{\text{fix}} \in \mathbb{R}^+$: Fixed cost of selecting data center d_λ .
- $C_{d_\lambda}^{\text{var}} \in \mathbb{R}^+$: Variable cost for operating one resource unit in data center d_λ .
This cost is assumed independent of the provided service.

IT services are characterized by their QoS attributes. Consequently, for IT service provisioning QoS guarantees from provider side and QoS requirements by the user side need to be stated:

- $Q_{d_\lambda, u_\mu, q_\xi}^{g_{u\alpha}} \in \mathbb{R}^+$: QoS guarantee of data center d_λ w.r.t. user u_μ for QoS attribute q_ξ .
- $Q_{u_\mu, s_v, q_\xi}^{req} \in \mathbb{R}^+$: QoS requirement of user u_μ w.r.t. service s_v for QoS attribute q_ξ .

To cover the broad spectrum of conceivable QoS criteria presented in Section 2.3, we define them as positive real numbers \mathbb{R}^+ . The approach to match requirements and corresponding guarantees for different QoS attributes is described in detail in Section 4.2.2.3.

4.2.2 Mathematical Model

Based on the previously described problem and formal notations, a mathematical optimization problem can be formulated. Such a formalized problem consists of several major parts, such as decision variables, objective function, and constraints [82]. Following this order, we explain the required decision variables in the succeeding section. Subsequently, we describe the objective function and the required constraints to address the problem.

4.2.2.1 Decision Variables

Decision variables are related to the quantifiable decision of the optimization problem, i. e., used data centers and provided resources. First, the binary variable x_{d_λ} decides whether the corresponding data center d_λ is selected and used (cf. Equation 4.1). In this case fixed costs will arise. The decision variable $y_{d_\lambda, u_\mu, s_v}$ indicates the number of resource units a data center d_λ provides to a user cluster u_μ w.r.t. service s_v (cf. Equation 4.2).

$$x_{d_\lambda} \in \{0, 1\} \quad \forall d_\lambda \in D \quad (4.1)$$

$$y_{d_\lambda, u_\mu, s_v} \in \mathbb{N} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_v \in S \quad (4.2)$$

4.2.2.2 Objective Function

The objective of the previously described problem is to minimize the cost while adhering to the user's demands and the required QoS constraints. Therefore, we formulate a mathematical function based on the earlier introduced decision variables. The first sum within our objective function calculates the fixed costs that accrue when a data center is chosen. These costs are determined by the cost of data center infrastructure or IT equipment (cf. Section 2.2.1). Whether a data center is used or not depends on the decision variable x_{d_λ} . The second sum refers to the provided resource units and, thus, the variable costs. For each provided resource unit, variable operational costs, $C_{d_\lambda}^{var}$, are incurred. These costs depend on the selected data center and may differ between the data centers because of regionally fluctuating cost, such as expenses for wages or electricity. These costs do *not* depend on the service type or the user operating a service.

$$\text{Min. } C(x, y) = \sum_{\lambda=1..N} x_{d_\lambda} \times C_{d_\lambda}^{fix} + \sum_{\substack{\lambda=1..N \\ \mu=1..M \\ \nu=1..N}} y_{d_\lambda, u_\mu, s_\nu} \times C_{d_\lambda}^{var} \quad (4.3)$$

4.2.2.3 Constraints

Several constraints restrict the solution of this problem and need to be fulfilled: First, the total resource demand V_{u_μ, s_ν} needs to be smaller or equal than the summation of the provided resources (cf. Equation 4.4). Thus, in the planning model, we assume that the provider estimates the whole demand for a foreseeable future and aims to cover it completely. This assumption corresponds to the cloud characteristic of on-demand access to an (apparently) infinite number of computing resources [7].

$$\sum_{\lambda=1..N} y_{d_\lambda, u_\mu, s_\nu} \geq V_{u_\mu, s_\nu} \quad \forall u_\mu \in U, \forall s_\nu \in S \quad (4.4)$$

Regarding the capacity, each data center is characterized by a minimal and a maximal capacity bound. Equation 4.5 denotes that for every selected data center d_λ (i.e., $x_{d_\lambda} = 1$), the sum of the demand for all users over all services is higher or equal than $K_{d_\lambda}^{min}$, such that is economical to use.

$$\sum_{\substack{\mu=1..M \\ \nu=1..N}} y_{d_\lambda, u_\mu, s_\nu} \geq x_{d_\lambda} \times K_{d_\lambda}^{min} \quad \forall d_\lambda \in D \quad (4.5)$$

The upper capacity bound of a data center is determined by several factors, such as the given area, the performance of the server, and the cooling ability of the data

center. Therefore, Equation 4.6 states that for every chosen data center d_λ , the sum of the demand for all users over all services is lower than or equal to the maximum available capacity $K_{d_\lambda}^{\max}$.

$$\sum_{\substack{\mu=1..M \\ \nu=1..N}} y_{d_\lambda, u_\mu, s_\nu} \leq x_{d_\lambda} \times K_{d_\lambda}^{\max} \quad \forall d_\lambda \in D \quad (4.6)$$

The QoS requirements are given by user clusters and their requested services, and must be adhered to. Therefore, we introduce the binary variable $p_{d_\lambda, u_\mu, s_\nu}$, which indicates if a data center is able to guarantee the QoS requirements for a specific user cluster. In the case that one of the required QoS attributes is not fulfilled, a data center cannot provide any resources to respective user cluster (cf. Equation 4.7).

$$y_{d_\lambda, u_\mu, s_\nu} \leq p_{d_\lambda, u_\mu, s_\nu} \times K_{d_\lambda}^{\max} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S \quad (4.7)$$

$$p_{d_\lambda, u_\mu, s_\nu} = \begin{cases} 1 & \text{if } Q_{d_\lambda, u_\mu, q_\xi}^{\text{guar}} \geq Q_{u_\mu, s_\nu, q_\xi}^{\text{req}} \quad \forall q_\xi \in Q \\ 0 & \text{else} \end{cases} \quad (4.8)$$

Within the *if* condition in Equation 4.8 we use a *greater or equal* sign (\geq) to emphasize that the QoS guarantees $Q_{d_\lambda, u_\mu, q_\xi}^{\text{guar}}$ are required to be held and that they are required to be at least as good as the QoS requirements $Q_{u_\mu, s_\nu, q_\xi}^{\text{req}}$. However, to determine the value of $p_{d_\lambda, u_\mu, s_\nu}$ two cases need to be considered. First, for QoS criteria that require a lower or equal value, such as latency and, second, for QoS criteria that require a higher or equal value, such as availability. Since, $p_{d_\lambda, u_\mu, s_\nu}$ is a predefined constant, this procedure do not influence the optimization model at all. Equation 4.9 and 4.10 show these two examples as mathematical expressions. Since, all QoS requirements need to be fulfilled, a data center is only allowed to provide resources, if all conditions, independently of their type, are fulfilled.

$$\{Q_{d_\lambda, u_\mu, q_{\text{Latency}}}^{\text{guar}} \leq Q_{u_\mu, s_\nu, q_{\text{Latency}}}^{\text{req}}\} \rightarrow p_{d_\lambda, u_\mu, s_\nu, q_{\text{Latency}}} = 1 \quad (4.9)$$

$$\{Q_{d_\lambda, u_\mu, q_{\text{Availability}}}^{\text{guar}} \geq Q_{u_\mu, s_\nu, q_{\text{Availability}}}^{\text{req}}\} \rightarrow p_{d_\lambda, u_\mu, s_\nu, q_{\text{Availability}}} = 1 \quad (4.10)$$

The compete model is depicted in Model 5.

4.2.2.4 Problem Complexity

The presented optimization problem constitutes a *Mixed Integer Programming (MIP)* problem, which is a specific form of an Integer Programming (IP) problem. These classes of problems consist of at least one integer or one binary decision variable, respectively [82]. IPs and MIPs belong to the complexity class *NP-hard*, whereby *NP* is an abbreviation for *non-deterministic polynomial*. Therefore, no known algorithm exists that guarantees to solve all problem instances (including the hard ones) in polynomial time [166]. Ways to solve these problems are complete or incomplete enumeration. A well-known and frequently used technique for incomplete enumeration is the *branch-and-bound* algorithm [45]. To apply this solution technique in a comfortable way, free or commercial solver frameworks, such as IBM CPLEX (cf. Section 4.4.1.2), can be used. Nevertheless, NP-hard problems are often not solvable with reasonable effort. Therefore, heuristic approaches are required to solve combinatorial problems in practical applications, as presented in the following.

4.3 HEURISTIC OPTIMIZATION APPROACHES

Exact solution approaches for NP-hard problems may be applicable only for small problem instances and, thus, less appropriate for practical use cases. Rather than finding an optimal solution, *heuristics* aim to find good results within reasonable computational time [147]. Domschke et al. [45] separate heuristic procedures in three major groups: (i) start procedures, (ii) local search and improvement procedures, and (iii) incomplete exact procedures. Start procedures aim to find a first feasible solution for a given problem. The solution quality of this initial solution depends on the design and configuration of the used approach and consequently influences the computational effort. Local search and improvement procedures usually require a feasible solution as starting point to analyze the *neighborhood* of a current solution. Simple local search procedures update the current solution and stop immediately when an iteration does not find a better value compared to the actual solution. Such approaches tend to find solution values within local optima. However, these values may be much worse than the global optimum. To avoid this behavior, metaheuristics provide common guidelines and strategies to overcome local optima. The third group of heuristic solution approaches, incomplete exact procedures, interrupts an exact approach, before it has finished analyzing the complete solution space.

Most heuristic approaches are based on common-sense ideas. The challenge is to design and configure heuristic algorithms, which are perfectly fitting to a specific problem structure to achieve good results [82].

In our research, we are focusing on start and improvement procedures for the CDCSP. In the subsequent sections, we proceed as follows: In Section 4.3.1 we describe the concept of Linear Program (LP) relaxation as a simple method to

transform an integer program into a linear program, such that it can be solved by the means of efficient algorithms. Afterwards, in Section 4.3.2, we present various start heuristics based on priority and cost allocation rules. Building on these results, in Section 4.3.3, we introduce an improvement procedure that combines the best start procedures to achieve a higher solution quality. Finally, in Section 4.3.4, we present the metaheuristic *tabu search* as a sophisticated improvement procedure.

4.3.1 Linear Program Relaxation

The method of LP relaxation was first introduced in the Journal of Mathematics in 1954 by two different articles with the same title: *The Relaxation Method for Linear Inequalities*; One by Shmuel Agmon [3] and another one by Theodore S. Motzkin and Isaac J. Schoenberg [130]. The basic idea of this method is to *relax* the constraint that variables must be integer or binary. Allowing real numbers instead, transforms an IP problem or a MIP problem into a linear program (LP). Such problems belong to the complexity class P and can efficiently be solved in polynomial time [45]. Finally, the solution of a linear program needs to be rounded into integer or binary values, respectively.

The corresponding solution approach for our optimization problem is referred to as CDCSP-REL.KOM [69]. Here, we relax the decision variables x_{d_λ} and $y_{d_\lambda, u_\mu, s_\nu}$ and define them as real, instead of binary or integer. Thus, we end up with following new definitions (cf. Equation 4.11 and 4.12):

$$x_{d_\lambda} \in \mathbb{R}^+, 0 \leq x_{d_\lambda} \leq 1 \quad \forall x_{d_\lambda} \in D \quad (4.11)$$

$$y_{d_\lambda, u_\mu, s_\nu} \in \mathbb{R}^+ \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S \quad (4.12)$$

The major advantage of this approach is its simplicity. Like the described IP, it can directly be solved using an off-the-shelf solver, such as IBM CPLEX, with low additional implementation effort. Nevertheless, the approach is not optimized regarding the structure of the problem and our evaluation proves its inefficiency (cf. Section 4.4). Consequently, heuristics are required, which address the problem in a more efficient way. Therefore, in the subsequent sections we present heuristic start and improvement approaches to address the CDCSP.

4.3.2 Priority-based Start Heuristics

The most basic type of heuristic approaches to determine a first feasible solution of an optimization problem are called *start heuristics* [45]. To compute such an initial solution of a problem with low computational effort, literature proposes greedy approaches. These approaches make choices based on current *knowledge* of an analyzed problem, with the aim to improve the value of the objective function as much as possible [45]. Thereby, greedy algorithms make local optima choices with the aim to reach the global optimum. Depending on the specific problem and the algorithm design, they might be able to achieve nearly optimal solutions [37].

Representatives of greedy approaches are priority-based heuristics. Such heuristics are used for various optimization problems, such as job scheduling, layout planning, or facility location planning [5, 21, 22]. According to Borodin et al. [22], priority-based algorithms are characterized by two properties: First, they order all input values based on predefined rules and second, the decisions made by these heuristics are irrevocable. A decision within an iteration directly becomes part of the final solution. For the work at hand, these priority-based heuristics are of importance in two ways:

1. Because of their conceptual simplicity and computational efficiency, they are suitable to solve the CDCSP in short time. As shown later in our evaluation, they are the best approaches regarding computational effort.
2. These heuristics form the foundation of the subsequently described advanced heuristic solution approaches. They are part of both, the best-of-breed approach (cf. Section 4.3.3) and the tabu search heuristic (cf. Section 4.3.4).

In the subsequent sections, we describe the basic structure of the algorithm and the design of our priority rules. The content of this section is mainly based on our previous work [73]. We enhanced the content of our paper by an additional entity, i. e., *services*. In contrast to the model presented in the paper, which is only able to consider one service (class) per user cluster, we are now able to analyze multiple services with their different QoS requirements for one distinct user cluster. Furthermore, to archive consistency within this thesis we harmonized the mathematical notions w.r.t. to Chapter 5.

4.3.2.1 Basic Structure and Function of the Approach

In an iterative procedure, priority-based approaches assign an ordered list of elements to – for example – facilities. In the following sections we explain how the design principles of priority-bases heuristic are adopted for the CDCSP.

In the first step, before assigning resources, we have to initialize required lists and determine which data centers are able to serve demand w.r.t. specific QoS requirements. Algorithm 1 illustrates the corresponding pseudo code.

Algorithm 1 Initialization of the Assignment Procedure

Start:

```

1:  $D_{u_\mu, s_\nu}^{\text{per}} \leftarrow \emptyset$ 
2:  $U_s^{\text{res}} \leftarrow \emptyset$ 
3:  $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow \emptyset$ 
4: for all  $u_\mu \in U$  do
5:   for all  $s_\nu \in S$  do
6:     if  $V_{u_\mu, s_\nu} > 0$  then
7:        $U_s^{\text{res}} \leftarrow U_s^{\text{res}} \cup \{u_\mu, s_\nu\}$ 
8:        $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow V_{u_\mu, s_\nu}$ 
9:       for all  $d_\lambda \in D$  do
10:       $p_{d_\lambda, u_\mu, s_\nu} \leftarrow \text{true}$ 
11:      for all  $q_\xi \in Q$  do
12:        if  $Q_{d_\lambda, u_\mu, q_\xi}^{\text{gua}} < Q_{u_\mu, s_\nu, q_\xi}^{\text{req}}$  then
13:           $p_{d_\lambda, u_\mu, s_\nu} \leftarrow \text{false}$ 
14:        end if
15:        if  $p_{d_\lambda, u_\mu, s_\nu}$  then  $D_{u_\mu, s_\nu}^{\text{per}} \leftarrow D_{u_\mu, s_\nu}^{\text{per}} \cup \{d_\lambda\}$  end if
16:      end for
17:    end for
18:  end if
19: end for
20: end for
21: for all  $d_\lambda \in D_{u_\mu, s_\nu}^{\text{per}}$  do
22:    $K_{d_\lambda}^{\text{res}} \leftarrow K_{d_\lambda}^{\text{max}}$ 
23: end for

```

Initially, we store both entities referring to the quantitative service demand, i.e., user clusters u_μ and services s_ν , in the list U_s^{res} . This list stores IDs of the currently unserved service demand, i.e., key pairs of user clusters and services. For each service demand, we determine all appropriate data centers that are able to guarantee all QoS attributes $Q_{d_\lambda, u_\mu, q_\xi}^{\text{gua}}$ according to the QoS requirements $Q_{u_\mu, s_\nu, q_\xi}^{\text{req}}$ (cf. Line 9 - 17).

Therefore, we use the binary variable $p_{d_\lambda, u_\mu, s_\nu}$, which is initially set to *true* (cf. Line 10). As soon as one QoS requirement does not hold, it is set to *false* (cf. Line 13). Only if a data center is able to fulfill *all* necessary QoS requirements and, thus, $p_{d_\lambda, u_\mu, s_\nu}$ remains *true*, a data center is added to the list of permitted data centers $D_{u_\mu, s_\nu}^{\text{per}}$ (cf. Line 15). This corresponds to the constraint stated in Model 5 in Eq. A.6. Additionally, we store the residual demand in the list $V_{u_\mu, s_\nu}^{\text{res}}$ and the residual capacity of each data center in the variable $K_{d_\lambda}^{\text{res}}$ (cf. Line 8 and Line 22).

After initializing the required lists and the residual quantities, the assignment procedure starts and it is repeated until the whole demand is assigned to appro-

priate data centers. The pseudo code to find an initial solution is presented in Algorithm 2. At first, we choose a user cluster and its specific service demand out of the list $V_{u_\mu, s_\nu}^{\text{res}}$ (cf. Line 4) and a data center from the list $D_{u_\mu, s_\nu}^{\text{per}}$ (cf. Line 5) based on the priority and cost allocation rules (cf. Section 4.3.2.2). Hereby, the corresponding rules are set once at the beginning of this process and remain the same over the whole time.

Algorithm 2 Determination of an Initial Solution

```

Start:  $D^{\text{open}} \leftarrow \emptyset$ 
1: while  $|U_s^{\text{res}}| > 0$  do
2:   if  $|D_{u_\mu, s_\nu}^{\text{per}}| = 0$  then exit without solution end if
3:   // Choose elements for demand and supply
4:    $\{u_\mu, s_\nu\} \leftarrow \text{SelectServiceDemand}(V_{u_\mu, s_\nu}^{\text{res}})$ 
5:    $d_\lambda \leftarrow \text{SelectDataCenter}(D_{u_\mu, s_\nu}^{\text{per}})$ 
6:   // Reduce residual quantities and save assignment
7:    $y_{d_\lambda, u_\mu, s_\nu} \leftarrow \min(K_{d_\lambda}^{\text{res}}, V_{u_\mu, s_\nu}^{\text{res}})$ 
8:    $V_{d_\lambda, u_\mu, s_\nu}^{\text{ass}} \leftarrow y_{d_\lambda, u_\mu, s_\nu}$ 
9:    $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow V_{u_\mu, s_\nu}^{\text{res}} - y_{d_\lambda, u_\mu, s_\nu}$ 
10:   $K_{d_\lambda}^{\text{res}} \leftarrow K_{d_\lambda}^{\text{res}} - y_{d_\lambda, u_\mu, s_\nu}$ 
11:  // Add and remove elements from various lists
12:   $D^{\text{open}} \leftarrow D^{\text{open}} \cup \{d_\lambda\}$ 
13:  if  $V_{u_\mu, s_\nu}^{\text{res}} = 0$  then  $U_s^{\text{res}} \leftarrow U_s^{\text{res}} \setminus \{u_\mu, s_\nu\}$  end if
14:  if  $K_{d_\lambda}^{\text{res}} = 0$  then
15:    for all  $\langle u_\mu, s_\nu \rangle \in U_s^{\text{res}}$  do
16:       $D_{u_\mu, s_\nu}^{\text{per}} \leftarrow D_{u_\mu, s_\nu}^{\text{per}} \setminus \{d_\lambda\}$ 
17:    end for
18:  end if
19: end while

```

We assign resources depending on the residual service demand of a user cluster $V_{u_\mu, s_\nu}^{\text{res}}$ and the residual capacity $K_{d_\lambda}^{\text{res}}$. The assignment decision that is made in each iteration is final and will not be changed anymore within this procedure. We store the assignments in the list $V_{d_\lambda, u_\mu, s_\nu}^{\text{ass}}$ (cf. Line 8) and reduce the residual capacity and the residual demand accordingly (cf. Line 9 and Line 10). Furthermore, we store the chosen data centers in the list D^{open} (cf. Line 12). If we fulfill a service demand of a user cluster completely, we do not consider it in the following iteration (cf. Line 13). The same holds true for data centers. If the capacity is exhausted completely, we do not take the data center into account for the next iteration (cf. Line 16).

If the complete service demand $V_{u_\mu, s_\nu}^{\text{res}}$ is assigned, the process stops with a valid solution. Otherwise, the solution is invalid, since the constraint in Equation A.2 in Model 5 is not fulfilled. Using these *open* data centers D^{open} and assignments $V_{d_\lambda, u_\mu, s_\nu}^{\text{ass}}$ we calculate the total cost.

4.3.2.2 Priority and Cost Allocation Rules

The purpose of priority-based rules is to order elements in a proper sequence to assign them later on to facilities or machines. Bölte [21] divides such rules into three groups. First, using *serial* procedures all elements are prioritized before the assignment process. Thereby, the sequence will remain constant during the whole assignment procedure. To include already assigned elements and, thus, an already existing partial solution, *alternating* procedures order all remaining elements in each iteration. This procedure benefits from the fact that information about the current assignments are taken into consideration to find solutions with better quality. In both approaches, independent rules are used to order elements and appropriate facilities. With *simultaneous procedures*, elements and the referring facilities are selected within one single rule at the same time. Borodin et al. [22] distinguish between *fixed* and *adaptive* priority-based rules. Using the first one, a fixed order is set at the beginning of the assignment process, whereby, using the second group, a new order is determined in each iteration. This classification matches with Bölttes distinction between serial and alternating rules.

Priority Rules

The primary goal of our heuristic approach is to find a feasible solution that adheres to all constraints stated in Model 5. Of special importance for the priority rules is the constraint given by Equation A.2, which requires that the whole service demand V_{u_μ, s_v} is covered. The secondary goal is the minimization of the total cost. Therefore, priority rules are required to determine an appropriate sequence for the service demand before the assignment process starts. Based on that, we propose and evaluate three different rules that sort the service demand (i) with respect to the quantity of required resources units, (ii) with respect to the available capacities, or (iii) both. For a better readability in the evaluation section, we introduce an identifier ($[P_i]$) for each rule. For all of our rules we use the adaptive approach and re-order the service demand in each iteration. To prioritize the service demand, we use the following three rules:

- The *Demand Priority Rule* (P_1) orders the service demand descending by its residual quantity $V_{u_\mu, s_v}^{\text{res}}$. Using this rule, we prefer services (w.r.t. the user cluster) with higher demand. Here, we assume that serving the highest quantities first it is more likely to achieve a valid solution.
- The *Capacity Priority Rule* (P_2) focuses on the provided capacities. The service demand with the lowest quantity of provided capacity will be served first. Since the possible resources are selected in advance depending on the QoS guarantees, by this rule we take resource scarcity into consideration. For example, if a user cluster demands services which require a very low latency,

so that only a small portion of data centers is able to guarantee such a high QoS constraint, this service demand will be prioritized.

- The *Buffer Priority Rule* (P_3) combines the first two rules. For this purpose, in each iteration, a service buffer is calculated as the margin of residual capacities and residual service demand. The user cluster and service with the lowest buffer will be prioritized.

The first two rules are characterized by their simplicity and thus by a low computational effort. The third one matches demand and supply best. We conjecture that the utilization of the latter rule results in two advantages: First, it is more likely to find a feasible solution when appropriate resources are scarce. Second, since only a few data centers may be able to serve a demand with high QoS requirements, higher costs occur by *setting up* additional data centers. Thus, the selection of appropriate resources and consequently the solution quality might be improved. However, the higher complexity will influence the computational time negatively.

Cost Allocation Rules

Apart from the quantity-based priority rules, we consider the cost of the provided resources by the means of *Cost Allocation Rules*. To determine the cost per resource unit, fixed and variable cost need to be considered (cf. Equation 4.13). While variable cost can be easily charged regarding the used resources, the spread of fixed cost is a challenging task. Since we assign the demand in a step-by-step manner to the data centers, the total amount of resources each data center delivers is unknown until the whole assignment procedure is finished. The total cost of one provided resource unit is calculated by the following formula:

$$C_{d_\lambda}^{\text{unit}}(b) = C_{d_\lambda}^{\text{var}} + C_{d_\lambda}^{\text{fix}} \times (1/b) \quad (4.13)$$

In Equation 4.13 the fraction of the fixed cost charged for each delivered service unit is determined by parameter b . Depending on the value of this parameter, the share of fixed cost per service unit increases or decreases. Therefore, the strategy to estimate this parameter plays an important role for the final result of the optimization. Within the iterative process, there are generally two basic methods to determine this parameter. Either the parameter is calculated once prior to the complete assignment process or, analogous to the priority rules, in each iteration. Here, we differentiate between *static* and *dynamic* cost allocation rules.

Static Cost Allocation Rules: By the means of this class of rules, the earlier described parameter b is determined once at the beginning of the heuristic procedure and

remains constant during the whole assignment process. The following three rules determine the parameter b in Equation 4.13 depending on assumptions of the final data center utilization. For clearer reference, we introduce the identifier ($C[i]$) for each cost allocation rule. We propose the following static rules:

- Using the *Max Capacity Cost Allocation Rule* (C_1) we assume high resource utilization close to the maximum capacity $K_{d_\lambda}^{\max}$ for each data center. As already mentioned, so far the demand exceeds the supply of a single data center, a conclusive statement about the utilization can only be made at the end of the assignment process. So, if this assumption is wrong, the total cost of service provisioning will be underestimated and finally higher costs arise per resource unit.
- Assuming a low utilization, nearby the value of the minimal capacity constraint, the *Min Capacity Cost Allocation* (C_2) will be favorable. Using this rule, the minimum capacity of a data center is used as value for the parameter b and, thus, for calculation of the fixed cost share. The share of fixed cost is higher compared to rule C_1 , which results in higher total cost per service unit. If finally a higher utilization occurs, the cost per provided service unit is overestimated.
- For the last capacity based rule, *Med Capacity Cost Allocation Rule* (C_3), we assume an average utilization of each data center. Therefore, we use the average value between the minimum and maximum capacity of a data center to calculate the parameter b .

If we assume a given set of already existing data centers, with no margin in decision-making (like in classic assignment problems), we can neglect the fixed cost since it will occur in any case:

- The rule *No Fixed Cost Allocation Rule* (C_4) sets the value of the parameter b sufficiently large ($b \leftarrow \infty$), so that the fixed cost do not influence the cost per resource unit.

These static rules are computationally efficient since they need to be calculated only a single time at the beginning of the heuristic assignment procedure. However, they neglect the already assigned resources. To overcome this drawback, we introduce dynamic rules that constantly recalculate the parameter b in each iteration.

Using *Dynamic Cost Allocation Rules* we strive to achieve two goals: On the one hand, we want to achieve a more precise calculation of the fixed cost and, thus, a better solution quality. On the other hand, the implementation of strategies that, for example, are able to primarily choose data centers that are already in use.

- The *Current Utilization Cost Allocation Rule (C5)* calculates the fixed cost depending on the current utilization of a data center. At the beginning, a few number of assignments are charged with the whole fixed cost. With an increasing utilization of a data center the share of charged fixed cost per resource unit decreases. Thus, we reach a realistic cost model for service provisioning. Furthermore, already utilized data centers will be preferred over non utilized data center, which may have positive effects for the total cost.
- To achieve a better load balancing, the *Prefer Minimal Utilization Cost Allocation Rule (C6)* prefers data centers with lower utilization. For this reason, data centers with utilization beneath the minimal capacity value get a higher priority by considering only the variable cost while calculating the cost per unit. For all data centers with an assigned service demand above this threshold value, we also include the fixed cost as described in the previous rule. The advantage of this rule is two-fold: First, by this approach it is more likely that the minimal capacity constraint (cf. Equation A.4) holds. Second, in practical applications this may result in a more balanced workload between the data centers.
- The last dynamic rule is the *Penalize First Cost Allocation Rule (C7)*. Here, we aim to achieve a low number of possible data centers by penalizing for *opening* new data centers. If a new data center should be opened we add the full amount of fixed cost to the cost function. If an already existing data center is used, only the variable costs are considered.

4.3.2.3 Configuration of Heuristics

Within the related work (cf. Chapter 3), we separate between publications regarding the (i) initial placement of data centers placement and (ii) resource allocation using existing data centers. The former deals with location and capacity planning of data centers, the latter with the problem assigning resources to existing facilities. Therefore, we analyze two approaches; (i) a one-stage approach and (ii) a two-stage approach. Combining one priority and one cost allocation rule, we get the one-stage approach. Based on the described rules and their combination we are able to configure 21 different heuristic approaches.

Additionally, we analyze a two-stage approach by running two configurations subsequently. In the first phase, we choose out of a large set of available data centers the best ones. As a result, we get a list with data centers that serve as input for the second stage. Here, we only use these preselected data centers and again run a priority-based heuristic, using either the same rules or a new set of rules. In both of the phases, we can freely assign combinations of priority and cost allocation rules. Thereby, we end up with a total number of 441 different possible heuristics.

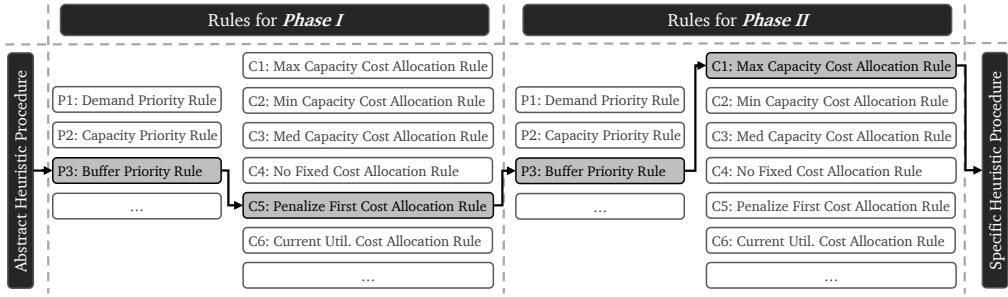


Figure 2: Composition of heuristic approaches (based on [73]).

However, some of the rules are more appropriate in one of the phases. For example, ignoring fixed cost in the first phase will most likely result in a poor solution quality. Figure 2 illustrates the rules and how they are combined to single heuristics.

In the second stage, we allow re-assigning the resources, including dropping, i. e., not choosing, unnecessary data centers. Thus, we expect an improvement in solution quality after running both stages. As we concatenate two heuristic approaches, we further expect the computation time to increase. A detailed analysis of solution quality and computational effort can be found in Section 4.4.3.1.

4.3.3 Best-of-Breed Heuristic

The previously described start heuristics puts us into the position to assemble various heuristics with different configurations. Each heuristic leads to a distinct solution quality and has a distinct performance regarding one similar problem. With the best-of-breed approach, we aim to combine various particular heuristics to ensure a better and constant solution quality by maintaining a favorable performance. The basic idea behind this approach is to run multiple single heuristics subsequently or in parallel for an identical problem instance. The result with the best solution quality, i. e., lowest total cost, is selected as final solution.

For the included heuristics, we choose the best solution approaches from all evaluated test cases. We named the approach *Best-of-Breed (BoB)* and it is published in [74], "Short Run: Heuristic Approaches for Cloud Resource Selection". In general, any possible heuristic may be included within this approach. However, with an increasing number of included heuristics the required computational effort, i. e., the computation time, increases as well. This growth in computation time depends on the number of included heuristics, the characteristics of each heuristic, and the implementation of the BoB approach (serial or parallel). Thus, including all available heuristics, e. g., each of the proposed start heuristic, may lead to a higher solution quality, but it consequently will increase the computation time considerably.

Therefore, the number of heuristics to be included has to be limited. The used priority and cost allocation rules finally determine solution quality and performance of the optimization result. Since we are using deterministic rules, for a predefined set of input parameters, it is easy to determine the best fitting rules. However, input parameters, such as the amount of available resources, also influence the solution quality. In real word scenarios, these parameters vary and they cannot be controlled. To determine which heuristics are to be included in our BoB approach, we use a three-step procedure. First, for each single test case we determine the heuristics with the best solution quality. Subsequently, out of this set, we determine those with the best performance for each single test case. Third, out of all the so far determined heuristics, we choose the minimum possible number of heuristics, that cover all evaluated test cases.

4.3.3.1 Examination of the Solution Quality

To determine the solution quality of our various heuristic approaches, we proceed as follows. First, we combine the proposed priority and cost allocation rules and determine the heuristics with the best solution qualities utilizing the *two-sample t-tests*. This statistical test is used to analyze the hypothesized difference between two samples means [100]. Since we compare two samples, i. e., two algorithms and those measurements were conducted using the same test case, i. e., as set of problem instances with predefined input parameters, and the same computer, these samples are dependent and are analyzed by the means of the *paired two-sample t-test* [95].

To analyze, whether the two samples differ from each other, we use the procedure described by Jain [95] and apply it to all test cases separately.

1. We sort the results of all heuristics ascending and choose the heuristic approach with the best average solution quality. This approach is compared pairwise to all other approaches.
2. We calculate the differences of the means of both samples and thus treat them as one sample in the following calculations. Using this sample we subsequently calculate all required statics measures, i. e., sample standard deviation, sample variance, and the 95% confidence interval.
3. Finally, we verify whether the confidence intervals include zero or not.

If zero is included, from a statistical point of view, there is no difference between the average best heuristic and the analyzed one [95]. As the result of this step, for most cases, we get a group of heuristic approaches that all deliver the *best* solution quality. In the second step, we analyze the heuristic in this set regarding their computation time.

4.3.3.2 Examination of the Computational Effort

For each single test case, we use the set of previously determined *best* heuristics, to find now the fastest algorithms. Again, we use the same procedure, i. e., *paired two-sample t-tests*, as in the preceding section. We determine the approach with the average lowest computation time, use them as benchmark and compare all other *best* heuristics with this fastest one. Finally, for each test case, we get a single heuristic or a group of heuristics, which are the best in terms of solution quality and computation time. Regarding the analyzed test cases, we get between 1 and 24 heuristics. Across all test cases, we obtain a total of 105 heuristics that meet the selection criteria. However, to achieve a favorable low computation time, this number must be limited.

4.3.3.3 Overall Selection of Suitable Heuristic

For the final composition of the heuristics, we have two basic requirements: First, the number of included heuristics should be as low as possible to minimize the computation time. Second, we aim to achieve the best solution quality for each test case, which is possible by the means of priority-based approaches.

To narrow the number of heuristics, we proceed as follows. At first, we count the number of occurrences of each heuristic (including all test cases) and order them ascendingly. The heuristic with the highest number will be included by default. Subsequently, we choose the next heuristic, which includes the most additional test cases. We repeat the procedure, until all test cases are included at least once. In literature, this procedure is referred to as *set covering problem* [188]. In Section 4.4.3.2 we present the evaluation results regarding this approach.

4.3.4 Tabu Search Heuristic

As described before, a NP-hard optimization problem can be addressed by high-performance priority-based algorithms. These algorithms are able to calculate results in short time but tend to find only local-optimum solutions. To overcome this drawback, such search procedures can be guided by higher level procedures, i. e., metaheuristics.

One of the most widely known metaheuristics is *tabu search*, which was introduced 1986 by Fred Glover [56]. With its manageable number of parameters, it is – compared to other approaches – efficiently applicable to various optimization problems. Furthermore, literature shows that it commonly outperforms most other metaheuristic approaches [57, 81, 187]. Therefore, we evaluate its applicability for our optimization problem as well.

Being an improvement procedure, tabu search requires an initial solution as starting point. In each iteration, an existing solution is transformed into multiple

neighborhood solutions. This transformation is determined by predefined *moves*. To overcome local optima, intermediate solutions with a worse objective value compared to a current solution are allowed. Already acquired solutions are stored in the *tabu list* for a defined number of iterations. Thus, directly cycling back to a previously obtained better solution is avoided. Furthermore, one or more stopping criteria are required to end the search process artificially, since – in contrast to exact procedures – we generally cannot make exact statements regarding the achieved solution quality. Such a stopping rule could be the number of performed iterations [82]. The basic structure of the tabu search process is illustrated in Figure 3. The major challenge in applying tabu search is the determination of the relevant parameters to fit specific optimization problems. These parameters are analyzed in the subsequent sections. First, in Section 4.3.4.1, we describe the influence of the initial solution. Afterwards, we explain the local search procedure in Section 4.3.4.2, followed by a description of the tabu list in Section 4.3.4.3. In Section 4.3.4.4 we describe the used stop procedures. The content of the following sections is mainly based on our publication: "*Short Run: Heuristic Approaches for Cloud Resource Selection*" [74].

4.3.4.1 Initial Solution

Since tabu search is an improvement procedure, it requires a starting point, from where it can iteratively begin to analyze the solution neighborhood. Scientific evaluations show that the quality of the initial solution has a substantial impact on the quality of the final result [54].

For the CDCSP, we analyze the influence of two different initial solutions regarding the final solution quality of one test case with a high number of data centers. Thereby, we avoid a rapid coverage of the complete solution space after only a few iterations. To calculate different initial solutions of the problem instances, we use the priority-based heuristics introduced in Section 4.3.2. Out of the sample we choose the one with the best average and the one with an in average worse solution quality:

- Start heuristic 1 - Stage 1: *Buffer Priority Rule (P₃)*, *Penalize First Cost Allocation Rule (C₇)*; Stage 2: *Buffer Priority Rule (P₃)*, *Max Capacity Cost Allocation Rule (C₁)*
- Start heuristic 2 - Stage 1: *Demand Priority Rule (P₁)*, *Max Capacity Cost Allocation Rule (C₁)* Rule; Allocation: *Demand Priority Rule (P₁)*, *Penalize First Cost Allocation Rule (C₇)*

As described in detail in Section 4.4.3.3, the evaluation confirms the findings in literature.

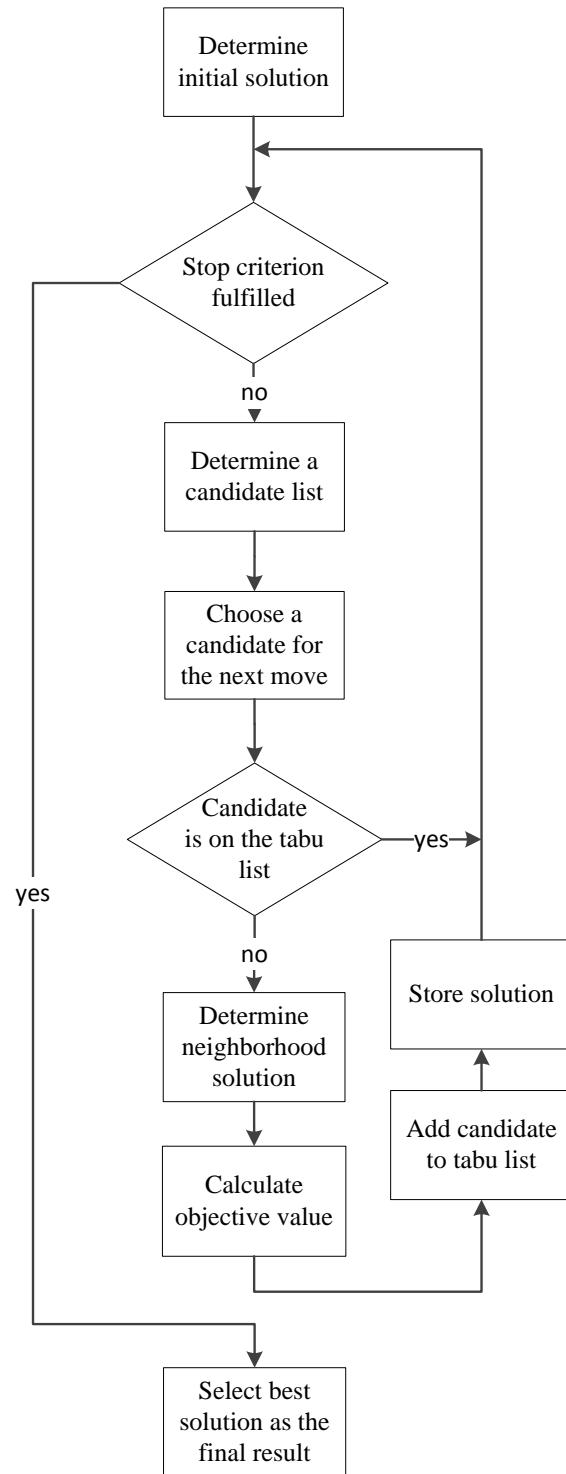


Figure 3: Generic tabu search flow chart. We use candidate lists as input to determine a neighborhood solution.

4.3.4.2 Local Search Procedure

In this step the neighborhood is analyzed based on the initial solution. Thus, as first step, an appropriate neighborhood needs to be defined. This step is crucial since it influences the solution quality and the computational effort of the algorithm. For example, the examination of the complete neighborhood may result in a better solution quality, but also in very high computational effort [59]. The transition of a current solution $x \in X$ into neighborhood solution $NB(x)$ is determined by predefined *moves*, i. e., allowed changes regarding the current solution. A move can consist of changes regarding one or more attributes. Consequently, these two categories are called single-attribute and multi-attribute moves [38].

Regarding our implementation we use single-attribute-moves, i. e., to *open* and to *close* a data center. Thus, within a single iteration it is only allowed to increase or reduce the number of data centers by *one*. Such an approach is also applied to facility location problems [168]. Varying only a single attribute, the number of possible neighborhood solutions corresponds linearly to the number of data centers. Thus, for a given number of m data centers, the solution space is restricted to only m possible neighborhood solutions. For example, assuming a set of 20 potential data centers to be used and a current solution which includes six of them, we have 14 possible candidates to open and six possible candidates to close.

Our moves, i. e., the data centers that are allowed to be opened or to be closed are stored in candidate lists. Such lists store the next values for the neighborhood search [59]. Referring to the two possible moves, we use two lists, one for the data centers that are currently unused $D_{x_{curr}}^{\text{unused}}$ and another one for the currently used data centers $D_{x_{curr}}^{\text{used}}$. The latter one includes all data centers which are part of the current solution. Both candidate lists are sorted by the total cost per resource unit $C_{d_\lambda}^{\text{unit}}$, which is calculated by Equation 4.14. To calculate the total cost, we assume a high utilization of the data centers near to the maximum capacity, since we are expecting that good solutions are characterized by low cost per resource unit. This assumption is also confirmed by the evaluation results of the priority-based start heuristics in Section 4.4.3.

$$C_{d_\lambda}^{\text{unit}} = C_{d_\lambda}^{\text{var}} + C_{d_\lambda}^{\text{fix}} \times (1/K_{d_\lambda}^{\max}) \quad (4.14)$$

The two candidate lists are sorted differently. The first one, $D_{x_{curr}}^{\text{unused}}$, is sorted in ascending order, i. e., data centers with lower estimated units costs have a higher priority. We assume that these data centers are more likely to be part of a better solution. For the list $D_{x_{curr}}^{\text{used}}$, we use the reverse order. Since these data centers are already used and are potential candidates to close, we assume that most expensive

data centers are less likely part of a good solution. Thus, we examine solution without those data centers first.

As starting point for the improvement process our algorithm requires a first feasible solution. The number of iterations, i. e., the number of different solutions analyzed, depends on the stopping criteria described in Section 4.3.4.4. As the first step, we delete the favorite, i. e., the best solution determined in the previous iteration (if we are in the second or a higher iteration). By doing so, we ensure that the algorithm is able to choose a worse solution to overcome a local optimum.

The candidate lists are based on the initial solution or the solution determined in the preceding iteration. After determining the candidate data centers, we order them as previously discussed. Subsequently, we analyze each list separately. The best solution out of these two steps is used for the upcoming iteration.

In Algorithm 3 we start to analyze the neighborhood by adding data centers to the solution from the list $D_{x_{curr}}^{\text{unused}}$ (cf. Line 6). Since the data centers are ordered with ascending cost, the first list entry is the cheapest data center.

As long as the data center is not included in the tabu list (cf. Line 7) we transform the current solution into a neighborhood solution and calculate the objective value, i. e., the total cost. To calculate the total cost, at first, we determine which service demand is provided by current data centers, and, referring to this demand, we examine which of them can be provided by the additionally opened data center w.r.t. the requested QoS characteristics. Subsequently, we analyze, if the additional data center may provide the demand at lower variable cost. If this is the case, we shift the assignments towards the new one. We only consider the variable cost, since we assume that the fixed costs accrue in any case for all data centers. Nevertheless, if we are able to shift the whole service demand from one current data center to the new one, we exclude the one without assignments from the solution and, thus, from the calculation of the total costs. The result of each calculation is stored in long term memory.

If the algorithm returns to a previously calculated solution, it checks, whether the solution value for a set of data centers already exists in memory to avoid additional computations. Literature shows that the effort to maintain such a memory is sufficiently low compared to (re-)solving a problem instance [168].

Afterwards, the costs of the neighborhood solution are compared with the costs of the currently best solution (cf. Line 13). If the new solution achieves lower cost, the current neighborhood solution x' becomes the *favorite* one x_{fav} . To overcome local optima, tabu search is required to allow moves that temporarily decrease the solutions quality. For our approach, we realize this principle by accepting the first calculated solution in each iteration as the *favorite* one. Subsequently, we repeat this process for the next data center on the candidate list until the total costs start to increase. In this case we abort the neighborhood search for the list $D_{x_{curr}}^{\text{unused}}$ and start to analyze the list $D_{x_{curr}}^{\text{used}}$ (cf. Line 17). Basically, we are following the same

procedure, but in this part, we reduce the number of data centers. Again, we analyze the candidate list step by step. But at this point we already have a *favorite* solution from analyzing the first list. Thus, in this step, better solution values are required. Here, the cost calculation differs from the previous part where data centers were added. Since we are reducing the number of resources, a complete re-calculation of the assignments is required. Therefore, we use the earlier described priority-based approach in its single stage form. As priority role we use the *Buffer Priority Rule* (P_3) and as cost allocation rule the *No Fixed Cost Allocation Rule* (C_4) (cf. Section 4.3.2). These two rules are most appropriate for the calculation. The Buffer Priority Rule delivers the most sophisticated order for the service demand (cf. Section 4.4.3). Because of the given and even reduced number of data centers, we only take the variable cost into consideration.

After analyzing the two lists, we complete the iteration by the following finishing steps. If we have found a new and better solution we store it in x_{fin} (cf. Line 29). In the last iteration, this result becomes the final one. Otherwise, the initial solution remains the result of the whole procedure. Finally, we increase the iteration counter by one (cf. Line 31). The best solution found in this iteration is stored as starting point for the next one (cf. Line 32) and the data center that lead to the new solution is added to the tabu list (cf. Line 33).

4.3.4.3 Tabu List

The essential and name giving component of tabu search is the *tabu list*. This short-term memory holds information on already visited solutions. Using this information, the algorithm avoids returning directly to a previously analyzed solution, after a worse one was chosen to overcome a local optimum.

In our implementation we use an attributive and recent-based tabu list [58]. Thereby, only attributes of recently analyzed solutions are stored, instead of all information required to describe a complete solution. In our case, we use a list of data centers whose states were recently changed by adding or removing them from a solution. Elements on the list cannot be changed for a given number of iterations. The tabu list follows the first-in-first-out principle: if the maximum of possible entries is reached, the oldest value is deleted first [65].

A major challenge adopting tabu search to a specific optimization problem is determining a proper size of the tabu list. A tabu list size that is too small may lead to oscillating behavior, i. e., solutions are repeatedly visited and thus the algorithm is not able to overcome a local optimum. In contrast, choosing a value for the tabu list size that is too large may lead to an early termination of the search process, since a large amount of possible solutions is forbidden. Both cases may result in a degradation of the solution quality [59]. The evaluation shows that *three* is an appropriate value for the chosen setup (cf. Section 4.4.3.3).

Algorithm 3 Analysis of the Neighborhood

Start: Initial Solution $x_{\text{init}} \in X$

```

1:  $x_{\text{fin}} \leftarrow x_{\text{init}}$ ;  $x_{\text{curr}} \leftarrow x_{\text{init}}$ 
2:  $c_{\text{Iter}} \leftarrow 0$ ;  $c_{\text{noImpr}} \leftarrow 0$ 
3: while ( $c_{\text{iter}} \leq \text{iterMax}$ ) do
4:    $x_{\text{fav}} \leftarrow \text{Null}$ ;  $\text{costs}(x_{\text{fav}}) \leftarrow \infty$ 
5:   determine  $D_{x_{\text{curr}}}^{\text{used}}$ ,  $D_{x_{\text{curr}}}^{\text{unused}}$ 
6:   for all  $d_\lambda \in D_{x_{\text{curr}}}^{\text{unused}}$  do
7:     if  $d_\lambda \notin \text{tabuList}$  then
8:        $x' \leftarrow \text{transformSolutionOpen}(d_\lambda, x_{\text{curr}})$ 
9:       if  $\text{cost}(x') < \text{cost}(x_{\text{fav}})$  or  $\text{cost}(x_{\text{fav}}) = \infty$  then
10:         $x_{\text{fav}} \leftarrow x'$ 
11:         $d_{\lambda}^{\text{fav}} \leftarrow d_\lambda$ 
12:       else
13:         stop analyzing candidate list  $D_{x_{\text{curr}}}^{\text{unused}}$ 
14:       end if
15:     end if
16:   end for
17:   for all  $d \in D_{x_{\text{curr}}}^{\text{used}}$  do
18:     if  $d_\lambda \notin \text{tabuList}$  then
19:        $x' \leftarrow \text{transformSolutionClose}(d_\lambda, x_{\text{curr}})$ 
20:       if  $\text{cost}(x') < \text{cost}(x_{\text{fav}})$  then
21:          $x_{\text{fav}} \leftarrow x'$ 
22:          $d_{\lambda}^{\text{fav}} \leftarrow d_\lambda$ 
23:       else
24:         stop analyzing candidate list  $D_{x_{\text{curr}}}^{\text{unused}}$ 
25:       end if
26:     end if
27:   end for
28:   if  $\text{cost}(x_{\text{fav}}) < \text{cost}(x_{\text{fin}})$  then
29:      $x_{\text{fin}} \leftarrow x_{\text{fav}}$ 
30:   end if
31:    $c_{\text{iter}}++$ 
32:    $x_{\text{curr}} \leftarrow x_{\text{fav}}$ 
33:    $\text{tabuList} \leftarrow \text{tabuList} \cup \{d_{\lambda}^{\text{fav}}\}$ 
34: end while

```

4.3.4.4 Stopping Conditions

Tabu search, as a metaheuristic, does not have a natural stopping condition. Thus, appropriate rules for ending the search process are required. Such rules must ensure a good trade-off between solution quality and computational effort.

In their work, Glover and Tailard [59] name four possible stopping conditions: (i) The optimal solution was found, (ii) no valid neighborhood solution can be calculated, (iii) a predefined number of iterations without improvement were performed, and (iv) a given number of iterations has been reached.

The first option may hold true for a low number of elements to be assigned, but for combinatorial problems with an exponentially growing solution space, this option requires the analysis of the complete solution space. Thus, for our approach, this option is not applicable and we only consider the last three.

The second stopping condition comes into play if no valid solution exists or all possible solutions are blocked by the tabu list. In our case, this rule does not apply for two reasons: First, by adding data centers to a solution, we are relaxing the problem. Starting with a feasible solution, additional data centers cannot turn a valid solution into an invalid one. Furthermore, since we set the size of the tabu list lower than the total number of data centers, this will not happen either. Thus, the case that no valid neighborhood solution can be calculated will not occur in our implementation. However, this procedure may lead to numerous iterations and, thus, a bad trade-off between solution quality and computational effort. Here, the third and the fourth stopping condition might come into play. The evaluation shows that the average solution quality correlates with the increasing number of iterations. This behavior is comprehensible since we finish each tabu search process with the best solution found so far. Even with rare and small improvements, while analyzing lots of problem instances, the average solution quality will increase with a higher number of iterations. Therefore, we only use the fourth rule as stopping criterion for our algorithm and we limit the total number of iterations (cf. Section 4.4.3.3).

4.4 EVALUATION

In the previous sections we presented solution approaches for the *Cloud Data Center Selection Problem (CDCSP)*. In this section we derive suitable configurations for each heuristic approach. Furthermore, we evaluate all approaches to conclude which of them is most suitable for the given problem. In Section 4.4.1, we describe our prototypical implementation. Subsequently, in Section 4.4.2, we present the setup for our evaluation. All developed heuristic approaches require a profound analysis of their respective configuration. Therefore, in Section 4.4.3, we explain and evaluate the configuration of our approaches. Afterwards, we compare our heuristics to each other and discuss the results in Section 4.4.4.

4.4.1 *Prototypical Implementation*

To evaluate our approaches, we prototypically implemented them using the object-oriented programming language Java and the relational data base MySQL. In the subsequent sections, we introduce our Java code packages, the external libraries utilized in our project, and the required evaluation databases.

4.4.1.1 *Java Code Packages*

For our implementation, we adopt and extend an evaluation framework based on former results of our research group. We use a similar basic structure and the same naming convention for all packages as described in [103]. The classes are distributed across several packages in accordance with their functionality:

- **ENTITIES:** This package includes all classes to represent basic entities of the CDCSP, including *DataCenter*, *UserCluster*, *ServiceType*, and *QoSAttribute*. Each generated object is able to store the corresponding parameters, e. g., capacity constraints or demand.
- **EVALUATION:** This package includes the *CdcspEvaluationHelper*, which contains all required methods for the evaluation, such as starting the optimization of a problem instance or storing the evaluation results.
- **GENERATOR:** This package contains classes to generate problem instances for the CDCSP. For example, the *UsCountiesCloudDataCenterProblemGenerator* uses information from the 2010 US Census to generate problem instances based on real population data.
- **OPTIMIZATION:** This package includes classes to implement the exact optimization approach CDCSP-EXA.KOM and the LP-relaxed optimization approach CDCSP-REL.KOM. By the means of these classes, we compile and configure the linear equation system. Furthermore, the class *OptimizationManager* provides methods commonly required by all optimization approaches, e. g., to calculate the total cost, store detailed assignment information, or to verify the validity of a problem.

The required approaches for the heuristic approaches are stored in the following packages:

- **OPTIMIZATION.HEURISTICS:** This package includes all classes to run the heuristic approaches, such as the priority-based start heuristic, the BoB approach, or the tabu search.
- **OPTIMIZATION.HEURISTICS.PRIORITYIZATION:** Here, all classes to realize the priority rules are included.

- **OPTIMIZATION.HEURISTICS.COSTING:** Within this package, all required classes that realize the cost allocation rules are stored.
- **OPTIMIZATION.HEURISTICS.TABU:** This package features all classes required to configure the tabu search, such as classes to set the parameters for the tabu list and the number of iterations.

To setup and run an evaluation, we implement two different approaches. First, we use the already described classes within the package `EVALUATION.CONF`. The classes are named using the schema *Evaluation_[identifier]_[year]* and include the optimization approaches that should be evaluated, the required problem generator, the number of analyzed basic entities, and the values of the input parameter, e. g., capacity constraints. Furthermore, each of these evaluation classes contains a *main* method to execute the evaluation. Using this implementation, we subsequently generate problem instances. For each single problem instance, we run all desired optimization approaches and finally store their results. Afterwards, we generate the next problem instance and repeat the process. This procedure is very resource efficient regarding required storage capacity and beneficial for a low number of optimization approaches to be evaluated.

However, to assess various heuristics with a multitude of different configurations it is more favorable to use the same problem instances repeatedly. By doing so, the solution of the exact optimization approach only needs to be calculated once and heuristics with various configurations can quickly be analyzed. Therefore, we generate test cases and related problem instances which we store in *source* databases (cf. Section 4.4.1.3).

Furthermore, to minimize the influence of external factors while running an evaluation, we disable – as far as possible – all unnecessary background processes, such as software updates, since these processes may impact the measurement of the computation time. Additionally, we generate executable files that are runnable outside our Eclipse development environment. These files are started via command line whereby the desired optimization approaches and configurations are determined using configuration files and additional command line instructions. These configuration files also define the source database and the result database to log the performance measures of the approaches. To implement these two improvements, we add the following two packages:

- **GENERATOR.DATABASE:** This package encompasses all classes to generate problem instances and store them within a database. Furthermore, it includes classes to load information from the source databases and re-create the problem instances so that they can be processed by the optimization approaches.
- **RUNNER:** This package features all classes to run the evaluation through executable *jar* files.

Apart from our own source code, we also require external components, which are described in the subsequent selection.

4.4.1.2 External Libraries

Apart from our own source code, external libraries are required. For solving the exact and relaxed problem we use CPLEX¹, a commercial optimization engine for solving such problems expressed as mathematical programming models [91]. Its first version was released in 1988 by Robert Bixby as an implementation of the simplex algorithm [20]. In 2009, IBM acquired the software company ILOG and thus CPLEX became a part of it [89]. For our evaluation we use version 12.5 which was released in 2012. At the time of writing, version 12.7 is available. For reasons of comparability with our former publications, we use version 12.5 in this work.

To use CPLEX, we rely on the Java ILP interface². This generic interface supports various free and commercial solver frameworks and, thus, offers the means to switch easily between them without adjusting the source code. Java ILP is freeware, distributed under the terms of the GNU Lesser General Public License (LGPL). For our implementation we use the most recent version 1.2a released in 2011.

Furthermore, several databases are required for our evaluation (cf. Section 4.4.1.3). We use the relational database management system MySQL. To connect the database, we use the JDBC driver *MySQL Connector/J*³, which is provided under the GNU General Public License (GPL). We use this connector in version 5.1.22.

Finally, to import the data of the US Census (cf. Section 4.4.2), a library for parsing delimited text data from files is required. For that purpose, we use the *Opencsv* library⁴, which is provided under the terms of the Apache 2.0 license. For our implementation, we use version 3.8.

4.4.1.3 Source and Result Databases

Apart from the previously described components we require a database system that stores the generated problem instances and the results of our evaluation. Therefore, we use MySQL⁵ in version 5.6.20. The first group of databases provides the generated test cases and the related problem instances. These databases are named `CDCSP_DATA_[IDENTIFIER]`. With the *identifier*, we separate between different source datasets. These databases contain tables which describe the test case and the problem instance: the table `TEST_CASE` includes the test case name, a unique identifier, and the quantity of each *basic entity*, i. e., the number of data centers, user cluster, services, and QoS attributes. The table `PROBLEM` stores the problem

¹ <http://www-03.ibm.com/software/products/de/ibmilogcpleoptistud>

² <https://sourceforge.net/projects/javailp/files/>

³ <https://dev.mysql.com/downloads/connector/j/>

⁴ <http://opencsv.sourceforge.net/>

⁵ <https://www.mysql.com/>

name, a unique identifier, and the test case identifier to connect a test case with its corresponding problem instances. Regarding the base entities and their parameters, we further use the following tables to store all required information: DATA_CENTER, USER_CLUSTER, QOS_ATTRIBUTE, QOS_GUARANTEE, QOS_REQUIREMENT, SERVICE, and SERVICE_DEMAND. The values within these source databases are generated using the class *CDCSPGenerator*. In conjunction with classes from the *generator* package, problem instances are generated based on predefined configurations as described in Section 4.4.2. For the work at hand we use two different source databases. A *learning* database to evaluate the configurations of the heuristics as described in Section 4.4.3 and an *evaluation* database, which is used to evaluate the heuristics in their final configurations and compare them against each other (cf. Section 4.4.4).

The second group of databases, CDCSP_LOGS_[IDENTIFIER], is used to store the evaluation results. These databases consist of three tables each. As described in [103], we also use the tables TEST_CASES and PROBLEMS. The first one stores information about the test case, e. g., the test case name and the identifier, which correspond to the values stored in the source databases. The second table stores information about the evaluated problem instances. This comprises the problem identifier, the used algorithm, and the results, i. e., total cost and computation time. Apart from that, we use an additional table, *assignments*, which stores the final results of the approaches, i. e., the numeric values of the decision variables $y_{d_\lambda, u_\mu, s_\nu}$. Thereby, we are able to analyze the results in detail, e. g., the changes in each iteration of our tabu search algorithm. However, storing these results requires a lot of computational effort and I/O operations. Hence, it influences the measurements regarding the computation time. Consequently, this table is not used for the performance evaluation. Generally, for the whole evaluation in this thesis, only one logging database is required. Nevertheless, we use different ones to process and compare the results with less computational effort.

4.4.2 Experimental Setup

In this section, we present our experimental setup. With a proper design, we efficiently aim to conduct meaningful statements regarding our solution approaches. For our setup, we need to define the number of basic entities to be analyzed and suitable values for all input parameters. Furthermore, we need to determine the variables that measure the performance of our algorithms. In this context, literature separates between independent and dependent variables. Independent variables are those that are controlled by a researcher [127]. Those variables are factors that influence the outcome of an experiment [95]. The configuration of a system and the workload applied during the experiments are dependent variables. They are characterized by their *level*, i. e., their assigned values.

In contrast, dependent variables represent the measured outcome of an experiment or the performance of a system [127]. Consequently, Jain [95] denote this type of variables as *responsive variable*. Regarding the analysis of heuristic approaches, Silver [161], proposes two major measures for the outcome, i. e., the performance of the algorithms: (i) the objective function value of a heuristic compared to the one of the optimal solution and (ii) the computational effort required to calculate the respective value.

Within our optimization model the independent variables are the basic entities and their parameters. To derive statements with theoretical and practical relevance, we aim to use realistic values for all required parameters. When determining appropriate test cases, we also have to consider the computational effort associated with the calculation of the exact solution. As stated earlier, our problem features exponential time complexity in the worst case and, thus, is only applicable for small problem instances. Without an exact solution, we cannot derive clear statements about the solution quality of an analyzed heuristic. Therefore, we have to limit the number of analyzed basic entities to appropriate values, which then enable us to draw conclusions about larger test scenarios.

To generate appropriate test scenarios, we have to define the number of basic entities and the values of the corresponding parameters. As described in the *formal notations* (cf. Section 4.2.1), the model consists of four basic entities: (i) data centers, (ii) user clusters, (iii) services, and (iv) QoS attributes. Regarding the number of data centers, we rely on data provided by large cloud providers. Google discloses 15 data center locations [62] and Amazon offers 42 availability zones, i. e., isolated data center locations, in 16 regions [4]. Therefore, we analyze test cases that contain 10 to 50 data centers. Regarding user clusters and services, we assume a broad range of values to evaluate their impact on the performance of the heuristic algorithms. Their practical characteristics largely depend on the application scenario and the service demand. Thus, no common assumptions about their values exist. A possible way to cluster users or determine the number of potential user clusters is to rely on statistical data about population in metropolitan areas, e. g., large cities in the United States [179]. To identify correlations between independent and dependent variables, we analyze test cases that contain between 100 and 500 user clusters. Depending on the non-functional requirements, several services have to be distinguished. Here, we use between one and four services that differ in their QoS requirements. An overview of the considered number of basic entities for different test cases is given in Table 2. Within the evaluation we use different combinations of these independent variables, which we describe in detail in the respective sections.

In our model, data centers are characterized by their capacity, costs, and QoS guarantees. Regarding the capacity, online sources mention different values, for instance, a capacity between 50,000 and 80,000 servers in case of Amazon [34]. Although Google does not release an official number of their data center capacities

Table 2: Independent variables along with their assumed values.

Variable	Values
Number of data centers d_A	10, 20, 30, 40, 50
Number of user clusters u_M	100, 200, 300, 400, 500
Number of services s_N	1, 2, 3, 4

[39], its largest data centers have an estimated capacity of over 400,000 servers [140]. Such huge amounts of servers per data center may be technically possible but not desired in all cases. For example, Amazon prefers data centers with a total amount of servers less than 100,000, since data centers are seen as single units of failure [128]. For our evaluation, we determine the maximum possible capacity of a data center randomly, using the recommended values in [34]. We set the upper bound between 50,000 and 80,000 servers. The minimal capacity is set to 10% of the selected maximal capacity. Regarding the costs for providing such cloud resources, as well, there is no common agreement in literature (cf. Section 2.2.1). As presented in Section 2.2.1, distinct sources assume different costs for infrastructure, IT equipment, and operational expenses. Therefore, we are required to make assumptions to set values for the cost variables in our model, i. e., for the fixed and variable costs. We set their values based on the cost estimations stated in [101, 189]. These publications estimate the fixed costs, including IT equipment to account for a share of about two-thirds of the total costs. The remaining third is accounted for variable cost. Furthermore, we normalize the estimated cost to a period of one year. Based on the figures given in literature (cf. Section 2.2.1), we assign accumulated annual fixed cost of between 700 and 800 dollars (including the required data center infrastructure) and variable cost between 300 and 350 dollars per server. For the fixed cost we randomly select one value out of the given range and multiple it with the average capacity of the data center. Equation 4.15 shows the calculation of the fixed cost for one data center, whereby $C_{d_\lambda}^{\text{fix_rnd}}$ is a random value out of the cost range between 700 and 800 dollars.

$$C_{d_\lambda}^{\text{fix}} = \frac{K_{d_\lambda}^{\text{max}} + K_{d_\lambda}^{\text{min}}}{2} \times C_{d_\lambda}^{\text{fix_rnd}} \quad (4.15)$$

The randomly chosen value for the variable costs is used without further modifications. For both, fixed and variable costs, we use random values to reflect local price differences, e. g., for property leases, electricity prices, or labor force. Although these assumptions may reflect realistic values, absolute values play only a minor

role in our evaluation, since we compare cost ratios. These are calculated by dividing the objective value of a heuristic by that of the exact approach.

The last characteristics of data centers within our model are QoS guarantees. Due to readability, we describe this part along with the QoS requirements of the user cluster later on in this section. Apart from these parameters, each data center is assigned a discrete location. This location is randomly chosen based on the US Census⁶ dataset. This census was conducted in 2010 and provides – among other data – information about population and median income on county level. Furthermore, GPS coordinates for each county are given, which we use as coordinates for potential data center locations.

The user clusters are characterized by their service demand V_{u_μ, s_v} and their QoS requirements $Q_{u_\mu, s_v, q_\xi}^{req}$. In our model, the service demand is expressed in server units. This demand may be requested by higher level SaaS providers. As an approximation for the demand we randomly select counties based on the US census and use the population and the median income as an indicator for the service demand. We use the medium income as an influencing factor, since we assume that wealthier citizens are more likely to use cloud services. A user cluster states demand for specific *services*, which constitutes the third basic entity in our model.

The last of the four basic entities are the QoS attributes. These QoS attributes are non-functional characteristics of provided services (cf. Section 2.3). In contrast to the other basic entities, the QoS attributes do not influence the number of decision variables. They are part of the constraints and restrict the number of data centers which are able to provide resources to certain user clusters. Thereby, this parameter does not affect the solution space and, thus, the complexity of the problem. Therefore, we focus on a single QoS criterion, i.e., latency. This value depends on various factors, such as distance, network topology, and used transmission medium between provided resources and a user [31]. Assuming efficient routing algorithms, a large geographical distance is the most important factor that significantly influences the latency and excludes data centers in distant regions from multimedia service provisioning [104, 105]. Therefore, in our model, we approximate the latency as a function of the distance between a data center and a user cluster.

Furthermore, to consider the processing time within a data center, we add a random value between 15 and 25 milliseconds. These values are based on the assumption that 20 milliseconds are required for processing and rendering cloud gaming content [30]. The resulting sum of network and computation delay is the QoS guarantee $Q_{d_\lambda, u_\mu, q_\xi}^{guar} \in \mathbb{R}^+$ a data center can provide to a specific user cluster regarding the latency. The corresponding requirement $Q_{u_\mu, s_v, q_\xi}^{req} \in \mathbb{R}^+$ is then defined depending on user cluster and service. We set this requirement for the first

⁶ <https://www.census.gov/2010census/>

service of each user cluster within a problem instance to 100 milliseconds referring to gaming [35]. For all subsequently generated services we randomly select a value between 60 milliseconds and 200 milliseconds to roughly reflect the requirements stated in literature (cf. Section 2.3).

Using the preciously described independent variables, the parameters, and their respective values, we generate 100 problem instances for each test case. We store these instances in a source database (cf. Section 4.4.1.3) and, thus, the same problem instances are available for various experiments. We conduct all evaluations using a workstation equipped with an Intel Xeon CPU E5-1620 v3 with 3.50 GHz and 16 GB of memory, operating under Microsoft Windows 7.

4.4.3 Configurations of the Heuristic Approaches

In Section 4.3 we presented heuristic approaches for the CDCSP. In the following, we evaluate the influence of their configuration parameters on the achieved solution quality. In Section 4.4.3.1, we describe the evaluation of the priority-based start heuristic and the assessment of the best combination of involved rules. Subsequently, we analyze the Best-of-Breed approach regarding the included heuristics in Section 4.4.3.2. Finally, in Section 4.4.3.3, we evaluate the relevant parameters for the tabu search heuristic.

4.4.3.1 Priority-based Start Heuristics

In Section 4.3.2.2, we described various priority and cost allocation rules to configure different priority-based start heuristics. By means of these heuristics, we assign user clusters to appropriate data centers following a defined sequence. Therefore, we use priority rules to order the user clusters and cost allocation rules to calculate the unit cost per data center and, thus, determine an appropriate order for these entities. Regarding these rules and their possible combinations, within this section, we address the following questions:

- Which rule combination delivers the best solution quality?
- Which category, priority or cost allocation rule influences solution quality and computation time the most?
- How does a two-stage process perform compared to a one-stage process?

To address these questions, we first compare all priority rules against each other in a one-stage process. Therefore, we use a single cost allocation rule, i. e., the *Med Capacity Cost Allocation Rule* (C_3) in all combinations to analyze only the influence of the priority rules. For better readability, Table 3 summarizes the rules presented in Section 4.3.2.2.

Table 3: Overview of priority and cost allocation rules.

Priority rules		Cost allocation rules	
ID	Rule	ID	Rule
P ₁	Demand Priority Rule	C ₁	Max Capacity Cost Allocation Rule
P ₂	Capacity Priority Rule	C ₂	Min Capacity Cost Allocation
P ₃	Buffer Priority Rule	C ₃	Med Capacity Cost Allocation Rule
		C ₄	No Fixed Cost Allocation Rule
		C ₅	Current Utilization Cost Allocation Rule
		C ₆	Prefer Minimal Utilization Cost Allocation Rule
		C ₇	Penalize First Cost Allocation Rule

We analyze a set of six test cases based on the values in Table 2. For each entity we use the smallest and largest value, while keeping the values of the remaining entities constant, which results in the following test cases (no. data center / no. user cluster / no. services): 10/300/3; 50/300/3; 30/100/3; 30/500/3; 30/300/1; 30/300/4. Table 4 shows the results regarding the solution quality, i. e., the ratio of cost between the evaluated approach and the exact solution (rounded to four digits). Clearly recognizable, the *Buffer Priority Rule* (P_3) outperforms the other two approaches regarding this performance indicator. Furthermore, the paired t-tests between the *Buffer Priority Rule* (P_3) and the other two shows a statistical significant difference for all test cases.

Regarding the computation time presented in Table 5, there are two clear results: First, the *Demand Priority Rule* (P_1) outperforms the other two approaches and is the most efficient approach regarding computational effort. Second, the *Buffer Priority Rule* (P_3) outperforms the *Capacity Priority Rule* (P_2) regarding both performance measures, i. e., solution quality and computation time.

Since we primarily focus on solution quality, we proceed with this best priority rule and now only vary the cost allocation rules. The results (rounded to four digits) are shown in Table 6. For better readability we highlight the best solution quality for each test case. In most cases, the heuristic using the *Current Utilization Cost Allocation Rule* (C_5) or the *Penalize First Cost Allocation Rule* (C_7) achieve the best results. Furthermore, the paired t-test between the two heuristics using these rules shows no statistical significant differences regarding the solution quality.

Both rules are based on the same principle that additional data centers cause disproportionately high unit costs, which is consequently best practice approach for the CDCSP. Thus, to answer the first question, the combination of the *Buffer Priority Rule* (P_3) with one of these two cost allocation rules results in the best single-stage approaches.

Table 4: Solution quality using different priority rules and a fixed cost allocation rule. The highlighted combination achieves the best results for all test cases.

	10 / 300 / 3	50 / 300 / 3	30 / 100 / 3	30 / 500 / 3	30 / 300 / 1	30 / 300 / 4
Heuristic [P1] [C3]	1.6724	1.2299	1.2741	1.3937	1.1516	1.4257
Heuristic [P2] [C3]	1.6327	1.1871	1.2108	1.3304	1.1233	1.3446
Heuristic [P3] [C3]	1.4776	1.0892	1.1245	1.1905	1.0669	1.2033

Table 5: Average computation time in seconds using different priority rules and a fixed cost allocation rule. The highlighted combination achieves the best results for all test cases.

	10 / 300 / 3	50 / 300 / 3	30 / 100 / 3	30 / 500 / 3	30 / 300 / 1	30 / 300 / 4
Heuristic [P1] [C3]	0.0550	0.0612	0.0178	0.1408	0.0171	0.0927
Heuristic [P2] [C3]	0.1505	0.3084	0.0392	0.6531	0.0431	0.5054
Heuristic [P3] [C3]	0.0681	0.1120	0.0198	0.2343	0.0197	0.1493

Table 6: Solution quality using a fixed priority rule and varying cost allocation rules. The highlighted cells show the best result per test cases.

	10 / 300 / 3	50 / 300 / 3	30 / 100 / 3	30 / 500 / 3	30 / 300 / 1	30 / 300 / 4
Heuristic [P3] [C1]	1.4754	1.0926	1.1343	1.2055	1.0726	1.2038
Heuristic [P3] [C2]	1.4541	1.1290	1.1443	1.1993	1.1125	1.2205
Heuristic [P3] [C3]	1.4776	1.0892	1.1245	1.1905	1.0669	1.2033
Heuristic [P3] [C4]	1.4823	1.0990	1.1569	1.2130	1.0901	1.2236
Heuristic [P3] [C5]	1.1075	1.0740	1.0890	1.0872	1.0951	1.0891
Heuristic [P3] [C6]	3.3517	4.2624	3.9426	4.0726	4.1891	4.3762
Heuristic [P3] [C7]	1.0949	1.0739	1.0891	1.0872	1.0951	1.0891

Table 7: Average computation time in seconds using a fixed priority rule and varying cost allocation rules. The highlighted cells show the best result per test cases.

	10 / 300 / 3	50 / 300 / 3	30 / 100 / 3	30 / 500 / 3	30 / 300 / 1	30 / 300 / 4
Heuristic [P ₃] [C ₁]	0.0667	0.1092	0.0206	0.2326	0.0193	0.1497
Heuristic [P ₃] [C ₂]	0.0667	0.1152	0.0201	0.2265	0.0199	0.1537
Heuristic [P ₃] [C ₃]	0.0681	0.1120	0.0198	0.2343	0.0197	0.1493
Heuristic [P ₃] [C ₄]	0.0659	0.1132	0.0198	0.2337	0.0208	0.1482
Heuristic [P ₃] [C ₅]	0.0631	0.1118	0.0196	0.2279	0.0197	0.1480
Heuristic [P ₃] [C ₆]	0.0735	0.1557	0.0251	0.3063	0.0250	0.2012
Heuristic [P ₃] [C ₇]	0.0658	0.1107	0.0205	0.2221	0.0195	0.1474

In the next step, we analyze the influence of the cost allocation rules regarding the computation time. The results, stated in Table 7, generally do not show a correlation between the used cost allocation rule and the computational effort. The fastest heuristics (in terms of average computation time) are highlighted with a grey background. Comparing the two approaches with the best solution quality (*Heuristic [P₃][C₅]* and *Heuristic [P₃][C₇]*), it is notable that the *Heuristic [P₃][C₇]*, relying on the *Penalize First Cost Allocation Rule*, on average has lower computation times for most test cases (the best results out of these two heuristics are highlighted with bold font). Nevertheless, the paired t-test shows that the heuristic including this rule is significantly better for the test cases 30/100/3 and 30/500/3, but worse for the test case 10/300/3. For the remainder of the test cases, the t-test shows no differences. Since there is generally no clear difference, we consider both heuristics as equally well suited.

Still, for the evaluation of the two-stage approach, we use the *Penalize First Cost Allocation Rule* since it features at least in average a better computational efficiency. We combine it with the *Buffer Priority Rule* (*P₃*) for the first stage. For the second stage, we use all possible combinations of priority and cost allocation rules to evaluate the influence on the final performance. Table 8 shows the results from one test case, whereby the following common statements are true for all other test cases:

- We are able to improve the average solution quality for most test cases using a second stage.
- All rule combinations that achieve the average best solution quality include the *Buffer Priority Rule* (*P₃*) in both stages.
- For all test cases, using the *Max Capacity Cost Allocation Rule* for the second stage results in the best or second best average solution quality. Furthermore, it is most often part of the best heuristic approach. This result is well comprehended by the *Penalize First Cost Allocation Rule*.

hensible: even assuming a good solution quality as a result of the first stage, a low number of data centers with a high utilization was chosen. Using this selection as input for the second stage, a cost allocation rule calculating the unit price based on very high utilization most likely achieves the best results.

- In general, the computation time increases by between 0.02% to up to 25.02% in the worst case using a second stage compared to using only one stage.

We additionally evaluate the *Current Utilization Cost Allocation Rule (C5)*, which is also appropriate for the first stage. Using this rule within the first stage, we achieve similar results as already stated in the list above for the rules in the second stage. With these final experiments, we answer our third question: A two-stage approach always leads to equal or better solution quality, at the cost of higher computational effort. As the result of this evaluation procedure, we determine the heuristic with the following rule combination as the best with respect to the solution quality:

- Rules for the first stage: *Buffer Priority Rule (P₃)* and *Penalize First Cost Allocation Rule (C₇)*
- Rules for the second stage: *Buffer Priority Rule (P₃)* and *Max Capacity Cost Allocation Rule*.

This priority-based start heuristic is part of the final evaluation (comparing all heuristics against each other) and is further used to determine the initial solution for the tabu search heuristic as discussed later in Section 4.4.3.3

4.4.3.2 Best-of-Breed Approach

The Best-of-Breed approach combines different single heuristic approaches, executes them subsequently, and selects the best out of these single runs. As described in Section 4.3.3, we include selected priority-based start heuristics within the BoB approach. Our goal is to achieve better results than obtained by a single start heuristic while maintaining a favorable computation complexity. The challenge lies here in the selection of the best fitting heuristics to achieve this goal.

Here, we follow two different approaches: (i) we use the results of the previous Section 4.4.3 and (ii) the statistical selection process described in Section 4.3.3. As result of the evaluation of the two stage heuristic approaches, we found three rule combinations that result in the best solution qualities for the evaluated test cases.

- Rules on stage one: *Buffer Priority Rule (P₃)* and *Penalize First Cost Allocation Rule (C₇)*; rules on stage two *Buffer Priority Rule (P₃)* and *Max Capacity Cost Allocation Rule*
- Rules on stage one: *Buffer Priority Rule (P₃)* and *Penalize First Cost Allocation Rule (C₇)*; rules on stage two *Buffer Priority Rule (P₃)* and *Penalize First Cost Allocation Rule (C₇)*

Table 8: Comparison of the one-stage approach against two-stage approach for test case $|D|=50/|U|=300/|S|=3$, with 95% confidence intervals. The results are ordered by solution quality.

Heurisitc	Cost ratio		Comp. time [s]	
	Average	Conf.	Average	Conf.
Heuristic [P ₃] [C ₇] - [P ₃] [C ₁]	1.0718	0.0049	0.1384	0.0035
Heuristic [P ₃] [C ₇] - [P ₃] [C ₃]	1.0720	0.0047	0.1167	0.0028
Heuristic [P ₃] [C ₇] - [P ₃] [C ₄]	1.0724	0.0047	0.1339	0.0026
Heuristic [P ₃] [C ₇] - [P ₂] [C ₁]	1.0731	0.0049	0.1907	0.0054
Heuristic [P ₃] [C ₇] - [P ₃] [C ₂]	1.0735	0.0049	0.1296	0.0026
Heuristic [P ₃] [C ₇] - [P ₂] [C ₄]	1.0736	0.0048	0.1847	0.0049
Heuristic [P ₃] [C ₇] - [P ₁] [C ₁]	1.0737	0.0048	0.0921	0.0024
Heuristic [P ₃] [C ₇] - [P ₂] [C ₇]	1.0737	0.0048	0.1762	0.0053
Heuristic [P ₃] [C ₇] - [P ₂] [C ₂]	1.0737	0.0048	0.1938	0.0050
Heuristic [P ₃] [C ₇] - [P ₁] [C ₆]	1.0737	0.0048	0.0910	0.0021
Heuristic [P ₃] [C ₇] - [P ₁] [C ₄]	1.0737	0.0048	0.0905	0.0026
Heuristic [P ₃] [C ₇] - [P ₂] [C ₅]	1.0737	0.0048	0.1715	0.0044
Heuristic [P ₃] [C ₇] - [P ₁] [C ₇]	1.0738	0.0048	0.0911	0.0026
Heuristic [P ₃] [C ₇] - [P ₂] [C ₆]	1.0738	0.0048	0.2017	0.0040
Heuristic [P ₃] [C ₇] - [P ₁] [C ₃]	1.0738	0.0048	0.0911	0.0021
Heuristic [P ₃] [C ₇] - [P ₂] [C ₃]	1.0738	0.0048	0.1905	0.0055
Heuristic [P ₃] [C ₇] - [P ₁] [C ₅]	1.0738	0.0048	0.0884	0.0020
Heuristic [P ₃] [C ₇] - [P ₁] [C ₂]	1.0738	0.0048	0.0922	0.0027
Heuristic [P ₃] [C ₇] - [P ₃] [C ₇]	1.0738	0.0048	0.1103	0.0021
Heuristic [P ₃] [C ₇] - [P ₃] [C ₅]	1.0738	0.0048	0.1310	0.0029
Heuristic [P ₃] [C ₇]	1.0739	0.0048	0.1107	0.0028
Heuristic [P ₃] [C ₇] - [P ₃] [C ₆]	1.0739	0.0048	0.1151	0.0027

- Rules on stage one: *Buffer Priority Rule (P₃)* and *Penalize First Cost Allocation Rule (C₇)*; rules on stage two *Buffer Priority Rule (P₃)* and *No Fixed Cost Allocation Rule*

Utilizing these three heuristics we are able to cover all previously calculated datasets with the best solution quality. With the second approach for selecting suitable heuristics, we consider solution quality and computational effort as being equally important. Therefore, we use 16 different test cases based on the evaluation published in [74] with varying numbers of data centers and user clusters. Here, we evaluate all possible rule combinations (excluding C₄ and C₆ in the first stage) for each test case. For each heuristic and test case, we run 100 problem instances, which leads to 504,000 result sets. Subsequently, for each test case and heuristic we aggregate the datasets by calculating the mean values of cost ratio and total computation time.

Further, for each test case, we select the single heuristic with the average best solution quality and compare it with the results of all other heuristics by the means of the *paired two-sample t-test* (cf. Section 4.3.3). Thus, for each single test case we determine a group of *best* heuristics, whereby all approaches – in a statistical sense – have the same solution quality. Afterwards, for each single test case, we use this set of heuristics with the best solution quality and determine the fastest one out of this set. Subsequently, again using the *paired two-sample t-test*, we compare it with the remaining heuristics regarding the statistical significance of the difference in computation time. This procedure reduces the number of appropriate heuristics further to those that are the best regarding the solution quality *and* computation time. Finally, out of all heuristics, we select those that cover all test cases with the lowest number of included heuristics.

Table 9 shows the final results of this selection process. The rows list the best heuristics and the columns show the evaluated test cases. The value 1 indicates whether a heuristic is suitable for a test case regarding the above described analysis. The finally chosen heuristics are highlighted by a colored background.

Table 9: Overview of the best heuristics and the selection results.

	10 / 50	10 / 100	10 / 150	10 / 200	20 / 100	20 / 200	20 / 300	20 / 400	30 / 150	30 / 300	30 / 450	30 / 600	Sum per heuristic
Heuristic [P ₃] [C ₃] - [P ₃] [C ₄]	1		1		1	1	1	1	1		1		8
Heuristic [P ₃] [C ₁] - [P ₃] [C ₄]	1			1		1		1	1	1		1	7
Heuristic [P ₃] [C ₃] - [P ₃] [C ₇]	1					1	1	1	1		1	1	7
Heuristic [P ₃] [C ₁] - [P ₃] [C ₇]	1				1	1	1	1					6
Heuristic [P ₃] [C ₃] - [P ₃] [C ₅]	1					1	1	1			1	1	6
Heuristic [P ₃] [C ₃] - [P ₃] [C ₆]						1	1			1		1	5
Heuristic [P ₃] [C ₁] - [P ₃] [C ₁]						1	1	1				1	4
Heuristic [P ₃] [C ₁] - [P ₃] [C ₅]	1					1	1	1					4
Heuristic [P ₃] [C ₃] - [P ₃] [C ₁]						1	1		1		1		4
Heuristic [P ₃] [C ₃] - [P ₃] [C ₃]						1	1		1		1		4
Heuristic [P ₃] [C ₅] - [P ₃] [C ₃]	1		1				1	1	1				4
Heuristic [P ₃] [C ₁] - [P ₃] [C ₃]						1	1	1					3
Heuristic [P ₃] [C ₁] - [P ₃] [C ₂]						1	1	1					3
Heuristic [P ₃] [C ₁] - [P ₃] [C ₆]							1	1			1		3
Heuristic [P ₃] [C ₃] - [P ₃] [C ₂]							1				1	1	3
Heuristic [P ₁] [C ₃] - [P ₃] [C ₇]	1	1	1										3
Heuristic [P ₃] [C ₅] - [P ₃] [C ₁]	1							1	1				3
...													
Heuristic [P ₁] [C ₅] - [P ₃] [C ₃]		1											1
Sum per test case	13	1	5	1	2	13	14	12	6	1	7	7	82

Figure 4 and 5 show the ratio of cost and the computation time of both BoB configurations. According to the order we name them within the charts *CDCSP-BoB.KOM 1* and *CDCSP-BoB.KOM 2*. We analyze the cost ratio and computation time for a varying number of data centers. Regarding the solution quality, the first approach outperforms the second for each test case. Nevertheless, the difference in solution quality declines with increasing test case size. The difference for the smallest test case (10 / 300 / 3) amounts 6.73 percentage points, while this difference decreases to 3.38 percentage points for the largest evaluated test case.

For the first BoB approach, we include heuristics that rely on more sophisticated but also more complex algorithms, which is clearly recognizable in computation time. The computation time of this approach increases disproportionately compared to the second one. While the difference for the first test case amounts to only about 11 milliseconds, it increases to about 94 milliseconds for the largest one. Thus, for larger test cases the second approach is more favorable regarding the computational effort. Nevertheless, for a global scenario with a realistic number of data centers as discussed in Section 4.4.2, we select the first one as more appropriate. Therefore, use the first configuration in our comparison of all heuristics presented in Section 4.4.4.

4.4.3.3 Tabu Search

In this section, we evaluate important parameters of the tabu search heuristic, to find a configuration that provides the best performance for the given optimization problem. First, we analyze the influence of the initial solution regarding the solution quality. Subsequently, we evaluate the value of the tabu list size and finally, we analyze the influence of the number of iterations regarding the performance.

Initial solution: As already described in Section 4.3.4.1, our tabu search approach requires an initial solution as starting point for its iterative improvement process. To determine a first valid solution, we use a priority-based heuristic as described in Section 4.3.2. However, as shown in Section 4.4.3.1, these start heuristics have different performances, depending on their configurations. In this section, we show the relation between the solution quality of the initial solution and the overall solution quality of the tabu search after a predefined number of iterations. Utilizing these results, we determine a suitable start heuristic for the tabu search. As start heuristics, we use the following two approaches:

- Start heuristic 1: Selection: *Buffer Priority Rule (P₃)*, *Penalize First Cost Allocation Rule (C₇)*; Allocation: *Buffer Priority Rule (P₃)*, *Max Capacity Cost Allocation Rule (C₁)*
- Start heuristic 2: Selection: *Demand Priority Rule (P₁)*, *Max Capacity Cost Allocation (C₁) Rule*; Allocation: *Demand Priority Rule (P₁)*, *Penalize First Cost Allocation Rule (C₇)*

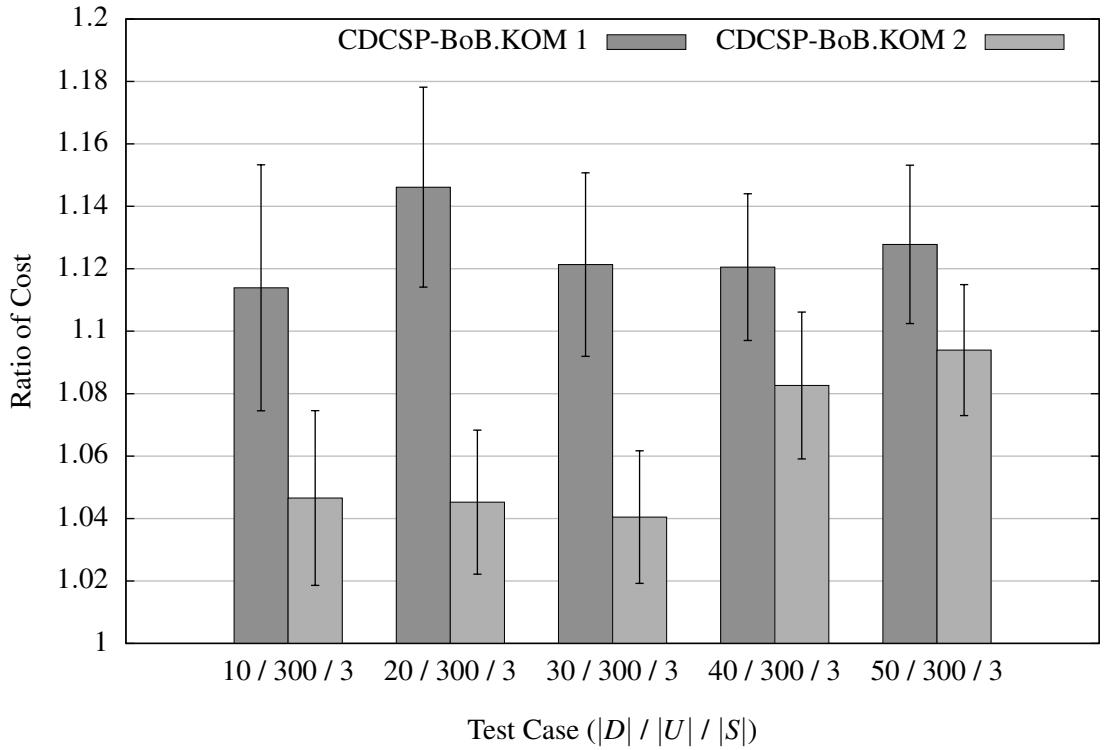


Figure 4: Ratio of costs (based on macro-average; with 95% confidence intervals) between the exact approach and the best-of-breed approaches by test case.

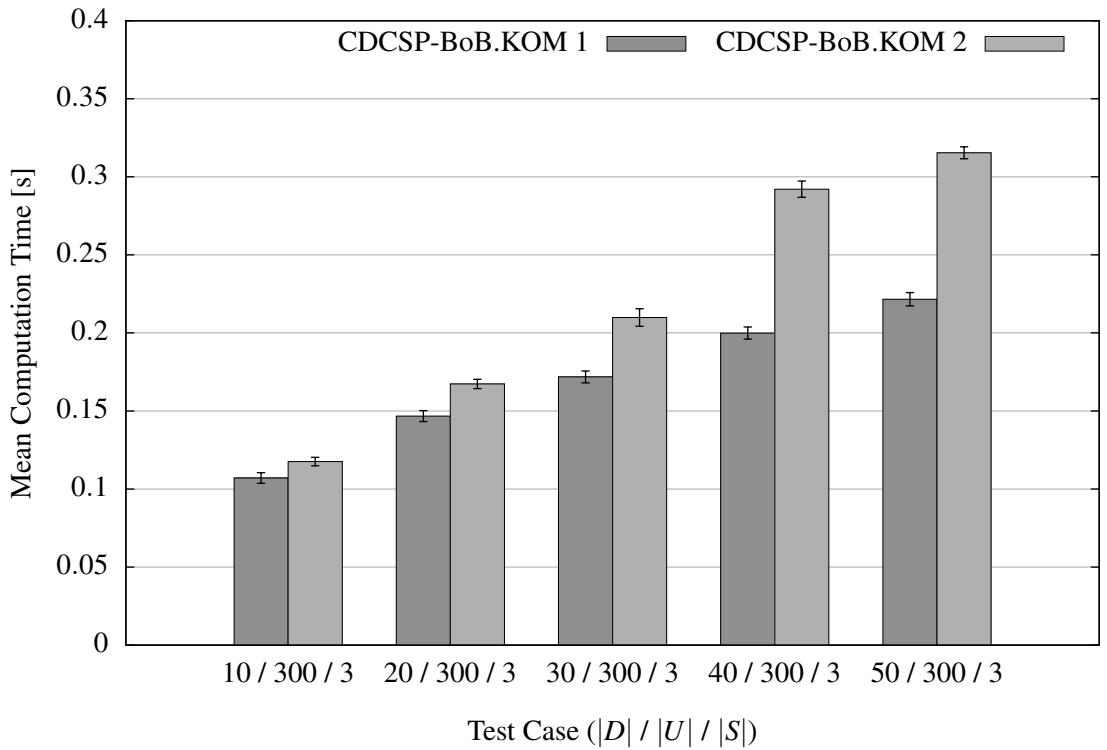


Figure 5: Observed mean computation times (with 95% confidence intervals) by test case.

The first approach is characterized by a better solution quality and higher computational effort compared to the second one. We determined this configuration in Section 4.4.3.1 as the best priority-based heuristic. The second one was proven as a heuristic with minor solution quality but low computational effort in one of our former publications [73]. To evaluate the influence of the overall solution quality, we evaluate the tabu search heuristic with different numbers of iterations. We use the test case with the following amount of basic entities: Data centers $|D| = 30$, user clusters $|U| = 300$, and services $|S| = 3$. Thus, we make sure that the solution space is not completely analyzed after a few iterations.

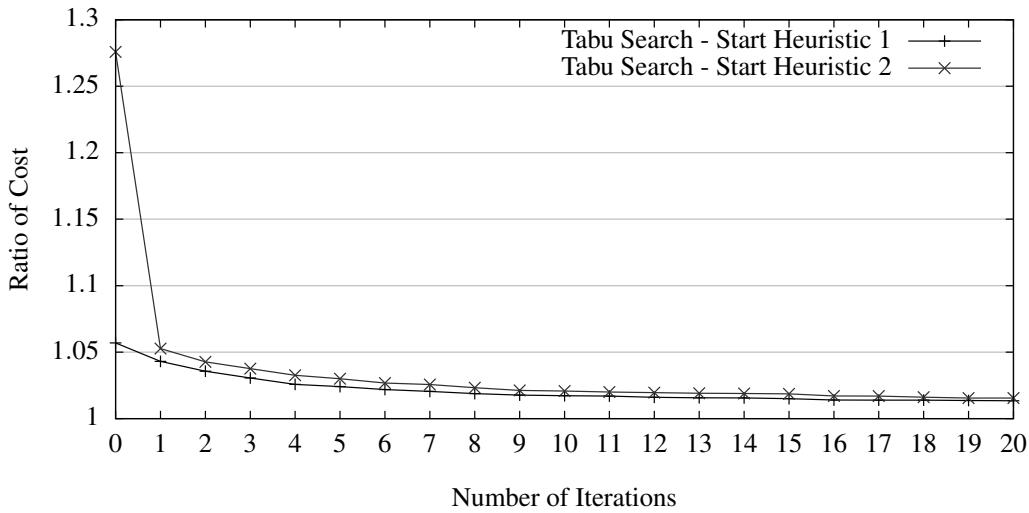


Figure 6: Solution quality of two start heuristics depending on the number of iterations.

As shown in Figure 6, the solution quality of the two initial solutions significantly differs. These initial values are illustrated at *zero* iterations. While the first start heuristic calculates an average objective value that is 5.69% worse than the exact one, the second heuristic causes a difference of 27.86%. The gap between the two start heuristics amounts to 21.9 percentage points at the beginning. The tabu search heuristic is able to quickly reduce this difference. After the first iteration, the difference amounts to only 0.95 percentage points. Finally, after 20 iterations, an average difference of only 0.21 percentage points remains. The reason can be seen in the local search procedure used by the tabu search algorithm.

As the result, it can be stated that the initial solution influences the overall solution quality. The tabu search is able to significantly improve an initial solution even with a very low solution quality. Nevertheless, in average, there remains a small difference in solution quality even after 20 iterations. In the course of this section, we show that the number of iterations substantially influences the computation effort of the tabu search algorithm. Therefore, a good initial solution is beneficial in any case and, consequently, we choose the better heuristic in general for our tabu search approach.

Another major influence parameter of the algorithm is the *tabu list size*. This parameter specifies how many solutions or attributes of solutions are temporarily forbidden. Utilizing this list, we avoid that the algorithm directly returns to a local optimum after switching to a worse solution. As described earlier, a small value for the tabu list size may lead to a cycling behavior; a large value may result in a big amount of solutions that are forbidden. Both alternatives cause a low solution quality (cf. Section 4.3.4.3). Therefore, a suitable value for this parameter is required. As depicted in Figure 7, to assess a proper value, we choose a fixed number for the size of the tabu list between one and twenty. Further, for all experiments we set the number of iterations, i. e., the number of analyzed neighborhoods, to twenty independently of the number of the data centers.

In Figure 7a, the influence regarding the solution quality is illustrated. We measure the solution quality as cost ratio between the tabu search heuristic and the exact solution approach. Thereby, a cost ratio of one indicates that a heuristic achieves the same objective value as the exact approach. Since the tabu list only stores data centers, we focus on this basic entity, i. e., we analyze test cases with a different amount of data centers, while keeping the number of user clusters and services constant. Starting with a tabu list size of one and increasing its value, the cost ratio decreases. Afterwards, with a further increasing value the cost ratio increases again. This holds true for all test cases independently of the number of data centers. Nevertheless, this effect is most notable in the test case with the lowest number of data centers ($|D| = 10$). The achieved solution quality lies by an average value of 1.026 using a tabu list that stores only one value. Using a list size of four, we achieve a solution quality which is only 0.92 percentage points worse compared to the exact approach. Increasing the size of the tabu list further results again in a lower solution quality. In the analyzed evaluation setup, there is no clearly recognizable relation between the number of data centers and the size of the tabu list. For all test cases the best solution quality predominantly is achieved using values between two and four. Thus, a value of three for the tabu list size is a good compromise, since it results in a good solution quality for all test cases. As illustrated in Figure 7b, there is no correlation between the tabu list size and the computation time. Thus, to choose a proper value for this parameter, we only need to consider the solution quality.

The *number of iterations* determines how many neighborhood solutions are analyzed. To find a suitable value for this parameter, we evaluate different configurations with a different amount of iterations from one to twenty. As previously indicated, we fix the tabu list size to three. Figure 8 illustrates the results regarding the performance of the different configurations. In contrast to the tabu list size, this parameter influences both, solution quality and computation time. Consequently, there is no single best value which can be set for this parameter. For example, the cost ratio of test case three ($|D| = 30$) improves by 1.72 percentage points after five

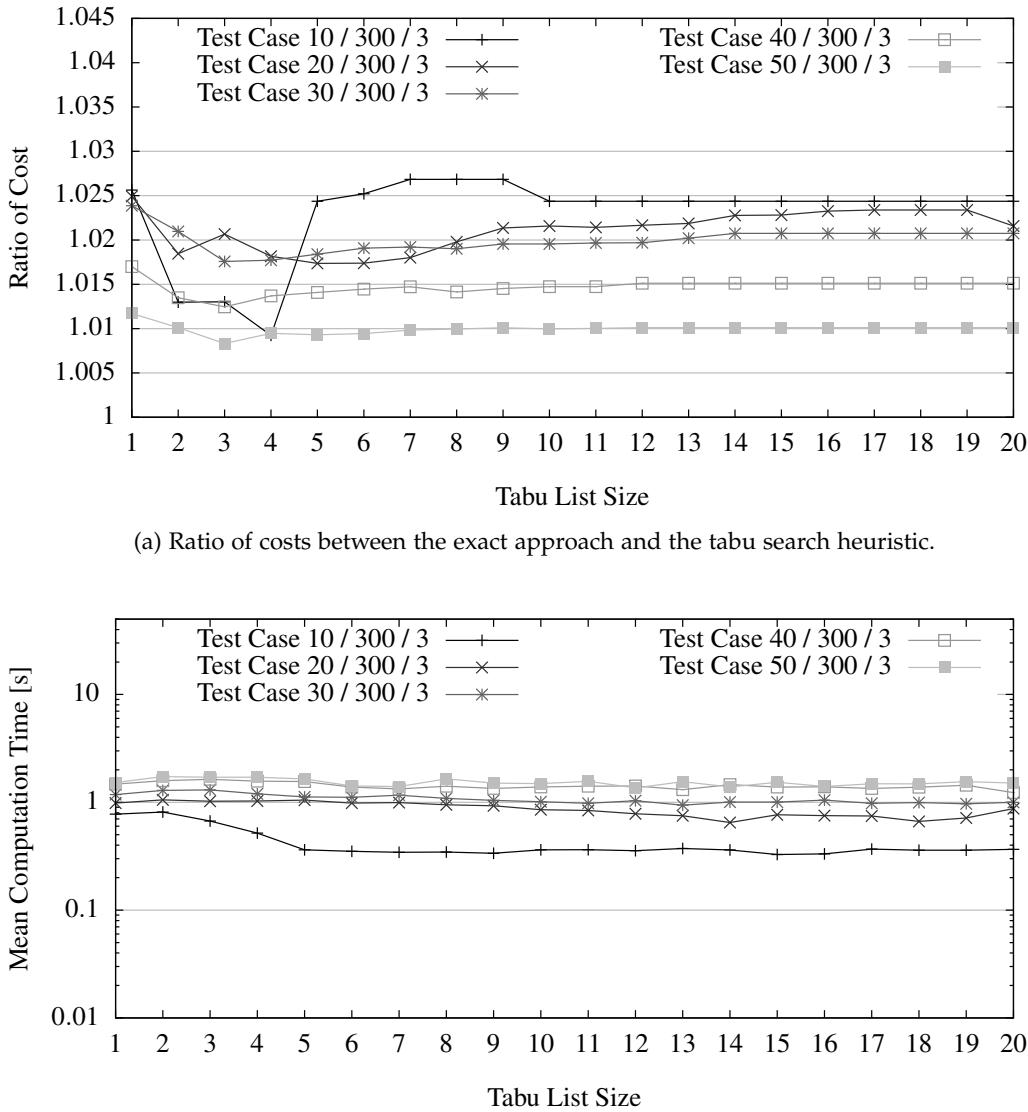


Figure 7: Evaluation results for predefined tabu list sizes.

iterations and by 3.55 percentage points after 20 iterations compared to the first one. Clearly recognizable is the decreasing improvement with an increasing number of iterations. Thus, the first iterations improve the objective value more than the later iterations.

A different illustration regarding the trade-off between solution quality and computation effort is shown in Figure 9. Here, the x-axis describes the ratio of computation time of the tabu search approaches compared to the exact approach. The higher computation time is the result of a larger number of iterations. The y-axis shows the ratio of cost. Because of scaling issues, we use one chart for the first test case ($|D| = 10$) and another one for the remaining test cases. Again, the trade-off between solution quality and computational effort is clearly recogniz-

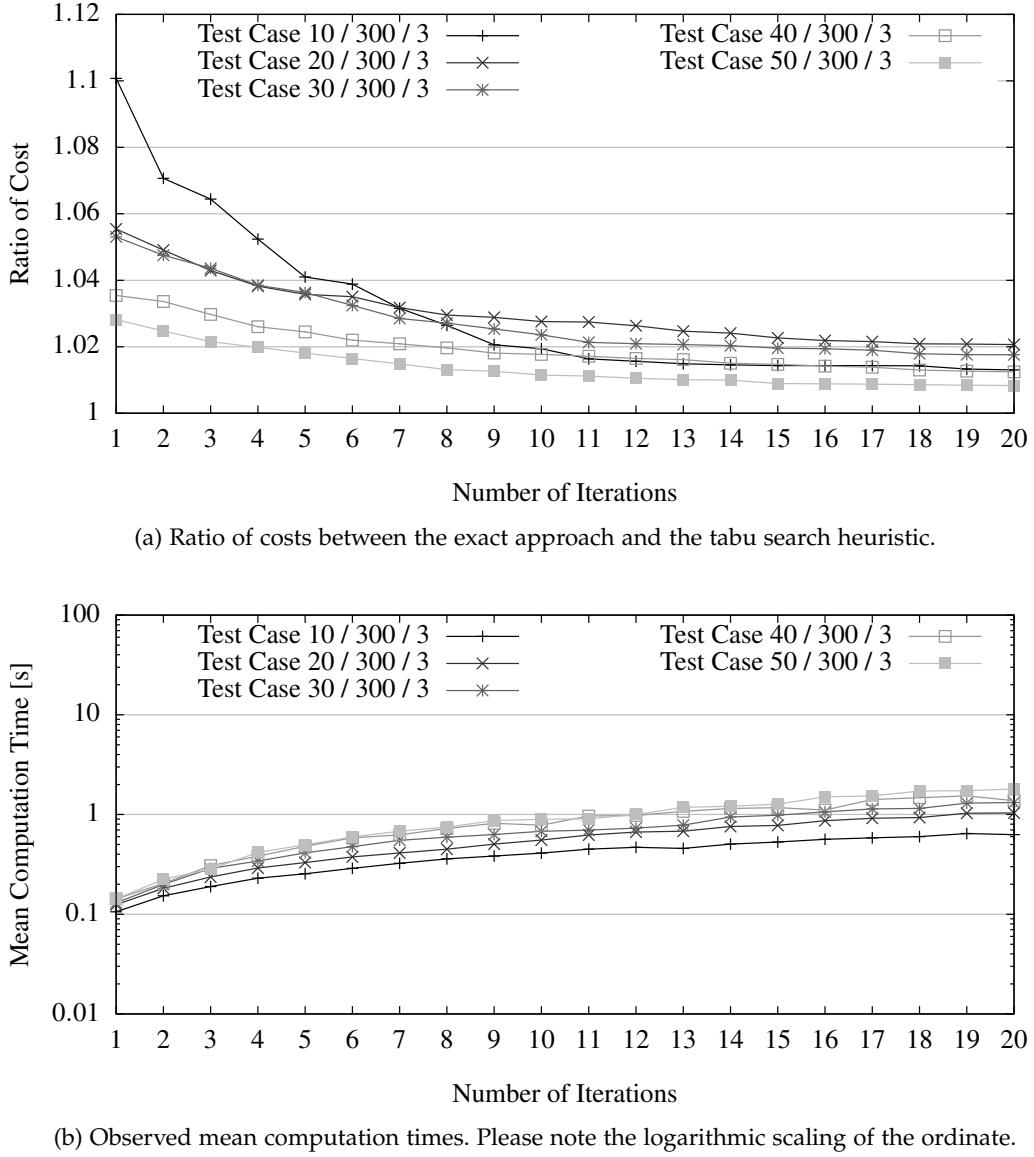
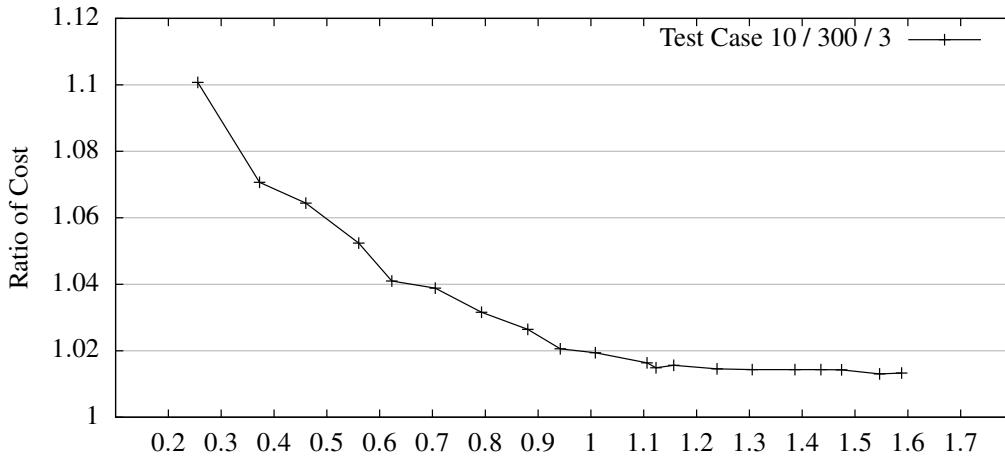


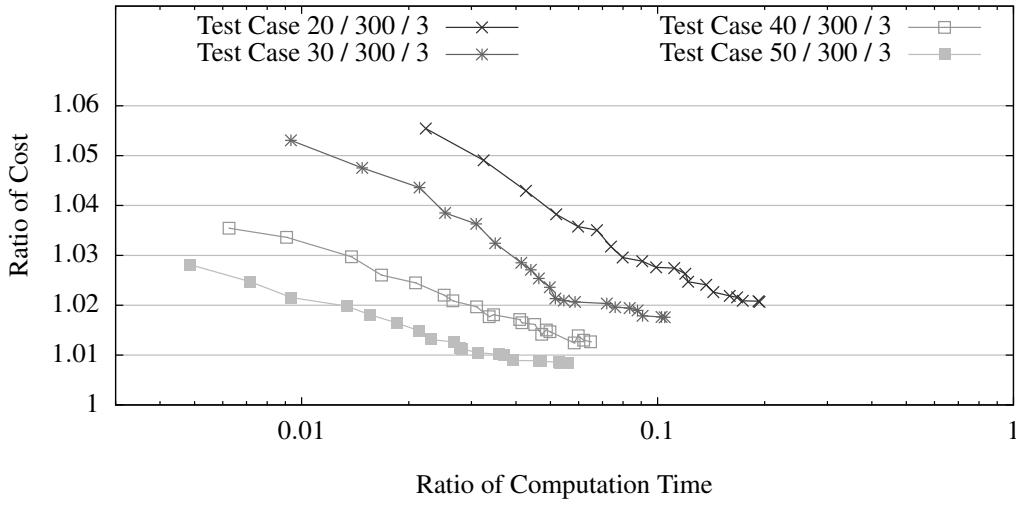
Figure 8: Evaluation results for different numbers of iterations.

able. Additional insights can be gained using these charts. First, even with high computational effort, we are only able to come close to the objective value of the exact approach but cannot achieve it. This confirms our statement given in Section 4.3.4.4 that *finding the optimal solution* cannot be used as a stopping condition for our tabu search approach. Second, an artificial stopping condition is required. As illustrated in Figure 9a, using a high amount of iterations, the computational effort of a heuristic approach may exceed the one of the exact approach, without substantially increasing the solution quality. Third, especially large test cases benefit from the tabu search heuristic regarding solution quality *and* computation time as shown in Figure 9b.



Ratio of Computation Time

(a) Test Case 1.



Ratio of Computation Time

(b) Test Case 2 to 5.

Figure 9: Trade off between ratio of cost and ratio of computation time.

However, for all approaches the computation time increases by a factor of about ten when using 20 iterations instead of one. As illustrated in Figure 9a, with each additional iteration, the average value of the cost ratio decreases, i.e., the solution quality increases. However, the largest improvement is achieved within the first 10 iterations. This behavior is particularly visible for the test case with the lowest number of data centers. With an increasing number of data centers, the number of neighborhood solutions also increases. To enable the algorithm to analyze a larger number of neighborhood solutions, correlating to the number of data centers, we propose the following formula to determine an appropriate number of iterations: $i = \sqrt{|D|} + 7$. This formula, for example, results in 10 iterations for 10 data centers and, thus, achieves high solution quality even for small test cases (cf. Figure 8a).

Table 10: Independent and controlled variables, along with their corresponding values.
Underlined values are the controlled ones that remain constant.

Variable	Values
Number of data centers d_A	10, 20, <u>30</u> , 40, 50
Number of user clusters u_M	100, 200, <u>300</u> , 400, 500
Number of services s_N	1, 2, <u>3</u> , 4

Hence, it considers a relatively high number of iterations for test cases with a low number of data centers and a slow rise for an increasing amount of data centers. Nevertheless, the *best* value for this parameter may also depend on the application scenario. For calculations with time constraints, a smaller value may be more appropriate. For a higher solution quality, a larger number of iterations are more beneficial.

To summarize the evaluation of the tabu search parameters, we can state that the utilization of this improvement procedure is beneficial in any case. Even with only one iteration, we are able to improve the objective value of our optimization problem. Furthermore, we find that a good initial solution influences the overall solution quality. Therefore, we select an approach with more complex priority rules as the starting point of the improvement process. Regarding the tabu list size, we determined *three* as a proper value for this parameter in our scenarios. Finally, we determine a formula to calculate the number of iterations depending on the number of data centers as stated in the previous section. Using these insights, we configure our tabu search approach to finally compare it to all other heuristics in the subsequent Section 4.4.4.

4.4.4 Comparison of all Heuristics

In this section, we present the last part of our evaluation regarding the CDCSP. Here, we compare the presented heuristic approaches in their previously determined best configuration against each other. In Section 4.4.2, we presented the decision variables of our optimization problem, the parameters, and corresponding values. These independent variables are used to derive suitable test cases. We use a fractional factorial experiment design to assess the impact of individual independent variables on the achieved computational efficiency and cost, as proposed in [95]. The remaining independent variables are assumed as fixed, i. e., controlled. An overview of the independent variables along with their values are given in Table 10.

Thereby, we end up with a total number of 12 different test cases. For each test case we generate 100 problem instances, yielding a total of 1,200 individual problem instances within this final evaluation step.

As performance measures we use the cost and the computation time. Since the cost depends to a large extent on the characteristics and specifications of a problem, the total cost is not of significance. Hence, we rely on cost ratios, calculated by dividing the solution of a heuristic approach by the solution of the exact optimization approach. In the subsequent sections we present the evaluation results for individual decision variables.

4.4.4.1 *Data Centers*

First, we analyze the influence of the number of data centers on the performance of our optimization algorithms. As described before, values for this independent variable are selected based on information regarding the most important infrastructure providers within the cloud market.

To begin with, the computation times for all optimization approaches are given in Figure 10a. As expected, the approaches solved utilizing the CPLEX solver framework, i. e., CDCSP-EXA.KOM and CDCSP-REL.KOM, cause the highest computation times. Both require less than 500 milliseconds for the smallest test case with 10 data centers, but the LP-relaxed approach requires the most computation time for this test case. Due to its exponential growth in computational complexity, the exact approach requires for the last test case nearly 17 seconds, while the relaxed approach calculates the results in about four seconds. All remaining heuristic approaches proposed in this thesis, achieve significantly lower computation times. Compared to all others, the priority-based heuristic achieves the lowest computational effort and reduces the average time required to calculate a solution by between 76.3% for the first test case and 99.4% for the largest one. For the last test case, the BoB approach reduces the computation time by about 98.1% and the tabu search by about 97.5% compared to the exact approach. However, the tabu search requires with 420 milliseconds approximately 4.2 times more time than the priority-based approach. The computation times of all three heuristics grow linear with the number of data centers with deviations at single test cases. For example, the computation time of the tabu search heuristic increases by a factor of about 1.5 from 270 milliseconds required for the first test case (10/300/3) to 420 milliseconds for the last one (50/300/3), whereby it requires 562 milliseconds for the fifth test case (40/300/3).

Regarding the solution quality, the LP-relaxed approach achieves unexpected results. For the chosen input parameters, it delivers by far the worst solution quality. The cost ratio indicates that the results of this approach are 71% and 82% worse

compared to the results of the exact solution approach, which is also significantly lower solution quality compared to the rest of the heuristics.

Therefore, to present the remaining heuristic in detail, we only illustrate the cost ratio of these three heuristics in Figure 10b. The priority-based approach and the BoB approach show a similar behavior regarding the cost ratio for all test cases. The difference ranges range from 0.6 to 1.5 percentage points and remains similar for an increasing amount of data centers.

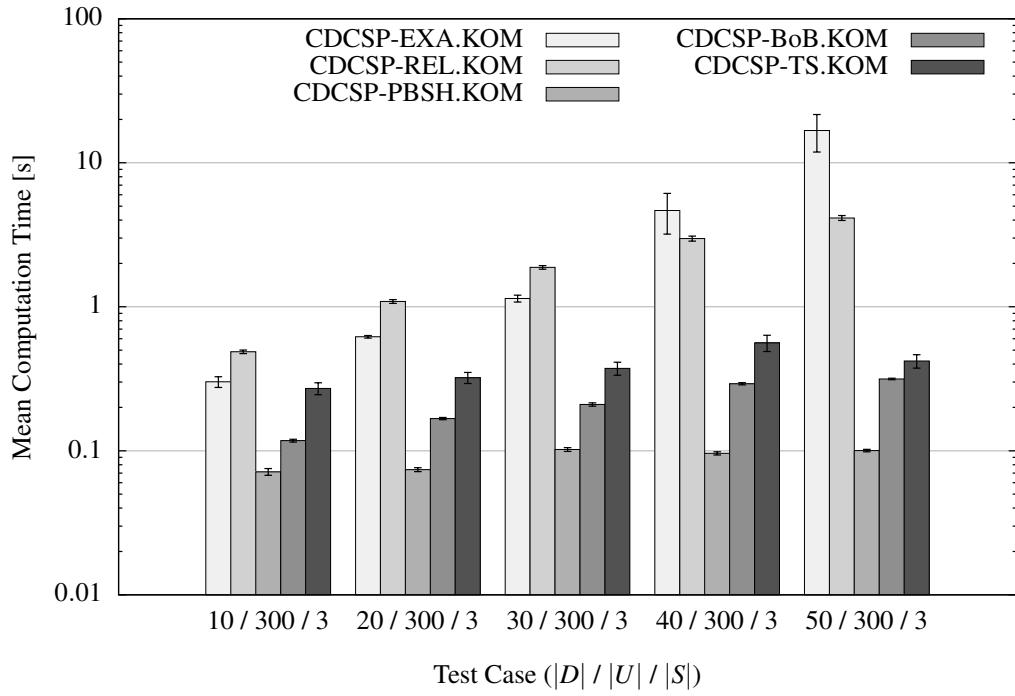
However, for both approaches there is a trend recognizable that the solution quality slightly decreases for an increasing number of data centers. The same holds true for the tabu search heuristic but on a substantially lower level. The difference between the best approach (CDCSP-TS.KOM) and the second best approach (CDCSP-BoB.KOM) increases from 4.6 to 6.5 percentage points from the first to the last test case. Also worth mentioning is the solution quality of the tabu search. For the first test case it achieves nearly the optimal solution value, with the result differing by 0.1%. For the last test case with fifty data centers tabu search causes 2.9% higher cost.

4.4.4.2 *User Clusters*

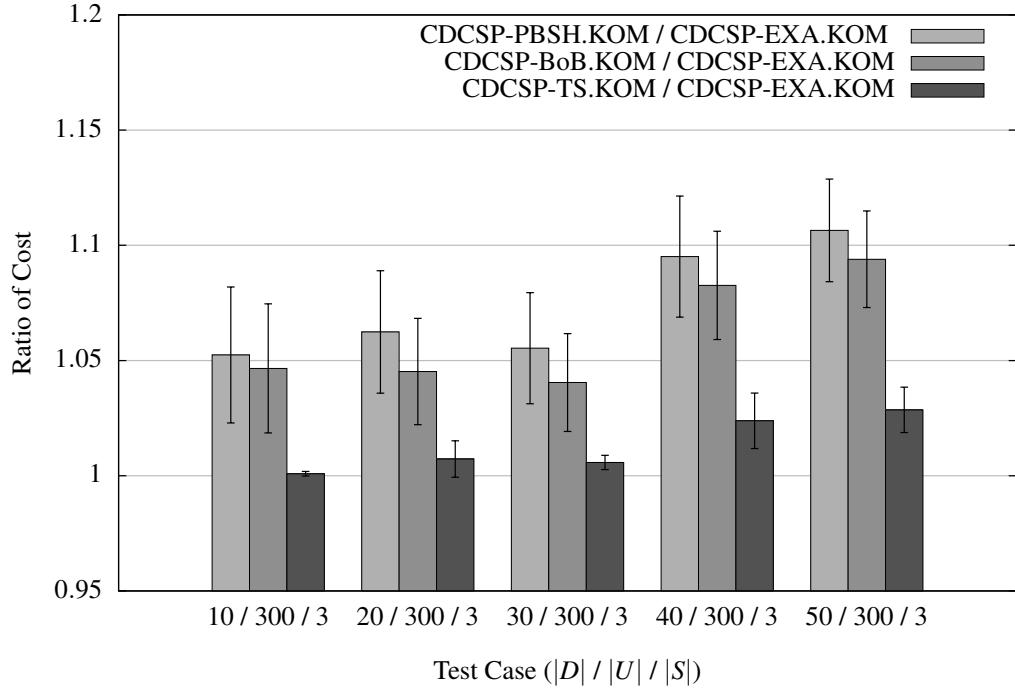
Next, we examine the impact of variations in the number of user clusters. A user cluster comprises a distinct number of single users, for example, in a metropolitan area. The absolute computation times regarding the changing number of user clusters for the optimization algorithms are given in Figure 11a.

First, in contrast to the varying number data centers, the growth in the number of user clusters causes no consistent growth in computation time for the exact solution approach. For example, from test case 30/300/3 to test case 30/400/3 the computation time increases significantly but remains constant on the same magnitude of about 10 seconds for the remaining test case. An explanation regarding this behavior might be the internal operation of the solver, which is not transparent to us. The second main observation concerns the heuristic approaches. All of our custom-tailored heuristics show a similar behavior regarding the computational effort: the growth in computation time is faster than the growth of the number of user clusters. Again, the priority-based approach is the fastest and able to calculate a solution for the largest test case within 219 milliseconds while the exact approach requires over 16.5 seconds, which correspond to a factor of about 75. Further, comparing the tabu search algorithm to the exact approach, we achieve for the first test case a reduction in computation time of 89.3% and for the last test case a reduction of 92.2%.

The cost ratios are illustrated in Figure 11b. Again, the LP-relaxed approach achieves results that are between 63.0% and 89.1% worse compared to the exact approach. Therefore, we provide its result only in the text and exclude it from the



(a) Observed mean computation times (with 95% confidence intervals). Please note the logarithmic scaling of the ordinate.



(b) Ratio of costs between the exact approach and heuristic approaches (with 95% confidence intervals).

Figure 10: Impact of different data center quantities.

chart to provide a better readability for the remaining algorithms. For the remaining heuristics there are no correlations between the number of user cluster and the solution quality recognizable. The best results are achieved by the tabu search heuristic which lead to an increase of cost between 0.06% and 2.13% compared to the exact approach.

4.4.4.3 Number of Services

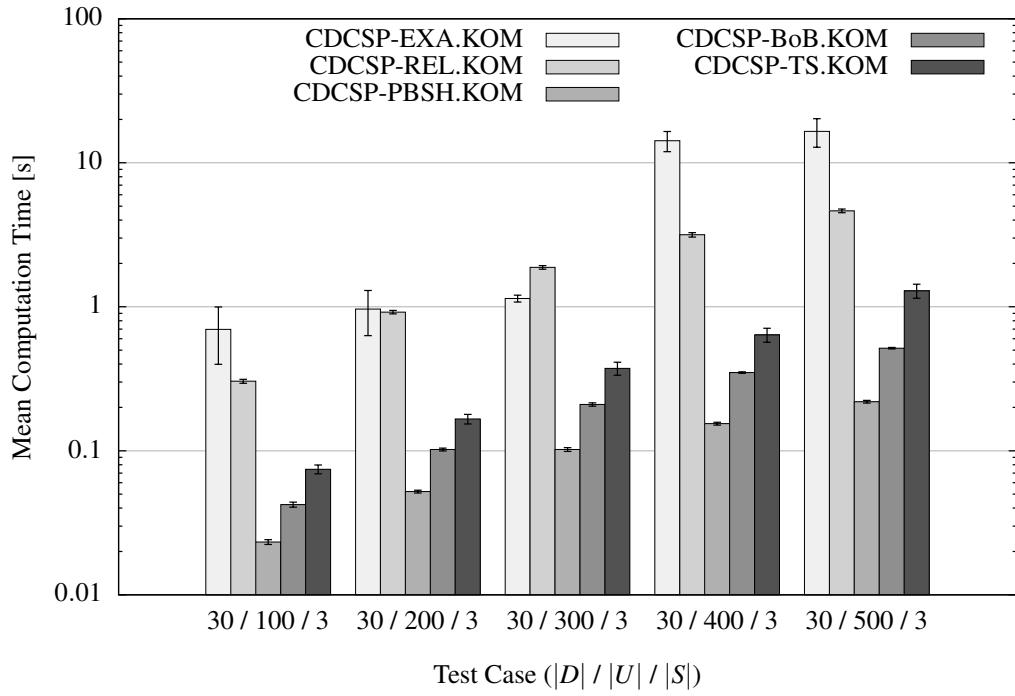
Finally we focus on the number of services. Expect for the smallest test case which will be analyzed separately, the results are similar to those for different number of user clusters. Regarding the computational effort, the exact approach shows an exponentially growing behavior, while we assume a polynomial complexity for the other approaches (according to the growth ratio between the increasing test cases). The absolute computation times are depicted in Figure 12a.

Regarding the solution quality (cf. Figure 12b), there is no correlation between test case size and the solution quality recognizable. The best results are achieved by the tabu search approach, which deteriorates the solution quality compared to the exact approach by between 0.87% for the test case with two services (30/300/2) and 3.78% for the test case with only one service (30/300/1).

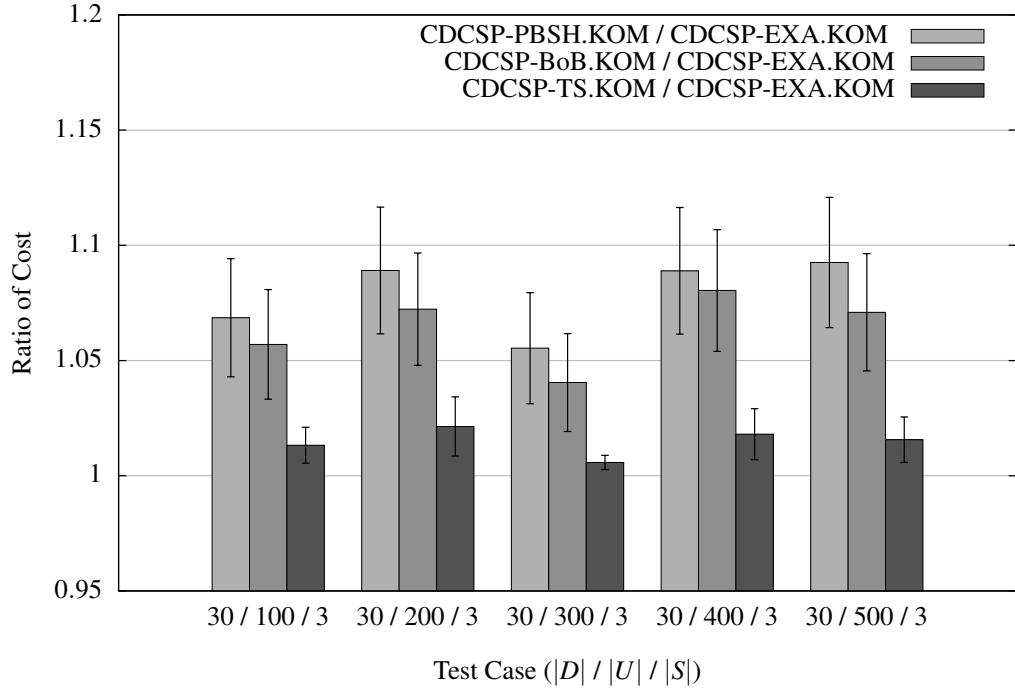
The latter test case is an outlier within the evaluation regarding computation time and solution quality. Two main effects are recognizable: (i) the exact approach requires a disproportionately high computation time and (ii) the solution quality of the *simpler* heuristics (CDCSP-PBSH, CDCSP-BoB) are the worst within the whole evaluation. This is cause by the selected input parameters that result in problem instances that are difficult to solve. The exact approach finds the optimal solution by a completely enumerating the solution space, which requires a substantial amount of time. The simpler heuristic approaches require significantly less time but only achieve a poor solution quality in such a case. Furthermore, the BoB approach does not improve the results in such a case. It just requires more time without a significant improvement of the results.

The tabu search approach clearly outperforms all other approaches. With a computation time of only 97 milliseconds (93.6% less than the exact approach) it achieves the best solution value of all heuristics with a degradation in solution quality of only 3.78%.

In summary, the evaluation shows several key findings. Regarding the LP-relaxed approach, the conducted experiments show that it is not a suitable solution approach for the CDCSP since it performs significantly worse regarding computation time and cost. The priority-based approach performs best regarding computational effort. Consequently, if fast results for large problem instances are required, this approach might be beneficial. Nevertheless, it causes the second highest total cost

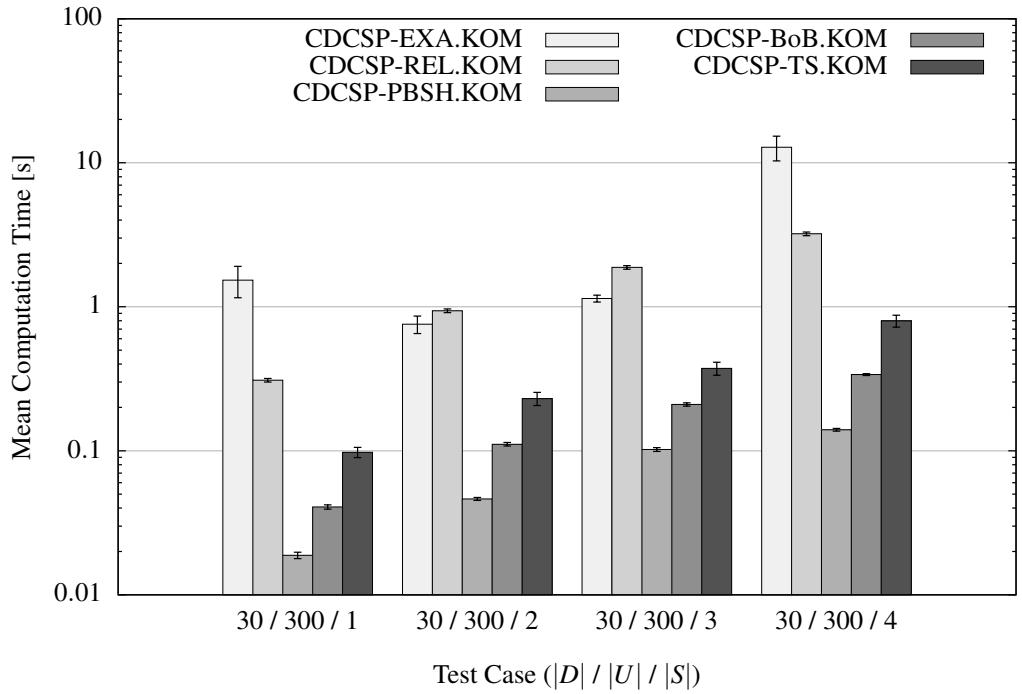


(a) Observed mean computation times (with 95% confidence intervals). Please note the logarithmic scaling of the ordinate.

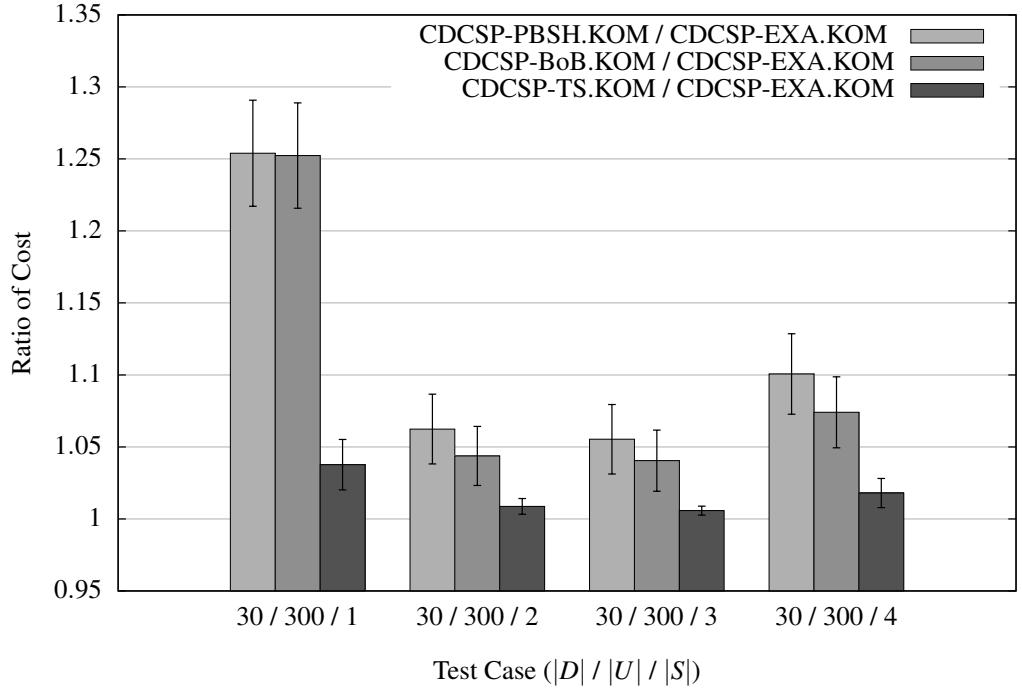


(b) Ratio of costs between the exact approach and heuristic approaches (with 95% confidence intervals).

Figure 11: Impact of different user cluster quantities.



(a) Observed mean computation times (with 95% confidence intervals). Please note the logarithmic scaling of the ordinate.



(b) Ratio of costs between the exact approach and heuristic approaches (with 95% confidence intervals).

Figure 12: Impact of different service quantities.

for all test cases. The Best-of-Breed approach delivers slightly better results than a single priority-based heuristic at the expenses of higher computation time. Since the benefits regarding the solution quality are small and the increase in computational effort compared to a single heuristic is significant, the Best-of-Breed approach is not favorable to solve the CDCSP. The most suitable heuristic approach with the best trade-off between solution quality and computational effort is the tabu search heuristic. It requires more time compared to the priority-based and BoB approaches, but it always achieves the best results out of all evaluated heuristics. Furthermore, the results have consistently good solution quality even for problems which are difficult to solve, as shown before.

The number of user clusters and services has the highest impact on the computation time needed by all approaches. Therefore, when determining the amount of user clusters, their total number should be as small as possible, e. g., by including a larger supply area. However, heterogeneous services with higher requirements for latency will unavoidably cause a higher number of user clusters.

4.5 CHAPTER SUMMARY

Cloud infrastructure providers have to face the trade-off between providing resources at low cost and offer an appropriate service quality. On the one hand, huge centralized data centers are most efficient in term of cost, since they highly benefit from economies of scale. However, on the other hand, services with high QoS requirements, such as latency, require a more decentralized infrastructure which causes higher total cost. Consequently, taking multiple services with various QoS requirements into consideration an appropriate selection of data centers is vital to address the two opposite goals.

Therefore, in this chapter we examined the Cloud Data Center Selection Problem (CDCSP) to provide cost efficient infrastructure provisioning on a global scale while considering heterogeneous services and their corresponding QoS requirements. At first, we formulated the corresponding mixed integer optimization problem, which enables us to compute the exact solution using a solver framework such as IBM CPLEX. Since this approach exhibits exponential time complexity in the worst case, it is hardly applicable for practical problems with a multitude of data centers, user clusters, and services. To find solutions in reasonable time, we developed and analyzed appropriate heuristic algorithms.

We started with a simple relaxation of the decision variables to transform the mixed integer problem into a linear problem which could be solved as well by utilizing off-the-shelf solvers frameworks. Since, this approach has proved to be not suitable for the CDCSP, we developed heuristics to better fit the given problem structure. We presented a priority-based approach that is highly efficient regarding

computational complexity. Based on these results, we developed a Best-of-Breed approach which achieves better results compared to a single priority-based approach, but only to a small extent. Finally, as an improvement procedure, we analyzed the usability of tabu search for our problem.

Within the evaluation we showed that the LP relaxed approach is not suitable to address the CDCSP, since it suffers from a significant deterioration of the solution quality and causes high computational effort. The priority-based approach achieves the lowest computation time but, causes results with worse solution quality. By utilizing the Best-of-Breed approach, we are able to increase this solution quality by combining several priority-based approaches. However, since the improvements are very small and the approach increases the computation time notably, it is less advantageous to address the optimization problem. The best trade-off between computational effort and the solution quality in this comparison is provided by tabu search. By the means of this heuristic approach we are able to find a solution for the CDCSP in reasonable time with a constantly good solution quality of all evaluated test cases.

DYNAMIC APPROACH FOR CLOUDLET ENHANCED INFRASTRUCTURES

CLOUD COMPUTING changed the way how software applications are provided to customers. One decade ago, these products were mainly installed on local desktop computers at work or at home and were used locally. Over the last years, also the way users consume software *services* changed rapidly. Nowadays, mobile devices, such as smartphones and tablets, are used for multimedia services, to communicate with friends, or to play real-time online games. This mobile utilization of cloud-based services is referred to as mobile cloud computing.

Especially regarding multimedia services, mobile cloud computing poses two major challenges: (i) a cloud provisioning model which is based on large centralized data centers and (ii) the resource limitation of mobile devices. In traditional cloud computing resources were mainly provided by large cloud data centers to benefit from economies of scale and lower operational cost, but at the expense of high latency. In 2012, Choy et al. [30] published a study which shows that only 70% of the US population can be served with highly interactive multimedia services that require network latencies of below of 80 milliseconds using public cloud infrastructure. This study only focuses on stationary computers and does not consider mobile devices. Using such devices for (multimedia) service consumption results in several additional challenges, as discussed in the following.

Inherently, mobile devices are limited in size and provide only restricted space for the battery. Hence, computationally intensive applications or a high utilization of cellular communication leads to a high battery drain and, thus, a short usage lifetime [87]. Furthermore, latency in cellular networks compared to the latency in Wi-Fi has been shown to be higher [162] or at least possess a higher variance [41].

Apart from battery drain and latency, the utilization of cellular networks causes economical disadvantages for their user. Since *real* flat rates are very rare, the cost of cellular data transfer is high in comparison to fixed subscriber lines. With regard to the high amount of data transfer – which occurs, for example, during cloud gaming – a cellular data link is rarely applicable for such kinds of services [104]. Nevertheless, using a Wi-Fi connection cloud gaming could be highly interesting for mobile devices with respect to energy consumption, as the computation load is offloaded from the mobile device into the cloud [72].

Consequently, these challenges can be addressed by locally provided, decentralized resources within the Wi-Fi range of mobile devices. Such local resources may be

installed in different public meeting places, for example, in cafes in a metropolitan area. In contrast to the cloud paradigm with its appearance of unlimited, cost-efficient resources, these local resources are limited in capacity and associated with higher costs.

In this chapter, we examine the augmentation of cloud infrastructures by the means of decentralized resource units, so-called, cloudlets. To take time-dependent service demands into consideration, we analyze the placement and selection of decentralized cloudlets using a dynamic, multi-period optimization model. The contents presented in this chapter are in part accepted for publication at a scientific conference [76] and were published in part as technical report [77].

5.1 PROBLEM STATEMENT

We assume again the role of a cloud provider which owns the infrastructure and provides resources for higher level cloud services, e.g., to software as a service providers. Thereby, the infrastructure provider is able to dispose freely over all resources. The resource units are located in data centers with different sizes: On the one hand, resources are provided by large and medium sized centralized cloud data centers. They are characterized by a huge amount of available resources and relatively low cost per resource unit but may suffer from a high distance to the user. On the other hand, the smallest possible *data centers* are cloudlets. They are characterized by a small amount of available resources, which limits the number of users that can be served without a degradation of service quality. Furthermore, cloudlets are characterized by additional cost per resource unit compared to large data centers. They are located close to the potential users, which results in low end-to-end latency. Further, we assume that the provider is in possession of global knowledge, i.e., at each time the provider knows about its available resources and is able to estimate the demand.

The quantity of required resources is determined by the given demand, which itself is determined by the number of distributed users and their consumer behavior. We aggregate all users within the Wi-Fi range of a selected hotspot into a user cluster (cf. Section 5.2). Each user cluster constitutes a specific demand for a given number of services that fluctuates over time. We assume different types of services: (i) services that require cloudlets, (ii) services that benefit from cloudlets, and (iii) services that can be easily provided by remote data centers.

The provider chooses the data centers and the corresponding resources that are located in different locations. Augmenting the current infrastructure by cloudlets may cause additional fixed cost for installing required infrastructure components. This cost could be zero, if only personal computers were used as cloudlets. However, equipping a professional server room with power and cooling capacities may cost

several thousand euros. In our model, these fixed costs occur once for the whole planning period and are assumed to be independent of the number of installed servers.

Furthermore, the provisioned resource units over the planning period, i. e., buy or leasing of servers, may incur certain fixed costs. These costs arise for the whole planning period and do not depend on whether a resource unit is used or not. But a data center does not necessarily have to be equipped with its maximum possible capacity. In addition, for each provided resource unit variable costs for operation arise, e. g., for electricity bills. Since such costs may fluctuate depending on the time period, they are modeled as time-dependent variables.

If a provider is unable to satisfy user demand, penalty costs arise, e. g., for violation of SLOs agreed in a SLA. The planning horizon covers multiple periods, where the selection decision of one period depends on the previous ones. By migration, the physical location where services are provided can be changed to accomplish an efficient resource utilization. If services are moved, i. e., migrated, migration costs arise. These costs are assumed to be independent of the type of data center or the distance between the data centers. A detailed explanation of all cost types is presented in Section 5.5.2.

The goal of a cloud infrastructure provider is to allocate resources and services to data centers and take decisions regarding the required capacity, i. e., decide on the number of resource units for each data center to minimize the overall cost of the solution. In the following, we refer to this problem as *Dynamic Cloudlet Placement and Selection Problem (DCPSP)*.

5.2 SYSTEM ARCHITECTURE

This section explains the augmentation of the core cloud infrastructure by locally installed cloudlets. Therefore, we first explain the basic concepts and different characteristics of cloudlets in Section 5.2.1. Afterwards, in Section 5.2.2 we describe their integration in the global cloud context via a metropolitan area.

5.2.1 *Cloudlets*

The core cloud infrastructure is based mainly on large, centralized data centers placed in a few location dispersed over the whole world. The provided resources are used to offload computation and data from proprietary, local resources to remote cloud data centers. Since especially mobile devices suffer from limited resources and a finite battery capacity, offloading computation and data to the cloud is a very promising concept for mobile applications. The resources of mobile devices are inherently poor compared to servers in data centers. Thus, even with advances in

technology over the last years, mobile devices are limited by factors such as battery capacity, weight, and size. These limitations are inherently to the system and will not be changed in the future [152].

By using cloud resources, tasks with high computational effort can be offloaded to more powerful remote resources. However, by using distant and centralized cloud resources, users suffer from a high Wide Area Network (WAN) latency which decreases their satisfaction when using mobile applications [181]. Furthermore, for highly interactive remote applications, even a moderate end-to-end latency of about 150 milliseconds may degrade their usability [173]. In addition to the network latency, which increases with the distance between the compute resources and a user, bandwidth limitations cause additional delays [152]. Especially graphic and processing intensive tasks, such as video capturing and cloud gaming are natural candidates for offloading. To provide these tasks as remote software services an appropriate bandwidth is required. Due to technical limitations and cost for data transfer volume in cellular networks, these conditions cannot be guaranteed in any case.

To address these challenges, literature proposes the utilization of resources at the edge of the network [30], often referred to as *cloudlets*. The term *cloudlet* has multiple different definitions. The first and most prominent definition is given by Satyanarayanan et al. [152]: "*A cloudlet is a trusted, resource-rich computer or cluster of computers, that's well-connected to the Internet and available for use by nearby mobile devices.*" Later on, Verbelen et al. [181] state that all nearby devices within a local area network that offer computational resources can be seen as a cloudlet. This also includes other smartphones that offer free resources to each other in a Peer-to-Peer (P2P) manner [29].

Cloudlets are located at the edge of the Internet and may be just one hop away from the user's mobile device. According to Satyanarayanan et al. [153], they pose different advantages for mobile applications: First, they offer computation power close to the customer and, thus, form a basis for providing compute-intensive and latency-sensitive applications. Second, they reduce the bandwidth consumption between a remote cloud data center and the mobile devices. Third, cloudlets can serve as a proxy and, thus, enhance the robustness and availability for provided cloud services.

In literature, two ways are proposed to connect mobile devices to cloudlets. The first approach proposes a commonly used Local Area Network (LAN), where cloudlets are provided by enhanced Wi-Fi hotspots, a personal computer, a cluster of servers, or even a small data center. Furthermore, also mobile devices such as notebooks or smartphones can form a cloudlet to provide free resources within a LAN [152, 181]. Using such a LAN-based approach, users benefit from a low propagation delay and a high bandwidth. The second proposed approach is the realization of cloudlets within a provider's cellular infrastructure, i. e., in the base

stations of a cellular network [26]. The major advantage of this deployment is the utilization of the existing infrastructure. Thus, no additional locations for deploying hardware are required and, compared to Wi-Fi hotspots, a wider area can be covered, which enables increased user mobility. Nevertheless, in contrast to a Wi-Fi connection, the utilization of a cellular connection poses some drawbacks: (i) it requires more energy compared to a Wi-Fi connection and (ii) depending on the number of users, the available bandwidth is substantially decreased.

Edge resources can be provided utilizing different hardware devices, which results in two different cloudlet types: (i) stationary and (ii) ad hoc cloudlets [181]. Stationary cloudlets consist of permanently installed hardware, such as servers. In contrast, ad hoc cloudlets are built by dynamic nodes, which may join or leave a cloudlet at any time. Such cloudlets register themselves to a local repository or they organize themselves in a P2P network [29, 181].

Regarding the ownership, two different deployment models can be distinguished. First, cloudlets may be owned and offered by providers within their owned infrastructure. Hence, such a provider bears the entrepreneurial risk, but owns the complete profits [152, 181]. In contrast to this top-down approach, cloudlets can be deployed in a bottom-up manner. Hence, they are installed by local business owners on their premises in cafés or medical offices to provide their customers additional services such as free Wi-Fi today [152]. Furthermore, cloudlets can be deployed and used within companies or private homes [181].

To provide services via cloudlets, different software deployment methods are proposed in literature: (i) based on virtual machines [26, 152] and (ii) based on component offloading [181]. Utilizing VM-based environments leads to a simple and stringent separation of different services. Nevertheless, the major drawback of this approach is its inefficient resource utilization. Since cloudlet resources are very limited, only a limited amount of VMs can be supported at the same time. To address this challenge, Verbelen et al. [181] propose a component-based offloading approach. Components are software programs with defined interfaces that can be called via remote procedure calls (RPC) and offer a more flexible and efficient way for resource allocation.

Literature especially emphasizes the advantages of cloudlet-enhanced environments for applications with high computational effort and low latency requirements. For example, due to their fast Internet connection cloudlets are used as caches for video streaming [50]. In sophisticated application scenarios, cloudlets are used for live video content analysis [153] or for augmented reality applications [152, 181]. Further, cloudlets are proposed for cooperative file editing, real-time data analytics, or local servers for multiplayer gaming [50, 153]. They are also utilized to offer *desktop as a service* applications to provide office programs to mobile devices independently of the underlying hardware architecture. Here, nearby compute resources execute the guest operating system, such as Microsoft Windows, and transfer the

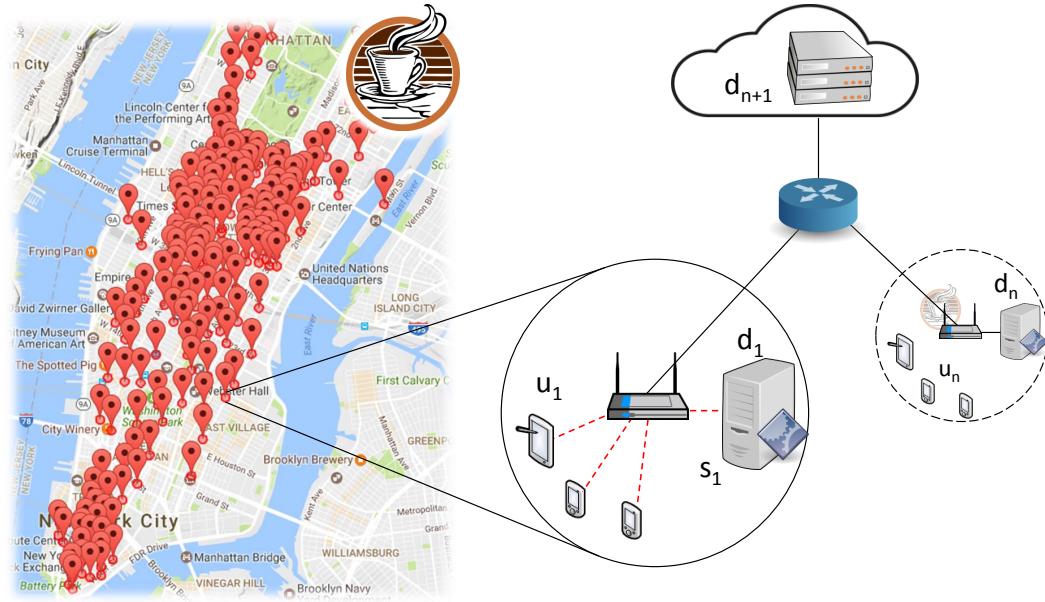


Figure 13: Integration of cloudlets within a metropolitan area.¹

video stream to a smartphone or tablet [153]. Since this approach is not limited to Microsoft operating systems and software, any business-driven application and service can be provided this way.

In our model, we focus on stationary installed cloudlets, which are connected via Wi-Fi to the users. As deployment model we assume a top-down approach, where the provider decides where to place the cloudlets. As possible location we assume cafes that already provide Wi-Fi to their customers. Although the provider makes the placement decisions, cooperation contracts between the provider and local shops are necessary, which is out of the scope of this theses. We also do not consider a dedicated software deployment method since we estimate the required resources as service units. If different software deployment methods cause different resource demand, these demands could be modeled with the input parameters of our model.

5.2.2 Local Integration of Cloudlets

As described in the previous section, to augment a global cloud infrastructure by local resources, we are focusing on stationary installed cloudlets that provide their services via Wi-Fi to customers. Further, in our proposed approach a provider decides where to place cloudlets. As possible locations we propose local businesses that are already equipped with Wi-Fi hotspots. Figure 13 presents the integration of

¹ Starbucks coordinates in Manhattan: <https://nycstarbucks.com/the-list/>; Map data ©Google 2017

cloudlets in (i) a metropolitan area and (ii) the connection to the cloud. We illustrate the stores of a single café chain in Manhattan with an approximation of their Wi-Fi ranges (red circles). Adding more fast food chains or other local businesses, a broad area of Manhattan could be covered by Wi-Fi and, hence, by local computation resources. Users within the Wi-Fi range of a restaurant form a user cluster u_n . The number of user clusters depends on the number of considered restaurants. We further assume that all or at least some of these restaurants could be equipped with hardware to build a cloudlet. We denote a cloudlet as data center d_n that is placed in a restaurant that offers Wi-Fi connectivity to the user cluster u_n (cf. Figure 13).

The particular cafés are connected via a Metropolitan Area Network (MAN) and can communicate with each other. This approach corresponds to findings in literature that cloudlets provide a delay reduction if the maximum number of hops does not exceed two [50]. The cloud, i. e., remote cloud data centers, are connected via a WAN to the local user clusters. In our model the cloud is represented as data center d_{n+1} . Since the cloud resources are located far away from the user clusters, we assume a notably higher latency compared to the locally installed cloudlets. In the following section we combine all described aspects into a common optimization model.

5.3 OPTIMIZATION MODEL

To address the described problem, we use different optimization approaches. The required formal notations are stated in Section 5.3.1. Subsequently in Section 5.3.2 we define our exact optimization approach *DCPSP-EXA.KOM*. A heuristic optimization approach for the problem is stated in Section 5.4.

5.3.1 Formal Notations

To provide a mathematical model, several formal notations are required. First, we formally define the basic entities of the *Dynamic Cloudlet Placement and Selection Problem (DCPSP)*.

- $D = \{d_1, \dots, d_N\}$: Set of (potential) data centers. The term *data center* encompasses large centralized data centers and micro data centers, i.e., cloudlets.
- $U = \{u_1, \dots, u_M\}$: Set of user clusters. A user cluster encompasses all users in range of a Wi-Fi hotspot.
- $S = \{s_1, \dots, s_N\}$: Set of demanded services.
- $Q = \{q_1, \dots, q_\Xi\}$: Set of considered QoS attributes.
- $T = \{t_1, \dots, t_O\}$: Set of discrete time slots within the planning period.

Based on the previously introduced basic entities, subsequently we describe the required parameters regarding resource demand and supply:

- $V_{u_\mu, s_v, t_o} \in \mathbb{N}$: Service demand of user cluster u_μ for service s_v at time t_o .
- $K_{d_\lambda}^{\min} \in \mathbb{N}$: Minimal capacity of data center d_λ .
- $K_{d_\lambda}^{\max} \in \mathbb{N}$: Maximal capacity of data center d_λ .

The connection to users in a user cluster is determined by link capacities for the LAN and the MAN connection as illustrated in Figure 14. We distinguish between download and upload capacities:

- $K_{u_\mu}^{LAN_{\text{down}}} \in \mathbb{R}^+$: LAN download capacity of user cluster u_μ .
- $K_{u_\mu}^{LAN_{\text{up}}} \in \mathbb{R}^+$: LAN upload capacity of user cluster u_μ .
- $K_{u_\mu}^{MAN_{\text{down}}} \in \mathbb{R}^+$: MAN download capacity of user cluster u_μ .
- $K_{u_\mu}^{MAN_{\text{up}}} \in \mathbb{R}^+$: MAN upload capacity of user cluster u_μ .

For service provisioning various types of costs may arise for an infrastructure provider:

- $C_{d_\lambda}^{\text{fix}} \in \mathbb{R}^+$: Fixed costs for choosing or constructing a data center d_λ .
- $C_{d_\lambda}^{\text{hw}} \in \mathbb{R}^+$: Fixed costs for buying or leasing hardware for data center d_λ .
- $C_{d_\lambda, t_o}^{\text{op}} \in \mathbb{R}^+$: Variable costs for operating one resource unit for one time unit in data center d_λ at time t_o . This cost is assumed agnostic of the service running on top.
- $C_{s_v}^{\text{mig}} \in \mathbb{R}^+$: Migration costs for moving service s_v from one data center to another between two subsequent time periods. The migration costs are assumed to be independant of the time.
- $C_{u_\mu, s_v}^{\text{pen}} \in \mathbb{R}^+$: Penalty cost per service unit not provided to user u_μ w.r.t. service s_v .

For service provisioning QoS guarantees and QoS requirements are needed:

- $Q_{d_\lambda, u_\mu, q_\xi}^{\text{qua}} \in \mathbb{R}^+$: QoS guarantee of data center d_λ w.r.t. user u_μ for QoS attribute q_ξ .
- $Q_{u_\mu, s_v, q_\xi}^{\text{req}} \in \mathbb{R}^+$: QoS requirement of user u_μ w.r.t. service s_v for QoS attribute q_ξ .

Further, each service requires a predefined data rate for download and upload:

- $L_{s_v}^{\text{down}} \in \mathbb{R}^+$: Required downstream for service s_v per resource unit.
- $L_{s_v}^{\text{up}} \in \mathbb{R}^+$: Required upstream for service s_v per resource unit.

Lastly, in order to model the DCSP as optimization problem, various decision variables are required.

- $x_{d_\lambda} \in \{0, 1\} \quad \forall d_\lambda \in D$: Indicates whether a data center d_λ will be used or not.
- $y_{d_\lambda, u_\mu, s_v, t_o} \in \mathbb{N} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T$: Number of resources a data center d_λ provides to a user cluster u_μ regarding a service s_v in time period t_o
- $y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}} \in \mathbb{N} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T$: Number of resources that are migrated from one to another data center in between the time periods t_{o-1} and t_o .
- $y_{u_\mu, s_v, t_o}^{\text{pen}} \in \mathbb{N} \quad \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T$: Demand that is not satisfied by the provider. These not provided resources will cause penalty costs for the provider.
- $z_{d_\lambda} \in \mathbb{N} \quad \forall d_\lambda \in D$: Number of hardware resource units provided within a data center d_λ . This number represents the upper bound of the available resources.

5.3.2 Mathematical Model

Based on the problem description from Section 5.1 and on the formal notations from Section 5.3.1, we formulate a mathematical optimization model. In the following subsections we describe the decision variables, followed by the objective function and the corresponding constraints. Finally, to formulate a Mixed Integer Program, we describe the required auxiliary constructs.

Based on the previously described problem, we formulate the objective function. Therefore, due to readability, we use an abbreviated notation for the sum. The left-hand side formula shows our abbreviation, the right hand side formula the complete notation (cf. Equation 5.1). This example applies to all subsequent summations.

$$\sum_{\substack{\lambda=1..M \\ \mu=1..M}} \dots \hat{=} \sum_{\lambda=1}^M \sum_{\mu=1}^M \dots \quad (5.1)$$

The goal of the optimization function is to minimize the total cost depending on the values of the decision variables (cf. Equation 5.2).

$$\begin{aligned}
\text{Min.C}(x, y, z) = & \sum_{\lambda=1..N} x_{d_\lambda} \times C_{d_\lambda}^{\text{fix}} \\
& + \sum_{o=1..O} \left(\sum_{\substack{\lambda=1..N \\ \mu=1..M \\ v=1..N}} y_{d_\lambda, u_\mu, s_v, t_o} \times C_{d_\lambda, t_o}^{\text{op}} + \sum_{\substack{\mu=1..M \\ v=1..N}} y_{u_\mu, s_v, t_o}^{\text{pen}} \times C_{u_\mu, s_v}^{\text{pen}} \right) \\
& + \sum_{o=2}^O \sum_{\substack{\lambda=1..N \\ \mu=1..M \\ v=1..N}} y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}} \times C_{s_v}^{\text{mig}} + \sum_{\lambda=1..N} z_{d_\lambda} \times C_{d_\lambda}^{\text{hw}}
\end{aligned} \tag{5.2}$$

The first summand depends on the decision variable x_{d_λ} . It represents the fixed cost which occurs when a data center is used, e.g., for installation or long-term lease of the facilities. In case of cloudlets placed in decentralized locations, such as cafes, this cost will be expenses for installing a rack, air conditioning, or an appropriate power supply. This fixed cost occurs once for the planning period and does not depend on whether a data center provides resource units to user clusters or not.

The second part of the term refers to resource units that are provided in a certain time period. For the provisioning of one resource unit $y_{d_\lambda, u_\mu, s_v, t_o}$ variable operational costs $C_{d_\lambda, t_o}^{\text{op}}$ arise. These costs depend on which data centers are used and the time period. The cost per data center may differ because of different wages and other regional factors influencing the variable operational cost. In addition, the variable cost may change during the course of time. With this time dependency, we are, for example, able to reflect fluctuating electricity prices throughout the day.

The third summand refers to the capacities that cause penalty costs $y_{u_\mu, s_v, t_o}^{\text{pen}}$. Such capacities are requested by a user cluster u_μ but they are not provided by one of the data centers. This under-provisioning may come into place, when the provisioning of additional resources may cause higher total cost compared to the penalty cost $C_{u_\mu, s_v}^{\text{pen}}$.

The forth summand reflects the migration cost. This cost depends on the number of resource units $y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}}$ which are affected by a migration, e.g., if software components of a provided service need to be moved from one location to another. The migration cost $C_{u_\mu, s_v}^{\text{mig}}$ includes the data transfer cost from one data center to another. Migration cost occurs only from the second time period on, since in our model, launching a new service does not cause migration cost. Because of their complexity, the calculation and the mathematical formulation of the migration cost are described separately in Section 5.3.2.2.

The fifth and last summand refers to the hardware units z_{d_λ} that need to be bought or leased. Installing such resource units, even if they are not used in a time period leads to the cost for servers $C_{d_\lambda, s_v}^{\text{hw}}$.

5.3.2.1 Constraints

To guarantee a valid solution, the objective function is subject to several constraints, which are explained subsequently. We begin with the demand for resources V_{u_μ, s_v, t_o} . Each user cluster demands a specific amount of resources regarding different services in each time period. In our optimization approach, the provider has the choice to fulfill the demand or not. The summation of the provided and not provided capacities (with the latter causing penalty cost) must be equal or greater to the resource demand of all user clusters and all requested services at each point in time (cf. Equation 5.3).

$$y_{u_\mu, s_v, t_o}^{pen} + \sum_{\lambda=1..M} y_{d_\lambda, u_\mu, s_v, t_o} \geq V_{u_\mu, s_v, t_o} \quad \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T \quad (5.3)$$

Each data center has a minimal and a maximal capacity. The maximal capacity of a data center is obvious and limited by the available area and technology restrictions. The minimal capacity refers to economic considerations. Large data centers can only be cost-efficiently operated when a minimum number of resources is provided. Because of the decentralized nature of cloudlets this value can be very low compared to remote cloud data centers. These lower and upper capacity bounds determine the number of hardware resources z_{d_λ} , basically servers, which can be provided by each data center. Furthermore, a data center d_λ may only provide resources, if it is actually selected in the initial placement and, thus, the binary variable x_{d_λ} is equal to *one* (cf. Equation 5.4 and 5.5).

$$\sum_{\substack{\mu=1..M \\ v=1..N}} y_{d_\lambda, u_\mu, s_v, t_o} \leq z_{d_\lambda} \quad \forall d_\lambda \in D, \forall t_o \in T \quad (5.4)$$

$$\begin{aligned} z_{d_\lambda} &\leq x_{d_\lambda} \times K_{d_\lambda}^{\max} & \forall d_\lambda \in D \\ z_{d_\lambda} &\geq x_{d_\lambda} \times K_{d_\lambda}^{\min} & \forall d_\lambda \in D \end{aligned} \quad (5.5)$$

Further, all agreed QoS requirements must be met. The binary variable $p_{d_\lambda, u_\mu, s_v}$ determines whether a data center is able to provide resources or not (cf. Equation 5.6). Only if all QoS guarantees $Q_{d_\lambda, u_\mu, q_\xi}^{guar}$ are equal to or better than the QoS requirements $Q_{u_\mu, s_v, q_\xi}^{req}$, the variable $p_{d_\lambda, u_\mu, s_v}$ gets the value one and thus a data center is given the permission to provide resources.

$$y_{d_\lambda, u_\mu, s_\nu, t_o} \leq p_{d_\lambda, u_\mu, s_\nu} \times K_{d_\lambda}^{max} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (5.6)$$

$$p_{d_\lambda, u_\mu, s_\nu} = \begin{cases} 1 & \text{if } Q_{d_\lambda, u_\mu, q_\xi}^{guar} \geq Q_{u_\mu, s_\nu, q_\xi}^{req} \forall q_\xi \in Q \\ 0 & \text{else} \end{cases} \quad (5.7)$$

Within the case differentiation in Equation 5.7 we use the *greater or equal* sign (\geq) to emphasize that the QoS guarantees are required to at least fulfill the QoS requirements. Nevertheless, to calculate the value of $p_{d_\lambda, u_\mu, s_\nu}$ two cases have to be considered separately. First, QoS criteria which need to be minimized, such as latency, and second, QoS criteria that need to be maximized, such as availability. Therefore, in the implementation we differentiate these two cases. Equation 5.8 and 5.9 show two corresponding examples. Only if all of these conditions are fulfilled for all QoS criteria q_ξ , a data center is able to provide resources to a specific user cluster w.r.t. a specific service.

$$Q_{d_\lambda, u_\mu, q_{Latency}}^{guar} \leq Q_{u_\mu, s_\nu, q_{Latency}}^{req} \rightarrow p_{d_\lambda, u_\mu, s_\nu, q_{Latency}} := 1 \quad (5.8)$$

$$Q_{d_\lambda, u_\mu, q_{Availability}}^{guar} \geq Q_{u_\mu, s_\nu, q_{Availability}}^{req} \rightarrow p_{d_\lambda, u_\mu, s_\nu, q_{Availability}} := 1 \quad (5.9)$$

Each user cluster is connected to two different networks: A LAN, i.e., Wi-Fi to connect the users within a café, and a MAN to connect other user clusters including their cloudlets, as well as to connect to a remote cloud data center. Each service that is used within a user cluster requires a specific average amount of bandwidth, regardless of whether the service is provided by the local cloudlet or remote resources. The total amount of required bandwidth must be less than or equal to the available download and upload capacities. We differentiate between download capacity, $K_{u_\mu}^{LAN_{down}}$, and upload capacity, $K_{u_\mu}^{LAN_{up}}$, to reflect the different requirements of the services, e.g., high download requirements for cloud gaming. The corresponding conditions are represented in the subsequent equations (cf. Equation 5.10 and 5.11). The different data links are illustrated in Figure 14.

$$\sum_{\lambda=1..N} \sum_{\nu=1..N} y_{d_\lambda, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{\text{down}} \leq K_{u_\mu}^{\text{MAN}_{\text{down}}} \quad \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (5.10)$$

$$\sum_{\lambda=1..N} \sum_{\nu=1..N} y_{d_\lambda, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{\text{up}} \leq K_{u_\mu}^{\text{MAN}_{\text{up}}} \quad \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (5.11)$$

A user cluster requires the MAN connection to consume services provided by remote resources, i. e., other cloudlets within and cloud data centers. Additionally, the MAN is used to provide services to other user clusters within a common metropolitan area. For all services which are provided by the local cloudlet and used by the local users, no MAN capacities are required at all. In Equation 5.12 and Equation 5.13, we consider multiple services with their specific bandwidth requirements and differentiate between download and upload capacities.

$$\begin{aligned} & \sum_{\substack{\lambda=1..N \\ \lambda \neq \alpha}} \sum_{\nu=1..N} y_{d_\lambda, u_\alpha, s_\nu, t_o} \times L_{s_\nu}^{\text{down}} + \sum_{\substack{\mu=1..M \\ \mu \neq \alpha}} \sum_{\nu=1..N} y_{d_\alpha, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{\text{up}} \\ & \leq K_{u_\alpha}^{\text{MAN}_{\text{down}}} \quad \forall d_\alpha \in D, \forall u_\alpha \in U, \forall s_\nu \in S, \forall t_o \in T \end{aligned} \quad (5.12)$$

$$\begin{aligned} & \sum_{\substack{\lambda=1..N \\ \lambda \neq \alpha}} \sum_{\nu=1..N} y_{d_\lambda, u_\alpha, s_\nu, t_o} \times L_{s_\nu}^{\text{up}} + \sum_{\substack{\mu=1..M \\ \mu \neq \alpha}} \sum_{\nu=1..N} y_{d_\alpha, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{\text{down}} \\ & \leq K_{u_\alpha}^{\text{MAN}_{\text{up}}} \quad \forall d_\alpha \in D, \forall u_\alpha \in U, \forall s_\nu \in S, \forall t_o \in T \end{aligned} \quad (5.13)$$

The complete optimization approach is depicted in the Models 6 and 7 in Section A.2. The calculation of the migration cost requires a case differentiation and a transformation to a linear equation system. Both are explained in detail in the subsequent section.

5.3.2.2 Transformation of the Migration Condition

In Section 5.5.2 we described in detail the reasons and the assumed values for the migration cost. The calculation of the total amount is stated in the fourth summand of the objective function. The isolated term is given subsequently in Equation 5.14.

$$\sum_{\substack{\lambda=1..M \\ \mu=1..M \\ \nu=1..N \\ \alpha=1..O}} y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}} \times C_{s_\nu}^{\text{mig}} \quad (5.14)$$

Migration cost can arise starting from time period two, when provided capacities are shifted from one data center to another. The identification and modeling of the migrated units are crucial parts of the optimization and require a case analysis, followed by a transformation to a linear equation system. To determine the amount of migrated resource units, we differentiate between two cases:

1. The aggregated number of resources provided to one user cluster u_μ w.r.t. a specific service is either constant or increases from one planning period (t_{o-1}) to the next (t_o), while the resource share provided by the considered data center d_α decreases.
2. The aggregated number of resources provided to one user cluster u_μ w.r.t. a specific service decreases from one planning period (t_{o-1}) to the next (t_o), while the resource share provided by the considered data center d_α increases.

In both case, we assume that, for example, the data center d_1 is used to provide resources for a specific service s_ν . In the subsequent period the resources are used for another purpose and the service that is provided to the user cluster is moved to another data center d_2 . Shutting down one service in a data center and starting the required software components in another one, as well as transferring the required data, implicate additional effort compared to continuous service provisioning.

Hence, if a service s_ν for a user cluster u_μ is terminated at d_λ and a new instance of this service s_ν is spawned at another data center, e.g., $d_{\lambda+1}$, we define this as a migration. Therefore, migration cost arise. If neither the first nor the second condition holds true, zero resource units are migrated. The mathematical formulation of these conditions is represented in Equation 5.15.

Table 11: Decomposition of the two *if* conditions

	Part p_1	Part p_2
Cond. c_1	$\sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_o} \geq \sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_{o-1}}$	$y_{d_\lambda, u_\mu, s_v, t_o} \leq y_{d_\lambda, u_\mu, s_v, t_{o-1}}$
Cond. c_2	$\sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_o} < \sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_{o-1}}$	$y_{d_\lambda, u_\mu, s_v, t_o} > y_{d_\lambda, u_\mu, s_v, t_{o-1}}$

$$y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}} = \begin{cases} y_{d_\lambda, u_\mu, s_v, t_{o-1}} - y_{d_\lambda, u_\mu, s_v, t_o} & \text{if} \\ \sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_o} \geq \sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_{o-1}} \\ \wedge y_{d_\lambda, u_\mu, s_v, t_o} \leq y_{d_\lambda, u_\mu, s_v, t_{o-1}} \\ y_{d_\lambda, u_\mu, s_v, t_o} - y_{d_\lambda, u_\mu, s_v, t_{o-1}} & \text{if} \\ \sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_o} < \sum_{\alpha=1.. \wedge} y_{d_\alpha, u_\mu, s_v, t_{o-1}} \\ \wedge y_{d_\lambda, u_\mu, s_v, t_o} > y_{d_\lambda, u_\mu, s_v, t_{o-1}} \\ 0 & \text{else} \end{cases} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T \quad (5.15)$$

To transfer this case differentiation to a linear equation system, various transformation steps are required. The required transformation of the *if* condition and the logic *and* operations were adopted based on the instructions and examples in [167].

Each of the two cases consist of an *if* condition (c_1 and c_2) and both conditions again consist of two parts (p_1 and p_2), which are connected via an *and* operator (cf. Table 11). This nomenclature is reflected later on in the naming of various auxiliary variables. Table 11 illustrates the conditions and their parts. First, to calculate the final amount of the migrated resource units, we introduce two additional auxiliary variables which refer to the amount of migrated resources, each resulting from one of the two conditions (cf. Equation 5.16 and 5.17).

$$y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}, c_1} \in \mathbb{N} \quad (5.16)$$

$$y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}, c_2} \in \mathbb{N} \quad (5.17)$$

The sum of the two conditions results in the total amount of migrated units (cf. Equation 5.18).

$$y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}} = y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}, c_1} + y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}, c_2} \quad (5.18)$$

To calculate the migrated resource units, we split the two conditions in two separate case differentiations (cf. Equations 5.19 and 5.20). Within the two new case differentiations, we replace the *if* conditions by two auxiliary variables, that state whether condition c_1 or c_2 is fulfilled.

$$y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}, c_1} = \begin{cases} y_{d_\lambda, u_\mu, s_v, t_{o-1}} - y_{d_\lambda, u_\mu, s_v, t_o} & \text{if } h_{d_\lambda, u_\mu, s_v, t_o}^{c_1} = 1 \\ 0 & \text{else} \end{cases} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T \quad (5.19)$$

$$y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}, c_2} = \begin{cases} y_{d_\lambda, u_\mu, s_v, t_o} - y_{d_\lambda, u_\mu, s_v, t_{o-1}} & \text{if } h_{d_\lambda, u_\mu, s_v, t_o}^{c_2} = 1 \\ 0 & \text{else} \end{cases} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T \quad (5.20)$$

The first auxiliary variable (cf. Equation 5.21) is assigned a value of *one* if the first condition (c_1) is fulfilled and the second auxiliary variable (cf. Equation 5.22) indicates the corresponding value for the second condition (c_2).

$$h_{d_\lambda, u_\mu, s_v, t_o}^{c_1} \in \{0, 1\} \quad (5.21)$$

$$h_{d_\lambda, u_\mu, s_v, t_o}^{c_2} \in \{0, 1\} \quad (5.22)$$

Only if at least one of both auxiliary variables is assigned a value of *one*, a migration occurs, i. e., the amount of migrated resources exceeds the value of zero. Furthermore, the value of resource units that can be migrated is restricted to the maximal capacity of a data center and is calculated as follows. Subsequently, we explain the

calculation and transformation of the first case differentiation $y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_1}$ in detail. The second one $y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_2}$ is calculated and transformed analogously.

If condition c_1 is *true*, $h_{d_\lambda, u_\mu, s_v, t_o}^{c_1}$ is assigned the value *one*. In this case, the resource units that need to be migrated are calculated as stated in Equation 5.23 and 5.24. If the binary variable $h_{d_\lambda, u_\mu, s_v, t_o}^{c_1}$ has a value of *one*, the right-hand side of the inequation becomes zero and the units to be migrated are calculated using the difference between the provided units in the previous time period and the provided units in the current time period.

$$y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_1} - (y_{d_\lambda, u_\mu, s_v, t_{o-1}} - y_{d_\lambda, u_\mu, s_v, t_o}) \leq K_{d_\lambda}^{max} \times (1 - h_{d_\lambda, u_\mu, s_v, t_o}^{c_1}) \quad (5.23)$$

$$(y_{d_\lambda, u_\mu, s_v, t_{o-1}} - y_{d_\lambda, u_\mu, s_v, t_o}) - y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_1} \leq K_{d_\lambda}^{max} \times (1 - h_{d_\lambda, u_\mu, s_v, t_o}^{c_1}) \quad (5.24)$$

The second case requires that $y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_1} \leftarrow 0$ if $h_{d_\lambda, u_\mu, s_v, t_o}^{c_1} \in \{0, 1\} = 0$. To express this requirement, we use Equation 5.25. The maximum number of migrated units for one data center is restricted by its maximum capacity.

$$y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_1} \leq K_{d_\lambda}^{max} \times h_{d_\lambda, u_\mu, s_v, t_o}^{c_1} \quad (5.25)$$

For condition c_2 we use the same approach. Only if $h_{d_\lambda, u_\mu, s_v, t_o}^{c_2}$ is equal to one, $y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_2}$ can exceed the value zero. The calculations are stated in Equation 5.26 to 5.27.

$$y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_2} - (y_{d_\lambda, u_\mu, s_v, t_o} - y_{d_\lambda, u_\mu, s_v, t_{o-1}}) \leq K_{d_\lambda}^{max} \times (1 - h_{d_\lambda, u_\mu, s_v, t_o}^{c_2}) \quad (5.26)$$

$$(y_{d_\lambda, u_\mu, s_v, t_{o-1}} - y_{d_\lambda, u_\mu, s_v, t_o}) - y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_2} \leq K_{d_\lambda}^{max} \times (1 - h_{d_\lambda, u_\mu, s_v, t_o}^{c_2}) \quad (5.27)$$

$$y_{d_\lambda, u_\mu, s_v, t_o}^{mig, c_2} \leq K_{d_\lambda}^{max} \times h_{d_\lambda, u_\mu, s_v, t_o}^{c_2} \quad (5.28)$$

Up to now, we have considered the *if* conditions as one single construct which is represented by a binary variable that can be either *one* or *zero* to state whether the condition is *true* or *false*. However, both conditions consist of two parts that are con-

nected by an *and* operation and, thus, both parts of each condition must be *true* for the whole condition to become *true*. Therefore, we use dedicated boolean variables which represent the two parts of the logic *and* relationship (cf. Equation 5.29 to 5.32). Each of the variables need to become *one* if the respective right hand side of the inequation is *true*.

$$h_{d_\lambda, u_\mu, s_v, t_o}^{c_1, p_1} \in \{0, 1\} : \sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_o} \geq \sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_{o-1}} \quad (5.29)$$

$$h_{d_\lambda, u_\mu, s_v, t_o}^{c_2, p_1} \in \{0, 1\} : \sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_o} < \sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_{o-1}} \quad (5.30)$$

$$h_{d_\lambda, u_\mu, s_v, t_o}^{c_1, p_2} \in \{0, 1\} : y_{d_\lambda, u_\mu, s_v, t_o} < y_{d_\lambda, u_\mu, s_v, t_{o-1}} \quad (5.31)$$

$$h_{d_\lambda, u_\mu, s_v, t_o}^{c_2, p_2} \in \{0, 1\} : y_{d_\lambda, u_\mu, s_v, t_o} > y_{d_\lambda, u_\mu, s_v, t_{o-1}} \quad (5.32)$$

To determine the values of the binary variables for the first part p_1 of each of the conditions we proceed as follows. Depending on the difference between the total resource provisioning of two succeeding time periods, these variables require the value *zero* or *one* to fulfill the conditions in Equation 5.33 and 5.34.

$$\sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_o} - \sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_{o-1}} \leq \sum_{\alpha=1..A} K_{d_\alpha}^{\max} \times h_{d_\lambda, u_\mu, s_v, t_o}^{c_1, p_1} \quad (5.33)$$

$$\sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_{o-1}} - \sum_{\alpha=1..A} y_{d_\alpha, u_\mu, s_v, t_o} < \sum_{\alpha=1..A} K_{d_\alpha}^{\max} \times h_{d_\lambda, u_\mu, s_v, t_o}^{c_2, p_1} \quad (5.34)$$

For the second part of each condition, we use the same approach as described before.

$$y_{d_\lambda, u_\mu, s_v, t_{o-1}} - y_{d_\lambda, u_\mu, s_v, t_o} \leq K_{d_\lambda}^{\max} \times h_{d_\lambda, u_\mu, s_v, t_o}^{c_1, p_2} \quad (5.35)$$

$$y_{d_\lambda, u_\mu, s_v, t_o} - y_{d_\lambda, u_\mu, s_v, t_{o-1}} < K_{d_\lambda}^{\max} \times h_{d_\lambda, u_\mu, s_v, t_o}^{c_2, p_2} \quad (5.36)$$

To ensure that (only) one of both binary variables becomes one for *part p₁* and *part p₂*, respectively, we introducing the following Equations 5.37 and 5.38.

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} = 1 \quad (5.37)$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} = 1 \quad (5.38)$$

Each of the entire conditions (*c₁* or *c₂*) becomes *true* when both parts (*p₁* and *p₂*) of each condition are *true*. These relationships can be expressed though multiplications of the binary auxiliary variables (cf. Equation 5.39 and 5.40).

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} = h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} \quad (5.39)$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} = h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} \quad (5.40)$$

However, since these multiplications are again non-linear expressions, they require an additional transformation. The linear equivalents of Equation 5.39 are stated in Equations 5.41 to 5.43. The transformation of Equation 5.40 follows in Equations 5.44 to 5.46.

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} \leq 0 \quad (5.41)$$

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} \leq 0 \quad (5.42)$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} \leq 1 \quad (5.43)$$

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} \leq 0 \quad (5.44)$$

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} \leq 0 \quad (5.45)$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} \leq 1 \quad (5.46)$$

All of these calculations, equations, and inequations are required to express the case differentiation in Equation 5.15 as a linear equation system. By the means of this transformation we are able to relate the service provisioning of different time periods with each other. The complete linearization is presented in Model 8 and 9.

The presented optimization problem constitutes a Mixed Integer Programming (MIP) problem and, thus, belong to the complexity class *NP-hard* as described in Section 4.2.2.4. For these problems, no known algorithms exists that are able to solve all problem instances in polynomial time. Hence, NP-hard problems are often not solvable with reasonable effort, which requires heuristic approaches to solve such combinatorial problems in practical applications. A heuristic approach to solve the DCSPS is presented in the following Section.

5.4 HEURISTIC OPTIMIZATION APPROACH

Within a dynamic multi-period optimization problem, in each time period, the optimization decisions depend on the recent input parameters and on decisions made in the previous periods [45]. With the presented DCSPS, we minimize the cost for the augmentation of the existing cloud infrastructure with locally installed resource units. In Chapter 4, we focus on heuristic approaches for a single period optimization problem. In this section, we address a problem with multiple dependent periods and we analyze appropriate strategies that can be used by our heuristic approach.

The goal of our optimization model is the minimization of the total costs that depend on the fixed cost $C_{d_\lambda}^{fix}$, the hardware cost $C_{d_\lambda}^{hw}$, the variable costs for operating the resource units C_{d_λ, t_o}^{op} , the migration costs $C_{s_v}^{mig}$ for moving the service from one location to another, and the penalty costs C_{u_μ, s_v}^{pen} per service unit that is not provided to a user cluster. Regarding the different types of costs and their occurrence, we focus on two different strategies for our heuristic approach: (i) a strategy to cover as much service demand as possible to avoid penalty cost and (ii) a strategy to limit the provided resources to the average service demand to avoid over-provisioning for peak loads and, thus, reducing fixed costs for infrastructure and hardware. To address these aspects, we implement a heuristic approach for both strategies. For this approach additional variable are required:

- $l_{LAN} \in \mathbb{N}$: Lot size of a LAN for an additional quantity of services.
- $l_{MAN} \in \mathbb{N}$: Lot size of a MAN for an additional quantity of services.
- $k^{max} \in \mathbb{N}$: Maximal number of provided capacities by cloudlets (only required for second strategy).

- $k_{\text{count}}^{\max} \in \mathbb{N}$: Counter for capacities provided by cloudlets (only required for second strategy).

Furthermore, for our implementation several lists to store interim results are necessary:

- $D_{u_\mu, s_\nu}^{\text{per}}$: List of permitted data centers hinge on QoS requirements and guarantees.
- U_s^{res} : List of residual user clusters and their requested services.
- $K_{d_\lambda}^{\text{res}}$: List of data centers with their residual (unused) capacities.
- $N_{u_\mu}^{\text{res}}$: List of residual network capacities of a user cluster (incl. LAN, MAN, down link, and up link capacities).
- $V_{d_\lambda, u_\mu, s_\nu}^{\text{ass}}$: Already assigned demand w.r.t. user cluster and data center.

Algorithm 4 illustrates the corresponding pseudo code. The value of k^{\max} (cf. Line 2) defines the used strategy. The initialization process of required data structure is implemented according to Algorithm 1 presentment in Section 4.3.2.1.

The first approach DCSPS-HEU1.KOM can freely dispose over all available resources subject to the constraints given by the Models 6 and 7. This strategy trades higher fixed cost against low penalty cost, with the practical implication of a higher user satisfaction. With the second approach DCSPS-HEU2.KOM, we aim to minimize over-provisioning for single peak loads. Covering these potentially scarce events would increase the overall fixed costs, although the resources may be required in a single period. To avoid such a provisioning, we limit the resources provided by cloudlets to the average utilization over all time slots.

Although both approaches follow different strategies, they have several components in common that are explained subsequently. For the two approaches, we use a start procedure as described in Section 4.3.2 in the initial iteration. By the means of this approach, we are able to compute a solution with minimal computational effort, but with the drawback that allocation choices most likely result in locally optimal solutions [37]. However, due to their efficiency and our goal to find an appropriate strategy to link decisions between subsequent periods, we focus on these approaches. To avoid migration as well as penalty costs, from the second period on, both strategies consider the assignment decisions made in the previous time period (cf. Line 8 to 17). As described in Section 5.1 and 5.5.2, we consider three services with different migration and penalty costs in our simulation. The first service causes the highest migration and penalty cost and the second service is assumed to cause the second highest migration costs. To consider both aspects, we assign the required demand for the first and for the second service to the same resources as in the prior period (cf. Line 11). The third service is not considered,

which gives us the possibility to provide the free resources to additional demands for the two more sophisticated services. Hence, for the third service we accept additional cost to ensure that resources are available for the most expensive and restrictive service classes.

As mentioned, the two heuristic approaches differ in the number of resources they are allowed to assign. With the first approach, DCPSP-HEU1.KOM, we pursue the goal to provide as much as possible resources to cover the whole demand and to avoid penalty cost. The risk regarding this strategy is that resources are provided for a single peak load only and, thus, cause additional costs. To take peak loads into consideration, we implemented a second strategy, DCPSP-HEU2.KOM, that limits the available resources. Comparable to the exact solution approach, we assume global knowledge and calculate the average service demand per time slot for the first and the second service (cf. Line 2). Hence, the value of k^{\max} defines the upper bound for the provided resources.

Utilizing this average demand, we limit the provided resources to this value. As stated in Section 5.1 only the first service type requires cloudlets. However, we include the second service type as well to increase the number of available resources. Hence, we are able to cover demand fluctuations and avoid penalties. The resources used for all time periods are determined in the initial solution and do not change in the following periods. During the assignment process, we differentiate between remote and local resources as well as between different network types. Consequently, we are able to provide local resources even when the maximum MAN capacity of a user cluster is reached (cf. Line 28). In the subsequent section, we evaluate the applicability of our approaches.

5.5 EVALUATION

In the previous sections we described the *Dynamic Cloudlet Placement and Selection Problem* (DCPSP) and different solution approaches to solve it. In this section we evaluate and compare these approaches. First, in Section 5.5.1, we describe the basic aspects of our prototypical implementation. Subsequently, in Section 5.5.2, we explain the setup of our evaluation and give an overview of the relevant input parameters and their values. Finally, in Section 5.5.3, we compare the performance of our approaches against each other.

5.5.1 Prototypical Implementation

To evaluate our approaches, we prototypically implement them relying on the framework, the external components, and the database as described in Section 4.4.1. As earlier described, we use the object-oriented programming language Java and

Algorithm 4 Determination of a Solution for the DCSPS**Start:**

```

1:  $D_{u_\mu, s_\nu}^{\text{per}} \leftarrow \emptyset$ ,  $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow \emptyset$ ,  $V_{u_\mu, s_\nu, t_{o-1}}^{\text{ass}} \leftarrow \emptyset$ 
2:  $k^{\max} \leftarrow \text{SetMaximalProvidedCapacity}()$ ,  $k_{\text{count}}^{\max} \leftarrow 0$ 
3: for all  $t$  do
4:    $D_{u_\mu, s_\nu}^{\text{per}} \leftarrow \text{InitilizeDataCenters}()$ 
5:    $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow \text{InitilizeResidualDemand}()$ 
6:    $K_{d_\lambda}^{\text{res}} \leftarrow \text{InitilizeResidualCapacity}()$ 
7:    $N_{u_\mu}^{\text{res}} \leftarrow \text{InitilizeNetworkCapacities}()$ 
8:   // Transfer previous assignments
9:   for all  $V_{u_\mu, s_\nu, t_{o-1}}^{\text{ass}}$  do
10:    if  $v = 1$  or  $v = 2$  then
11:       $y_{d_\lambda, u_\mu, s_\nu} \leftarrow \text{CalculateLotSize}(V_{d_\lambda, u_\mu, s_\nu, t_{o-1}}^{\text{ass}}, V_{u_\mu, s_\nu}^{\text{res}})$ 
12:       $V_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{ass}} \leftarrow y_{d_\lambda, u_\mu, s_\nu}$ 
13:       $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow V_{u_\mu, s_\nu}^{\text{res}} - y_{d_\lambda, u_\mu, s_\nu}$ 
14:       $K_{d_\lambda}^{\text{res}} \leftarrow K_{d_\lambda}^{\text{res}} - y_{d_\lambda, u_\mu, s_\nu}$ 
15:       $N_{u_\mu}^{\text{res}} \leftarrow \text{CalculateResidualNetworkCapacities}(N_{u_\mu}^{\text{res}}, y_{d_\lambda, u_\mu, s_\nu})$ 
16:    end if
17:  end for
18:  // Assign open demand
19:  while  $|V_{u_\mu, s_\nu}^{\text{res}}| > 0$  or  $k_{\text{count}}^{\max} < k^{\max}$  do
20:    // Choose element for demand
21:     $\langle u_\mu, s_\nu \rangle \leftarrow \text{SelectServiceDemand}(V_{u_\mu, s_\nu}^{\text{res}})$ 
22:     $l_{\text{LAN}} \leftarrow \text{CalculateLotSizeLAN}(N_{u_\mu}^{\text{res}}, s_\nu)$ 
23:     $l_{\text{MAN}} \leftarrow \text{CalculateLotSizeMAN}(N_{u_\mu}^{\text{res}}, s_\nu)$ 
24:    // Choose element for supply
25:    if  $l_{\text{MAN}} \geq 1$  then
26:       $d_\lambda \leftarrow \text{SelectDataCenter}(D_{u_\mu, s_\nu}^{\text{per}})$ 
27:    else if ExistLocalCloudlet then
28:       $d_\lambda \leftarrow \text{SelectLocalCloudlet}(D_{u_\mu, s_\nu}^{\text{per}})$ 
29:    else
30:       $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow V_{u_\mu, s_\nu}^{\text{res}} \setminus \{u_\mu, s_\nu\}$  // no assignment possible
31:    end if
32:    // Reduce residual quantities and save assignment
33:     $y_{d_\lambda, u_\mu, s_\nu} \leftarrow \text{CalculateLotSize}(V_{u_\mu, s_\nu}^{\text{res}}, K_{d_\lambda}^{\text{res}}, N_{u_\mu}^{\text{res}})$ 
34:     $V_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{ass}} \leftarrow y_{d_\lambda, u_\mu, s_\nu}$ 
35:     $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow V_{u_\mu, s_\nu}^{\text{res}} - y_{d_\lambda, u_\mu, s_\nu}$ 
36:     $K_{d_\lambda}^{\text{res}} \leftarrow K_{d_\lambda}^{\text{res}} - y_{d_\lambda, u_\mu, s_\nu}$ 
37:     $N_{u_\mu}^{\text{res}} \leftarrow \text{CalculateResidualNetworkCapacities}(N_{u_\mu}^{\text{res}}, y_{d_\lambda, u_\mu, s_\nu})$ 
38:    if ( $d_\lambda$  isCloudlet) then  $k_{\text{count}}^{\max} += y_{d_\lambda, u_\mu, s_\nu}$  end if
39:    // Add and remove elements from various lists
40:    if ( $V_{u_\mu, s_\nu}^{\text{res}} = 0$  or  $K_{d_\lambda}^{\text{res}} = 0$  or  $l_{\text{LAN}} < 1$  or (!ExistLocalCloudlet and  $l_{\text{MAN}} = 0$ )) then  $V_{u_\mu, s_\nu}^{\text{res}} \leftarrow V_{u_\mu, s_\nu}^{\text{res}} \setminus \{u_\mu, s_\nu\}$ 
41:    end if
42:    if  $K_{d_\lambda}^{\text{res}} = 0$  then
43:      for all  $\langle u_\mu, s_\nu \rangle \in V_{u_\mu, s_\nu}^{\text{res}}$  do  $D_{u_\mu, s_\nu}^{\text{per}} \leftarrow D_{u_\mu, s_\nu}^{\text{per}} \setminus \{d_\lambda\}$  end for
44:    end if
45:  end while
end for

```

the relational database management system MySQL². We embed the additionally developed classes using the package structure explained in Section 4.4.1.1. We implement classes to generate problem instances for the DCSP and to solve the exact optimization approach DCSP-EXA.KOM as well as the heuristic approaches DCSP-HEU1.KOM and DCSP-HEU2.KOM. Furthermore, all required components such as classes to store generated problem instances within a source database and classes to run the evaluation through executable *jar* files are implemented for the new problem.

As described in Section 4.4.1.2, several external libraries are used for our implementation. First, for the exact solution approach, we use the commercial optimization engine CPLEX 12.5 provided by IBM³. To access this solver, we utilize the generic Java ILP interface⁴ in the most recent version 1.2a. To store the problem instances and the results of the evaluation we use MySQL in version 5.6.20, which is accessed by the JDBC driver in version 5.1.22MySQL Connector/J⁵.

The databases that store the generated problem instances are named regarding the problem type DCSP_DATA_[IDENTIFIER]. With the *identifier*, we are able to separate between different source datasets. These source databases contain tables to define the test cases and their problem instances. The basic characteristics of a test case such as case name, a unique identifier, and the quantity of each basic entity are stored in the table TEST_CASE. The corresponding problem instances for each test case are stored in the table PROBLEM. This table includes columns for the problem name, the unique identifier of each problem, and the test case identifier to connect a test case with its corresponding problem instances. Regarding the basic entities and their parameters, we further use the following tables to store the associated information: DATA_CENTER, DATA_CENTER_COST_VAR, USER_CLUSTER, QOS_ATTRIBUTE, QOS_GUARANTEE, QOS_REQUIREMENT, SERVICE, SERVICE_DEMAND, SERVICE_PENALTY. By the means of such a database, we are able to store all required parameters as described within the formal notations (cf. Section 5.3.1), e. g., network capacities, hardware costs, or the characteristics of the assessed services. All stored problem instances are generated based on the predefined configurations as described in the subsequent Section 5.5.2.

Apart from the source database we use the database DCSP_LOGS_[IDENTIFIER] to store the evaluation results. Since we use the same performance measures as described in Chapter 4, i. e., computation time and cost ratio, we rely on the same data base structure as described in Section 4.4.1.3. In the first table TEST_CASES we store information as test case name and the identifier. In the second table, PROBLEMS, we the store the measured evaluation results for each algorithm.

² <https://www.mysql.com/>

³ <http://www-03.ibm.com/software/products/de/ibmilogcpleoptistud>

⁴ <https://sourceforge.net/projects/javailp/files/>

⁵ <https://dev.mysql.com/downloads/connector/j/>

5.5.2 Experimental Setup

In this section we describe the experimental setup for our evaluation. As already stated in Section 4.4.2, a proper evaluation setup requires a proper declaration of independent and dependent variables. The independent variables that are controlled by a researcher, are defined a priory to an experiment and directly influence its results [95, 127]. Here, they include the number of basic entities, such as data centers and the corresponding parameters. Dependent variables represent the measured outcome of an experiment [127]. In accordance with Silver [161], we focus on the performance measures computational effort and cost ratio. To derive statements with theoretical and practical relevance, we use realistic values for all required parameters conducted by a literature review.

Additionally, the computational effort associated with the calculation of the exact solution needs to be considered while determining the test cases to be evaluated. To derive statements about the solution quality of the analyzed heuristic, the solution of the exact approach is required. Due to the fast-growing number of decision variables when considering multiple time periods, we focus on smaller problem instances and derive statements for scenarios with a large amount of entities.

To generate test cases, we have to define the number of basic entities and the values of the corresponding parameters. As described in the formal notations (cf. Section 5.3.1), the model consists of five basic entities: (i) data centers, (ii) user clusters, (iii) services, (iv) QoS attributes, and (v) discrete time slots. Regarding the number of data centers, we differentiate between remote cloud data centers and locally installed cloudlets. As representation for the cloud we choose a single remote data center, which could be the closest to a considered metropolitan area. We augment the global cloud infrastructure with local compute units. As described in Section 5.2.1 possible locations to install cloudlets are local businesses, such as cafes or restaurants.⁶

In our model, each café forms a user cluster determined by the range of its Wi-Fi. Since each single café may host computation resources, we assume an equal number of (local) data centers and user clusters. As illustrated in Figure 13, even a single café chain may provide over 200 stores only in Manhattan [124]. Including other restaurants in metropolitan areas, hundreds or even thousands possible cloudlet locations exist. Due to the complexity of the problem, we are required to focus on a substantially smaller number of considered entities. Therefore, to achieve results with reasonable computation effort, we limit the number of local data centers and user cluster to 20 each (plus an additional cloud data center). A detailed

⁶ For the sake of readability in the following descriptions and explanations, we use cafés as a representation for local businesses that provide Wi-Fi hotspots to their customers and, thus, that are possible locations for deploying cloudlets.

analysis of the impact of the number of data centers and user clusters regarding the performance of heuristics is given in Section 4.4.4.

Data centers are characterized by their capacity, costs, and QoS guarantees. Within the model we distinguish between two types of data centers: (i) large remote cloud data centers and (ii) small local data centers, i. e., cloudlets. According to the commonly agreed cloud characteristics (cf. Section 2.1.1), we assume that the remote data center provides sufficient resources to be able to cover the entire service demand according to its provided QoS guarantees. Therefore, we set the number of available resources, i. e., the maximal capacity $K_{d_\lambda}^{\max}$, substantially higher than the demand of a (single) considered city. We set this value to 80.000 resource units, which corresponds to the findings in Section 4.4.2. Since we augment the existing cloud infrastructure by local resources, we assume that (remote) cloud resources already exist and no additional cost for construction accrue. Again, according to the cloud characteristics, these resources can be accessed on-demand and charged in a pay-as-you-go manner. Consequently, we set the minimum capacity constraint $K_{d_\lambda}^{\min}$ for this data center to zero. Furthermore, for this already existing remote data center, we consider only variable unit cost.

The locations to deploy cloudlets, i. e., cafés, are limited in their capacity to host hardware resources. The reason of this restriction lies in limited space for IT equipment and in infrastructure constraints such as electricity supply and cooling capacity. Thereby, we set the maximal capacity of cloudlets between 1 to 20 resource units. Consequently, if a cloudlet is deployed in a dedicated location, at least one server is required. Furthermore, we limit the maximum capacity to 20 servers, which approximately corresponds to the space provided by a single server rack.

To operate a cloudlet, we assume different costs for infrastructure, IT equipment, and operational expenses. In Section 2.2 we analyzed the related work regarding the cost arising when operating data centers and we found no consensus on the cost distribution between different cost types. Possible reasons include the size of the data centers and the assured availability level. Based on the findings in the literature, we set the fixed costs in our static model in Section 4.4.2 to account for a share of about two-thirds of the total costs and the remaining one-third is set to account for variable cost. Within this model, we roughly stick to this cost distribution. However, we separate the fixed cost into two parts: (i) fixed cost for infrastructure and (ii) fixed cost for hardware. By doing so, we can equip the cloudlets with a flexible number of servers and provide a planning model that is able to consider the demand fluctuations of multiple time slots. Hence, we are able to plan the resource provisioning to cover an average load and avoid over-provisioning when focusing on serving the entire demand.

Regarding the absolute values of these costs, we approximately use the distribution given in [64, 68]: 50% expenditures for servers, 25% for infrastructure cost, and 25% for variable operational cost.

If we assume 3,000 dollars for the price of a server used in cloud data centers [64] and a five year amortization period, we end up with annual fixed cost per server of 600 dollars. To reflect variations when acquiring hardware and infrastructure elements, we assume cost ranges. For a (cloudlet) server we set a cost range between about 550 to 650 dollars. Assuming cloudlets with medium capacity of 10 servers, we end up with an estimated range between 2,800 and 3,300 dollars annual cost for infrastructure, when taking the described cost ratios into consideration. Due to the fact that different cloudlets provide different maximal capacities, we take different fixed cost for infrastructure into consideration. If a cloudlet provides only one server, no fixed cost occurs. Up to a medium capacity of 10 servers we assume a value of 60% of the annual fixed cost for infrastructure regarding the above stated value, since, for example smaller cooling capacity is required. For higher capacities we set the fixed infrastructure cost to the described range between 2,800 and 3,300 dollars. In our model, the variable costs are defined to be fluctuating between different time slots, e. g., to reflect fluctuations in electricity prices. As stated in Section 2.2 expenses for energy are the major part of the variable cost. Since, the energy consumption additionally depends on the used hardware and the overall utilization of a server, we assume the annual variable cost per server in a range between 280 and 330 dollars. For the evaluation of test cases having more than one time slot ($t > 1$), we split this value depending on the considered discrete time slots.

Generally, for a fully equipped cloudlet, we assume higher annual unit costs compared to the unit cost in a remote data center to reflect the fact that small data centers cause relatively higher cost [175, 176]. For the resources provided by the remote data center we assume only variable unit cost. According to Section 4.4.2 we set this cost to an annual value of 1,000 dollars. Since we assume this value to be a variable cost, we split it depending on the considered discrete time slots. The last characteristic of data centers within our model are QoS guarantees. Due to readability, we describe this part along with the QoS requirements of the different services later on in this section.

To connect data centers and user cluster, we consider three different network types within our model: Wide Area Networks (WANs), Metropolitan Area Networks (MANs), and Local Area Networks (LANs) (cf. Figure 14). A WAN connects the remote cloud data center with a dedicated area, such as a city. From a modeling perspective, we treat a remote data center as a cloudlet with sufficient capacity for computation resources and network bandwidth to cover the entire demand of a single city w.r.t. the offered QoS guarantees.

Within our model, the considered cafés in a metropolitan area and, hence, the cloudlets are connected via a MAN with each other. Sticking to our example of cafés in New York City, we consider the most common network bandwidth in this area. According to the *NYC Broadband Map* published by the New York City Economic

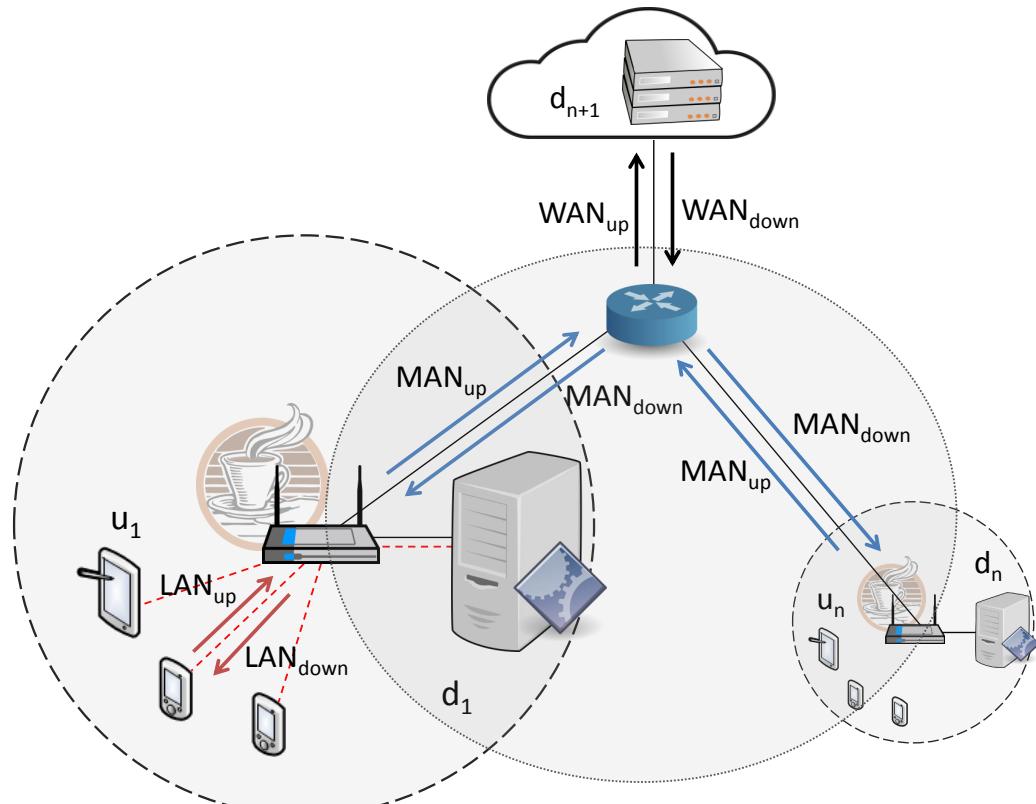


Figure 14: Integration of cloudlets within a network typology.

Development Corporation (NYCEDC) [135], in most parts of New York City an Internet connection with a symmetric bandwidth of 1 Gbps, provided via fiber channel network is available. Consequently, we assume this value for all considers user clusters for both, download capacity $K_{u_\mu}^{MAN_{down}}$ and upload capacity $K_{u_\mu}^{MAN_{up}}$. We associate these capacities with the user clusters, since the network connection is also required, if only Wi-Fi is provided to the users without an additional cloudlet.

Within the LAN the users' mobile devices are connected via Wi-Fi. The available download capacity $K_{u_\mu}^{LAN_{down}}$ and the upload capacity $K_{u_\mu}^{LAN_{up}}$ of this connection depend on various factors. First of all, the maximal possible bandwidth depends on the utilized standard, such as IEEE 802.11n⁷ or IEEE 802.11ac⁸. Furthermore, the available capacity in influenced by the number of provided access points and environmental factors, such as interferences or walls. Sticking to the most recent standard, IEEE 802.11ac, a theoretical physical layer bit rate of 6.7 Gbps is possible when using eight spatial streams with a bit rate of 867 Mbps each. However, measurements show that a bit rate of about 500 Mbps up to a distance of approximately 15 meter is more likely [42]. To increase the overall capacity within a Wireless Local Area Network (WAN), multiple Wi-Fi routers are combined. Hence,

⁷ <http://standards.ieee.org/findstds/standard/802.11n-2009.html>

⁸ <http://standards.ieee.org/findstds/standard/802.11ac-2013.html>

to take full advantage of the potential MAN capacity with a bit rate of 1 Gbps, we assume that at least two Wi-Fi routers are installed within a café. Since, depending on the store size, additional routers may be beneficial, we set a range between two and six routers per café, whereby the final number of routers deployed in a café is randomly determined in our simulation.

As illustrated in Figure 14 there is also a connection between the Wi-Fi router and the local computation resources required. However, in our model, we do not consider this hard-wired connection, since we assumed a sufficient large capacity to cover the whole traffic to provide services via Wi-Fi to the local users and to other user clusters via the MAN.

Apart from data centers and user clusters, *services* are the third entity in our model. As described in Section 5.2.1, the utilization of cloudlets can be beneficial for different types of applications, such as cloud gaming, video streaming, or even simple programs such file editing or chatting [50, 153]. Therefore, we define three different services classes for our evaluation:

1. Cloud services with high computation effort, real-time constraints, and high bandwidth requirements, e. g., cloud gaming.
2. Cloud services that require a high bandwidth, but do not necessarily have real-time constraints, e. g., on-demand video streaming.
3. Cloud services that pose low QoS requirements regarding latency or bandwidth, e. g., chat tools.

For the first class of services, we assume that cloudlets are required to achieve appropriate service quality. This class requires stringent QoS guarantees and, thus, a prioritized allocation on cloudlets. We further assume that such services are provided as premium services that cause the highest loss of profit for a provider if they are not available. Regarding the second class of services, the service quality can be enhanced by a high bandwidth offered via Wi-Fi hotspots. In case of this service class, users benefit from locally provided content by an economical utilization of smartphone resources in terms of reduced utilization of the available cellular data volume or in terms of energy consumption. For these kinds of services cloudlets are not necessarily required but offer additional benefits to the customers. Regarding the third group of services, cloudlet-enhanced infrastructures only offer marginal benefits, since these services can easily be provided by the current means of cellular networks and cloud data centers. Nevertheless, if capacities are available, these services can be provided by cloudlets. QoS attributes and network requirements of the considered services are presented in the remainder of this section. For each service, we assume a demand for resource units in a range from 1 to 20.

To reflect the characteristics of our dynamic model, we introduce two additional cost types: (i) penalty cost and (ii) migration cost. Penalty cost occurs if a provider is

not able to fulfill the promised or required service demand. Penalty cost, C_{u_μ, s_ν}^{pen} , can be interpreted in three different manners: (i) opportunity cost, (ii) cost to acquire resources on short term, or (iii) *real* penalty cost as specified within SLAs.

Opportunity cost reflects the loss of profit if a service or a product cannot be provided [172]. In such a case an infrastructure provider will not be able to earn money due to a lack of free resources and, in the worst case, lose customers. The second way to interpret penalty cost, are the expenses of a provider to acquire resources from its competitors on short term to fulfill some demands. Assuming a provider with cost-efficient resource utilization, leasing resources from other providers is more expensive than utilizing own equipment since the profit margin of the competitor also has to be paid [189]. In the last case, penalty costs could be seen as *real* penalty cost, which are specified within SLAs for violation of SLOs (cf. Section 2.3.2). For example, Google promises to their customers a monthly up time percentage of at least 99.95%. If this SLO is missed and the up time falls below 95.00%, customers are eligible to receive 50% of their monthly payment [61].

In the work at hand, we assume that a provider may completely fail to offer the demanded resources at a certain point of time. In our simulation scenario with complete knowledge, this would be the case, when it is uneconomical to provide resources to cover a single peak load. To calculate the penalty cost, we use the average unit price and multiply it by 1.2 to reflect the loss in revenue, including a 20% profit margin. Hence, we end up with the following calculation: $(3000/10 + 600 + 300) \times 1.2$. Since, we assume the first service as a premium service that requires local resources, we assume higher penalty cost and increase it again by the factor 1.2.

The second additional cost type is the migration cost $C_{s_\nu}^{mig}$. This cost is associated with moving services from one system to another. The origin of this cost can be manifold, e. g., cost for data transfer between two data centers or a negative impact of business due to downtime [84, 88]. Primarily, migration cost occurs in the context of large scale and planned migration projects, such as data migration to roll out new storage resources. Nevertheless, this type of cost also occurs for smaller scale migrations.

Related work states that the real value of migration costs is hardly to be determined due to the fact that these costs are often hidden [88]. Consequently, for the work at hand the value of migration cost can only be estimated based on data transfer prices given by public cloud computing providers. Furthermore, this cost is also highly dependent on the amount of data that needs to be migrated. As a consequence, we set different values for the migration cost for each service individually. According to Amazon's cost overview⁹ one GB transfer volume is charged with \$0.090 (whereby this value depends also on the monthly used transfer volume).

⁹ <https://aws.amazon.com/s3/pricing/>

Table 12: Characteristics of the assumed services.

	Service 1	Service 2	Service 3
Migration cost $C_{u_\mu, s_v}^{\text{mig}}$	$1.00 * C_{u_\mu}^{\text{mig}}$	$0.75 * C_{u_\mu}^{\text{mig}}$	$0.50 * C_{u_\mu}^{\text{mig}}$
Penalty cost $C_{u_\mu, s_v}^{\text{pen}}$	$1.20 * C_{u_\mu}^{\text{pen}}$	$1.00 * C_{u_\mu}^{\text{pen}}$	$1.00 * C_{u_\mu}^{\text{pen}}$
Req. download capacity $L_{s_v}^{\text{down}}$ (Mbps)	40	40	20
Req. upload capacity $L_{s_v}^{\text{up}}$ (Mbps)	10	10	20
Latency requirement $Q_{u_\mu, s_v, q_\xi}^{\text{req}}$ (ms)	50	100	250

Assuming a data volume of 500 GB to be transferred for a server with all required data, migration cost, C^{mig} , of 50 dollars occur. Since the transferred data volume highly depends on the server operation system, the service, and overall implementation, this value is only a rough estimation. To reflect different services and assess the approaches in such a heterogeneous scenario, we assume the following value for the three described service classes: $C_{s_1}^{\text{mig}} = 100\% * C^{\text{mig}}$, $C_{s_2}^{\text{mig}} = 75\% * C^{\text{mig}}$, and $C_{s_3}^{\text{mig}} = 50\% * C^{\text{mig}}$.

Apart from the cost, the provided services are also characterized by their bandwidth requirements and the QoS constraints. The bandwidth an application requires is hard to estimate. For example, network traffic caused by a video stream is substantially influenced by the used streaming strategy [145]. Such strategies are able to reduce the video quality in case of low available bandwidth. Therefore, we determine the corresponding requirements by considering the service characteristics. For the first two services, we assume high download requirements and low upload requirements as it is the case for cloud gaming or on demand video streaming. For the third service, e.g., file editing, we assume a similar amount of required download and upload bandwidth. The assumed average values per provided resource units are presented in Table 12.

In accordance with the explanations in Section 4.4.2, we focus on the QoS criterion latency. As shown in the analysis in Section 2.3.1.1, related work suggests latency between 20 milliseconds and 150 milliseconds for interactive multimedia applications [136, 165, 173]. Since first person shooter games require latencies below 75 milliseconds [14], we set the latency requirement for the first service to 50 milliseconds. The requirement for the second service is set to 100 milliseconds as an appropriate value for less latency sensitive games or desktop services [173]. Regarding applications with less stringent latency constraints, we set the latency requirements of service three to 250 milliseconds. Table 12 summarizes the specification for each service.

To assure stringent QoS constraint, we augment the cloud infrastructure by local resource units to reduce the high latencies caused by large geographical distances [104, 105]. Depending on the involved network, we assume different QoS guarantees

$Q_{d_\lambda, u_\mu, q_\xi}^{gua}$. If a service is provided by the means of locally installed computation resources, i. e., a cloudlet is installed in a café, we only consider a LAN latency of five milliseconds. If a service is provided by a cloudlet that is installed in another café within the MAN, we consider an additional latency of 35 milliseconds. Finally, if a service is provided by the remote cloud data center we add the latency caused by the WAN. We approximate this latency by the geographical distance between the user cluster, i. e., the considered café and the data center. On top of this network latency, we add the processing time within a data center. Based on the assumption that 20 milliseconds are required for processing and rendering cloud gaming content [30], we add a uniform random value between 15 and 25 milliseconds. The resulting sum of network and computation delay is the QoS guarantee a data center can provide to a specific user cluster regarding the latency.

Finally, we have to determine the number of discrete time slots. Since we assume a rapid growth in computation time for the exact solution approach, we limit this number in this work to a value of five time slots. Hence, with predefined numbers of data centers, user clusters, services, and QoS attributes, we generate five test cases in total. Using the previously described independent variables, the parameters, and their respective values, we generate 100 problem instances for each test case and we store these instances in a source database (cf. Section 5.5.1). We conduct all experiments using a workstation equipped with an Intel Xeon CPU E5-1620 v3 with 3.50 GHz and 16 GB of memory, operating under Microsoft Windows 7.

5.5.3 Comparison of all Heuristics

In this section we compare the two heuristic approaches against the exact approach to assess their practicability in the above described scenario. In Section 5.5.2, we described the decision variables of our optimization problem, relevant parameters, and their values. This dynamic problem links several static problems into an overall problem, where the decisions at each stage depend on the previous ones. While the exact approach is able to analyze all assignment decisions as a single problem, the decisions of the heuristics in each dedicated time slot depend on the decisions of the previous slot. Since we use a predefined number of services and QoS attributes, the evaluation focuses and the number of data centers, i. e., cloudlets, the number of user cluster, and the number of discrete time slots.

5.5.3.1 Discrete Time Slots

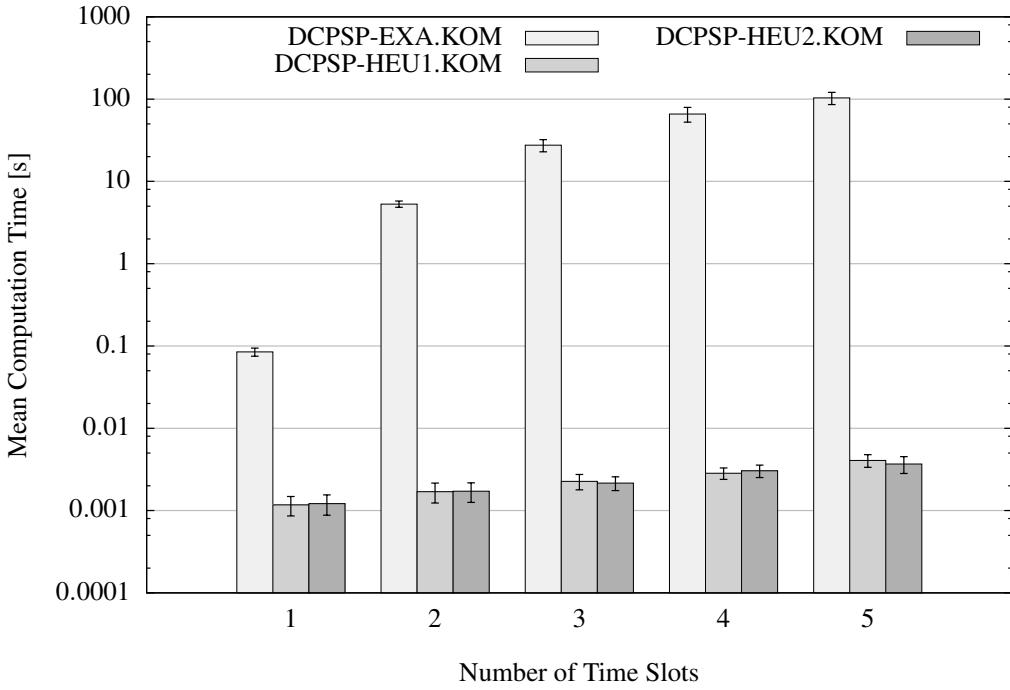
An important influencing variable in our dynamic model are the considered time slots since the decisions of related time slots depend on each other. While we evaluate the number of time slots, all remaining independent variables, i. e., number of data centers, number of user clusters, number of services, and number of attributes

are fixed. We set these values to $|D|=20$, $|U|=20$, $|S|=3$, and $|Q|=1$. As described in Section 5.4, the heuristic approaches follow two different strategies. The first one, DCPSP-HEU1.KOM, disposes freely over all available resources, only limited by the capacity constraints, i. e., resource units and bandwidth. The second approach, DCPSP-HEU2.KOM, avoids the coverage of peak loads by limiting the available resources to the average value for all time periods.

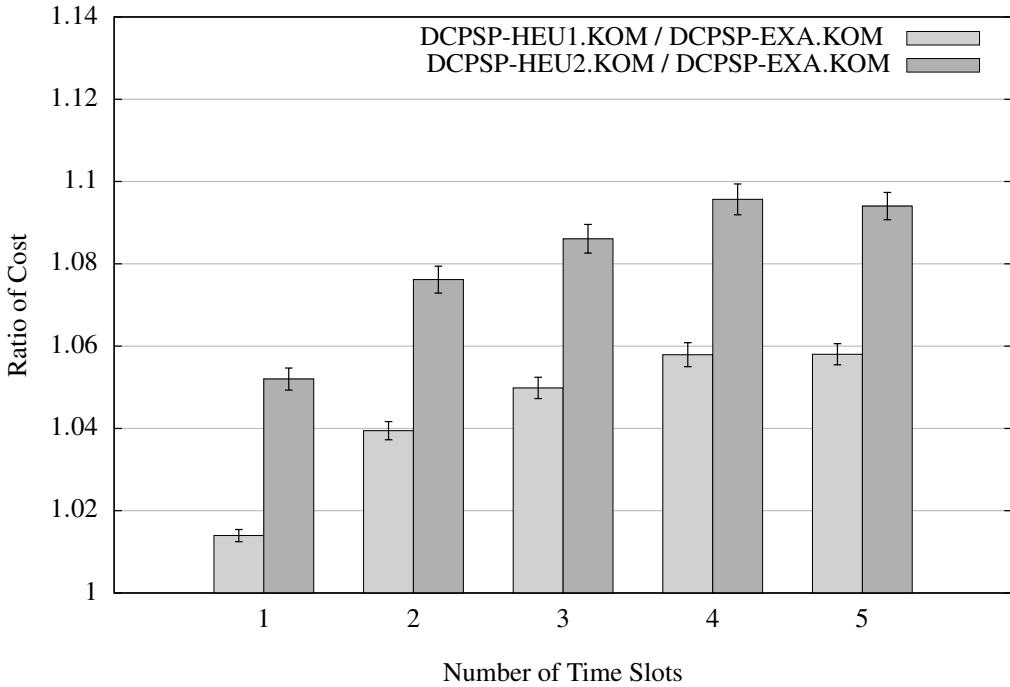
Due to the computational effort of the exact approach, we limit the overall number of time slots to five. To begin with, the computation times for all optimization approaches are given in Figure 15a. As expected, the exact approach DCPSP-EXA.KOM causes the highest computation times. Furthermore, this value is rapidly growing with an increasing number of considered time slots. For the first test case ($t=1$) a computation time of only 84 milliseconds is required. However, due to its rapid growth in computational complexity, the exact approach requires over 103 seconds for the last test case ($t=5$), which is an increase in computation time by a factor of about 1230. In contrast, the heuristic approaches, show a linear growth in computation time. Both require a similar amount of time which amounts for the approach DCPSP-HEU1.KOM by about 1.2 milliseconds for the first test case and 4.1 milliseconds for the last test case. These results reduce the computation time of about 98.6% for the first test case and over four orders of magnitude for the last one compared to the exact solution approach.

The solution quality, i. e., the cost ratios of the two heuristics are illustrated in Figure 15b. Here, two basic results are notable: (i) the first heuristic approach achieves better results than the second one and (ii) the solutions quality deteriorates with an increasing number of time slots but stabilizes from the fourth time slot on. Regarding the used strategy, limiting the resources to the average demand of service one and two is not a promising strategy for the chosen evaluation scenario. For all test cases the first heuristic (DCPSP-HEU1.KOM) is between about 3.6 and 3.8 percentage points better compared to the second one. These differences are caused by a high amount of not provided resources, i. e., high penalty cost. The reasons why the second strategy (DCPSP-HEU2.KOM) failed are manifold. One reason can be seen in the uniform distribution of the workload in a predefined range. Because of this assumption no period with an outstanding peak workload exists, in which the approach is able to show its advantages. Furthermore, the resource restriction is determined by the requested resources only; the bandwidth demand is not considered. This may lead to an underestimated number of provided locations.

The second main result of this evaluation is the depicted overall trend of the solution quality as a function of time. Focusing on the better approach (DCPSP-HEU1.KOM), it is notable that the cost ratio deteriorates with an increasing number of time slots. For the first time slot the difference to the exact approach amounts about 1.4% and it increases up to the fifth time slot to about 5.8%. However, the



(a) Observed mean computation times (with 95% confidence intervals). Please note the logarithmic scaling of the ordinate.



(b) Ratio of costs between the exact approach and heuristic approaches (with 95% confidence intervals).

Figure 15: Impact an increasing number of time slots, i. e., the evolution of the optimization problem with time.

growth between each time slot decreases notably and, thus, for the fourth and fifth time slot nearly the same solution quality is achieved. The reason here lies in the used strategy to fulfill as much demand as possible. For each provided resource unit fixed costs for the entire planning horizon occur. Since we assume a service demand in a given range, by a certain point in time no additional resources are needed or can be provided due to the given constraints, such as network bandwidth. Since these boundaries also apply for the exact solution approach we infer that the solution quality will stay at this level also for a higher number of time slots.

5.5.3.2 Data Centers and User Clusters

In this section we analyze the influence of the number of data centers and user clusters on the performance of our optimization algorithms. We increase both entities equally since each café or business location provides space for computation units and provides the resources to a user cluster through its Wi-Fi. While we increase the number of these entities, the number of the remaining entities are assumed as constant, i. e., $|S| = 3$, $|Q| = 3$, and $|T| = 3$. The computation times for all optimization approaches are given in Figure 16a. The ordinate shows the number of considered cafés and, thus, the number of considered data centers, i. e., cloudlets, and user clusters. It should be noted that only local data centers are included. The additional remote cloud data center is not included within this number.

The exact solution approach causes the highest computational effort, which ranges from about 2.8 seconds for the first test case ($|D| / |U| = 10$) to 61.9 seconds for the last evaluated test case ($|D| / |U| = 30$). Compared to the increase of computation time caused by an increased number of time slots, the influence of these entities is relatively small. However, both of our evaluated heuristics reduce the time required to find a solution by at least three orders of magnitude for the last test case. The difference between the two heuristics themselves lies in any case under a value of one millisecond and, thus, can be seen as negligible.

As mentioned in the previous subsection, the limitation of the resources in the given test cases is not a promising strategy for a good solution quality. This holds also true for different amounts of data centers/ user clusters. Because of this resource limitation of the second heuristic, the first heuristic achieves results that are between 3.4 and 3.7 percentage points better. A surprise here is the trend of the solution quality regarding an increasing number of data centers and user clusters. This performance measure slightly improves with a higher number of considered entries. While the difference between exact approach and the first heuristic approach (DCPSP-HEU1.KOM) amounts to 5.0% for the first test case ($|D| / |U| = 10$) it reduces to 3.3% for the last test case ($|D| / |U| = 30$). Here, the reasons for this result can be traced back to a higher number of available resources. Even with an increased demand, the capacity restrictions are softening since more entities are

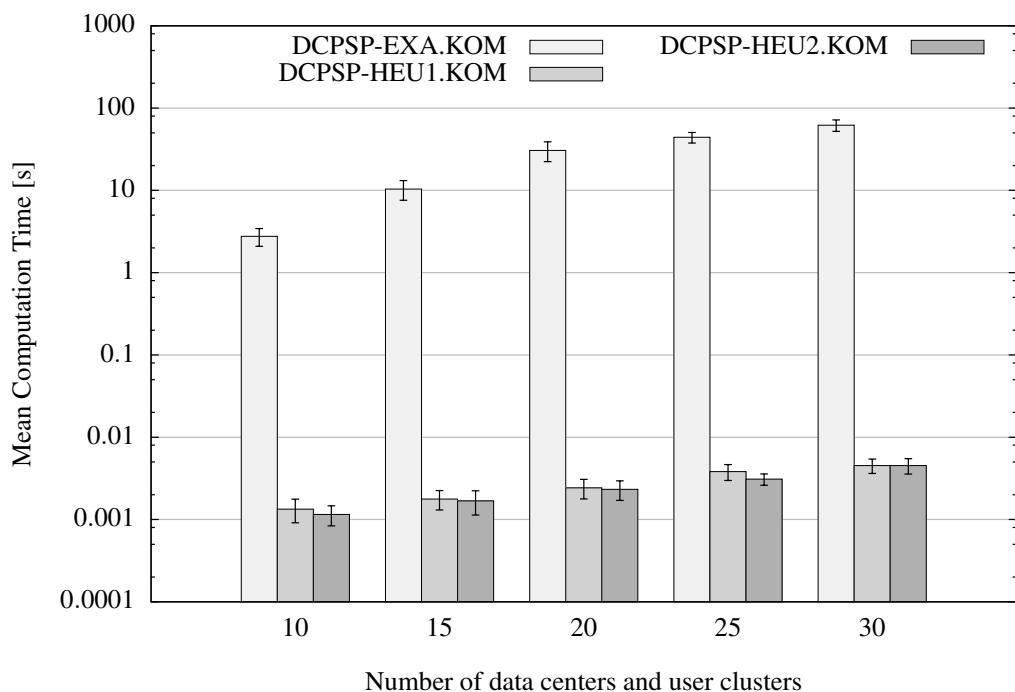
able to provide resources. Hence, the inefficient resource allocation becomes less relevant, which generally causes a reduction of penalty cost. Furthermore, with a higher number of data centers and user clusters the overall number of provided resources and, hence, the total costs increase. Therefore, single wrong assignment decisions become less influential to the cost ratio.

To summarize, our heuristic approaches significantly outperform the exact approach regarding the computation time. The achieved solution quality depends on the number of considered time slots, i. e., the time length of the problem, and number of data centers and user clusters. With increasing number of time slots the solution quality of the heuristics decreased. Nevertheless, regarding the trend of this result, we see a maximum cost increase by 6% with a stabilization of the solution quality degradation. With an increasing number of data centers and user clusters, the solution quality of our heuristics improves since the assignment decisions are easier to make due to the fact that user clusters with fluctuating demand can easier be served by a higher number of cloudlets.

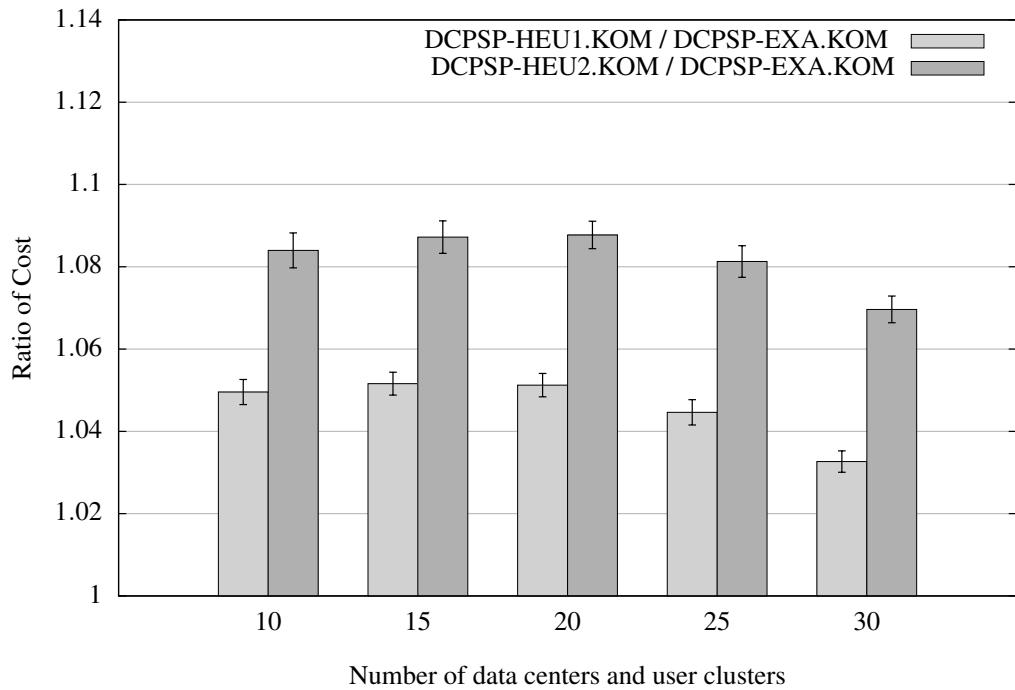
5.6 CHAPTER SUMMARY

Over the last years, the utilization of mobile devices and, thus, consumption of mobile cloud services have increased. To provide services with stringent QoS requirements, such as cloud gaming, an augmentation of the centralized cloud infrastructure by locally installed cloudlets is a beneficial approach to improve the customers' satisfaction. Such an approach helps to overcome the technical and economical limitations of mobile devices associated with computational resources and the utilization of cellular networks. However, the augmentation of the infrastructure may be costly for a provider. Furthermore, due to capacity constrains regarding the space to host resources and limitations in the available bandwidth, services cannot be offered in an arbitrary amount.

Therefore, we examined in this chapter the Dynamic Cloudlet Placement and Selection Problem (DCPSP) to provide the means of a cost-efficient infrastructure augmentation by locally installed resource units. First, we formulated the corresponding mixed integer optimization problem, which enables us to compute the exact solution using a solver framework such as IBM CPLEX. By the means of the approach we are able to address time varying service demand, resource migration, and penalty costs. Due to the high time complexity of the exact approach, we developed two heuristic approaches that implement different strategies. The first one focuses on a resource supply that covers as much demand as possible subject to the given constraints, such as network capacities. Hence, it focuses on a minimization of penalty costs and, as a practical implication, on high user satisfaction. With the



(a) Observed mean computation times (with 95% confidence intervals). Please note the logarithmic scaling of the ordinate.



(b) Ratio of costs between the exact approach and heuristic approaches (with 95% confidence intervals).

Figure 16: Impact of different (local) data centers and user clusters

second approach, we follow the strategy to avoid the resource provisioning for peak load demands. The evaluation shows that the first heuristic approach is the most promising one in a scenario with homogeneous demands, providing resources in a cost-efficient manner. It reduces the required time to find a solution by over three orders of magnitude compared to the exact approach, while decreasing the solution quality by only 5.8% in the worst case.

6

SUMMARY AND OUTLOOK

In this final chapter, we summarize the work at hand and emphasize our major contributions and key findings. In Section 6.1, we briefly summarize the content of the previous chapters, present the main contributions, and the obtained results. Section 6.2 provides an outlook on potential future work.

6.1 SUMMARY

In Chapter 1 we motivated and explained the research challenges. Furthermore, we outlined our main contributions. To achieve our research goal, i. e., a cost-efficient cloud resource provisioning in heterogeneous environments, the work at hand provides two optimization approaches on a global and on a local level.

In Chapter 2 we provided an overview of the cloud computing paradigm, including its characteristics, the service models, and deployments models. Based on this foundation, we analyzed the costs and pricing schemes for different cloud service models and the involved market participants. Afterwards, we outlined the most relevant Quality of Service (QoS) attributes in the context of multimedia communication. In Chapter 3 we presented and discussed related work and closely related research topics. We presented current approaches for data center and cloudlet placement, as well as, approaches for resource allocation with the focus on cost and QoS parameters.

Chapter 4 presents our first main contribution. Here, we focused on the issue of QoS-aware and cost-efficient resource provisioning in a global context. With our first contribution, we address the trade-off between cost minimal service provisioning and the quality requirements of multimedia services that are facilitated by cloud infrastructure providers. We formulated the underlying Cloud Data Center Selection Problem (CDCSP) as a mathematical optimization problem. We prototypically implemented the exact solution approach (CDCSP-EXE.KOM) that has been solved using the commercial solver framework CPLEX provided by IBM. The resulting mixed integer problem exhibits NP-hardness and exponential time complexity. In consequence, the optimal solution is suitable only for small problem instances. To solve the problem efficiently, we designed suitable heuristic approaches for the optimization problem.

As the first heuristic optimization approach we used a linear program relaxation (CDCSP-REL.KOM). Therefore, the mixed integer program was transferred to

a linear program that can be solved by more efficient algorithms, as provided by the IBM CPLEX solver framework. Since this approach is not fitted to the specific characteristics of our problem, it performs poor regarding both performance measures, solution quality and computation time.

Consequently, we developed and examined three customized approaches: (i) priority-based heuristics (CDCSP-PBSH.KOM), (ii) a best-of-breed approach (CDCSP-BoB.KOM), and (iii) tabu search (CDCSP-Tabu.KOM). To begin with, we used priority-based heuristics which conduct a resource assignment based on predefined priority rules. We developed and analyzed various priority and cost allocation rules. By the means of a comprehensive evaluation, we determined a configuration that most likely delivers the best results for different problem instances. To realize further improvements in solution quality, we analyzed the results of over 500,000 measurements for over 300 configurations of the priority-based start heuristics using statistical testing methods. The best heuristics found within this procedure were bundled to the *Best-of-Breed* approach. This approach runs the selected heuristics subsequently and uses the best result of the included heuristics as the final one. As the most sophisticated approach, we implemented and examined the metaheuristic tabu search. In addition, all optimization approaches were evaluated in a quantitative manner. Using realistic input data based on scientific studies, we generated multiple test cases considering different independent variables, such as the number of data centers, user clusters, and services. We found that all heuristic approaches significantly reduce the required computation time compared to the exact approach. We also observed the deterioration in solution quality, where the Linear Program relaxed approach delivers the worst results regarding this performance measure. The tabu search heuristic provides a significant enhancement in solution quality across all considered test cases. Additionally, even for large problem instances it requires only minimal more computation time compared to all other heuristic optimization approaches.

Our second main contribution in Chapter 5 deals with the *Dynamic Cloudlet Placement and Selection Problem (DCPSP)*. In this model we augmented the global cloud infrastructure with local resources to provide higher service quality (e.g., w.r.t. end-to-end latency) and to efficiently provide mobile cloud services via local area networks. This model considers different network types and a fluctuating service demand. We formalize the DCPSP as a mathematical model. Considering multiple time periods, the exact optimization approach (DCPSP-EXA.KOM) is suitable only for small problem instances and it is hardly applicable in real-world scenarios. Therefore, we developed a heuristic multi-period approach. The evaluation results show a significant reduction of the computation time of over 99.9% by a reduction of the solution quality by only about 5.8%. Hence, we showed the practical applicability for this approach for large problem instances.

In summary, this thesis provides approaches for resource provisioning in heterogeneous environments. First, we provide a static approach for large and centralized data centers on a global scale. Second, we contribute a dynamic approach for an augmented infrastructure that also includes regionally provided resources. With the presented heuristic optimization approaches we find a reasonable trade-off between solution quality and required computational effort. This allows a infrastructure provider to analyze and implement an efficient resource planning over multiple time spans with manageable effort.

6.2 OUTLOOK

Based on the work at hand we identify several research directions. The first possible research direction refers to the number of considered decision variables. In environments with multiple data centers and user clusters, the number of decision variables can easily reach several millions. Especially for future scenarios, with an increasing number of units that might offer computational power, such as router and switches, even for the fastest presented heuristics, the time required to calculate a solution might be too high for practical applications. Therefore, algorithms that pool data center and user cluster might be required. The same holds true for data centers. For example, well connected cloudlets with similar characteristics, could be aggregated to larger virtual data centers and, thus, the amount of different assignment possibilities can be reduced. A second research direction, with regard to the dynamic approach, is the aggregation of time periods to reduce their number and, thus, decreasing the computational effort. Here, an interesting research question is the appropriate length of these time intervals to find a good trade-off between computational effort and an efficient resource provisioning for a given time.

Our third identified research direction concerns prior knowledge about the service demand. In our model we assume to have complete knowledge of the service demand in each period of time. This approach is very beneficial for evaluating the performance of the heuristic approaches and to select appropriate heuristics for the optimization. Nevertheless, in real market situations, a provider is only able to estimate the service demand with a certain probability based on historical data. Therefore, an interesting research challenge might be to develop efficient algorithms that consider such demand estimates.

BIBLIOGRAPHY

- [1] Sharad Agarwal, John Dunagan, Navendu Jain, Stefan Saroiu, Alec Wolman, and Harbinder Bhogan. "Volley: Automated Data Placement for Geo-Distributed Cloud Services." In: *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI)*. 2010, pp. 1–16.
- [2] Vaneet Aggarwal, Xu Chen, Vijay Gopalakrishnan, Rittwik Jana, K. K. Ramakrishnan, and Vinay A. Vaishampayan. "Exploiting Virtualization for Delivering Cloud-based IPTV Services." In: *Proceedings of the 30th International Conference on Computer Communications Workshops (INFOCOM)*. 2011, pp. 637–641.
- [3] Shmuel Agmon. "The Relaxation Method for Linear Inequalities." In: *Canadian Journal of Mathematics* 6 (1954), pp. 382–392.
- [4] Amazon. *AWS Global Infrastructure*. Online. 2017. URL: <https://aws.amazon.com/about-aws/global-infrastructure/>.
- [5] Spyros Angelopoulos and Allan Borodin. "The Power of Priority Algorithms for Facility Location and Set Cover." In: *Algorithmica* 40.4 (2004), pp. 271–291.
- [6] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. *Above the Clouds: A Berkeley View of Cloud Computing*. Tech. rep. Electrical Engineering and Computer Sciences Department, University of California, Berkeley, 2009.
- [7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. "A View of Cloud Computing." In: *Communications of the ACM* 53.4 (2010), pp. 50–58.
- [8] Stefan Badertscher, Frank Neugebauer, and Daniel Meier. "Economic Challenges and New Opportunities Using Cloud Computing." In: *Internet Economics V*. Ed. by Burkhard Stiller, Fabio Hecht, Maurizio Lo Bosco, Peter Racz, Guilherme Machado, and Andrei Vancea. University of Zurich, 2010, pp. 155–178.
- [9] Werner Ballhaus, Niklas Wilke, Bin Song, Tobias Gräber, Maria Popova, and Friedrich-Alexander Meyer. *Media Trend Outlook 2015 – Cloud Gaming: Vielseitiger Einfluss auf die Videospiel-Industrie*. Online. 2015. URL: https://www.pwc.de/de/technologie-medien-und-telekommunikation/assets/pwc-media-trend-outlook_cloud-gaming.pdf.

- [10] Gaurav Baranwal and Deo Prakash Vidyarthi. "A Fair Multi-attribute Combinatorial Double Auction Model for Resource Allocation in Cloud Computing." In: *Journal of Systems and Software* 108 (2015), pp. 60–76.
- [11] Eric Bauer and Randee Adams. *Reliability and Availability of Cloud Computing*. John Wiley & Sons, 2012.
- [12] Christian Baun, Marcel Kunze, Jens Nimis, and Stefan Tai. *Cloud Computing: Web-based Dynamic IT Services*. Springer, 2011.
- [13] Mark Bedner and Tobias Ackermann. "Schutzziele der IT-Sicherheit." In: *Datenschutz und Datensicherheit* 34.5 (2010), pp. 323–328.
- [14] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003®." In: *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games*. 2004, pp. 144–151.
- [15] Daniel Beimborn, Thomas Miletzki, and Stefan Wenzel. "Platform as a Service (PaaS)." In: *Wirtschaftsinformatik* 53.6 (2011), pp. 371–375.
- [16] Umesh Bellur, Arpit Malani, and Nanjangud C. Narendra. "Cost Optimization in Multi-site Multi-cloud Environments with Multiple Pricing Schemes." In: *Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD)*. IEEE, 2014, pp. 689–696.
- [17] Rainer Berbner. "Dienstgüteunterstützung für Service-orientierte Workflows." PhD thesis. Technische Universität Darmstadt, 2007.
- [18] Josep L. Bernal, Íñigo Goiri, Thu D. Nguyen, Ricard Gavaldá, Jordi Torres, and Ricardo Bianchini. "Building Green Cloud Services at Low Cost." In: *Proceeding of the 34th IEEE International Conference on Distributed Computing Systems*. 2014, pp. 449–460.
- [19] Ofer Biran, Antonio Corradi, Mario Fanelli, Luca Foschini, Alexander Nus, Danny Raz, and Ezra Silvera. "A Stable Network-aware VM Placement for Cloud Systems." In: *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 2012, pp. 498–506.
- [20] Robert E. Bixby. "Solving Real-World Linear Programs: A Decade and More of Progress." In: *Operations Research* 50.1 (2002), pp. 3–15.
- [21] Andreas Bölte. *Modelle und Verfahren zur innerbetrieblichen Standortplanung*. 4. Physica, 1994.
- [22] Allan Borodin, N. Morten Nielsen, and Charles Rackoff. "(Incremental) Priority Algorithms." In: *Algorithmica* 37.4 (2003), pp. 295–326. ISSN: 1432-0541.

- [23] Peter Buxmann, Heiner Diefenbach, and Thomas Hess. *Die Softwareindustrie: Ökonomische Prinzipien, Strategien, Perspektiven*. Springer, 2015.
- [24] Rajkumar Buyya, Chee Shin Yeo, Srikuamar Venugopal, James Broberg, and Ivona Brandic. "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility." In: *Future Generation Computer Systems* 25.6 (2009), pp. 599–616.
- [25] Nicolò Maria Calcavecchia, Ofer Biran, Erez Hadad, and Yosef Moatti. "VM Placement Strategies for Cloud Scenarios." In: *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD)*. 2012, pp. 852–859.
- [26] Alberto Ceselli, Marco Premoli, and Stefano Secci. "Cloudlet Network Design Optimization." In: *Proceedings of the IFIP Networking Conference (IFIP Networking)*. 2015, pp. 1–9.
- [27] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. "Optimal Virtual Machine Placement Across Multiple Cloud Providers." In: *Proceedings of the Asia-Pacific Services Computing Conference (APSCC)*. IEEE. 2009, pp. 103–110.
- [28] Shin-Jyh Frank Chang, Susmit Harihar Patel, and James Marc Withers. "An Optimization Model to Determine Data Center Locations for the Army Enterprise." In: *Proceeding of the IEEE Military Communications Conference*. 2007, pp. 1–8.
- [29] Min Chen, Yixue Hao, Yong Li, Chin-Feng Lai, and Di Wu. "On the Computation Offloading at Ad Hoc Cloudlet: Architecture and Service Modes." In: *IEEE Communications Magazine* 53.6 (2015), pp. 18–24.
- [30] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. "The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-User Latency." In: *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*. 2012.
- [31] Cisco. *Design Best Practices for Latency Optimization*. Tech. rep. Cisco Systems, Inc., 2007. URL: https://www.cisco.com/application/pdf/en/us/guest/netsol/ns407/c654/ccmigration_09186a008091d542.pdf.
- [32] Cisco. *Cisco Global Cloud Index: Forecast and Methodology, 2015-2020*. Online. 2016. URL: <http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf>.
- [33] Citrix. *2015 Desktops-as-a-Service global market trends: The service provider perspective*. Online. 2015. URL: https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/daas-global-market-trends.pdf.

- [34] Jack Clark. *5 Numbers That Illustrate the Mind-Bending Size of Amazon's Cloud*. Online. 2014. URL: <https://www.bloomberg.com/news/2014-11-14/5-numbers-that-illustrate-the-mind-bending-size-of-amazon-s-cloud.html>.
- [35] Mark Claypool and Kajal Claypool. "Latency and Player Actions in Online Games." In: *Communications of the ACM* 49.11 (2006), pp. 40–45.
- [36] Reiner Clement and Dirk Schreiber. *Internet-Ökonomie: Grundlagen und Fallbeispiele der vernetzten Wirtschaft*. Springer, 2013.
- [37] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.
- [38] Frank Dammeyer and Stefan Voß. "Dynamic Tabu List Management Using the Reverse Elimination Method." In: *Annals of Operations Research* 41.1-4 (1993), pp. 31–46.
- [39] Data Center Knowledge. *Google Data Center FAQ*. Online. 2017. URL: <http://www.datacenterknowledge.com/archives/2017/03/16/google-data-center-faq/>.
- [40] Lien Deboosere, Bert Vankeirsbilck, Pieter Simoens, Filip De Turck, Bart Dhoedt, and Piet Demeester. "Cloud-Based Desktop Services for Thin Clients." In: *IEEE Internet Computing* 16.6 (2012), pp. 60–67.
- [41] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. "WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance." In: *Proceedings of the Internet Measurement Conference (IMC)*. 2014.
- [42] Mihaela-Diana Dianu, Janne Riihijarvi, and Marina Petrova. "Measurement-Based Study of the Performance of IEEE 802.11ac in an Indoor Environment." In: *Proceedings of the IEEE International Conference on Communications (ICC)*. 2014, pp. 5771–5776.
- [43] Marios D. Dikaiakos, Dimitrios Katsaros, Pankaj Mehra, George Pallis, and Athena Vakali. "Cloud Computing: Distributed Internet Computing for IT and Scientific Research." In: *IEEE Internet Computing* 13.5 (2009), pp. 10–13.
- [44] Savio Dimatteo, Pan Hui, Bo Han, and Victor Li. "Cellular Traffic Offloading Through WiFi Networks." In: *Proceedings of the IEEE 8th International Conference on Mobile Adhoc and Sensor Systems*. 2011.
- [45] Wolfgang Domschke, Drexl Andreas, Robert Klein, and Armin Scholl. *Einführung in Operations Research*. Springer Gabler, 2015.
- [46] Schahram Dustdar, Harald Gall, and Manfred Hauswirth. *Software-architekturen für verteilte Systeme: Prinzipien, Bausteine und Standardarchitekturen für moderne Software*. Springer, 2013.

- [47] Schahram Dustdar, Yike Guo, Benjamin Satzger, and Hong-Linh Truong. "Principles of Elastic Processes." In: *IEEE Internet Computing* 15.5 (2011), pp. 66–71.
- [48] Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer, and Schahram Dustdar. "Low Level Metrics to High Level SLAs-LoM2HiS Framework: Bridging the Gap Between Monitored Metrics and SLA Parameters in Cloud Environments." In: *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS)*. 2010, pp. 48–54.
- [49] Sifat Ferdousi, Ferhat Dikbiyik, M. Farhan Habib, Massimo Tornatore, and Biswanath Mukherjee. "Disaster-Aware Datacenter Placement and Dynamic Content Management in Cloud Networks." In: *IEEE Journal of Optical Communications and Networking* 7.7 (2015), pp. 681–694.
- [50] Debessay Fesehaye, Yunlong Gao, Klara Nahrstedt, and Guijun Wang. "Impact of Cloudlets on Interactive Mobile Cloud Applications." In: *Proceedings of the 16th IEEE International Conference on Enterprise Distributed Object Computing Conference*. 2012.
- [51] Gerhard P. Fettweis. "The Tactile Internet: Applications and Challenges." In: *IEEE Vehicular Technology Magazine* 9.1 (2014), pp. 64–70.
- [52] Simson L. Garfinkel. "The Cloud Imperative." In: *Technology Review* 114.6 (2011), pp. 74–75.
- [53] Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya. "SMICloud: A Framework for Comparing and Ranking Cloud Services." In: *Proceedings of the fourth IEEE International Conference on Utility and Cloud Computing (UCC)*. 2011, pp. 210–218.
- [54] Michel Gendreau, Alain Hertz, and Gilbert Laporte. "A Tabu Search Heuristic for the Vehicle Routing Problem." In: *Management Science* 40.10 (1994), pp. 1276–1290.
- [55] Martin Glinz. "On Non-Functional Requirements." In: *Proceedings of the 15th IEEE International Requirements Engineering Conference*. 2007.
- [56] Fred Glover. "Future Paths for Integer Programming and Links to Artificial Intelligence." In: *Computers & Operations Research* 13.5 (1986), pp. 533–549.
- [57] Fred Glover. "Tabu Search – Part I." In: *ORSA Journal on Computing* 1.3 (1989), pp. 190–206.
- [58] Fred Glover, Manuel Laguna, and Rafael Marti. "Principles of Tabu Search." In: *Approximation Algorithms and Metaheuristics* 23 (2007), pp. 1–12.
- [59] Fred Glover, Eric Taillard, and Dominique de Werra. "A User's Guide to Tabu Search." In: *Annals of Operations Research* 41.1–4 (1993), pp. 3–28.

- [60] Íñigo Goiri, Kien Le, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. "Intelligent Placement of Datacenters for Internet Services." In: *Proceedings of the 31st International Conference on Distributed Computing Systems*. 2011.
- [61] Google. *Google Compute Engine Service Level Agreement (SLA)*. Online. 2016. URL: <https://cloud.google.com/compute/sla>.
- [62] Google. *Standorte von Rechenzentren*. Online. 2017. URL: <https://www.google.com/about/datacenters/inside/locations/index.html>.
- [63] Dimitris Gouscos, Manolis Kalikakis, and Panagiotis Georgiadis. "An Approach to Modeling Web Service QoS and Provision Price." In: *Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE)*. 2003, pp. 121–130.
- [64] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. "The Cost of a Cloud: Research Problems in Data Center Networks." In: *ACM SIGCOMM Computer Communication Review* 39.1 (2008), pp. 68–73.
- [65] Tore Grünert and Stefan Irnich. *Optimierung im Transport*. Shaker, 2005.
- [66] Abhishek Gupta, Laxmikant V. Kale, Dejan Milojicic, Paolo Faraboschi, and Susanne M. Balle. "HPC-aware VM Placement in Infrastructure Clouds." In: *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*. 2013, pp. 11–20.
- [67] Francisco Guzmán, Temi Abimbola, and Frank Linde. "Pricing Information Goods." In: *Journal of Product & Brand Management* 18.5 (2009), pp. 379–384.
- [68] James Hamilton. "Cooperative Expendable Micro-Slice Servers (CEMS): Low Cost, Low Power Servers for Internet-Scale Services." In: *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*. 2009.
- [69] Ronny Hans. "Selecting Cloud Data Centers for QoS-Aware Multimedia Applications." In: *Proceedings of the PhD Symposium at the 2nd European Conference on Service-Oriented and Cloud Computing (ESOCC)*. 2013, pp. 11–16.
- [70] Ronny Hans, Ulrich Lampe, and Ralf Steinmetz. "QoS-Aware, Cost-Efficient Selection of Cloud Data Centers." In: *Proceedings of the 6th IEEE International Conference on Cloud Computing (CLOUD)*. 2013, pp. 946–947.
- [71] Ronny Hans, Ulrich Lampe, Michael Pauly, and Ralf Steinmetz. "Cost-efficient Capacitation of Cloud Data Centers for QoS-aware Multimedia Service Provision." In: *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER)*. 2014, pp. 158–163.
- [72] Ronny Hans, Ulrich Lampe, Daniel Burgstahler, Martin Hellwig, and Ralf Steinmetz. "Where Did My Battery Go? Quantifying the Energy Consumption of Cloud Gaming." In: *Proceedings of the 3rd IEEE International Conference on Mobile Services (MS)*. 2014, pp. 63–67.

- [73] Ronny Hans, David Steffen, Ulrich Lampe, Björn Richerzhagen, and Ralf Steinmetz. "Setting Priorities: A Heuristic Approach for Cloud Data Center Selection." In: *Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER)*. 2015, pp. 221–228.
- [74] Ronny Hans, David Steffen, Ulrich Lampe, Dominik Stingl, and Ralf Steinmetz. "Short Run: Heuristic Approaches for Cloud Resource Selection." In: *Proceedings of the IEEE 9th International Conference on Cloud Computing (CLOUD)*. 2016, pp. 569–576.
- [75] Ronny Hans, Melanie Holloway, The An Binh Nguyen, Alexander Müller, Marco Seliger, and Jürgen Lang. *Cloud-Applikationen in der Finanzindustrie - Dienstgütemerkmale und deren Überwachung*. Tech. rep. Technische Universität Darmstadt, 2017.
- [76] Ronny Hans, Björn Richerzhagen, Amr Rizk, Ulrich Lampe, Ralf Steinmetz, Sabrina Klos, and Anja Klein. "Little Boxes: A Dynamic Optimization Approach for Enhanced Cloud Infrastructures." In: *Proceedings of the 7th European Conference on Service-Oriented and Cloud Computing (ESOCC)*. 2018.
- [77] Ronny Hans, Björn Richerzhagen, Amr Rizk, Ulrich Lampe, Ralf Steinmetz, Sabrina Klos, and Anja Klein. *Little Boxes: A Dynamic Optimization Approach for Enhanced Cloud Infrastructures*. Tech. rep. 1807.02615. Technische Universität Darmstadt, 2018. URL: <https://arxiv.org/abs/1807.02615>.
- [78] Sabbir Hasan and Eui-Nam Huh. "Heuristic based Energy-aware Resource Allocation by Dynamic Consolidation of Virtual Machines in Cloud Data Center." In: *Transactions on Internet and Information Systems* 7.8 (2013), pp. 1825–1842.
- [79] Mohammad Mehedi Hassan, M Shamim Hossain, AM Jehad Sarkar, and Eui-Nam Huh. "Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform." In: *Information Systems Frontiers* 16.4 (2014), pp. 523–542.
- [80] Robert Heininger, Holger Wittges, and Helmut Krcmar. "Literaturrecherche zu IT-Servicemanagement im Cloud Computing." In: *HMD Praxis der Wirtschaftsinformatik* 49.6 (2012), pp. 15–23.
- [81] Alain Hertz and Dominique de Werra. "Using Tabu Search Techniques for Graph Coloring." In: *Computing* 39.4 (1987), pp. 345–351.
- [82] Frederick Hillier and Gerald Lieberman. *Introduction to Operations Research*. 9th. McGraw-Hill, 2010.

- [83] Ralph Hintermann and Jens Clause. *Rechenzentren in Deutschland: Eine Studie zur Darstellung der wirtschaftlichen Bedeutung und der Wettbewerbssituation.* Tech. rep. Borderstep Institut, 2014. URL: https://www.bitkom.org/files/documents/Borderstep_Institut_-_Studie_Rechenzentren_in_Deutschland_05-05-2014%281%29.pdf.
- [84] Hitachi Data Systems. *Reduce Costs and Risks for Data Migrations.* Tech. rep. Hitachi Data Systems, 2015.
- [85] Tobias Hoßfeld, Raimund Schatz, Martin Varela, and Christian Timmerer. "Challenges of QoE Management for Cloud Applications." In: *IEEE Communications Magazine* 50.4 (2012), pp. 28–36.
- [86] Melanie Holloway. "Service Level Management in Cloud Computing: Pareto-Efficient Concurrent Multiple-Issue Negotiations, Reliable Consumer-Side Availability Monitoring and Strategies for Robust Monitor Placement." PhD thesis. Technische Universität Darmstadt, 2016.
- [87] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. "A Close Examination of Performance and Power Characteristics of 4G LTE Networks." In: *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys).* 2012, pp. 225–238.
- [88] IBM. *The Hidden Costs of Data Migration.* Tech. rep. IBM Global Technology Services, 2007. URL: <https://www-935.ibm.com/services/us/gts/pdf/hiddencostsdatamigration-wp-gtw01279-usen-01-121307.pdf>.
- [89] IBM. *IBM Completes Acquisition of ILOG.* Online. 2009. URL: <http://www-03.ibm.com/press/us/en/pressrelease/26403.wss>.
- [90] IBM. *A Brief History of Cloud.* Online. 2015. URL: <https://www.ibm.com/blogs/cloud-computing/2015/04/a-brief-history-of-cloud-1950-to-present-day/>.
- [91] IBM. *IBM ILOG CPLEX Optimization Studio V12.7.0 Documentation.* Online. 2016. URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html.
- [92] IEEE. "Standard Glossary of Software Engineering Terminology." In: *IEEE Standard 610.12-1990.* 1990.
- [93] Internap. *Latency: The Achilles Heel of Cloud Computing.* Online. 2010. URL: http://tomx.inf.elte.hu/twiki/pub/Team/CloudComputing/1_14241_WP-IP-Achilles-Heel-Latency.pdf.
- [94] Michael Isard. "Autopilot: Automatic Data Center Management." In: *ACM SIGOPS Operating Systems Review* 41.2 (2007), pp. 60–67.

- [95] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991.
- [96] Joe Wenjie Jiang, Tian Lan, Sangtae Ha, Minghua Chen, and Mung Chiang. "Joint VM Placement and Routing for Data Center Traffic Engineering." In: *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM)*. 2012, pp. 2876–2880.
- [97] Sravanthi Kalepu, Shonali Krishnaswamy, and Seng Wai Loke. "Verity: A QoS Metric for Selecting Web Services and Providers." In: *Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE)*. 2003, pp. 131–139.
- [98] Matthias Keller, Stefan Pawlik, Peter Pietrzk, and Holger Karl. "A Local Heuristic for Latency-optimized Distributed Cloud Deployment." In: *Proceedings of the IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC)*. 2013, pp. 429–434.
- [99] Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. "A pareto-based Genetic Algorithm for Optimized Assignment of VM Requests on a Cloud Brokering Environment." In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. 2013, pp. 2496–2503.
- [100] Roger E Kirk. *Statistics: An Introduction*. Belmont, CA, USA: Thomson Wadsworth, 2008.
- [101] Jonathan Koomey, Kenneth Brill, Pitt Turner, John Stanley, and Bruce Taylor. *A Simple Model for Determining True Total Cost of Ownership for Data Centers*. Tech. rep. Uptime Institute, 2007, p. 2007.
- [102] Vineet Kumar. "Making "Freemium" Work." In: *Harvard Business Review* 92.5 (2014), pp. 27–29.
- [103] Ulrich Lampe. "Monetary Efficiency in Infrastructure Clouds - Solution Strategies for Workload Distribution and Action-based Capacity Allocation." PhD thesis. Technische Universität Darmstadt, 2013.
- [104] Ulrich Lampe, Ronny Hans, and Ralf Steinmetz. "Will Mobile Cloud Gaming Work? Findings on Latency, Energy, and Cost." In: *Proceedings of the 2nd IEEE International Conference on Mobile Services (MS)*. 2013, pp. 960–961.
- [105] Ulrich Lampe, Qiong Wu, Sheip Dargutev, Ronny Hans, André Miede, and Ralf Steinmetz. "Assessing Latency in Cloud Gaming." In: *Cloud Computing and Services Science*. Ed. by Markus Helfert, Frédéric Desprez, Donald Ferguson, and Frank Leymann. Springer, 2014, pp. 52–68.

- [106] Ulrich Lampe, Ronny Hans, Marco Seliger, and Michael Pauly. "Pricing in Infrastructure Clouds – An Analytical and Empirical Examination." In: *Proceedings of the 20th Americas Conference on Information Systems (AMCIS)* (2014), pp. 1–10.
- [107] Federico Larumbe and Brunilde Sansò. "Cloptimus: A Multi-objective Cloud Data Center and Software Component Location Framework." In: *Proceedings of the IEEE 1st International Conference on Cloud Networking (CloudNet)*. 2012.
- [108] Federico Larumbe and Brunilde Sansò. "Optimal Location of Data Centers and Software Components in Cloud Computing Network Design." In: *Proceeding of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*). 2012.
- [109] Federico Larumbe and Brunilde Sansò. "A Tabu Search Algorithm for the Location of Data Centers and Software Components in Green Cloud Computing Networks." In: *IEEE Transactions on Cloud Computing* 1.1 (2013), pp. 22–35.
- [110] Sonja Lehmann and Peter Buxmann. "Pricing Strategies of Software Vendors." In: *Business & Information Systems Engineering* 1.6 (2009), pp. 452–462.
- [111] Sonja Lehmann, Tobias Draisbach, Peter Buxmann, and Corina Koll. *Pricing Models of Software as a Service Providers: Usage-Dependent Versus Usage-Independent Pricing Models*. Tech. rep. Technische Universität Darmstadt, 2010.
- [112] Sonja Lehmann, Tobias Draisbach, Peter Buxmann, and Petra Dörsam. "Pricing of Software as a Service- An Empirical Study in View of the Economics of Information Theory." In: *Software Business*. Springer, 2012.
- [113] Stefanie Leimeister, Markus Böhm, Christoph Riedl, and Helmut Krcmar. "The Business Perspective of Cloud Computing: Actors, Roles and Value Networks." In: *Proceedings of the 18th European Conference on Information Systems (ECIS)*. 2010.
- [114] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. "What's Inside the Cloud? An Architectural Map of the Cloud Landscape." In: *Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing*. 2009, pp. 23–31.
- [115] Frank Leymann, Christopg Fehling, Ralph Mietzner, Alexander Nowak, and Schahram Dustdar. "Moving Applications to the Cloud: An Approach Based on Application Model Enrichment." In: *International Journal of Cooperative Information Systems* 20.3 (2011), pp. 307–356.

- [116] Minghong Lin, Zhenhua Liu, Adam Wierman, and Lachlan L.H. Andrew. "Online Algorithms for Geographical Load Balancing." In: *Proceedings of the International Green Computing Conference (IGCC)*. 2012, pp. 1–10.
- [117] Minghong Lin, Adam Wierman, Lachlan L.H. Andrew, and Eno Thereska. "Dynamic Right-sizing for Power-proportional Data Centers." In: *IEEE/ACM Transactions on Networking (TON)* 21.5 (2013), pp. 1378–1391.
- [118] Yutu Liu, Anne H. Ngu, and Liang Z. Zeng. "QoS Computation and Policing in Dynamic Web Service Selection." In: *Proceedings of the 13th International World Wide Web Conference (WWW)*. 2004, pp. 66–73.
- [119] Ying Lu, Tarek Abdelzaher, Chenyang Lu, and Gang Tao. "An Adaptive Control Framework for QoS Guarantees and its Application to Differentiated Caching." In: *Proceedings of the 10th IEEE International Workshop on Quality of Service*. 2002, pp. 23–32.
- [120] Jose Luis Lucas-Simarro, Rafael Moreno-Vozmediano, Ruben S. Montero, and Ignacio M. Llorente. "Scheduling strategies for optimal service deployment across multiple clouds." In: *Future Generation Computer Systems* 29.6 (2013), pp. 1431–1441.
- [121] Heiko Ludwig, Alexander Keller, Asit Dan, Richard King, and Richard Franck. "A Service Level Agreement Language for Dynamic Electronic Services." In: *Electronic Commerce Research* 3.1 (2003), pp. 43–59.
- [122] Zaigham Mahmood and Richard Hill. *Cloud Computing for Enterprise Architectures*. Springer, 2011.
- [123] Michele Mazzucco and Dmytro Dyachuk. "Optimizing Cloud Providers Revenues via Energy Efficient Server Allocation." In: *Sustainable Computing: Informatics and Systems* 2.1 (2012), pp. 1–12.
- [124] John McCourt. *Starbucks and the City*. Online. 2013. URL: <https://nycstarbucks.com/the-list/>.
- [125] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing*. Tech. rep. letzter Zugriff am 03.02.2015. National Institute of Standards and Technology, 2011. URL: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>.
- [126] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement." In: *Proceedings of the 29th IEEE Conference on Computer Communications (INFO-COM)*. 2010, pp. 1–9.
- [127] Lawrence S. Meyers, Glenn Gamst, and Anthony J. Guarino. *Applied Multivariate Research: Design and Interpretation*. Sage Publications, 2006.

- [128] Rich Miller. *Inside Amazon's Cloud Computing Infrastructure*. Online. 2015. URL: <http://datacenterfrontier.com/inside-amazon-cloud-computing-infrastructure/>.
- [129] Hossein Morshedlou and Mohammad Reza Meybodi. "Decreasing Impact of SLA Violations: A Proactive Resource Allocation Approach for Cloud Computing Environments." In: *IEEE Transactions on Cloud Computing* 2.2 (2014), pp. 156–167.
- [130] Theodore Samuel Motzkin and Isaac Jacob Schoenberg. "The Relaxation Method for Linear Inequalities." In: *Canadian Journal of Mathematics* 6 (1954), pp. 393–404.
- [131] Gerald Münzl, Michael Pauly, and Martin Reti. *Cloud Computing als neue Herausforderung für Management und IT*. Springer Vieweg, 2015.
- [132] John Naughton. *Brief History of the Future: Origins of the Internet*. Phoenix, 2000.
- [133] Mahyar Movahed Nejad, Lena Mashayekhy, and Daniel Grosu. "Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds." In: *IEEE Transactions on Parallel and Distributed Systems* 26.2 (2015), pp. 594–603.
- [134] Nuttapong Netjinda, Booncharoen Sirinaovakul, and Tiranee Achalakul. "Cost Optimal Scheduling in IaaS for Dependent Workload with Particle Swarm Optimization." In: *The Journal of Supercomputing* 68.3 (2014), pp. 1579–1603.
- [135] New York City Economic Development Corporation (NYCEDC). *NYC Broadband Map*. Online. 2017. URL: <https://www.nycbbmap.com/>.
- [136] Lothar Pantel and Lars C. Wolf. "On the Impact of Delay on Real-Time Multiplayer Games." In: *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. 2002, pp. 23–29.
- [137] Chintan Patel, Kaustubh Supekar, and Yugyung Lee. "A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows." In: *Proceedings of the 14th International Conference on Database and Expert Systems Applications (DEXA)*. 2003, pp. 826–835.
- [138] Komal Singh Patel and A. K. Sarje. "VM Provisioning Method to Improve the Profit and SLA Violation of Cloud Service Providers." In: *Proceeding of the IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE. 2012, pp. 1–5.
- [139] Pankesh Patel, Ajith H. Ranabahu, and Amit P. Sheth. *Service Level Agreement in Cloud Computing*. Tech. rep. Kno.e.sis Publications, 2009.

- [140] James Pearn. *How any servers does Google have?* Online. 2012. URL: <https://plus.google.com/+JamesPearn/posts/VaQu9sNxJuY0>.
- [141] Jing Tai Piao and Jun Yan. "A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing." In: *Proceeding of the 9th International Conference on Grid and Cloud Computing*. 2010, pp. 87–92.
- [142] Axel Pols and Peter Heidkamp. *Cloud-Monitor 2016*. Online. 2016. URL: <https://www.bitkom.org/Presse/Anhaenge-an-PIs/2016/Mai/Bitkom-KPMG-Charts-PK-Cloud-Monitor-12-05-2016.pdf>.
- [143] Dang Minh Quan, Robert Basmadjian, Hermann De Meer, Ricardo Lent, Toktam Mahmoodi, Domenico Sannelli, Federico Mezza, Luigi Telesca, and Corenten Dupont. "Energy Efficient Resource Allocation Strategy for Cloud Data Centres." In: *Computer and Information Sciences II*. Springer, 2011, pp. 133–141.
- [144] Shuping Ran. "A Model for Web Services Discovery with QoS." In: *ACM SIGecom Exchanges* 4.1 (2003), pp. 1–10.
- [145] Ashwin Rao, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi Barakat, and Walid Dabbous. "Network Characteristics of Video Streaming Traffic." In: *Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. 2011, pp. 1–12.
- [146] Patrick Raycroft, Ryan Jansen, Mateusz Jarus, and Paul R. Brenner. "Performance Bounded Energy Efficient Virtual Machine Allocation in the Global Cloud." In: *Sustainable Computing: Informatics and Systems* 4.1 (2014), pp. 1–9.
- [147] Colin R. Reeves. "Modern Heuristic Search Methods." In: *Modern Heuristic Techniques*. Ed. by Vic J. Rayward-Smith, Ibrahim H. Osman, and Colin R. Reeves. Wiley, 1996.
- [148] Regierungskommission Deutscher Corporate Governance Kodex. *Deutscher Corporate Governance Kodex*. Online. 2017. URL: http://www.dcgk.de//files/dcgk/usercontent/de/download/kodex/170424_Kodex_finale_Version_D.pdf.
- [149] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. "A Taxonomy and Survey of Cloud Computing Systems." In: *Proceedings of the 5th International Joint Conference on INC, IMS and IDC* (2009), pp. 44–51.
- [150] Suzanne Robertson and James Robertson. *Mastering the Requirements Process: Getting Requirements Right*. 3rd. Addison-Wesley, 2014.
- [151] Parnia Samimi and Ahmed Patel. "Review of Pricing Models for Grid & Cloud Computing." In: *Proceedings of the IEEE Symposium on Computers & Informatics (ISCI)*. 2011, pp. 634–639.

- [152] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. "The Case for VM-based Cloudlets in Mobile Computing." In: *Pervasive Computing* 8.4 (2009), pp. 14–23.
- [153] Mahadev Satyanarayanan, Rolf Schuster, Maria Ebling, Gerhard Fettweis, Hannu Flinck, Kaustubh Joshi, and Krishan Sabnani. "An Open Ecosystem for Mobile-Cloud Convergence." In: *Communications Magazine, IEEE* 53.3 (2015), pp. 63–70.
- [154] Henry E. Schaffer. "X as a Service, Cloud Computing, and the Need for Good Judgment." In: *IT Professional* 11.5 (2009), pp. 4–5. doi: 10.1109/MITP.2009.112.
- [155] Lutz Schubert, Keith Jeffery, and Burkhard Neidecker-Lutz. *The Future of Cloud Computing - Opportunities For European Cloud Computing Beyond 2010*. Online. 2011. URL: <http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf>.
- [156] Dieter Schuller. "QoS-aware Service Selection: Optimization Mechanisms and Decision Support for Complex Service-based Workflows." PhD thesis. Technischen Universität Darmstadt, 2013.
- [157] Maurice Shahd. *Sicherheit für IT-Unternehmen das Thema des Jahres*. Online. 2016. URL: <https://www.bitkom.org/Presse/Presseinformation/Sicherheit-fuer-IT-Unternehmen-das-Thema-des-Jahres.html>.
- [158] Huajie Shao, Lei Rao, Zhi Wang, Xue Liu, Zhibo Wang, and Kui Ren. "Optimal Load Balancing and Energy Cost Management for Internet Data Centers in Deregulated Electricity Markets." In: *IEEE Transactions on Parallel and Distributed Systems* 25.10 (2014), pp. 2659–2669.
- [159] Mohamed Abu Sharkh, Abdelkader Ouda, and Abdallah Shami. "A Resource Scheduling Model for Cloud Computing Data Centers." In: *Proceedings of the IEEE 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2013, pp. 213–218.
- [160] Jane Siegel and Jeff Perdue. "Cloud Services Measures for Global Use: The Service Measurement Index (SMI)." In: *Proceedings of the 1st Annual SRII Global Conference*. 2012, pp. 411–415.
- [161] Edward Allen Silver. "An Overview of Heuristic Solution Methods." In: *Journal of the Operational Research Society* 55.9 (2004), pp. 936–956.
- [162] Joel Sommers and Paul Barford. "Cell vs. WiFi: On the Performance of Metro Area Mobile Connections." In: *Proceedings of the ACM Conference on Internet Measurement*. 2012, pp. 301–314.

- [163] Statista. *Number of Smartphone Users in Germany 2013-2021*. Online. 2017. URL: <https://www.statista.com/statistics/467170/forecast-of-smartphone-users-in-germany/>.
- [164] Ralf Steinmetz. *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer, 2000.
- [165] David Strom and Jelle Frank van der Zwet. *Truth and Lies About Latency in the Cloud*. Tech. rep. Interxion, 2015. URL: http://www.interxion.com/globalassets/_documents/whitepapers-and-pdfs/cloud/WP_TRUTHANDLIES_en_0715.pdf.
- [166] Leena Suhl and Taieb Mellouli. *Optimierungssysteme: Modelle, Verfahren, Software, Anwendungen*. 1st ed. Springer Verlag, 2006.
- [167] Leena Suhl and Taieb Mellouli. *Optimierungssysteme: Modelle, Verfahren, Software, Anwendungen*. Springer Verlag, 2013.
- [168] Minghe Sun. "A Tabu Search Heuristic Procedure for the Capacitated Facility Location Problem." In: *Journal of Heuristics* 18.1 (2012), pp. 91–118.
- [169] Arun Sundararajan. "Nonlinear Pricing of Information Goods." In: *Management Science* 50.12 (2004), pp. 1660–1673.
- [170] Andrew S. Tanenbaum. *Computer Networks*. 4th ed. Prentice Hall, 2003.
- [171] Telecommunication Standardization Selector of ITU (ITU-T). *Definitions of Terms Related to Quality of Service (E.800)*. Tech. rep. The International Telecommunication Union (ITU), 2008.
- [172] Jean-Paul Thommen and Ann-Kristin Achleitner. *Allgemeine Betriebswirtschaftslehre*. Gabler, 2001.
- [173] Niraj Tolia, David G Andersen, and Mahadev Satyanarayanan. "Quantifying Interactive User Experience on Thin Clients." In: *Computer* 39.3 (2006), pp. 46–52.
- [174] Johan Tordsson, Rubén S Montero, Rafael Moreno-Vozmediano, and Ignacio M Llorente. "Cloud Brokering Mechanisms for Optimized Placement of Virtual Machines Across Multiple Providers." In: *Future Generation Computer Systems* 28.2 (2012), pp. 358–367.
- [175] W. Pitt Turner and Kenneth G. Brill. *Cost Model: Dollars per kW plus Dollars per Square foot of Computer Floor*. Tech. rep. The Uptime Institute, 2008.
- [176] W. Pitt Turner and John H. Seader. *Dollars per kW plus Dollars per Square Foot are a Better Datacenter Cost Model than Dollars per Square Foot Alone*. Tech. rep. The Uptime Institute, 2006.
- [177] W. Pitt Turner, John H. Seader, and Kenneth G. Brill. *Tier Classifications Define Site Infrastructure Performance*. Tech. rep. The Uptime Institute, 2006.

- [178] Tobias Unger, Frank Leymann, Stephanie Mauchart, and Thorsten Scheibler. "Aggregation of Service Level Agreements in the Context of Business Processes." In: *Proceeding of the IEEE 12th International Enterprise Distributed Object Computing Conference (EDOC)*. 2008, pp. 43–52.
- [179] United States Census Bureau. *Cities and Towns Population Totals Tables: 2010–2015*. Online. 2015. URL: <https://www.census.gov/data/tables/2015/demo/popest/total-cities-and-towns.html>.
- [180] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. "A Break in the Clouds: Towards a Cloud Definition." In: *ACM SIGCOMM Computer Communication Review* 39.1 (2008), pp. 50–55.
- [181] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. "Cloudlets: Bringing the Cloud to the Mobile User." In: *Proceedings of the 3rd ACM Workshop on Mobile Cloud Computing and Services*. 2012.
- [182] Wei Wang, Di Niu, Baochun Li, and Ben Liang. "Dynamic Cloud Resource Reservation via Cloud Brokerage." In: *Proceedings of the IEEE 33rd International Conference on Distributed Computing Systems (ICDCS)*. 2013, pp. 400–409.
- [183] Yefu Wang and Xiaorui Wang. "Performance-controlled Server Consolidation for Virtualized Data Centers with Multi-tier Applications." In: *Sustainable Computing: Informatics and Systems* 4.1 (2014), pp. 52–65.
- [184] Christof Weinhardt, Arun Anandasivam, Benjamin Blau, Nikolay Borissov, Thomas Meinl, Wibke Michalk, and Jochen Stößer. "Cloud Computing – A Classification, Business Models, and Research Directions." In: *Business & Information Systems Engineering* 1.5 (2009), pp. 391–399.
- [185] Joe Weinman. *The 10 Laws of Cloudonomics*. Online. 2008. URL: <https://gigaom.com/2008/09/07/the-10-laws-of-cloudonomics/>.
- [186] Olga Wenge. "Optimization in Inter-Cloud Markets - Solution Strategies for the Formation of Profitable Cloud Collaborations in Relation to Quality of Service, Information Security, and Regulatory Aspects." PhD thesis. Technische Universität Darmstadt, 2017.
- [187] Marino Widmer and Alain Hertz. "A New Heuristic Method for the Flow Shop Sequencing Problem." In: *European Journal of Operational Research* 41.2 (1989), pp. 186–193.
- [188] H Paul Williams. *Model Building in Mathematical Programming*. 5th. Wiley, 1999.
- [189] Caesar Wu and Rajkumar Buyya. *Cloud Data Centers and Cost Modeling*. Morgan Kaufmann, 2015.

- [190] Zhen Xiao, Weijia Song, and Qi Chen. "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment." In: *IEEE Transactions on Parallel and Distributed Systems* 24.6 (2013), pp. 1107–1117.
- [191] Yagiz Onat Yazir, Chris Matthews, Roozbeh Farahbod, Stephen Neville, Adel Guitouni, Sudhakar Ganti, and Yvonne Coady. "Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis." In: *na* (2010).
- [192] Lamia Youseff, Maria Butrico, and Dilma Da Silva. "Toward a Unified Ontology of Cloud Computing." In: *Proceeding of the IEEE Grid Computing Environments Workshop (GCE)*. IEEE. 2008, pp. 1–10.
- [193] Sharrukh Zaman and Daniel Grosu. "A Combinatorial Auction-based Mechanism for Dynamic VM Provisioning and Allocation in Clouds." In: *IEEE Transactions on Cloud Computing* 1.2 (2013), pp. 129–141.
- [194] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Znf Sheng. "Quality Driven Web Services Composition." In: *Proceedings of the 12th International Conference on World Wide Web (WWW)*. 2003, pp. 411–421.
- [195] Jiangtao Zhang, Hejiao Huang, and Xuan Wang. "Resource Provision Algorithms in Cloud Computing: A Survey." In: *Journal of Network and Computer Applications* 64 (2016), pp. 23–42.
- [196] Li Zhang and Danilo Ardagna. "SLA Based Profit Optimization in Autonomic Computing Systems." In: *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC)*. 2004, pp. 173–182.
- [197] Qi Zhang, Lu Cheng, and Raouf Boutaba. "Cloud Computing: State-of-the-Art and Research Challenges." In: *Journal of Internet Services and Applications* 1.1 (2010), pp. 7–18.
- [198] Qi Zhang, Quanyan Zhu, Mohamed Faten Zhani, Raouf Boutaba, and Joseph L. Hellerstein. "Dynamic Service Placement in Geographically Distributed Clouds." In: *IEEE Journal on Selected Areas in Communications* 31.12 (2013), pp. 762–772.
- [199] Jieming Zhu, Zibin Zheng, Yangfan Zhou, and Michael R Lyu. "Scaling Service-oriented Applications into Geo-distributed Clouds." In: *Proceedings of the IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*. 2013, pp. 335–340.

All web pages cited in this work have been checked in June 2017. However, due to the dynamic nature of the World Wide Web, the long-term availability of web pages cannot be guaranteed.

ACRONYMS

CDCSP Cloud Data Center Selection Problem

CaaS Communication as a Service

DaaS Desktop as a Service

DBaaS Databases as a Service

DCPSP Dynamic Cloudlet Placement and Selection Problem

HaaS Hardware as a Service

IP Integer Programming

IaaS Infrastructure as a Service

IT Information Technology

ITU International Telecommunication Union

LAN Local Area Network

LP Linear Program

MAN Metropolitan Area Network

MIP Mixed Integer Programming

NIST National Institute of Standards and Technology

P2P Peer-to-Peer

PaaS Platform as a Service

PC Personal Computer

QoS Quality of Service

SaaS Software as a Service

SLA Service Level Agreement

SLO Service Level Objective

WAN Wide Area Network

WLAN Wireless Local Area Network

XaaS Everything as a Service

A

APPENDIX

In this appendix, we provide the complete optimization models presented in Chapters 4 and 5.

A.1 OPTIMIZATION MODEL FOR THE CLOUD DATA CENTER SELECTION PROBLEM

This optimization approach addresses the challenges of cost minimal and QoS-aware data center selection for multiple services on a global scale.

Model 5 Cloud Data Center Selection Problem (CDCSP)

$$\text{Min. } C(x, y) = \sum_{\lambda=1..A} x_{d_\lambda} \times C_{d_\lambda}^{\text{fix}} + \sum_{\substack{\lambda=1..A \\ \mu=1..M \\ \nu=1..N}} y_{d_\lambda, u_\mu, s_\nu} \times C_{d_\lambda}^{\text{var}} \quad (\text{A.1})$$

$$\sum_{\lambda=1..A} y_{d_\lambda, u_\mu, s_\nu} \geq V_{u_\mu, s_\nu} \quad \forall u_\mu \in U, \forall s_\nu \in S \quad (\text{A.2})$$

$$\sum_{\substack{\mu=1..M \\ \nu=1..N}} y_{d_\lambda, u_\mu, s_\nu} \geq x_{d_\lambda} \times K_{d_\lambda}^{\text{min}} \quad \forall d_\lambda \in D \quad (\text{A.3})$$

$$\sum_{\substack{\mu=1..M \\ \nu=1..N}} y_{d_\lambda, u_\mu, s_\nu} \leq x_{d_\lambda} \times K_{d_\lambda}^{\text{max}} \quad \forall d_\lambda \in D \quad (\text{A.4})$$

$$y_{d_\lambda, u_\mu, s_\nu} \leq p_{d_\lambda, u_\mu, s_\nu} \times K_{d_\lambda}^{\text{max}} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S \quad (\text{A.5})$$

$$p_{d_\lambda, u_\mu, s_\nu} = \begin{cases} 1 & \text{if } Q_{d_\lambda, u_\mu, q_\xi}^{\text{gua}} \geq Q_{u_\mu, s_\nu, q_\xi}^{\text{req}} \quad \forall q_\xi \in Q \\ 0 & \text{else} \end{cases} \quad (\text{A.6})$$

.....

$$x_{d_\lambda} \in \{0, 1\} \quad \forall d_\lambda \in D \quad (\text{A.7})$$

$$y_{d_\lambda, u_\mu, s_\nu} \in \mathbb{N} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S \quad (\text{A.8})$$

A.2 OPTIMIZATION MODEL FOR THE DYNAMIC CLOUDLET PLACEMENT AND SELECTION PROBLEM

This dynamic optimization approach focuses on the selection and capacity planning of local resources, i. e., cloudlets. It considers a time-depended service demand for services with different QoS characteristics. Apart from the capacity constraints for computation units, it also considers different types of networks and the bandwidth restrictions. Due to the complexity of the model, we divide it into several parts. The Models 6 and 7 present the basic form including a non-linear expression for the migration cost.

Model 6 Dynamic Cloudlet Placement and Selection Problem (Part 1)

$$\begin{aligned}
\text{Min. } C(x, y, z) = & \sum_{\lambda=1..N} x_{d_\lambda} \times C_{d_\lambda}^{\text{fix}} \\
& + \sum_{o=1..O} \left(\sum_{\substack{\lambda=1..N \\ \mu=1..M \\ v=1..N}} y_{d_\lambda, u_\mu, s_v, t_o} \times C_{d_\lambda, t_o}^{\text{op}} + \sum_{\substack{\mu=1..M \\ v=1..N}} y_{u_\mu, s_v, t_o}^{\text{pen}} \times C_{u_\mu, s_v}^{\text{pen}} \right) \\
& + \sum_{o=2}^O \sum_{\substack{\lambda=1..N \\ \mu=1..M \\ v=1..N}} y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}} \times C_{s_v}^{\text{mig}} + \sum_{\lambda=1..N} z_{d_\lambda} \times C_{d_\lambda}^{\text{hw}}
\end{aligned} \tag{A.9}$$

$$y_{d_\lambda, u_\mu, s_v, t_o}^{\text{mig}} = \begin{cases} y_{d_\lambda, u_\mu, s_v, t_{o-1}} - y_{d_\lambda, u_\mu, s_v, t_o} & \text{if} \\ \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_v, t_o} \geq \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_v, t_{o-1}} \\ \wedge y_{d_\lambda, u_\mu, s_v, t_o} \leq y_{d_\lambda, u_\mu, s_v, t_{o-1}} \\ y_{d_\lambda, u_\mu, s_v, t_o} - y_{d_\lambda, u_\mu, s_v, t_{o-1}} & \text{if} \\ \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_v, t_o} < \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_v, t_{o-1}} \\ \wedge y_{d_\lambda, u_\mu, s_v, t_o} > y_{d_\lambda, u_\mu, s_v, t_{o-1}} \\ 0 & \text{else} \end{cases} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T \tag{A.10}$$

$$y_{u_\mu, s_v, t_o}^{\text{pen}} + \sum_{\lambda=1..N} y_{d_\lambda, u_\mu, s_v, t_o} \geq V_{u_\mu, s_v, t_o} \quad \forall u_\mu \in U, \forall s_v \in S, \forall t_o \in T \tag{A.11}$$

$$\sum_{\substack{m=1..n \\ v=1..N}} y_{d_\lambda, u_\mu, s_v, t_o} \leq z_{d_\lambda} \quad \forall d_\lambda \in D, \forall t_o \in T \tag{A.12}$$

$$\begin{aligned}
z_{d_\lambda} &\leq x_{d_\lambda} \times K_{d_\lambda}^{\max} \quad \forall d_\lambda \in D \\
z_{d_\lambda} &\geq x_{d_\lambda} \times K_{d_\lambda}^{\min} \quad \forall d_\lambda \in D
\end{aligned} \tag{A.13}$$

Model 7 Dynamic Cloudlet Placement and Selection (Part 2)

$$y_{d_\lambda, u_\mu, s_\nu, t_o} \leq p_{d_\lambda, u_\mu, s_\nu} \times K_{d_\lambda}^{max} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (A.14)$$

$$p_{d_\lambda, u_\mu, s_\nu} = \begin{cases} 1 & \text{if } Q_{d_\lambda, u_\mu, q_\xi}^{gua} \geq Q_{u_\mu, s_\nu, q_\xi}^{req} \forall q_\xi \in Q \\ 0 & \text{else} \end{cases} \quad (A.15)$$

$$\sum_{\lambda=1..N} \sum_{\nu=1..N} y_{d_\lambda, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{down} \leq K_{u_\mu}^{LAN_{down}} \quad \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (A.16)$$

$$\sum_{\lambda=1..N} \sum_{\nu=1..N} y_{d_\lambda, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{up} \leq K_{u_\mu}^{LAN_{up}} \quad \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (A.17)$$

$$\begin{aligned} & \sum_{\substack{\lambda=1..N \\ \lambda \neq \alpha}} \sum_{\nu=1..N} y_{d_\lambda, u_\alpha, s_\nu, t_o} \times L_{s_\nu}^{down} + \sum_{\substack{\mu=1..M \\ \mu \neq \alpha}} \sum_{\nu=1..N} y_{d_\alpha, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{up} \\ & \leq K_{u_\alpha}^{MAN_{down}} \quad \forall d_\alpha \in D, \forall u_\alpha \in U, \forall s_\nu \in S, \forall t_o \in T \end{aligned} \quad (A.18)$$

$$\begin{aligned} & \sum_{\substack{\lambda=1..N \\ \lambda \neq \alpha}} \sum_{\nu=1..N} y_{d_\lambda, u_\alpha, s_\nu, t_o} \times L_{s_\nu}^{up} + \sum_{\substack{\mu=1..M \\ \mu \neq \alpha}} \sum_{\nu=1..N} y_{d_\alpha, u_\mu, s_\nu, t_o} \times L_{s_\nu}^{down} \\ & \leq K_{u_\alpha}^{MAN_{up}} \quad \forall d_\alpha \in D, \forall u_\alpha \in U, \forall s_\nu \in S, \forall t_o \in T \end{aligned} \quad (A.19)$$

.....

$$\begin{aligned} & x_{d_\lambda} \in \{0, 1\} \quad \forall d_\lambda \in D \\ & y_{d_\lambda, u_\mu, s_\nu, t_o} \in \mathbb{R}^+ \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \\ & y_{d_\lambda, u_\mu, s_\nu, t_o}^{mig} \in \mathbb{R}^+ \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \\ & y_{u_\mu, s_\nu, t_o}^{pen} \in \mathbb{R}^+ \quad \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \\ & z_{d_\lambda} \in \mathbb{N} \quad \forall d_\lambda \in D \end{aligned} \quad (A.20)$$

Model 8 Linearization Migration Cost (Part 1)

$$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}} = \begin{cases} y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} - y_{d_\lambda, u_\mu, s_\nu, t_o} & \text{if} \\ \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_\nu, t_o} \geq \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_\nu, t_{o-1}} \\ \wedge y_{d_\lambda, u_\mu, s_\nu, t_o} \leq y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} \\ y_{d_\lambda, u_\mu, s_\nu, t_o} - y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} & \text{if} \\ \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_\nu, t_o} < \sum_{\alpha=1..N} y_{d_\alpha, u_\mu, s_\nu, t_{o-1}} \\ \wedge y_{d_\lambda, u_\mu, s_\nu, t_o} > y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} \\ 0 & \text{else} \end{cases} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (\text{A.21})$$

.....

$$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}} = y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_1} + y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_2} \quad (\text{A.22})$$

$$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_1} = \begin{cases} y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} - y_{d_\lambda, u_\mu, s_\nu, t_o} & \text{if } h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} = 1 \\ 0 & \text{else} \end{cases} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (\text{A.23})$$

$$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_2} = \begin{cases} y_{d_\lambda, u_\mu, s_\nu, t_o} - y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} & \text{if } h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} = 1 \\ 0 & \text{else} \end{cases} \quad \forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T \quad (\text{A.24})$$

$$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_1} \leq K_{d_\lambda}^{\max} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} \quad (\text{A.25})$$

$$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_2} \leq K_{d_\lambda}^{\max} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} \quad (\text{A.26})$$

$$(y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} - y_{d_\lambda, u_\mu, s_\nu, t_o}) - y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_1} \leq K_{d_\lambda}^{\max} \times (1 - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1}) \quad (\text{A.27})$$

$$(y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} - y_{d_\lambda, u_\mu, s_\nu, t_o}) - y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_1} \leq K_{d_\lambda}^{\max} \times (1 - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1}) \quad (\text{A.28})$$

Model 9 Linearization Migration Cost (Part 2)

$$y_{d_\lambda, u_\mu, s_\nu, t_o}^{mig, c_2} - (y_{d_\lambda, u_\mu, s_\nu, t_o} - y_{d_\lambda, u_\mu, s_\nu, t_{o-1}}) \leq K_{d_\lambda}^{max} \times (1 - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2}) \quad (A.29)$$

$$(y_{d_\lambda, u_\mu, s_\nu, t_o} - y_{d_\lambda, u_\mu, s_\nu, t_{o-1}}) - y_{d_\lambda, u_\mu, s_\nu, t_o}^{mig, c_2} \leq K_{d_\lambda}^{max} \times (1 - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2}) \quad (A.30)$$

$$\sum_{\alpha=1..\Lambda} y_{d_\alpha, u_\mu, s_\nu, t_o} - \sum_{\alpha=1..\Lambda} y_{d_\alpha, u_\mu, s_\nu, t_{o-1}} \leq \sum_{\alpha=1..\Lambda} K_{d_\alpha}^{max} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} \quad (A.31)$$

$$\sum_{\alpha=1..\Lambda} y_{d_\alpha, u_\mu, s_\nu, t_{o-1}} - \sum_{\alpha=1..\Lambda} y_{d_\alpha, u_\mu, s_\nu, t_o} < \sum_{\alpha=1..\Lambda} K_{d_\alpha}^{max} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} \quad (A.32)$$

$$y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} - y_{d_\lambda, u_\mu, s_\nu, t_o} \leq K_{d_\lambda}^{max} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} \quad (A.33)$$

$$y_{d_\lambda, u_\mu, s_\nu, t_o} - y_{d_\lambda, u_\mu, s_\nu, t_{o-1}} < K_{d_\lambda}^{max} \times h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} \quad (A.34)$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} = 1 \quad (A.35)$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} = 1 \quad (A.36)$$

Model 10 Linearization Migration Cost (Part 3)

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} \leq 0 \quad (\text{A.37})$$

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} \leq 0 \quad (\text{A.38})$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2} - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1} \leq 1 \quad (\text{A.39})$$

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} \leq 0 \quad (\text{A.40})$$

$$-h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} \leq 0 \quad (\text{A.41})$$

$$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1} + h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2} - h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2} \leq 1 \quad (\text{A.42})$$

.....

$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}}$	$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_1}$	$y_{d_\lambda, u_\mu, s_\nu, t_o}^{\text{mig}, c_2}$	$\forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T$
$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1}$	$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2}$	$\in \{0, 1\}$	$\forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T$
$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_1}$	$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_1, p_2}$	$\in \{0, 1\}$	$\forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T$
$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_1}$	$h_{d_\lambda, u_\mu, s_\nu, t_o}^{c_2, p_2}$	$\in \{0, 1\}$	$\forall d_\lambda \in D, \forall u_\mu \in U, \forall s_\nu \in S, \forall t_o \in T$

(A.43)

AUTHOR'S PUBLICATIONS

MAIN PUBLICATIONS

- [1] Ronny Hans. "Selecting Cloud Data Centers for QoS-Aware Multimedia Applications." In: *Proceedings of the PhD Symposium at the 2nd European Conference on Service-Oriented and Cloud Computing (ESOCC)*. 2013, pp. 11–16.
- [2] Ronny Hans, Ulrich Lampe, and Ralf Steinmetz. "QoS-Aware, Cost-Efficient Selection of Cloud Data Centers." In: *Proceedings of the 6th IEEE International Conference on Cloud Computing (CLOUD)*. 2013, pp. 946–947.
- [3] Ronny Hans, Sebastian Zöller, Sven Abels, and Ralf Steinmetz André Miede. "Enabling Collaboration in Virtual Manufacturing Enterprises with Cloud Computing." In: *Proceedings of the 19th Americas Conference on Information Systems (AMCIS)*. 2013, pp. 1–10.
- [4] Ronny Hans, David Dahlen, Sebastian Zöller, Dieter Schuller, and Ulrich Lampe. "Enabling Virtual Manufacturing Enterprises with Cloud Computing – An Analysis of Criteria for the Selection of Database as a Service Offers." In: *Advances in Sustainable and Competitive Manufacturing Systems*. Ed. by Américo Azevedo. Springer, 2013, pp. 427–438.
- [5] Ronny Hans, Manuel Zahn, Ulrich Lampe, Apostolos Papageorgiou, and Ralf Steinmetz. "Energy-efficient Web Service Invocation on Mobile Devices: The Influence of Compression and Parsing." In: *Proceedings of the 2nd International Conference on Mobile Services (MS)*. 2013, pp. 1–6.
- [6] Ronny Hans, Ulrich Lampe, Michael Pauly, and Ralf Steinmetz. "Cost-efficient Capacitation of Cloud Data Centers for QoS-aware Multimedia Service Provision." In: *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER)*. 2014, pp. 158–163.
- [7] Ronny Hans, Ulrich Lampe, Daniel Burgstahler, Martin Hellwig, and Ralf Steinmetz. "Where Did My Battery Go? Quantifying the Energy Consumption of Cloud Gaming." In: *Proceedings of the 3rd IEEE International Conference on Mobile Services (MS)*. 2014, pp. 63–67.
- [8] Ronny Hans, Daniel Burgstahler, Alexander Mueller, Manuel Zahn, and Dominik Stingl. "Knowledge for a Longer Life: Development Impetus for Energy-efficient Smartphone Applications." In: *Proceedings of the 4th International Conference on Mobile Services (MS)*. 2015, pp. 1–8.

- [9] Ronny Hans, David Steffen, Ulrich Lampe, Björn Richerzhagen, and Ralf Steinmetz. "Setting Priorities: A Heuristic Approach for Cloud Data Center Selection." In: *Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER)*. 2015, pp. 221–228.
- [10] Ronny Hans, Hendrik Zuber, Amr Rizk, and Ralf Steinmetz. "Blockchain and Smart Contracts: Disruptive Technologies for the Insurance Market." In: *Proceedings of the 23th Americas Conference on Information Systems (AMCIS)*. 2017, pp. 1–10.
- [11] Ronny Hans, Melanie Holloway, The An Binh Nguyen, Alexander Müller, Marco Seliger, and Jürgen Lang. *Cloud-Applikationen in der Finanzindustrie - Dienstgütemerkmale und deren Überwachung*. Tech. rep. Technische Universität Darmstadt, 2017.
- [12] Ronny Hans, Björn Richerzhagen, Amr Rizk, Ulrich Lampe, Ralf Steinmetz, Sabrina Klos, and Anja Klein. "Little Boxes: A Dynamic Optimization Approach for Enhanced Cloud Infrastructures." In: *Proceedings of the 7th European Conference on Service-Oriented and Cloud Computing (ESOCC)*. 2018.
- [13] Ronny Hans, Björn Richerzhagen, Amr Rizk, Ulrich Lampe, Ralf Steinmetz, Sabrina Klos, and Anja Klein. *Little Boxes: A Dynamic Optimization Approach for Enhanced Cloud Infrastructures*. Tech. rep. 1807.02615. Technische Universität Darmstadt, 2018. URL: <https://arxiv.org/abs/1807.02615>.

CO-AUTHORED PUBLICATIONS

- [14] Daniel Burgstahler, Nils Richerzhagen, Frank Englert, Ronny Hans, and Ralf Steinmetz. "Switching Push - Pull: An Energy Efficient Notification Approach." In: *Proceedings of the 3rd International Conference on Mobile Services (MS)*. 2014, pp. 68–75.
- [15] Rahul Dwarakanath, Jérôme Charrier, Frank Englert, Ronny Hans, Dominik Stingl, and Ralf Steinmetz. "Analyzing the Influence of Instant Messaging on User Relationship Estimation." In: *Proceedings of the IEEE 5th International Conference on Mobile Services (MS)*. 2016, pp. 49–56.
- [16] Melanie Holloway, Marvin Dickhaus, Ronny Hans, Bastian Emondts, Amr Rizk, and Ralf Steinmetz. "Cloud Adoption in the Spotlight – Empirical Insights from German IT Experts." In: *Proceedings of the 23th Americas Conference on Information Systems (AMCIS)*. 2017, pp. 1–10.
- [17] Ulrich Lampe, Ronny Hans, and Ralf Steinmetz. "Data Center Selection for QoS-Aware Service Provision." In: *efl quarterly* 2013.4 (2013), pp. 4–5.

- [18] Ulrich Lampe, Ronny Hans, and Ralf Steinmetz. "Will Mobile Cloud Gaming Work? Findings on Latency, Energy, and Cost." In: *Proceedings of the 2nd IEEE International Conference on Mobile Services (MS)*. 2013, pp. 960–961.
- [19] Ulrich Lampe, Melanie Siebenhaar, Ronny Hans, Dieter Schuller, and Ralf Steinmetz. "Let the Clouds Compute: Cost-efficient Workload Distribution in Infrastructure Clouds." In: *Proceedings of the 9th International Conference on Grid Economics and Business Models (GECON)*. 2012, pp. 91–101.
- [20] Ulrich Lampe, Qiong Wu, Sheip Dargutev, Ronny Hans, André Miede, and Ralf Steinmetz. "Assessing Latency in Cloud Gaming." In: *Proceedings of the 3rd International Conference on Cloud Computing and Services Science (CLOSER)*. 2013, pp. 52–68.
- [21] Ulrich Lampe, Qiong Wu, Ronny Hans, André Miede, and Ralf Steinmetz. "To Frag or to Be Fragged-An Empirical Assessment of Latency in Cloud Gaming." In: *Proceedings of the 3rd International Conference on Cloud Computing and Services Science (CLOSER)*. 2013, pp. 5–12.
- [22] Ulrich Lampe, Qiong Wu, Sheip Dargutev, Ronny Hans, André Miede, and Ralf Steinmetz. "Assessing Latency in Cloud Gaming." In: *Cloud Computing and Services Science*. Ed. by Markus Helfert, Frédéric Desprez, Donald Ferguson, and Frank Leymann. Springer, 2014, pp. 52–68.
- [23] Ulrich Lampe, Ronny Hans, Marco Seliger, and Michael Pauly. "Pricing in Infrastructure Clouds – An Analytical and Empirical Examination." In: *Proceedings of the 20th Americas Conference on Information Systems (AMCIS)* (2014), pp. 1–10.
- [24] The An Binh Nguyen, Melanie Siebenhaar, Ronny Hans, and Ralf Steinmetz. "Role-based Templates for Cloud Monitoring." In: *Proceedings of the IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*. 2014, pp. 242–250.
- [25] Björn Richerzhagen, Dominik Stingl, Ronny Hans, Christian Groß, and Ralf Steinmetz. "Bypassing the Cloud: Peer-assisted Event Dissemination for Augmented Reality Games." In: *Proceedings of the International Conference on Peer-to-Peer Computing (P2P)*. 2014, pp. 1–10.
- [26] Dieter Schuller, Ronny Hans, Sebastian Zöller, and Ralf Steinmetz. "On Optimizing Collaborative Manufacturing Processes in Virtual Factories." In: *Proceedings of the International Conference on Interoperability for Enterprise Systems and Applications (I-ESA), Workshop on ICT Services and Interoperability for Manufacturing*. 2014, pp. 66–74.

- [27] Dieter Schuller, Melanie Siebenhaar, Ronny Hans, Olga Wenge, and Ralf Steinmetz. "Towards Heuristic Optimization of Complex Service-based Workflows for Stochastic QoS Attributes." In: *Proceedings of the IEEE 21st International Conference on Web Services (ICWS)*. 2014, pp. 361–368.
- [28] Melanie Siebenhaar, Olga Wenge, Ronny Hans, Hasan Tercan, and Ralf Steinmetz. "Verifying the Availability of Cloud Applications." In: *Proceedings of the 3rd International Conference on Cloud Computing and Services Science (CLOSER)*. 2013, pp. 489–494.
- [29] Philipp Waibel, Johannes Matt, Christoph Hochreiner, Olena Skarlat, Ronny Hans, and Stefan Schulte. "Cost-optimized Redundant Data Storage in the Cloud." In: *Service Oriented Computing and Applications* 11.4 (2017), pp. 411–426.
- [30] Sebastian Zöller, Ronny Hans, Dieter Schuller, Ulrich Lampe, and Ralf Steinmetz. "Integrating Smart Objects as Data Basis in Virtual Enterprise Collaborations." In: *Enterprise Interoperability VI – Interoperability for Agility, Resilience and Plasticity of Collaborations*. Ed. by Kai Mertins, Frédéric Bénaben, Raúl Poler, and Jean-Paul Bourrières. Springer, 2014, pp. 297–305.

SUPERVISED THESES

- [1] Karim Abousedera. *Cost-efficient Selection of Cloud Providers under Consideration of QoS*. Master Thesis. Technische Universität Darmstadt, 2016.
- [2] Klajd Agoli. *Mutual Influence of Functional and Non-functional Properties in Service-based Systems*. Bachelor Thesis. Technische Universität Darmstadt, 2013.
- [3] Tayfun Atik. *Data Storage in the Cloud - Implementation of a DaaS-Interface*. Master Thesis. Technische Universität Darmstadt, 2012.
- [4] Syed Hissam Aziz. *Challenges in Mobile Cloud Gaming - An Analytical and Empirical Examination with a Focus on Energy Consumption*. Bachelor Thesis. Technische Universität Darmstadt, 2013.
- [5] David Dahlen. *Data Storage in the Cloud - Analysis of Database as a Service*. Bachelor Thesis. Technische Universität Darmstadt, 2012.
- [6] Marvin Dickhaus. *Reasons for the Usage of Cloud Computing and Requirements for cloud-based*. Master Thesis. Technische Universität Darmstadt, 2015.
- [7] Daniel Gutjahr. *FinTech: Analysis of the New Digital Ecosystem in the Financial Sector*. Studienarbeit. Technische Universität Darmstadt, 2016.
- [8] Dirk Heinen. *Blockchain in Finance: Transactions with and without Intermediate*. Studienarbeit. Technische Universität Darmstadt, 2017.
- [9] Martin Hellwig. *Implementation and Evaluation of a Mobile Cloud Gaming System based on Android*. Bachelor Thesis. Technische Universität Darmstadt, 2013.
- [10] Peter Jelesnak. *Challenges in Cloud Gaming - Analysis of Dependencies between Server Load and Quality of Service Attributes*. Dilpoma Thesis. Technische Universität Darmstadt, 2013.
- [11] Benjamin Klein. *Quality of Service Requirements of Mobile Cloud Applications - An Empirical Examination in the Financial Services Industry*. Master Thesis. Technische Universität Darmstadt, 2015.
- [12] Svenja Knafl. *Pricing in the Cloud Computing Market: An Analytical and Empirical Assessment of Infrastructure*. Studienarbeit. Technische Universität Darmstadt, 2015.
- [13] Sebastian Mihm. *Cloud Data Center Selection - Development of a Heuristic Approach*. Bachelor Thesis. Technische Universität Darmstadt, 2013.

- [14] Deniz Mohr. *Cloud Data Center Selection - Development of a Heuristic Improvement Approach*. Bachelor Thesis. Technische Universität Darmstadt, 2015.
- [15] Florian Staubach. *Dynamic Optimization of Interrelated Resource Allocation Decisions for Cloudlets*. Bachelor Thesis. Technische Universität Darmstadt, 2015.
- [16] David Steffen. *Cloud Data Center Selection - Development of Heuristic Approaches*. Studienarbeit. Technische Universität Darmstadt, 2014.
- [17] Marek Walasek. *Towards Pricing Fairness in Cloud Services: Modelling the Energy Cost of Virtual Machines in Infrastructure Clouds*. Master Thesis. Technische Universität Darmstadt, 2015.

D

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 20. Juni 2017

Dipl.-Wirtsch.-Ing. Ronny Hans

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available at <https://bitbucket.org/amiede/classicthesis/>.