Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. P. Tran-Gia

Optimization of Controller Placement and Information Flow in Softwarized Networks

Stanislav Lange

Würzburger Beiträge zur
Leistungsbewertung Verteilter Systeme

Bericht 3/18

Würzburger Beiträge zur
Leistungsbewertung Verteilter Systeme

Herausgeber

Prof. Dr.-Ing. P. Tran-Gia
Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Am Hubland
D-97074 Würzburg

Tel.: +49-931-31-86630
Fax.: +49-931-31-86632
email: trangia@informatik.uni-wuerzburg.de

Satz

Reproduktionsfähige Vorlage des Autors.
Gesetzt in LaTeX Linux Libertine 10pt.

ISSN 1432-8801

Optimization of Controller Placement and Information Flow in Softwarized Networks

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius–Maximilians–Universität Würzburg

vorgelegt von

Stanislav Lange

geboren in

Pawlodar

Würzburg 2018

Eingereicht am: 09.11.2018
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr.-Ing. Phuoc Tran-Gia
2. Gutachter: Prof. Dr.-Ing. Markus Fiedler
3. Gutachterin: Prof. Michela Meo
Tag der mündlichen Prüfung: 20.12.2018

Danksagung

In den Jahren, in denen diese Dissertation entstanden ist, haben mich zahlreiche Menschen motiviert und unterstützt. Da dies auf unterschiedliche Weise sowohl die Arbeit an sich als auch ihren erfolgreichen Abschluss ermöglicht hat, möchte ich mich herzlich bedanken.

In erster Linie möchte ich meinem Doktorvater Prof. Dr.-Ing. Phuoc Tran-Gia meinen Dank aussprechen. Er hat mir nach meinem Studium nicht nur die Möglichkeit gegeben, am Lehrstuhl für Kommunikationsnetze zu promovieren, sondern hat auch während der gesamten Zeit ein hervorragendes Arbeitsklima aufrechterhalten. Dadurch ermöglichte er mir neben einer schnellen fachlichen wie persönlichen Integration in die Gruppe auch die Entwicklung neuer Forschungsideen, welche durch gemeinsame Diskussionen vertieft und schließlich auf Fachkonferenzen präsentiert wurden.

Weiterhin gilt mein Dank Prof. Dr.-Ing. Markus Fiedler, der das Zweitgutachten meiner Dissertation übernommen hat und mich durch sein wertvolles Feedback sowie die wissenschaftlichen Diskussionen unterstützt hat. Ebenfalls danke ich Prof. Michela Meo, die das dritte Gutachten angefertigt hat. Des Weiteren bedanke ich mich herzlich bei Prof. Alexander Wolff, der sich trotz der kurzfristigen Anfrage bereit erklärt hat, als Prüfer bei meiner Disputation zu fungieren.

Besonderer Dank geht an meinen ehemaligen Gruppenleiter Prof. Dr. Thomas Zinner. Durch seine kontinuierliche Motivation und zahlreiche fachliche Diskussionen hat er meine wissenschaftliche Entwicklung enorm geprägt. Neben der Förderung in fachlicher Hinsicht möchte ich mich aber auch besonders herzlich für die Wegbereitung und Unterstützung bei meiner persönlichen Ent-

wicklung danken. Dank seiner Art konnte er ein sehr freundschaftliches Klima in der Gruppe schaffen und hat insbesondere auch Synergien zwischen den Mitgliedern gefördert, was nicht nur zu Wissenstransfer, sondern auch zur Bildung von Freundschaften geführt hat.

Bei den Dres. Rastin Pries, Michael Jarschel und Daniel Schlosser möchte ich mich für die Betreuung meiner Bachelorarbeit sowie für den damit frühen Kontakt mit OpenFlow und SDN bedanken. In dieser Zeit habe ich viele Grundlagen wissenschaftlicher Arbeit gelernt, die im weiteren Verlauf des Studiums sehr hilfreich waren. Ebenfalls bedanke ich mich bei Dr. Michael Seufert für die Betreuung meiner Masterarbeit und die Integration in zahlreiche Aktivitäten sowohl am Lehrstuhl als auch außerhalb.

Prof. Dr. Tobias Hoßfeld danke ich für meine frühzeitige Integration in den Lehrstuhl als Hiwi, welche mir zahlreiche wertvolle Erfahrungen bereitet hat und einen Vorgeschmack auf die Tätigkeit als wissenschaftlicher Mitarbeiter gegeben hat. Nach seiner Rückkehr nach Würzburg hat er mir sehr viel Vertrauen entgegengebracht und mir ermöglicht, weitere Erfahrungen bei der Definition und Organisation von Projekten zu sammeln. Auch für die Funktion als Prüfungsvorsitzender bei meiner Disputation danke ich ihm.

Für die durchgehend gemeinschaftliche Atmosphäre am Lehrstuhl sowie zahlreiche wissenschaftliche Kollaborationen und auch Aktivitäten außerhalb des Lehrstuhls bedanke ich mich außerdem bei allen ehemaligen und aktuellen Kolleginnen und Kollegen. Besonderer Dank gilt Dr. Steffen Gebert und Stefan Geißler, die über die Jahre mit mir ein Büro geteilt haben. Außerdem bedanke ich mich bei den Gruppenleitern Dr. Matthias Hirth, Dr. Florian Wamser und Dr. Florian Metzger, die mir mit ihrer Erfahrung in vielen Aspekten geholfen haben. Auch für die enge Zusammenarbeit mit den Mitgliedern der NGN-Gruppe – Dr. Steffen Gebert, Stefan Geißler, Nicholas Gray, Alexej Grigorjew, Dr. Anh Nguyen-Ngoc, Susanna Schwarzmann, Prof. Dr. Thomas Zinner – sowie für die wöchentlichen Gruppentreffen und die damit verbundenen anregenden Diskussionen bedanke ich mich herzlich. Weiterhin danke ich Kathrin Borchert, Valentin Burger, Lam Dinh-Xuan, Dr. Michael Duelli, Dr. Matthias Hartmann, Dr.

David Hock, Dr. Dominik Klein, Frank Loh, Christopher Metter, Christian Moldovan, Dr. Christian Schwartz und Anika Schwind für die gemeinsame Zeit am Lehrstuhl.

Darüber hinaus danke ich Alison Wichmann für die Hilfe bei der Projektverwaltung und auch für die Unterstützung bei der Abwicklung von Dienstreisen. Zudem bedanke ich mich bei allen Hiwis und Studenten, mit denen ich über Vorlesungen, Übungen, Seminare und Praktika oder Abschlussarbeiten in Kontakt gekommen bin, für die gute Zusammenarbeit.

Abschließend richte ich riesigen Dank an meine Mutter Maria, meine Oma Hilda, meine gesamte Familie und alle meine Freunde. Ohne deren Beistand, Rat und Unterstützung wäre diese Arbeit nicht möglich gewesen. Besonders meiner Mutter Maria danke ich von Herzen für die durchgehende Förderung und Motivation seit Schulzeiten, welche das Fundament für das Studium und damit auch die Promotion gelegt haben. Dank ihrer Beratung bei kleinen und großen Hürden konnte ich die Arbeit letzten Endes abschließen.

Contents

1 Introduction

Current communication and networking use cases such as video streaming, vehicular communications, and the Internet of Things (IoT) confront network operators with numerous challenges. Firstly, the continuously increasing number of devices and services leads to a corresponding growth in terms of resource demands and therefore requires systems that can scale appropriately. Secondly, heterogeneous communication paradigms and requirements as well as their temporal dynamics require a high degree of automation and adaptability in order to maintain high performance levels while operating the network in a resource-efficient manner.

These challenges are addressed by softwarization paradigms such as Software Defined Networking (SDN) and Network Functions Virtualization (NFV). By separating the data plane and the control plane as well as logically centralizing the latter in a software-based controller that runs on commodity hardware, the SDN paradigm enables flexible network configuration that is required to quickly adapt to dynamically changing network conditions. Additionally, by providing open interfaces for the communication between the control entity and components from the data plane, applications, legacy devices, and other SDN domains, a high degree of network programmability can be achieved [26]. These interfaces are referred to as southbound, northbound, eastbound, and westbound API, respectively and the resulting SDN architecture is depicted in Figure 1.1. The outlined programmability, in turn, allows automating many vital traffic engineering tasks like policy-based rerouting of specific flows without manual reconfiguration.

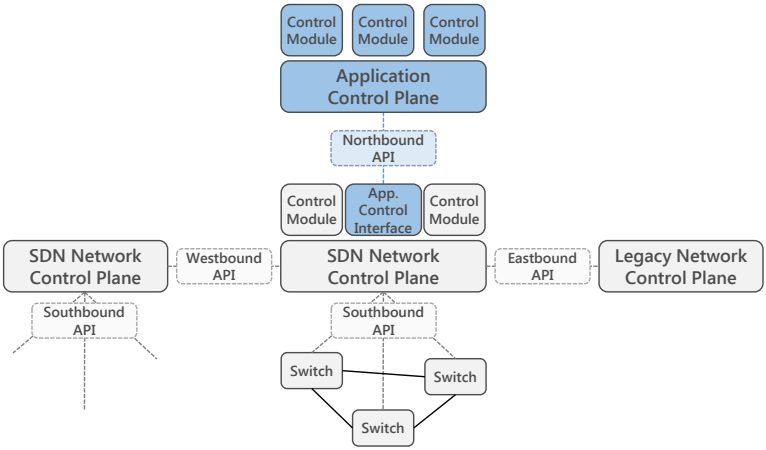


Figure 1.1: Interfaces in the SDN architecture [26].

In the case of NFV, specialized hardware middleboxes that perform networking tasks like firewalling and load balancing are replaced with software instances that run on commercial off-the-shelf (COTS) servers. These software instances are referred to as Virtualized Network Functions (VNFs) and can be dynamically deployed, migrated, and scaled. Therefore, following the NFV paradigm can increase the resource efficiency by instantiating only as many VNFs as are required to handle the network traffic at a given time. Similarly to SDN, management and orchestration interfaces that are defined in modern NFV architectures such as the one by the European Telecommunications Standards Institute (ETSI) [27] or the OpenStack project [28] enable programmability and automation that are required to maintain the aforementioned resource efficiency.

In order to fully reap the benefits of these paradigms, however, novel challenges need to be tackled. These include the management and orchestration of

softwarized networks as well as performance considerations that result from replacing special-purpose hardware with software that runs on COTS hardware.

Due to the fact that numerous tasks that were previously performed by dedicated hardware elements are taken over by software entities, the latter need to be carefully dimensioned in order to meet stringent performance requirements. In this context, measurements and performance benchmarks of individual components like SDN controllers and virtualized network functions can provide valuable insights into aspects like their capacity, resource usage, and operational regimes. Similarly, performance data of novel components like SDN-enabled switches that can be either software- or hardware-based is required. While measurements can provide insights into particular constellations regarding parameters like traffic characteristics and intensity, they can also serve as input for analytical models. Such models abstract the components' behavior and can be used to predict their performance under various conditions, optimize their parameters to fit a particular use case, or identify bottlenecks.

Furthermore, both SDN and NFV pose placement problems in which the number and location of entities such as SDN controllers and virtualized network functions is optimized according to multiple objectives. On the one hand, this requires developing algorithms that can cope with the huge parameter space of such a multi-objective optimization task in a timely manner. On the other hand, mechanisms that automatically choose one distinct solution from the resulting set of Pareto optima are required in order to maintain a high performance and efficiency even in the face of highly dynamic environments. For both placement tasks, results from the abovementioned benchmarks and models constitute a crucial input since they provide information on the capacity and imposed delay of placed components.

Finally, when considering softwarized networks from an architectural point of view, an interaction between the SDN control plane and existing centralized instances like Network Management Systems (NMSs) can be beneficial to the performance of the entire network. This performance improvement can be achieved by providing the entities with detailed monitoring and configuration

data that they can integrate into their control and management decisions, respectively.

The following two sections cover research in the area of softwarized networks that has been conducted by the author and give an overview of the scientific contributions that are discussed in detail in this monograph, respectively.

1.1 Scientific Contribution

As outlined in the previous paragraphs, the domain of softwarized networks spans a wide research field that exceeds the capacity of a single monograph. Hence, in order to provide a clear context, we first provide an overview of research activities in Figure 1.2. The selected subset of topics and references that this monograph covers in detail is highlighted in the graphic alongside the corresponding chapter.

We categorize the research with respect to two dimensions. While the x-axis represents the main topic of a work, i.e., SDN or NFV, the y-axis is used to classify by methodology. The latter can be theory-focused as in the case of analytical models, mathematical optimization approaches, and algorithm design or focused on practical aspects like measurements, benchmarks, and performance evaluations of proof-of-concept deployments. Additionally, topics that are closely related to each other, are grouped via dashed boxes.

Placement Optimization. Placement problems play a crucial role in both, the SDN [1, 8] as well as the NFV [16, 19] context. On the one hand, they share similarities in terms of high level goals like finding suitable locations for entities while simultaneously optimizing multiple objectives. Additionally, they are usually approached with heuristics due to the fact the corresponding parameter space prohibits an exhaustive evaluation of all possible placements. On the other hand, additional requirements in the context of VNF placement like the availability of different functions and function types, the routing of demands through function chains, and meeting QoS constraints call for new algorithms that are specifically tailored to the placement of VNFs rather than SDN controllers. In

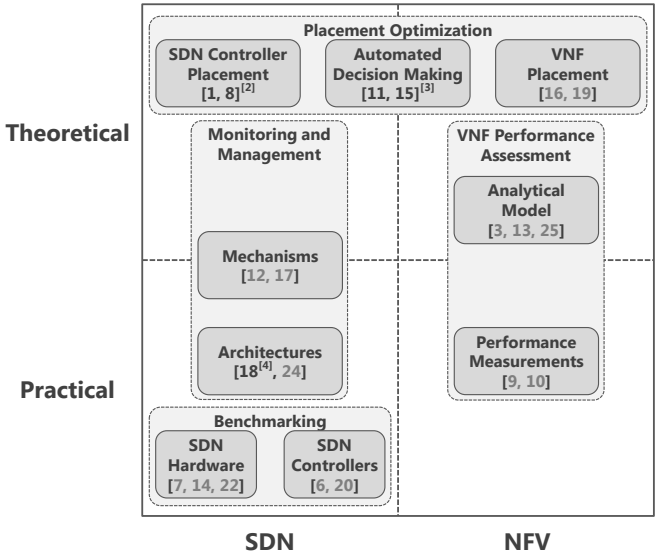


Figure 1.2: Scientific contribution of this work as a categorization of research studies conducted by the author. References that use a dark font color are covered in this monograph while superscript indexes indicate the corresponding chapter. For example, the notation $[1, 8]^{[2]}$ *in the case of the topic "SDN Controller Placement" means that this topic is addressed in Chapter 2 and is based on references [1] and [8].*

both cases, we design generic as well as specialized multi-objective heuristics that are evaluated on a wide range of real world topologies and parameters in order to highlight their applicability to realistic large scale problem instances. These evaluations also cover trade-offs between the different optimization objectives as well as trade-offs between the available run time and the resulting accuracy.

Rather than returning one distinct optimum, these multi-objective optimization algorithms return a Pareto frontier, i.e., a set of solutions that represent different trade-offs between possibly competing objectives. In the second graphic of Figure 1.3, an exemplary Pareto frontier is represented by the blue squares that correspond to trade-offs between minimization objectives f_1 and f_2 that are displayed on the x- and y-axis, respectively. Since these solutions are incomparable among each other, we study automated decision making mechanisms for Pareto frontiers. Such decision making mechanisms constitute an important building block for systems that can adapt to dynamically changing conditions without human intervention. The latter is particularly error-prone when dealing with thousands of available alternatives and high-dimensional data that are common in the context of placement problems. We evaluate the mechanisms using instances of the controller placement problem and show a high level of agreement in terms of the top-ranked solution [11, 15], indicating the applicability of these mechanisms in this context.

Monitoring and Management. SDN-based networks not only offer novel approaches to network control and programmability but also enable new monitoring concepts by exposing statistics like packet counters on a per-flow basis. We leverage this behavior to develop and evaluate a technique for identifying heavy-hitting flows in a network [12]. In addition to this specific use case, we investigate selective flow monitoring strategies that aim at increasing the general monitoring efficiency in SDN-based networks. In particular, querying long-lasting flows less frequently usually does not have a large impact on the accuracy while significantly reducing the amount of sent statistics requests [17].

Furthermore, SDN controllers offer a northbound API for communicating with external entities like Network Management Systems (NMSs). Hence, it is not only possible to share monitoring information with these systems in order to improve their management decisions but also to receive and integrate their information into the controller's control plane. We demonstrate the feasibility of such an information exchange in [18], where detailed information regarding link utilization in a network is sent from an NMS to the SDN controller in order to optimize the total throughput and per-flow fairness in the network. Additionally, [24] shows how a heterogeneous network that is comprised of SDN-enabled switches as well as legacy devices can be managed and configured by synchronizing state information between SDN controllers and NMSs.

Benchmarking. In order to properly dimension and operate an SDN-based network, it is required to accurately assess the capabilities as well as the performance of the components in use, i.e., switches and controllers. To this end, we conducted several performance benchmarks of SDN-enabled hardware switches w.r.t. the flow rule installation time [14] and its sensitivity towards control plane delays [22]. While these benchmarks allow quantifying the heterogeneous performance of available hardware, functional tests regarding the isolation between virtual networks [7] can be used to determine the suitability of a switch for a particular use case. Likewise, performance benchmarks of various SDN controllers demonstrate that characteristics of the network topology like the number of switches and links can have a significant impact on crucial performance metrics like the path provisioning time and the network discovery time [6, 20].

VNF Performance Assessment. Similarly, performance aspects need to be taken into consideration when migrating from dedicated hardware to VNFs on COTS servers or deciding between different implementations of the same network function. On the one hand, we demonstrate the performance gap in terms of the processing time and maximum capacity between a dedicated firewall middlebox and a software-based implementation in [10]. On the other hand, we illustrate that while the use of acceleration techniques can significantly improve

the packet processing performance of VNFs, it might also reveal new bottlenecks that need to be addressed to maximize the overall throughput [9].

With an in-depth understanding of the technical system and its components as well as measurements of characteristics like the service time of individual packets, analytical models of VNFs are derived [25, 13, 3]. Firstly, these models allow to predict the performance of network functions under various conditions like different load levels, packet arrival processes, and parameter settings. Secondly, the reported performance indicators such as waiting and processing times are provided not only in terms of their mean but also w.r.t. their distribution, allowing a detailed assessment of the performance as well as an optimization of parameter settings. Finally, measurements in a dedicated testbed confirm the agreement between the output of the abstract model and the actual VNF.

1.2 Outline of the Thesis

The remainder of this monograph is organized as follows. Chapters 2 to 4 cover the SDN controller placement problem, automated decision making mechanisms, and the improved NMS-aware SDN controller, respectively. In each of these chapters, a brief discussion of related work, our proposed approach, as well as its performance evaluation methodology and results are provided. Finally, Chapter 5 concludes the monograph with a summary of results and contributions. Figure 1.3 displays the structure and interplay between the three main chapters that are outlined in the following.

SDN Controller Placement. A particularly important task in SDN architectures is that of controller placement, i.e., identifying controller locations that simultaneously optimize multiple objectives such as the number of controller instances, network delays, and the load distribution among instances. Due to its complexity and large parameter space, we approach this problem with heuristics that trade-off accuracy against run time [1, 8]. The top part of Figure 1.3 illustrates this step. Given an input topology, multi-objective optimization algorithms calculate a Pareto frontier, i.e., a set of possible solutions that repre-

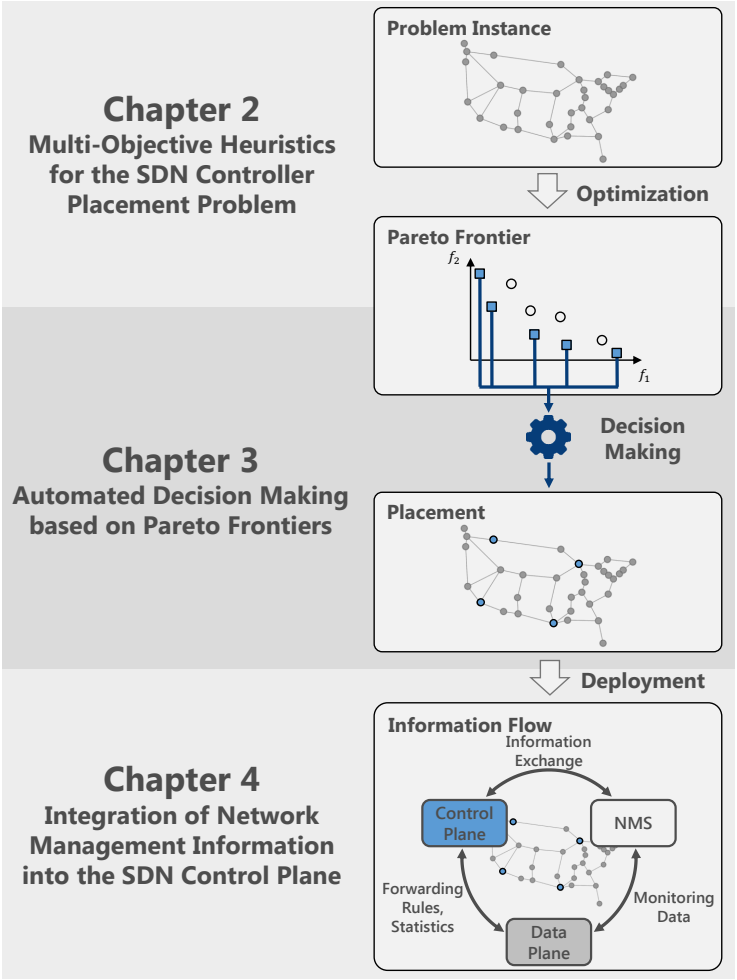


Figure 1.3: Overview of the structure and contribution of this monograph.

sent different trade-offs between the competing objectives and are incomparable among each other. We design two different heuristics, a generic one that supports arbitrary objective functions as well as a specialized one that works only with a particular set of objectives. Our evaluations on more than 50 different real world topologies from the Internet Topology Zoo [29] highlight their feasibility in the context of large scale problem instances and illustrate the performance gains that can be achieved by means of specialization.

Automated Decision Making. The multi-objective nature of the placement problem results in sets of Pareto optimal solutions rather than distinct optima. Therefore, scenarios with dynamically changing network conditions require mechanisms for automated decision making based on such Pareto frontiers in order to function without manual interaction. Hence, we investigate techniques from the domain of multi-attribute decision making that aggregate the performance of placements into a single numeric score and compare the resulting rankings. Evaluations featuring real world topologies demonstrate the viability of the proposed mechanisms as well as their agreement regarding top-ranked solutions [11, 15].

Interaction between SDN and NMS. Finally, an integration of SDN components into existing ecosystems is required for a smooth transition from legacy to SDN-based networks. One of the key aspects for this integration consists of interactions between the SDN controller and other centralized entities such as Network Management Systems. In particular, we focus on improved SDN control plane decisions based on monitoring data that is regularly provided by an NMS. To this end, we design, implement, and compare two variants of the ONOS controller. Alongside the default implementation, these represent different trade-offs regarding the complexity of the resulting system and its performance. In addition to evaluations that show a significant performance improvement when using the optimized controllers, a parameter study investigates the performance impact of network characteristics [18].

2 Multi-Objective Heuristics for the SDN Controller Placement Problem

Architectures that follow the SDN paradigm are characterized by the separation of the control and the data plane as well as a logically centralized control plane. This is achieved by moving control plane functions from individual network devices to a dedicated controller software that runs on commodity hardware. Communication between this centralized control plane and the data plane is then performed via the southbound API [26] which is implemented by protocols like OpenFlow [30]. Furthermore, the scalability and resilience of an SDN infrastructure can be enhanced by physically distributing the logically centralized control plane as proposed by approaches like HyperFlow [31] and ONOS [32].

However, such a physically distributed control plane also introduces additional challenges. These include not only identifying the number of SDN controllers that are required for a targeted performance or resilience level, but also the appropriate placement of these controllers, based on the relevant objectives for a given use case. These objectives cover aspects like load balancing among controller instances and communication delays between the involved control and data plane instances in heterogeneous network environments. Furthermore, operators often need to cope with dynamically changing network conditions [33] which require a periodic recalculation of viable placements in order to adapt to these changes in an appropriate manner. Various investigations of traffic characteristics [34, 35] and traffic engineering approaches in data

center networks [36] have demonstrated that techniques for dealing with such dynamics need to operate at the time scale of seconds in order to maximize their effectiveness. Therefore, the time consumption of a placement algorithm constitutes one of its key performance indicators.

The controller placement problem for the SDN domain was first introduced in [37], where an optimization regarding the latency from nodes to their assigned controller is performed. This optimization is equivalent to the *facility location problem*, a task which is known to be NP-hard [38]. The authors of [37] perform an exhaustive evaluation of all possible placements in order to analyze the characteristics of the optimal solutions. However, the run time and memory requirements of the exhaustive approach do not scale well with the network size and are therefore not sufficient to cope with the abovementioned dynamics. Hence, we explore heuristic approaches to the multi-objective controller placement problem. While such approaches do not guarantee to find optimal solutions, they are capable of yielding results significantly faster than their exhaustive counterpart.

An extended version of the controller placement problem is investigated in [39] and deals with multiple optimization objectives, e.g., resilience considerations and inter-controller latencies. In realistic environments, such performance objectives are often competing, thus there is usually no definite solution that satisfies all goals optimally. Instead, a trade-off between the competing objectives that fits the particular use case needs to be chosen. On the one hand, the multi-objective approach eliminates the necessity to impose constraints on objective values a priori which might result in excluding feasible alternatives or even a problem instance without admissible solutions. On the other hand, it allows for a clearer illustration of the trade-offs between competing criteria. Additionally, the decision maker's preferences with respect to the different objectives usually depend on available alternatives [40]. This, in turn, decreases the feasibility of transformations to single objective optimization problems, e.g., via a weighted combination of individual objectives.

In this chapter, we design and evaluate two multi-objective combinatorial optimization heuristics in order to solve large scale instances of the controller placement problem in a time- and resource-efficient manner while maintaining a high degree of accuracy. These two algorithms differ in terms of their degree of specialization. On the one hand, an algorithm that is based on Pareto Simulated Annealing (PSA) [41] allows optimizing with respect to arbitrary objective functions. On the other hand, a mechanism that is based on the k-Medoids approach [42] specializes in optimizing a set of exactly two particular objectives but achieves lower run times and results that are closer to the optimum.

Our evaluations are performed on a collection of over 60 real world network topologies from the Internet Topology Zoo [29]. Furthermore, we utilize reference solutions that are obtained by means of an exhaustive evaluation of all possible placements in order to quantify the gap between optimal and heuristic solutions. Additionally, the results can also be used to infer guidelines regarding the parameter and algorithm choice in the context of huge problem instances where reference values are not available. Finally, we integrate the proposed algorithms into POCO [43], a software framework that deals with the controller placement problem and features a graphical user interface.

This chapter is based on content that has been published in [1, 8] and is structured as follows. In Section 2.1, we discuss the main principles of softwarized networks as well as relevant related work on the mathematical background of the presented problem and work that is related to the controller placement problem in particular. After formally defining the problem statement and the corresponding notation in Section 2.2, the two heuristic algorithms are presented alongside their evaluation in Sections 2.3 and 2.4, respectively. Finally, we discuss lessons learned in Section 2.5.

2.1 Background and Related Work

In this section, we first provide the necessary background regarding the control plane in SDN-based networks. Subsequently, we give an overview of related

work, including work on the particular problem of SDN controller placement as well as literature that covers the underlying facility location problem. Finally, we discuss different approaches to the problem of multi-objective optimization.

2.1.1 SDN Control Plane

The key principle of SDN consists of the separation of data plane and control plane. The design of the externalized control plane can be performed in different ways. On the one hand, there is a choice between controller architectures, e.g., featuring a single, centralized controller, multiple equally important controllers responsible for partitions of the network [44], or a set of distributed controllers arranged in a hierarchy [45]. On the other hand, purely software-based or hardware solutions for the control plane allow trade-offs between flexibility, performance, and cost aspects. Furthermore, connections involving controllers may be realized inband or outband, i.e., either sharing the same physical links as data plane traffic or utilizing dedicated lines.

Typically, core networks contain pre-configured primary and backup paths for the aggregated traffic between different nodes in the network. Thus, there is no need for communication with the SDN controllers regarding each individual TCP flow but only in case of certain occasions like outages or traffic management actions. In such environments, a single controller can be sufficient for a viable network operation. Nonetheless, growing network sizes and the importance of resilience against failures increase the amount of required controllers. Moreover, network functions that are deployed on the SDN control plane further raise its load, leading to even higher numbers of required controllers. For the remainder of this paper, it is assumed that all sites possess the capability to run a software-based SDN controller. Furthermore, communication with SDN controllers is assumed to be performed inband, i.e., via the same physical links as regular traffic.

2.1.2 Controller Placement in SDN-based Networks

Apart from the aforementioned publication by Heller et al. [37], more and more authors have addressed facility location in the context of SDN controller placement.

Bari et al. [33] address dynamic controller provisioning, i.e., controller placements changing over time depending on the number of flows in the network. They propose an Integer Linear Program (ILP) formulation of their "Dynamic Controller Provisioning Problem" as well as two different heuristic algorithms to solve it for larger problem instances. The authors focus their metrics on flow setup time and minimal communication overhead regarding state synchronization. However, controller failures, network failures, or a combination of multiple criteria such as the controller load imbalance or worst case latencies are not addressed.

Zhang et al. [46] propose a resilient optimization of the controller placement that considers the outage of nodes, links, or connections between nodes and controllers. They do not reassign nodes to new controllers if the connection to the original controller fails, but assume that these nodes are controller-less and thus not able to communicate with other nodes anymore. They propose a placement heuristic and simulation with the objective of minimizing the amount of lost node-to-node routes due to link and node failures as well as controller-less nodes.

The works of Hu et al. [47, 48] go in a similar direction. They introduce and compare different heuristic approaches to increase the resilience of software defined networks against connection failures between nodes and controllers. Ros et al. [49] also consider similar scenarios and aim at maximizing the reliability of the controller placement. They heuristically search for the minimum number of controllers that are assigned to each node and the controllers' placement to reach a certain reliability threshold, e.g., "five nines".

All these studies [46–48] focus only on resilience against network failures and do not consider any additional metrics such as the controller load imbalance or worst case latencies. In particular, the trade-off between their metrics and

other objectives, such as the worst case latency, is not addressed. Furthermore, compared to the evaluation of the entire solution space, no guarantee for the optimality of the presented results can be given.

2.1.3 Facility Location Problem

As already mentioned and indicated by Heller et al. [37], the topic of general controller placement is well explored. In particular, the very basic version of controller placement according to the latency of switches to their controller is also well discussed in the context of choosing the best location for plants, warehouses, or any other facilities in a given network topology. The problem is therefore also known as *plant*, *facility*, or *warehouse location problem* and it is a typical example for a Mixed Integer Linear Program (MILP) provided with software suites such as *IBM ILOG CPLEX* [50].

If the objective is to minimize the maximum latency between switches and controllers, the problem is called *k-centers problem*, if the objective consists of optimizing the average latency, it is called *k-median* or *k-mean problem*. Further references to this general problem are provided in Heller's work [37]. Overviews of different aspects of the facility location problem and of different methodological approaches are also given in [38] for the general case and in [51] with a focus on uncertainty regarding aspects such as traffic demands or latencies. However, these works have a rather general and theoretical focus. They do not address the particular issues of controller placement in SDN networks with respect to multiple criteria and a focus on resilience. The following overview of related work focuses on variants of the controller placement problem which are closely related to the problems discussed in this chapter.

A variant of the problem similar to the node-to-controller balancing discussed in this chapter has been introduced by Archer et al. [52] as *load-balanced facility problem*. However, the authors address this problem in a different context concerning particular questions arising in the area of computer graphics. Furthermore, they provide only approximations to the problem regarding their par-

ticular optimization goals. In the context of load balancing, the terms *capacitated* and *uncapacitated facility problem* are often used, e.g., [53] and contained references. In the capacitated version of the problem, the maximum number of nodes that can be assigned to a single controller is assumed to be limited.

Different authors, among others Khuller et al. [54] and Chaudhuri et al. [55], look at variants called *fault tolerant* or *p-neighbor k-center problems*. These variants are similar to what is called "controller failure resilient placements" here. The works focus only on the theoretical methodology of the problem and provide approximation algorithms.

2.1.4 Multi-Objective Optimization Algorithms

For a given combination of objectives, there are various approaches for multi-criteria facility location in literature, e.g., [56–61] and references within. However, most of these works investigate optimization approaches for specific predefined sets of objectives rather than providing generic heuristics. Algorithms dealing with the aforementioned capacitated facility location problem, for example, consider the equivalent of the average switch to controller latency and controller load imbalance metrics used in this work. Metaheuristics such as the presented Pareto Simulated Annealing (PSA)[41] mechanism, on the other hand allow adding arbitrary objectives into the evaluation and are not limited with respect to the number of objectives that are taken into account during optimization. The only requirement is a function that maps elements of the search space to their performance regarding a particular objective. While techniques from the domain of evolutionary algorithms [40, 62] or genetic algorithms [63] in particular are also capable of performing multi-objective optimization, these algorithms are often at risk of getting stuck in local optima [64, 65], i.e., solutions that are optimal within their immediate neighborhood in the search space but can be significantly worse than the global optimum. PSA reduces this risk by accepting some worse solutions in order to escape such local optima while still achieving convergence by employing a time dependent acceptance probability.

2.2 Problem Statement

This section provides a formal definition of the SDN controller placement problem alongside the necessary notation.

2.2.1 Controller Placement Problem

While minimizing latencies between each node and its assigned controller constitutes a crucial aspect of the controller placement problem, there are numerous other, possibly competing, objectives that require consideration. In the following, objectives that are covered in this work are presented along with examples that motivate their necessity in different use cases. The Internet2 OS3E network is used as example topology and the best placement with respect to maximum node-to-controller latency for $k = 5$ controllers as shown in the work of Heller et al. [37] acts as reference. Figure 2.1 displays this placement's performance when evaluated with respect to different objective measures and different conditions, e.g., latencies and load balance in the presence of node or controller failures. In contrast to the work of Zhang et al. [46], this work assumes that the node-to-controller assignment can change when failures occur.

Figure 2.1a illustrates the latency between each node and its assigned controller when multiple controllers stop working. Each node's color indicates its latency to the closest functioning controller. The latency is normalized with the graph's diameter which is defined as the maximum distance in terms of latency between any pair of nodes in the graph. In particular, the green color represents a latency of zero, the yellow color represents a latency that corresponds to 50% of the graph's diameter, and the red color indicates a latency that is equal to the diameter. While the distributed controller structure asserts low latencies in the failure-free case [37], the illustrated failure scenario highlights the fact that in the presence of failures, the position of each controller matters. The only controller that is not affected by the failure is located at the edge of the network and thus, control traffic of many nodes needs to traverse almost the entire network. In order to better cope with scenarios of this kind, the optimization mechanism

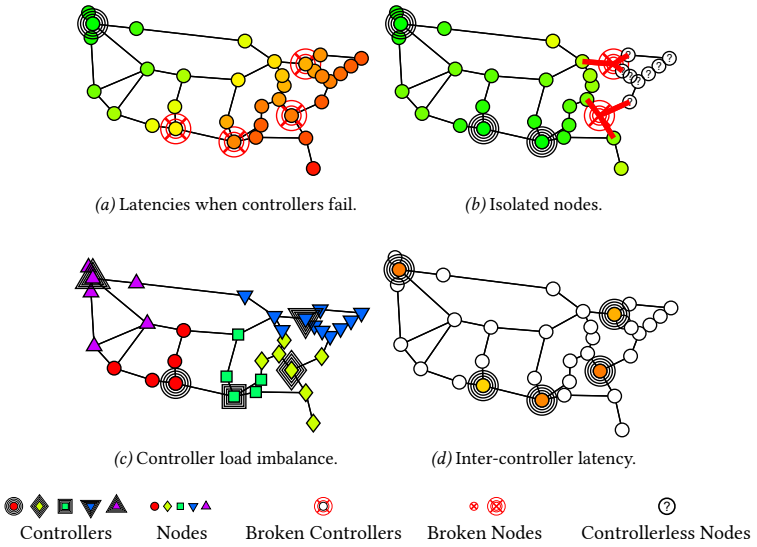


Figure 2.1: Assessing the quality of controller placements with different objective measures.

for resilient controller placements should consider failure scenarios. Depending on the specific use case, average and maximum latencies might be useful measures.

In addition to controller failures caused by software issues, physical network elements may also suffer from hardware problems and stop working. This type of failure has more severe consequences as it induces changes in the topology, which in turn results in changes of shortest paths. Thus, node-to-controller assignments tend to change and in extreme cases, the network graph can even become disconnected. While the nodes in each connected component are still fully operational, not being able to connect to any controller in the connected

component prohibits any functionality beyond forwarding according to previously installed rules. Figure 2.1b displays the case of two network nodes failing at the same time, which results in both of the aforementioned phenomena. First, the graph is decomposed into two disjoint components. Second, the right part of the graph does not contain a controller. Thus, nodes in this part of the graph lose access to any functionality realized by the controller. These nodes are represented by question mark icons ⍰.

Assuming that nodes connect to their nearest controller, certain placements tend to result in imbalanced assignments, i.e., some controllers provide instructions for significantly more switches than others. Consequently, environments with a high intensity or frequency of control plane communication can run into problems, like increased delays, due to queueing at controller instances. Figure 2.1c illustrates the imbalance aspect of the latency-optimal placement. Each node is colored and shaped according to the controller it is assigned to. While the blue controller is responsible for ten network elements, the green and red controllers need to manage just four nodes, i.e., less than half as many. Additionally, previous investigations show [66] that for some controller implementations, the number and order of connected switches might cause unfairness with respect to aspects like the switches' flow setup times. Thus, load balancing should be a part of the decision criteria when choosing a controller placement. Additionally, the *link assignment* task presented in [44] corresponds to minimizing the imbalance, further supporting the relevance of this aspect.

Previously discussed scenarios, especially those involving failure tolerance, indicate the necessity of a distributed control plane. However, such an architecture also requires various forms of state synchronization between the individual controllers. This ensures proper functionality in case of outages and allows making decisions that are not limited to a local view on a part of the network. Therefore, another goal of the controller placement task is to maintain a small inter-controller latency in order to minimize synchronization times. A visualization of this measure is provided in Figure 2.1d, where each controller is colored according to the distance to the controller that is farthest away from it. Like

in Figure 2.1a, the distances are normalized with respect to the graph's diameter. For most controllers, the shown placement results in high maximum latencies to other controllers which might be not acceptable for certain use cases. Therefore, inter-controller latency is part of the set of objectives that are analyzed in this work. Additionally, the pair of inter-controller latency and node-to-controller latency constitutes a set of competing objectives. While a tight cluster of controllers results in low inter-controller latencies and high node-to-controller latencies, a spatially widespread distribution of controllers leads to the opposite. Such relationships between objectives are the motivation for the analysis of Pareto optimal placements, which allows decision makers to express their preferences after inspecting possible placements.

Furthermore, the set of objectives taken into account is not restricted to those presented in this section and can be extended according to use case specific requirements. These also include management aspects which could be introduced as separate constraints. For example, metrics like the expected service quality provided by a network in the context of a given placement could be added into the evaluation and decision process.

2.2.2 Notation

In this section, the notation that is used throughout this chapter is introduced. For reference, Table 2.1 provides a summary of variables and functions.

Formally speaking, the controller placement problem is a multi-objective combinatorial optimization (MOCO) problem. The network is represented as a graph $G = (V, E)$ with the set of nodes V that contains n nodes which are connected by edges from the set E. Additionally, shortest path latencies between each pair of nodes are stored in a distance matrix D, where $d_{i,j}$ denotes the latency from node i to node j. Latencies in D are normalized with the graph's diameter, i.e., $d_{i,j} \in [0, 1]$. Given the desired number of controllers k, there is a finite set of $\binom{n}{k}$ possible placements, hence the term *combinatorial optimization*. The goal of the MOCO task is to find controller placements from the set of size k

Table 2.1: Overview of the notation that is used in this chapter.

Symbol	Description
$G = (V, E)$	Network graph.
$v \in V$	Set of nodes in the network graph.
n	Number of nodes in the network graph, $n = \|V\|$.
$e \in E$	Set of edges in the network graph.
k	Number of controllers to be placed in the network.
$D, d_{i,j} \in [0, 1]$	Distance matrix with entry $d_{i,j}$ representing the normalized latency between nodes $i, j \in V$.
$\mathcal{P}_k = \left\{\mathcal{P} \in 2^V \mid \|\mathcal{P}\| = k\right\}$	Set of all size k placements with 2^V denoting the power set of V, i.e., the set of all subsets of V.
$f_i \colon 2^V \to \mathbb{R}_0^+$	Objective functions that map placements to numerical values, $i \in \{1, \ldots, J\}$.
w_i	Normalization factor for f_i, $w_i^{-1} = \max_{p \in 2^V} f_i(p) - \min_{p \in 2^V} f_i(p)$.
$c \colon 2^V \times 2^V \to [0, 1]$	Distance measure for comparing placements.
$\delta_i \colon 2^{(2^V)} \times 2^{(2^V)} \to [0, 1]$	Distance measures for comparing Pareto frontiers, $i \in \{1, 2\}$.

placements $\mathcal{P}_k = \left\{\mathcal{P} \in 2^V \mid \|\mathcal{P}\| = k\right\}$ that are Pareto optimal with respect to various objective functions f_i ($i \in \{1, \ldots, J\}$). These are discussed informally in the previous section and are presented in detail in the following. A placement x is considered Pareto optimal, if and only if there is no placement y such that y is at least as good as x for all objectives and strictly better than x for at least one objective, i.e., $\forall i\ f_i(y) \leq f_i(x)$ and $f_i(y) < f_i(x)$ for at least one i. The set of all Pareto optimal solutions is referred to as Pareto frontier.

While it is possible to perform an exhaustive evaluation of all $\binom{n}{k}$ possible placements for small and medium sized networks, a heuristic approach is suggested in case of instances that are too huge to be fully evaluated in a practical time frame. For example, when increasing the desired number of controllers beyond 7, an exhaustive evaluation of a network with 50 nodes can take between several minutes and hours on a machine with an Intel Core i7 4770 CPU at 3.40 GHz and 16 GB of RAM.

In order to quantify the loss of accuracy caused by switching to the heuristic approach, different measures for the difference between the actual and the estimated Pareto frontier have been adopted from [41]. In the following, R denotes the original Pareto frontier which is used as reference, and M represents the estimate provided by the heuristic approach. Before the distance between Pareto frontiers is defined, a distance metric for two placements is introduced. According to Equation 2.1, $c(x, y)$ defines the distance between two placements as the maximum weighted difference between individual objective values achieved by the placements. The weight w_j corresponds to objective f_j's range and is used for normalization, i.e., $c(x, y) \in [0, 1]$. Adding zero to the argument of the maximum asserts that no negative distance is returned. With this distance metric, it is possible to define measures for the distance between two Pareto frontiers, of which one is known to be better than the other and is therefore used as reference. The first metric, δ_1, is shown in Equation 2.2 and measures the average distance between each element in R and its closest element from M. While δ_1 measures the average difference and may hide outliers, δ_2 considers the maximum distance between each element from R and its closest element in M. Its formal definition is provided in Equation 2.3 and allows for a worst case analysis of the estimate M:

$$c(x, y) = \max_{j=1,\dots,j} \{0, w_j(f_j(x) - f_j(y))\}, \tag{2.1}$$

$$\delta_1(R, M) = \frac{1}{|R|} \sum_{y \in R} \left\{ \min_{x \in M} \{c(x, y)\} \right\}, \tag{2.2}$$

$$\delta_2(R, M) = \max_{y \in R} \left\{ \min_{x \in M} \{c(x, y)\} \right\}. \tag{2.3}$$

As motivated in Section 2.2.1, numerous competing objective functions need to be considered when evaluating a given controller placement. In the following, we provide formal definitions of objective functions that are commonly used in the context of the SDN controller placement problem. Additionally, an overview of the functions alongside a brief description is provided in Table 2.2.

Table 2.2: Overview of the main objective functions that are used in this chapter.

Function	Description
$\pi^{\text{max latency}} \colon 2^V \to [0, 1]$	Maximum node-to-controller latency.
$\pi^{\text{avg latency}} \colon 2^V \to [0, 1]$	Average node-to-controller latency.
$\pi^{\text{max controller-latency}} \colon 2^V \to [0, 1]$	Maximum inter-controller latency.
$\pi^{\text{avg controller-latency}} \colon 2^V \to [0, 1]$	Average inter-controller latency.
$\pi^{\text{imbalance}} \colon 2^V \to \mathbb{N}_0$	Controller imbalance.

First, the node-to-controller latency provides information on the connectivity between each node and its assigned controller. Similar to δ_1 and δ_2, latency measures can be analyzed either by calculating the average across all latencies in the examined placement or by taking the maximum value for a worst case analysis. For a placement $\mathcal{P} \in 2^V$ and distance matrix D, the maximum node-to-controller latency $\pi^{\text{max latency}}$ can be defined according to Equation 2.4. In an analogous fashion, the average node-to-controller latency $\pi^{\text{avg latency}}$ is determined as per Equation 2.5.

$$\pi^{\text{max latency}}(\mathcal{P}) = \max_{v \in V} \min_{p \in \mathcal{P}} d_{v,p} \ , \tag{2.4}$$

$$\pi^{\text{avg latency}}(\mathcal{P}) = \frac{1}{|V|} \sum_{v \in V} \left(\min_{p \in \mathcal{P}} d_{v,p} \right). \tag{2.5}$$

When resilience with respect to controller outages is part of the analysis, additional calculations are necessary. Let $\mathcal{C} = 2^{\mathcal{P}} \setminus \{\emptyset\}$ denote all alternative placements that result from the failure of up to $k-1$ controllers. Then, $\pi_{\mathcal{C}}^{\text{max latency}}$ denotes the maximum node-to-controller latency which, in addition to the failure free case, also accounts for any failure scenario that spares at least one controller. Equation 2.6 shows the formal definition of $\pi_{\mathcal{C}}^{\text{max latency}}$ as well as the average-based $\pi_{\mathcal{C}}^{\text{avg latency}}$.

$$\begin{aligned} \pi_{\mathcal{C}}^{\text{max latency}}(\mathcal{P}) &= \max_{P \in \mathcal{C}} \max_{v \in V} \min_{p \in P} d_{v,p} \ , \\ \pi_{\mathcal{C}}^{\text{avg latency}}(\mathcal{P}) &= \frac{1}{|\mathcal{C}|} \sum_{P \in \mathcal{C}} \left(\frac{1}{|V|} \sum_{v \in V} \left(\min_{p \in P} d_{v,p} \right) \right). \end{aligned} \tag{2.6}$$

With the above definition of the node-to-controller latency, the definition of inter-controller latency follows by means of analogy. For a given placement $\mathcal{P}$ and any pair of controllers p_1, p_2 in this placement, the inter-controller latency $\pi^{\text{controller-latency}}(\mathcal{P})$ can be defined either with respect to the maximum or with respect to the average latency between p_1 and p_2. A formal representation of these relationships is presented in Equation 2.7.

$$\begin{aligned} \pi^{\text{max controller-latency}}(\mathcal{P}) &= \max_{p_1, p_2 \in \mathcal{P}} d_{p_1,p_2} \ , \\ \pi^{\text{avg controller-latency}}(\mathcal{P}) &= \frac{1}{\binom{|\mathcal{P}|}{2}} \sum_{p_1, p_2 \in \mathcal{P}} d_{p_1,p_2} \ . \end{aligned} \tag{2.7}$$

While metrics regarding latency strive for short communication paths, controller load balance considerations also need to be taken into account when a reliable network operation is desired. In order to comply with the problem definition and previous metrics, the imbalance metric is introduced rather than a balance metric so that the goal is to minimize the metric's value. For each placement $\mathcal{P}$ and controller p, the total number of nodes that are assigned to p when each node connects to its closest controller is defined as n_p. The imbalance metric $\pi^{\text{imbalance}}$ captures the difference in n_p for the two controllers with the lowest and highest amount of assigned nodes, respectively. Additionally, imbalance in the presence of failures can be quantified by analyzing imbalance in assignments that result from different failure scenarios. In these cases, n_p^s indicates the number of nodes assigned to controller p when failure scenario s occurs. Equation 2.8 defines both imbalance metrics. The indices $\emptyset$ and $\mathcal{X}$ denote the failure free case and the set of considered failure scenarios, respectively. Furthermore, the imbalance metrics can be normalized by division with $|V|$ as this is the maximum amount of nodes that can be assigned to a single controller in the worst case.

$$\begin{aligned} \pi_{\emptyset}^{\text{imbalance}}(\mathcal{P}) &= \max_{p \in \mathcal{P}} n_p^{\emptyset} - \min_{p \in \mathcal{P}} n_p^{\emptyset} \ , \\ \pi_{\mathcal{X}}^{\text{imbalance}}(\mathcal{P}) &= \max_{s \in \mathcal{X}} \left(\max_{p \in \mathcal{P}} n_p^s - \min_{p \in \mathcal{P}} n_p^s \right) . \end{aligned} \tag{2.8}$$

Finally, node and link failures can lead to a decomposition of the network graph, thus isolating nodes from all controllers. As discussed in Section 2.2.1, nodes that are not connected to a controller have very limited functionality and are therefore undesired. Hence, $\pi_{\mathcal{X}}^{\text{controller-less}}(\mathcal{P})$ computes the maximum amount of such nodes for any failure scenario specified in $\mathcal{X}$. The calculation of this metric is performed using a connectivity matrix E^s whose entries $e_{i,j}^s$ are equal to 0, if and only if node i can reach node j in failure scenario s and 1 otherwise. As controller outages that spare at least one controller do not affect the amount of

controller-less nodes, only node and link failures are considered for $\pi^{\text{controller-less}}$. These scenarios are summarized in the set $\mathcal{N}$, finally yielding Equation 2.9.

$$\pi_{\mathcal{N}}^{\text{controller-less}}(\mathcal{P}) = \max_{s \in \mathcal{N}} \sum_{v \in V} \min_{p \in \mathcal{P}} e_{v,p}^{s} \ . \tag{2.9}$$

2.3 Pareto Simulated Annealing

A key contribution of this work is the analysis of the trade-off between accuracy and cost with respect to time and memory resources when employing heuristic methods or performing an exhaustive evaluation to solve the controller placement problem. In order to incorporate the heuristic mechanisms into real world decision processes, guidelines for deciding whether to use an exhaustive evaluation or switch to a heuristic approach are derived. Furthermore, the influence of different parameters of heuristic algorithms is investigated in order to infer viable parameter values for different use cases and requirements. The heuristic algorithm proposed in this section is based on Pareto Simulated Annealing [41] and analyses are performed on numerous realistic network topologies from the Internet Topology Zoo [29].

After an overview of the Pareto Simulated Annealing algorithm, we describe the procedure that is used for our evaluations.

2.3.1 Design of the Multi-Objective Optimization Algorithm

Although an exhaustive evaluation of all possible placements for a given network topology and desired number of controllers guarantees finding all Pareto optima, its time and memory requirements rapidly increase with the size of the search space. This stems from the fact that the latter is proportional to $\binom{n}{k}$, the number of possible placements of size k in a network that consists of n nodes. Even for relatively small n, this number rises drastically when the number of controllers, k, approaches $\frac{n}{2}$, e.g., $\binom{34}{4} = 46.376$ while $\binom{34}{17} = 2.333.606.220$.

When performing just a single network planning task before deployment, an exhaustive evaluation is also justified for bigger instances even if it requires a high computational effort and a large amount of time. However, in the context of a dynamic and flexible network that needs to adapt to changes in the environment and usage patterns, time is a limiting factor.

In the context of finding the global optimum of a function that has a large domain, i.e., the optimization problem has a large search space, simulated annealing [67] is a popular heuristic approach. Simulated annealing is a Monte Carlo method and has two distinctive properties. First, during the exploration of the search space, moves to solutions that are worse than the current one are permitted in order to avoid getting stuck in a local optimum. This is achieved by incorporating a control parameter that is referred to as temperature, which determines the probability of accepting such moves. Second, the probability of moving to a worse solution gradually decreases with the number of iterations. Additionally, the acceptance probability depends on the difference between the objective values of the current and the proposed solution. The rationale behind this behavior is that accepting rather bad solutions at the beginning allows for a broader coverage of the search space while the lower acceptance probability at the end helps with convergence. However, simulated annealing does not support optimization problems with multiple objectives. Therefore, we utilize a multi-objective combinatorial optimization (MOCO) algorithm.

While there are many different options in the MOCO domain, we use Pareto simulated annealing (PSA), an algorithm that is inspired by simulated annealing. This decision is based on multiple criteria. First, PSA incorporates mechanisms that assert that the resulting output has a high degree of dispersion, i.e., that Pareto optima with respect to different objectives are found. This aspect of PSA is similar to the notion of *recall* in the domain of information retrieval. Recall is used to quantify the ratio between the amount of documents relevant to a given query that are returned by a search algorithm and the total amount of relevant documents. Second, PSA is an anytime algorithm and can thus provide a set of solutions at any time. Due to this property, it is not necessary to find

algorithm parameters that fit with particular time constraints. Instead, it is possible to just run the algorithm and stop it when results are needed. Finally, a heuristic approach based on simulated annealing has been successfully applied to a related single objective problem [33], demonstrating its feasibility in the controller placement context.

The PSA procedure that is used in this work is based on the algorithm presented in [41]. Algorithm 1 outlines the structure of the developed approach. The input consists of two parts. On the one hand, there is problem specific data like the topology graph G and the desired number of controllers k. On the other hand, there are parameters for the PSA mechanism. These include the number of placements that are evaluated during each iteration s, the number of iterations per temperature level m, as well as T_0 and ρ which control the *annealing schedule*, i.e., the initial temperature and the rate of temperature decrease. Initially, a set S of s random placements of size k is generated. For each combination of placement and objective, random weights Λ are assigned. Later, these weights help achieving the dispersion property discussed in the previous paragraph. During the whole procedure, the Pareto frontier of all visited placements M, as well as the corresponding placements are updated.

Starting with temperature T_0, the algorithm decreases the current temperature T by a factor of ρ after each m iterations until T falls below 1. Thus, following Equation 2.10, a total of $\left\lceil -\frac{\log T_0}{\log \rho} \right\rceil$ temperature levels are traversed.

$$\begin{aligned} T_0 \rho^i &\leq 1 \\ \rho^i &\leq \frac{1}{T_0} \\ i &\leq -\frac{\log T_0}{\log \rho} \end{aligned} \tag{2.10}$$

In each iteration, alternative placements that are "close" to those in S are generated. As usual in the Monte Carlo context, these placements are referred to as the *neighbors* of S and are stored in the variable Y. It is possible to define

Algorithm 1 Pareto simulated annealing.

input: $G = (V, E)$, k, s, m, T_0, ρ
$n = |V|$
$S = \text{generateRandomPlacements}(n, s, k)$
$\Lambda = \text{generateRandomWeights}(S)$
$M = \text{paretoFrontier}(\text{evaluatePlacements}(S))$
$T = T_0$
while $T > 1$ **do**
  $Y = \text{drawNeighbors}\left(S, n, \left\lceil \frac{kT}{2T_0} \right\rceil\right)$
  $\text{updateParetoFrontier}(M, Y)$
  $\Lambda = \text{updateWeights}(S)$
  $S :=$ accept $y \in Y$ with probability $P(S, Y, T, \Lambda)$
  if m iterations were performed at T **then**
    $T = T\rho$
  end if
end while
return M and corresponding placements

different neighbor relations between placements. One common way to define neighborhood is to allow replacement of at most one element, i.e., two placements are considered neighbors if they share all but one controller location. However, experiments with different definitions of neighborhood suggest using another method in the presented use case. Depending on the current temperature, placements are allowed to differ in up to $\left\lceil \frac{k}{2} \right\rceil$ elements to be considered neighbors. This ensures further dispersion at the beginning of the procedure and thus results in a broader coverage of the search space. At higher temperatures, the number of replaced controller locations decreases in order to favor convergence.

After generation, the proposed neighboring placements are integrated into M immediately. Then, the weight matrix Λ is recalculated according to [41]. The iteration ends with an update of S. In this step, an element of S is replaced with its corresponding neighbor from Y with probability $P(S, Y, T, \Lambda)$. While this probability equals 1 for placements that constitute an improvement over

their predecessor, it decreases for placements that are worse. The decrease in probability depends on the amount of deterioration as well as on the current temperature T and the weight matrix Λ. At last, the Pareto frontier M and the corresponding placements are returned.

Figure 2.2 illustrates the described mechanisms by providing different views of a single PSA run. The particular scenario consists of finding a good placement of size $k = 6$ for the Internet2 OS3E network topology which contains $n = 34$ nodes. The input parameters for the PSA algorithm are $m = 90, s = 10, T_0 = 50, \rho = 0.9$, resulting in $\left\lceil -\frac{\log 50}{\log 0.9} \right\rceil = 38$ temperature levels in which up to $90 \cdot 10 = 900$ distinct placements are evaluated. Thus, PSA explores only $\frac{38 \cdot 900}{\binom{34}{6}} = 2.5\%$ of the search space. Before launching the PSA algorithm, we perform an exhaustive evaluation in order to provide reference data. During the run, the current Pareto frontier of the PSA routine is periodically compared to the reference by means of the distance measures δ_1 and δ_2 (cf. Equations 2.2 and 2.3). Additionally, the development of the approximated Pareto frontier is visualized by taking snapshots of the two-dimensional Pareto frontier with respect to a subset of two metrics. These snapshots are presented at the top of Figure 2.2.

Each of the four plots displays the original Pareto frontier with respect to $\pi^{\text{imbalance}}$ on the x-axis and $\pi^{\text{max controller-latency}}$ on the y-axis alongside the set of Pareto optimal points explored by the PSA algorithm. Plot captions provide the relative progress of the PSA algorithm. While at the beginning, the heuristic solutions are clustered at a $\pi^{\text{imbalance}}$ value of around 0.25, the dispersion mechanism manages to explore a wide range of different solutions quickly. Thus, already at 10% of the algorithm's run time, many different combinations of metric values are available. With increasing progress, the margin between the two frontiers narrows and finer trade-offs become visible as more solutions are available. In some instances, the exact Pareto optimal placements are identified.

In contrast to the limited view on just two metrics that is displayed at the top, the bottom part of the figure shows how the values of δ_1 for the average distance and δ_2 for the maximum distance between the elements of the Pareto frontiers

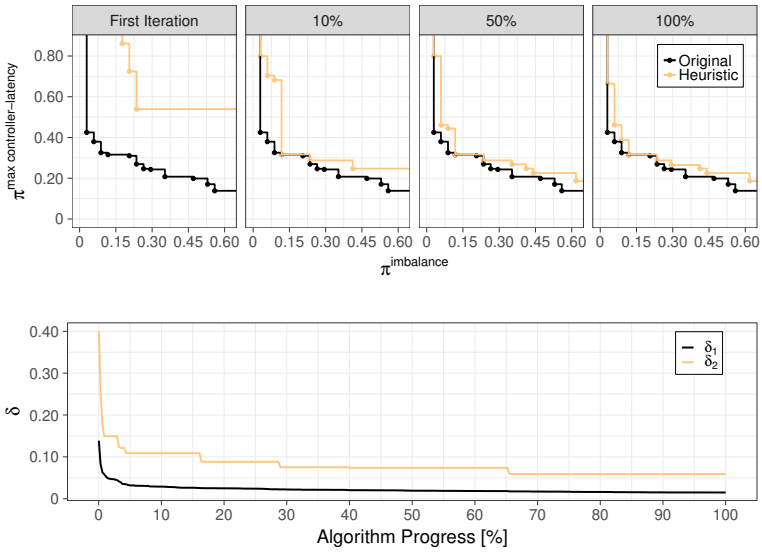


Figure 2.2: Development of the Pareto frontier estimate and corresponding δ_1 and δ_2 values during a single PSA run.

with respect to all considered metrics develop. The x-axis shows the percental algorithm progress while the logarithmically scaled y-axis denotes the δ values. Two phases can be identified in this plot. First, the segment from 0 to 5%, in which a rapid decrease of both metrics can be observed. As already mentioned, this is due to the quick exploration of the solution space when the temperature setting in the PSA routine is still high. The second phase is characterized by a slow but steady decrease of δ_1 and a stepwise decrease of δ_2. The jagged shape of the curve that displays the development of δ_2 can be explained by the fact that δ_2 considers only the maximum distance between any point of the reference set

and its closest counterpart in the approximation. Thus, not every improvement in the approximation is immediately reflected by δ_2. As soon as the solution that causes the maximum value is replaced with a better alternative, δ_2 represents the next worst placement. In contrast to this behavior, every improvement is covered by the average distance δ_1, leading to a smooth curve. At the end of the run, values of δ_1 and δ_2 reach 1.5% and 5.5%, respectively. This highlights the efficiency of PSA, i.e., even when evaluating only 2.5% of the entire search space, a mean error of 1.5% and a worst case error of 5.5% are achieved.

2.3.2 Performance Evaluation Methodology

In order to analyze the performance of the proposed PSA approach, various evaluation schemes are carried out. Special focus lies on quantifying the trade-off between the time saved when using the heuristic approach and the loss in terms of accuracy that is entailed. For this purpose, numerous network topologies from the Internet Topology Zoo are first evaluated in an exhaustive fashion. Results from these evaluations serve as reference for the accuracy assessment of the heuristic. By varying the input parameters of the PSA algorithm as well as by investigating its behavior in the context of different topologies and numbers of controllers, guidelines with respect to viable parameter combinations for real world scenarios are derived. These allow operators to express their specific needs in terms of accuracy requirements and time constraints.

The evaluation works as follows. Initially, combinations of network sizes and numbers of controllers are determined that can be handled by our exhaustive evaluation routine without exceeding the RAM of the machine in use. The motivation for choosing these scenarios is twofold. First, the described combinations pose the highest time and memory requirements while avoiding distortive performance degradation due to issues like swapping. Second, resulting computation times that are beyond multiple minutes exceed many practical constraints, especially when aiming for dynamic controller placement. Thus, these scenar-

ios represent use cases in which decision makers need to resort to heuristic approaches in order to comply with time and resource constraints.

Subsequently, parameters for the PSA algorithm are determined for different target specifications regarding accuracy and reliability. These specifications are represented by triples consisting of the reference metric $\delta \in \{\delta_1, \delta_2\}$ as defined in Section 2.2.2, a threshold τ for δ, and f_{min}, the desired fraction of instances in which δ is below τ. Given such a specification, parameters for the PSA procedure are calculated as follows. Starting with the lowest amount of iterations for the PSA routine, i.e., setting $m = s = 1$, PSA is applied to the problem instance 40 times. For each repetition, the difference between the Pareto frontier returned by the heuristic and the actual Pareto frontier obtained via exhaustive evaluation is computed with respect to the metric δ. If the percentage of instances in which δ does not exceed τ is beyond f_{min}, the current parameters of the PSA algorithm are returned. Otherwise, the search for parameters continues in a fashion similar to binary search, i.e., increasing m and s until the constraints are met and consequently decreasing them in order to obtain the smallest viable values.

In addition to finding the minimal parameter values for a given specification, performance statistics are recorded. On the one hand, the time consumption of the exhaustive evaluation is compared with that of PSA. For this, the Matlab function `timeit`[1] is applied to both computation procedures, yielding times t^{exh} and t^{PSA}, respectively. However, absolute times are specific to the used hardware and are thus difficult to interpret. This issue is tackled by combining both values into a ratio

$$t^{\text{rel}} = \frac{t^{\text{PSA}}}{t^{\text{exh}}} \tag{2.11}$$

which denotes the speed achieved by PSA relative to the exhaustive approach. On the other hand, the fraction of the search space that is explored by the PSA algorithm is recorded in order to investigate possible relationships between this

[1] http://www.mathworks.com/help/matlab/ref/timeit.html

fraction and the achieved performance. Especially when deciding upon parameters for network configurations for which no reference values are available, this can provide reasonable settings for PSA. As described in the previous section, PSA goes through $\left\lceil -\frac{\log T_0}{\log \rho} \right\rceil$ temperature levels. With m iterations per temperature level and s proposed neighbors in each iteration, up to $m \cdot s \cdot \left\lceil -\frac{\log T_0}{\log \rho} \right\rceil$ distinct elements of the search space are visited. This number is denoted as b^{PSA} and is referred to as *budget*. In an analogous fashion to the run time analysis, $b^{\text{exh}} = \binom{n}{k}$ refers to the budget requirement of the exhaustive evaluation and

$$b^{\text{rel}} = \frac{b^{\text{PSA}}}{b^{\text{exh}}} \tag{2.12}$$

describes the relative budget.

For all considered scenarios, the parameter k which indicates the desired number of controllers to be placed is assumed to be known beforehand. This assumption simplifies the problem by reducing the size of the search space from $\sum_{k=1}^{n} \binom{n}{k}$, where all possible numbers of controllers k are considered, to $\binom{n}{k}$.

All evaluations are carried out with the same set of objectives, namely node-to-controller latency, inter-controller latency, and controller load imbalance. This setup results in a total of five objectives as average and maximum values are optimized for the latency measures. Due to the fact that each considered placement is evaluated with respect to each of the chosen objectives, the number of objectives as well as their individual complexity affect the total run time of the exhaustive and the heuristic method. However, the dependency on the number of objectives is relative since both methods use the same subroutines for evaluating placements and are thus equally affected by changes in the number and complexity of objectives.

2.3.3 Investigation of Main Performance Factors and Resource Consumption

Based on the evaluation technique introduced in Section 2.3.2, computations with different specifications are performed. On the one hand, the general performance of the PSA heuristic is analyzed. On the other hand, trade-offs between the invested computational effort and the resulting accuracy are investigated. Beyond that, guidelines regarding the choice of algorithms and input parameters for the PSA algorithm are derived from the analysis of real world datasets. Finally, a comparison of the absolute time consumption of the exhaustive and the heuristic approach is presented. This demonstrates the feasibility of the heuristic approach in the presence of large problem instances and dynamic environments where recalculations of controller locations need to be performed on a regular basis.

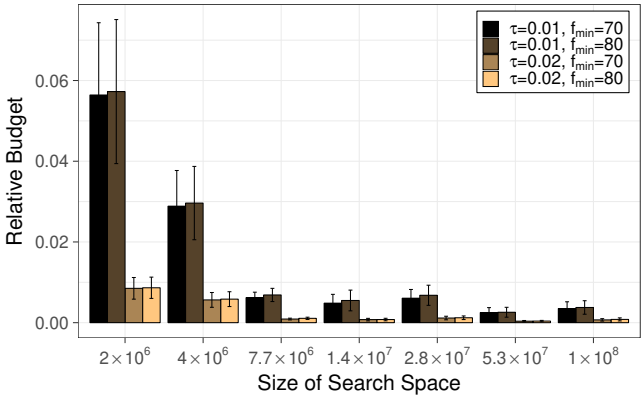


Figure 2.3: Size of the search space and the corresponding relative budget that is required in order to achieve various performance levels.

When facing an instance of the controller placement problem whose search space is too large to analyze in an exhaustive fashion while meeting time and

resource constraints, the heuristic approach presented in this work can be applied. While the input parameters of the PSA algorithm allow calculating an upper bound for the amount of analyzed placements, the absence of reference data in such cases prohibits making a statement about the resulting accuracy. In order to provide practical guidelines for the parameter choice, the relationship between performance constraints in terms of accuracy and the required relative budget b^{rel} (cf. Equation 2.12) is investigated. Therefore, an evaluation involving more than 60 graphs from the Internet Topology Zoo is performed. For each graph, four different numbers of controllers are tested and range from 5 to 15, depending on the graph's size. With graph sizes ranging from 25 to 50 nodes, these scenarios feature search spaces containing between one and 100 million different placements.

Figure 2.3 illustrates results from this evaluation. The data is grouped into logarithmically spaced bins according to the size of the search space in each scenario. Labels on the x-axis indicate the bins' thresholds, e.g., the first bin contains all scenarios for which $\binom{n}{k} < 2 \times 10^6$ holds. The y-axis displays b^{rel}, the relative budget required for achieving the performance goals set by the specification. While the bars' height denotes the mean value of b^{rel} in the performed evaluations, whiskers represent 95% confidence intervals and bar colors show different accuracy specifications. The investigated specifications use δ_1 as performance measure and feature combinations of accuracy thresholds $\tau \in \{0.01, 0.02\}$ and fractions $f_{\text{min}} \in \{70, 80\}$. There are three main observations. First, in the context of increasingly large search spaces, the required relative budget b^{rel} decreases. This behavior demonstrates the efficiency of the PSA mechanism. While the size of the search space, $\binom{n}{k}$, grows extremely fast with n and k, the algorithm's search strategy finds feasible solutions early on. Second, when facing rather small search spaces that contain around 4 million or less options, the b^{rel} values that are required in order to obtain good results express a high degree of variation as indicated by an increased width of confidence intervals in this range. For such sizes of the search space, an exhaustive evaluation is usually sufficiently fast, and thus is a viable alternative to the heuristic approach. Third,

for each bin, two groups of bars can be identified. The two groups correspond to the two values of τ, with b^{rel} values for $\tau = 0.01$ being higher than those for $\tau = 0.02$. This observation confirms the intuition that increased accuracy demands require a higher search budget. On average, the budget requirements for $\tau = 0.01$ are 5.6 times higher than for $\tau = 0.02$. The used values of f_{min} on the other hand do not have a significant influence on the relative budget b^{rel}. While the mean values of b^{rel} corresponding to $f_{\text{min}} = 70$ are strictly smaller than those corresponding to $f_{\text{min}} = 80$, their confidence intervals overlap, thus prohibiting a statistically significant statement.

In order to provide a size independent view of the evaluation data, Figure 2.4 presents the empirical cumulative distribution function (CDF) of b^{rel} values for different accuracy specifications. The x-axis is logarithmically scaled and shows the relative budget required for meeting specific performance constraints. On the y-axis, the fraction of cases in which the required budget is smaller than or equal to a particular value is displayed. As in the previous figure, different specifications are represented by different colors. In order to improve readability, the curves for specifications with $f_{\text{min}} = 70$ are omitted as they overlap with those that display values for $f_{\text{min}} = 80$. Again, a gap between the curves that correspond to different values of τ can be identified. Furthermore, the CDF representation allows for more generic recommendations with respect to the choice of parameters for the PSA algorithm. In particular, the graph shows that a relative budget of 1% is sufficient in over 90% of tested cases when the threshold for δ_1 equals 0.02. Increasing the accuracy demand by setting $\tau = 0.01$ raises the 90% quantile to a budget of 10%. However, even in the second case, a budget of 1% suffices in 80% of instances.

As described in Section 2.3.2, not only the budget of the PSA algorithm is measured but also its relative time demand in comparison to the exhaustive evaluation. Using the scenarios described at the beginning of this section, Figure 2.5 presents resulting t^{rel} values for different sizes of the search space. The size thresholds of logarithmically scaled bins are displayed on the x-axis, while the y-axis shows corresponding values of t^{rel}, the relative time consumption of

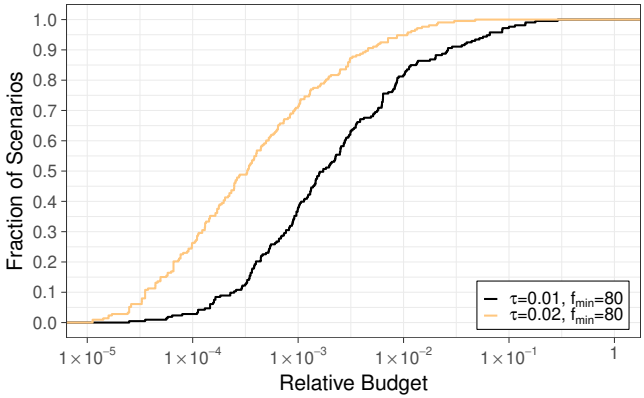


Figure 2.4: Distribution of the relative budget that is required in order to achieve various performance levels.

the PSA algorithm compared to the exhaustive evaluation (cf. Equation 2.11). Bars' heights indicate the average value per bin and whiskers mark the respective 95% confidence intervals. The different accuracy specifications for δ_1 are represented by the bars' colors. With increasing size of the search space, the relative time requirement of the heuristic approach decreases steadily, dropping below 10% for sizes beyond 4 million in the case of $\tau = 0.02$ and for sizes beyond 7.7 million in the case of $\tau = 0.01$, respectively. Additionally, the confidence intervals become narrower, indicating an increase in reliability. In contrast to the budget analysis, the t^{rel} values provide a straightforward assessment of the time-accuracy trade-off offered by the PSA heuristic. For example, PSA can deliver a set of placements more than 50 times faster than the default exhaustive evaluation when the scenario's search space contains 14 million placements or more and an average deviation of 2% with regards to accuracy is tolerable.

Consolidating the different size levels and calculating probabilities of observed t^{rel} values yields Figure 2.6, showing CDF curves for different specifica-

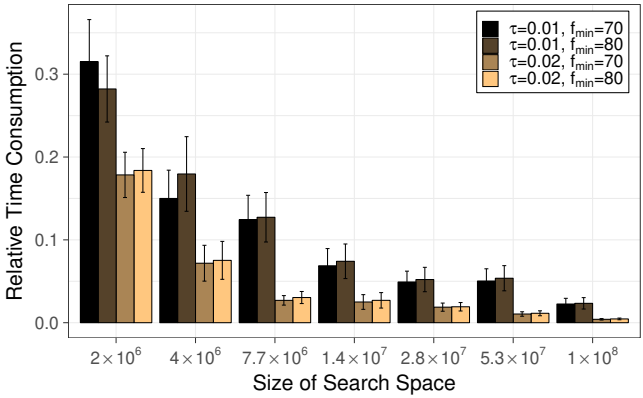


Figure 2.5: Size of the search space and the corresponding relative time that is required in order to achieve various performance levels.

tions. The latter are denoted by different colors, while x and y-axes provide t^{rel} values and corresponding cumulative probabilities, respectively. While the t^{rel} differences between consecutive 10% quantiles are rather small until 80%, they increase significantly beyond that threshold. This can be explained by the fact that the CDF also takes into account the evaluation data obtained from instances whose search space is rather small while PSA exerts its speed advantage in the context of huge search spaces. Nonetheless, the 80% quantiles for $\tau = 0.01$ and $\tau = 0.02$ with t^{rel} values of roughly 15% and 5% indicate a potential speedup by a factor larger than 6 and 20 when incorporating PSA rather than an exhaustive evaluation.

While considering the relative time consumption of the heuristic approach provides a hardware independent comparison, absolute times allow reasoning about the practical feasibility of the mechanism in the context of a particular hardware configuration and use case. Hence, Figure 2.7 presents the absolute time consumption of the exhaustive and the heuristic approach when applied to

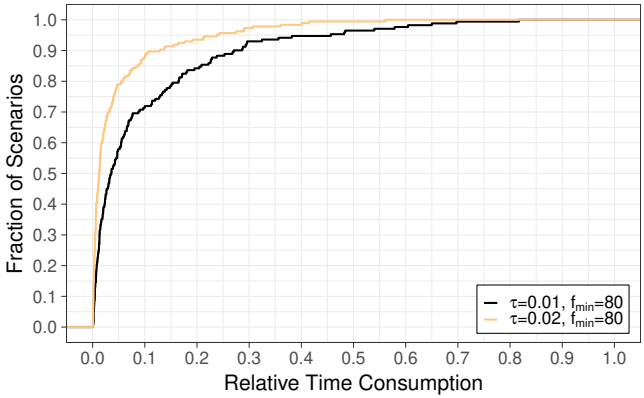


Figure 2.6: Distribution of the relative time that is required in order to achieve various performance levels.

the problem instances discussed in this section. In the case of the PSA algorithm, an accuracy demand of $\tau = 0.02$ is set. All measurements were performed on an Intel Core i7 4770 CPU at 3.40 GHz and 16 GB of RAM running Windows 7 and Matlab version R2014a. Again, the x-axis provides the thresholds of logarithmically scaled bins that indicate the size of the investigated search space. The y-axis displays the absolute time consumption of the two mechanisms and is logarithmically scaled as well. While differently colored bars correspond to different algorithms, bars' whiskers and heights indicate 95% confidence intervals and mean values, respectively.

In accordance with previous observations, an exhaustive evaluation can be completed nearly as fast as the heuristic approach in the context of small problem instances. On average, the exhaustive approach finished in less than a minute for problem instances with a search space size of up to four million elements. However, its run time increases dramatically for scenarios that feature a larger search space. For the largest instances, the exhaustive evaluation takes

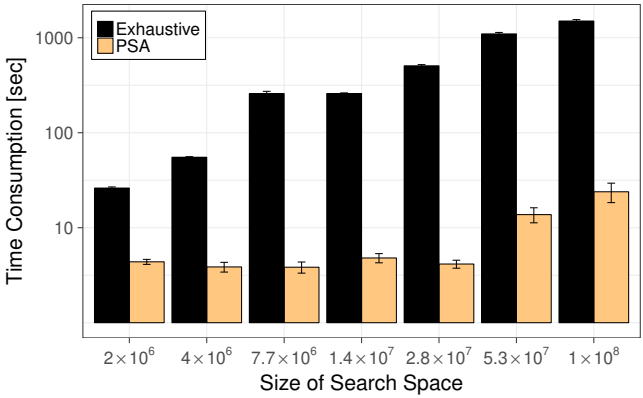


Figure 2.7: Size of the search space and the corresponding absolute run time requirements for an exhaustive evaluation and the proposed PSA heuristic with $\tau = 0.02$.

nearly half an hour while PSA delivers a solution within less than an average of 30 seconds. In general, using PSA is one to two orders of magnitude faster than performing an exhaustive evaluation.

Furthermore, an interesting effect can be observed in the case of the third and fourth run time values of the exhaustive approach, where almost identical run times are reported despite a seemingly twofold increase in problem size. This increase, however, is with respect to the bin margins and not the actual size of problem instances in the corresponding bin. Further inspection of the data shows that, for these particular bins, the average size of contained instances is almost identical. This stems from the fact that the sizes of problem instances in the third bin are close to its upper margin while the sizes of those in the fourth bin are close to its lower margin.

Additionally, the growth of the time consumption in the context of PSA is slower. While it remains relatively constant at around 5 seconds for the first five sizes, it rises to 15 and 25 seconds for the last two bins, respectively. Thus,

even large instances can be handled on a scale of seconds. This behavior highlights the practical feasibility of the PSA approach for dynamic environments where frequent changes require fast reaction times in order to adapt to the new situation quickly. Hence, the fast computation times allow for an automated approach to the dynamic controller placement problem. This can be achieved by summarizing an operator's preferences with respect to the objectives into a single score beforehand and choosing the highest-scored placement from the Pareto frontier returned by the PSA routine.

2.4 Pareto Capacitated k-Medoids

The second heuristic algorithm that is proposed in this chapter combines ideas from several graph theoretical algorithms in order to construct an approximation of the Pareto frontier with respect to two objective functions. These functions include the average node-to-controller latency and the imbalance regarding controller load. When considering only the node-to-controller latency, a clustering based approach is sufficient as it provides the location of controllers as well as an assignment from nodes to controllers which minimizes the latency between them. However, such algorithms do not take into account the amount of nodes that are assigned to each controller and thus might return arbitrarily bad results with respect to $\pi^{\text{imbalance}}$.

Therefore, the k-Medoids clustering algorithm [42] is enhanced with a capacity bound ρ which restricts the number of nodes that can be assigned to a single controller. By iteratively increasing the bound ρ, the maximum resulting imbalance can be influenced and thus, different trade-offs between the two objectives can be explored and summarized in a Pareto frontier. While the parameter ρ primarily affects the minuend of Equation 2.8, i.e., $\max_{p \in \mathcal{P}} n_p$, it also has an effect on the resulting imbalance metric. This is caused by the fact that the capacity bound implicitly limits the range of values $\pi^{\text{imbalance}}$ can attain. In contrast to other clustering algorithms such as k-Means [68], the centers returned by the

k-Medoids algorithm coincide with the nodes of the input graph. Therefore, it is chosen as the foundation for the proposed heuristic.

2.4.1 Iterative Pareto Frontier Generation

Algorithm 2 illustrates the capacitated k-Medoids approach. In addition to the distance matrix D, the network size n, and the number of controllers k, the algorithm receives the capacity bound ρ. First, the unmodified k-Medoids algorithm is applied to the problem instance (line 2). However, instead of assigning each node to its closest controller, a latency minimal balanced assignment of nodes to controllers is determined. Finding such an assignment corresponds to finding a cost minimal perfect matching in a bipartite graph which is constructed in lines 3 and 4 of the algorithm. The first partition, N, consists of n vertices representing the network's nodes, while controllers are placed in the second partition, which is referred to as F. In order to enforce the desired capacity limit of controllers, each controller is replicated $\lceil \frac{n}{k} \rceil + \rho$ times. Thus, a value of $\rho = 0$ corresponds to the tightest bound, i.e., the number of instances per controller prohibits configurations in which one controller manages significantly more nodes than another controller.

Next, a complete bipartite graph is created by adding an edge between each pair of nodes from the two partitions. Edge weights in this graph correspond to the entries in the distance matrix D. Consequently, a cost minimal perfect matching in the resulting bipartite graph yields an assignment of nodes to the controllers provided by the k-Medoids algorithm. As this assignment does not necessarily match the one intended by the k-Medoids algorithm, a shift of centers inside the partitions defined by the assignment might improve the latency in each partition without affecting imbalance. Hence, in each cluster, each node is considered as the new center. If this relocation improves the sum of latencies inside the cluster, it is accepted. Afterwards, the last two steps (i.e., calculating assignments given centers and calculating centers given clusters) are repeated

until convergence with respect to the latency is reached. This process corresponds to the while loop in lines 7 to 12 of Algorithm 2.

The final output consists of C, the cluster centers and A, the assignment of each node to its center. Due to the fact that the exhaustive evaluation assumes an assignment that is based on the minimization of shortest path latencies, the capacitated k-Medoids algorithm can produce placements that are not analyzed by the exhaustive approach. However, this does not have a negative impact on the results of the performance evaluation as the distance measure defined in Equation 2.1 defaults to zero when the estimate performs better than the reference solution.

Algorithm 2 Capacitated k-Medoids.

1: **input**: D, n, k, ρ
2: $C = \text{kMedoids}(D, n, k)$ $(= \{c_1, \ldots, c_k\})$
3: $N = \{1, \ldots, n\}$
4: $F = \left\{ c_1^1, c_1^2, \ldots, c_1^{\lceil \frac{n}{k} \rceil + \rho}, c_2^1, \ldots, c_k^{\lceil \frac{n}{k} \rceil + \rho} \right\}$
5: $(A, costs) = \text{match}(N, F, D)$
6: $(C', costs') = \text{recalculateCenters}(A)$
7: **while** $costs' < costs$ **do**
8: $\quad C = C'$
9: $\quad F = \left\{ c_1^1, \ldots, c_k^{\lceil \frac{n}{k} \rceil + \rho} \right\}$
10: $\quad (A, costs) = \text{match}(N, F, D)$
11: $\quad (C', costs') = \text{recalculateCenters}(A)$
12: **end while**
13: **return** (C, A)

For a single value of ρ, Algorithm 2 calculates a placement which minimizes the average node-to-controller latency while respecting the imbalance constraint introduced by ρ. However, the goal is to develop an algorithm that is capable of providing insights into the available alternatives and their associated trade-offs with respect to different objectives. Therefore, Algorithm 3 combines the results of multiple runs of the capacitated k-Medoids algorithm with differ-

ent values of ρ into a Pareto frontier of possible solutions. Additionally, parts of the k-Medoids heuristic rely on random numbers. Thus, the quality of results can be further improved by running the algorithm multiple times for each configuration. In total, there are two parameters that control the run time and performance of the Pareto capacitated k-Medoids (PCKM) algorithm. First, P, the set of values for the capacity bound ρ, and second, n_r, the number of times the capacitated k-Medoids routine is called for each $\rho \in \mathrm{P}$. The placements that are obtained during these $|\mathrm{P}| \cdot n_\mathrm{r}$ iterations are stored in the set S which is constructed in line 5 of the algorithm. Finally, the Pareto optimal placements are determined by evaluating the elements in S with respect to the two metrics under consideration and deriving their Pareto frontier.

Algorithm 3 Pareto capacitated k-Medoids.

1: **input**: D, n, k, P, n_r
2: $S = \emptyset$
3: **for all** $i \in \{1, \ldots, n_\mathrm{r}\}$ **do**
4: **for all** $\rho \in \mathrm{P}$ **do**
5: $S = S \cup \mathrm{capacitatedKMedoids}(D, n, k, \rho)$
6: **end for**
7: **end for**
8: $S = \{s \in S \mid s \in \mathrm{paretoFrontier}(\mathrm{evaluate}(S))\}$
9: **return** S

2.4.2 Evaluation Environment, Topologies, and Parameters

During the performance evaluation of the proposed Pareto capacitated k-Medoids (PCKM) algorithm, we focus on two main aspects. First, an analysis of the algorithm's run time for different sets of parameters and the distance between the resulting and the actual Pareto frontier provides the data that is required for quantifying the achieved trade-off between time and accuracy. Second, the specialized heuristic is compared to the generic Pareto simulated an-

nealing discussed in Section 2.3.1 as well as to a baseline algorithm that is based on randomly guessing placements.

This comparison also explores the consequences of specialization by calculating Pareto frontier distances with respect to the subset of objectives optimized by the Pareto capacitated k-Medoids approach as well as with respect to a larger set of objectives.

Both parts of the evaluation are carried out on a set of real world network topologies from the Internet Topology Zoo [29]. To provide reference values for the time and accuracy assessment, an exhaustive evaluation of placements is performed for each of the selected networks and desired number of controllers beforehand.

Similar to the evaluation methods that are outlined in Section 2.3.2, investigated problem instances are chosen based on two factors. First, an exhaustive evaluation of the search space corresponding to the problem instance should not exceed the available memory of the used machine. We motivate this by the fact that problem sizes beyond this threshold cause phenomena like page thrashing which in turn make the comparison unfair. Second, the computation times for an exhaustive evaluation of the chosen instances tend to be in the order of magnitude of several to tens of minutes. Such run times exceed the constraints that often appear in practice. Hence, these scenarios correspond to use cases which can be made tractable by employing heuristics. Overall, more than 60 graphs from the Internet Topology Zoo are evaluated. Their sizes range from 25 to 50 nodes and the experiments cover numbers of controllers between 5 and 15, resulting in state spaces that contain between one and 100 million distinct possible placements. All algorithms are implemented in Matlab and are executed on a server that is equipped with an Intel Xeon CPU at 2.10 GHz as well as 128 GB of memory and runs the 64-bit version of Ubuntu 13.10 and Matlab version R2014a.

In addition to the absolute run times of all investigated algorithms and parameter sets which are recorded via Matlab's `timeit` function, the relative time consumption of the heuristics are computed. While absolute run times provide insights into the time scales which can be achieved by using heuristics, state-

ments about the relative time consumption allow for a hardware-independent comparison of algorithms. The relative run time of a heuristic approach is defined as the ratio of the heuristic's computation time and the time required for an exhaustive evaluation.

As discussed in Section 2.4.1, the performance of the Pareto capacitated k-Medoids algorithm with respect to run time and accuracy can be controlled with two input parameters. On the one hand, n_r, the number of algorithm runs per configuration, controls the number of investigated placements and can be used to gather more reliable results. In this work, we use $n_\mathrm{r} = \{2, 4, \ldots, 10\}$. Increasing n_r beyond 10 did not show significant improvements regarding the algorithm's accuracy. It is worth noting that n_r is not the number of experiment repetitions used in the performance evaluation, but rather an input parameter to the PCKM algorithm that controls its run time. On the other hand, the parameter P controls the range of capacity bounds that are evaluated in order to construct the two dimensional Pareto frontier. Two options regarding the choice of P are analyzed. First, $\mathrm{P} = \{0, 1, \ldots, 9\}$, which covers 10 consecutive values for ρ and aims at thoroughly analyzing the trade-off between latency and imbalance. Second, $\mathrm{P} = \{0, 2, \ldots, 18\}$, which also contains 10 distinct values for ρ but targets covering a wider range of trade-offs.

In order to provide a context for PCKM's performance, it is compared with two additional algorithms. These include the Pareto simulated annealing (PSA) heuristic [1, 41] that is discussed in Section 2.3 as well as an algorithm that evaluates a given number of randomly generated placements and calculates the Pareto frontier of the resulting set of solutions. Due to the fact that the performance and the run time of the different mechanisms depends on their input parameters, the following methodology is used in order to achieve a fair comparison.

For a given set of input parameters of the PCKM approach, the average run time is determined. Afterwards, the input parameters of the alternative algorithms are tweaked in such a fashion that their run time equals that of PCKM. Hence, the comparison allows statements about the different algorithms' per-

formance when equipped with a particular time budget. In order to obtain statistically significant results, 10 repetitions are performed and evaluated for each combination of algorithm, network graph, and set of input parameters.

2.4.3 Performance Comparison and Key Influence Factors

This section presents results of the performance evaluation setup outlined in Section 2.4.2. On the one hand, the influence of the parameter choice on the performance of the Pareto capacitated k-Medoids algorithm is investigated. On the other hand, a comparison of the Pareto capacitated k-Medoids (PCKM) mechanism with the Pareto simulated annealing heuristic as well as a baseline approach based on random guessing is presented.

Figure 2.8 illustrates the results that are obtained when using the proposed evaluation scheme. It shows the cumulative distribution of the Pareto frontier distance δ with respect to the two objectives that are optimized by the algorithm, i.e., the average node-to-controller latency and the controller imbalance. The x-axis shows increasing values of δ and the y-axis represents the fraction of scenarios in which the PCKM approach achieves a distance of at most δ. While the capacity range P is represented by the line style, with the fine grained capacity range $\{0, 1, \ldots, 9\}$ as solid lines and the coarse grained capacity range $\{0, 2, \ldots, 18\}$ as dashed lines, the number of repetitions n_r is represented by the lines' color and takes on values 2, 6, and 10.

There are three main observations.

a) An increase in n_r leads to an increase in accuracy. Increasing n_r not only affects the total number of placements analyzed by the algorithm, but also adds diversity to the solution as the k-Medoids subroutine starts its optimization from a different set of centers in each iteration.

b) The accuracy gains between consecutive n_r values decrease for higher n_r. For example, the 90% quantiles of the distance in case of $\mathrm{P} = \{0, 1, \ldots, 9\}$ take on values of roughly 5%, 3.4%, and 3.2% for

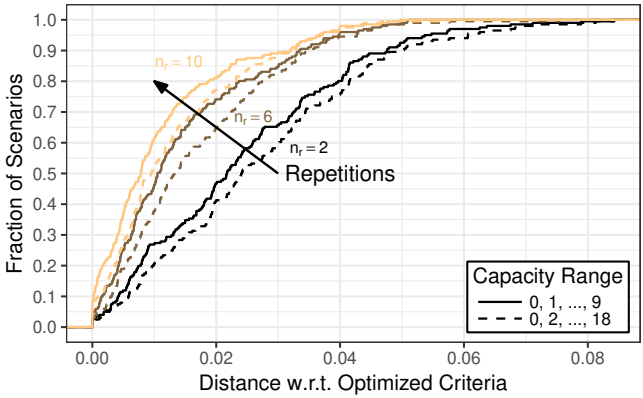


Figure 2.8: CDF of the algorithm's error with respect to average latency and imbalance for different numbers of repetitions n_r and capacity ranges P.

$n_\mathrm{r} = 2, 6, 10$, respectively. This phenomenon hints at a converging behavior, i.e., the accuracy doesn't improve significantly beyond a particular value of n_r.

c) The fine grained capacity range yields a higher accuracy than its coarse grained counterpart for all the scenarios that are covered in this work. This behavior stems from the fact that the reference Pareto frontier usually contains many distinct imbalance values in the lower range while the coverage of higher values is rather sparse. Hence, $\mathrm{P} = \{0, 1, \ldots, 9\}$ is used for the remainder of this work.

In order to determine input parameters for the alternative algorithms that are used in the performance comparison, the absolute run times of the PCKM algorithm are measured for the different configurations. Figure 2.9 presents the distributions of these run times. Differently colored curves represent different values of the number of repetitions n_r. When the number of repetitions n_r is increased from 2 to 10 in steps of 2, the median run time increases from roughly

0.5 seconds to 2.7 seconds in almost equidistant steps. This behavior is in line with the fact that each repetition is performed independently of the other, and thus n_r affects the total run time in a linear fashion. However, the interquantile range also increases with n_r in the analyzed scenarios. This can be explained by the varying run times of consecutive repetitions of the k-Medoids algorithm. For each repetition, the number of iterations spent inside the k-Medoids routine can differ. Increasing n_r implies a wider interval of possible values for the sum of these iterations and thus a higher variance is observed.

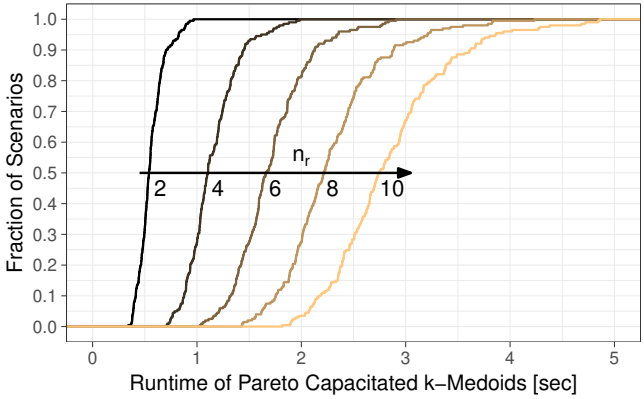


Figure 2.9: CDF of the absolute time consumption of the Pareto capacitated k-Medoids algorithm for different numbers of repetitions n_r.

The proposed algorithm is compared with three reference algorithms by means of Pareto frontier distances. The results are depicted in Figures 2.10 and 2.11. In addition to PCKM, a two dimensional version of the Pareto simulated annealing algorithm (PSA2D) and an algorithm based on random guessing (RND) are analyzed.

Before aggregated results are presented in Figure 2.11, Figure 2.10 illustrates the different algorithms' behavior by displaying the Pareto frontiers returned

during a single run of each algorithm and comparing them to the reference Pareto frontier obtained with the brute force approach. For this example, the Sinet topology is chosen. In this network of 47 nodes, 5 controllers are placed within a time budget of one second. The Pareto frontiers are determined with respect to the two objectives that are optimized, i.e., the average node-to-controller latency and the controller load imbalance. X- and y-positions of individual points show the values of the objective functions that are achieved by the corresponding placements. Additionally, the imbalance metric is normalized with the number of nodes in the topology. As a visual aid, each set of Pareto optimal points is connected with line segments that are not part of the Pareto frontier. Different algorithms are represented with different colors and marker shapes.

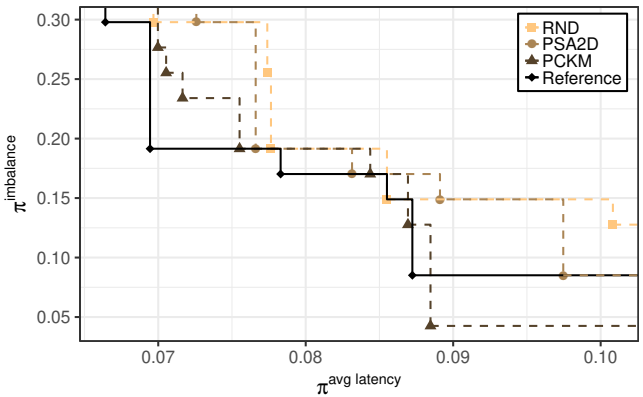


Figure 2.10: Exemplary Pareto frontiers that are obtained with the algorithms that are discussed in this work. Settings: Sinet topology (47 nodes), 5 controllers, and a time budget of 1 second.

There are four main observations.

a) The Pareto frontier that is returned by the PCKM algorithm has the highest cardinality of all discussed approaches. This corresponds to a thorough coverage of the different possible trade-offs between the optimized objectives and is achieved by PCKM's iterative approach with respect to the capacity limits. By imposing different imbalance constraints, the resulting latency is varied and the search space is explored in a systematic fashion.

b) The PCKM algorithm discovers a solution that is not captured by the reference Pareto frontier. In contrast to the brute force approach, PCKM is not restricted to assigning nodes to controllers based on latency and thus explores a larger search space than the other algorithms.

c) The PSA2D algorithm is also characterized by the diversity of solutions, i.e., the trade-offs between objectives are reflected in the resulting Pareto set. However, the available time budget is not sufficient to achieve convergence, so that some regions contain only few solutions or feature outliers. These phenomena can be observed in the sparse coverage of the 0.2 to 0.3 range of the imbalance metric, where PSA2D yields only two solutions, as well as the rightmost outlier with respect to the latency objective.

d) Finally, the placements found by the RND approach are scattered throughout the objective space. While the mechanism finds one Pareto optimal solution by chance, its result set also features an extreme outlier. This demonstrates the high variance and thus low reliability of the RND algorithm.

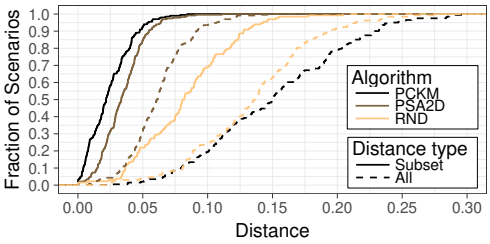
While the preceding discussion is focused on one individual run, Figure 2.11 presents the algorithms' performance in an aggregated fashion. Furthermore, not only the Pareto frontier distance regarding the two objectives optimized by PCKM and PSA2D is presented, but also the distance from a five dimensional Pareto frontier is taken into account. The extended set of criteria contains the

maximum node-to-controller latency as well as the average and maximum latency among controllers. Such an analysis provides insights into the strategies that are employed by the different algorithms in order to explore the search space and find feasible solutions.

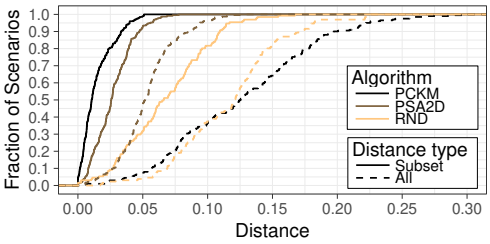
While the different algorithms are represented by differently colored curves, the line style indicates the kind of distance measure under investigation. All three subfigures of Figure 2.11 display cumulative distributions of these distances, each resulting from different algorithm parameters obtained according to Section 2.4.2.

A comparison between the algorithms' performance with respect to the Pareto frontier distance regarding the average node-to-controller latency and imbalance shows that the PCKM algorithm consistently outperforms the PSA2D heuristic for all three time constraints that are used in Figures 2.11a, 2.11b, and 2.11c. This demonstrates the gain achieved by utilizing a specialized heuristic over a generic method given the same time budget on identical hardware. Moreover, the distance in case of the RND algorithm is significantly higher than those of PCKM and PSA2D as the RND approach does not systematically explore the solution space but rather evaluates random placements. Although the small absolute differences between achieved Pareto frontier distances for PCKM and PSA2D might suggest that utilizing PCKM over PSA2D provides only a slight improvement, the relative increase is significant. For example, when inspecting the 90% quantiles of the distributions in Figure 2.11b, PCKM achieves a distance of 3% with respect to the subset of optimized criteria while PSA2D produces an error of 5%. Hence, choosing PCKM corresponds to a relative gain of $1 - \frac{0.03}{0.05} = 40\%$ in terms of accuracy.

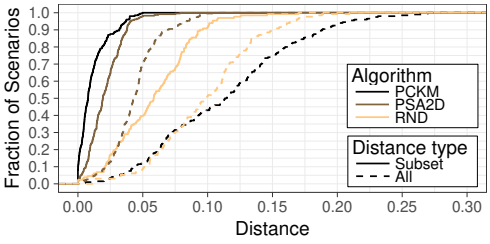
When extending the distance measure to take into account additional objectives that represent the maximum and average inter-controller latency and maximum node-to-controller latency, two phenomena are observed. First, the absolute distance values increase for all algorithms. Such a behavior is expected as none of the algorithms explicitly optimizes for the additional objectives and the available time budget is not increased to accommodate for the increased com-



(a) One second, corresponding to $n_r = 2$.



(b) Two seconds, corresponding to $n_r = 6$.



(c) Four seconds, corresponding to $n_r = 10$.

Figure 2.11: Comparison of algorithms' performance with respect to different distance types given different time constraints.

plexity, either. Furthermore, the relative order of algorithms with respect to the achieved distance changes. In the context of all three time settings, the PSA2D mechanism provides the highest accuracy, i.e., the lowest distance values. The reasons for PSA2D's advantage in this domain are twofold. On the one hand, it generally explores a larger number of placements than PCKM which results in a higher chance of coming across solutions which are viable with respect to the newly added optimization goals. On the other hand, the PSA2D approach follows a more systematic path through the solution space than RND, which lowers the chance of visiting the same placement multiple times. Due to having the lowest amount of evaluated placements, PCKM falls short of RND when the extended distance measure is of interest.

2.4.4 Integration into the POCO Framework

The two algorithms that are proposed in this chapter are also integrated into the POCO framework. POCO does not only evaluate the possible placements for a given topology and number of controllers, but is also capable of providing a visualization of the corresponding solution space in a graphical user interface. Furthermore, the parameters of the heuristic algorithms can be changed from their default values by means of forms that can be accessed via the POCO GUI.

Both the POCO GUI as well as its core are implemented in Matlab and are available as open source software [43, 69]. Users can choose between metrics for investigation and are presented with a plot that displays the performance of all possible placements with regard to the chosen metrics. Each placement is represented as a point in the plot whose x and y-axes denote its metric value. Additionally, points located on the Pareto frontier with respect to these metrics are highlighted and connected with line segments.

Figure 2.12 shows an example session of the POCO GUI featuring the Internet2 OS3E topology with $k = 4$ controllers. The top half shows the topology including the currently selected placement by highlighting controllers with double circles. The color of each node, based on a traffic light color scheme, denotes the

latency to the node's controller, which is selected based on the lowest distance. Additionally, the implications of a failed node (#16) and a failed controller (#4) for this particular placement are illustrated. Further information can be added to the representation of the placement. Check boxes on the right hand side allow enhancing the graphic with information regarding the number of controller-less nodes or different latency measures as well as applying a vertex coloring to visualize node-to-controller assignments for imbalance analyses.

In the bottom part of the interface, the Pareto frontier of all possible placements with respect to the maximum node-to-controller (x-axis) and controller to controller latency (y-axis) is displayed. By clicking on any point in the Pareto plot, the detailed visualization at the top is updated according to the selected placement that corresponds to the clicked point. This allows the decision maker to interactively explore the search space according to her/his preferences and current use case.

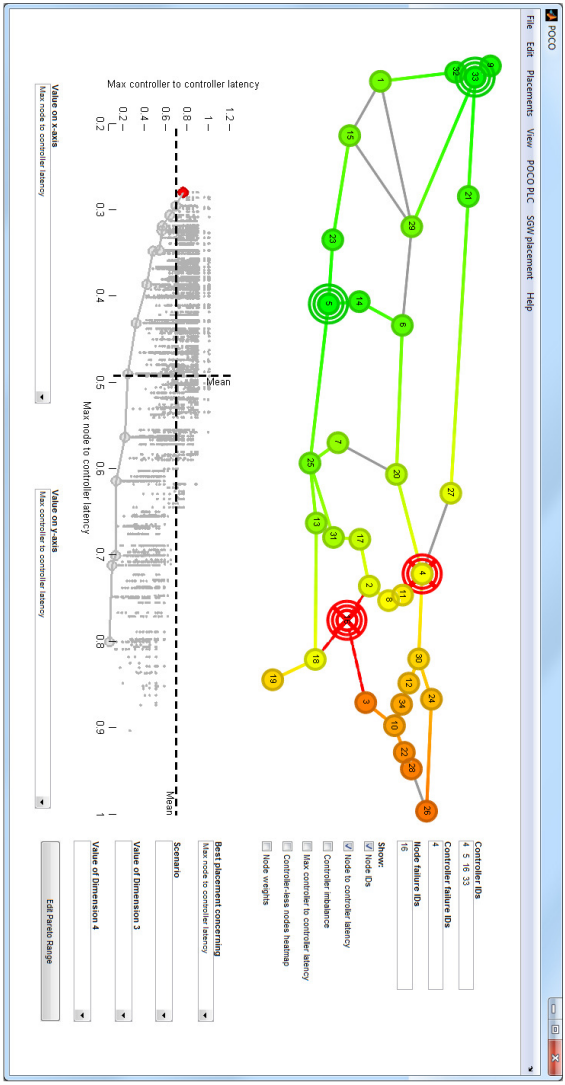


Figure 2.12: The POCO GUI displays the current placement with the color coded switch to controller latency at the top and the whole solution space of all possible placements at the bottom.

Features

In order to allow an even more in-depth analysis of the set of placements, POCO's GUI [43] is extended with options to display up to four dimensions of the solution space. This is achieved by applying different transformations to the size and color of the points presented at the bottom of Figure 2.12. First, the set of Pareto optimal placements with regard to the four chosen metrics are calculated. Then, minimum and maximum values for the two additional metrics are computed. Using the traffic light color scheme as in Figures 2.1 and 2.12, the points' colors and sizes are adapted so that they reflect the placements' performance with regard to the additional metrics. Figure 2.13 illustrates this new, four dimensional visualization of the scenario that is displayed in the previous figure.

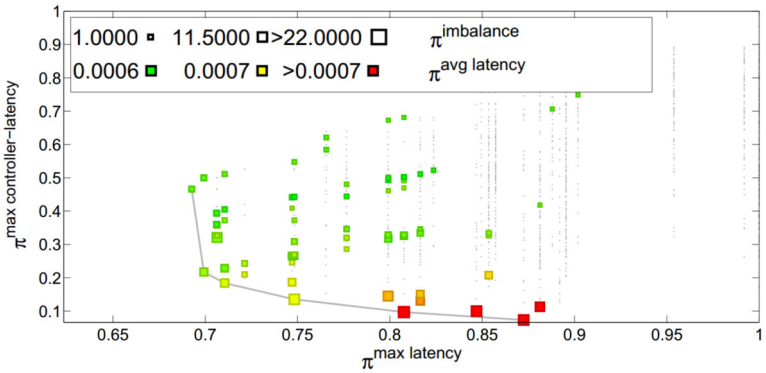


Figure 2.13: Visualization of the 4-dimensional Pareto frontier in POCO.

While the maximum node-to-controller latency $\pi^{\text{max latency}}$ and maximum inter-controller latency $\pi^{\text{max controller-latency}}$ remain on the x and y-axes, the imbalance with respect to the number of assigned nodes per controller $\pi^{\text{imbalance}}$ and the average node-to-controller latency $\pi^{\text{avg latency}}$ are included via the size and color of the points. This perspective shows further trade-offs and inter-

dependencies between metrics, and thus allows the decision maker to make a well-founded choice reflecting her/his preferences for a particular use case.

Through the new functionality "Edit Pareto Range", a multi-step filtering of the results can be performed. After investigating the Pareto frontier for a certain combination of metrics and the performance of all placements, the user can filter out placements that exceed a certain threshold. These placements are then removed from the stored result set of placements so that switching to a view that shows a different combination of metrics will not display these placements again. In this next step, the user can further reduce the number of Pareto optimal results, in order to finally choose from only a little number of Pareto optimal results that represents the best trade-off given the objectives that are important for the particular use case.

Using this improved mechanism, it is possible for a network engineer to set thresholds for metrics after investigating the whole solution space instead of having to define upper bounds before starting the computation.

2.5 Lessons Learned

Designing the control plane of an SDN-based architecture poses several challenges to network operators. Even when the required number of entities in the control plane is known beforehand, their locations have a significant impact on numerous performance aspects of the network. This results in the multi-objective optimization task that is known as the controller placement problem. Its solution contains sets of controller locations that represent possible trade-offs between different objectives like control plane communication delays or the balanced load distribution among controller instances. While an exhaustive approach to this placement problem is practically feasible for small and medium-sized problem instances, the time and resource demands of large problem instances call for alternative mechanisms. Such mechanisms usually involve heuristics that sacrifice accuracy or optimality guarantees for significantly faster run times.

This chapter works towards quantifying the trade-offs that result from employing various heuristics as well as providing guidelines with respect to the choice of algorithms and parameters for different use cases. To this end, we design and evaluate two multi-objective optimization heuristics for the controller placement problem.

On the one hand, these heuristics allow analyzing problem instances that are too large to evaluate in an exhaustive fashion. On the other hand, in the presence of time constraints in highly dynamic environments, they allow for a trade-off between run time and accuracy. This trade-off is analyzed in detail via an evaluation featuring over 60 real world topologies. Depending on the operator's requirements with regards to accuracy, speedups beyond a factor of 20 are possible when compared to an exhaustive evaluation. For example, the proposed Pareto simulated annealing (PSA) heuristic can reduce the run time for optimizing the placement of 7 controllers in a network with 50 nodes from nearly half an hour to less than 30 seconds if an error of up to 2% is acceptable. Furthermore, large problem instances that cannot be solved due to their enormous memory requirements can be evaluated with the proposed heuristics.

Beyond that, the specialized Pareto capacitated k-Medoids heuristic optimizes a particular subset of objective functions in order to further improve the performance within its narrower scope of applicability. This is achieved by leveraging characteristic properties of the particular objectives that are optimized. For the scenarios that have been investigated in this work, this specialization results in relative performance improvements of up to 40% when compared to the generic PSA heuristic. Hence, operators also need to keep in mind available alternatives in terms of algorithms and the characteristics of their particular problem instances.

3 Automated Decision Making based on Pareto Frontiers

In current and emerging network architectures, placement problems regarding SDN controllers and Virtualized Network Functions (VNFs) play an important role. As discussed in the previous chapter, the location of the distributed controller instances in SDN-based networks can affect performance aspects like the latency between switches and controllers, resilience, or the synchronization delay between controllers. Similarly, optimizing the locations of VNF instances in the network functions virtualization (NFV) context can lead to a better resource utilization as well as QoS. Finally, cloud service placement strategies aim towards increasing end-user QoE while minimizing costs.

Conceptually, the abovementioned problems have several similarities. Firstly, heuristics are used in most real world scenarios since the combination of the underlying NP hard facility location problem and the huge solution space prohibits exact solutions. Secondly, there are usually multiple, possibly competing, objective functions that need to be optimized simultaneously. Hence, the output of the aforementioned heuristics does not consist of a single distinct optimum but a Pareto frontier that represents different trade-offs between the objectives. Therefore, the solutions that are returned can not always be directly compared with each other due to different domains and units of the objectives. The upper portion of Figure 3.1 demonstrates an exemplary Pareto frontier that is returned by such a multi-objective optimization algorithm.

Especially in the context of systems that adapt to changing network conditions in an automated and dynamic fashion, however, algorithms need to choose

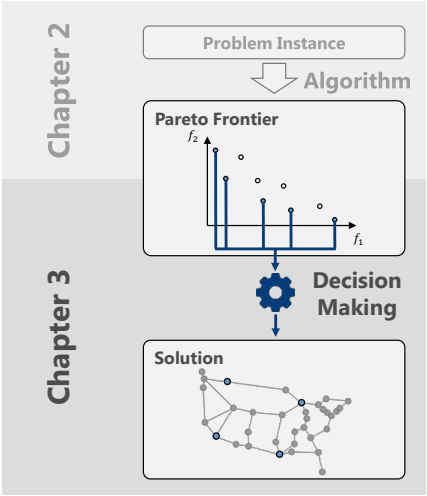


Figure 3.1: Overview of the optimization and decision making process.

one distinct solution. This behavior is highlighted in the bottom part of the figure where the decision making mechanism chooses a particular placement from the set of Pareto optima. In this chapter, we design, evaluate, and compare mechanisms that enable such an automated decision making. To this end, we follow a three-step-approach. First, four methods for determining the relative importance of different objectives are selected and compared with each other. In contrast to approaches that determine such weights a priori, the methods used in this work take into account characteristics of the solutions that are returned by the multi-objective optimization algorithm. Hence, they can adapt to the characteristic behavior of objectives in the specific context of a particular problem instance. Second, four mechanisms for aggregating the performance of a multi-dimensional solution into a single score are selected. Finally, the rankings of solutions that result from different combinations of weighting and aggregation techniques are characterized. On the one hand, analyzing solutions

that consistently achieve high ranks according to many approaches might lead to more efficient methods for identifying viable placements. On the other hand, the comparison can help to derive guidelines for choosing the appropriate ranking mechanism for a particular problem.

In a case study featuring the SDN controller placement problem, we demonstrate the particular behavior of the investigated mechanisms for an exemplary network and three objective functions, i.e., three optimization goals. Furthermore, we provide an extensive analysis of 58 real-world network topologies from the Internet Topology Zoo [29], a total of five objective functions, and varying numbers of controllers that are placed. By aggregating the results of these analyses, we can compare the different weighting and ranking methods in terms of aspects like agreement and consistency across problem instances.

The content of this chapter is based on [11, 15] and its remainder is structured as follows. After an overview of related work in Section 3.1, the data set is presented alongside the resulting problem instances in Section 3.2. The selected methods for assessing the weight of each objective dimension are introduced and compared in Section 3.3. These methods are then used as input for aggregation algorithms that assign a single score to each placement. In Section 3.4, the four selected aggregation algorithms are discussed and compared with respect to the rankings of placements they produce. Finally, Section 3.5 concludes the chapter with an overview of lessons learned.

3.1 Background and Related Work

Placement problems constitute an important challenge in the context of management and orchestration of softwarized networks. In particular, they include the SDN controller placement problem [37], the NFV function chain placement problem [70], as well as virtual machine (VM) placement in the area of cloud resource management [71].

In the context of SDN controller placement, widely used methodologies include heuristics whose goal functions are based on weighted sums of objec-

tives [33], integer linear programs (ILPs) [33], as well as multi-objective heuristics [1, 8]. A similar set of techniques is also applied in the domain of NFV function placement. While the authors of [72] investigate an approach that is based on a weighted sum, optimal function placements are identified by means of an ILP in [73]. Furthermore, the authors of [74] combine ILP-based approaches with a Pareto analysis in order to investigate trade-offs between different objectives.

Correspondingly, strategies for VM placement in cloud data centers can be divided into two groups. The first group consists of single objective optimization approaches that focus on minimizing the energy consumption [75], minimizing the costs [76], optimizing the resource utilization [77], or maintaining QoS guarantees [78]. In the second group, multi-objective mechanisms that use heuristics like multi capacity bin packing [75] are used to achieve a viable balance between objectives that include the power consumption, delays, and costs simultaneously [76, 79, 80].

However, in the case of all abovementioned multi-objective approaches, no automated decisions can be made from the Pareto frontiers. In this chapter, we propose an approach to address this automated decision making problem which is based on transforming a Pareto frontier into a ranked list of alternatives. To compare the rankings when the underlying order of alternatives is unknown, we rely on correlation coefficients and techniques that are based on probabilistic ranking models.

In [81], the rank correlation between pairs of rankings is calculated using either Spearman's ρ or Kendall's τ. The authors of [82] propose a measure of agreement between rankings based on removal of disputable elements. A basic model for order statistics is developed by Thurstone [83], and Luce [84] constructs an equivalent model based on choice probabilities. Mallow [85] presents simplified and analytically tractable models that are induced by paired comparison. In [86], concordance between different judges, i.e., rankings, is investigated. The corresponding mechanism is based on Mallow's model to detect outlier rankings. Cohen [87] proposes comparing the distribution of ranks by box plots and derive a degree of discordance based on the inter-quartile range. The good-

ness of fit of simple ranking models is investigated in [88], and metric-based ranking models are discussed in [89]. A classification of probabilistic ranking models can be found in [90].

3.2 Characteristics of the Network Topologies Under Study

In order to investigate the practical feasibility of the different weighting and ranking methods that are discussed in this chapter, realistic input data is required. To this end, we use 58 different network graphs from the Internet Topology Zoo [29] and use the freely available POCO tool [39] to exhaustively evaluate all possible controller placements with respect to a total of up to five objective functions. While results are consistent among different networks, some characteristics depend on statistics like the number of nodes and the diameter of the graph. On the one hand, we present detailed results and statistics for the Internet2 OS3E topology which is chosen as an exemplary representative. This allows for accurate insights into the functionality of the different weighting and scoring mechanisms. On the other hand, we provide aggregated results and statistics regarding the whole data set in order to identify topology-independent relationships between the various mechanisms.

3.2.1 Internet2 OS3E

Table 3.1 provides an overview of the Internet2 graph as well as the resulting problem instance. In order to keep the solution space small enough to visually illustrate the effects and behavior of the presented methods, only four controllers are placed in the network and the number of objectives is limited to three. Although this results in a total of $46,376$ distinct placements, only ten of those are Pareto optimal and thus relevant during the decision making process. In the context of larger search spaces, e.g., when placing more controller instances or dealing with networks that have more nodes, an exhaustive evaluation of all

possible placements might not be feasible due to time and resource constraints. For such cases, a trade-off between accuracy and run time can be achieved by employing heuristic approaches that can approximate the Pareto frontier [1].

Table 3.1: Information regarding the Internet2 OS3E topology and the corresponding problem instance that is used in the case study.

Property	**Value**
Number of nodes	34
Number of placed controllers	4
Number of distinct placements	46, 376
Number of Pareto optimal placements	10
Objective functions	Mean node-to-controller latency $\pi^{\text{avg latency}}$ Maximum node-to-controller latency $\pi^{\text{max latency}}$ Imbalance between controller instances $\pi^{\text{imbalance}}$

As mentioned in the previous paragraph, three different objective functions are taken into account when assessing the performance of each placement. These include two latency-related measures, namely, the mean and maximum latency between controllers and switches. We use the longitude and latitude information that is provided for each node to calculate the Euclidean distance between nodes and approximate the latency of each link. For multi-hop paths, the latency is defined as the sum of latencies of all involved links. Furthermore, the load imbalance between controller instances is defined as the difference between the number of switches that are assigned to the instance with the highest and lowest amount of switches, respectively.

Several statistical properties of these objective functions are presented in Table 3.2. Additionally, Figure 3.2 displays the cumulative distribution function of objective values that are attained across all placements. Due to the fact that the

Table 3.2: Various statistics of the objective functions that are used in the case study.

Objective	Number of distinct values	Mean	Variance
$\pi^{\text{avg latency}}$	45,311	0.195	0.001
$\pi^{\text{max latency}}$	244	0.491	0.013
$\pi^{\text{imbalance}}$	29	0.305	0.019

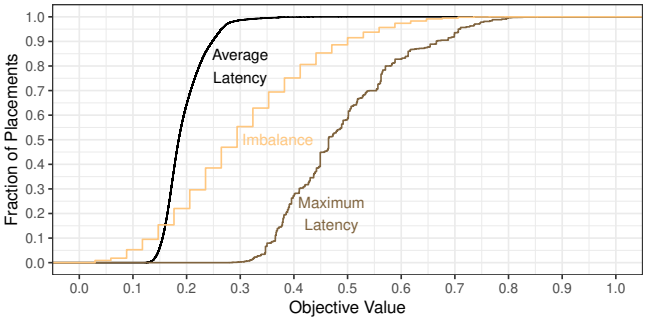


Figure 3.2: Empirical CDFs of objective values that are attained in the example scenario.

latency measures are continuous, they yield significantly more distinct values, resulting in smooth CDF curves. In contrast, the imbalance is always an integer value which is constrained by the number of nodes in the topology. Hence, individual steps are visible in the plot. Since the average node-to-controller latency is calculated from 34 individual latencies, outliers are smoothed out during the calculation of the mean and the resulting variance of attained values is relatively low. The values of the maximum latency objective have a higher variance and fewer distinct values since the maximum does not necessarily change between similar placements that share multiple controller locations. In summary, these characteristics suggest that in order to capture the sensitivity of objec-

tives towards changing placements, the relative importance that is assigned to an objective should correlate with the variance of the attained objective values.

3.2.2 Internet Topology Zoo

For the evaluation of networks from the Internet Topology Zoo, we use networks whose size n ranges between 25 and 50 nodes. This ensures that the exhaustive evaluation of all possible placements with POCO can be performed within a reasonable time frame. Using each of the resulting 58 topologies, we calculate placements of $k \in \{3, 4, 5\}$ controllers and evaluate them with respect to a total of five objectives, resulting in a total of 174 problem instances. In addition to the abovementioned imbalance and latency measures, the average and maximum latency between each pair of controller instances is also taken into account. These objectives are referred to as $\pi^{\text{avg inter-latency}}$ and $\pi^{\text{max inter-latency}}$, respectively. In the following, we present various aggregated statistics of the set of problem instances that are discussed in this chapter.

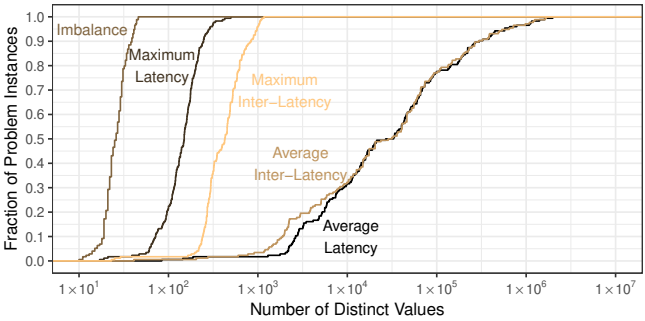


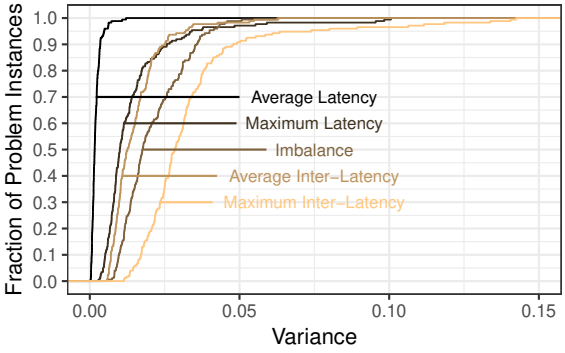
Figure 3.3: Empirical CDFs of the number of distinct function values per problem instance for each objective function.

Similarly to Table 3.2, Figure 3.3 presents the distribution of the number of distinct values that are attained by each objective function per problem instance.

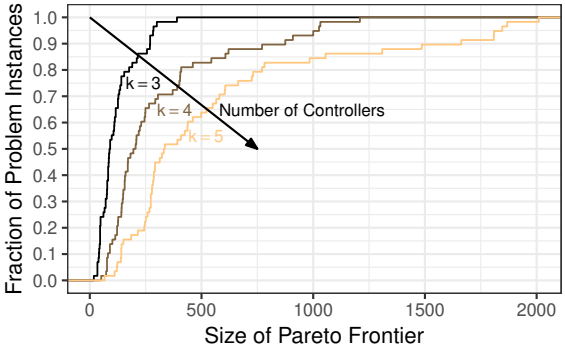
In this context, a problem instance is characterized by the number of placed controllers, k, and the network graph. Qualitatively, the statistics across all problem instances are similar to those in the Internet 2 graph, i.e., the continuous average latency measures have the largest number of distinct values. They are followed by the objectives that consider the maximum latency, which are also continuous. Finally, the imbalance is integer-valued and restricted by the number of nodes in the network, n. Hence, it attains the lowest amount of distinct values.

In addition to the number of distinct values, the variance plays an important role when quantifying the relative importance of an objective. For all problem instances, Figure 3.4 displays the distribution of the variance of each objective across all possible placements for a problem instance. Although the average latency measures attain the highest number of distinct values, they show the lowest variance. The reason for this characteristic is that the averages are formed from many individual latencies and do not differ much between placements. Furthermore, the variance of the average inter-controller latency is higher than that of the average node-to-controller latency. This stems from the fact that $\pi^{\text{avg inter-latency}}$ is based on fewer individual latencies and can take on extreme values when all instances are placed close to each other in a cluster or are distributed at the edge of the network, respectively. As discussed in the previous section, objectives that quantify the maximum latency have a higher variance due to the wider range of attained values (cf. Figure 3.2). Similarly, the imbalance measure is based on the maximum load difference between controller instances and takes on a large number of values, resulting in a high variance. For all objectives, the $90\,\%$ quantiles of the variance distribution are below 0.05. Additionally, the variance of the five objectives fluctuates significantly between individual problem instances. This phenomenon indicates that the order of objectives is a characteristic of the specific problem instances and therefore highlights the importance of assessing the relative importance of objectives on a per-problem basis.

To further motivate the need for mechanisms that map a multi-obective result vector to a single score, the distributions of the number of Pareto-optimal



(a) Variance per objective function across all problem instances that are investigated in this work.



(b) Number of Pareto optimal placements for different numbers of controllers that are placed.

Figure 3.4: Empirical CDFs of the variance per objective and the size of the Pareto frontier.

placements for different numbers of placed controllers, k, are illustrated in Figure 3.4b. For a given number m of Pareto-optimal placements on the x-axis, the

value on the y-axis represents the fraction of problem instances whose five dimensional Pareto frontier includes up to m elements. The different numbers of controllers, k, are denoted by differently colored curves. The number of placed controller instances has a direct impact on the total number of distinct placements, which can be calculated as the binomial coefficient $\binom{n}{k}$. Hence, the size of the Pareto frontier also increases due to the increase of incomparable pairs of objective vectors, in particular. While a human decision maker might compare and choose from a couple of alternatives in an objective space with few dimensions, comparing one thousand and more different solutions in a practical time frame is unlikely. This highlights the necessity for the automated decision making mechanisms that are developed in this work.

3.3 Weighting Methods

In order to aggregate the performance of a placement that is evaluated with respect to multiple objective functions into a single value, the mechanisms that are analyzed in this chapter require weights for each considered dimension. Hence, we first discuss methods for obtaining these weights based on the set of placements and the corresponding objective values.

In the following, the weight of the $j-$th objective is denoted as w_j and all weights are normalized, i.e., $\sum_{j=1}^{m} w_j = 1$ in case of m objective functions. Additionally, objective values are also normalized prior to applying the weighting mechanisms. The observed values for n placements and m objective dimensions are stored in an $n \times m$ matrix A which is transformed into the normalized matrix R according to Equation 3.1:

$$r_{i,j} = \frac{a_j^{\max} + a_j^{\min} - a_{ij}}{a_j^{\max} + a_j^{\min}}. \tag{3.1}$$

In this equation, $a_j^{\min} = \min_i a_{ij}$ and $a_j^{\max} = \max_i a_{ij}$ refer to the minimum and maximum values of the j-th objective, respectively. Apart from constraining the domain of each positive-valued objective to $[0; 1)$, this transformation also

reverses the order of the values. This is done in order to account for the fact that the underlying optimization goal consists of minimizing objectives that represent various latencies and the load imbalance among controllers. Consequently, the lowest values in the objective domain are assigned the highest weights.

3.3.1 Uniform Weighting

As a baseline naïve approach, we use a weighting mechanism that does not take into account any observed data and assigns equal weights to every objective, i.e., $w_j^{\text{uni}} = \frac{1}{m}$.

3.3.2 Entropy-Based Weighting

In information theory, (the Shannon) entropy is used as a means to quantify the amount of information that is stored in a message [91]. The key idea behind the entropy-based weighting method consists of assigning higher weights to objective dimensions that carry more information, i.e., those that have a higher number of distinct values and low individual occurrence probabilities for each value. Based on [92], the weights are calculated in three steps. First, observed values are normalized for each dimension via

$$p_{i,j} = \frac{r_{i,j}}{\sum_{i=1}^{n} r_{i,j}},\ j \in \{1, \ldots, m\}. \tag{3.2}$$

Then, the entropy is determined by means of

$$e_j = -\frac{1}{\ln n} \sum_{i=1}^{n} p_{i,j} \ln p_{i,j},\ j \in \{1, \ldots, m\}. \tag{3.3}$$

Finally, the weight is calculated as

$$w_j^{\text{ent}} = \frac{1 - e_j}{\sum_{i=1}^{m} (1 - e_i)},\ j \in \{1, \ldots, m\}. \tag{3.4}$$

3.3.3 Weighting Based on the Coefficient of Variation

Intuitively, objectives whose values cover a wide range of different values tend to have a higher impact on the total resulting performance of a placement than objectives that attain only few values or values that are very close to each other. Hence, we investigate the suitability of the coefficient of variation for quantifying the relative importance of an objective. The coefficient of variation is defined as the ratio between the standard deviation and the mean of observed values. Thus, the weights are calculated according to

$$w_j^{\text{cv}} = \frac{\frac{\sigma_j}{\mu_j}}{\sum_{i=1}^{m} \frac{\sigma_i}{\mu_i}}, \; j \in \{1, \ldots, m\}. \tag{3.5}$$

In this equation, σ_j and μ_j refer to the standard deviation and mean of the j-th objective, respectively.

3.3.4 Weighting Based on the Standard Deviation

Similarly to the weighting approach that is based on the coefficient of variation, the following equation is used to calculate relative weights that are derived from the standard deviation:

$$w_j^{\text{sd}} = \frac{\sigma_j}{\sum_{i=1}^{m} \sigma_i}, \; j \in \{1, \ldots, m\}. \tag{3.6}$$

3.3.5 Comparison

In order to allow for a comparison between the different weighting mechanisms, Figure 3.5 presents the weights of individual objectives according to the four weighting approaches for the Internet2 OS3E topology. The x-axis denotes the three objectives, the height and color of the bars represent the weight and weighting method, respectively.

While the weights that are returned by the different mechanisms differ in terms of absolute values, the relative order of objectives within each weight-

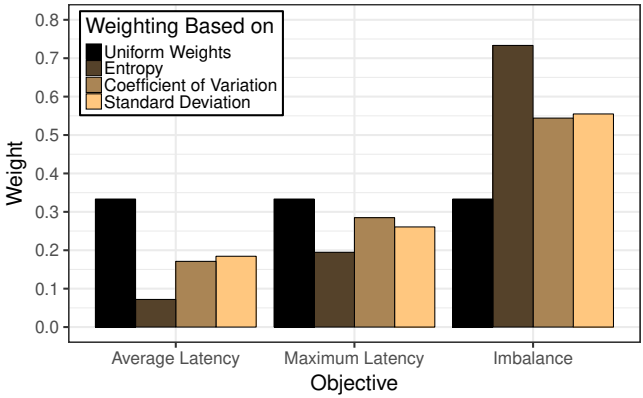


Figure 3.5: Relative weights of objectives according to different weighting mechanisms for the Internet2 OS3E topology.

ing mechanism is consistent. Having the lowest variance and the narrowest interquartile range, the node-to-controller latency is assigned the lowest weights. As discussed in Section 3.2, the maximum-based measure has a higher variance and thus also results in higher weights when compared to its average-based counterpart. The highest weights are assigned to the imbalance measure. This can be explained by the high variance that is observed for the imbalance objective.

A comparison of the absolute weights that are assigned by the weighting methods shows that the mechanisms that are based on standard deviation and the coefficient of variation return similar values. This phenomenon can be explained by the fact that objective values are normalized prior to applying the weighting methods. Thus, the normalization using the mean that is applied in the context of the latter does not have a large impact on the final weights. Finally, the entropy-based weighting approach yields the widest range of weights, i.e., between less than 0.1 and more than 0.6. This indicates a higher sensitiv-

ity towards the objectives' variance, which is a significant influence factor on the resulting weight for all weighting methods that take into account observed objective values.

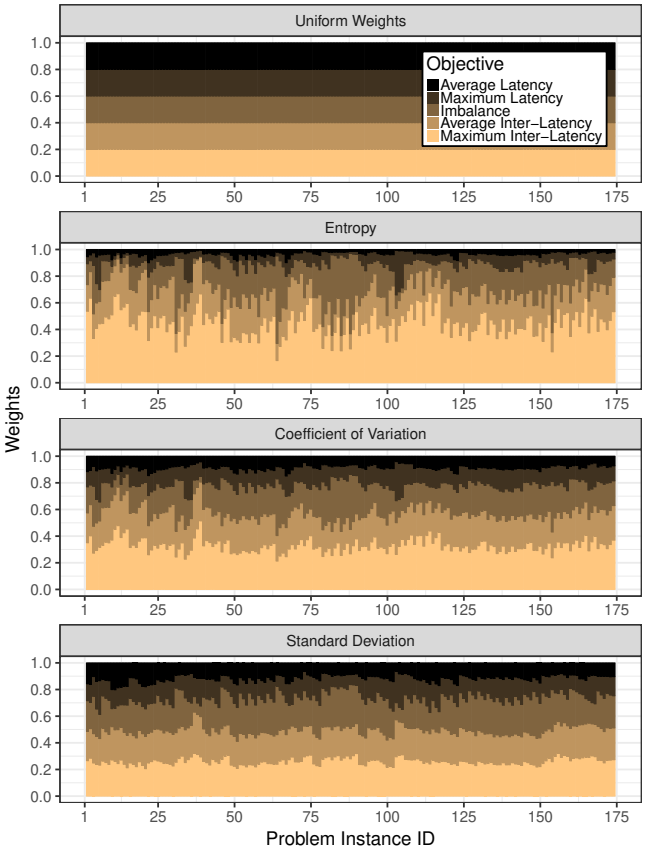


Figure 3.6: Relative weights per objective, problem instance, and weighting mechanism.

Figure 3.6 shows the weights that are returned when applying each weighting method to the problem instances of the Topology Zoo and five dimensions. The first subfigure shows weights for uniform weighting, the second subfigure those for entropy-based weighting, and the bottom plots show the resulting weights based on the coefficient of variation and standard deviation, respectively. The x-axis of each plot represents the IDs of the different problem instances. Each bar shows the weights of each dimension according to the investigated weighting method. From top to bottom, the weights of average node-to-controller latency (black), maximum node-to-controller latency (dark brown), imbalance (light brown), average inter-latency (dark orange), and maximum inter-latency (orange) are stacked. For better visibility, only the first subfigure features a legend that applies to all four plots.

Since the number of objective functions is the only information that is used by the uniform weighting approach, this mechanism assigns a weight of 0.2 to each objective in each problem instance. The entropy-based weighting gives high scores to the maximum inter-latency, which receives a weight of around 0.5 for most of the topologies. As in the case study of the Internet2 OS3E topology, the high variance of the maximum inter-latency (cf. Figure 3.4a) is responsible for these high weights. The second highest scores are given either to average inter-latency or imbalance, which fluctuate significantly depending on the particular problem instance. Maximum latency and average latency receive the smallest weights due to their small variance. As already observed for the Internet2 OS3E topology, the weighting based on the coefficient of variation has a large degree of similarity to the weighting based on standard deviation also in the context of problem instances from the Internet Topology Zoo. Furthermore, both weight distributions are less skewed than entropy-based weights. While the weighting based on the coefficient of variation generally gives higher weights to maximum inter-latency and lower weights to average latency, the resulting weights of the standard deviation method are closer to the uniform weighting. In this case, maximum inter-latency, average inter-latency, and im-

balance have weights of around 0.25 each, and maximum latency and average latency share the remaining 0.25 almost equally.

In summary, the choice of the weighting method has a significant impact on the proposed decision making process. Aside from uniform weighting, the weighting algorithms emphasize the characteristics of the dimensions differently, which results in divergent weightings. This is not only observed for the case study of the Internet2 OS3E graph, but also for the problem instances of the Internet Topology Zoo. Especially, the entropy-based weighting results in the most skew weights and shows a high variability for different topologies. In contrast, weighting based on the coefficient of variation and standard deviation results in less skew and consistent weights over all graphs.

3.4 Ranking Methods

To aggregate the scores $a_{i,j}$ of the different attributes j of the placement i to an overall ranking score ρ_i, four well-known multi-attribute decision methods are considered.

First, we consider Simple Additive Weighting (SAW) [93], which computes the overall score of the placement i by adding the normalized attribute scores $r_{i,j} = \frac{a_j^{\min}}{a_{i,j}}$ multiplied by the weights w_j:

$$\rho_i^{\text{SAW}} = \sum_{j=1}^{m} w_j \cdot r_{i,j}.$$

A similar ranking method is Multiplicative Exponent Weighting (MEW) [94], which calculates the overall score as the product of the normalized attribute scores $r_{i,j} = \frac{a_j^{\min}}{a_{i,j}}$, which are given the respective weight as exponent:

$$\rho_i^{\text{MEW}} = \prod_{j=1}^{m} r_{i,j}^{w_j}.$$

When using the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [93], the attributes are first normalized via $r_{i,j} = \frac{a_{i,j}}{\sum_i a_{i,j}^2}$. Secondly, the distances to an optimal placement with all best weighted normalized attribute values $v_j^{\min} = \min_i(w_j r_{i,j})$ and to a worst placement composed of all worst weighted normalized attribute values $v_j^{\max} = \max_i(w_j r_{i,j})$ are computed. Thirdly, the separation between the optimal and the worst placement is quantified by

$$s_i^{\min} = \sqrt{\sum_{j=1}^{m}(w_j r_{i,j} - v_j^{\min})^2}$$

and

$$s_i^{\max} = \sqrt{\sum_{j=1}^{m}(w_j r_{i,j} - v_j^{\max})^2}.$$

Finally, the resulting score corresponds to the relative closeness to the ideal solution:

$$\rho_i^{\text{TOPSIS}} = \frac{s_i^{\max}}{s_i^{\min} + s_i^{\max}}.$$

VIKOR [95] relies on the best and worst attribute values, $a_j^{\min}$ and $a_j^{\max}$. For each placement, scores are calculated by two strategies,

$$S_i = \sum_{j=1}^{m} w_j \frac{a_j^{\min} - a_{i,j}}{a_j^{\min} - a_j^{\max}}$$

and

$$R_i = \max_j \left(w_j \frac{a_j^{\min} - a_{i,j}}{a_j^{\min} - a_j^{\max}} \right).$$

These two strategies correspond to an average case and a worst case approach, respectively. A parameter γ represents the relative importance of the two aforementioned strategies and ranges from 0 to 1, i.e., $0 \leq \gamma \leq 1$. Finally, the best

and worst values of S_i and R_i are taken into account when computing the final score of each placement, i.e.,

$$\begin{aligned} S^{\text{min}} &= \min_i S_i, \\ S^{\text{max}} &= \max_i S_i, \\ R^{\text{min}} &= \min_i R_i, \text{and} \\ R^{\text{max}} &= \max_i R_i. \end{aligned}$$

This leads to the following expression when using the VIKOR approach:

$$\rho_i^{\text{VIKOR}} = \gamma \frac{S_i - S^{\text{min}}}{S^{\text{max}} - S^{\text{min}}} + (1-\gamma) \frac{R_i - R^{\text{min}}}{R^{\text{max}} - R^{\text{min}}}.$$

We set $\gamma = 0.5$ to give equal weight to both strategies.

Combining the four mechanisms that are discussed in this section with the four weighting methods that are presented in Section 3.3 results in a total of 16 different ranking methods, i.e., weighting-ranking combinations, for assessing the quality of solutions to the multi-objective placement problem. Due to the vast amount of distinct placements, we apply the 16 methods only to the subset of Pareto optimal placements.

3.4.1 Case Study of the Internet2 OS3E Topology

First, the performance of the weighting-ranking combinations is investigated for the Internet2 OS3E topology and three dimensions, i.e., rankings of the ten Pareto-optimal points are compared. Table 3.3 lists the highest and lowest correlations between different combinations in terms of Kendalls's τ and Spearman's ρ rank order correlation coefficients. It can be seen that it is possible to achieve high correlations between all ranking algorithms. In contrast, small negative correlations only result from combinations that include the VIKOR approach

with uniform weights. These observations suggest an inherent order of the elements which the investigated algorithms agree upon. Furthermore, it should be noted that combinations that use uniform weights only achieve high correlations with other combinations that use uniform weights but not with other weighting mechanisms. This is an indicator for the complementary view that is provided when using uniform weights.

Table 3.3: Highest and lowest correlations between different combinations of the weighting and ranking methods on the Internet2 OS3E topology.

Method 1	**Method 2**	**Kendall's τ**	**Spearman's ρ**
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{ent}}, \rho^{\text{MEW}})$	1.00	1.00
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{ent}}, \rho^{\text{TOPSIS}})$	1.00	1.00
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{ent}}, \rho^{\text{VIKOR}})$	1.00	1.00
$(w^{\text{sd}}, \rho^{\text{MEW}})$	$(w^{\text{sd}}, \rho^{\text{TOPSIS}})$	1.00	1.00
$(w^{\text{ent}}, \rho^{\text{MEW}})$	$(w^{\text{ent}}, \rho^{\text{TOPSIS}})$	1.00	1.00
$(w^{\text{ent}}, \rho^{\text{MEW}})$	$(w^{\text{ent}}, \rho^{\text{VIKOR}})$	1.00	1.00
$(w^{\text{uni}}, \rho^{\text{MEW}})$	$(w^{\text{uni}}, \rho^{\text{TOPSIS}})$	1.00	1.00
$(w^{\text{ent}}, \rho^{\text{TOPSIS}})$	$(w^{\text{ent}}, \rho^{\text{VIKOR}})$	1.00	1.00
$(w^{\text{sd}}, \rho^{\text{SAW}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	-0.11	-0.16
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	-0.11	-0.15
$(w^{\text{ent}}, \rho^{\text{MEW}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	-0.11	-0.15
$(w^{\text{ent}}, \rho^{\text{TOPSIS}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	-0.11	-0.15
$(w^{\text{ent}}, \rho^{\text{VIKOR}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	-0.11	-0.15

Another metric for measuring the agreement between rankings is proposed by Gordon [82]. Gordon's α is defined as the number of objects that contribute to the agreement between rankings: $\alpha := N - \delta$. This value can be computed as the difference between the length of the ranking, N, and the minimum number of objects that have to be removed to ensure a perfect agreement between the reduced rankings, δ. Gordon's α confirms the high correlation coefficients, as

there are many pairs of rankings with a perfect agreement of $\alpha = N = 10$, cf. Table 3.4. The lowest value of α is 4, highlighting that no weighting-ranking combination completely inverts the resulting order of placements.

Table 3.4: Highest and lowest Gordon α scores for weighting-ranking combinations on Internet2 OS3E topology.

Method 1	Method 2	Gordon's α
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{ent}}, \rho^{\text{MEW}})$	10
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{ent}}, \rho^{\text{TOPSIS}})$	10
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{ent}}, \rho^{\text{VIKOR}})$	10
$(w^{\text{sd}}, \rho^{\text{MEW}})$	$(w^{\text{sd}}, \rho^{\text{TOPSIS}})$	10
$(w^{\text{ent}}, \rho^{\text{MEW}})$	$(w^{\text{ent}}, \rho^{\text{VIKOR}})$	10
$(w^{\text{uni}}, \rho^{\text{MEW}})$	$(w^{\text{sd}}, \rho^{\text{TOPSIS}})$	10
$(w^{\text{ent}}, \rho^{\text{TOPSIS}})$	$(w^{\text{ent}}, \rho^{\text{VIKOR}})$	10
$(w^{\text{ent}}, \rho^{\text{SAW}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	4
$(w^{\text{uni}}, \rho^{\text{SAW}})$	$(w^{\text{sd}}, \rho^{\text{VIKOR}})$	4
$(w^{\text{sd}}, \rho^{\text{MEW}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	4
$(w^{\text{uni}}, \rho^{\text{MEW}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	4
$(w^{\text{sd}}, \rho^{\text{TOPSIS}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	4
$(w^{\text{uni}}, \rho^{\text{TOPSIS}})$	$(w^{\text{uni}}, \rho^{\text{VIKOR}})$	4

Probabilistic ranking models provide another approach for comparing the obtained rankings. Luce [84] constructs probabilities for a ranking $\rho = (i_1, i_2, \ldots, i_N)$ from conditional probabilities. Thus, after $r-1$ stages, p_{i_r} is defined as the probability that the element i_r is the most preferred element from the set of remaining elements $B = \{i_r, \ldots, i_N\}$. By repeating the choice, this gives the probability of the rating ρ as:

$$P(\rho) = \prod_{r=1}^{N-1} \frac{p_{i_r}}{\sum_{j \in B} p_j}.$$

The highest Luce probabilities are obtained by a ranking that is created by the combinations $(w^{\text{uni}}|w^{\text{sd}}, \rho^{\text{MEW}})$ and $(w^{\text{uni}}|w^{\text{cv}}|w^{\text{sd}}, \rho^{\text{TOPSIS}})$. This indicates that this ranking gives high ranks to the elements that are most preferred by many weighting-ranking combinations. Note that this ranking is also the modal ranking in the resulting set of rankings. All four entropy-based algorithms output the same ranking, which reaches the second highest Luce probabilities. Towards the other end, the SAW and VIKOR algorithms and the weighting approaches that are based on the standard deviation and the coefficient of variation output rankings with low probabilities, with the abovementioned exceptions.

Mallow's Φ-model is based on paired comparison of the ranked elements. It can be formulated as

$$P_{\rho_0,\theta}(\rho) = \left(\sum_{\rho} \theta^{X(\rho_0,\rho)} \right)^{-1} \cdot \theta^{X(\rho_0,\rho)},\ 0 \leq \theta < \infty,$$

in which $X(\rho_0, \rho)$ is Kendall's τ distance, i.e., the number of disagreements between ρ_0 and ρ. ρ_0 is an a priori set location parameter (e.g., the modal ranking or an averaged ranking), and θ is a measure of variation, which is fitted from the rankings with a table given in [86]. Following the methodology presented by Feigin and Cohen in [86], the model also allows detecting outlier rankings.

When using the averaged ranking as location parameter and fitting θ accordingly, the highest probability is obtained by the ranking of $(w^{\text{cv}}, \rho^{\text{MEW}})$. The second highest probabilities are achieved by the modal ranking, which already accounts for the highest Luce probabilities. Again, the entropy rankings have the third highest probability. This means that these three rankings are closest to the averaged ranking, which was chosen as location parameter.

When the modal ranking is used as location parameter, the order of the first and second rating changes, but the entropy rating still receives the third highest probability. The outlier detection, which mainly depends on the fitting of θ, indicates that $(w^{\text{cv}}, \rho^{\text{MEW}})$ is an outlier ranking whose probability is too high,

and that $(w^{\text{uni}}, \rho^{\text{SAW}})$ and $(w^{\text{uni}}, \rho^{\text{VIKOR}})$ are outliers whose probabilities are too low, with values close to 0.

Following the approach described in [87], Figure 3.7 shows a boxplot of the ranks of the different placements sorted by their median which is highlighted by the bold horizontal bar in each box. In this plot, the lower and upper hinges of each box correspond to the first and third quartiles while whiskers cover values whose distance to the hinges is no further than 1.5 times the inter-quartile range (IQR). Points that are not within the whiskers' range are considered to be outliers and are displayed individually.

It can be observed that the boxes for the first five placements are very narrow, indicating a large agreement among the different weighting-ranking combinations. Only for the last five placements, there is some disagreement among the different rankings. Despite this, several outliers can be observed for the entire range of placements. However, a detailed analysis of the data shows that most of these outlier ratings stem from the uniform weighting mechanism. Hence, this weighting method can be used to generate a complementary view of the placements. Such a view can be particularly useful in the context of short algorithm run times during which only a limited number of available alternatives are observed and can result in highly fluctuating weights as well as unreliable estimates of the coefficient of variation, standard deviation, and entropy.

In summary, the different ranking methods show a high agreement among each other, especially for the top-ranked placements. This means that, among the investigated methods, no weighting-ranking combination stands out and most of them are well suited to combine the Pareto-optimal placements into a single score. Furthermore, the results suggest that using uniform weights can lead to outlier rankings, which do not reproduce the majority rankings. This stems from the fact that these rankings do not take into account characteristics of the individual objectives that can be extracted from the Pareto frontier.

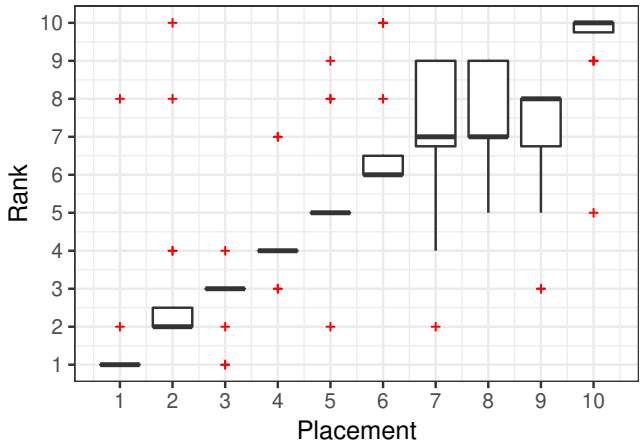


Figure 3.7: Pareto optimal placements and their ranks according to the presented ranking mechanisms.

3.4.2 Broad Evaluation on the Topology Zoo

The presented case study is generalized by applying the methodology to the 174 problem instances of the Internet Topology Zoo and five dimensions. The aggregated performance over all instances provides better insights on the performance of each of the weighting-ranking combinations. The first plot of Figure 3.8 shows the average Kendall τ correlation coefficient for all pairs of weighting-ranking combinations. Similarly, the corresponding Spearman ρ correlation coefficients are displayed in the second graph. In both plots, the color at area (i, j) indicates the average correlation coefficient between combination i and combination j over all problem instances according to the color scale on the right. A perfect correlation is indicated by an orange area (e.g., $(i, i)\ \forall\, i$), while darker colors indicate lower correlations. A correlation coefficient of 0 is shown in brown, while dark brown to black colors represent negative correlations.

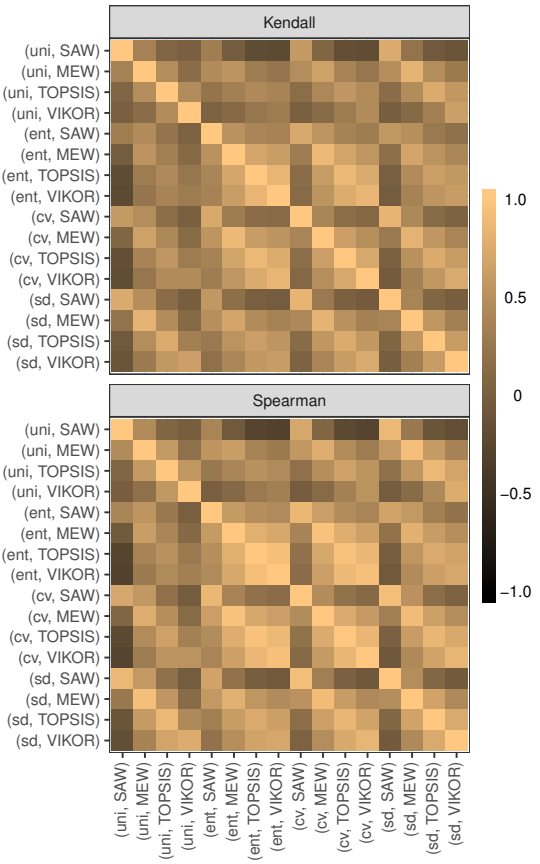


Figure 3.8: Average correlation coefficients between rankings that result from different weighting-ranking combinations for all topologies.

It can be observed that both average correlation coefficients give similar results for all pairs of combinations. The darker colors of row/column 1, 5, 9, and 13 indicate that the rankings generated by SAW generally have little or even negative correlations with the other rankings. The average Kendall and Spearman correlations are especially low for combination 1, i.e., $(w^{\text{uni}}, \rho^{\text{SAW}})$. Moreover, using uniform weights (1 – 4) results in lower average correlations, which is consistent with the prior observations regarding outlier rankings. The highest correlations are achieved with combinations that feature entropy-based weighting (5 – 8), which means that those combinations generate rankings that are likely to have a high degree of similarity not only among each other but also with rankings that are produced by other combinations.

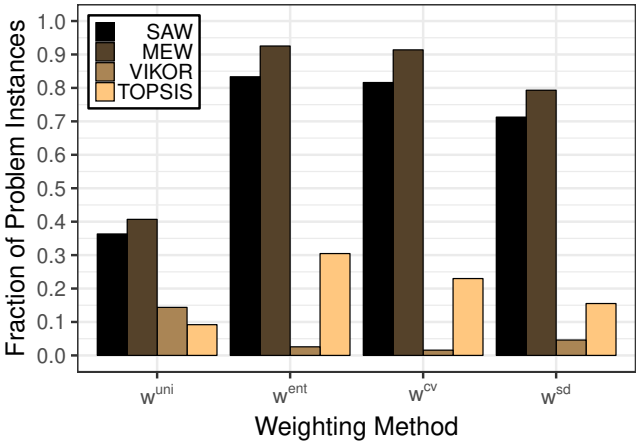


Figure 3.9: Fraction of problem instances for which the placement that is proposed by a weighting-ranking combination is in the top 3 of placements according to the ranking that uses Luce probabilities.

Figure 3.9 shows the aggregated performance of all weighting-ranking combinations according to the Luce probabilities. While the x-axis indicates the

weighting method, the color of each bar represents the ranking mechanism. For each combination, the y-axis represents the fraction of problem instances for which the placement that is returned by the corresponding combination is among the top 3 placements according to the Luce-based ranking of placements. The rationale for this type of evaluation is to tolerate small perturbations in the ranking.

We observe that the largest fraction of top 3 placements is identified when using combinations that are based on SAW and MEW. Several of these combinations manage to successfully identify top 3 placements for more than 90% of the investigated problem instances. In contrast, rankings that are based on TOPSIS and VIKOR perform significantly worse in this evaluation. In the case of the latter, this effect can be explained by the fact that VIKOR tries to achieve a trade-off between the worst and the average case rather than fully focusing on one aspect. Except for uniform weights that lead to outlier rankings and thus, poor performance, the chosen weighting technique constitutes a secondary impact factor on the performance. The highest success ratio is achieved when using the entropy and coefficient of variation-based approaches.

All in all, the problem instances of the Internet Topology Zoo reveal findings similar to those from the case study with the Internet2 OS3E topology. Uniform weightings are likely to produce rankings that exhibit a low degree of correlation with the rankings of other weighting-ranking combinations. Furthermore, using the SAW ranking algorithm provides lower correlation coefficients, but results in rankings that have high Luce probabilities on a significant portion of problem instances. A worse performance is observed for the TOPSIS and VIKOR algorithms, which rarely output rankings with high Luce probabilities. Instead, when aiming towards rankings with high Luce probabilities, combinations that are based on the MEW mechanism should be considered for ranking the Pareto-optimal placements.

3.5 Lessons Learned

Many challenges in softwarized networks revolve around placement problems in which the locations of entities such as SDN controllers, VNF instances or chains, and VMs are optimized. Since this optimization is often performed in a multi-objective fashion, operators are confronted with a multitude of possible solutions that are incomparable among each other.

The goal of this chapter consists of designing and evaluating mechanisms that enable automated decision making between such multi-dimensional solutions. To this end, we leverage techniques from the domain of multi-attribute decision making that aggregate the performance of placements to a single numeric score. We investigate a total of four weighting methods that allow determining the relative importance of individual objectives and four methods for aggregating the performance of Pareto frontiers that are returned by multi-objective optimization algorithms.

Our evaluations are performed in the context of the SDN controller placement problem and feature a case study with an exemplary topology as well as a broad evaluation on more than 170 problem instances with networks from the Internet Topology Zoo. A comparison between the resulting rankings of placements demonstrates that many techniques produce similar results in terms of the highest ranked placements. Hence, decisions based on a majority vote from multiple mechanisms can be used to identify viable candidates in the context of automated decision making and improve the robustness of the resulting decision. Moreover, the high degree of agreement between different mechanisms suggests the presence of an inherent order of placements that can be extracted with these mechanisms.

Additionally, we confirm our findings by means of two probabilistic models that derive probabilities for rankings. In this context, we show that in over 90% of cases, the combination of entropy-based weights and scoring based on multiplicative exponent weighting manages to successfully identify a placement that is among the top 3 according to the probabilistic model.

Finally, we demonstrate that using uniform weights leads to outlier rankings that exhibit a high degree of dissimilarity towards rankings that are based on the remaining weighting techniques. This phenomenon can be explained by the fact that uniform weights do not take into account available information regarding characteristics of the objectives in general and their behavior within specific problem instances in particular. Nevertheless, such outlier rankings can provide a complementary view on the ranked placements.

4 Integration of Network Management Information into the SDN Control Plane

With Software Defined Networking (SDN), operators benefit from a higher flexibility, cost efficiency, as well as programmability of their networks [26]. In addition to their southbound capabilities for controlling the forwarding plane of SDN-enabled switches, modern controller implementations also offer northbound interfaces (cf. Figure 1.1) that can be used to exchange information with entities such as Network Management Systems (NMSs). Similarly to the controller, an NMS offers a centralized view of the entire network as well as configuration capabilities. In contrast to controllers, however, an NMS can also provide information regarding legacy devices in the context of hybrid SDN deployments and usually has access to more detailed monitoring information, e.g., bandwidth and delay data on a per-link basis. A key difference between these centralized control and management entities consists of the different time scales and granularities at which information is collected, processed, and stored [96, 97].

This gives rise to several research questions. Firstly, does exchanging relevant information improve network performance in terms of aspects like throughput and fairness? In this context, metrics for quantifying the improvement are required. Secondly, what are the resulting trade-offs in terms of complexity of implementation, communication overhead, and additional hardware or software requirements? Furthermore, the migration from NMS-managed legacy networks to SDN-based networks poses a multitude of challenges. A frequently discussed

question in this context revolves around choosing an architecture and the responsibilities of involved actors, e.g., whether NMS and SDN controllers coexist and communicate or their functionalities are merged into one entity [98]. Finally, options for an incremental migration [99] and mechanisms for handling the resulting heterogeneity [100] need to be considered.

In this chapter, we study the information exchange between an NMS and an SDN controller. Our main goal consists of evaluating the impact of the additional information on the quality of control plane decisions. Hence, in order to maintain a separation of concerns between the management and control entities, our main focus is the control loop. To this end, we extend the popular ONOS (Open Network Operating System) controller [32] with the capability to receive and utilize NMS-based measurements regarding available link bandwidth as well as the bandwidth consumption of individual flows via its REST interface. Furthermore, we leverage ONOS's intent framework and annotation mechanisms to enable NMS-awareness, maintain a fair bandwidth distribution between network links, and therefore maximize the overall throughput. Our evaluations are performed in a purely SDN-based deployment without legacy components.

In addition to the abovementioned NMS-aware controller variant, we improve ONOS's default path choice algorithm in order to achieve a better flow distribution across multiple equal-cost paths. Furthermore, this modified variant represents another trade-off between the overall system complexity and required implementation effort of the NMS-aware controller and the default controller implementation.

In order to demonstrate the feasibility of the proposed NMS-aware controller, we perform an in-depth evaluation of the behavior of all three controllers in a proof-of-concept scenario. Afterwards, we investigate the influence of different parameters such as flow interarrival times, flow duration, and the number of active flows on the performance in terms of throughput and several fairness measures. Finally, the source code of all modified controller versions as well as

the corresponding evaluation framework is published via github[1,2] so that other researchers can reproduce the presented results or use them as foundation for further extensions.

This chapter is based on content that has been published in [18] and is organized as follows. In Section 4.1, we discuss related work. Section 4.2 covers details regarding the three controller implementations as well as the evaluation setup and methodology. After presenting the results of the performance comparison and the parameter study in Section 4.3, Section 4.4 concludes the chapter with an overview of lessons learned.

4.1 Background and Related Work

In this section, we provide the necessary background regarding the ONOS SDN controller in general as well as its intent and annotation mechanisms in particular. Furthermore, uses of SDN in the context of enhanced QoS control are discussed. Finally, we survey literature regarding approaches that combine SDN with network management techniques in order to improve network performance.

4.1.1 ONOS SDN Controller Platform

During the course of this work, we utilize the ONOS SDN controller [32]. Apart from offering the basic set of features that are necessary for dictating the forwarding behavior of SDN-enabled switches, the ONOS platform supports advanced features such as distributed deployments across multiple servers for scalability and fault tolerance as well as an extensible northbound interface that can be exposed via REST. Since ONOS is an open source project with partners[3] that include telecommunication providers, hardware vendors, and research institutes, it is widely used both in academia and industry.

[1] https://github.com/lsinfo3/nms-aware-onos
[2] https://github.com/lsinfo3/nms-onos-meas
[3] https://onosproject.org/members/

Furthermore, ONOS provides features that are particularly useful for the proposed interaction with external information sources like NMSs and the integration of this information into control plane decisions.

Firstly, the annotation framework provides means to enrich the controller's internal topology representation with custom information such as the available and remaining bandwidth of individual links. By exposing this feature via an API, external entities can supply their monitoring data and the controller can take it into account during its decision making process.

Secondly, the intent framework offers abstractions for defining QoS-level requirements and policies that are internally translated to appropriate flow rules. With this abstraction, an operator or network administrator does not have to explicitly define the desired path from source to destination by means of individual flow rules on the corresponding switches. Instead she just needs to specify source, destination, and possibly constraints as well as requirements w.r.t. delay or bandwidth and can rely on the controller's intent compiler to generate appropriate flow rules, install them on the switches, and even migrate the flows in order to maintain the requirements. In the latter case, upon receiving updates regarding the network's state, the controller automatically recompiles the intents and, if necessary, modifies installed flow rules in order to meet the corresponding constraints. In the case that an intent can no longer be fulfilled, the controller can handle the respective portion of the traffic in a best-effort fashion and notify the administrator or NMS of the issue. Although intent mechanisms are in development for other controller platforms such as OpenDaylight [98], at the time of writing, the one provided by ONOS was the most mature in terms of documentation, features, and stability.

4.1.2 SDN for QoS Control

Even without using a dedicated network management system, SDN-based control can be leveraged in order to improve different aspects of network performance. For instance, [101, 102] propose new OpenFlow controllers that focus on

improving the quality of multimedia applications such as adaptive video streaming. Furthermore, [103] introduces an SDN controller capable of providing suitable QoS levels for specific multimedia applications. However, in contrast to our work, the controller itself is used for data collection and monitoring rather than an external entity such as an NMS. This places an additional burden on the controller and is limited to monitoring SDN-enabled devices. Similar issues can arise with integrated controller platforms like OpenDaylight [98] that provide plugins for legacy protocols such as SNMP and therefore add responsibilities to the SDN controller. Finally, [104] and [105] focus on the use of SDN for bandwidth-related optimization. In particular, the authors of [105] demonstrate that OpenFlow can be used to enable per-flow bandwidth guarantees. However, the bandwidth requirements and flow-to-path assignments are already preset whereas they are exchanged and calculated dynamically in our case. Furthermore, this prior knowledge of flows' bandwidth requirements allows enforcing said bandwidth guarantees by performing admission control at the controller, i.e., rejecting flows whose bandwidth requirements exceed the remaining bandwidth of available paths.

4.1.3 Management Architectures with SDN Components

Numerous research efforts focus on management frameworks that leverage the SDN paradigm. By using an integrated network management and control system (i-NMCS), [106] proposes a QoS control architecture that can be configured by means of policies. In contrast to developing and using a specialized SDN controller such as the i-NMCS, our NMS-aware approach is based on the publicly available and frequently updated ONOS controller. In [107], an NMS is used to manage an SDN-based network by extending the SDN controller with northbound SNMP capabilities. While this approach enables monitoring SDN components, it lacks means to configure the controller. We address this by actively using NMS information in SDN control plane decisions.

The authors of [108] present two methods of integrating SDN and non-SDN management mechanisms. In this context, distributed operations centers collect network information and communicate with SDN controllers either by means of their southbound or east- and westbound interfaces that are extended with support for non-SDN entities. We follow a third path by utilizing the global view of the NMS and minimizing the management overhead that is incurred at the controller.

A completely new management framework including a high level language to achieve event driven network control is presented in [109]. By using policies that evolve based on aspects like time, history, users, or traffic characteristics, decisions can be made at a fine granularity. In contrast, we enable the integration of existing NMSs by providing appropriate interfaces at the SDN controller. Policies can then be implemented on demand by means of the controller's intent and constraint mechanisms. Finally, a decentralized network management framework is proposed in [110]. While this framework relies on a novel protocol for the communication with control entities, we use existing REST interfaces for the exchange of management information.

4.2 Measurement Evnironment and Components

In this section, the measurement environment is described alongside its core components. This includes details regarding the three ONOS controller types, the NMS, and the traffic generator. Finally, we present an overview of our experiments as well as performance indicators and parameters.

4.2.1 SDN Controllers

In order to evaluate trade-offs in terms of complexity and performance gains, we consider a total of three types of ONOS controllers [32] in this work. As discussed in the previous section, we chose ONOS for several reasons. Firstly, its annotation framework provides means to directly integrate external infor-

mation into the control plane. Secondly, in combination with the intent framework, flow rules can be recompiled and modified automatically upon receiving updated information.

Default ONOS

Firstly, we use the default ONOS controller that is directly available from the project's git repository[4]. The default controller does not utilize external information sources. This ONOS variant uses a path assignment algorithm for flows that deterministically chooses the first available path in the case of equal-cost paths. Hence, using this controller can lead to significant load imbalance in networks that feature many such paths. In extreme cases, this can lead to one link or path being overloaded while others are completely idle. Therefore, we implement a controller type that alleviates this issue.

During the course of this work, we show results from ONOS version 1.7.0 which was the latest stable version when our work started. However, we updated the code to work with ONOS version 1.12.1 and verified that the update did not have a significant performance impact on any of the three controller variants.

Hash-based path assignment

In order to address the abovementioned drawback, we modify the path assignment algorithm that is invoked when *packet_in* events arrive at the controller as follows. Like in the case of the default ONOS controller, a list of shortest paths from source to destination is determined. However, instead of assigning the flow that corresponds to the received packet_in event to the first path in the list, a hash of the five-tuple (src IP, src port, dst IP, dst port, protocol) is calculated. Afterwards, the hash is taken modulo the number of available paths and assigned to the corresponding path. This serves two goals. On the one hand, an expected uniform distribution of flows across paths is achieved [111]. On the other hand, flows that have the same five-tuple are always assigned to the

[4] https://github.com/opennetworkinglab/onos

same path since the result of the hash function does not change. All described changes are performed in ONOS's *ConncectivityIntentCompiler* class and have been submitted to the official repository as a pull request[5].

NMS-aware, intent-based ONOS

In the case of the NMS-aware ONOS controller, external information from the NMS is received regularly via the controller's REST API. For the presented proof-of-concept implementation, this information includes the bandwidth utilization of all flows as well as the capacity and utilization of all links in the network. While the bandwidth data could be inferred from switch statistics by the controller, we argue that this would put an additional computational burden on the controller and would also break the separation of concerns. Furthermore, the set of statistics can be easily extended with more information such as link delays or application specific estimates of the Quality of Experience (QoE) for end users.

Since ONOS provides a representation of the network topology by default, we use this *TopologyStore* to annotate the links according to the information provided by the NMS. The information is then used in three ways. Firstly, the path assignment of flows on packet_in events is based on the remaining bandwidth of involved links. In particular, flows are assigned to the shortest path with the highest remaining bandwidth. For a given path, this value is calculated as the minimum remaining bandwidth across all the links it is composed of. The rationale for this path assignment strategy consists of maximizing the probability that a flow's throughput is not limited due to bandwidth scarcity.

The second use of the external information consists of adapting to dynamically changing network conditions. When the NMS provides updated measurements, the controller can reallocate flows onto paths via bin packing in order to maximize throughput or perform load balancing across links.

Finally, ONOS's intent framework is used for maintaining flow-level QoS requirements. This is achieved with constraints such as the required or minimum

[5] https://github.com/opennetworkinglab/onos/pull/41

bandwidth of a flow that are provided by the NMS. In the case of a constraint violation, the controller automatically attempts to find an alternative path for the corresponding flow and reallocates it. Otherwise, a warning can be sent to the NMS or the network operator who can react appropriately. The majority of described modifications and adaptations are performed in ONOS's *HostToHostIntentCompiler* and its *IntentReactiveForwarding* application.

4.2.2 Testbed Setup and Interaction between Components

Since one of the main goals of this work consists of demonstrating the feasibility of the proposed NMS-aware controller, we use a simple network topology that is emulated in Mininet[6]. The individual components, such as the controller, the network, and the NMS are placed in different virtual machines (VMs) on a Linux PC[7]. Figure 4.1 shows the measurement setup that is used in the course of this work. Each VM is assigned one CPU core as well as 2 GB of RAM. The ONOS controller is deployed on the first VM. This allows to conveniently switch between controller implementations and logging their respective resource utilization. The emulated network and the traffic generating hosts are deployed on the second VM. In the context of the NMS-aware controller scenario, the NMS is also deployed on the second VM.

We use a topology that is comprised of a total of four switches that are connected according to Figure 4.1. This results in two paths between switches S1 and S3 that have equal costs in terms of the hop count. Additionally, emulated hosts are connected to switches S1 and S3 that act as client and server of the iperf[8] load generator, respectively. During an experiment, these hosts exchange TCP and UDP packets at varying rates that correspond to different flows in the network.

In order to limit the resource utilization on the Mininet VM, we restrict the available bandwidth of each path between S1 and S3 to 1 Mbps by using the

[6] http://mininet.org/
[7] Ubuntu 16.04. Intel Core i5 2520M with 2 cores and 4 threads, 3.2 GHz clock speed. 8 GB of RAM.
[8] https://iperf.fr/

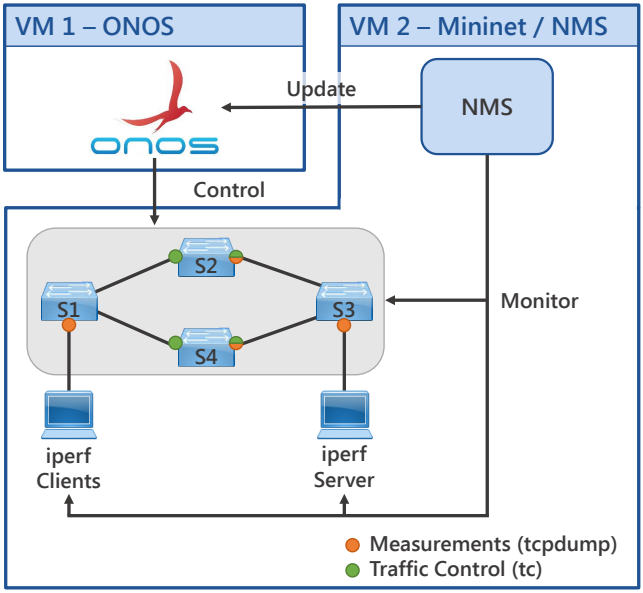


Figure 4.1: Measurement setup.

Linux traffic control tool tc on the ingress and egress ports of switches S2 and S4, respectively (cf. Figure 4.1). To allow an in-depth investigation of achieved bandwidths, packet loss, and packet reordering phenomena, we use tcpdump at different points in the network to capture and evaluate packet data. The measurement points are located at the ingress port of S1, representing the offered load, egress ports of S2 and S4, representing the load distribution between available paths, and the egress port of S3, representing the traffic that arrives at the sink.

The role of the NMS is fulfilled by a Python-based tool that was implemented during the course of this work and which is also published on github[9] alongside the controller implementations. Using a configuration file, this tool is aware of the aforementioned bandwidth limits and capacities of each link. Furthermore, SSH connections are used to instantiate iperf clients on the corresponding host and monitor the output of the iperf server at the sink. Using this data and information on the switches' flow table entries, it can regularly infer the flow distribution across paths as well as the bandwidth requirements and utilization of individual links. These updates are transferred to the controller by sending appropriate messages to its REST API. With this controller interface, it is possible to integrate external information from an arbitrary source as long as it implements the interface. This source can be a full-blown NMS or a simple measurement provider such as the one that is outlined above. Using the latter is a conscious decision to abstract the functionality of involved instances.

In scenarios in which the controller does not use the NMS as external information source, the Python tool acts as orchestrator for experiments. In particular, it sets up the measurement environment and collects as well as processes measurement information for the evaluation.

4.2.3 Experiment Design

In order to ensure a reproducible and convenient execution of experiments, we use the procedure that is described in this section. After specifying a set of parameters in a configuration file, measurement data for our evaluations is automatically collected as follows. As a first step, both VMs are freshly instantiated in order to avoid side effects from previous runs. These include unexpected forwarding behavior due to installed flows and intents in the switches and the controller that have not yet timed out as well as performance fluctuations due to unused data structures inside the controller that have not yet been removed by the garbage collector. Then, the appropriate controller type is launched inside

[9] https://github.com/lsinfo3/nms-aware-onos-measurements

the first VM. Afterwards, the configured network topology is started in Mininet and connected to the controller. This step also includes setting the bandwidth limits with tc. To enable the remote instantiation of traffic flows, SSH clients are launched on the two hosts.

Finally, the NMS regularly performs its monitoring and update tasks according to the configuration. During the entire experiment, tcpdump instances at the four points of interest that are highlighted in Figure 4.1 capture all passing traffic. After each run, this data is used to calculate the performance indicators that are defined in the next section.

4.2.4 Parameters and Performance Indicators

Several network parameters can affect the performance of the three controller types that are discussed in this work. Therefore, our evaluation framework allows varying them and quantifying their impact. The parameters can be subdivided into two categories. The first category is comprised of traffic characteristics. These include the flow interarrival time, the mean number of active flows, the mean duration and bandwidth of flows, as well as the transport protocol that is used for transmission. These parameters also implicitly control the offered load that is applied to the network since it can be derived from the number of flows and their average requested bandwidth.

Parameters related to the NMS make up the second category. One of the main influence factors is the frequency of the information exchange between the NMS and the SDN controller. This parameter is also referred to as the NMS update interval, i.e., the number of seconds between consecutive updates. Therefore, it can also be used to reflect the fact that an NMS usually operates at a coarser time scale than the controller. While control tasks are reported to have required response times in the sub-second range, it is sufficient to perform management tasks in the order of several seconds [96]. Furthermore, measurements in real world networks might be less accurate than those in a controlled experimental environment. This, in turn, can affect the controller's decision making process.

In order to evaluate the impact of such inaccuracies, our framework allows specifying a percentage by which measurement data is distorted before being sent to the controller.

In order to make quantitative statements regarding the performance implications of the different controller types, we use several metrics that cover different performance aspects. One of the most important performance indicators in a network is the throughput which also represents the resource efficiency when compared to its theoretical maximum. In this work, we define the throughput indicator $T(t)$ at time t as

$$T(t) = \frac{T_{\text{out}}(t)}{\min\left(T_{\text{in}}(t), b_{\text{max}}\right)}\,. \tag{4.1}$$

Hence, given $T_{\text{in}}(t)$ and $T_{\text{out}}(t)$, the throughput that is measured at the source and sink at time t, respectively, we calculate their ratio. This ratio also takes into account the possibility that more traffic is offered to the network than it can handle according to the bandwidth limits of the network links, b_{max}, by using the minimum operator in the denominator. Therefore, this performance indicator can attain values between 0 and 1. In this context, a value of 1 represents the best case, i.e., either the entire traffic that is sent from the source arrives at the sink or, in the case of overload, as much data as the links can handle is transported.

Apart from the throughput, a fair traffic distribution in the network is important in order to avoid overload situations. We represent this fairness from two perspectives, namely link-based and flow-based. In the case of the link-based fairness, we strive for a uniform distribution of the ratio between the traffic that passes a link and its capacity, with respect to all links. Similarly, in the case of flow-based fairness, the ratio between the throughput that a flow achieves and its requested bandwidth is considered. In both cases, we use the fairness metric defined in [112]. In contrast to other fairness metrics like Jain's fairness index, this metric has a fixed range between 0 and 1 which does not depend on the range or the composition of observed values. It is defined as

$$F = 1 - \frac{\sigma}{\sigma_{\max}}, \tag{4.2}$$

and is based on the ratio between the standard deviation of observed values and their maximum possible standard deviation. Again, a value of 1 indicates the best outcome. This is the case when the standard deviation equals 0 and therefore yields a perfectly fair distribution where each link or flow achieves the same ratio between requested and received resources.

Furthermore, we log the CPU and memory utilization of the controller VM in order to assess the computational overhead that is caused by the additional processing of external information. To this end, the *top* tool is invoked once per second on the corresponding VM and its output is stored in a log file. Since flows can also be dynamically reallocated in the case of the NMS-aware controller (cf. Section 4.2.1), we also log the number of flow reallocations per minute in order to quantify the fluctuations caused by this behavior. In case such reallocation behavior is unwanted due to sensitivity towards effects like packet reordering, it can be deactivated by performing few changes in the code.

4.3 Performance Evaluation of the NMS-Aware SDN Controller

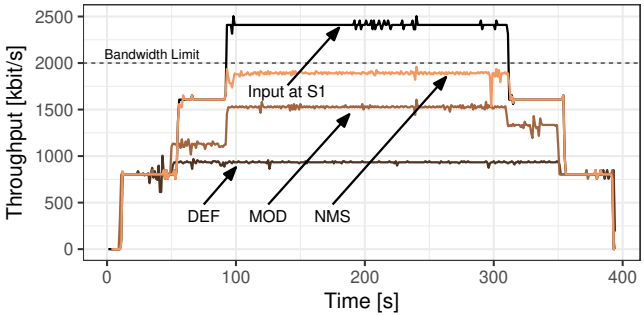
This section provides the results of the experiments that are described in Section 4.2. In order to demonstrate the viability of the NMS-aware approach and illustrate its key mechanisms, we begin with an in-depth investigation into the behavior of the three controller types in a simple scenario. Subsequently, we vary several network parameters and evaluate the performance gains across many conditions, in order to determine whether the performance improvements persist. Finally, we focus on the influence of individual network and NMS parameters on the performance of the NMS-aware controller. This allows identifying key influence factors and parameter combinations that require special attention in a real world deployment.

4.3.1 Detailed Case Study of the Controller Behavior in a Bandwidth-Limited Environment

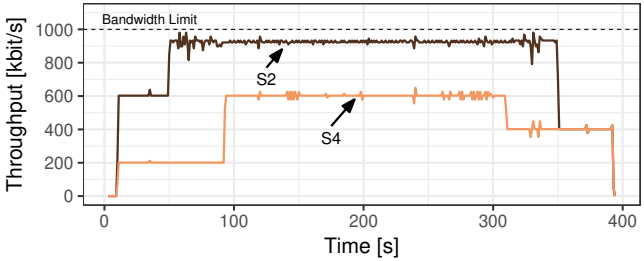
For the proof-of-concept scenario, we highlight the throughput and traffic distribution that is achieved with the three controller variants under dynamically varying load conditions. These range from a low load scenario with few active flows to an overload situation in which the total requested bandwidth exceeds the capacity of the available links. To this end, we utilize the following parameters and procedure. At the beginning of the experiment, the traffic source that is attached to switch S1 launches four iperf clients that each send UDP traffic at a rate of 200 kbps for 300 seconds. After 40 and 80 seconds, additional groups of four clients are launched with the same settings. The experiment has a total duration of 400 seconds in order to accommodate the active period of all flows. During the interval between 80 and 300 seconds, a total of 12 flows are active, resulting in an offered bitrate of 2,400 kbps. Hence, the bandwidth limit of the network which is set to a total of 2 Mbps (1 Mbps per path) is exceeded and packet loss is expected. In the remaining periods, it is possible to deliver the entire traffic from source to sink without loss. For the NMS scenario, an update interval of 10 seconds is used.

Figure 4.2 displays the throughput between source and sink during the experiment when using each of the three controller types. For the remainder of this section, these are referred to as *DEF*, *MOD*, and *NMS*, respectively.

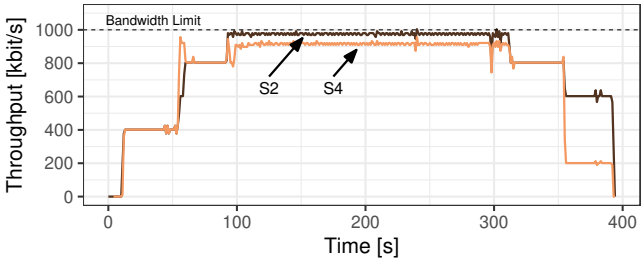
Additionally, the distribution of the traffic across the two possible paths is presented in order to highlight each type's particular behavior in terms of path assignment. In Figure 4.2a, the y-axis represents the total throughput that is measured at the ingress port of S1 and the egress port of S3, respectively. Since the former is identical for all three controller variants, it is represented by one curve. In contrast, there are three curves that correspond to the throughput that is achieved by the individual controller types. The x-axis shows the relative timestamp of each measurement.



(a) Throughput between source and sink for all controller variants.



(b) Throughput per path for modified ONOS.



(c) Throughput per path for NMS-aware ONOS.

Figure 4.2: Exemplary development of the throughput at the four measurement points for all ONOS variants.

In the case of the default ONOS controller, the throughput rises to 800 kbps as soon as the first batch of flows start. When the number of flows exceeds 5, the total throughput remains at slightly less than 1 Mbit due to the fact that all flows are routed through the first path while the second remains idle. This behavior motivates the need for the modification that is used by our hash-based path assignment mechanism. Using this mechanism results in a significantly better performance regarding the throughput. Even after more than 5 flows are active in the network, the throughput increases, up to a value of roughly 1,500 kbps. However, the gap between MOD and the curves that represent the bandwidth limit and the offered load at S1 indicates that the two available paths are not fully utilized. Finally, the NMS-aware controller manages to maintain a near optimal throughput for the majority of the experiment run. As discussed in the previous paragraph, it is not possible to achieve an exact match between source and sink during the entire experiment. This stems from the fact that during the interval between 80 and 300 seconds, more traffic is offered to the network than it can candle. However, during that interval of overload, a value close to the maximum possible throughput according to the 2 Mbps limit is achieved.

In Figures 4.2b and 4.2c, the traffic distribution among the two paths for the MOD and NMS mechanisms are displayed. These help explaining the observations in the first figure. In the case of the modified controller, the majority of flows is assigned to the path that contains S2. Therefore, this path is overutilized while the path that contains S4 has remaining unused capacity. Since the hash function does not have a memory to take into account the current flow distribution, such imbalances can occur, especially in cases with a relatively low number of flows. In contrast, the NMS-aware controller manages to distribute the flows evenly across the available paths. This results in almost equal throughput on each path and a high utilization even during the overload phase. Furthermore, the adaptation and reallocation mechanism can be seen at 60 seconds. Here, a new flow arrives and is routed through S4. Upon receiving the NMS update regarding the imbalance between paths, it is reallocated to S2 in order to maintain the balance.

Since this proof-of-concept scenario highlights the poor performance of the default path assignment strategy, we focus on the results of the MOD and NMS versions for the remainder of this work. They function as representatives for controllers that work without and with external information, respectively.

4.3.2 Investigation of Throughput, Fairness, and Overhead in Networks with Dynamic Traffic Fluctuations

In order to ensure that the observed performance gains persist beyond the proof-of-concept scenario, we now present aggregated results that are obtained from multiple runs and numerous parameter combinations. For each parameter combination, we perform experiment runs that each last 10 minutes. Since performance indicators are reported on a per-second basis and only minor performance fluctuations are observed between different runs for a given parameter combination, 10 runs are sufficient in order to obtain statistically reliable results. We use a negative exponential distribution for flow interarrival times and vary their mean between 20 and 60 seconds with a step size of 10 seconds. The mean flow duration takes on values between 160 and 480 seconds. We use combinations of the flow interarrival time and the flow duration to ensure that an average of 8 flows with 200 kbps are active in the network. This results in an average offered load of 80%. In the following, we present average values and distributions of the performance metrics achieved by the two controller types.

For all runs and all parameter combinations, Figure 4.3 shows the mean values and 95% confidence intervals for four performance metrics. These include the throughput, link and flow fairness, and the CPU load of the controller VM. In this plot, differently colored bars correspond to different controller variants.

For all network performance metrics, using the NMS-aware controller results in an improvement, as indicated by higher mean values and non-overlapping confidence intervals. In particular, a significantly higher link fairness is achieved by the NMS-aware controller. This highlights its focus on load balancing with

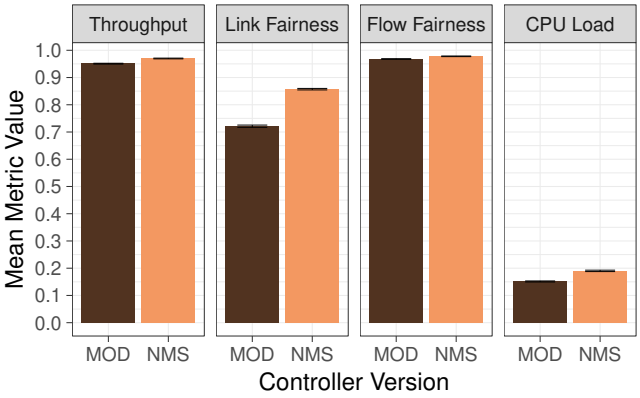
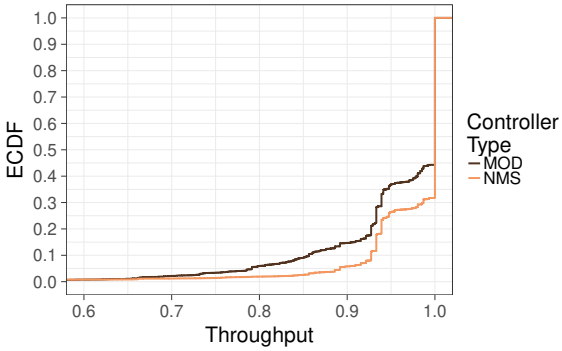


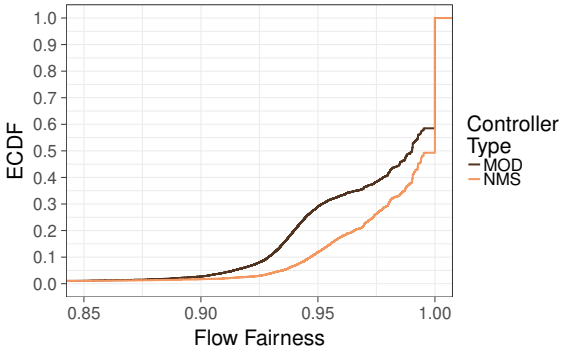
Figure 4.3: Comparison of performance indicators for modified and NMS-aware ONOS variants across all experiments.

respect to shortest paths. Additionally, the trade-off that is caused by the computational overhead for processing external NMS information is reflected by an increased CPU load at the controller. Since the two controller types achieve a similar performance with respect to the mean throughput and flow fairness metrics, we show their distributions in Figure 4.4.

The distribution of the throughput metric that is displayed in Figure 4.4a provides two main insights. Firstly, there is a significant difference in quantiles that correspond to the optimal throughput value of 1. In particular, the modified ONOS controller achieves this value in roughly 55% of the time while the NMS-aware controller does so in almost 70% of cases. This observation highlights the effect of the optimized flow to path assignment strategy that is employed by the NMS-aware controller. Secondly, using the NMS-aware controller leads to significantly fewer outliers in terms of the throughput, as indicated by the 10 percent quantiles. These differ by roughly 8% and demonstrate the stability and reliability of the NMS-aware approach.



(a) Throughput.



(b) Flow fairness.

Figure 4.4: Distributions of throughput and flow fairness values for modified and NMS-aware ONOS across all experiments.

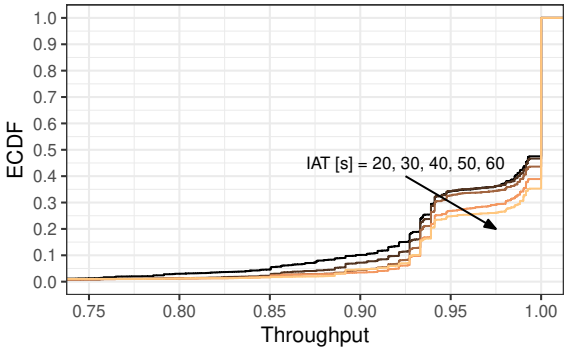
Figure 4.4b contains the distributions of the flow fairness metric. The narrower interval of displayed x-values indicates that high values are achieved by both controller variants. This can be explained by the fact that the utilized transport protocols already take care of many aspects that regard our flow fairness metric, i.e., fair resource sharing among flows on the same link. Still, the NMS-aware controller outperforms the modified ONOS controller, as highlighted by the gap between the two curves. This phenomenon stems from the fact that a better distribution of flows across links leads to a better throughput, fewer congested links, and therefore, a more stable and fair per-flow resource distribution.

4.3.3 Implications of the Flow Interarrival Time and the Information Exchange Rate

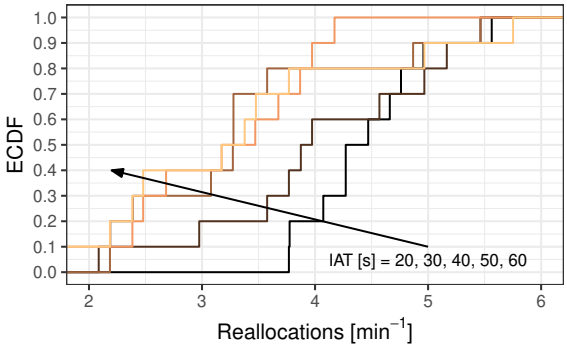
After confirming the performance improvements that are achieved with the NMS-aware controller in the previous section, this section is devoted to investigations regarding the performance impact of different network and NMS characteristics on the NMS-aware controller. Again, the presented results are based on 10 experiment repetitions per parameter value and each experiment has a run time of 10 minutes.

Figure 4.5 displays the influence of the average flow interarrival time on the throughput and the flow reallocation rate of the NMS-aware controller. Both subfigures contain empirical cumulative distribution functions for the two metrics, i.e., the y-axis shows the fraction of measurements for which the metric value is smaller than or equal to the value on the x-axis. Differently colored curves correspond to different flow interarrival times (IAT) that are varied between 20 and 60 seconds.

The results regarding the throughput metric are shown in Figure 4.5a. There are two main observations. Firstly, the throughput metric never falls below 75% and the 25 percent quantile is at roughly 93% for all parameter values. This highlights the focus of the NMS-aware controller on maximizing throughput. Secondly, the achieved throughput values increase monotonically with the flow



(a) Throughput.



(b) Flow reallocation rate.

Figure 4.5: Influence of the average flow interarrival time on the performance of the NMS-aware ONOS controller.

interarrival time. This can be explained by the fact that a higher flow interarrival time results in a lower fluctuation and that flows remain optimally distributed across links after an NMS update.

Similar observations can be made regarding the results with respect to the flow reallocation rate that is presented in Figure 4.5b. A low flow interarrival time leads to an increased fluctuation of the flow population in the network and thus, results in a higher number of flow reallocations. Note that since the reallocation rate is computed on a per-run basis rather than for each point in time as in the case of the remaining metrics, its distribution contains fewer distinct x-values.

In addition to traffic characteristics like the flow interarrival time, parameters that govern the behavior of the NMS can have a significant impact on the overall performance. Hence, Figure 4.6 displays performance results with respect to the throughput metric when the NMS update interval is varied. For this parameter, we use values between 10 and 120 seconds and present the empirical CDF of the throughput. In the figure, differently colored curves represent different update intervals.

Our first observation is that the throughput metric monotonically improves for shorter update intervals. This phenomenon can be explained by the fact that a shorter update interval leads to more optimizations at the controller and therefore faster convergence towards configurations that permit a higher throughput. On the other hand, faster update cycles also lead to more reallocations and thus, represent a trade-off between stability and optimality. However, it should be noted that most of the differences occur below the 40% quantile. Even for an update interval of 2 minutes, a median of 97% for the throughput metric is achieved. This indicates that even though the controller and the NMS operate at different time scales, the network performance can be improved by the proposed information exchange.

Further evaluations show the effects of the number of flows and the transport protocol on the performance metrics. When the number of flows is varied for a given amount of offered load, performance metrics tend to improve with

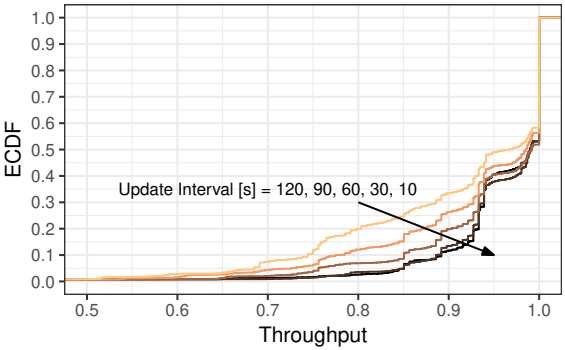


Figure 4.6: Influence of the NMS update interval on the performance of the NMS-aware ONOS controller.

a higher number of flows. This can be explained by the fact that a finer granularity for the flow to path assignment is possible in such scenarios. However, the number of reallocations also increases with the number of flows. In the case of offered load, we observe that most metrics achieve better values in scenarios with low loads. This is caused by the resulting absence or low number of reallocations and therefore stable performance throughout the measurements. Furthermore, we tested our NMS-aware controller with TCP and UDP traffic and did not observe significant performance differences between the two.

4.4 Lessons Learned

In order to efficiently control and manage large amounts of heterogeneous devices, several technologies are utilized in today's networks. On the one hand, paradigms such as software defined networking (SDN) separate the control plane from the data plane of forwarding devices and provide open interfaces in order to improve network flexibility and enable network programmability. On the other hand, network management systems (NMSs) allow monitoring,

configuring, and managing large networks. Due to the centralized view of both, SDN and NMS, and their different levels of information granularity and time scales, a mutual information exchange can be used in order to achieve various performance improvements.

In this chapter, we investigate mechanisms for integrating external NMS information into the control plane decisions of SDN controllers. Furthermore, we analyze the trade-offs with respect to the complexity of the resulting systems and their performance gains over SDN controllers that operate without external information sources.

To this end, we design and implement an NMS-aware version of the popular ONOS controller that receives external measurements via its REST API and uses them in conjunction with its intent framework to improve the decision making process with respect to the assignment of flows to paths in the network. The NMS measurements include information regarding the remaining bandwidth capacity per link as well as the bandwidth utilization per flow. Furthermore, we fix an issue in ONOS's default path assignment algorithm in order to allow for a fair comparison between the mechanisms. We publish the implementation of all controller variants as well as the framework that is used for their performance evaluation on github, enabling reproducibility and encouraging future extensions.

Our evaluations yield three main contributions. Firstly, a proof-of-concept study demonstrates the practicability of the proposed NMS-aware controller. In this context, improvements in terms of throughput and fair load distribution among flows and links are observed. Secondly, numerous parameters such as the flow interarrival time, the mean number of flows, and the average flow duration are varied in order to perform an in-depth performance comparison. Again, using the NMS-aware controller results in a significantly improved performance regarding throughput and fairness. However, we also highlight that these benefits come at the price of an increased CPU load that needs to be taken into account prior to deployment. Finally, we conduct a parameter study to determine the influence of the aforementioned parameters on the performance of the

NMS-aware controller with respect to the achieved throughput, fairness, and the number of flow reallocations. The study shows that even in the case of an information update interval that is as large as two minutes, a median throughput that corresponds to 97% of the optimal value is achieved. Hence, significant performance improvements can be achieved with the proposed information exchange mechanism between SDN controllers and NMSs while maintaining a low communication overhead.

5 Conclusion

Network operators face many challenges due to the heterogeneity and dynamicity of network traffic that is caused by a steadily growing number of devices, applications, and services that involve different stakeholders as well as strict performance requirements. In this context, the traditional approach that involves manual reconfiguration of individual devices and high capital expenditures for specialized hardware middleboxes does not provide a feasible amount of flexibility and scalability to operate the network in a resource efficient way while meeting performance goals. In order to cope with these drawbacks, network softwarization paradigms such as Software Defined Networking (SDN) and Network Functions Virtualization (NFV) promise programmable networks that heavily rely on software-based implementations of network functions. On the one hand, automating large parts of the configuration process reduces misconfigurations that are caused by human errors and allow the configuration process to happen at a significantly finer temporal granularity. On the other hand, software instances that run on standard servers can be dimensioned and migrated in such a fashion that they fulfill the performance requirements in a given situation while minimizing the amount of overprovisioned resources.

However, in order to fully reap the benefits that are offered by these network softwarization paradigms, novel challenges in the domain of management and orchestration need to be tackled. In this monograph, we propose optimization-based approaches at different levels and stages of the deployment process. Firstly, these include mechanisms for SDN controller placement that optimize the location of SDN controllers in order to address multiple objectives that affect the performance of SDN-based networks. Secondly, decision making

techniques enable automating the process of choosing a feasible solution when facing a set of Pareto optima that are returned by the multi-objective placement algorithms. Finally, approaches for integrating external information from sources like Network Management Systems (NMSs) into the SDN control loop help meeting and maintaining QoS requirements during the operational phase.

In the first part of this monograph, we design, implement, and evaluate two multi-objective heuristics for the SDN controller placement problem. By employing these heuristics, we can significantly speed up the optimization of SDN controller locations while maintaining a high level of accuracy even when facing large-scale problem instances. In addition to run times that are orders of magnitude faster than those of an exhaustive evaluation, the multi-objective approach enables an analysis of trade-offs between competing objectives such as various latency measures and the load distribution among controller instances. Furthermore, specialized heuristics like the proposed Pareto Capacitated k-Medoids (PCKM) algorithm constitute another trade-off with respect to the set of objectives that can be optimized and the resulting accuracy and run time behavior. We demonstrate the feasibility and applicability of our proposed mechanisms by evaluating them in the context of more than 50 real world topologies.

In contrast to schemes that minimize or maximize a single objective function and therefore return one distinct optimum, multi-objective optimization algorithms return Pareto frontiers of solutions that represent trade-offs between competing objectives. Hence, in order to maintain optimal performance levels even in a dynamically changing environment, techniques for quickly and automatically choosing one particular placement from the Pareto frontier are required. To this end, we propose mechanisms for automated decision making in the second part of the monograph. By assessing the relative importance of individual objectives according to four weighting methods and aggregating the performance of a given solution into one numerical score with four ranking methods, each of the 16 resulting combinations produces a ranking of Pareto optima. Our evaluations on realistic instances of the SDN controller placement

problem show a high level of agreement among the produced rankings. In particular, top-ranked placements which constitute the most relevant outcome during the decision making process are identified consistently.

Finally, we investigate the performance gains that can be achieved by integrating external information from sources like Network Management Systems (NMSs) in SDN controller decisions. Such external information can include data regarding the bandwidth, e.g., per-flow bandwidth requirements as well as per-link bandwidth utilization, but corresponding mechanisms can also be extended to take into account metrics regarding latency or application state. In addition to designing and implementing such an NMS-aware controller, we analyze and compare its performance under numerous network conditions. These include the average number of active flows in the network, the interarrival time of flows, as well as their average duration. On the one hand, our evaluations demonstrate that significant improvements in terms of throughput and fair load distribution can be achieved when leveraging this NMS-awareness. On the other hand, we show that employing such mechanisms entails a trade-off in terms of the CPU load at the controller which needs to process additional data. However, this overhead can be reduced by choosing an appropriate level of granularity for the information exchange as well as sufficiently large intervals between consecutive updates. We show that even when these intervals are in the order of magnitude of minutes, significant performance improvements can be achieved.

In summary, the mechanisms proposed in the thesis improve different performance aspects of softwarized networks for various stakeholders, ranging from network operators to end users. By utilizing placement optimization techniques during the planning phase of an SDN-based network, operators can properly dimension their network and efficiently allocate available resources in order to maintain desired performance levels. Furthermore, automated decision making algorithms can be used in conjunction with fast and specialized placement heuristics in order to dynamically react to changes in the system and maximize control plane performance. Finally, an integrated architecture that allows sharing information between centralized entities such as the SDN controller and

an NMS enables coping with network dynamics at a flow-level granularity and helps delivering a high quality of service to end users. All proposed mechanisms have been evaluated in an extensive set of realistic scenarios that cover a wide range of relevant parameters and use cases in order to highlight their feasibility and applicability.

There are several directions for future work. In the area of SDN controller placement, heuristics that are specifically tailored to environments with dynamically changing conditions could further improve the run time performance as well as lower the induced migration costs for changing placements. This could be achieved by initializing the search process of the placement algorithm with the placement at a given time instance and adding the number of configuration changes that need to be performed in order to reach a target placement as a new objective function. Similarly, the process of assessing the importance of objectives that is performed during automated decision making could be adapted in a way that the importance scores are updated in an incremental fashion rather than recalculating them for each new set of placements. Finally, in the context of NMS-aware SDN control, investigations into mechanisms for minimizing the number of reallocations or maximizing the performance while completely avoiding reallocations can improve the performance in scenarios that are sensitive to effects like packet reordering. Furthermore, the integration of additional external information like latency, QoE, or resilience can further improve various performance aspects.

Bibliography and References

Bibliography of the Author

Journal Papers

[1] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks," *IEEE Transactions on Network and Service Management - Special Issue on Efficient Management of SDN and NFV-based Systems*, 2015.

[2] M. Seufert, S. Lange, and T. Hoßfeld, "More than Topology: Joint Topology and Attribute Sampling and Generation of Social Network Graphs," *Computer Communications*, 2016.

[3] T. Zinner, S. Geissler, S. Lange, S. Gebert, M. Seufert, and P. Tran-Gia, "A Discrete-Time Model for Optimizing the Processing Time of Virtualized Network Functions," *Computer Networks*, 2017.

Conference Papers

[4] D. Klein, T. Zinner, S. Lange, V. Singeorzan, and M. Schmid, "Video Quality Monitoring based on Precomputed Frame Distortions," in *IFIP/IEEE International Workshop on Quality of Experience Centric Management (QCMan)*, 2013.

[5] D. Klein, T. Zinner, K. Borchert, S. Lange, V. Singeorzan, and M. Schmid, "Evaluation of Video Quality Monitoring based on Pre-computed Frame Distortions," in *EUNICE Workshop on Advances in Communication Networking*, 2013.

[6] C. Metter, S. Gebert, S. Lange, T. Zinner, P. Tran-Gia, and M. Jarschel, "Investigating the Impact of Network Topology on the Processing Times of SDN Controllers," in *IFIP/IEEE International Workshop on Management of the Future Internet*, 2015.

[7] A. Nguyen-Ngoc, S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, and M. Jarschel, "Investigating Isolation between Virtual Networks in Case of Congestion for a Pronto 3290 Switch," in *Workshop on Software-Defined Networking and Network Function Virtualization for Flexible Network Management (SDNFlex)*, 2015.

[8] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev, and P. Tran-Gia, "Specialized Heuristics for the Controller Placement Problem in Large Scale SDN Networks," in *International Teletraffic Congress (ITC)*, 2015.

[9] S. Lange, A. Nguyen-Ngoc, S. Gebert, T. Zinner, M. Jarschel, A. Koepsel, M. Sune, D. Raumer, S. Gallenmüller, G. Carle, and P. Tran-Gia, "Performance Benchmarking of a Software-Based LTE SGW," in *International Workshop on Management of SDN and NFV Systems (ManSDN/NFV)*, 2015.

[10] S. Gebert, A. Müssig, S. Lange, T. Zinner, N. Gray, and P. Tran-Gia, "Processing Time Comparison of a Hardware-Based Firewall and its Virtualized Counterpart," in *EAI International Conference on Mobile Networks and Management (MONAMI)*, 2016.

[11] M. Seufert, S. Lange, and M. Meixner, "Automated Decision Making Methods for the Multi-objective Optimization Task of Cloud Service Placement," in *International Workshop on Programmability for Cloud Networks and Applications (PROCON)*, 2016.

[12] S. Gebert, S. Geissler, T. Zinner, A. Nguyen-Ngoc, S. Lange, and P. Tran-Gia, "ZOOM: Lightweight SDN-based Elephant Detection," in *International Workshop on Programmability for Cloud Networks and Applications (PROCON)*, 2016.

[13] S. Gebert, T. Zinner, S. Lange, C. Schwartz, and P. Tran-Gia, "Performance Modeling of Softwarized Network Functions Using Discrete-Time Analysis," in *International Teletraffic Congress (ITC)*, 2016.

[14] A. Nguyen-Ngoc, S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, and M. Jarschel, "Performance Evaluation Mechanisms for FlowMod Message Processing in OpenFlow Switches," in *IEEE International Conference on Communications and Electronics*, 2016.

[15] M. Seufert, S. Lange, and M. Meixner, "Automated Decision Making based on Pareto Frontiers in the Context of Service Placement in Networks," in *International Teletraffic Congress (ITC)*, 2017.

[16] S. Lange, A. Grigorjew, T. Zinner, P. Tran-Gia, and M. Jarschel, "A Multi-Objective Heuristic for the Optimization of Virtual Network Function Chain Placement," in *International Teletraffic Congress (ITC)*, 2017.

[17] A. Nguyen-Ngoc, S. Lange, T. Zinner, M. Seufert, P. Tran-Gia, N. Aerts, and D. Hock, "Performance Evaluation of Selective Flow Monitoring in the ONOS Controller," in *International Workshop on Management of SDN and NFV Systems (ManSDN/NFV)*, 2017.

[18] S. Lange, L. Reinhart, T. Zinner, D. Hock, N. Gray, and P. Tran-Gia, "Integrating Network Management Information into the SDN Control Plane," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2018.

[19] A. Grigorjew, S. Lange, T. Zinner, and P. Tran-Gia, "Performance Benchmarking of Network Function Chain Placement Algorithms," in *International Conference on Measurement, Modelling and Evaluation of Computing Systems*, 2018.

[20] A. Nguyen-Ngoc, S. Raffeck, S. Lange, S. Geissler, T. Zinner, and P. Tran-Gia, "Benchmarking the ONOS Controller with OFCProbe," in *International Conference on Communications and Electronics (ICCE)*, 2018.

[21] K. Borchert, S. Lange, T. Zinner, and M. Hirth, "Identification of Delay Thresholds Representing the Perceived Quality of Enterprise Applications," in *International Workshop on Quality of Experience Management*, 2018.

[22] A. Nguyen-Ngoc, S. Lange, S. Geissler, T. Zinner, and P. Tran-Gia, "Estimating the Flow Rule Installation Time of SDN Switches when Facing Control Plane Delay," in *International Measurement, Modelling and Evaluation of Computing Systems*, 2018.

[23] M. Hirth, S. Lange, M. Seufert, and P. Tran-Gia, "Performance Evaluation of Mobile Crowdsensing for Event Detection," in *International Workshop on Crowd Assisted Sensing, Pervasive Systems and Communications*, 2018.

[24] N. Gray, S. Lange, T. Zinner, B. Pfaff, and D. Hock, "Evaluation of a Distributed Control Plane for Managing Heterogeneous SDN-enabled and Legacy Networks," in *International Conference on Communications and Electronics (ICCE)*, 2018.

Technical Reports

[25] S. Gebert, T. Zinner, S. Lange, C. Schwartz, and P. Tran-Gia, "Discrete-Time Analysis: Deriving the Distribution of the Number of Events in an Arbitrarily Distributed Interval," University of Wuerzburg, Tech. Rep., 2016, Available online: https://www3.informatik.uni-wuerzburg.de/TR/tr498.pdf.

General References

[26] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, Attributes, and Use Cases: A Compass for SDN," *IEEE Communications Magazine*, 2014.

[27] *ETSI GS NFV 002 - Architectural Framework*, Group Specification, European Telecommunications Standards Institute, 2014. [Online]. Available: https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20002v1.2.1%20-%20GS%20-%20NFV%20Architectural%20Framework.pdf.

[28] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an Open-Source Solution for Cloud Computing," *International Journal of Computer Applications*, 2012.

[29] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications (JSAC)*, 2011.

[30] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review (CCR)*, 2008.

[31] A. Tootoonchian and Y. Ganjali, "HyperFlow: a Distributed Control Plane for OpenFlow," in *Internet Network Management Conference on Research on Enterprise Networking*, 2010.

[32] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, *et al.*, "ONOS: towards an open, distributed SDN OS," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2014.

[33] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic Controller Provisioning in Software Defined Networks," in *International Conference on Network and Services Management (CNSM)*, 2013.

[34] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *ACM SIGCOMM Conference on Internet Measurement*, 2010.

[35] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *ACM SIGCOMM Conference on Internet Measurement*, 2009.

[36] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2011.

[37] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2012.

[38] Z. Drezner, *Facility Location: A Survey of Applications and Methods*. Springer, 1995.

[39] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," in *International Teletraffic Congress (ITC)*, 2013.

[40] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective optimization: Interactive and evolutionary approaches*. Springer, 2008.

[41] P. Czyżżak and A. Jaszkiewicz, "Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, 1998.

[42] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[43] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, "POCO-Framework for Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[44] S. Schmid and J. Suomela, "Exploiting locality in distributed SDN control," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013.

[45] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2012.

[46] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in *IEEE Globecom*, 2011.

[47] Y. nan Hu, W. dong Wang, X. yang Gong, X. rong Que, and S. duan Cheng, "On the Placement of Controllers in Software-Defined Networks," *The Journal of China Universities of Posts and Telecommunications (JCUPT)*, 2012.

[48] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware Controller Placement for Software-Defined Networks," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2013.

[49] F. J. Ros and P. M. Ruiz, "Five Nines of Southbound Reliability in Software-Defined Networks," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2014.

[50] *IBM ILOG CPLEX Optimizer*, 2010. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[51] S. H. Owen and M. S. Daskin, "Strategic Facility Location: A Review," *European Journal of Operational Research*, 1998.

[52] A. Archer and S. Krishnan, "Importance Sampling via Load-Balanced Facility Location," in *Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2008.

[53] F. J. F. Silva and D. S. de la Figuera, "A Capacitated Facility Location Problem with Constrained Backlogging Probabilities," *International Journal of Production Research (IJPR)*, 2007.

[54] S. Khuller, R. Pless, and Y. Sussmann, "Fault Tolerant K-center Problems," *Theoretical Computer Science*, 1997.

[55] S. Chaudhuri, N. Garg, and R. Ravi, "The p-Neighbor k-Center Problem," *Information Processing Letters (IPL)*, 1998.

[56] U. Bhattacharya, J. R. Rao, and R. N. Tiwari, "Fuzzy Multi-Criteria Facility Location Problem," *Fuzzy Sets and Systems*, 1992.

[57] M. Ehrgott, *Multicriteria Optimization*. Springer, 2005.

[58] I. Harris, C. Mumford, and M. Naim, "The Multi-Objective Uncapacitated Facility Location Problem for Green Logistics," in *IEEE Congress on Evolutionary Computation (CEC)*, 2009.

[59] A. Lancinskas and J. Zilinskas, "Solution of Multi-Objective Competitive Facility Location Problems Using Parallel NSGA-II on Large Scale Computing Systems," in *Applied Parallel and Scientific Computing*, 2013.

[60] T. Xifeng, Z. Ji, and X. Peng, "A Multi-Objective Optimization Model for Sustainable Logistics Facility Location," *Transportation Research Part D: Transport and Environment*, 2013.

[61] S. H. A. Rahmati, V. Hajipour, and S. T. A. Niaki, "A Soft-Computing Pareto-based Meta-Heuristic Algorithm for a Multi-Objective Multi-Server Facility Location Problem," *Applied Soft Computing*, 2013.

[62] A. Abraham and L. Jain, *Evolutionary multiobjective optimization*. Springer, 2005.

[63] L. Davis *et al.*, *Handbook of genetic algorithms*. Van Nostrand Reinhold, 1991.

[64] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[65] A. Auger and B. Doerr, *Theory of randomized search heuristics: Foundations and recent developments*. World Scientific, 2011.

[66] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," in *European Workshop on Software Defined Networks (EWSDN)*, 2012.

[67] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simmulated Annealing," *Science*, 1983.

[68] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, 1982.

[69] *POCO: A Framework for the Computation of Pareto-based Optimal Controller-Placements*, University of Würzburg. [Online]. Available: http://www3.informatik.uni-wuerzburg.de/poco.

[70] H. Moens and F. De Turck, "VNF-P: A Model for Efficient Placement of Virtualized Network Functions," in *International Conference on Network and Service Management (CNSM)*, 2014.

[71] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," *Journal of Network and Systems Management*, 2015.

[72] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On Orchestrating Virtual Network Functions," in *International Conference on Network and Service Management (CNSM)*, 2015.

[73] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of Virtual Network Functions," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[74] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and Placing Chains of Virtual Network Functions," in *International Conference on Cloud Networking (CloudNet)*, 2014.

[75] I. Hwang and M. Pedram, "Hierarchical virtual machine consolidation in a cloud computing system," in *IEEE International Conference on Cloud Computing (CLOUD)*, 2013.

[76] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, and T. D. Nguyen, "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.

[77] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Mathematical and Computer Modelling*, 2013.

[78] A. Dalvandi, M. Gurusamy, and K. C. Chua, "Time-aware vm-placement and routing with bandwidth guarantees in green cloud data centers," in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2013.

[79] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *IEEE/ACM International Conference on Green Computing and Communications (GreenCom)*, 2010.

[80] T. Yapicioglu and S. Oktug, "A traffic-aware virtual machine placement method for cloud data centers," in *IEEE/ACM International Conference on Utility and Cloud Computing*, 2013.

[81] M. G. Kendall, *Rank Correlation Methods*. Griffin, 1948.

[82] A. Gordon, "A Measure of the Agreement between Rankings," *Biometrika*, 1979.

[83] L. L. Thurstone, "A Law of Comparative Judgment," *Psychological Review*, 1927.

[84] R. D. Luce, *Individual Choice Behavior*. Courier Corporation, 1959.

[85] C. L. Mallows, "Non-Null Ranking Models. I," *Biometrika*, 1957.

[86] P. D. Feigin and A. Cohen, "On a Model for Concordance between Judges," *Journal of the Royal Statistical Society. Series B (Methodological)*, 1978.

[87] A. Cohen, "Analysis of Large Sets of Ranking Data," *Communications in Statistics-Theory and Methods*, 1982.

[88] A. Cohen and C. L. Mallows, "Assessing Goodness of Fit of Ranking Models to Data," *The Statistician*, 1983.

[89] P. Diaconis, "Group Representations in Probability and Statistics," *Lecture Notes-Monograph Series*, 1988.

[90] D. E. Critchlow, M. A. Fligner, and J. S. Verducci, "Probability Models on Rankings," *Journal of mathematical psychology*, 1991.

[91] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, 1948.

[92] J. Huang, "Combining Entropy Weight and TOPSIS Method for Information System Selection," in *IEEE Conference on Cybernetics and Intelligent Systems*, 2008.

[93] C.-L. Hwang and K. Yoon, *Multiple Attribute Decision Making - Methods and Applications: A State-of-the-art Survey*. Springer, 1981.

[94] S. H. Zanakis, A. Solomon, N. Wishart, and S. Dublish, "Multi-Attribute Decision Making: A Simulation Comparison of Select Methods," *European Journal of Operational Research*, 1998.

[95] S. Opricovic and G.-H. Tzeng, "Compromise Solution by MCDM Methods: A Comparative Analysis of VIKOR and TOPSIS," *European Journal of Operational Research*, 2004.

[96] A. Clemm, *Network Management Fundamentals*. 2006.

[97] M. Subramanian, *Network Management: Principles and Practice*. 2010.

[98] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven sdn controller architecture," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2014.

[99] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, "Incremental deployment of SDN in hybrid enterprise and ISP networks," in *Proceedings of the Symposium on SDN Research*, 2016.

[100] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain SDN controllers," in *IEEE Network Operations and Management Symposium (NOMS)*, 2014.

[101] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in *Asia-Pacific Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2012.

[102] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "An optimization framework for QoS-enabled adaptive video streaming over OpenFlow networks," *IEEE Transactions on Multimedia*, 2013.

[103] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," in *22nd Telecommunications Forum Telfor*, 2014.

[104] P. Qin, B. Dai, B. Huang, and G. Xu, "Bandwidth-aware scheduling with sdn in hadoop: A new trend for big data," *IEEE Systems Journal*, 2015.

[105] H. Krishna, N. L. van Adrichem, and F. A. Kuipers, "Providing bandwidth guarantees with OpenFlow," in *Symposium on Communications and Vehicular Technologies (SCVT)*, 2016.

[106] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim, and D. Pinheiro, "Enhancing network management frameworks with SDN-like control," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2013.

[107] Y. Zhang, X. Gong, Y. Hu, W. Wang, and X. Que, "SDNMP: Enabling SDN management using traditional NMS," in *IEEE International Conference on Communication Workshop (ICCW)*, 2015.

[108] D. Kim, J.-M. Gil, G. Wang, and S.-H. Kim, "Integrated SDN and non-SDN network management approaches for future internet environment," in *Multimedia and Ubiquitous Engineering*, 2013.

[109] H. Kim, A. Voellmy, S. Burnett, N. Feamster, and R. Clark, "Lithium: Event-driven network control," Tech. Rep., 2012.

[110] D. Valocchi, D. Tuncer, M. Charalambides, M. Femminella, G. Reali, and G. Pavlou, "SigMA: Signaling Framework for Decentralized Network Management Applications," *IEEE Transactions on Network and Service Management*, 2017.

[111] R. Martin, M. Menth, and M. Hemmkeppler, "Accuracy and dynamics of hash-based load balancing algorithms for multipath Internet routing," in *International Conference on Broadband Communications, Networks and Systems (BROADNETS)*, 2006.

[112] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela, "Definition of QoE fairness in shared systems," *IEEE Communications Letters*, 2017.

ISSN 1432-8801