Objective-Driven Operations of Self-Organizing Networks

Christoph Frenzel

Dissertation
for the degree of
Doctor of Natural Sciences (Dr. rer. nat.)

University of Augsburg
Department of Computer Science
Software Methodologies for Distributed Systems

March 2016

Objective-Driven Operations of Self-Organizing Networks

Supervisor:	**Prof. Dr. Bernhard Bauer** Department of Computer Science, University of Augsburg, Germany
Advisor:	**Prof. Dr. Jörg Hähner** Department of Computer Science, University of Augsburg, Germany
Thesis Defense:	July 27, 2016

Abstract

Mobile communications are an integral part of modern society. People want to talk to their friends, play on-line games, or watch YouTube videos while they are on the go. This forces Mobile Network Operators (MNOs) to continuously upgrade and extend their mobile networks in order to keep up with the ever growing data rate requirements and provide the users with satisfactory Quality of Service (QoS). However, user are not willing to pay more for mobile communications leading to a decrease in the price per data unit. Due to this development, MNOs are faced with a serious pressure to reduce their Operational Expenditure (OPEX). The focus, thereby, lies on the reduction of manual efforts in network operations, i.e., the configuration and control of the Base Stations (BSs) such that the operational objectives by the MNO, defined as target values for network Key Performance Indicators (KPIs), are satisfied.

Self-Organizing Network (SON) is a concept for *automating* the operation of mobile networks by relieving human operators from repetitive, low-level network operations tasks. In principle, a SON consists of a set of close-loop control functions, referred to as SON functions, which perform specific operations tasks. For instance, Coverage and Capacity Optimization (CCO) improves the network capacity and Mobility Robustness Optimization (MRO) optimizes the handover performance. The introduction of SON lifts the work of human operators from network operations to SON operations: instead of analyzing and configuring all the BSs in the network, the operational personnel has to configure and control the SON functions in order to adapt them to the network environment and specific operational objectives. This is due to the gap between the objectives and the low-level, technical configuration interface of the SON functions. As a result, MNOs ask for a further increase in the level of automation.

This thesis introduces the Objective-Driven SON Operations (ODSO) concept that enables *autonomic* mobile network operations, i.e., the control of network operations through operational objectives on network KPIs. ODSO is built on top of SON and automates three manual tasks of SON operations: SON management that configures the SON functions, SON coordination that detects and resolves run-time conflicts between the concurrently running SON functions, and SON self-healing that detects and resolves failures in the network and the SON system. The specific contributions of this thesis are:

1. We analyze the specific requirements for autonomic operations of a mobile network and develop an architecture that extends an existing SON system with the ability to process the operational objectives of the MNO. The architecture integrates three components, one for each SON operations task, in order to reduce the manual control efforts.

2. We develop a generic decision making process that enables the ODSO components to autonomically control SON functions based on two types of information: a formalized representation of the operational objectives in an objective

model, and a formalized representation of the task-specific technical knowledge in several technical models. On the foundation of Multiattribute Utility Theory (MAUT), the ODSO components are able to process these models in order to automatically operate the SON functions such that they optimize the network to satisfy the operational objectives. In this way, the MNO's objectives serve as a unified, high-level operations interface for SON.

3. We apply the generic decision making process to each of the three SON operations tasks which enables them to determine the best configuration of the SON functions, the best approach to resolve run-time conflicts, and the best recovery action to overcome failures with respect to the provided operational objectives. Thereby, we are able to account for the diverse characteristics of each task with our flexible approach.

4. We evaluate the feasibility and performance of the ODSO approach in a simulation of a realistic mobile network. As a result, we are able to show that the increased automation in SON operations reduces manual efforts and may also lift additional optimization potential in network operations. This leads to improved network performance compared to SON-based network operations.

Acknowledgments

I am deeply grateful to Prof. Dr. Bernhard Bauer for giving me the opportunity to write my dissertation under his supervision. His encouragement and guidance provided me with the motivation and inspiration to overcome the problems and doubts that one inevitably is faced with during such an undertaking. Furthermore, the attendance at numerous conferences and meetings would have not been possible without his financial support. With the same gratitude, I would like to thank Dr. Henning Sanneck for enabling me to conduct my research as an internal Ph.D. student with Nokia. Our regular discussions were invaluable for me as he always challenged my results and ideas which provided me with very helpful feedback for further research. I also thank Prof. Dr. Jörg Hähner, who accepted to be the advisor of my thesis.

Special thanks go to Christoph Schmelz and Simon Lohmüller for the uncountable discussions that contributed to my thesis in numerous ways. They continually inspired me to create and evolve my own ideas. I am thankful that they agreed to review the result, as well. Apart from this, I thank them for the great time that we had at several conferences and numerous project meetings.

I thank all my current and prior colleagues at the Software Methodologies for Distributed Systems Lab and at Nokia for creating a friendly and happy atmosphere. I enjoyed the open discussions about challenging questions beyond my focused research topic which, more than once, triggered new ideas. Especially, I would like to thank Raphael Romeikat and Tobias Bandh for their support during the first years of my research, and Tsvetko Tsvetkov for his support during the last years. I particularly like to express my appreciation for the regular table football matches which were, more than often, a welcomed opportunity for a break.

I thank all other people who directly or indirectly supported me in writing my thesis. All informal support by my friends, fellow researchers who I met at project meetings or scientific conferences, and students had a very positive influence on myself and my research. Thereby, I like to especially thank Sebastian Eder for the numerous pub discussions that often pushed me forward when I got stuck.

I would like to express special thanks to my mother Helga for her encouragement and for providing me with an excellent starting point to perform my studies. Furthermore, I thank my brother Oliver and his family for all their support, understanding, and patience. They have always encouraged me to do my best in all matters of life.

Contents

Abstract iii

Acknowledgments v

1 Introduction 1
- 1.1 Problems and Objectives . . . 4
- 1.2 Solution Approach and Contributions . . . 8
- 1.3 Outline . . . 11
- 1.4 Publications . . . 13

2 Background 19
- 2.1 Self-Organizing Networks . . . 19
  - 2.1.1 Mobile Networks . . . 19
  - 2.1.2 Drivers for SON . . . 22
  - 2.1.3 SON Use Cases . . . 23
  - 2.1.4 SON Functions . . . 23
  - 2.1.5 SON Coordination . . . 25
- 2.2 Policy-Based Management . . . 26
  - 2.2.1 Action Policy . . . 28
  - 2.2.2 Goal Policy . . . 29
  - 2.2.3 Utility Function Policy . . . 29
- 2.3 Decision Theory . . . 30
  - 2.3.1 Preferences . . . 31
  - 2.3.2 Expected Utility Theory . . . 31
  - 2.3.3 Multiattribute Decision Making . . . 32
  - 2.3.4 Influence Diagrams . . . 35

3 Objective-Driven SON Operations 39
- 3.1 Problem and Motivation . . . 39
  - 3.1.1 Operational Objectives . . . 42
  - 3.1.2 Technical Expertise . . . 47
  - 3.1.3 Operations Process . . . 50
- 3.2 Goals and Requirements . . . 52
- 3.3 Architecture . . . 55
  - 3.3.1 Mobile Network . . . 55
  - 3.3.2 SON Functions . . . 57
  - 3.3.3 SON Operations . . . 60
- 3.4 Generic Component Design . . . 62
  - 3.4.1 Technical Model . . . 64
  - 3.4.2 Action Proposal . . . 69
  - 3.4.3 Objective Model . . . 71
  - 3.4.4 Action Selection . . . 81

3.5 Operations Process . . . 85
3.6 Related Work . . . 87
3.6.1 Policy-Based Management . . . 87
3.6.2 SON Operations . . . 94

4 SON Management 99
4.1 Problem and Motivation . . . 99
4.2 Goals and Requirements . . . 101
4.3 Component Design . . . 102
4.3.1 Trigger . . . 102
4.3.2 SON Function Model . . . 105
4.3.3 SON Function Model Combination . . . 107
4.3.4 SON Configuration Selection . . . 115
4.3.5 Enforcement . . . 115
4.3.6 SON Function Effect Computation . . . 115
4.4 Related Work . . . 116
4.4.1 SOCRATES Project . . . 117
4.4.2 UniverSelf Project . . . 117
4.4.3 COMMUNE Project . . . 118
4.4.4 SEMAFOUR Project . . . 118

5 SON Coordination 121
5.1 Problem and Motivation . . . 121
5.2 Goals and Requirements . . . 123
5.3 Component Design . . . 125
5.3.1 Conflict Detection and Technical Resolution . . . 126
5.3.2 SON Function Request Selection . . . 134
5.4 Approach for Asynchronous Coordination . . . 146
5.5 Related Work . . . 147
5.5.1 Off-line Coordination . . . 147
5.5.2 On-line Coordination . . . 148

6 SON Self-Healing 153
6.1 Problem and Motivation . . . 153
6.1.1 Involvement of SON . . . 155
6.1.2 Objective-Driven Degradation Recovery . . . 156
6.2 Goals and Requirements . . . 156
6.3 Component Design . . . 158
6.3.1 Detection of Degradations . . . 159
6.3.2 Root Cause Diagnosis . . . 162
6.3.3 Degradation Recovery . . . 163
6.3.4 Recovery Enforcement . . . 176
6.4 Related Work . . . 179
6.4.1 Involvement of SON . . . 179
6.4.2 Degradation Recovery . . . 180

7 Evaluation **183**
7.1 Scope and Approach Overview . . . 183
7.2 Simulation Setup . . . 184
7.2.1 LTE Network Simulator . . . 185
7.2.2 SON Function Engine . . . 186
7.2.3 Objective-Driven SON Operations . . . 190
7.2.4 Common Scenario . . . 192
7.3 SON Management . . . 194
7.3.1 Scenario . . . 196
7.3.2 Results . . . 200
7.4 SON Coordination . . . 207
7.4.1 Scenario . . . 207
7.4.2 Results . . . 209
7.5 SON Self-Healing . . . 212
7.5.1 Involvement of SON . . . 212
7.5.2 Degradation Recovery . . . 215
7.6 Summary and Discussion . . . 223

8 Conclusion and Outlook **227**
8.1 Summary . . . 227
8.2 Future Work . . . 230

Bibliography **235**

List of Acronyms **253**

List of Symbols **257**

List of Definitions **263**

List of Figures **265**

List of Tables **269**

List of Listings **271**

1 Introduction

Information and Communications Technology (ICT) is a major economic and social driver in the modern connected world. Thereby, mobile cellular communications manifested as the key means of access: in 2015, 96.8% of the world population had mobile telephone access while only 10.8% had access to a fixed-line telephone [ITU15]. The total direct and indirect contribution, e.g., through increasing productivity of other sectors, to the worldwide gross domestic product by mobile telecommunications was 3.8% in 2014 [GSM15]. And mobile telecommunication is expected to grow tremendously in the future, mainly driven by three forces: First, the extended growth of mobile broadband users. Today, only 47.2% of the world population has mobile broadband access with at least Third Generation (3G) level [ITU15]. However, between 2015 and 2021, the number of subscriptions is expected to grow annually by 15% [Eri15]. Second, the increased data requirements by the users. Driven by the user growth, but even more by the prevalence of high data rate applications like mobile video streaming via YouTube, the mobile traffic will multiply by 10 in the next years from 5.3 exabytes per month in 2015 to 51 exabytes per month in 2021 [Eri15]. Third, the amount of Machine-to-Machine (M2M) communications, driven by trends like Internet of Things (IoT), will constitute an increasing share of mobile connections, growing form 4% in 2015 to 10% in 2020 [GSM15]. As a result of this enormous dynamic in mobile communications, the total contribution to the worldwide gross domestic product is expected to increase to 4.2% by 2020 [GSM15].

In order to keep up with the dynamic growth, Mobile Network Operators (MNOs) are continuously extending their networks along two dimensions: On the one hand, new Base Stations (BSs) that span up the mobile network are deployed. This aims at extending the spatial range of the network such that mobile network coverage is provided in new areas of the world. However, the deployment of small cells as a new cell layer can also densify the mobile network in order to increase its capacity such that the data throughput is increased. On the other hand, the vendors of mobile networks are developing new, improved Radio Access Technologies (RATs) that MNOs often quickly adopt. Whereas 3G networks, e.g., Universal Mobile Telecommunications System (UMTS), provided up to 14 megabits per second (Mbps) peak data rate, the subsequent Fourth Generation (4G) technology, e.g., Long Term Evolution (LTE), can provide up to 1 gigabits per second (Gbps), and future Fifth Generation (5G) networks are expected to increase this to 50 Gbps peak data rate [Sam15]. The extension of mobile networks along these two dimensions turns them into Heterogeneous Networks (HetNets) that consist of multiple cell layers and RATs [Sar+11]. This

trend results in an increased difficulty of operating such complex mobile networks.

Although the growth in mobile communications increases the revenue of MNOs worldwide, the price per data unit is constantly decreasing [ITU15]. In the developed countries, e.g., Germany [Bun15], the revenue is even stagnating. For instance, the price per minute for a mobile telephone connection decreased by 62% from 2004 to 2014 [GSM15]. In the future, this is expected to continue as the International Telecommunication Union (ITU) set the goal that "[w]orldwide, telecommunication/ICT should be 40% more affordable by 2020" [ITU15, p. 6]. At the same time, MNOs face huge Capital Expenditures (CAPEXs) of 1.4 trillion US$ in their mobile networks between 2014 and 2020 in order to meet future requirements [GSM15]. Hence, MNOs need to decrease the Operational Expenditure (OPEX) to manage an ever growing network in order to stay profitable as the revue per bit decreases. As a result of the expected future development, MNOs see the need to control their costs and increase network capacity as two major drivers for the further development of mobile networks [Sch+13]. Thereby, a shift "away from classical, human operator-driven network management towards automated and autonomous management" [Sch+13, p. 25] of mobile networks, also referred to as Operations, Administration, and Maintenance (OAM), is the key. This is underpinned by the decreasing number of employees in telecommunications, e.g., in Germany going down from 225 300 in 2004 to 168 900 in 2014 [Bun15].

Self-Organizing Network (SON) is an approach, driven mainly by the Next Generation Mobile Networks Alliance (NGMN) and 3rd Generation Partnership Project (3GPP), to increase the automation in OAM of mobile networks [HSS11]. Specifically, SON identifies and describes several concrete OAM tasks, also referred to use cases, as candidates for automation. These span a broad range of configuration, optimization, and troubleshooting tasks. In traditional network operations, these use cases are performed manually by the operational personnel. A typically SON use case, e.g., Coverage and Capacity Optimization (CCO), involves monitoring the Performance Management (PM), Configuration Management (CM), and Failure Management (FM) data from the mobile network for specific problem patterns, compiling a new network configuration, and deploying it to the network. The goal is to achieve the operational objectives regarding a set of Key Performance Indicators (KPIs) that describe the desired mobile network's performance. With the introduction of SON, these use cases are implemented as SON functions which performs the steps automatically in closed-loop control. Consequently, all SON functions together are supposed to provide self-configuration, self-optimization, and self-healing capabilities for mobile networks.

The introduction of SON lifts traditional mobile network operations to SON operations as depicted in Figure 1.1. Instead of directly controlling the network, the operational personnel needs to configure and control the execution of SON functions. This is important since the SON functions affect the performance of the mobile network. SON operations comprises three tasks: First, *SON management* refers to the configuration of SON functions, particularly regarding network optimization. A SON function configuration usually comprises low-level parameters, e.g., thresholds on PM measurements or constraints for allowed network configurations, which con-

trol the automatic algorithm. Second, *SON coordination* refers to the control of the concurrent execution of SON functions. This is necessary due to the interactions between them resulting in negative network performance if they are uncontrolled executed. Third, *SON self-healing* refers to the identification and resolution of failures in the network. Configuring and coordinating a set of SON functions is not enough to ensure smooth operations as there can be unforeseen problems that affect the SON and, so, network performance.

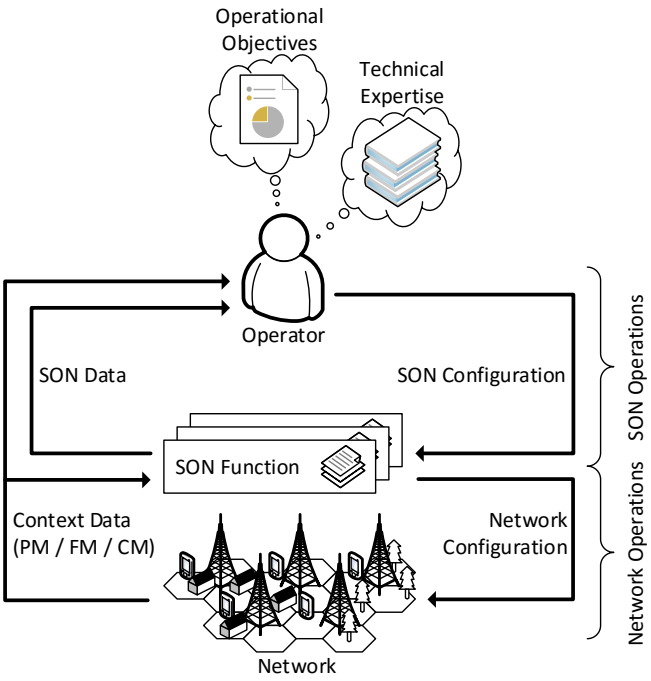


Figure 1.1: Overview of SON operations.

The goal of network as well as SON operations is to satisfy the mobile network's users with respect to their Quality of Service (QoS) requirements like data rates, network coverage, and voice quality. These user-related goals cannot be directly measured by the MNO so that they are mapped to several network-related KPIs which are directly measurable and allow QoS estimations. Consequently, the MNO derives KPI objectives from the QoS goals that drive the operation of the SON. Just as neither the network topology nor the user requirements are uniform in space and time, so are the KPI objectives usually not the same in the whole network and at all times. For instance, it may be important to provide high data rates in urban areas so that, consequently, the objective for the cell load, a KPI indicating the possible data rate, may be more challenging in urban areas than in rural area.

As shown in Figure 1.1, the human operator is the connecting link between the objectives and the SON. Specifically, they compile configurations for SON functions, coordination and self-healing policies that control the automatic algorithms to satisfy the objectives. Therefore, they rely on deep technical knowledge of the

SON system as well as the underlying network in order to determine the expected performance effects of a configuration in specific situations. However, operational personnel faces two main problems when doing this: First, the complexity of current networks makes is already now impossible to continuously adapt the configuration of a SON to changing objectives. Instead, MNOs often deploy a fixed, general configuration which produces acceptable performance in most scenarios. Second, MNOs actually only have limited knowledge of the SON. Specifically algorithms of SON functions are kept secret by the vendors of SON systems as part of their valuable intellectual property. This results in the practice that the SON is configured once by the vendor when it is rolled out according to the MNO's objectives at that time. If the objectives change later, however, this is seldom reflected in the SON configuration due to the risk of negative impacts. The growing complexity of mobile networks in the future, especially driven by HetNets, in terms of size, i.e., number of BSs in multiple layers, and necessary knowledge, i.e., multiple RATs and vendors, will even increase the required manual efforts. Due to the parallel pressure to reduce OPEX, it can be foreseen that MNOs will have the SON configured uniformly once during deployment in order get an acceptable performance with minimal efforts. This not optimal SON configuration, however, leaves a lot of optimization potential provided by the SON unused: On the one hand, SON functions may configure the network poorly leading to inferior network performance and the need for additional network extension, thereby, increasing CAPEX. On the other hand, the operator misses opportunities to diversify itself from competitors through, e.g., paramount network performance in specific hot spot areas, which may lead to missed revenues.

The manual efforts for SON operations originate from the fact that SON is designed as an *automatic* system in which fixed procedures are executed mechanically according to a low-level configuration. In contrast to that, an *autonomic* SON system would be designed to directly process the MNO's objectives and act accordingly. Therefore, we propose and describe Objective-Driven SON Operations (ODSO), an extension on top of the established SON approach that automates SON operations. The goal is to make a SON autonomic by automatically transforming a formalized model of the operational objectives of the MNO into a SON configuration driven by separate, formalized models that capture the required technical expertise. Thus, the efforts for the creation of the SON configuration are shifted from the human personnel to the computer. In this way, ODSO's objective model provides a single, uniform, high-level interface to operate a whole SON which reduces OPEX for SON operations. Additionally, the close and timely alignment of the SON to the operator objectives allows the MNO to lift its full optimization potential.

1.1 Problems and Objectives

This chapter describes the problems in SON operations that are the subject of this thesis in more detail and derives the objectives to be solved. Following that, we take a detailed look at the three task of SON operations and derive specific problems and objectives for each of them.

SON Operations

The current, automatic mode of SON operation introduces a manual gap between the MNO's objectives on the network performance and the SON configuration. In principle, this gap can be split into two specific problems that need to be solved: the *automation gap* and the *dynamics gap*.

The automation gap refers to the manual efforts to derive the SON configuration from the operational objectives based on personal technical expertise. Therefore, the personnel needs to understand the objectives, the network technology, and the SON system. This also involves handling a multitude of different configuration interfaces for SON management, SON coordination, and SON self-healing. Even more, since these SON operations tasks are handled separately, the operators have to perform a similar intellectual reasoning three times. In summary, the automation gap causes considerable manual efforts for SON operations.

The dynamics gap is related to the rate at which a SON needs to be reconfigured. Since the SON configuration mingles both the operational objectives and the technical expertise, the SON, potentially, needs to be reconfigured whenever either changes. Due to the fierce competition in the telecommunication sector, MNOs quickly need to adjust their focus to the needs of the customers in order to minimize churn. Even more, the need for ever higher data rates makes them introduce new technologies requiring new technical expertise at an increasing pace. However, the highest rate of change is caused by the fact that the objectives as well as the effects of the SON functions may depend on the operational context. For example, the importance of saving energy might change with every alternation of day and night. In summary, the dynamics gap causes the manual efforts caused by the automation gap to multiply with the rate of change.

Objective 1. *The concept of SON should evolve to be holistic and autonomic, i.e., SON management, SON coordination and SON self-healing should be integrated, and consistently controlled through a high-level interface in form of a formalized objective model that defines network-wide performance objectives. The objectives need to be automatically processed in order to configure and coordinate the SON functions as well as to control self-healing. The necessary technical expertise for this step should be provided in technical models that enable automatic reasoning. The technical and objective models should be separated in order to allow for an independent evolution of both. In this way, the manual gap of SON operations can be closed leading to less manual efforts and Operational Expenditure.*

SON Management

In order to configure the SON functions such that they optimize the network configuration to satisfy the operator objectives, the operational personnel requires an in-depth knowledge about the active SON functions and the properties of the mobile network. This is necessary in order to estimate and evaluate the network performance of the combination of the SON Function Configurations (SFCs). Whereas the network knowledge is available through years of experience with mobile networks

management, the SON function knowledge is difficult to gain for an MNO. The reason is that the vendors of SON functions invested huge amounts in research on the implemented algorithms and, thus, want to ensure that their intellectual property is kept secret in order to prevent others to simply copy them. As a result, the SON functions are typically seen as black-boxes by the operators. Furthermore, the functions are not independent of each other and might interact. Thus, it needs to be ensured that they are not configured inconsistently such that they work against each other. Hence, the SFCs, especially if they are provided by different vendors, need to be aligned in order to avoid negative performance. As a result of the difficulty and complexity of SON management, MNOs are inclined to not touch a running system and leave the SON configuration as initially installed. This way, however, they miss potential gains by an optimized SON configuration.

Objective 2. *The SON Function Configurations should be determined autonomically based on the SON operations-wide objective model and technical models as described in Objective 1. Thereby, the technical models need to provide information about the SON functions that enable to estimate their effects on network performance whilst allowing the vendors to keep their intellectual property secret. Furthermore, it needs to be automatically ensured that the SON functions are not configured in a way that they optimize the network inconsistently.*

SON Coordination

The concurrent execution of several SON functions may lead to inferior network performance if two SON functions negatively interfere with each other. These so-called SON function conflicts may even occur between SON functions running on different BSs. SON coordination aims to detect such conflicts and resolve them, typically by blocking one of the conflict partners, such that a conflict-free execution is enforced. As a result, some SON functions are not executed simultaneously but successively. In principle, a SON function is active if it identified a performance problem. Thus, a SON function conflict implies that there are two identified performance issues in the network that cannot be resolved at the same time. Instead, the conflict resolution needs to decide which problem to resolve first. This decision may depend on either technical constraints that require one specific execution order of the SON functions, or on the operational objectives of the MNO regarding which performance issue is more severe and should be resolved first.

Since the need for SON coordination has already been identified in the early stages of SON research [Sch+08], there are some proposals for conflict resolution. Typically, they are based on the assignment of priorities to the SON functions or a set of decision rules. These approaches, however, have three drawbacks: First, the priorities and rules mix up the technical constraints and operational objectives that led to their creation. Hence, in case the operational objectives change, it is not possible to identify the affected rules but instead, the whole conflict resolution policy needs to be recreated. Second, these approaches solely allow for fixed, abstract operational objectives since they neglect the actual network performance in their decision. Especially, they do not allow evaluating which network performance

issue is most severe with respect the MNO's objectives and, thus, which conflicting SON function promises the highest improvement in network performance. Third, the naive conflict pair-based selection may not make the best decision if there are complex, non-transitive conflicts spanning several network cells.

Objective 3. *The resolution of SON function conflicts in SON coordination should be autonomic as described in Objective 1. Thereby, the technical models describing the technical resolution constraints as well as the expected effects of SON functions (see Objective 2) need to be separated from the operations-wide objective model. The conflict resolution needs to adhere to the technical constraints while maximizing the expected improvement in network performance with respect to the Mobile Network Operator's objectives. Furthermore, it should ensure optimal decisions even in complex conflict situation.*

SON Self-Healing

Self-healing capabilities for mobile networks have been a requirement for SON from the beginning. Consequently, there has already been considerable research on concepts for the detection of failures in the network and for the diagnosis of their root causes. Especially probabilistic approaches show promising results, however, they are commonly missing two important aspects for comprehensive SON self-healing: the SON itself and degradation recovery.

SON self-healing is focusing solely on the mobile network, i.e., detecting and diagnosing failures in the hardware, software, and configuration of the BSs based on PM, CM, and FM data from the network. Thereby, they neglect an important source of failures in the network: the SON itself. That is, a SON function may act unexpected and, consequently, may cause failures in the network. This can be for two reasons: Obviously, it can be caused by a bug in the SON function algorithm, which is also just a piece of software. However, the erroneous activity may also be the consequence of an undetected network failure that a SON function encounters and cannot overcome. In other words, the SON function behavior is a symptom for a network problem. Neither is considered in state-of-the-art self-healing concepts.

In contrast to detection and diagnosis approaches, the concepts for the recovery of failures, i.e., the selection of suitable countermeasures, are rather sparsely researched. Either, this self-healing step is seen as a manual task performed by the operational personnel or simple rule systems are proposed. However, both approaches have severe shortcomings. The former renders self-healing into an open-loop, partially automated process which requires considerable manual effort by the MNO. Actually, this approach does also not fit to the automation vision of SON. The latter does only allow for simple decision making without considering root cause probabilities, the severity of a failure regarding the objectives, or the efforts for the recovery actions. Consider that a failure causes a minor but still acceptable degradation of the network performance. In this case, the MNO may be inclined to try some automatic recovery attempt, e.g., an automatic reboot, even if it has only a little chance of recovery, before assigning a human engineer to troubleshooting. The reason is that these actions are less costly than a manual inspection. However, if the

degradation is severe and the root cause of the failure is not clear then the MNO might actually prefer an immediate manual inspection in order to waste no time. This simple example already shows that autonomic decision making is important to the MNO.

Objective 4. *The objective regarding SON self-healing is twofold: First, it needs to take the SON as an additional source of failures and information into account. Second, the degradation recovery should be autonomic as described in Objective 1. Thereby, it is necessary to consider the diagnosed root causes, the effects of the actions described in the technical models, the SON-wide operational objectives regarding the network performance, and preferences regarding the recovery actions.*

1.2 Solution Approach and Contributions

This chapter gives an overview of the solution approach for achieving autonomic network operations with SON and outlines the specific contributions of this thesis to the state of the art. Similar to the previous chapter, this presentation is subsequently detailed out for each of the tasks of SON operations.

SON Operations

In order to achieve an autonomic mode of network operations with SON that closes the manual gap as required by Objective 1, we propose to tackle the problems from two sides: we develop an autonomic decision making concept and we integrate the three operational tasks into an ODSO architecture.

The generic, autonomic decision making concept is based on Multiattribute Utility Theory (MAUT). It considers two separate inputs: On the one hand, we define a formalized, SON-wide objective model that enables expressing context-dependent, prioritized and weighted target values for KPIs. This expressive model allows capturing the MNO's objectives accurately in order to automate decision making. On the other hand, the technical models contain the technical expertise for SON operations. In principle, they allow the determination of feasible actions and their effects in a specific operational situation. Due to the uncertainty in the behavior of mobile networks in their physical environment, the effects are probabilistic. For autonomic decision making, both models are combined and evaluated automatically at run time. This generic concept provides the following approaches solving Objective 2, Objective 3, and Objective 4 with a tool to implement autonomic behavior.

We integrate SON management, SON coordination, and SON self-healing into the ODSO architecture depicted in Figure 1.2. Since each task solves a separate problem, they all require their specific technical model to determine possible decision options. However, all of them are controlled by the very same general objective model that represents the MNO's objectives on the KPIs. Furthermore, the tight integration allows the components to share specific technical knowledge which makes the ODSO architecture more efficient. ODSO is not designed as an alternative to conventional,

automatic SON but instead it lies on top of the SON functions and instruments them. This becomes obvious by comparing Figure 1.1 with Figure 1.2.

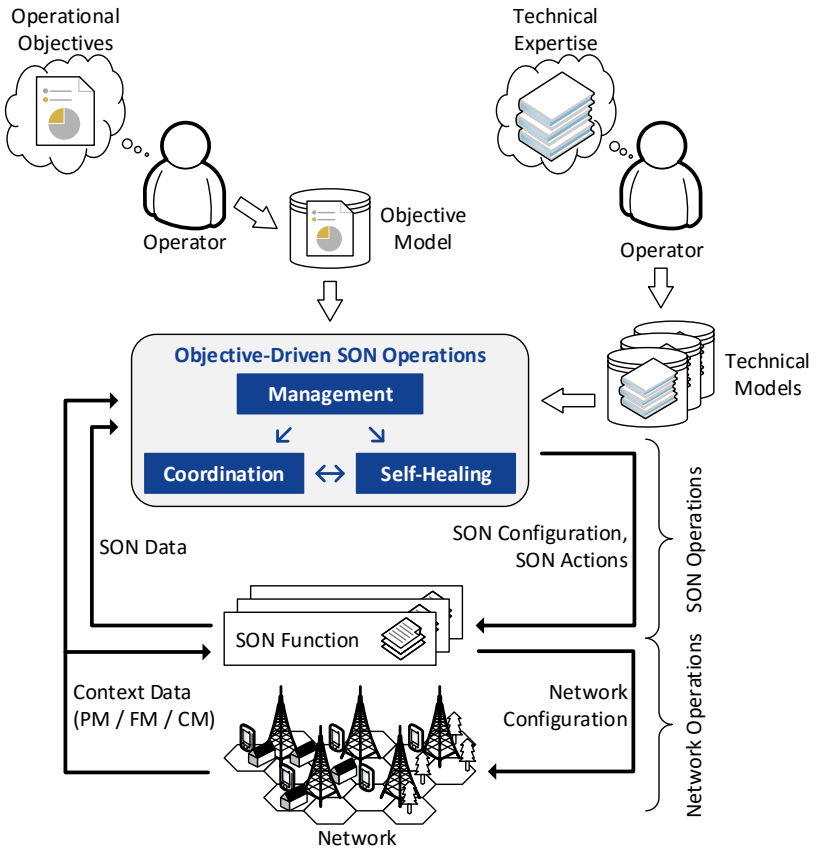


Figure 1.2: Overview of objective-driven SON Operations

Solution and Contribution 1. *We provide a generic concept for objective-driven, autonomic decision making in mobile networks operations. This includes the definition of a formalized, generally applicable objective-model. Furthermore, we propose the Objective-Driven SON Operations architecture that integrates SON management, SON coordination, and SON self-healing into a holistic, objective-driven, autonomic approach for network operations.*

SON Management

In order to satisfy Objective 2, the generic concept developed in Solution and Contribution 1 is applied to create an autonomic SON management concept. As outlined before, the decision making is based on the very same objective model that controls overall SON operations. However, the technical model is specifically defined for the task. The technical models for ODSO SON management are provided as SON function models. Each SON function model provides a description of the effects by a

specific SON function. In order to enable vendors keeping the internal algorithms closed, the SON function models solely provide a mapping from different possible configurations of the function to expected, probabilistic effects on KPIs. The definition of a generic, formalized semantics of the model allows autonomic management of diverse SON functions.

The ODSO SON management component combines the diverse SON function models and determines an overall SON configuration, i.e., a configuration for every SON function. Thereby, possible SON configurations are checked for incompatible, conflicting optimization goals based on a probabilistic approach. A similar process is performed to determine the overall expected performance of a SON configuration in order to evaluate it against the MNO's objectives.

Solution and Contribution 2. *We present an autonomic approach for SON management which is based on the Objective-Driven SON Operations (ODSO) concept and integrated into the ODSO architecture. Based on a set of SON function models, one for each SON function, it determines a SON configuration that contains only compatible SON Function Configurations and is expected to maximize the satisfaction of the overall operational objectives by the Mobile Network Operator.*

SON Coordination

In order to provide autonomic conflict resolution for SON coordination as required by Objective 3, the generic concept developed in Solution and Contribution 1 is applied. The decision, which SON functions should be blocked, is driven by the overall objective model as well as a technical model containing technical constraints. Conflict resolution first determines the expected performance improvement that the execution of a conflicting SON function promises based on the expected performance of the current SON configuration (see Solution and Contribution 2). Second, the performance improvement is evaluated against the objective model.

In order to execute a set of conflict-free SON functions that maximizes the satisfaction of the operator objectives even in complex conflict situations, we formulate a Goal Programming (GP) problem (see [JT10]) and utilize an off-the-self solver for finding a guaranteed optimal solution.

Solution and Contribution 3. *We present an autonomic conflict resolution approach for SON coordination. It is based on the Objective-Driven SON Operations (ODSO) decision making concept and integrated into the ODSO architecture which enables sharing information with ODSO SON management. The introduced concepts resolves the conflicts by, first, enforcing a set of technical constraints and, second, blocking SON functions with a small improvement of the mobile network performance with respect to the overall ODSO objective model. By relying on the well-founded Goal Programming approach, it is ensured that the best set of SON functions is selected even in complex situations.*

SON Self-Healing

We propose a design for SON self-healing that aims at Objective 4. It solves two main issues: involving the SON in self-healing and autonomic degradation recovery.

We outline how the SON in general and the SON functions in particular can be integrated into failure detection. Therefore, we present two orthogonal approaches: First, new SON functions may be extended with the ability to raise alarms that notify SON self-healing about some internal problem they encountered during their operation. Second, the activity of the SON functions is monitored for abnormal behavior which might be an indication for an error of the SON function or a network failure that cannot be resolved. Both provide SON self-healing with an additional source of information for detecting new types of failures and to increase the accuracy of failure diagnosis. Thereby, these approaches introduce only little overhead.

We present an autonomic degradation recovery based on common probabilistic diagnosis approaches and the generic concept developed in Solution and Contribution 1. The recovery knowledge about countermeasures for diagnosed failure root causes is encoded in a technical model referred to as recovery model. By using information about the expected performance of the current SON configuration (see Solution and Contribution 2), possible recovery actions for a failure can be determined and their probabilistic effects considering the root cause probabilities computed. Finally, these effects are evaluated against the common objective model as well as a special recovery cost model that represents the MNO's preferences regarding the execution of an action.

Solution and Contribution 4. *We present a comprehensive SON self-healing approach that extends the state of the art in two ways: first, we improve degradation detection by adding SON function monitoring capabilities and, second, we provide a sophisticated degradation recovery concept. The former enables the detection of new failures and increases the accuracy of root cause diagnosis. The latter introduces an autonomic degradation recovery for SON self-healing that is based on the Objective-Driven SON Operations (ODSO) decision making concept and integrated into the ODSO architecture which enables sharing information with ODSO SON management. Thereby, the decision making is based on an evaluation of the effects of the recovery actions with respect to the overall ODSO objective model and the probabilistic failure diagnosis, as well as the preferences of the Mobile Network Operator regarding the actions. In order to represent the latter, a cost model is introduced.*

1.3 Outline

The logical structure of this thesis is presented in Figure 1.3.

Chapter 1 Introduction motivates the research conducted for this thesis. Furthermore, it presents an overview of the problems in SON operations and derives the objectives for this thesis. Finally, the chapter sketches out how we aim to solve the objectives and summarizes the contributions of our research.

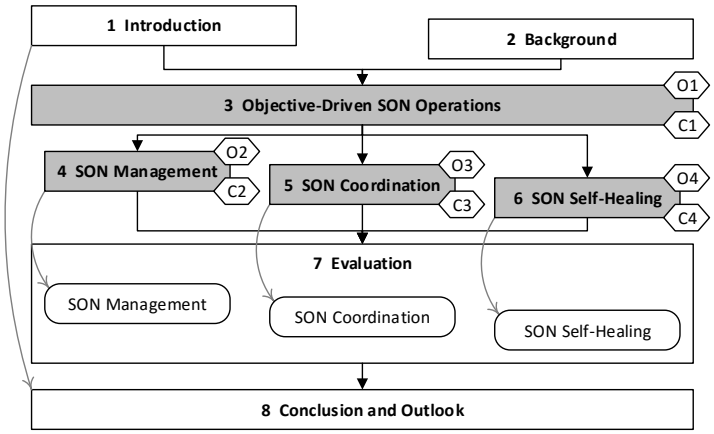


Figure 1.3: Outline of this thesis (the Ox or Cy show that the chapter covers Objective x or Solution and Contribution y respectively).

Chapter 2 Background provides an introduction into the problem domain, i.e., mobile networks and SON. Furthermore, it gives a general overview of the theories that will play a major role in developing the solution, namely the concept of Policy-Based Management (PBM) and MAUT.

Chapter 3 Objective-Driven SON Operations presents the ODSO architecture and the generic design of the autonomic components. Therefore, it first presents the manual gap of SON operations that led to Objective 1 in more details, and then explains our Solution and Contribution 1. Hence, it lays the foundations for the following three chapters. Finally, it compares the results to related work.

Chapter 4 SON Management presents the SON management component of the ODSO architecture. Therefore, it first presents the need for autonomic SON management that led to Objective 2 in more details, and then describes our Solution and Contribution 2. Finally, it compares the results to related work.

Chapter 5 SON Coordination presents the SON coordination component of the ODSO architecture. Starting with a detailed presentation of the need for autonomic conflict resolution that led to Objective 3, it explains our Solution and Contribution 3. Finally, it compares the results to related work.

Chapter 6 SON Self-Healing finally presents the SON self-healing component of the ODSO architecture: First, it outlines the need for the involvement of SON and an autonomic degradation recovery that led to Objective 4 in more details. Second, it describes the solutions for the two problems that make up our Solution and Contribution 4. Finally, it compares the results to related work.

Chapter 7 Evaluation presents an evaluation of the ODSO architecture. Concretely, it introduces the simulation system that we use to simulate a realistic mobile network. Based on this, it presents four scenarios, one for SON management, one for SON coordination, and two for SON self-healing, that demonstrate the specific advantages of ODSO.

Chapter 8 Conclusion and Outlook summarizes the thesis by recapitulating the objectives for ODSO and how our contributions solved them. Finally, we give an outlook on emerging and desirable future developments for SON operations.

The black arrows in Figure 1.3 show the general flow of argument in the thesis and provide a recommended reading order. Consequently, starting with this chapter, the reader may skip Chapter 2 if he or she is already familiar with the presented topics. The main part of the thesis starts with Chapter 3. After that, Chapter 4, Chapter 5, and Chapter 6 may be read in any order since the presented components are only remotely connected. Chapter 7 then presents the combined evaluation of the concepts for the three ODSO components. As the gray arrows indicate, it is, thus, also possible to read each component chapter and the respective evaluation section together. Finally, Chapter 8 should be read in the end. For a brief summary of the thesis, it is also possible to read Chapter 1 and Chapter 8, as the gray arrow indicates.

1.4 Publications

Some parts of the approach and results in this thesis have already been presented in a number of other publications. The following listing shows all publications by the author, together with a short outline of the contribution, which have been written during the creation of this thesis. Some of them are not related to the topic of this thesis, though. Ideas and result produced by the author and published in one of these publications are not additionally cited in this thesis.

Scientific Publications

1. Bernhard Bauer et al. "Resource-oriented Consistency Analysis of Engineering Processes". In: *Proceedings of the 14th International Conference on Enterprise Information Systems*. SciTePress, June 2012, pp. 206–211 [Bau+12]
   The author of this thesis, together with two follow researchers, developed an ontology-based mapping approach that enables detecting insufficient resources for the execution of business processes. The results have not been integrated into this thesis.

2. Christoph Frenzel, Henning Sanneck, and Bernhard Bauer. "Automated rational recovery selection for self-healing in mobile networks". In: *2012 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, Aug. 2012, pp. 41–45 [FSB12]
   The author of this thesis describes a concept for objective-driven degradation

recovery for self-healing in SON based on a probabilistic root cause diagnosis and simplified operator objectives. This work is integrated into Chapter 6.

3. Christoph Frenzel, Henning Sanneck, and Bernhard Bauer. "Rational Policy System for Network Management". In: *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, May 2013, pp. 776–779 [FSB13b]
   The author of this thesis describes an extension of the previously introduced objective-driven degradation recovery concept [FSB12] for general network management, specifically coordination of network configuration actions. This work is integrated into Chapter 3 and Chapter 5.

4. Christoph Frenzel, Henning Sanneck, and Bernhard Bauer. "A Fuzzy, Utility-Based Approach for Proactive Policy-Based Management". In: *Proceedings of the 7th International Symposium on Theory, Practice, and Applications of Rules on the Web - 7th International Symposium, RuleML 2013, Seattle, WA, USA, July 11-13, 2013*. Springer, July 2013, pp. 84–98 [FSB13a]
   The author of this thesis describes the implementation of the previously introduced objective-driven network management concept [FSB13b] as a fuzzy inference system. It enables fuzzy system states for the objectives and fuzzy context. This work is integrated into Chapter 3.

5. Christoph Frenzel, Simon Lohmüller, and Lars Christoph Schmelz. "Dynamic, context-specific SON management driven by operator objectives". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, May 2014, pp. 1–8 [FLS14a]
   The author of this thesis, together with two fellow researchers, identifies and analyzes the manual gap of SON management, and presents the SON objective manager approach with simple, binary objectives and deterministic action effects. This work is integrated into Chapter 3 and Chapter 4.

6. Lars Christoph Schmelz et al. "SON management demonstrator". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, May 2014, pp. 1–2 [Sch+14b]
   The author of this thesis, together with several fellow researchers, presents a demonstrator implementation showing the SON objective manager concept described in [Sch+14a]. The results have not been integrated into this thesis.

7. Christoph Frenzel et al. "Detection and resolution of ineffective function behavior in Self-Organizing Networks". In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, June 2014, pp. 1–3 [Fre+14]
   The author of this thesis, together with a fellow researcher, identifies and analyzes the problem of ineffective SON functions and presents a SON self-healing approach comprising a problem detection by monitoring the SON and a problem mitigation by triggering specific SON functions. This work is integrated into Chapter 6.

8. Christoph Frenzel, Simon Lohmüller, and Lars Christoph Schmelz. "SON management based on weighted objectives and combined SON Function models". In: *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. IEEE, Aug. 2014, pp. 149–153 [FLS14b]
   The author of this thesis, together with two fellow researchers, presents an extension of the previously introduced SON objective manager [FLS14a] with objectives consisting of KPI value ranges and weights as well as a run-time decision making as presented in this thesis. This work is integrated into Chapter 3 and Chapter 4.

9. Christoph Frenzel et al. "Operational Troubleshooting-Enabled Coordination in Self-Organizing Networks". In: *Mobile Networks and Management - 6th International Conference, MONAMI 2014, Würzburg, Germany, September 22-24, 2014, Revised Selected Papers*. Springer, Feb. 2015, pp. 149–162 [Fre+15b]
   The author of this thesis, together with a fellow researcher, presents an extension of the previously introduced SON self-healing approach [Fre+14] that also uses SON functions as problem detection probes. This work is integrated into Chapter 6.

10. Simon Lohmüller et al. "Policy-Based SON Management Demonstrator". In: *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE, May 2015, pp. 1–2 [Loh+15]
    The author of this thesis, together with several fellow researchers, presents an extended demonstrator based on [Sch+14b]. The results have not been integrated into this thesis.

11. Christoph Frenzel et al. "Objective-driven coordination in self-organizing networks". In: *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, Aug. 2015, pp. 1453–1458 [Fre+15a]
    The author of this thesis describes an approach for objective-driven SON coordination using probabilistic effects and utility function-based objectives. This work is integrated into Chapter 3 and Chapter 5.

12. Tsvetko Tsvetkov et al. "A Constraint Optimization-Based Resolution of Verification Collisions in Self-Organizing Networks". In: *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2015 [Tsv+15]
    The author of this thesis, together with a fellow researcher, presents a constraint programming-based approach to determine a schedule to revert possibly erroneous network configuration changes. The results have not been integrated into this thesis.

Book Chapters

1. Christoph Frenzel, Henning Sanneck, and Seppo Hämäläinen. "Future Research Topics". In: *LTE Self-Organising Networks (SON)*. ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley &

Sons, Ltd, Dec. 2011. Chap. 11, pp. 379–390 [FSH11]
The author of this thesis describes future research topics for SON, specifically the adoption of autonomic principles to form cognitive radio networks that performing ration decision making based on machine-learned models and controlled by complex, utility function-based operator objectives. This work is integrated into Chapter 3 and Chapter 8.2.

Patent Applications

1. Henning Sanneck, Péter Szilágyi, and Christoph Frenzel. *Sub-cell Level, Multi-layer Degradation Detection, Diagnosis and Recovery.* WIPO Pub. No. WO 2013/143572 A1. Oct. 2013 [SSF13]
   The author of this thesis, together with two fellow researchers, presents a device to detect and localize network problems with sub-cell-level granularity based on data from several cell layers in a HetNet. The results have not been integrated into this thesis.

2. Henning Sanneck, Péter Szilágyi, and Christoph Frenzel. *Self Organizing Network Operation Diagnosis Function.* WIPO Pub. No. WO 2014/023347 A1. Feb. 2014 [SSF14]
   The author of this thesis, together with two fellow researchers, presents a device to detect and mitigate problems in a SON. This work is integrated into Chapter 6.

3. Lars Christoph Schmelz, Christoph Frenzel, and Simon Lohmüller. *Network Entity and Method for Controlling a SON-Function.* WIPO Pub. No. WO 2014/191469 A1. Dec. 2014 [SFL14]
   The author of this thesis, together with two fellow researchers, presents a device for objective-driven SON management based on SON function models and operator objectives. This work is integrated into Chapter 3 and Chapter 4.

Project Deliverables

The following deliverables have been developed within the Celtic project COgnitive network ManageMent under UNcErtainty (COMMUNE) [COM11] and the European Union (EU) FP7 project SElf-MAnagement FOr Unified heterogeneous Radio access networks (SEMAFOUR) [SEM12].

1. Enda Barrett et al. *Specification of cognitive network management framework and functions.* Deliverable D3.1. COMMUNE Project, Nov. 2013 [Bar+13a]
   The author of this thesis outlines the idea of objective-driven SON self-healing based on a knowledge model about the action effects and operator objectives. This work is integrated into Chapter 6.

2. Enda Barrett et al. *Specification of knowledge-based reasoning algorithms Project.* Deliverable D4.1. COMMUNE Project, Dec. 2013 [Bar+13b]

The author of this thesis presents a case-based reasoning algorithm for diagnosis and recovery in SON self-healing based on the work by [SN12]. This work is integrated into Chapter 6.

3. Sana Ben Jemaa et al. *Integrated SON Management - Requirements and Basic Concepts.* Deliverable D5.1. SEMAFOUR Project, Dec. 2013 [Ben+13b]
   The author of this thesis, together with several fellow researchers, presents the preliminary idea and architecture for objective-driven SON management. This work is integrated into Chapter 3 and Chapter 4.

4. Lars Christoph Schmelz et al. *Integrated SON Management - Policy Transformation and Operational SON Coordination (first results).* Deliverable D5.2. SEMAFOUR Project, June 2014 [Sch+14a]
   The author of this thesis, together with several fellow researchers, presents an implementation for objective-driven SON management [Ben+13b] based on the work presented in [FLS14a]. This work is integrated into Chapter 3 and Chapter 4.

5. Luis Miguel Campoy Cervera et al. *Integrated SON Management Implementation Recommendations.* Deliverable D5.4. SEMAFOUR Project, Aug. 2015 [Cam+15]
   The author of this thesis, together with a fellow researcher, presents the results of an analysis of the network management processes of a MNO. This work is integrated into Chapter 3 and Chapter 4.

2 Background

This chapter aims at providing an introduction to the technologies and theories that will be covered in this thesis. It focuses on the information which is necessary to understand the problems and solution approaches presented later. Specifically, we provide an overview of SON, PBM, and decision theory. Readers that are comfortable with these topics may skip this chapter.

2.1 Self-Organizing Networks

SON is a concept for the automation of mobile network operations in order to reduce the costs for providing users with telecommunication services, e.g., telephone calls or data, using radio signals. It was introduced by the NGMN in 2007 [NGM07a][NGM07b] in order to motivate research and industry to address the challenges in network operations due to the foreseen complexity increase in terms of multiple technologies, specifically LTE. This chapter provides an overview of the technology of modern mobile networks as well as an introduction to the SON concept. Most of the content is based on the basic work [HSS11].

2.1.1 Mobile Networks

Although SON has been introduced to face the operational burden of the new 3GPP LTE networks, the concepts are not tied to it. Instead, the focus of SON lies on the radio access domain of a mobile network in general. Therefore, we describe the general structure of a 3GPP mobile network with the objective to provide enough information about Radio Access Networks (RANs) to understand the basic principles of SON. Note that all examples and evaluation scenarios (see Chapter 7.2) in this thesis are considering an LTE network.

3GPP [3GP16, "About 3GPP"] is a standardization project that unites the most important telecommunications standard organizations as well as numerous vendors and MNOs in order to develop worldwide specifications for mobile network technologies. Today, it is the most important standardization body and the specified technologies are the most common in the world. A 3GPP network consists of three main parts [3GP15e]: the User Equipments (UEs), e.g., mobile phones, the RAN consisting of BSs that provide UEs with radio access, and a core network that connects the BSs among each other and to other networks, e.g., public land line, mobile networks of other MNOs, or the Internet.

A mobile network can consist of several RANs that differ in the RAT for communicating with the UEs. Most prominently are

- GSM EDGE Radio Access Network (GERAN) [3GP15a] that is part of the Second Generation (2G) standard Global System for Mobile Communications (GSM) [Ebe+09],
- UMTS Terrestrial Radio Access Network (UTRAN) [3GP15d] that is part of the 3G standard UMTS [HT10], and
- Evolved Universal Terrestrial Radio Access Network (E-UTRAN) [3GP15b] that is part of the 4G standard LTE [HT11][STB11].

This list of RATs may be extended in the future as work on the upcoming 5G standard is currently underway [Dah+14][Nok14][NGM15].

Independent of the concrete RAT, a BS in a RAN is connected to an antenna which spans up an area of radio coverage called network cell. Hence, mobile networks are often also referred to as cellular networks. A network cell is the area in which the UEs may connect to this antenna since the perceived radio signal is good. However, a cell is limited in two dimensions: On the one hand, the coverage area of a network cell is limited. Hence, the RAN consists of numerous BSs and antennas that, together, are able to cover a huge area like a whole country. Thereby, the MNOs typically employ directional antennas and connect three of them to one BS in order to cover 360° as depicted in Figure 2.1. This common setup is called 3-sector site. On the other hand, the capacity of a cell is limited, i.e., there is an upper bound for the achievable throughput within one cell. A RAN typically consists of macro cells with a big coverage area that provide general network access. However, in specific areas with huge traffic amounts, it is possible to deploy additional small cells with a small coverage area that aim to provide further capacity. According to the size of the cell, these are often called micro, pico, and femto cells. The combination of different RANs with different RATs and different layers of cell types with respect to their size into one mobile network is referred to as HetNets [Sar+11] and shown in Figure 2.2.

Due to the erratic layouts of the landscapes and cities in particular, the layout of mobile networks in terms of the BSs' positions is also irregular. Furthermore, the environment has a strong effect on the propagation properties of radio waves and, thus, the network cells may show very different properties in terms of their, e.g., size, signal quality, and user population. Figure 2.1 shows this in the different cells sizes and the fuzzy, uneven borders between cells. Consequently, the operational personnel typically need to apply a specific configuration for each and every cell such that the network as a whole provides the best performance. In order to do that, the operational personnel requires information about the performance, healthiness, and configuration of the whole network down to each and every network cell. PM data provides an indication of different performance aspects of the network like the signal quality, the handover performance, or the capacity. Typically, it comprises a huge number of measurements and KPIs [3GP15g], e.g., Channel Quality Indicator (CQI), handover ping-pong rate, or cell load. FM data enables the detection of failures in the network induced by faulty hardware or software. It consists of well-defined

Figure 2.1: Exemplary LTE mobile network with mostly 3-sector BSs taken from the simulation system presented in Chapter 7.2.

alarms that indicate some unexpected outage or behavior. CM data comprises the configuration of the BSs. This information may be stored in and retrieved from a central database for network operations purposes. However, if the BS provides automatic mechanisms to adapt the configuration, the centralized information needs to be updated.

PM data is continuously generated by the BSs and, so, may produce huge data amounts. Constant streaming of this data to a central operations center would occupy a significant amount of the bandwidth in the core network which would not be available for user data anymore. Hence, PM data is not continuously provided as a stream but in chunks at regular time intervals referred to as granularity periods [3GP15f]. During a granularity period the measurement counters are maintained

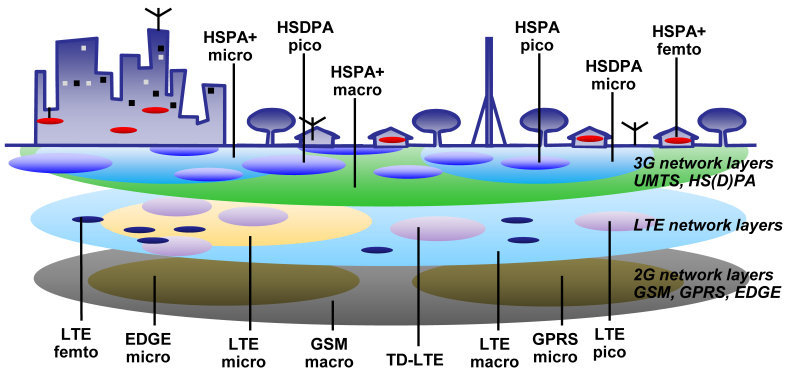


Figure 2.2: Overview of a HetNet [Sch12].

in the BS and at its end the measurements are aggregated to KPIs and sent to the central operations system. This periodic PM data provisioning affects network operations as it defines the minimum reaction time to a performance incident.

2.1.2 Drivers for SON

Due to the complexity of mobile networks, driven by the huge number of network elements, the interdependencies between them, and the increasing number of RATs that need to by integrated, network operations is a very complex task. It is typically based on a centralized OAM architecture. Thereby, the configuration, optimization, and troubleshooting of the network elements, i.e., BSs in the RAN, is performed using specialized tools. These are typically semi-automated and require tight human supervision resulting in network operations being time-consuming, error-prone, and expensive due to the required high degree of expertise [Wal+11][LWN07].

In principle, network operations is an extensive open-loop control system: First, the PM, FM, and CM is collected. Second, the different data streams are integrated into a comprehensive picture in order to analyze the network for performance issues and failures. Third, a new or better configuration is determined and deployed to the network elements. Although this process is guided by operational procedures and supported by planning and optimization tools, the workflow is challenging and strongly depends on human experience.

The vision of SON is to increase the automation in network operations by closing the control loop of network operations. It is motivated by a number of technical and business drivers [Kür+10]: On the one hand, SON can be seen as a natural evolution of the automation in low-level radio resource management towards network management. Furthermore, manual operations of ever more complex networks leads to more errors and inconsistencies in the network configuration that directly affect the users. Finally, the necessity to optimize the network in real time, in order to quickly react to changes in the environment and user behavior, makes manual operations impracticable. On the other hand, the pressure on MNOs to reduce their costs is constantly increasing as the revenue increase does not follow the traffic increase. This is exacerbated by the complexity introduced by new RAT technologies, e.g., the operational complexity of a 5G network is expected to increase 53 to 67 times compared to 4G [IZ14]. Hence, "[i]t is obvious to most that SON should provide a productivity benefit to the operator by increasing the productivity of staff or reducing the number of staff required to do a specific job." [Wal+11, p. 68] Furthermore, as a SON-enabled network may perform closer to the optimum, an MNO might be able to postpone network extensions, e.g., deploying new small cells. Hence, SON may provide cost savings regarding OPEX and CAPEX [Kür+10][RAH11]. However, the higher QoS in a mobile network might also increase the revenue of the MNO due to reduced churn and higher attraction of new customers [RAH11].

2.1.3 SON Use Cases

In order to make the MNO's visions and requirements for SON concrete, the NGMN published a list and description of operator use cases that a SON should perform in 2007 [NGM07a][NGM07b]. It categorized the use cases into planning, deployment, optimization, and maintenance which included fault management. The driver of this effort was the ongoing specification of the 4G technology LTE which was expected to considerably increase the complexity of network management. In a successive white paper published 2008 [NGM08], the use cases where reorganized and presented in today's common categorization along the key OAM areas: self-configuration, self-optimization, and self-healing.

Based on this work, 3GPP started the standardization of SON in 2008 with the goal to transition from open-loop network operations to closed-loop operations [3GP14b]. Thereby, it targeted those NGMN use cases that require standardization, i.e., the ones that require an information exchange between network elements by different vendors or some other vendor-independent treatment. For each new release of the standards, 3GPP aimed to specify of set of use cases by defining their functionality as well as their management interfaces. It started with self-configuration and the definition of, e.g., Automatic Neighbor Relationship setup (ANR) and automatic Physical Cell ID (PCI) management which both are important for smooth handover execution between networks cells [San+11]. The next step was the definition of self-optimization algorithms like Mobility Robustness Optimization (MRO) which reduces erroneous handovers and Mobility Load Balancing (MLB) which aims to provide higher throughput for users by balancing the load between neighboring network cells [Las+11]. Finally, 3GPP also specified some self-healing functionality like Cell Outage Detection (COD) which detects errors and Cell Outage Compensation (COC) which compensates a failed network cells using its neighbors [Nov+11]. The reader may refer to [Ali+13] for a survey of SON from an academic point of view and to [Jor+14] for a summarizing overview of the current and future 3GPP-related standardization.

The result of the use case driven development is that the term SON in the area of mobile networks does not refer to a specific technology of technique that is used. Instead, it is generally understood as the idea to automate operations, and, in particular, it refers to the automation of the set of use cases defined by NGMN and 3GPP. Notably, it is not the principal focus of SON to create some emergent behavior as it is considered in research on self-organization in general [GH03], although these concepts are often stated to as a basis of SON.

2.1.4 SON Functions

The concrete implementation of a SON use case is referred to as SON function. Consequently, a SON function has a specific goal, given by the implemented use case, which it wants to achieve. With respect to self-optimization, the goal is typically defined in terms of KPI values. For instance, an MLB SON function may aim to balance the KPI cell load among a set of network cells. In general, a SON function can be seen as a control or feedback loop [Sch+08][Ban13][3GP12] as depicted in

Figure 2.3. It continuously monitors the performance data provided by the network, i.e., PM, FM, and CM data, for triggering situations that would require its intervention, e.g., an MLB function could be triggered if the load of a cell exceeds a given threshold. In that case, it implements complex algorithms to compute the necessary reconfiguration of the network cells in order to resolve the triggering situation, e.g., adapting the Cell Individual Offset (CIO) in order to hand over some UEs to neighboring cells. Finally, the actions are executed.

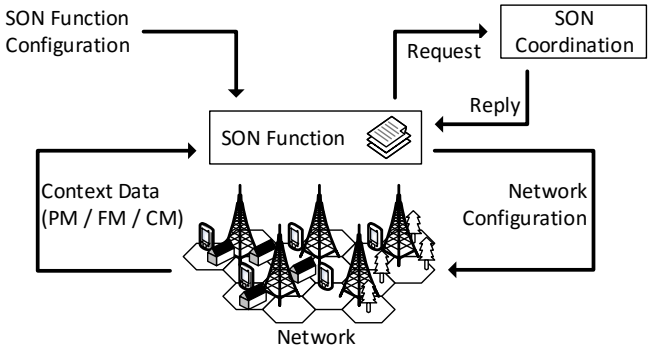


Figure 2.3: Conceptual view on a SON function as a coordinated feedback loop.

The algorithms for detecting the triggering conditions and computing reconfigurations are subject of active research. Thereby, some use cases can be implemented with deterministic rule systems, e.g., the self-optimization functions MRO and MLB, whereas others require more sophisticated approaches including techniques from machine learning, e.g., the self-healing function COD. The reader may refer to [HSS11] and [RH12] for a comprehensive overview of the different approaches.

Although the implementation approaches might be quite different, all SON function algorithms have control parameters that need to be configured by the operator, e.g., by setting thresholds on KPIs for monitoring or defining parameters of the reconfiguration computation. This configuration defines the goal of a SON function and determines how the function proceeds to achieve it. As this configuration is essential for a SON function, 3GPP defined possible parameters, referred to as targets [3GP12], for the configuration of some of their specified SON functions. However, these are solely a part of a concrete configuration of a SON function as 3GPP aims to avoid enforcing specific algorithms or types of algorithms. Therefore, these targets are general definitions of triggering conditions for SON functions and have limited relevance in research.

The execution of a SON function can follow three different architectures [Sch+08]. In a distributed architecture, an individual instance of a SON function is running on each BS. This solution, in principle, scales very well with the size of the network, and may allow a fast reaction by the SON functions below the granularity period at which the network performance data is provided to the central operations center. However, this comes at the cost of a limited view of each SON function over solely the network cells of the BS and maybe some neighbors. The centralized architecture assumes

that each SON function is executed on a central node, e.g., in the central operations center. Thus, a SON function instance can access and analyze performance data from the whole network, however, only at the rate of the granularity periods. Finally, the hybrid architecture combines both former architectures as some SON functions are executed in a distributed fashions whereas other run centrally.

2.1.5 SON Coordination

It has been identified from the very beginning of research on SON that parallel changes of the network configuration by concurrently acting SON function may lead to conflicts [Sch+08][Sch+11][Cha11][3GP12]. Thereby, a conflict between SON functions refers to "negative interactions" [Ban+11b, p. 322] which "may decrease the network system performance" [Ban+11b, p. 323]. In order to provide smooth operation of the network, SON functions need to be coordinated such that conflicts are avoided. This alignment of the SON functions becomes particularly important if SON functions by multiple vendors are active in one network [NGM14]. The importance of SON coordination as an operational aspect led to the situation, that sometimes it is used interchangeably with SON operations, e.g., in [Ban+11b].

There are different classes of SON function conflicts, e.g., if two SON functions affect the same network configuration parameter or if they change different parameters which, though, have a similar effect [Ban13]. To exemplify this, Figure 2.4 shows the results of a study about SON functions and the network configuration parameters they change. As can be seen, there is a considerable overlap which may lead to conflicts. However, the occurrence of a SON function conflict also depends on spatial and temporal aspects of the involved configuration changes by the SON functions, i.e., which network cells are when affected.

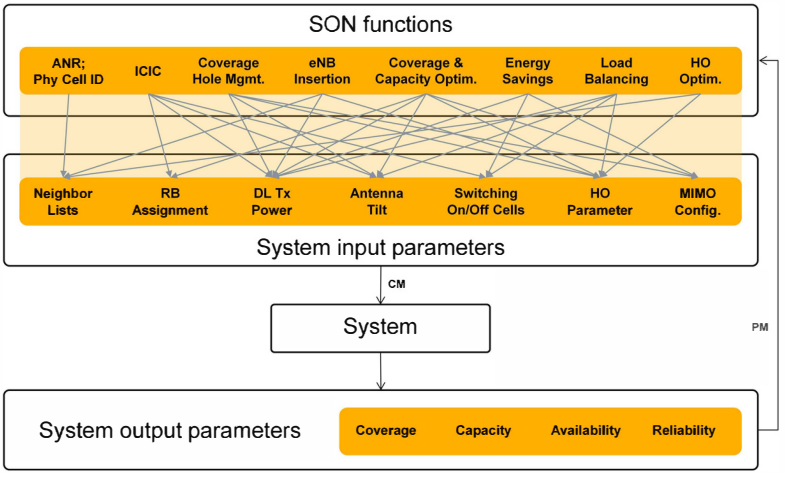


Figure 2.4: Potential control parameter conflicts [Ban+11a].

There are two general approaches to SON coordination that we refer to as off-line and on-line coordination [Ban13]. The former aims to prevent the occurrence of conflicts by adjusting the behavior of SON functions such that they avoid conflicting

actions. This can be achieved in two ways: on the one hand, by enforcing specific constraints on the design of SON functions or develop them together, referred to, e.g., "co-design" [Ban13, p. 90], "coordination by information" [GBK11, p. 1] or "design principles" [Alt+14, p. 455]; on the other hand, by adjusting the configuration of the SON functions prior to their deployment, referred to, e.g., "heading harmonization" [Sch+11, p. 195] or "fixed measurement intervals for separation" [GBK11, p. 4] as exemplified in [KK13].

However, since it is almost impossible to avoid all conflicts off-line, an on-line coordination approach aims to handle occurring SON function conflicts at run time. The most common approach for this is pre-action coordination[1] which requires the SON functions to request changes of the network configuration at the SON coordinator *prior* to their execution as depicted in Figure 2.3. The coordinator can detect possible conflicts among the requests and resolve them by, e.g., rejecting the respective changes. Thereby the detection may be based on predefined, human operator-provided knowledge [Sch+11][Ban13][RT11] or learned by using machine-learning techniques [Iac+15]. The resolution of conflicts is often based on a policy (see Chapter 5.5) and may block or delay the execution of some SON function [Sch+11][Ban13][3GP13], or merge the requests to some combined new network configuration [Sch+11][Iac+14b]. Another approach for SON coordination is post-action coordination[2] which aims at detecting negative network performance due to conflicts *after* the SON functions have changed the configuration of the network and handling them by, e.g., triggering countermeasures [Sch+11] or reverting the changes [Tsv+14][AT16]. Whereas pre-action coordination may delay too many optimization attempts by blocking them, post-action coordination may cause inferior network performance for some time before conflicting changes can be reverted. The reader may refer to [Tsa+13][Ben+13c][LIA13] for a more comprehensive overview of the approaches.

2.2 Policy-Based Management

Managing complex systems like mobile networks is a challenging task that cannot be performed without automation. One way of achieving such automation is to cast specific management requirements into software and execute it. However, this would lead to an inflexible OAM system since accompanying new requirements requires rewriting the software. PBM is "a management paradigm that separates the rules governing the behavior of a system from its functionality" [BA07, p. 447]. The roots of PBM go back to the 1960's where it was used for ensuring security in mainframes. Later, it was also adopted by the networking community leading to, e.g., Policy-Based Network Management (PBNM) [Str03][3]. The basic idea of PBM is that a system's basic algorithmic decisions, i.e., its behavior, is controlled by a system-external *policy* that can be flexibly adapted by the system operator.

[1] In the majority of literature, this is in general referred to as conflict detection and resolution. [3GP13], however, denotes this approach conflict detection and prevention.

[2] [3GP13] refers to this as conflict resolution.

[3] see [BA07] for a comprehensive historical overview of PBM

In 2001, IBM coined the term Autonomic Computing (AC) as a vision of "computing systems that can manage themselves given high-level objectives from administrators" [KC03, p. 41]. The term was inspired by the human autonomic nervous system which controls, e.g., the heart beat and the body temperature, without conscious thinking of a person. In the same way, computer systems should control their function themselves. This self-management involves self-configuration, self-optimization, self-healing, and self-protection, commonly referred to as self-x aspects [KC03]. These activities are performed in a Monitor-Analyze-Plan-Execute (MAPE) loop as shown in Figure 2.5, which is controlled by a policy that defines the objectives and constraints of the AC system.

AC allows to lift system management from automation to autonomic behavior. [Mov+12] defines the former as the "ability of performing one or more tasks without any manual intervention or external help. It does not include the performance optimization issues to better fit some sort of performance goals" [Mov+12, p. 465]; and the latter as "self-managing given a set of high-level objectives from administrators. High-level objectives define for a system what are its goals and the system attempts to accomplish them in the best manner. Consequently, an autonomic system is able to monitor its own performance and adapt itself accordingly, optimize its use of resources and overcome occurred events" [Mov+12, p. 465].

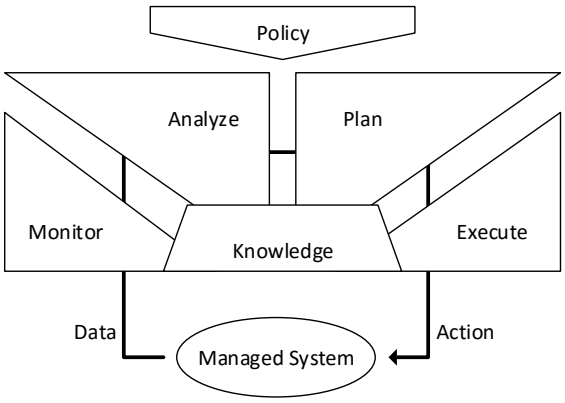


Figure 2.5: The general Monitor-Analyze-Plan-Execute (MAPE) loop in Autonomic Computing (AC) (adapted from [KC03]).

The principles of AC and the more general autonomic management also inspired SON, especially with respect to the self-x aspects [Wal+11][DAS11]. However, their goals diverge: both aim to achieve autonomic behavior but define the related terms differently. In [NGM07b], automatic behavior is defined as "a process where a significant part of the action is handled by a machine but some human interaction is required (e.g., where an operator is required to activate machine specific macros, or required to verify results)" [NGM07b, p. 6], whereas autonomic behavior is related to "a process which is fully controlled by a machine without requiring human interaction." [NGM07b, p. 6] Hence, it seems that the NGMN definition of autonomic behavior is broader and, actually, comprises both automatic, i.e., non-objective-

driven, and autonomic, i.e., objective-driven, behavior as defined in AC. In other words, the NGMN does not distinguish between the AC community's automatic and autonomic behavior. In order to avoid confusion, we will commonly refer to the AC definition of automatic and autonomic in the following.

Autonomic network management [SSH06][Jen+07][SK09][Ago11][Mov+12] is another fork of AC. In contrast to SON, it is more concerned with network management in general and not specific to RANs. However, this concept is closely aligned with AC since an autonomic network enables "anticipating, diagnosing and circumventing any impairment in the functionalities of the underlying network in an independent and autonomic manner, driven by a set of higher-level business-oriented goals, with minute human intervention or supervision." [SK09, p. 22]

Since AC and related concepts are focus on a policy, they can be seen as an instance of PBM that is business-driven [BA07]. This is in contrast to other PBM concepts that automatically follow a policy. The differences between these approaches can be outlined using the classification introduced in [KW04]. In principle, there are three types of policies: action policies, goal policies, and utility function policies. This classification is inspired by Artificial Intelligence (AI), particularly the three types of agents introduced in [RN10]: reflex agents follow an action policy, goal-based agents follow a goal policy, and utility-based agents follow a utility function policy.

In order to outline the different policy types, consider a system, e.g., a mobile network, which can be in different states S representing, e.g., the performance and configuration of the network. In this most basic system representation, the PBM has a set of actions A which it can perform. Thereby, each action $a \in A$ triggers a transition from the current state $c \in S$ to a future state $s \in S$. The fact that the transition is deterministic simplifies the explanation below, however, the approach also allows stochastic actions, i.e., a may transition to any state s with the probability $\Pr(V_S = s \mid a, c)$ with V_S denoting the random variable for the future state. Based on that, the PBM system selects an action $a \in A$ given the current system state c according to the policy in order to achieve the overall objectives of the system.

2.2.1 Action Policy

An action policy is the most concrete form of policy which dictates the action a that the system should take based on the current system state c [KW04]. Typically, the policy is expressed as condition-action rules of the form **IF** condition(c) **THEN** a, whereby condition is a predicate over the current state c and a is the action to perform. That means that the author of the policy knows that if c satisfies the condition then a should be performed in order to achieve the overall objectives. In order to compile such policy, the operator needs to project his thoughts into each possible system state c, derive the resulting state s for each action a, assess s with respect to the objectives, and create a rule that selects the action a in c which satisfies the objectives the most. Obviously, this is a complex process. Furthermore, the maintenance of the policy is costly since the action results and the objectives are

mixed up: if either the transition model or the objectives change, the policy author, in principle, has to delete the policy redo the whole process again since it is not possible to determine which rules might be affected by the change. Nevertheless, action policies are widely used since the approach is easy to comprehend and the behavior of the system easy to predict.

An action policy should be complete and conflict-free. The former means that for each possible state c there is a corresponding action a proposed, if this is necessary. This property is often not hard to achieve. The latter means that the proposed actions for some state c must be allowed to be executed together by the system. This is a more challenging problem that attracted a lot of research under the term policy conflict detection and resolution. In simple cases, the conflict detection can be simplified by the fact that only a single action a is allowed for each c. The resolution of such conflicts, i.e., the selection of one action from a set of actions, can be done by analyzing the rules that triggered the actions, the conditions of the rules, or the actions. [IS89] provides a general formalization of conflict resolution in rule systems. Practical approaches in available rule system are the assignment of priorities to the rules, e.g., JBoss Drools [The13] or OpenRules [Ope13], or using defeasible logic to prefer more specific rules over more general rules, e.g., Ponder2 [Lup13] or OpenRules [Ope13]. Recently, the Object Management Group (OMG) also published the Decision Model and Notation specification [OMG15] that aims to standardize the diverse action rule approaches.

2.2.2 Goal Policy

A goal policy directly defines a set of goal states $S_g \subseteq S$ that the MNO aims to achieve, i.e., states that satisfy the objectives [KW04]. In other words, an action policy defines *how* a PBM system should react whereas a goal policy defines *what* a PBM system should achieve. The PBM system's task is to compute the action $a \in A$ in the current state $c \in S$ that will lead the system immediately or eventually to some desired state $s \in S_g$. Therefore, the PBM system requires a model of the managed system in order to predict the state s to which an action a transitions from state c. Consequently, this approach trades the efforts for defining an action policy with the efforts for modeling the system. However, the separation of the objectives from this functional knowledge permits greater flexibility and eases the maintenance since both can evolve independently.

A goal policy can also be conflicting if there is a situation $c \in S$ in which the system cannot reach one of the goal state S_g. In this case, the system does not have any hint which action to select. Notice that, in related work, a goal policy is often not used to select a single action but to plan an action sequence to achieve a goal [RN10], e.g., as shown in [Ban+04].

2.2.3 Utility Function Policy

A utility function policy aims to directly represent the objectives by assigning each state $s \in S$ a utility value $u(s) \in \mathbb{R}$ [KW04]. Thereby, $u(s)$ represents the degree of

satisfaction of the objectives by s. The definition of the utility functions is related to decision theory that will be presented in more detail in Chapter 2.3. The task of the PBM system is then to select the action a in the current state c that immediately or eventually maximizes the utility. As can be seen, a utility function policy is a generalization of a goal policy: whereas the latter defines a binary classification into desirable and undesirable states, the former defines a continuous measure of desirability. Again, a model describing the transitions for the actions is required.

Interestingly, a utility function PBM system can resolve conflicts among the action by itself. Since the general task is to increase the utility of the state, the system always selects the action that transitions to the state with the highest utility. If two state have the same utility, then they are considered equally preferred and it does not matter which action to take. Specifically, this also enables making trade-offs between different objectives that are not achievable together. In principle, a utility function policy can be transformed into a goal policy using mathematical optimization or into an action policy using decision-theoretic planning [RN10].

2.3 Decision Theory

"[D]ecision theory and its companion methodology of decision analysis deal with the merits and making of decisions" [DT99, p. 55], i.e., it is about "goal-directed behaviour in the presence of options" [Han05, p. 6]. It provides answers to the questions: "What Is a decision? [...] What Makes a Decision Good? [...] How Should One Formalize Evaluation of Decisions? [...] How Should One Formulate the Decision Problem Confronting a Decision Maker?" [DT99, pp. 56-57] In principle, two types of decision theory can be distinguished [HBH91]: normative decision theory shows how a rational decision maker should behave in order to achieve its goal and descriptive decision theory describes how people actually, often non-rationally behave. Consequently, the latter is related to psychology and an important basis for economics in order to predict human behavior. On the contrary, the former provides an executable framework to determine which action should be made. Decision theory is a comprehensive field of research that goes back to the 1940s [NM53, 1st edition], however, only since 1990s, normative decision theory has been extensively used in artificial intelligence, particularly for planning under uncertainty [Dom+11]. In summary, decision theory provides a sound theoretical foundation for PBM based on a utility function policy (see Chapter 2.2.3).

"Decision theory is based on the axioms of probability and utility. Where probability theory provides a framework for coherent representation of uncertain beliefs, utility theory adds a set of principles for consistency among beliefs, preferences, and decisions." [HBH91, p. 66] This section aims to provide a focused introduction into decision theory such that the concepts applied in this thesis can be understood. Thereby, we concentrate on expected utility theory and multi-attribute decisions. We assume that the reader has knowledge about general probability theory and probabilistic reasoning as presented in [RN10, Ch. 13, 14]. For a more theoretical overview on decision theory, please refer to the seminal works [Fis70][Kee82][KR93],

as well as the intelligible tutorial [BD09]. Most of the information given in this chapter is based on [Dye05].

2.3.1 Preferences

"Preferences describe a decision maker's relative ordering of the desirability of possible states of the world." [HBH91, p. 66] In principle, a preference is a binary relation $\succsim$ over a set S of choices or states which can be, e.g., different cars like a sports car, a limousine, or a compact car. In this example, sports car $\succsim$ compact car means that the decision maker strictly prefers a sports car over a compact car, i.e., sports car $\succ$ compact car, or is indifferent between both, i.e., sports car $\sim$ compact car. Consequently, $\succsim$ is referred to as a weak preference relation. In order to allow formalization and reasoning over preferences, $\succsim$ is generally assumed to be a weak order which is transitive and complete, i.e., $\forall x, y, z \in S. x \succsim y \wedge y \succsim z \implies x \succsim z$ and $\forall x, y \in S. x \succsim y \vee y \succsim x$ [Dye05].

There are numerous approaches for the representation of and reasoning about preferences [BD09][Dom+11], e.g., graphical, logic-based, soft constraint-based representations. In the context of this thesis, it is important to consider that the actions which may be performed are stochastic as described in Chapter 2.2, i.e., there is an uncertainty which state $s \in S$ might materialize after executing an action. For this setting, the expected utility theory presented in the following is the most common representation. The reader may refer to [Ste05] for some alternative preference representations involving uncertainty.

2.3.2 Expected Utility Theory

Expected utility theory by Neumann and Morgenstern [NM53] is perhaps the most significant and most commonly used "preference representation for risky options, where the risky options are defined as lotteries or gambles with outcomes that depend on the occurrence from a set of mutually exclusive and exhaustive events." [Dye05, p. 271] Thereby, the term lottery is a bit misleading as it actually refers to a probability distribution over a random variable V_S of the possible states S, i.e., $\Pr(V_S)$. For instance, a lottery can be actually a lottery ticket with a probability distribution over a set of cars which can be won. "The key result of [expected] utility theory is that, given fundamental properties of belief and action, there exists a scalar function describing preferences for uncertain outcomes." [HBH91, p. 66] It is also important to note that, due to its long history, the theoretical foundations of expected utility theory are well investigated [Kar14].

Expected utility theory is mostly described using the three basic axioms presented in [Fis70]. Therefore, consider P as a set of probability distributions over the set of possible states S, e.g., different urns containing lottery tickets with different chances to win the cars. Consequently, a lottery $p \in P$ is a function $p : S \to [0, 1]$ from a possible state $s \in S$ to its probability with $\sum_{s \in S} p(s) = 1$. A preference relation $\succsim$ fulfills the axioms

Order , i.e., $\succsim$ is a weak order;

Independence , i.e., $\forall a \in [0,1]. \forall p, q \in P. p \succsim q \implies ap + (1-a)r \succsim aq + (1-a)r$;

Continuity , i.e., $\forall p, q, r \in P. p \succ q \succ r \implies \exists a, b \in (0,1). ap + (1-a)r \succsim q \succsim bp + (1-b)r$ (note the strict preference between p, q, r)

if and only if there exists a real-valued utility function $u : S \to \mathbb{R}$ such that

$$\forall p, q \in P. p \succsim q \iff \mathrm{E}[u(p)] \geq \mathrm{E}[u(q)] \iff \sum_{s \in S} p(s)u(s) \geq \sum_{s \in S} q(s)u(s) \quad (2.1)$$

with $\mathrm{E}[u(p)] = \sum_{s \in S} p(s)u(s)$ referring to the expected value of u given p. Even more, u is unique up to positive linear transformations. These axioms are generally accepted and intuitively formalize preferences that can be traded-off against each other based on their probabilities. The reader may refer to [BD09] for a more extensive explanation.

A rational decision maker then selects the action whose probability distribution $p \in P$ over the outcomes maximizes the expected utility [RN10, Ch. 16], i.e.,

$$\arg\max_{p \in P} \mathrm{E}[u(p)]. \quad (2.2)$$

The existence of the utility function u is the key result of the theory as the expected values can be easily compared in order to enable a computer to automatically select actions that the user prefers. It is commonly assumed that $u(\cdot) \in [0,1]$, i.e., u(least preferred outcome) $= 0$ and u(most preferred outcome) $= 1$ as this normalization simplifies the creation of and reasoning with u. However, although u exists, the theory does not provide a general way to generate it. Nevertheless, [BD09], for instance, presents an approach which is based on a trade-off question: Given some outcome s, "For what value p is it the case that you are indifferent between getting [s] for sure and a lottery in which you obtain [most preferred outcome] with probability p and [least preferred outcome] with probability $1 - p$." [BD09, p. 75]

Interestingly, "[u]tility theory also provides ways to express attitudes toward uncertainty about outcome values, such as risk aversion." [HBH91, p. 66] A decision maker is risk-averse, risk-neutral, or risk-seeking if and only if the utility function u is strictly concave, linear, or strictly convex respectively [KR93]. Chapter 3.4.3.1 as well as [Win04] exemplify this. As a result, u encodes both the preferences for the outcomes as well as the risk attitude.

2.3.3 Multiattribute Decision Making

Expected utility theory requires the decision maker to define $u(s)$ for every single possible outcome $s \in S$, e.g., each specific car. If there are numerous outcomes, this is impracticable. Multiattribute decision making assumes that the outcomes are characterized by some structure which is important for the preferences, and exploits this structure. The structural characteristics are referred to as attributes [BD09]. For example, a car may be characterized by the attributes top speed with values $\{\text{fast}, \text{slow}\}$, fuel consumption with $\{\text{economical}, \text{lavish}\}$, and space with $\{\text{spacious}, \text{cramped}\}$. Consequently, $S = \prod_{i=1}^{n} S_i$ with n attributes, where S_i is the

domain of the attribute i. Thus, a specific outcome $s \in S$ is a tuple $s = (s_1, \ldots, s_n)$ of concrete values for each attribute $s_i \in S_i$.

The idea of multiattribute decision making is to decompose the decision making over possible outcomes s into a decision making over the attribute values s_i and combine them appropriately. In the following, we present the most significant theory of this type called Multiattribute Utility Theory (MAUT). Additionally, we present lexicographic preference orders as a complementary approach.

Notice that another common way to treat multiattribute decision problems is to use a dominance relation between the outcomes of different actions in order to create a so-called Pareto set or Pareto front [Ste05][RN10, Ch. 16][Dom+11]. This set is a subset of the outcomes of the actions which contains only those action outcomes that are not strictly dominated by any other action outcome. This means that for an action outcome in the subset there is no other actions outcome which is weakly preferred regarding all attributes and strictly preferred regarding at least one attribute. The drawback of this approach is that the Pareto set contains several actions and the approach provides no means to automatically select the best action from them. In contrast, the approach solely aims to filter the set of actions that a decision maker can select from.

2.3.3.1 Multiattribute Utility Theory

Multiattribute Utility Theory (MAUT) is based on expected utility theory and extends it to multiattribute decision making. The goal of MAUT is to decompose the global utility function $u(s_1, \ldots, s_2)$ into individual, marginal utility functions $u_i(s_1)$ for each attribute, and to aggregate the partial or marginal utilities into a global utility value using an appropriate aggregation. Based on this, the decision maker should select the action that maximizes the expected aggregated utility over all attributes as shown in Equation 2.2. Notice, that MAUT allows the comparison of attributes with different measures, e.g., compare apples and oranges, by normalizing them to the common preference measure utility. As outlined before, we assume the utility functions to be scaled such that $\forall i = 1, \ldots, n.\, u_i(\cdot) \in [0, 1]$.

The challenge of MAUT is to define the aggregation. There are numerous forms of decomposition of the global utility function [Dom+11]. Typically, they differ in the strength of the required independence properties on the decision makers preferences [FK74]. In general, it can be said that the weaker the independence requirements, the wider the range of preference relations that can be expressed and, however, the more additional parameters need to be elicited from the decision maker to configure the aggregation.

The additive aggregation model is by far the most commonly used multiattribute utility function [Dye05][Ste05][WD92]. It is a very restrictive model requiring additive independence, i.e., "preferences between the multivariate lotteries depend only on the marginal probability distributions." [Ste05, p. 452] This means that the preferences only depend on the probabilities $p(s_i)$ for all $i = 1 \ldots n$ and not on the joint probability distribution $p(s_1, \ldots, s_n)$. Consider the car example with the attributes speed and fuel consumption, and the respective preferences fast $\succsim$ slow and

economical $\succsim$ lavish. Given the following two car lotteries p and q:

- $p\,(\text{fast}, \text{lavish}) = 0.5$, $p\,(\text{slow}, \text{economical}) = 0.5$ and
- $q\,(\text{fast}, \text{economical}) = 0.5$, $q\,(\text{slow}, \text{lavish}) = 0.5$,

the decision maker is indifferent between these, i.e., $p \sim q$, because for each single attribute, the probability to achieve each value is equally 0.5. However, this example shows also the restrictiveness of the model, as human decision makers will likely prefer p, i.e., $p \succsim q$, in order to avoid the chance to get the bad car (slow, lavish).

Given additive independence, the global utility function $u(s_1, \ldots, s_2)$ can be represented in the well-known additive form

$$u(s_1, \ldots, s_2) = \sum_{i=1}^{n} k_i \cdot u_i(s_i) \tag{2.3}$$

with $k_i \in [0, 1]$ being a scaling constant or weight specific for each attribute S_i, and $\sum_{i=1}^{n} k_i = 1$. [CSW99] presents several intuitive interpretations of these weights.

Although additive independence is a strong requirement that does not hold in general, the additive form is the most commonly used MAUT function [Ste05]. The reason for this is that it requires fewer parameters, i.e., k_i, to be elicited from the decision maker in comparison to more accurate, less restrictive models. Furthermore, [Ste95][Ste96] empirically show that the selection of an incorrect aggregation method, i.e., one that is too simple, has only little influence on the result in comparison to the selection of wrong utility functions. Hence, "[o]ur overall conclusion is thus that in the practical application of expected utility theory to decision making under uncertainty, the use of the additive aggregation model is likely to be more than adequate in the vast majority of settings." [Ste05, p. 454] Finally, the simplicity of the additive form facilitates understanding the decisions made by the system.

The elicitation of the parameters for a MAUT decision problem is a complex task and subject of extensive and ongoing research [CP04][BD09][Dom+11]. New approaches are specifically focused on automatically learning the preferences by providing the decision maker with examples or by monitoring human behavior. In principle, the elicitation can be formalized as a three step process [Kee74]:

1. A suitable aggregation form needs to be selected and its assumptions, e.g., additive independence, need to be verified. As already pointed out, it seems that in practice this model has little influence compared to the other parameters and, hence, the additive models seems preferable in the majority of the cases due to its simplicity in terms of the quantity and quality of the required parameters [Ste05]. In [DL97], the authors present a simple trade-off decision to check whether the additive model is applicable.

2. The marginal utility functions $u_i(\cdot)$ need to be defined. [KR93] provides several assessment strategies. In case of the additive form, each u_i can be evaluated independently of the other attributes by using the trade-off questions known from expected utility theory [Dye05]. [SGM05] presents a method to elicit

the utility functions from a set of example decisions. [Ste05] even argues that, in reality, probabilistic utility functions generally not differ remarkably from deterministic value function, and, thus, even simpler methods for value function elicitation (see [BD09]) may be used.

3. Finally, the parameters of the aggregation need to be determined. In case of the additive form, this refers to eliciting the weight k_i for all $i = 1, \ldots, n$. In principle, these weight can be assessed using trade-off questions similarly to the assessment of the utility functions [Kee74]. However, [Dye05] notices that, although not completely accurate, the famous Analytic Hierarchy Process (AHP) [Saa08], which is based on pairwise comparisons of possible outcomes, may be used to determine the global utility function, specifically the weights [Dye05].

[Kee77] provides a nice report which exemplifies this process in the area of energy policy.

2.3.3.2 Lexicographic Preference

Lexicographic preferences are a special form of preferences in multiattribute decision making [Fis74][Miy95]. They express priorities over the attributes of a decision outcome, i.e., they express that the achievement of a good value for a high priority attribute is more important than a good value for a low priority attribute. Particularly, a bad value for a high priority attribute cannot be traded off against a very good value for a low priority attribute.

Consider the attributes $S_1, \ldots, S_n$ with $S = \prod_{i=1}^{n} S_i$ to be sorted according to their priority with S_1 having the highest priority and S_n having the lowest priority. The lexicographic preference over two alternative outcomes $(s'_1, \ldots, s'_n) \in S$ and $(s''_1, \ldots, s''_n) \in S$ is then defined as

$$(s'_1, \ldots, s'_n) \succsim (s''_1, \ldots, s''_n) \iff \exists i \in \{1, \ldots, n\}.\, s'_i \succsim s''_i \wedge \forall j \in \{1, \ldots, i-1\}.\, s'_j \sim s''_j. \quad (2.4)$$

Thereby, the preferences regarding each attribute may be determined by utility functions. It has been shown, that a lexicographic preference relation, in general, cannot be represented as a single utility function [Miy95][Fis74]. Particularly, lexicographic preferences do not satisfy the continuity axiom of expected utility theory outlined in Chapter 2.3.2. As a result, an automatic decision maker implementing lexicographic preferences must, in general, implement the above outlined decision rule, i.e., a decision maker will select the output $(s_1, \ldots, s_n) \in \mathcal{S}$ from a set of possible outcomes $\mathcal{S} \subseteq S$ such that $\forall (s'_1, \ldots, s'_n) \in \mathcal{S}.\, (s_1, \ldots, s_n) \succsim (s'_1, \ldots, s'_n)$. The reader may refer to [Dom+11] for an overview of contemporary research on eliciting lexicographic preferences.

2.3.4 Influence Diagrams

A common graphical representation for complex decision problems under uncertainty are influence diagrams [HBH91][RN10, Ch. 16]. An influence diagram is an extension

of the well-known Bayesian net which is commonly used for modeling probabilistic inference problems [RN10, Ch. 14]. An example for a simple decision problem about the location of a new airport is depicted in Figure 2.6.

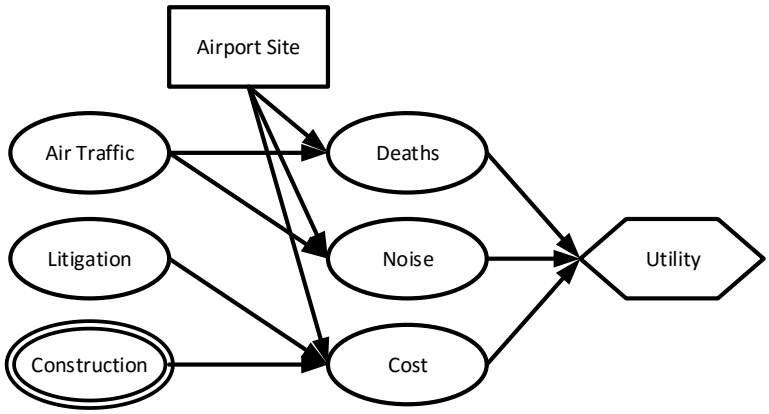


Figure 2.6: Example of an influence diagram for the decision where to build a new airport (adapted from [RN10])

An influence diagram is an acyclic directed graph consisting of arcs and nodes. The arcs are thereby directed from a parent node to a child node. There are four types of nodes [HBH91]:

Rectangles represent decision nodes that model a decision between different actions that a decision maker can do. Hence, such a node is assigned with a list of possible actions. For instance, the decision node "Airport Site" in Figure 2.6 represents the different location options for a new airport.

Ovals represent chance nodes that model random variables, similarly as they do in Bayesian nets. Hence, each chance node has an assigned conditional probability distribution which is dependent on the state of its parent nodes which can be other chance nodes or decision nodes. For instance, the chance node "Cost" in Figure 2.6 has the assigned conditional probability distribution $\Pr(\text{Cost} \mid \text{Litigation}, \text{Construction}, \text{Airport Site})$.

Double Ovals represent deterministic nodes which are specializations of the chance nodes. A deterministic node represents either a known constant or a variable which can be calculated from the states of its parents, i.e., which has a deterministic dependency to its parents. In Figure 2.6, "Construction" represents a constant, known type of construction of the airport.

Diamonds represent value nodes that model the utility functions of the decision maker. This function may be a multiattribute utility function which depends on the states of the parent nodes. In principle, any aggregation function can be used. For instance, the node "Utility" in Figure 2.6 has an assigned function $u(\text{Deaths}, \text{Noise}, \text{Cost})$.

An influence diagram enables a compact representation of a complex decision problem. However, they can also be executed in the sense that they are able to infer the expected utilities of the different decision options [RN10].

3 Objective-Driven SON Operations

This chapter introduces the general concept of Objective-Driven SON Operations (ODSO) for mobile networks. In order to motivate this work, we first present the fundamental problem of manual mobile network operations in detail: the manual gap that exists between the operational objectives and the network configuration. SON aims to automate network operations and, hence, shrinks the gap. However, we outline that there is a similar, though smaller manual gap between the operational objectives and the introduced SON configuration. Based on this motivation, we detail out Objective 1 of this thesis that is to close the manual gap of SON operations by introducing a holistic and autonomic approach for SON operations. In order to fulfill this, we present an architecture that integrates the three operational tasks SON management, SON coordination, and SON self-healing. Furthermore, an application-independent, generic design for each of these ODSO components is introduced which performs objective-driven decision making. These ideas make up the Solution and Contribution 1 of this thesis. The following chapters then map their operational task-specific use cases into this generic design.

Notice that we focus on the operations of the RAN, i.e., the numerous network cells that are spanned up by the BSs. Thus, throughout this work, the term mobile network refers to the RAN.

3.1 Problem and Motivation

Mobile network operations is the task of keeping a network running optimally regarding the performance and smoothly regarding the availability, i.e., it comprises performance optimization and troubleshooting. In a mobile network without SON, the human operators have to manually perform network operations as depicted in Figure 3.1. Thereby, a human operator continuously monitors the network's performance and healthiness through the PM, FM, and CM data, analyzes the collected data for performance issues or failures, and deploys a suitable new network configuration or recovery actions to mitigate them. This human-in-the-loop process is driven by operational objectives that define the desired network performance. The following presentation of network operations is partially based on a survey among MNOs [Cam+15].

Manual network operations is an involved task due to two reasons: On the one hand, a mobile network, typically consisting of thousands of network cells with different RATs spread over a huge area like a whole country, is inherently technical

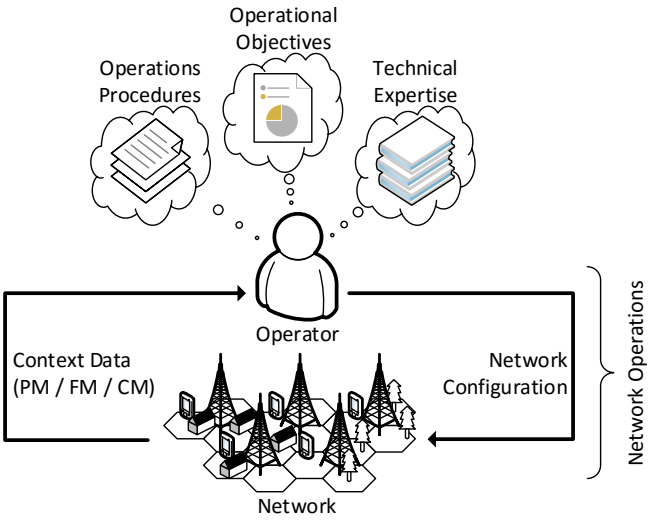


Figure 3.1: Manual network operations: the human operator monitors the network data and reacts to detected problems by changing the network configuration.

complex. Thereby the BSs are typically provided by different vendors requiring detailed technical knowledge for their configuration. On the other hand, the MNO's objectives are formulated in terms of KPIs values that the network should achieve. Hence, the operator has to map the operational objectives to the performance and health data collected from the network in order to determine network cells that do not meet the desired performance. Furthermore, the operator has to determine a suitable new network configuration in order to overcome the issue such that the satisfaction of the objectives by the network is improved. However, the radio properties and, thus, the cell performance in terms of KPI values strongly depends on the physical environment and is often hard to predict. We summarize these issues as the *manual gap of network operations* between operational objectives and network configuration. It requires considerable efforts by the operations personnel to overcome.

In order to support the operational personnel, MNOs develop operations procedures: step-by-step manuals explaining the actions that a human operator has to perform to detect and diagnose specific problems as well as feasible actions to overcome the issues. They are typically created as part of the network planning process by a separated team based on an abstract model of the network and abstract objectives. Hence, the application of these procedures in a concrete situation still requires considerable technical expertise from the operators. On the one hand, the complexity of the mobile network, with different types of complex BSs in different locations with different physical properties, makes it impossible to define strict recipes for the detection of problems as well as their mitigation. On the other hand, the procedures need to be adapted to the concrete objectives which are typically not uniform over the whole network and at all times. In summary, the operator needs to adapt the

operations procedures according to the operational context based on his technical expertise in order to achieve the operational objectives.

The introduction of SON aims at simplifying network operations by automating recurrent tasks in network operations. Therefore, in SON-enabled network operations, depicted in Figure 3.2, the operations procedures that support the human operator in manual network operations are substituted by automatic SON functions. In other words, each SON function can be seen as the implementation of a specific operations procedure including the problem-specific detection, diagnosis, and mitigation. Consequently, the deployment of SON functions reduces the necessary manual efforts for network operations since the human operators do not need to collect and analyze the network data, and derive suitable new network configurations anymore. Instead, the human operator has a new task that is to ensure that the SON is running optimally regarding the network performance and smoothly regarding the availability. Consequently, the manual efforts shift from network operations to SON operations.

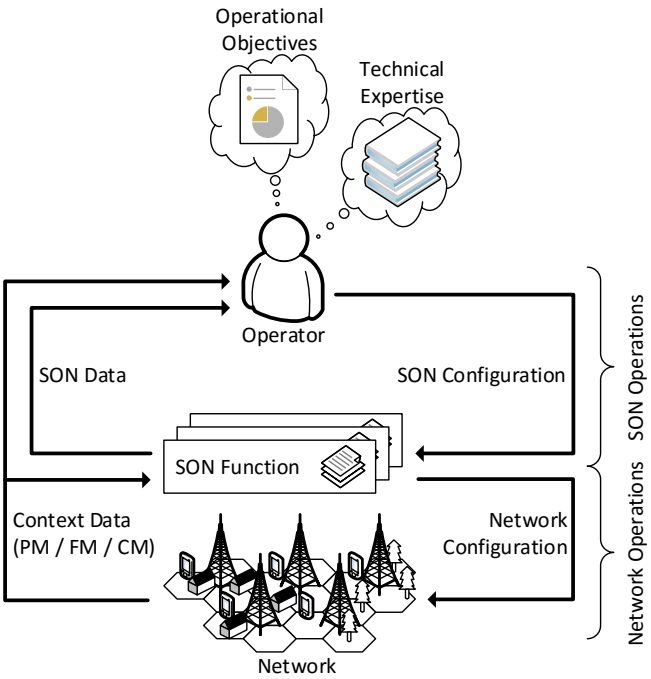


Figure 3.2: SON-enabled network operations: the human operator monitors the network and SON data and reacts to detected problems by changing the SON configuration.

Exactly like the operations procedures, the application of SON functions requires considerable technical expertise by the human operator in order to control the automated processes such that the specific operational objectives are satisfied under the concrete network conditions given by the operational context. This comprises three specific tasks:

SON management refers to the configuration of the SON functions, i.e., "setting targets, configuring the SON function behaviour at a high level, and monitoring SON function results" [Wal+11, p. 61].

SON coordination refers to avoiding negative interactions between SON functions.

SON self-healing refers to the automatic detection and mitigation of problems in the network or the SON system[1].

There are already approaches to partially automate some of these SON operations tasks, particularly SON coordination [Ban+11b][Kür+10][Bar+13a][Cam+15] and SON self-healing [Nov+11][Kür+10][Bar+13a]. However, all of them define automated functions, called SON functions as well, that need to be configured, often through an action policy, such that the specific operational objectives are satisfied under the concrete network conditions [Kür+10]. Hence, the SON configuration that the operational personnel needs to determine comprises of the configurations of all SON functions including coordination and self-healing.

Figure 3.3 summarizes the core task of manual SON operations: based on their technical expertise, the operational personnel needs to determine a SON configuration that optimizes the mobile network such that the satisfaction of the operational objectives in the operational context is maximized. This is still a complex task that requires considerable human efforts. Hence, we coin this the *manual gap of SON operations* between operational objectives and SON configuration. In the following, the gap is presented in more detail by describing the operational objectives and the technical expertise required for SON operations, as well as the induced efforts in the operations process.

3.1.1 Operational Objectives

The ultimate business goal of an MNO is to make profit. Therefore, the MNO defines a business strategy that materializes in high-level goals, referred to as operator goals [Ben+13b]. The goals are related to three areas that frame the general business of each MNO:

Cost: MNOs are under a constant pressure to reduce the cost of operating their network since revenues are stagnating while new technologies require extensive investments [GSM15]. Hence, they aim for reducing their expenditures, i.e., CAPEX and OPEX.

Customer satisfaction: Due to fierce competition, the users of mobile networks can easily switch the MNO. As a result, an MNO needs to provide a good QoS and Quality of Experience (QoE) in order to keep its current users happy and attract new users. Besides customer complaints, regular tests of the mobile network quality by governmental agencies, e.g., Autorité de Régulation des Communications Électroniques et des Postes (ARCEP) in France [ARC15b], or

[1] This is an extension of the self-healing OAM area which focuses solely on network outages (see Chapter 6.1.1).

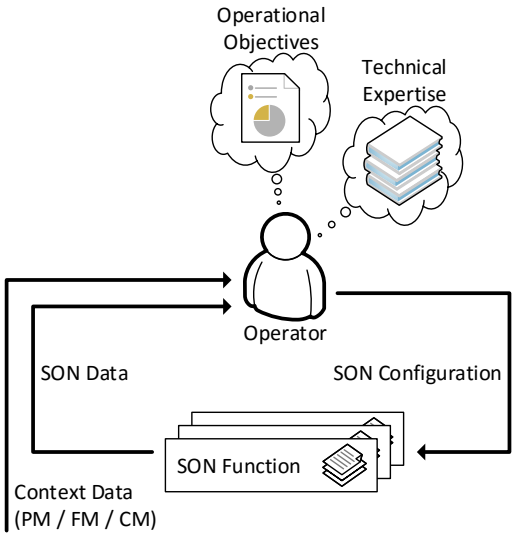


Figure 3.3: Manual SON operations means combining the operational context, operational objectives, and technical expertise in order to determine a good SON configuration.

private test institutions, e.g., connect in Germany [Con14], provide particularly valuable insights since, besides an evaluation of the network performance, they provide concrete requirements for network quality.

Regulations: In most countries, there are regulatory requirements and restrictions by official authorities that need to be respected. For instance, these can be minimal requirements on the coverage of areas with specific RATs as encountered in France [ARC15a]. The penalties for violating these rules can range from bad publicity in a public MNO ranking to the withdrawal of the MNO license.

The high-level goals are usually imprecisely expressed since there is no formalized process for their definition. For instance, some may contain vague satisfaction targets, e.g., "make customers happy", some may be hard to evaluate, e.g., "be an innovation leader", and some may be ambiguous, e.g., "be the best".

The operator goals are, of course, not fixed for all time. In order to stay competitive, the marketing divisions periodically define temporary campaigns which aim to attract new customers. Such campaigns are typically focused on specific users or services and have an impact on the whole network. Hence, such offers change and shift the MNO's goals for a limited period of time. Furthermore, the goals are sometimes adapted to special events that take place, e.g., sports events or trade fairs. These adaptations typically impact the goals in a limited region in the network and apply for a rather short time.

Due to the vague expression, high-level objectives are not suitable for network operations. Instead, the operational personnel requires clearly defined and measurable

objectives in order to evaluate whether the current network configuration is satisfactory with respect to the objectives and to determine countermeasures for insufficient performance. Therefore, the high-level goals are transformed to operational objectives which are based on technical network KPIs as depicted in Figure 3.4. This common problem has also been identified by, e.g., the TeleManagement Forum, a standardization body comprising vendors and MNOs from the telecommunication industry. It provides a standardized framework, referred to as *Frameworx*, which allows modeling of goals on different levels of abstraction and a translation between them [Tel04]. They specifically distinguish between product Key Quality Indicators (KQIs), service KQIs, and network KPIs. Thereby, the goals at the level of network KPIs closely corresponds to the operational objectives in this work.

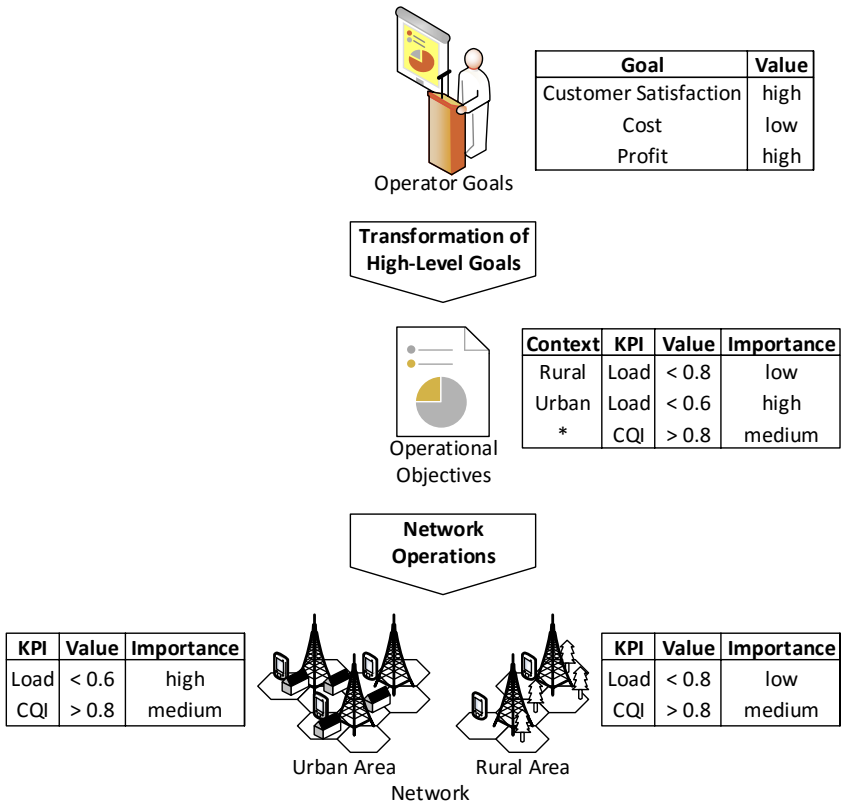


Figure 3.4: Derivation of operational objectives from operator goals, and their application in different operational contexts as part of SON operations.

The KPIs have three interesting features: they provide a fine-grained, technical view on the network performance, they can be calculated quickly from simple measurements and counters, and they are commonly understood, unambiguously defined and mostly standardized. For instance, the CQI captures the overall signal quality, the cell load captures the data performance, and the handover ping-pong rate is an indication for the mobility performance. Therefore, some of them are also considered by some SON functions as optimization targets. The transformation process

that breaks the operator goals down to operational objectives is, however, often performed implicitly and ad-hoc [Cam+15]. Typically this is done by experienced experts that draw on traditions, best practices, and their skills. Such imprecise processes can be barely automated. In this work, we start with the operational objectives and assume that they are provided by the MNOs.

The advantage of the unambiguous, measurable, and assessable operational objectives[2] is that network operations can fully concentrate on the technical aspects of their satisfaction. The objectives allow evaluating the network performance, identifying concrete deficiencies in terms of degraded KPIs, and ranking these issues according to their severity. Of course, the KPIs must be well-defined with respect to their calculation from low-level measurements and their aggregation in time, e.g., the mean value over 1 hour, and space, e.g., the mean value over a cell group. Additionally, the satisfaction of the objectives allows the MNO's management to monitor and evaluate the network operations department and react to problems quickly. For that reason, such objectives are also commonly used in Service Level Agreements (SLAs) between an MNO and external service providers which are defined as part of a network operations outsourcing contract.

3.1.1.1 KPI Targets

The simplest and most basic way to define an operational objective on a KPI is to define a threshold which separates the acceptable KPI values, which the network preferably should produce, from the unacceptable values. Consider the example shown in Figure 3.4, load < 0.6 means that a cell load between 0% and 60% is acceptable whereas a load between 60% and 100% is unacceptable. Such thresholds typically define the minimal objectives since they are easy to evaluate. However, they do not foster continuous improvement of the network performance since once the binary thresholds are satisfied, there is no differentiation between different performance states of the network. For instance, the threshold used above does not distinguish between a cell load of 20% and 59%. Hence, the satisficing [Wei04] minimal thresholds are often complemented with maximizing objectives that guide continuous optimization once all thresholds are satisfied, e.g., minimize the cell load since the smaller, the better. In SLAs, the failure to meet satisficing objectives is typically penalized, i.e., if they are not satisfied then the service provider gets payed less. In contrast to that, maximization objectives are promoted, i.e., the provider receives a bonus for their achievement [Tel12]. Furthermore, maximization objectives also allow determining the severity of a performance degradation if the satisficing objectives are not met. For instance, a cell load of 100% is a more severe degradation than a cell load of 85% and, hence, the former will get full attention by the operational personnel.

[2]The objectives can be seen as an instance of SMART goals [Hau14], i.e., specific, measurable, assignable, realistic, and time-related.

3.1.1.2 Trade-offs

The main limiting factors in network optimization are the limited number of deployed BSs and physics. Both make it impossible to maximize the satisfaction of all objectives simultaneously. Instead, the objectives are typically competing with each other, e.g., in order to reduce the load within a network cell some of the UEs need to be handed over to other, neighboring cells, which, however, worsens their signal quality and, thus, the CQI. In network operations it is well-known that "every optimisation is a trade-off between the optimisation goals coverage, capacity and performance (or quality)." [NGM07b, p. 12] As a result, MNOs are prioritizing or ranking the different objectives which allows them to differentiate themselves from the competition [Sch+13]. This prioritization, referred to as importance in Figure 3.4, provides network operations with a guidance which objectives the personnel should focus their attention on and which objectives may be sacrificed in order to maximize the satisfaction of others, i.e., which trade-offs should be made. Typically, objectives on KPIs that have a high impact on user satisfaction, i.e., that users can perceive, are considered more important than others. For instance, the Dropped Call Rate (DCR) is important since uninterrupted calls are expected by the users. However, overall it is most important that all KPIs satisfy the minimum required performance before any additional optimization is conducted.

It is important to note the difference between the importance of the objectives and the focus in manual network operations: the former defines desired states of the network whereas the latter determines the intentions of the operational personnel given the concrete performance of the network[3]. For instance, consider that the handover performance is an important objective for the MNO. In busy hours with a lot of fast moving UEs, especially car traffic, network operations has to focus the limited human operators on improving the handover performance since it is more challenging to achieve a good result. However, in quite hours, handover performance might not be a big problem and, thus, the focus of the operations personnel might shift to lower priority objectives like providing capacity. This example outlines that, although the importance of the objectives is always the same, the focus of manual network operations can shift depending on the network performance.

3.1.1.3 Context-Dependency

A large mobile network spans diverse areas in a country, e.g., rural areas with few users distributed over vast distances and urban areas that are loaded with UEs. The operational objectives also reflect this diversity, i.e., the KPI targets and the trade-offs often differ between various areas as exemplified in Figure 3.4 with the urban and rural area. Apart from this, the objectives may not solely differ in space but also in time, e.g., if the MNO differentiates between peak, busy hours, and off-peak hours. For instance, the importance and threshold for the KPI cell load is higher during business hours in urban areas compared to another time and cell location. Similarly, the context-dependent objectives may also reflect special events that take

[3]This is comparable to the concepts used in Believes, Desires, Intentions (BDI) agents [RG95].

place in a specific location.

In general, this can be condensed to the statement that the operational objectives for a specific network cell depend on its operational context. This context is a collection of attributes of a cell that defines its operational state and may include:

- Configuration Management (CM) data like the cell type (e.g., macro, micro, or femto cell), the cell location (e.g., urban, suburban, or rural area), RAT (e.g., 3G, 4G, 5G),
- Performance Management (PM) data like KPI values,
- Failure Management (FM) data like raised alarms, and
- environmental data like the time (e.g., peak traffic hours or night time) and the date (e.g., weekend or public holidays).

Usually, the MNOs do not have objectives for each and every cell but for a cell group with some specific value for a context attribute [Hah+15]. For instance, Figure 3.4 shows a differentiation between cells with different values, i.e., urban and rural, for the context attribute cell location.

3.1.2 Technical Expertise

The management of a mobile radio network requires a lot of specialized knowledge. It takes months of experience before a human employee can productively operate a mobile network alone. This is caused by the complexity of the physical environment on the one hand, and the diverse technology on the other hand (see [Lii+12]).

Network operations has to set a huge range of configuration parameters for each network cell in order to control its behavior, algorithms, and protocols. This low-level network configuration comprises parameters like the CIO which controls the selection of the serving cell by the UE, the Transmission Power (TXP) which controls the radio signal power, and the Remote Electrical Tilt (RET) which sets the principal, vertical sending direction of a cell's signal. These parameters influence the measured values of the KPIs and, thus, the satisfaction of the objectives. However, this relation depends on the environment in which the cell is situated. On the one hand, the performance effects of a configuration are typically non-linear and strongly depend on the cell and the physical environment. In a fixed network, the capacity of, e.g., a link between two routers is well defined by the used cable and network cards. In a mobile network, however, the capacity of a radio connection between a UE and a BS depends on the capabilities of both endpoints but also on the position and movement of the user, the buildings between them, and even on whether the trees are green or leafless. On the other hand, even if the exact dependency between the environment and the effects would be known, there is always some uncertainty about the exact, actual state of the environment. This is either because the environment cannot be fully observed, e.g., the real propagation characteristics, or because some natural processes are not certainly predictable, e.g., the user behavior. Hence, the relation between network configuration and measured network performance is

indeterministic, which we refer to as *physical indirection* depicted in Figure 3.5. It makes it difficult for the operational personnel to accurately predict the network performance that can be achieved with some network configuration.

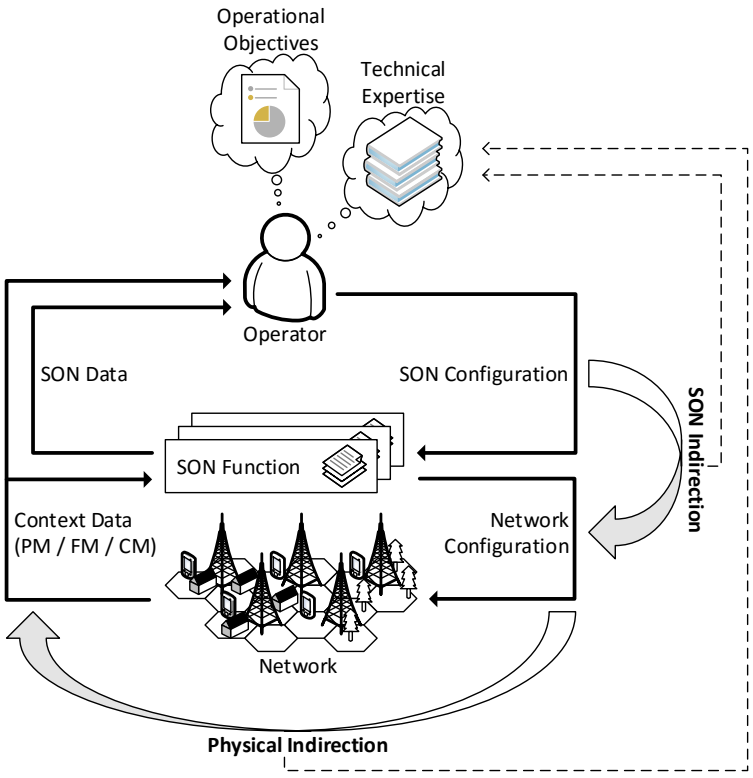


Figure 3.5: The two levels of indirection in SON operations, physical indirection and SON indirection, which must be overcome with technical expertise.

The physical indirection is typically overcome with two approaches: First, the MNOs try to create complex, realistic models of the real world, e.g., including buildings and street layouts, and use them to simulate the possible effects of some configuration. Second, in very complex situations, MNOs perform drive tests [Cam+15], i.e., vehicles with special measurement equipment are sent out to a specific area in order to measure the quality of the radio signal.

Network operations also has to face the technological complexity of modern mobile networks: First, there are typically several RATs, each with different functionality, and several cell types, like capacity-providing small cells and coverage-focused macro cells, active in parallel. Both build up a multi-RAT, multi-layer HetNet [Sar+11]. Second, the network equipment is often provided by several vendors leading to a diversity of interfaces and functionalities even within one RAT. It is the task of the operational personnel to integrate all these differences in order to come up with an overall network configuration that satisfies the operational objectives.

The introduction of SON for network operations reduced this complexity to some extent. In principle, SON functions are supposed to automatically adapt the network

configuration to the physical environment. However, this automation is currently limited and, hence, the operators have to configure the SON functions through a number of rather technical parameters. For instance, an MLB function may have the parameters: upper and lower CIO limit, the step size, the upper and lower cell load threshold, and the load average time [Kür+10]. Furthermore, the execution of the functions needs to be monitored and controlled by SON coordination and SON self-healing which themselves need to be configured.

Although the abstract functionality of SON functions is standardized, e.g., by 3GPP in [3GP11], the actual algorithms and the configuration interfaces are vendor-dependent. Especially in multi-vendor SON environments, this introduces a multitude of different interfaces that the operators must handle. Additionally, the prediction of the overall performance of these complex algorithms is not trivial as the behavior of a SON function with respect to its configuration is typically non-linear and shows no trends, i.e., a small change of the configuration of the SON function may trigger radical changes of the network configuration by the SON function. As a result, the enhanced automation by SON in network operations comes with another indirection, referred to as *SON indirection* as shown in Figure 3.5. It makes it difficult for the operational personnel to accurately predict the network configuration, and consequently the network performance, that a SON will produce with some SON configuration.

Putting this together, it can be noticed that the operational personnel controls the KPI performance with the SON configuration through two indirection steps:

Physical indirection means that human operators try to satisfy the operational objectives defined on the network KPIs that are indirectly controlled via the network configuration.

SON indirection means that human operators can only indirectly control the network configuration via the configuration of the SON.

As shown in [Hah+14], the resulting indirection makes it hard to predict the effects of some SON function with a given configuration. The knowledge and experience to overcome both indirection steps is what makes up the major part of the operational personnel's technical expertise.

The required extensive technical expertise to operate a SON impedes its fast adoption. On the one hand, MNOs see the "Efforts for transformation from manual to SON-enabled network operation" [Sch+13, p. 10] as an obstacle. Thereby, a considerable part of these efforts is building up the required technical experience in SON operations. On the other hand, the "uncertainty about how the self-organising networks will work in a real deployment is one of the main impediments for adopting SON" [Sch+13, p. 9], i.e., operators are not sure how to configure the SON and are uncertain whether a SON may result in inferior network performance. Thus, the operators often ask the vendors of the SON functions to provide a working configuration of the SON functions which they derives manually during the installation of the SON. However, this approach is non-optimal due to several reasons:

- The SON configuration is not derived from cell-specific objectives but instead from uniform objectives. Hence, the operator typically computes the strongest

common objectives, i.e., the objectives that fits the whole network the best, from the diverse operational objectives.

- The SON configuration does not adapt to seasonal variations of the environment, e.g., traffic changes during a day or changes in radio propagation during a year, but instead assumes a mean environment that is constant.

In summary, the default SON configuration is a uniform, static configuration that suits the network as a whole the best, but is not optimal for each and every single network cell.

3.1.3 Operations Process

The overall efforts for the operation of a SON are induced by mainly two factors: on the one hand, the complexity of the derivation of the SON configuration from to the operational objectives as presented in Chapter 3.1.2, and, on the other hand, the frequency of this derivation process. As depicted in Figure 3.6, an adaptation of the behavior of the SON, i.e., a change of the SON configuration, is in principle necessary if any of the following events happen:

Changes in context: Both the operational objectives (see Chapter 3.1.1.3) and the technical expertise (see Chapter 3.1.2) are context-dependent. Hence, whenever there is a change in the context of a network cell that affects the cell's objectives or relevant technical expertise, the SON configuration affecting that cell needs to be adapted. This event, e.g., the transition from peak traffic hours to night time, may happen rater often as indicated by the green marks on the time line.

Changes in operational objectives: Whenever the operational objectives change, e.g., due to a marketing campaign, they must be reevaluated for the whole network. Potentially, this can lead to an adaptation of the SON configuration for all network cells. The difference in the impact of objective changes to context changes can be described using Figure 3.4 (for the sake of this example, consider the context property time with the values peak and low traffic hours instead of the location with the values rural and urban). When the context of a cell changes, it might transition from peak to low traffic hours, hence, the lower derivation process (SON operations) needs to be executed for this particular cell. An objective change is the result of the execution of the upper derivation process (transformation of high-level goals) and leads to different objectives for the whole network. Consequently, the lower derivation process potentially needs to be performed for each and every network cell. Fortunately, this event does not happen often as indicated by the yellow marks on the time line.

Changes in technical expertise: The technical expertise might change when, e.g., the network is extended by deploying new BSs, or a new RAT or SON function is introduced. Consequently, this new knowledge might result in different

SON configurations for satisfying the operational objectives. However, since the technical expertise is not directly linked with a specific network cell, a change in the technical expertise potentially requires the adaptation of the SON configuration for the whole network. Like objectives changes, this event is rather seldom as indicated by the blue marks on the time line.

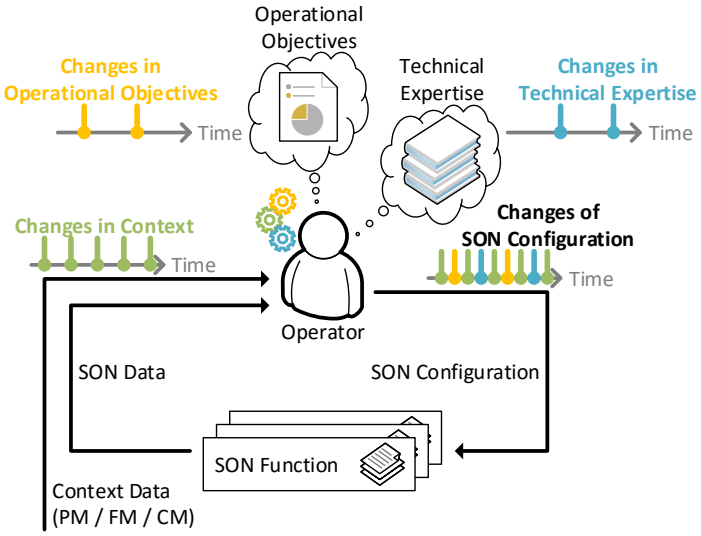


Figure 3.6: The frequency of changes in the context, the operational objectives, and the technical expertise, require very frequent adaptations of the SON configuration.

The different frequencies of these changes allow breaking the manual gap of SON operations down into two separate problems that need to be overcome:

Automation gap refers to the manual transformation of the operational objectives into a SON configuration based on the technical expertise. It is mainly concerned with the efforts for infrequent objective and technical changes which affect the whole network. These manual efforts are depicted as the yellow and blue cogwheels next to the operator in Figure 3.6.

Dynamics gap refers to the manual adaptation of the SON configuration in response to frequent context changes that affect single cells in the network. These manual efforts are depicted as the green cogwheel next to the operator in Figure 3.6.

Theoretically, the operational personnel needs to manually derive the SON configuration every time any of the three events happens as indicated by the marks on the SON configuration time line in Figure 3.6. Since these events are not synchronous, e.g., the operational objectives do typically not change in parallel to the technical expertise, this results in very frequent adaptations of the SON configuration and

considerable manual efforts. As a result, an initially deployed SON configuration is seldom adapted. Instead, the operational personnel monitors the network performance and only adapts the configuration of the SON if severe performance problems are detected. Thereby, possible optimization gains from an improved configuration are missed.

3.2 Goals and Requirements

The goal of ODSO, as stated in Objective 1, is to provide holistic and autonomic operations of a mobile network by extending the automatic operations concept of SON to enable human operators to control it directly with their objectives. SON functions can be seen as automatic control loops (see Chapter 2.2) that optimize the network according to some fixed values for low-level configuration parameters. However, they have no ability to understand the operator objectives and adapt their behavior accordingly. ODSO aims to close the manual gap of SON operations by extending SON such that the SON functions are autonomically operated according to the objectives.

The envisioned result is shown in Figure 3.7. The manual task of SON operations is automated by ODSO. It sits on top of the SON functions and controls them. Thereby, ODSO is driven by two distinct and clearly separated models: the objective model formalizing the operational objectives and several technical models that formalize the technical expertise. This separation allows independent evolution of both models. It is important to notice that ODSO does not aim to optimize network performance, which is done by specialized SON functions, but to use the SON functions more efficiently with respect to the diverse operator objectives.

At its core, the autonomic transformation of the operational objectives into actions to control the SON is what lifts automatic network operations by SON to autonomic network operations by ODSO (see Chapter 2.2). Whereas SON automated the execution of operations procedures, as can be seen by comparing Figure 3.1 and Figure 3.2, ODSO autonomically steers the SON towards satisfying the given operational objectives by applying the technical expertise, as outlined in the differences between Figure 3.2 and Figure 3.7. Figure 3.8 visualizes this by mapping network operations on the AC MAPE loop (cf. Figure 2.5). Whereas the automatic SON functions perform most of the monitor, analyze, plan, and execute work, they barely have any detailed knowledge about the network, i.e. technical expertise, and do not understand the policy, i.e., the operational objectives. ODSO should be wrapped around SON providing a utility function-based PBM interface to the operator that enables autonomic operations. Hence, it has the network knowledge and knows the objectives to steer the SON functions in a way that is desired by the MNO.

In order to provide autonomic SON operations, it is first necessary to formalize the inputs such that they can be automatically processed. The ODSO concept defines a machine-readable model of the operational objectives, referred to as *objective model*, that allows to:

- express objectives that are defined over network cell-related KPIs and allow

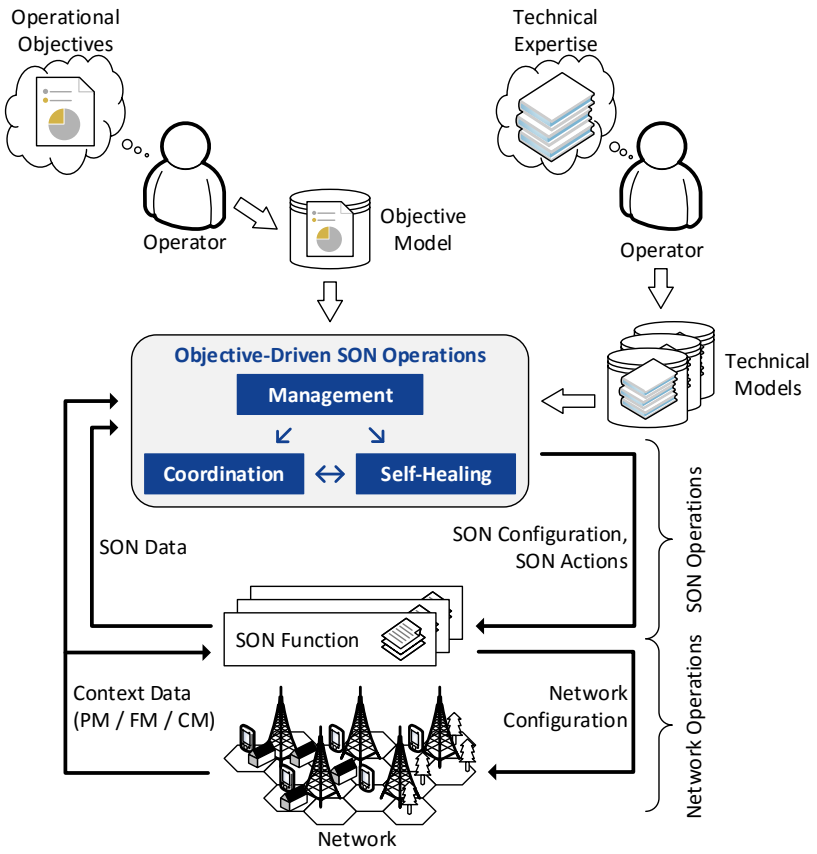


Figure 3.7: ODSO-enabled network operations: the human operator provides an objective model and technical models and the Objective-Driven SON Operations system performs autonomic SON operations accordingly.

the definition of required and desired performance targets,

- express trade-off preferences among the KPI-related objectives, and
- make the KPI-related objectives and trade-offs preferences dependent on the operational context.

Furthermore, the technical expertise needs to be captured in several machine-readable *technical models* which allow handling the inherent uncertainty and indirection in network and SON operations. These models are typically provided by both the vendors of the SON functions and the MNO and will be used to:

- automatically derive the expected effects of a SON function on the performance of a network cell for SON management,
- automatically detect conflicts among SON functions and determine execution constraints for SON coordination, and

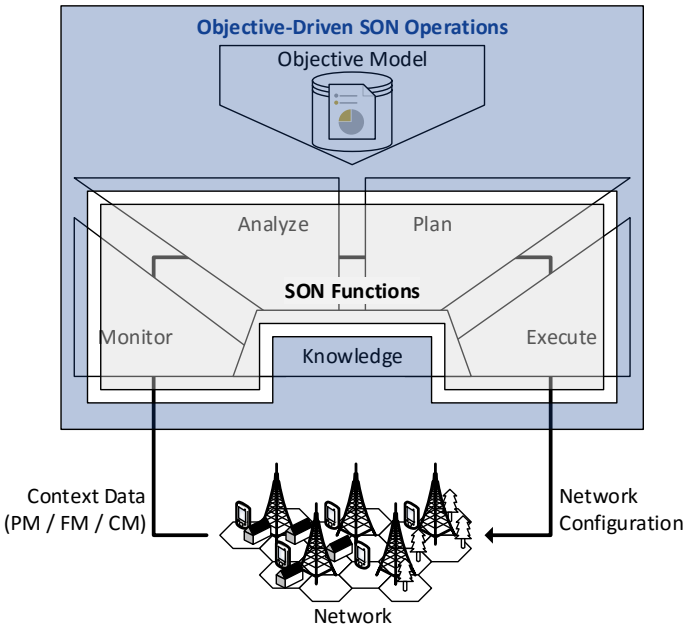


Figure 3.8: ODSO enables autonomic operations by allowing the operator to control the SON with utility-function policies (adapted from [KC03]).

- automatically determine possible recovery actions for failures as well as their effects in the scope of SON self-healing.

Based on the formalized objective and technical models, the ODSO concept proposes to integrate and automate the management, coordination, and self-healing of a SON-enabled network as outlined in Figure 3.7. That is, the translation of the operational objectives into a SON configuration, SON coordination actions, and SON self-healing actions is automated. It is important to notice that the approach does not require any adaptation of the SON functions, e.g., more complex algorithms. Instead, ODSO is supposed to be placed as an extension over existing SON implementations. In the end, ODSO simplifies general SON operations considerably since the operators can operate the network solely with their objectives instead of a low-level SON configuration. In that way, the objective model also provides a unified interface for SON operations that overcomes the multitude of vendor-dependent SON function interfaces. As a result, the adaptation of the SON to changes in the objectives or the network is accelerated and the required efforts reduced. Thus, the operational personnel is relieved from general SON operations tasks and can concentrate on more challenging tasks, e.g., managing the evolution of the mobile network. In this way, the manual gap of SON operations can be closed leading to less manual efforts and OPEX.

3.3 Architecture

This section presents the architecture of the ODSO approach, i.e., the components that build it up and their integration with a SON-enabled network. Therefore, a formalization and the basic assumptions about the underlying network and SON is outlined first.

3.3.1 Mobile Network

3.3.1.1 Network Cells

Chapter 2.1.1 shows that a modern mobile network is made up of numerous complex entities which need to work together in order to provide connectivity and services to the users. The ODSO approach is focused on the RAN since this is the focus of SON. The RAN is made up of numerous network cells, each defined as the coverage area of a specific antenna that is controlled by a BS. Although, a BS typically controls several antennas and, thus, cells, each of them is independently managed.

The network cells are the single, basic location tags for SON operations in this thesis, i.e., the SON function instances are managed and coordinated, and problems are healed considering the cell on which each of them occurs. Thereby, the term network cell refers to both macro cells and small cells. Nevertheless, each small cell is assigned to the macro cell in whose coverage area it is placed. This is reasonable as smalls cells are typically deployed to support a macro cell in serving busy areas.

Definition 3.1 (Network cell)**.** C is the set of cells in the mobile network. $|C|$ refers to the number of cells. A variable cell is denoted as $c \in C$.

For instance, the network used in the evaluation presented in Chapter 7.2.1 consists of 35 cells. Hence, $|C| = 35$ with $C = \{\text{Cell } 1, \ldots, \text{Cell } 35\}$. Thereby, Cell 33, Cell 34, and Cell 35 are small cells that are assigned to Cell 6.

3.3.1.2 KPIs

The performance of the mobile network is evaluated using a set of KPIs which are supposed to give an insight into specific performance aspects of the mobile network. For instance, using the CQI, a human operator can get an overview about the signal quality in the network.

It is assumed that the KPIs are standardized over the whole network with respect to their syntax, i.e., the domain of possible values of a KPI, and their semantics, i.e., the meaning of the values of a KPI. If this is not the case then the KPI values might be transformed into such a common definition, e.g., through data integration approaches (see [Len02]).

Definition 3.2 (KPI)**.** K represents the set of KPIs that are considered by the human operator for SON operations. $|K|$ refers to the number of KPIs. Each KPI $k \in K$ has an associated domain $\text{Dom}(k)$ of possible values $v \in \text{Dom}(k)$.

In this work, the examples focus on the $|K| = 4$ KPIs which are represented by the set $K = \{\text{CQI}, \text{Pipo}, \text{Load}, \text{Energy}\}$:

Channel Quality Indicator (CQI) is the weighted harmonic mean of the CQI channel efficiency [3GP15c] which is a value between 0.0 and 5.5547. It is an indication of the average signal quality of the cell for the served UEs. The CQI values are in the range $\text{Dom}(\text{CQI}) = [0.0, 5.5547]$ whereby the higher the values, the better.

Handover ping-pong rate (short form used in formula, code, and figures: Pipo) for a network cell is the maximum of the handover ping-pong rates of all neighbor relations, i.e., the ratio between the sum of the number of handover ping-pongs and the sum of the number of all handover attempts for each neighbor relation. It is an indication for the efficiency of the handover procedure. The values are in the range $\text{Dom}(\text{Pipo}) = [0.0, 1.0]$ whereby the lower the values, the better.

Cell load (short form used in formula, code, and figures: Load) is based on the average utilized Physical Resource Blocks (PRBs) of the cell. It is an indication of the amount of spare resources that the cell can use to serve additional UEs. The values are in the range $\text{Dom}(\text{Load}) = [0.0, 1.0]$ whereby the lower the values, the better.

Energy consumption (short form used in formula, code, and figures: Energy) of a macro cell is the ratio of the active to the overall number small cells which are assigned to the macro cell. It indicates the small cell power consumption for the area covered by the macro cell. The values are in the range $\text{Dom}(\text{Energy}) = [0.0, 1.0]$ whereby the lower the values, the better. The energy consumption of a small cell or a macro cell without assigned small cells is by definition 0. For instance, a macro cell with 3 assigned small cells, of these 1 is active and 2 are inactive, has the energy consumption $1/3 = 0.33$.

3.3.1.3 Operational context

The operational context provides network-wide PM, CM, FM and operational data about the RAN. This may also comprise historical information over a limited time horizon, e.g., the context might not solely provide the current PM data, i.e., KPI values, but also PM data from the last 24 hours. Based on this information, the ODSO system can determine the applicable actions and their correct effects as well as the applicable objectives. Examples of properties in the context are:

Performance Management (PM) data comprises the values of the KPIs for each network cell.

Configuration Management (CM) data comprises the configuration of the network cells, as well as fixed properties like its location, e.g., urban or rural area, its type, e.g., macro cell or micro cell, and its RAT, e.g., LTE or UMTS.

Failure Management (FM) data comprises raised alarms for a network cell.

Operational data comprises the current time and date, including information like day of the week.

As can be seen, the context comprises lots of information about the network that can be used by SON operations. It can be retrieved from the respective data bases that are either distributed over all BSs and contain the BS-specific information, or from centralized data bases.

The following formal presentation of the generic ODSO component design solely requires the PM data of all cells, i.e., the KPI values. In current network operations, the KPIs are usually evaluated on an aggregated level. This aggregation is spatial, i.e., the values from the cells are combined for a cell group of usually 10 to 100 cells, and temporal, i.e., the values of several granularity periods are combined for a time period like a whole day. The concrete combination procedure is thereby dependent on the KPI and may be an average, an extreme value, or the value of some percentile. This aggregation is necessary because human operators who currently manage the network are limited and cannot evaluate each single cell in every granularity period. In contrast to that, autonomic ODSO allows automated single-cell operations. Hence, all RAN-related KPIs considered for SON operations are measured and computed at each granularity period for each network cell individually, i.e., each network cell has its own value for each KPI at each period.

Definition 3.3 (Operational context)**.** An operational context $\mathbf{x} : C \times K \to \bigcup_{k \in K} \mathrm{Dom}(k)$ is a mapping from a network cell $c \in C$ and a KPI $k \in K$ to the context KPI value $v \in \mathrm{Dom}(k)$ of k in c. Typically, $\mathbf{x}$ refers to the *current* operational context. Hence, $\mathbf{x}(c, k) = v$ means that v is the current value of k in c. Notice that the context value for a KPI is always from the respective KPI domain, i.e., $\forall c \in C, k \in K. \mathbf{x}(c, k) \in \mathrm{Dom}(k)$. The set of all possible contexts is $\mathbf{X}$.

For simplicity, $\mathbf{x}_c : K \to \bigcup_{k \in K} \mathrm{Dom}(k)$ refers to the operational context of the network cell c, i.e., it is a mapping for the values of all KPIs in the cell c. It is a curried version of $\mathbf{x}$ with $\mathbf{x}_c = k \mapsto \mathbf{x}(c, k)$.

The formalization of the context which is used throughout this work focuses only on the current PM data, specifically the KPI values from the last granularity period. For instance, the context value $\mathbf{x}(\text{Cell 1}, \text{CQI}) = \mathbf{x}_{\text{Cell 1}}(\text{CQI})$ represents the value of the KPI CQI for the Cell 1.

3.3.2 SON Functions

In this work, mainly self-optimization SON functions are considered, since these have a direct impact on the performance of the mobile network and, hence, directly contribute to the satisfaction of the operator objectives. In contrast to this, SON functions from the self-configuration functional area are mainly concerned with the initial setup of new BS [San+11]. Hence, these functions are typically only executed once during BS setup, often in a fixed order referred to as workflow [Rom+10]. Furthermore, the goal of this kind of SON functions is to bring new BSs in an operable state. This state, i.e., the initial configuration of the BS is thereby determined

beforehand using network simulators. Hence, their configuration solely depends on technical constraints, e.g., the core network architecture and mobile network topology. For this reason, these SON functions are not considered in ODSO.

The ODSO framework has the following assumptions regarding the SON functions:

Cell-focused means the SON functions are designed in a cell-specific way, i.e., an instance of a SON function is operating on a specific cell in the sense that it tries to optimize the performance of this cell. However, during its execution, it might also consider data from other cells, e.g., MLB might consider the cell load of its neighbor cells. The implied consequences of this assumption are twofold: first, the SON functions can be configured for each and every cell differently and, second, configuration requests and alarms by the SON functions can be assigned to a single, specific cell.

Configurable means the algorithmic decision logic of the SON functions can be configured through a set of parameters. The set of parameters is specific to a SON function and may be different to similar SON functions provided by other vendors. For instance, an MLB SON function might provide parameters to configure the upper and lower load threshold, the upper and lower CIO, a step size, and a load averaging time (see [FLS14a]). A configuration value for a parameter can be, e.g., 40% for the lower load threshold. An SON Function Configuration (SFC) is a set of configuration values, one for each parameter of the function.

Coordination-aware means the SON functions continuously monitor the network cell and determine whether an optimization is required. However, if a SON functions needs to change the configuration of a cell, it requests permission for this at the SON coordinator. This allows the SON coordinator to reject the change if the request is in conflict with other, parallel requests (see pre-action coordination in Chapter 2.1.5). Upon a rejection of a request, the SON function starts the monitoring again and might request the same change later on again.

Synchronized means the execution of the SON functions is synchronous, periodic, and aligned with the granularity periods of the PM data in the network. That is, the network management system collects the measurements from the BSs in regular time intervals, calculates the values of the KPIs based on them, and triggers the execution of all SON function instances. The SON functions analyze the PM data and, if necessary, request a change of the configuration of a cell in the network. This process is outlined in the lower two swim lanes in Figure 3.10.

Failure-aware is an optional property of the SON functions that is useful for SON self-healing (see Chapter 6.3.1.2). If this functionality is available, the SON functions can inform the ODSO system about problems they have detect during their execution via alarms.

These assumptions are also often used in related work regarding SON operations, e.g., [Ban+11b][Kür+10][Ben+13a][RSB13][Iac+14b]. Especially the synchronized execution of SON functions is quite realistic for a centralized SON. In order to keep the overhead of operations low, PM data is often only distributed as aggregates at the end of a granularity period. Furthermore, this aggregation allows us to abstract away the numerous details about the pre-processing, e.g., handling missing measurements, and statistical analysis, e.g., whether the measurements allow a statistical correct KPI computation, of the bare measurements.

Definition 3.4 (SON function)**.** S represents the set of SON functions that are active in the SON-enabled mobile network. $|S|$ refers to the number of SON functions. A particular SON function is denoted as $s \in S$.

Note that a SON function $s \in S$ must be distinguished from a SON function instance (see Chapter 2.1.4) since s is not localized, i.e., it is not associated with a specific cell.

The examples in the following conceptual chapters focus on $|S| = 4$ simplified SON functions, specifically $S = \{\text{CCO}, \text{MRO}, \text{MLB}, \text{ESM}\}$ which are presented in the following (see [Las+11] for some more complex SON algorithms):

Coverage and Capacity Optimization (CCO) aims at improving the signal quality and capacity for the UEs of a network cell. Therefore, it monitors the KPIs CQI and cell load which are indicators for the signal quality and capacity, respectively. If the value of the CQI drops below a threshold, then CCO adapts the RET of the cell's antenna, i.e., it tilts the beam up or down. Thereby, the functional dependency between CQI and the control parameter RET is not monotonic, i.e., depending on the physical characteristics, a low CQI might be improved by either increasing or decreasing the RET, respectively. If the value of the cell load rises above a threshold, then CCO attempts to activate small cells that may be able to take over some of the UE generated load. Note that some CCO approaches may also adapt the TXP of the cell's antenna, i.e., they increase or decrease the power.

Mobility Robustness Optimization (MRO) aims at improving the handover performance between a cell and its neighboring cells by minimizing unnecessary handovers, specifically so-called ping-pongs. A handover ping-pong happens if a UE is handed over to a neighboring cell and quickly afterwards is handed over back to the initial cell. This may be caused by the layout of the roads, e.g., if the vertex of a street curve is just extending into the neighbor cell whereas the sides are in another cell. MRO monitors the KPI handover ping-pong rate of a cell and, if the ping-pong rate is above a threshold, adjusts the virtual cell borders by increasing the CIO to problematic neighbors. Note that MRO may also monitor and optimize other KPIs like too-early or too-late handover rates.

Mobility Load Balancing (MLB) aims at reducing the load of a cell by virtually shrinking the cell such that UEs are handed over to neighboring cells. Therefore, it monitors the KPI cell load of a cell and, if it is above a given threshold,

reduces the CIO to neighboring cells. This is supposed to cause UEs that are at the border between the two cells to handover to the neighbors which also transfers their load to the neighbors.

Energy Saving Management (ESM) is switching off small cells that are not necessary in order to save energy. ESM monitors the KPI cell load of a small cell which is an indication of the activity in a cell. If the KPI drops below a given threshold, then ESM turns it off and, thus, reduces the KPI energy consumption of the assigned macro cell (see Chapter 3.3.1.2). Hence, the ESM function aims to reduce the energy consumption on a best-effort basis without actually monitoring this KPI. The ESM function is configured and executed solely on macro cells and controls their assigned small cells. Consequently, ESM has no effect in a macro cell without any small cells.

SON functions are typically not orthogonal to each other, i.e., they interfere and influence each other (see SON function conflicts in Chapter 2.1.5). Among the 4 considered functions, there are specifically the following relations:

CCO and ESM work against each other since CCO may turn on small cells, whereas ESM may turn them off. In principle, CCO may increase the energy consumption and ESM may increase the cell load of the macro cell.

MRO and MLB work against each other since MRO may increase the CIO, whereas MLB may decrease it. In principle, MRO may increase the cell load and MLB may increase the handover ping-pong rate.

All might influence each other since their changes of the network configuration may affect other KPIs. For instance, if MRO virtually increases the cell size by shifting the virtual cell border, this may reduce the overall CQI in the cell as more distant UEs need to be served.

3.3.3 SON Operations

The ODSO system is supposed to automate all tasks necessary to operate a SON-enabled mobile network such that the objectives of the MNO are satisfied as much as possible. Therefore, the ODSO architecture, depicted in Figure 3.9, consists of three functional components that perform the necessary tasks.

SON management component configures the SON (see Chapter 4). This comprises, on the one hand, setting the SON function configuration for all SON functions and, on the other hand, providing the estimated SON function effects to the other two ODSO components. The latter is a model of the expected effects of the SON function with the current configuration. The computation is based on SON function models which allow predicting the expected effects of possible SFCs.

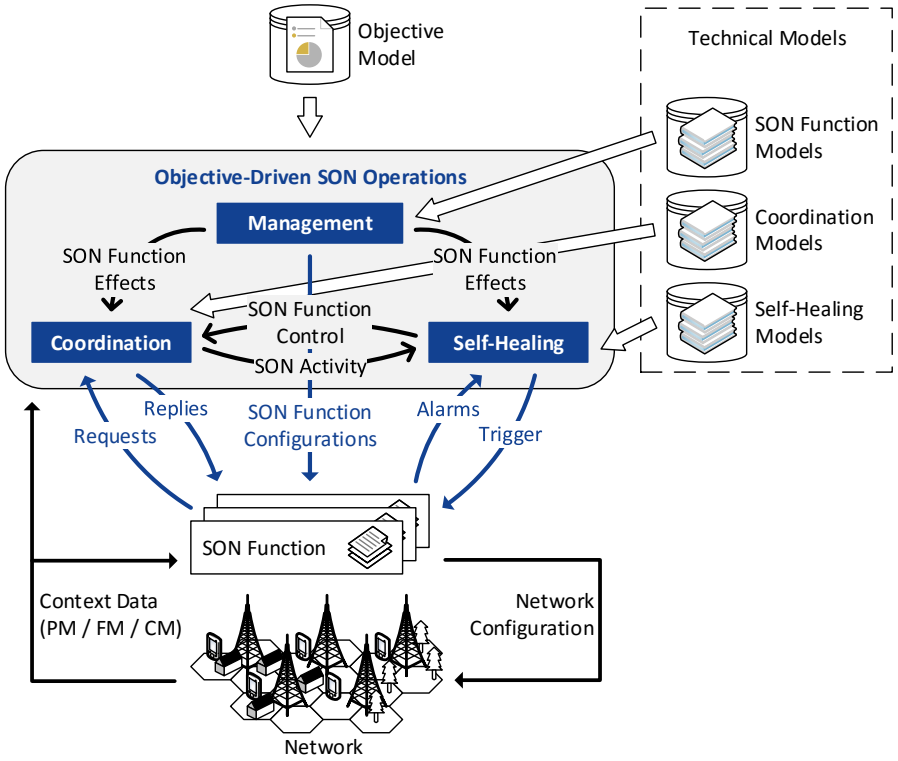


Figure 3.9: Overview of the architecture of ODSO.

SON coordination components aims at detecting conflicting network configuration changes by the SON functions and resolving them by selectively blocking conflicting changes (see Chapter 5). This means that a SON function requests the permission for a reconfiguration of a network cell and SON coordination either accepts or rejects it. The computation is based on coordination models, specifically a conflict detection model that allows identifying possible conflicts, a coordination constraint model that defines technical constraints for conflict resolution, and the SON function effects provided by SON management. Furthermore, SON self-healing is informed about the activity of the SON functions in order to detect anomalies in their behavior that might indicate failures.

SON self-healing component detects and mitigates uncommon problems that go beyond the capabilities of the SON functions (see Chapter 6). These can be hardware and software failures of the BSs but also rare environmental conditions that render the SON functions ineffective and even counterproductive. Therefore, it may control the execution of the SON functions by directly triggering them and adjusts SON coordination accordingly. The computation is based on self-healing models and the SON function effects provided by SON management. The former allow detecting and diagnosing problems based on monitored data from the network and alarms from the SON, as well as the determination of recovery actions.

Each of the three components draws on specific models that formalize the technical expertise necessary for the respective task. These models are supposed to be provided by both the vendors of the SON functions and the operational personnel. Whereas the former may mainly provide the SON function models, the latter can provide MNO-specific procedures and recovery actions. Nevertheless, from the operator point of view, ODSO is primarily steered through the formalized representation of the objectives in the objective model. Thereby, all three components base their computation on the same objective model. Apart from this, they are provided with PM, CM, and FM data in order to monitor the network.

Figure 3.10 shows the architecture from a run-time point of view and outlines the chronological activity of the three main components. The network collects measurements and provides the derived KPI values in regular intervals named granularity periods. The SON function execution is synchronized with this period such that the functions are triggered right after a granularity period has ended and a new set of KPI values is available. It can be seen that the management, self-healing, and coordination component are not executed in the same intervals: First, the coordination component is executed periodically after the execution of the SON functions. Second, the self-healing component is constantly monitoring the network and SON for problems, and, once a failure is detected, the diagnosis is performed on-demand and appropriate SON functions are triggered. Third, the execution of the management component is triggered in longer time intervals whenever the reconfiguration of the SON might be necessary, e.g., twice a day to adapt the SON to night and day respectively.

3.4 Generic Component Design

ODSO is not solely an architecture which structures the tasks related to SON operations into three components, but also proposes a framework for decision making in each component. Its main goal is to automate the integration of the technical expertise for the respective task with the overall operational objectives. Hence, the technical models needed for determining possible courses of action need to be separated from the operational objectives in the objective model, which should be accomplished with the actions. Based on this, the ODSO framework performs a two-step decision making process as depicted in Figure 3.11:

Action proposal evaluates the technical models in the operational context. The result of this step is a set of technically feasible actions in the current operational context along with the effects these actions might produce. Thereby, the effects are described in terms of stochastic KPI distributions, i.e., the expected KPI values after the execution of the action. A possible implementation of the action proposal can be an action policy system that determines the feasible actions and their effects depending on the operational context using condition-action rules.

Action selection performs a conflict resolution between the possible actions, since usually not all can and should be executed at the same time. Often, only

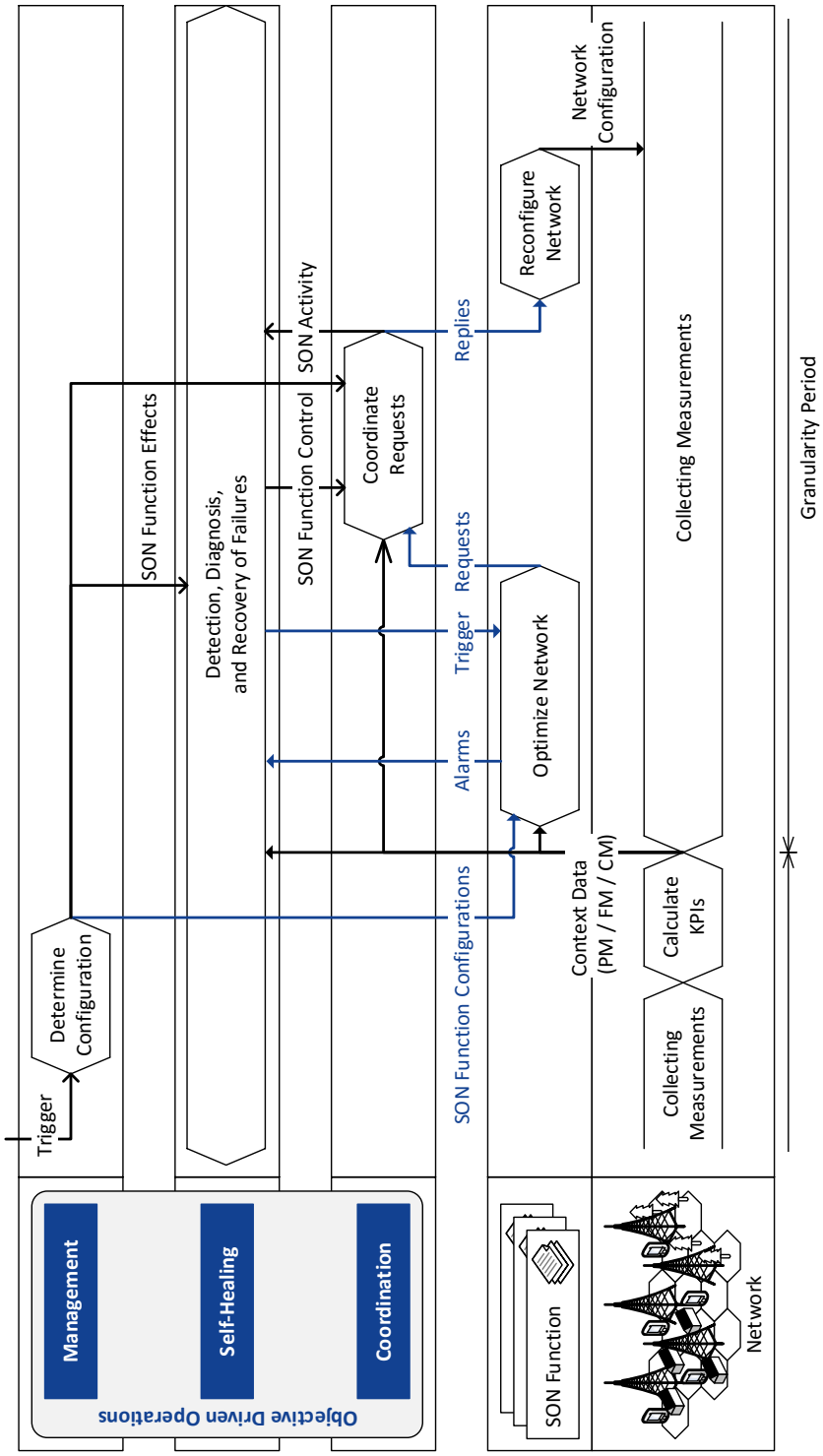


Figure 3.10: The run-time execution of ODSO (the arrow and component colors match Figure 3.9).

one single action should be executed. However, the conflict-resolution can also be more complex such as in SON coordination presented in Chapter 5. Therefore, this step evaluates the action effects with respect to the operator objectives and selects a conflict-free subset of the actions which maximizes the satisfaction of the objectives. Thereby, the objective model encodes the MNO's preferences over the KPIs in a specific operational context.

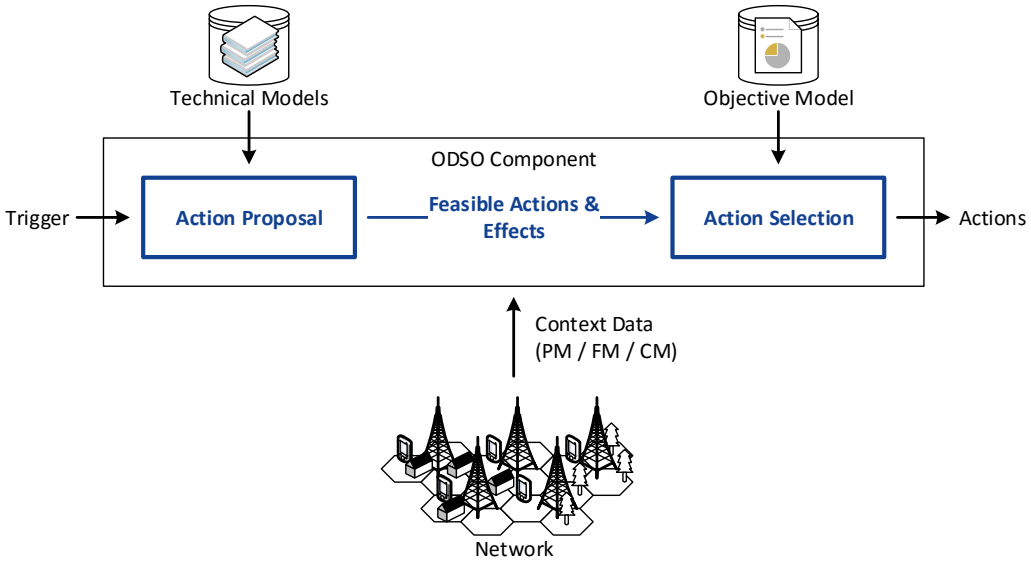


Figure 3.11: The two-step decision making process in ODSO. The steps of the generic ODSO component design are drawn in blue.

ODSO is a PBM system (see Chapter 2.2) based on knowledge, i.e., the technical models, and driven by a utility-function policy, i.e., the objective model. In this way, it enables autonomic operation of the SON. Technically, ODSO can also be seen from a different point of view: ODSO is a PBM system based on a conflict-containing action policy, i.e., the technical models, and an objective-driven conflict resolution. Both are valid views on the concept that help understanding ODSO.

In the following, the generic design of the ODSO components is presented. The action proposal can only be abstractly outlined since concrete implementations strongly depend on the implemented SON operations task. Because these three tasks are quite diverse, the specific implementations are presented in the sections presenting the respective tasks. However, the action selection, specifically the valuation of the feasible actions with respect to the operational objectives, is common among the tasks and, hence, presented in detail here.

3.4.1 Technical Model

The technical model contains the technical expertise of the MNO and the SON function vendors. This means that the technical models enables the action proposal to

determine applicable actions and their expected effects on the network performance with respect to a specific operational context. Thereby, the effects are usually not deterministic and, hence, a probabilistic representation is adopted.

3.4.1.1 Actions

The actions that the ODSO can perform depend on the implemented SON operations task, i.e., SON Management, SON coordination, or SON self-healing. For instance, SON management considers the configuration of the SON functions as an action whereas SON coordination sees the acknowledgment of a SON function execution request as an action. For the presentation of the generic component design, we adopt a generalized definition: the actions need to be atomic in the sense that they can be executed alone and then produce some effect.

It is important to distinguish between two concepts: the abstract method of performing some task and the instantiation of this method such that the task is performed on some concrete entity. In the ODSO concept, the former is referred to as a procedure whereas the latter is referred to as action. Specifically that means that, first, it is not defined on which network cell a procedure is executed, and, second, an action combines a procedure and the network cell it is performed on. For instance, the configuration of an MRO function with a specific SFC is a procedure, however, this specific configuration applied to a specific network cell is an action. Note that all SON functions and, consequently, all actions in the ODSO concept are focused on a specific cell as described in Chapter 3.3.2.

Definition 3.5 (Action)**.** $\tilde{A}$ represents the set of all possible procedures that an ODSO system can perform. A particular procedure is denoted as $\tilde{a} \in \tilde{A}$. $\tilde{a}$ contains no information where, i.e., on which network cell, it is executed.

An action $a = (\tilde{a}, c)$ with $a \in A$ such that $A = \tilde{A} \times C$ is an instantiation of the procedure $\tilde{a} \in \tilde{A}$ that should be executed on cell $c \in C$.

For simplicity, we define projections on A as $\text{proj}_{\tilde{A}}(a) = \tilde{a}$ and $\text{proj}_{C}(a) = c$ if and only if $a = (\tilde{a}, c)$.

3.4.1.2 Effects

The effect of an action describes the expected KPI values after the execution of the action in a probabilistic way. Therefore, for each KPI $k \in K$, we define V_k as a probabilistic variable over the possible values $\text{Dom}(k)$ of k. The effect represents the joint probability distribution $\Pr\left(\bigcup_{k \in K} V_k\right)$ over the random variables for all KPIs. Specifically for this thesis, $\Pr\left(V_{\text{CQI}}, V_{\text{Pipo}}, V_{\text{Load}}, V_{\text{Energy}}\right)$ is the joint probability distribution over the possible future values of the KPIs CQI, handover ping-pong rate, cell load, and energy consumption. Furthermore, $\Pr\left(V_{\text{CQI}} \in W\right)$ denotes the probability that the value for the CQI will be within a value range $W \subseteq \text{Dom}(\text{CQI})$ in the future.

The KPIs are assumed to be probabilistically independent of each other, hence, the joint probability can be calculated based on the marginal probabilities for each single

KPI as $\Pr\left(\bigcup_{k \in K} V_k\right) = \prod_{k \in K} \Pr(V_k)$ [Geo13]. Hence, we can define a univariate probability distribution for each KPI independently of the other KPIs instead of defining a multivariate distribution over all KPIs together. This assumption makes the following computations tractable, as explained in Chapter 3.4.4. Consequently, the effect of an action can be represented as a collection of continuous probability density functions one for each KPI. So, an effect can be seen as a probabilistic system state, i.e., the probability density functions expresses that an action has an impact on the KPI and the probability that the value of a KPI is within some range of the KPI values. Since the domains of the KPIs are continuous, the probability distributions are represented by continuous probability densities.

Definition 3.6 (KPI effect)**.** A KPI effect $f_k : \mathrm{Dom}(k) \to \mathbb{R}^+$ for some KPI k is a probability density function over the domain $\mathrm{Dom}(k)$ of $k \in K$, with $\mathbb{R}^+$ denoting the set of positive real numbers including zero, i.e., $\mathbb{R}^+ = [0, \infty)$. f_k is defined as a probability distribution over the random variable V_k for k, i.e., the probability that the future value of k will be in the range $W \subseteq \mathrm{Dom}(k)$ is $\Pr(V_k \in W) = \int_W f_k(v)\,\mathrm{d}v$. $F_k = [\mathrm{Dom}(k) \to \mathbb{R}^+]$ denotes the function space of all probability density functions for a $k \in K$. As for each probability density function, the probability densities must sum to 1, i.e., $\int_{\mathrm{Dom}(k)} f_k(v)\,\mathrm{d}v = 1$.

Definition 3.7 (Effect)**.** An effect $\mathbf{f} : K \to \bigcup_{k \in K} F_k$ is a mapping from a KPI $k \in K$ to its respective KPI effect $f_k \in F_k$ for the very same k. The set of all possible effects is denoted as $\mathbf{F}$.

Figure 3.12 depicts four exemplary probability distributions of an action on the KPI cell load. The uniform distribution in blue represents the expectation that the cell load will be less than 0.6 without any further knowledge, i.e., equally distributed. Hence, it is not expected that the value of cell load will greater than 0.6, e.g., $f_{\mathrm{Load}}(0.8) = 0$. However, below the threshold, the probability is uniformly distributed meaning that each value is equally likely, e.g., $f_{\mathrm{Load}}(0.5) = 1.67$. Similarly, the triangular distribution in green expresses the expectation that the cell load will be below 0.6. However, hereby lower values are less likely than higher values. Both can be seen as examples for an effect that has been defined by a human operator for an MLB function with the activation threshold 0.6. The triangular distribution is supposed to be more accurate for this example: it is more likely that the cell load is close to the threshold than close to 0 as the cell load, in general, is expected to be high and MLB only aims to keep it below the threshold. Furthermore, Figure 3.12 shows also a normal distributed KPI effect in red (with mean 0.5 and variance 0.1), which might be learned automatically from simulations. Similarly, the arbitrary distribution in cyan is a more complex example for a learned effect. Note that the maximum density values of the distributions can be greater than 1 since the integral of each, i.e., the area below the curve, must sum up to 1.

Figure 3.13 provides an example, how two KPIs effects are composed into a joint probability distribution over the two KPIs as the product of the marginal probabilities. Thereby, the graphs on the edges show the marginal KPI probability distributions and the graph in the middle shows the joint distribution.

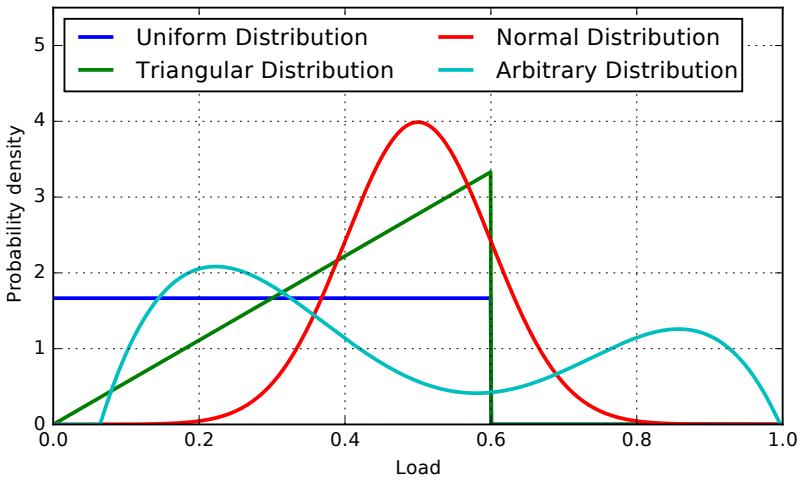


Figure 3.12: Exemplary uniform, triangular, normal, and arbitrary distributed effect of an action on the KPI cell load.

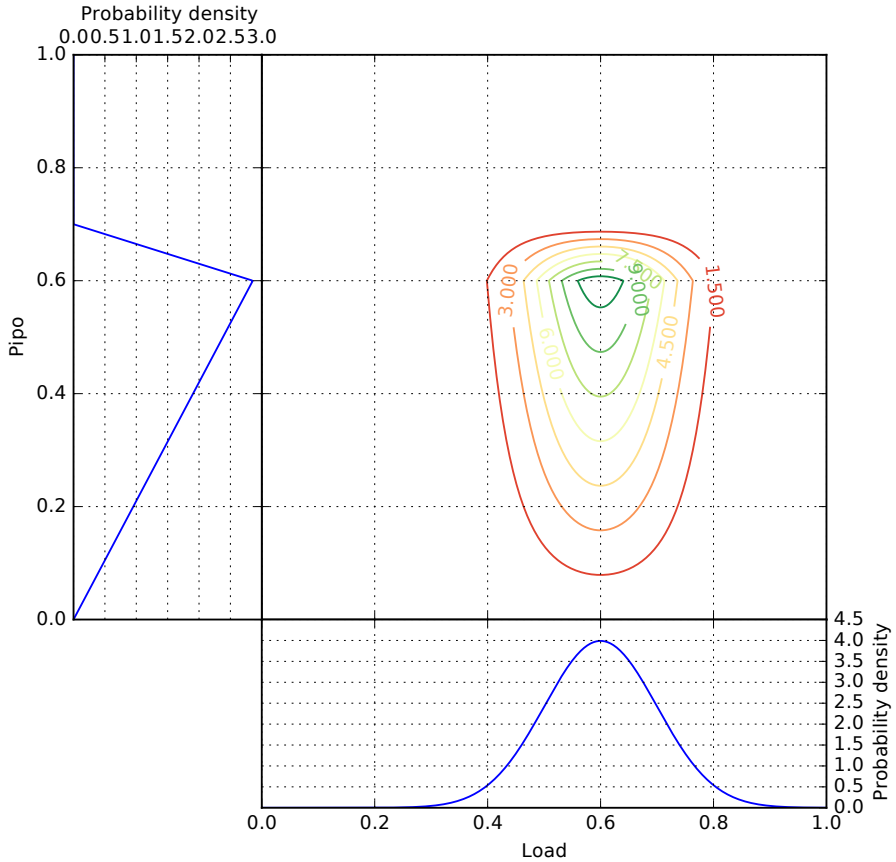


Figure 3.13: Contour plot of the three-dimensional, exemplary effect of an action on the KPIs cell load and handover ping-pong rate.

An effect $\mathbf{f}$ is referred to as *complete*, i.e., for each KPI $k \in K$ a probability density for the expected KPI values f_k is defined. However, an action must not necessarily affect all KPIs, e.g., an MRO function has, in general, no influence on the KPI energy consumption. In such cases, the action has a *partial effect*.

Definition 3.8 (Partial KPI effect)**.** A partial KPI effect $f_k^\perp : F_k \cup \{\perp\}$ for some KPI k is either a complete KPI effect $f_k : F_k$ or $\perp$ that denotes no effect on k. $F_k^\perp = F_k \cup \{\perp\}$ denotes the set of all partial KPI effects. Hence, the domain of complete KPI effects is a subset of the domain of partial KPI effects, i.e., $F_k \subset F_k^\perp$.

Definition 3.9 (Partial effect)**.** A partial effect $\mathbf{f}^\perp : K \to \bigcup_{k \in K} F_k^\perp$ is a mapping from a KPI $k \in K$ to its respective partial KPI effect $f_k^\perp \in F_k^\perp$ for the very same k. The set of all possible partial effects is denoted as $\mathbf{F}^\perp$.

3.4.1.3 Representation

The implementation of a technical model depends on the actual use case it should be used in and the way it is elicited. In its most simple form, a technical model $\mathrm{TM} : C \times \mathbf{X} \to \mathcal{P}(A \times \mathbf{F}^\perp)$ represents a mapping from a network cell $c \in C$ and an operational context $\mathbf{x} \in \mathbf{X}$ to a set of pairs $(a, \mathbf{f}^\perp) \in A \times \mathbf{F}^\perp$ representing the applicable actions $a \in A$ and their effects $\mathbf{f}^\perp \in \mathbf{F}^\perp$. The implicit assumption is that $\mathbf{f}^\perp$ affects the network cell $\mathrm{proj}_C(a) = c$ on which a is executed. In principle, there are two means to gather the required information: either a human user types the information in or the system automatically gathers it using machine learning techniques (see [RN10] for a general overview).

If the model is developed by a human expert using his knowledge, then a reasonable representation can be an action policy containing rules of the form:

```
IF condition(cell, context) THEN action YIELDS effect
```

Thereby, condition(cell, context) is some condition on the cell $c \in C$ and the current operational context $\mathbf{x} \in \mathbf{X}$, action is an action $a \in A$, and effect is a partial effect $\mathbf{f}^\perp \in \mathbf{F}^\perp$. In principle, it is a condition-action-effect representation, i.e., if the precondition condition(c, $\mathbf{x}$) is true then the action a could be executed and will lead to effect $\mathbf{f}^\perp$. Consequently, the rules need to be conflict-free in the sense that one action is not proposed with two different effects. Otherwise, a conflict resolution approach is required, e.g., a combination of the effects as described in Chapter 4.3.3.2. However, this action policy must not be confused with the utility function policy that controls the ODSO components.

In case the effect model is automatically generated using machine learning, such a human readable representation is not necessary. Instead, the computer can directly learn, e.g., the coefficients of a linear function mapping from an action and a context to the expected partial effects.

In any case, it might be necessary to restrict the technical model, e.g., in order to ensure that specific regulatory requirements like maximum transmission powers of BSs are met. Such additional constraints can either be accounted for

- during the creating of the technical model by limiting the proposed actions in some context,
- before the execution of the action proposal by preprocessing the technical model and, e.g., removing violating rules, or
- during the execution of the action proposal by removing violating actions from the set of feasible actions.

3.4.2 Action Proposal

Using the technical models and the current operational context, the action proposal determines a set of applicable action instances and their respective expected effects, referred to as action-effect set. Consequently, each action has exactly one assigned effect.

Definition 3.10 (Action-effect set)**.** The action-effect set

$$\mathrm{AF} = \{(a, \mathbf{f}) \mid a \in A \text{ is an applicable action}, \mathbf{f} \in \mathbf{F} \text{ is the expected effect of } a\}$$

is the set of pairs of applicable actions a and their respective complete effects $\mathbf{f}$. For a pair $(a, \mathbf{f}) \in \mathrm{AF}$, $\mathbf{f}$ is the expected resulting effect on the network cell $\mathrm{proj}_C(a) = c$ that a is executed on. That is, $\mathbf{f}$ represents the probability distribution $\Pr\left(\bigcup_{k \in K} V_k \mid a\right)$.

The technical model typically defines partial effects, however, the action-effect set AF contains solely complete effects. It is one of the tasks of action proposal to transform these partial effects into complete effects by using the current network performance, i.e., the expected complete effect $\mathbf{f}$ if the action a with the corresponding partial effect $\mathbf{f}^{\perp}$ is executed in the current operational context $\mathbf{x}$. The result is that all affected KPIs of $\mathbf{f}^{\perp}$, i.e., all $k \in K$ with $f_k^{\perp} \neq \perp$, will take on the expected KPI densities whereas the values of the non-affected KPIs, i.e., all $k \in K$ with $f_k^{\perp} = \perp$, will stay the same as in $\mathbf{x}$. In this case, the KPI effects of the non-affected KPIs are not probabilistic but deterministic, i.e., there is exactly one KPI value known to occur. The Dirac distribution allows combining discrete and continuous probability measures [Geo13] by modeling a discrete, deterministic effect as a probability density function:

$$\delta_\xi(v) = \begin{cases} \infty & \text{if } v = \xi \\ 0 & \text{otherwise} \end{cases} \qquad \text{with} \int_{\mathrm{Dom}(k)} \delta_\xi(v)\,\mathrm{d}v = 1. \tag{3.1}$$

For instance, the effect density for the KPI k with the deterministic effect 0.6 is $\delta_{0.6}(v)$. This effect is shown in Figure 3.14a in green. As can be seen, a Dirac distribution is a single, infinitely high spike at ξ and everywhere else 0.

Using the Dirac distribution, it is possible to define a merging function that transforms a partial effect into a complete effect based on the context. Thereby, this merge

can be weighted probabilistically with ρ in case the partial effect of an action does materialize with some probability. The ρ can be seen as the probability that the action will be effective. Based on this, the probability that a value from the combined effect of the expected KPI effect $f_k^\perp \neq \perp$ and the Dirac distribution $\delta_{\mathbf{x}(c,k)}$ of the current KPI value $\mathbf{x}(c,k)$ will be in the value range $W \subseteq \mathrm{Dom}(k)$ is

$$\begin{aligned}\Pr(V_k \in W \mid f_k^\perp, \mathbf{x}(c,k), \rho) &= \rho \cdot \left(\int_W f_k^\perp(v)\,\mathrm{d}v \right) + (1-\rho) \cdot \left(\int_W \delta_{\mathbf{x}(c,k)}(v)\,\mathrm{d}v \right) \\ &= \left(\int_W \rho \cdot f_k^\perp(v)\,\mathrm{d}v \right) + \left(\int_W (1-\rho) \cdot \delta_{\mathbf{x}(c,k)}(v)\,\mathrm{d}v \right) \\ &= \int_W \rho \cdot f_k^\perp(v) + (1-\rho) \cdot \delta_{\mathbf{x}(c,k)}(v)\,\mathrm{d}v. \end{aligned} \tag{3.2}$$

Definition 3.11 (Effect merging function)**.** The merging function $\mu : \mathbf{F}^\perp \times C \times \mathbf{X} \times [0,1] \to \mathbf{F}$ transforms a partial effect into a complete effect for the cell $c \in C$ using the operational context $\mathbf{x} \in \mathbf{X}$:

$$\mu(\mathbf{f}^\perp, c, \mathbf{x}, \rho) = k \mapsto \mu_k(\mathbf{f}^\perp(k), c, \mathbf{x}, \rho)$$

such that

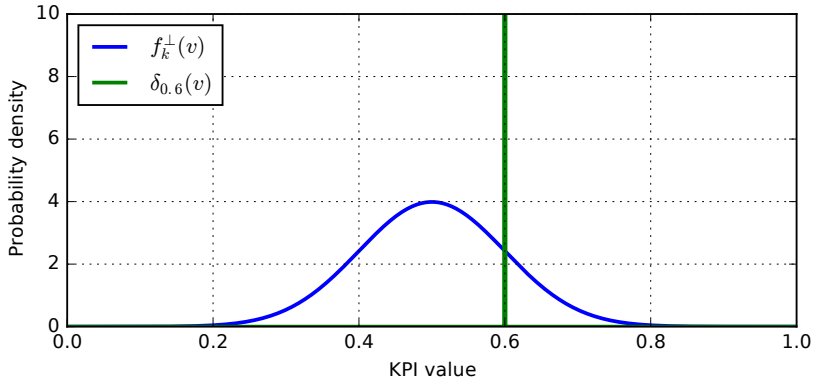
$$\mu_k(f_k^\perp, c, \mathbf{x}, \rho)(v) = \begin{cases} \rho \cdot f_k^\perp(v) + (1-\rho) \cdot \delta_{\mathbf{x}(c,k)}(v) & \text{if } f_k^\perp \neq \perp \\ \delta_{\mathbf{x}(c,k)}(v) & \text{otherwise.} \end{cases}$$

Thereby, the partial effect is supposed to materialize with the probability $\rho \in [0,1]$. As can be seen, $\mu_k(f_k^\perp, c, \mathbf{x}, \rho)$ represents the probability distribution $\Pr(V_k \mid f_k^\perp, \mathbf{x}(c,k), \rho)$ (see Equation 3.2) for $f_k^\perp \neq \perp$.

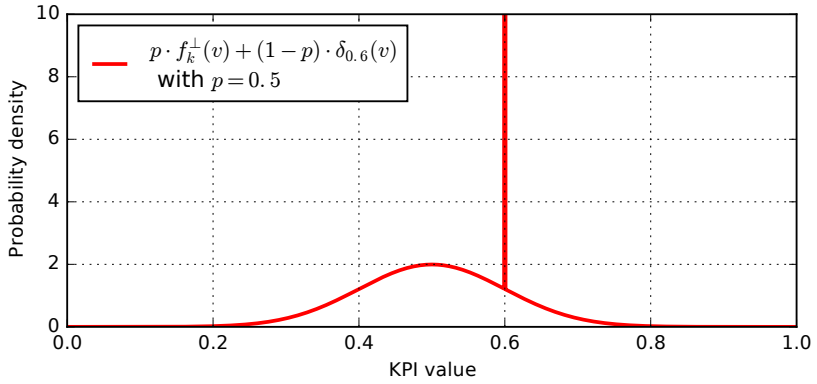
Figure 3.14 shows an example for merging an effect for the KPI k with probability $\rho = 0.5$. Figure 3.14a depicts a probabilistic KPI effect $f_k^\perp$, namely a normal distribution, in blue, and a deterministic KPI effect $\delta_{0.6}$ in green. Since $f_k^\perp \neq \perp$, i.e., there is a defined effect on k, the resulting merged distribution looks like the red plot in Figure 3.14b. As can be seen, the values of $f_k^\perp(v)$ are half as high since the probability that a specific v of $f_k^\perp$ materializes is $0.5 \cdot f_k^\perp(v)$. Similarly, the spike at 0.6 is also half as high, which, however, cannot be drawn. Notice that the result would be either exactly $f_k^\perp$ or $\delta_{0.6}$ if $\rho = 1.0$ or $\rho = 0.0$ respectively. One example use case with $0 < \rho < 1$ will be presented in Chapter 6: a network problem might be caused by a root cause with some probability ρ and, hence, the recovery of the root cause might produce a positive effect also with a probability ρ.

In the simplest form, the action proposal corresponds to a simple evaluation of the technical model TM for the network cell $c \in C$ in the current operational context $\mathbf{x} \in \mathbf{X}$ and a merging of the expected partial effects with $\mathbf{x}$, i.e.,

$$\mathrm{AF} = \{(a, \mathbf{f}) \mid a \in A, \mathbf{f} \in \mathbf{F}, (a, \mathbf{f}^\perp) \in \mathrm{TM}(c, \mathbf{x}), \mathbf{f} = \mu(\mathbf{f}^\perp, c, \mathbf{x}, 1.0)\}. \tag{3.3}$$



(a) The probabilistic action effect and the deterministic context effect.



(b) Resulting merged effect.

Figure 3.14: Visualization of the merging of an effect with the current context.

The following chapters describing the ODSO components will adapt the generic concepts of the technical model and the action proposal in order to fulfill the use case-specific requirements.

3.4.3 Objective Model

In ODSO, the execution of the SON is guided by operator objectives. MAUT provides a comprehensive and proven framework to analyze and make decisions in probabilistic environments. It is a natural solution to the decision problem encountered in ODSO: the system makes a decision between alternative actions based on the utility of their effects.

The objective model is a machine-readable representation of the operational objectives. As outlined in Chapter 3.1.1, there are three properties that the objectives should fulfill: they define context-dependent KPI targets that have a specific importance in order to allow trade-offs. This section presents a MAUT-based objective model that fulfills these requirements: prioritized utility functions define KPI targets, weights enable trade-offs between them, and context conditions allow context-dependency.

3.4.3.1 Utility Functions

MAUT is based on the definition of utility functions which map concrete values of attributes, i.e., in case of ODSO these are the KPIs, to a normalized measure representing the utility to the decision maker, i.e., MNO. The normalization of the KPI values to a range $[0, 1]$, with 0 representing the least preferred values and 1 representing the most preferred values, makes the different KPI measures commensurable, i.e., comparable, and enables the combination of the utilities of different KPIs.

Definition 3.12 (Utility function)**.** A utility function for a KPI $k \in K$ is defined by $o_k : \mathrm{Dom}(k) \to [0, 1]$.

The definition of the correct utility that accurately describes the preference of the MNO is essential for the system to make correct decisions. However, their elicitation is usually a difficult task. Hence, most approaches for multi-criteria decision making consider specific parametric functions and concentrate on the elicitation of the correct parameters. Thereby, a number of different utility functions are possible which have different properties (see [NGA08] for a comparison in the area of access network selection). There are four main types of functions as depicted in Figure 3.15. They are distinguished by their slope or marginal utility, i.e., the added utility by a slight increase in the underlying KPI value, at different regions of the KPI values. This change in the marginal utility can be interpreted as different risk attitudes of the MNO in uncertain situations [Win04, Ch. 13].

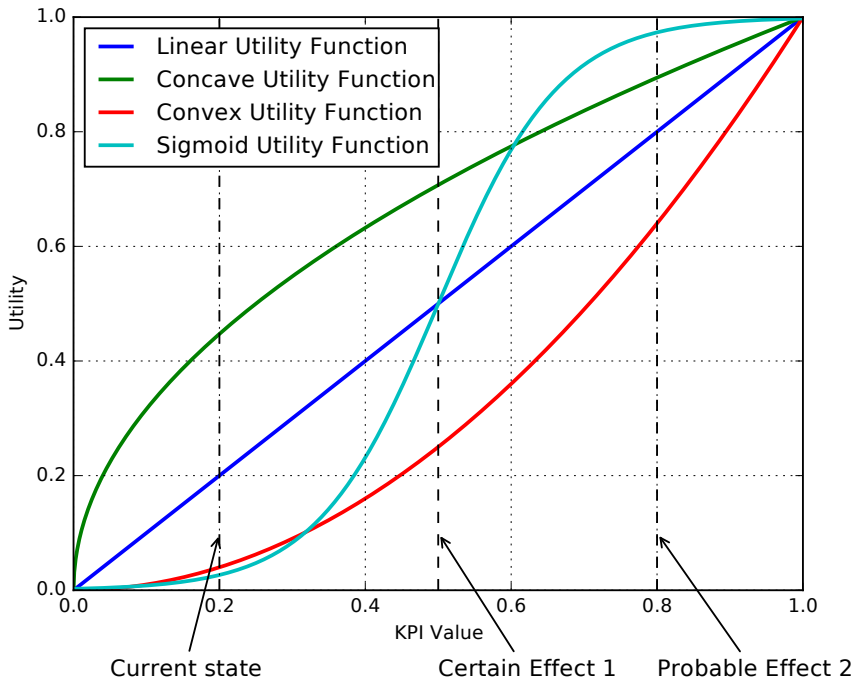


Figure 3.15: The four most common types of utility functions.

In order to present them, consider that the current performance of the KPI depicted in Figure 3.15 is 0.2 (left dashed line). Furthermore, there are two actions with two possible effects: Effect 1 represents the certain KPI value 0.5 (center dashed line) whereas Effect 2 represents the possibility to get a KPI value of 0.8 (right dashed line) with 50% probability and to stay at 0.2 with 50% probability. Note that the expected KPI value of both effects is 0.5.

Linear utility functions represent a constant marginal utility and expresses risk-neutral behavior. That is, the MNO is indifferent between Effect 1 and Effect 2 since the expected utility in both cases is 0.5. This is referred to as rational behavior.

Concave utility functions represent decreasing marginal utility and express risk-averse behavior. That is, the MNO prefers the deterministic Effect 1 with a utility of 0.71 to the probabilistic Effect 2 with an expected utility of 0.67. This means that the MNO is inclined to select an action whose effect slightly improves the KPI value with a high probability instead of an action whose effect strongly improves the value but with smaller probability.

Convex utility functions represent increasing marginal utility and express risk-seeking behavior. That is, the MNO prefers the probabilistic Effect 2 with a utility of 0.34 to the probabilistic Effect 1 with an expected utility of 0.25. This behavior is contrary to the concave utility function. It means that the MNO is inclined to select an action whose effect strongly improves the KPI value with a small probability instead of an action whose effect slightly improves the KPI value with a high probability.

S-shaped utility functions show an increasing marginal utility for low KPI values and a decreasing marginal utility for high values. Hence, it can be seen as a combination of the concave and convex utility function. As a result, the MNO is willing to take risks if the deterministic improvements are small but avoids risks if the certain effects are huge. Typically, the utility function is formalized using a sigmoid function.

3.4.3.2 Priorities

The definition of an accurate utility function is a difficult task. If they are not correctly set, then the system might focus on the improvement of the wrong KPI in a particular problem situation. However, network management as well as the reporting and evaluation by the operational personnel is currently mainly controlled by thresholds (see Chapter 3.1.1.1). These thresholds allow simple and comprehensible decision making and, so, provide some guarantee on the behavior of the system. With the definition of priority ranges on the KPIs, ODSO can mimic this comprehensible, goal-based behavior in order to ease objective model creation. The idea of priority ranges is twofold: KPI thresholds are transformed into priority ranges and utility functions define the operator preferences within each range.

Generic thresholds on KPI values are transformed into ranges of KPI values with different priorities. For instance, the threshold $v > 0.4$ for a KPI is translated into two priority ranges, $\{v \mid 0.4 < v\}$ and $\{v \mid v \leq 0.4\}$. Obviously, the former KPI range can be seen as the goal states regarding the threshold and the latter are the unacceptable KPI values. Since MNOs may like to define several thresholds per KPI, e.g., a minimal threshold for acceptable values and a stretching threshold for optimal values, the resulting priority ranges are ranked according to an assigned priority. Thereby, values of a range with higher priority are infinitely more preferred than the values of a range with a lower priority.

Definition 3.13 (Priority)**.** $D = \{1, \ldots, N_D\}$ is the totally ordered set of $N_D \in \mathbb{N}$ priorities under the normal $<$ order. Thereby, a priority $d_i \in D$ is more important than $d_j \in D$ if and only if $d_i < d_j$.

In principle, there can be any finite number of priority ranges defined. Given the requirements for the objectives, this work regards only three priority ranges, i.e., $N_D = 3$, and $1 \in D$ is more important than $2 \in D$ and $2 \in D$ is more important than $3 \in D$, which represent the following three identified priority ranges:

Unacceptable range defines the worst KPI values. That is, a value in this range is typically considered an error. This range has the priority 1.

Acceptable range defines the KPI values that are acceptable but that should be optimized. This range has the priority 2.

Optimal range comprises the optimal KPI values. This range has priority 3.

For a single KPI, the consideration of priority ranges is important for the evaluation of probabilistic effects, since the MNO prefers actions that maximize the probability to achieve values that satisfy the most important priority range first. For instance, consider Figure 3.15 with the certain Effect 1 and the probabilistic Effect 2 again, and suppose there is a threshold $v > 0.4$. Then the MNO, independently of the utility function, would prefer Effect 1 since it satisfies the priority range $\{v \mid v > 0.4\}$ with certainty whereas Effect 2 only has a 50% change for that. As a result, the definition of the priority ranges induces an infinitely risk-averse behavior. In other words, it is strictly more important to avoid KPI values below the threshold then it is to gain more optimal value above the threshold. This becomes even more obvious when considering several KPIs since the ranges are consistently defined for all KPIs, i.e., for each KPI the MNO defines the values for the same priority ranges. The important idea of priority ranges is then, that the MNO aims to satisfy the most important priority range for all KPIs before optimizing a specific, single KPIs. In other words, in contrast to general MAUT, the achievement of different priority ranges is not traded-off against each other.

In order to exemplify the concept of priority ranges, consider the two KPIs cell load and handover ping-pong rate depicted in Figure 3.16. Both are minimizing KPIs, i.e., their value should be as low as possible. For each KPI the three priority ranges have been defined by setting a threshold marking the minimum aspired KPI

value for each. The priority range below the threshold should be avoided since they are considered as a violation. The priority range above the threshold, however, is considered as good. Based on that, if two system states are compared, then the state which satisfies the higher priority range is seen as better. Consider the four depicted system states for these two KPIs marked with A, B, C, D:

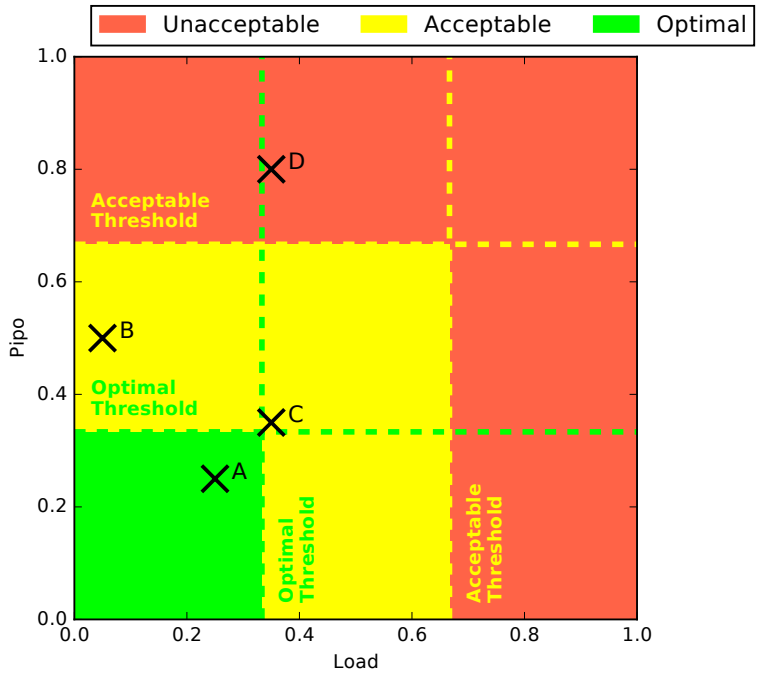
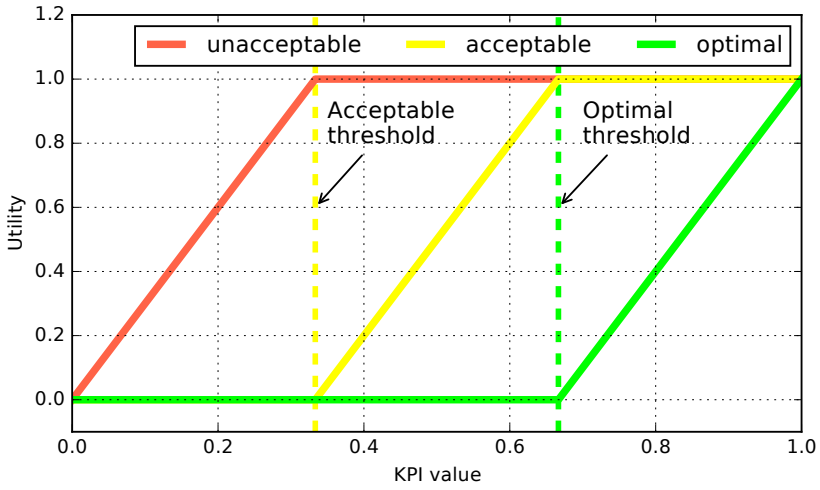


Figure 3.16: Example of three priority ranges for two KPIs.

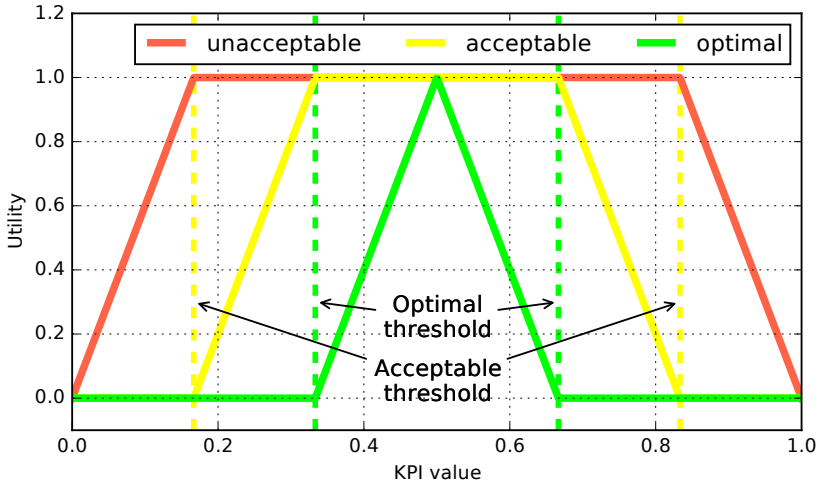
- The state A is the most preferred state, i.e., $A \gtrsim B, C, D$, because it is the only state that is in the optimal priority range for both KPIs. Note that even though state B is, compared with A, much better regarding the cell load and just a little worse regarding the ping-pong rate, A will always be preferred to B since the satisfaction of the priority ranges is strictly more important than the actual KPI values.
- The states B and C are more preferred than D, i.e., $B, C \gtrsim D$, because both satisfy the good state for at least one KPI. The preference order between B and C depends on the utility functions and weighting of the KPIs.
- D is the least preferred system state.

Within each priority range, utility functions define the preference of a specific system state as depicted in Figure 3.17. For each KPI $k \in K$ and each priority range $d \in D$, the MNO has to define a utility function $o_{k,d}$ which allows comparing the MNO preferences between two system states that are in the same priority range. Based on that function, the actual KPI values of a priority range are implicitly defined: a KPI value v is in a priority range if and only if its utility is greater than

0 and less than 1. Hence, the range for KPI $k \in K$ and priority d is given by $\{v \in \text{Dom}(k) \mid 0 < o_{k,d}(v) < 1\}$. As can be seen in Figure 3.17, the priority ranges are non-overlapping meaning that for each KPI value v, there is only one utility function with $0 < o_{k,d}(v) < 1$. For a specific utility function, on the one hand, KPI values with a utility of 0 are outside of the priority range and belong to a range with a higher priority, and, on the other hand, KPI values with a utility of 1 are outside of the priority range and belong to a range with lower priority.



(a) Maximization KPI objective.



(b) Achievement KPI objective.

Figure 3.17: Visualization of two KPI objectives, each with the three priority ranges.

Consider the maximizing KPI objective Figure 3.17a: there are the three priority ranges determined by three linear utility functions that are defined according to the acceptable and unacceptable threshold. One can see that, according to the definition, the unacceptable range covers KPI values $0 \leq v \leq 0.33$, whereas values $0.33 < v \leq 0.66$ and $0.66 < v \leq 1.0$ are considered acceptable and optimal respectively. In the same way, Figure 3.17b shows that this idea cannot solely be applied to KPIs

that should be maximized or minimized, but also to those for which a specific value should be achieved.

Based on that, we can define a KPI objective as a definition of the operator objectives regarding the KPI values.

Definition 3.14 (KPI objective)**.** A KPI objective $\mathbf{o}_k = (o_{k,1}, \ldots, o_{k,N_D})$ for a KPI $k \in K$ is a vector of N_D utility functions, one for each priority. The utility functions must fulfill two constraints: First, they must be ordered in the sense that the value of a utility function is always greater or equal to the lower prior utility functions, i.e.,

$$\forall v \in \mathrm{Dom}(k). \forall d_1, d_2 \in D, d_1 < d_2. o_{k,d_1}(v) \geq o_{k,d_2}(v).$$

Second, they must be non-overlapping in the sense that a KPI value must be in exactly one priority range, i.e.,

$$\forall v \in \mathrm{Dom}(k). \exists d_1 \in D. 0 < o_{k,d_1}(v) < 1 \wedge \\ \forall d_2 \in D, d_1 \neq d_2. o_{k,d_2}(v) = 0 \vee o_{k,d_2}(v) = 1.$$

The set of all possible KPI objectives for k is denoted $\mathbf{O}_k$.

Consequently, we solely consider KPI objectives of the form $\mathbf{o}_k = (o_{k,1}, o_{k,2}, o_{k,3})$ in this work. Furthermore, we define projections $\mathrm{proj}_d(\mathbf{o}_k) = o_{k,d}$ for all $d \in D$ to produce the utility function for the priority d. Note that the definition of the KPI objective does also allow leaving out a priority range if not necessary, e.g., if there is only 1 threshold for a KPI. In this case, the utility functions for the not needed priorities can be simply set to the constant 1.

It is important to notice that the priority ranges are not weighted with the priority but lexicographically ordered according to the priority. The lexicographical ordering requires a specific evaluation strategy since it cannot be represented with a continuous utility measure as explained in Chapter 2.3.3.2. In principle, the weight of a higher priority range would be infinite in comparison to a lower priority range. However, this cannot be calculated. If the priorities would be weights then it is possible to calculate a KPI utility capturing all priority ranges as $o_k(v) = \sum_{d \in D} d \cdot o_{k,d}(v)$ However, this is not intended.

The important advantage of the priority range approach is that the behavior of the ODSO system is to some extent directly controllable and, thus, predictable for the operator. The reason is that KPI values in different priority ranges are incommensurable, i.e., the system does not trade-off unacceptable performance of one KPI with the optimal performance of another KPI. Specifically, consider two performance states of a cell: the first comprises an unacceptable CQI and an optimal cell load whereas in the second both KPIs are acceptable. In classical utility theory without priorities, it is possible that both performance states are considered equally preferred since the optimal cell load compensates the unacceptable CQI. In contrast to that, the priorities in ODSO force the system to prefer the second state since acceptable KPIs are always preferred to unacceptable KPIs. As can be seen by this

discussion, the priority ranges enable the MNO to define guardrails that ensure simple and comprehensible decision making on large scales while also providing complex and autonomic decision making on small scales. This simplifies the migration to the utility-function based ODSO from traditional threshold-based network operations as it given the operators greater confidence in autonomic SON operations. Due to this guardrail control, the operators may even waive the definition of sophisticated utility functions and, instead stick to simple linear functions between the thresholds.

3.4.3.3 Weights

In order to combine the utility for the different KPIs, MAUT uses aggregation functions. The simplest aggregation is the additive aggregation which, however, requires additive independence as explained in Chapter 2.3.3.1. We assume this to be true since the KPIs are seen as independent of each other. However, if this does not hold, the additive aggregation is still preferable, as explained in Chapter 2.3.3.1, because of the small number of required parameters and the typically small error of this simple approach.

Definition 3.15 (KPI weight)**.** A KPI weight $w_k \in [0,1]$ represents the relative importance of the KPI $k \in K$ for the operator. Thereby, the applicable weights in a specific context must be normalized, i.e., $\sum_{k \in K} w_k = 1$. The set of all possible KPI weights is denoted as $W_k = [0,1]$

The weights allow comparing two system states by enabling to trade-off the different utilities per KPI and to calculate an overall utility. Figure 3.18 depicts a radar chart visualizing example weights for KPIs CQI, handover ping-pong rate, cell load, and energy consumption during busy hours and at night. In the former case, it is most important for the MNO to provide the customers with the best experience possible, so, that the three quality KPIs CQI, handover ping-pong rate, and cell load are weighted to be the most important ones. In the latter case, however, it is shown that the MNO is willing to trade the quality off with reduced costs by putting more weight on the goal to decrease the energy consumption. However, notice again that a utility is calculated for each priority range.

3.4.3.4 Context Condition

Like the effect model, the objective model is context-dependent, i.e., the KPI objectives and weights depend on the operational context, i.e., the characteristics of the network cell, in which they should be evaluated. For instance, consider an urban HetNet environment with several small cells that are deployed to provide capacity. The objectives for these cells during the day might be to improve the data performance, i.e., achieve a high throughput and reduce the load. However, at night, the objectives might change such that saving energy is more important, i.e., the energy consumption by the BS should be low. This is visualized in Figure 3.18.

The context condition can be any predicate that evaluates the properties of the operational context, i.e., PM, CM, FM, or operational data. However, for the

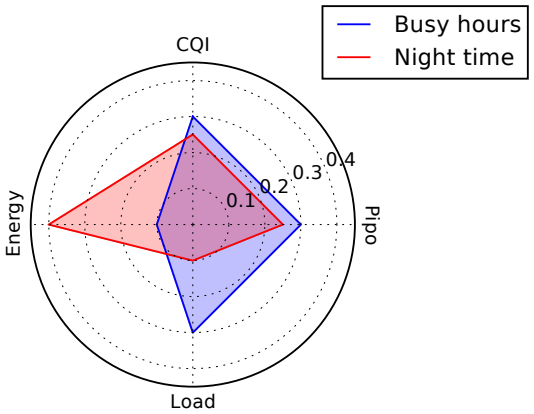


Figure 3.18: Examples for KPI weights during busy hours and night-time.

formalization of the context-dependency of the operational objectives, as presented here, the context dependency solely materializes in the fact that the objective model is not defined as a set of KPI objectives and weights but instead as a mapping.

Definition 3.16 (Objective model)**.** The objective model $\mathrm{OM} : C \times \mathbf{X} \to [K \to (\mathbf{O}_k \times W_k)_{k \in K}]$ is a function from a network cell $c \in C$ in a specific operational context $\mathbf{x} \in \mathbf{X}$ to a cell-context-specific objective model, i.e., a mapping from a KPI $k \in K$ to a pair of a KPI objective $\mathbf{o}_k \in \mathbf{O}_k$ and KPI weight $w_k \in W_k$ for k. $\mathrm{OM}(c, \mathbf{x})$ denotes the cell-context-specific objective model for cell c in the operational context $\mathbf{x}$, and $\mathrm{OM}(c, \mathbf{x}; k) = \mathrm{OM}(c, \mathbf{x})(k) = (\mathbf{o}_k, w_k)$ denotes the respective cell-context-specific KPI objective $\mathbf{o}_k$ and KPI weight w_k for KPI k

3.4.3.5 Representation

As outlined in Chapter 2.3.3.1, the elicitation of the utility functions and weights for the KPI objectives is a non-trivial task. The ODSO priorities are an attempt to support this process. Operators traditionally use KPI thresholds to define acceptable, unacceptable, and maybe optimal behavior. These thresholds are captured in, e.g., the description of the operational objectives or an SLA (see Chapter 3.1.1.1). ODSO priorities enable operators to directly express KPI thresholds in terms of priority ranges. Furthermore, the achievement awards or violation penalties assigned to the thresholds can be used as an indication for the weighting of the different KPI objectives.

For instance, consider two thresholds on the KPI cell load that define acceptable and optimal KPI values: for all $v \in \mathrm{Dom}(\mathrm{Load})$ with $\mathrm{Dom}(\mathrm{Load}) = [0.0, 1.0]$, $0.6 < v \leq 0.8$ are acceptable and all $v \leq 0.6$ are considered optimal. These thresholds can be represented with the KPI objective $\mathbf{o}_{\mathrm{Load}} = (o_{\mathrm{Load},1}, o_{\mathrm{Load},2}, o_{\mathrm{Load},3})$ whereby the objective functions must fulfill: $(0 < o_{\mathrm{Load},1}(v) < 1) \iff v \in (0.8, 1.0]$, $(0 < o_{\mathrm{Load},2}(v) < 1) \iff v \in (0.6, 0.8]$, and $(0 < o_{\mathrm{Load},3}(v) < 1) \iff v \in [0.0, 0.6]$. As a first step to define such utility functions, the operator might apply linear utility

functions of the form

$$\text{linear}(t_1, t_2)(x) = \min\left(\max\left(\frac{x - t_1}{t_2 - t_1}, 0\right), 1\right). \tag{3.4}$$

As can be seen, this defines a linear function with the range $[0, 1]$ and the defined values $\text{linear}(t_1, t_2)(t_1) = 0$ and $\text{linear}(t_1, t_2)(t_2) = 1$. Based on this, the operator can define $o_{\text{Load},1} = \text{linear}(1.0, 0.8)$, $o_{\text{Load},2} = \text{linear}(0.8, 0.6)$, and $o_{\text{Load},3} = \text{linear}(0.6, 0.0)$. The resulting KPI objective definition is depicted in Figure 3.19. These utility functions might later be replaced with, e.g., concave utility function, which represent the MNO preferences more correctly. Notice, it is also possible to define specific target values for KPIs instead of thresholds, e.g., the cell load should be as close to 0.6 as possible, with a similar approach using KPI objectives as shown in Figure 3.17b.

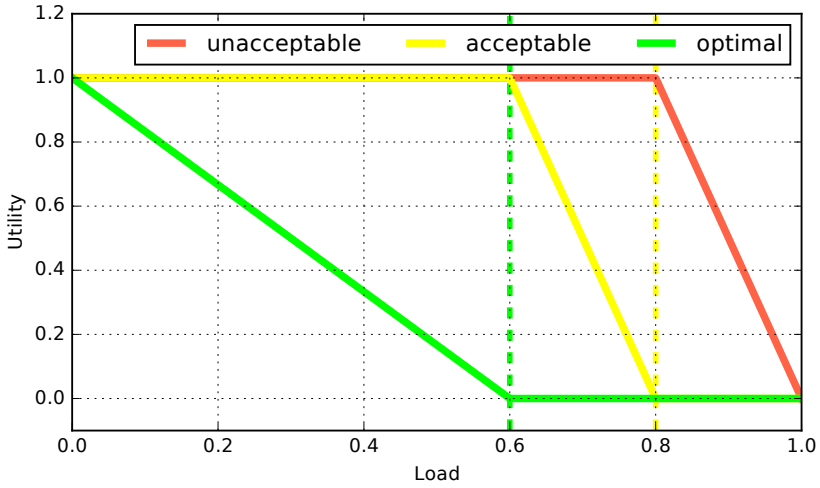


Figure 3.19: Visualization of the example KPI objective for cell load.

One way of implementing the objective model is a set of rules:

```
IF condition(cell, context) THEN kpi objective WITH weight
```

where `condition(cell, context)` is some condition on the cell to which the objective should be applied and the current operational context, `kpi objective` is the definition of the KPI objective, i.e., the prioritized utility functions, and `weight` is the weight of the KPI objective. For instance, the following rule defines that during the busy hours, i.e., between 08:00 and 17:59, in an urban location, the CQI is unacceptable below 0.5 and optimal above 0.8, and that the satisfaction of that KPI objective has a weight of 0.3.

```
IF time in [08:00, 17:59] AND location(cell) = urban
  THEN CQI = linear(0.6, 0.8) WITH 0.3
```

Thereby `linear(0.6, 0.8)` defines a KPI objective with linear utility functions as shown in Equation 3.4 for the thresholds 0.6 and 0.8. This results in the KPI objective for the CQI $\mathbf{o}_{\text{CQI}} = (\text{linear}(0.0, 0.6), \text{linear}(0.6, 0.8), \text{linear}(0.8, 5.5547))$.

Notice that an objective model comprising a set of rules is an action policy. Hence, the consistency of the resulting KPI objectives must be ensured in every possible context: first, there must be a KPI objective-weight-pair for every KPI, second, there must be a single KPI objective and a single KPI weight for each KPI, and, third, the weights must sum up the 1. Whereas the first requirement can be satisfied with the definition of default objectives and weights, the second can be handled with standard conflict resolution approaches for action policies (see Chapter 2.2.1). The last consistency constraint merely requires a simple normalization of the weights during the evaluation of the rules. In the following, we assume that these requirements are fulfilled.

3.4.4 Action Selection

In a specific operational context $\mathbf{x} \in \mathbf{X}$, the system needs to make a decision for an action to take. In order to act rational in this setting, the system needs to select the action based on the maximization of the expected utility, i.e., the action with the highest expected utility. However, since the KPI objectives define priority ranges, this selection has to consider the lexicographical order of the action utility within each range. Hence, the process is twofold: first, the expected utility for each action and each priority range is calculated and, second, the action utilities are compared considering the priority ranges.

In the following, consider the applicable KPI objectives and weights for the network cell $c \in C$ in the context $\mathbf{x} \in \mathbf{X}$ as $\mathrm{OM}(c, \mathbf{x}; k) = (\mathbf{o}_k, w_k)$ for each KPI $k \in K$. For the following presentation of the approach, consider the set of KPIs to be $K = \{1, \ldots, N_K\}$ with $|K| = N_K$. The calculation of the overall expected utility of an action $a \in A$ is only reasonable with respect to the objectives of a single priority range $d \in D$. Therefore, let $o_{*,d}$ be a mapping from a KPI $k \in K$ to the respective utility function with the priority d such that $o_{*,d}(k) = \mathrm{proj}_d(\mathbf{o}_k) = o_{k,d}$. Furthermore, consider $(a, \mathbf{f}) \in \mathrm{AF}$ to be a pair of a feasible action with $\mathrm{proj}_C(a) = c$ and its effect from the action-effect set produced by action proposal.

In general, the expected utility of a continuous action effect $\mathbf{f}$ with respect to the priority KPI objectives $o_{*,d}$ according to utility theory is:

$$\mathrm{E}\left[o_{*,d}(\mathbf{f})\right] = \int_{\times_{k \in K} \mathrm{Dom}(k)} \cdots \int \overline{\mathbf{f}}(v_1, \ldots, v_{N_K}) \cdot \overline{o_{*,d}}(v_1, \ldots, v_{N_K}) \, \mathrm{d}v_1 \ldots \mathrm{d}v_{N_K} \tag{3.5}$$

with $\bigtimes_{k \in K} \mathrm{Dom}(k) = \mathrm{Dom}(1) \times \cdots \times \mathrm{Dom}(N_K)$ denoting the cross product of the KPI domains for all KPIs $k \in K$, $\overline{\mathbf{f}}(\cdot)$ representing the joint probability distribution for the values $v_1, \ldots, v_{N_K}$ of all KPIs $1, \ldots, N_K$ induced by $\mathbf{f}$, and $\overline{o_{*,d}}(\cdot)$ being a respective multi-attribute utility function for $o_{*,d}$. As can be seen, this turns out to be a complex multiple integral over all KPI domains.

Since the system state considered by the utility function consists of different attributes, i.e., KPIs, a multiattribute utility function which combines the utilities per KPI is required. As motivated in Chapter 3.4.3.3, the weighted sum aggregation is

used, i.e.,

$$\overline{o_{*,d}}(v_1, \ldots, v_{N_K}) = \sum_{k \in K} w_k \cdot o_{k,d}(v_k) \tag{3.6}$$

with $o_{k,d} = o_{*,d}(k)$.

Equation 3.5 can be computationally very complex due to the multidimensional integral. However, the predicted effects for each KPI, i.e., the probability density functions, are independent of the predicted effects of the other KPIs, i.e., the multivariate probability density is the product of the univariate probability density distributions (see Chapter 3.4.1.2):

$$\overline{\mathbf{f}}(v_1, \ldots, v_{N_K}) = \prod_{k \in K} f_k(v_k) \tag{3.7}$$

with $\mathbf{f}(k) = f_k$.

Feeding Equation 3.6 and Equation 3.7 into Equation 3.5 finally allows us to rewrite the expected utility as the sum over the expected utilities per KPI (see [Fis70, pp. 149-150]) as shown in Equation 3.8 on Page 83.

The simplified calculation in Equation 3.8 is in practice computationally tractable. The complexity of the calculation is linear growing with the number of KPIs, i.e., $O(|K|)$. Furthermore, the single integrals over each KPI domain can be efficiently and accurately calculated using algorithms like Simpson's rule [Wei00]. Another approach could be to discretize all KPI domains, i.e., divide it into small, discrete buckets. This approach has been used in the evaluation of the ODSO concept as outlined in Chapter 7.2.3. If the multivariate probability $\overline{\mathbf{f}}(v_1, \ldots, v_{N_K})$ would not be the result of probabilistic independent marginal utility per KPI, i.e., if the KPIs are somehow connected, the expected utility in Equation 3.5 cannot be simplified like this. Hence, the complexity grows exponential with the number of KPIs, i.e., $O(c^{|K|})$, making the problem intractable. In such case, its value can be approximated using statistical integration methods based on Monte Carlo sampling [Wei00][Win04]. Unfortunately, these methods are computationally much more complex.

Using Equation 3.8 it is possible to calculate the expected utility over all KPIs for each priority range.

Definition 3.17 (Expected utility for one priority)**.** $u_d \in [0, 1]$ denotes the expected utility for the priority $d \in D$ of a complete effect $\mathbf{f}$ on cell $c \in C$ in the context $\mathbf{x} \in \mathbf{X}$ with respect to the objective model OM. It is calculated according to $u_d = \mathrm{E}\left[o_{*,d}(\mathbf{f})\right] = \sum_{k \in K} w_k \cdot \int_{\mathrm{Dom}(k)} f_k(v) \cdot \mathrm{proj}_d(\mathbf{o}_k)(v)\, \mathrm{d}v$ such that $(\mathbf{o}_k, w_k) \in \mathrm{OM}(c, \mathbf{x}, k)$ for each KPI $k \in K$.

Definition 3.18 (Expected utility vector)**.** $\mathbf{u} = (u_1, \ldots, u_{N_D})$ denotes an expected utility vector, or just utility vector, that contains the expected utility for all N_D priorities. Hence, $\mathbf{u} \in ([0, 1])_{d \in D}$. $\mathrm{E}\left[o_{*,*}(\mathbf{f})\right] \in ([0, 1])_{d \in D}$ denotes the expected utility vector of the complete effect $\mathbf{f}$ under the objectives $o_{*,*} = (o_{*,1}, \ldots, o_{*,N_D})$, i.e., $\mathrm{E}\left[o_{*,*}(\mathbf{f})\right] = \left(\mathrm{E}\left[o_{*,1}(\mathbf{f})\right], \ldots, \mathrm{E}\left[o_{*,N_D}(\mathbf{f})\right]\right) = \mathbf{u}$.

$$
\begin{aligned}
\mathrm{E}\left[o_{*,d}(\mathbf{f})\right] &= \int_{\times_{k\in K}\mathrm{Dom}(k)} \cdots \int \underbrace{\overline{\mathbf{f}}(v_1,\ldots,v_{N_K})}_{\text{Equation 3.7}} \cdot \underbrace{\overline{o_{*,d}}(v_1,\ldots,v_{N_K})}_{\text{Equation 3.6}} \,\mathrm{d}v_1 \ldots \mathrm{d}v_{N_K} \\
&= \int_{\times_{k\in K}\mathrm{Dom}(k)} \cdots \int \underbrace{\left(\prod_{k\in K} f_k(v_k)\right) \cdot \left(\sum_{k\in K} w_k \cdot o_{k,d}(v_k)\right)}_{\text{factor in sum}} \mathrm{d}v_1 \ldots \mathrm{d}v_{N_K} \\
&= \int_{\times_{k\in K}\mathrm{Dom}(k)} \cdots \int \underbrace{\sum_{k\in K}}_{\text{sum rule for integrals}} \\
&\qquad \left(\underbrace{\left(\prod_{k'\in K} f_{k'}(v_{k'})\right)}_{\text{rewriting}} \cdot w_k \cdot o_{k,d}(v_k) \right) \mathrm{d}v_1 \ldots \mathrm{d}v_{N_K} \\
&= \sum_{k\in K} \underbrace{\int_{\times_{k\in K}\mathrm{Dom}(k)} \cdots \int}_{\text{reordering of integrals}} \\
&\qquad \left(\prod_{k'\in K\setminus\{k\}} f_{k'}(v_{k'})\right) \cdot f_k(v_k) \cdot w_k \cdot o_{k,d}(v_k) \,\mathrm{d}v_1 \ldots \mathrm{d}v_{N_K} \\
&= \sum_{k\in K} \int_{\mathrm{Dom}(k)} \left(\int_{\times_{k'\in K\setminus\{k\}}\mathrm{Dom}(k')} \cdots \int \right. \\
&\qquad \left. \left(\prod_{k'\in K\setminus\{k\}} f_{k'}(v_{k'})\right) \cdot \underbrace{f_k(v_k) \cdot w_k \cdot o_{k,d}(v_k)}_{\text{constant for all } k'} \,\mathrm{d}v_1 \ldots \mathrm{d}v_{N_K} \setminus \mathrm{d}v_k \right) \mathrm{d}v_k \\
&= \sum_{k\in K} \int_{\mathrm{Dom}(k)} f_k(v) \cdot w_k \cdot o_{k,d}(v) \cdot \\
&\qquad \underbrace{\left(\int_{\times_{k'\in K\setminus\{k\}}\mathrm{Dom}(k')} \cdots \int \left(\prod_{k'\in K\setminus\{k\}} f_{k'}(v_{k'})\right) \mathrm{d}v_1 \ldots \mathrm{d}v_{N_K} \setminus \mathrm{d}v_k \right)}_{=1 \text{ by Definition 3.6}} \mathrm{d}v \\
&= \sum_{k\in K} \int_{\mathrm{Dom}(k)} f_k(v) \cdot \underbrace{w_k}_{\text{constant for all } v} \cdot o_{k,d}(v) \,\mathrm{d}v \\
&= \sum_{k\in K} w_k \cdot \int_{\mathrm{Dom}(k)} f_k(v) \cdot o_{k,d}(v) \,\mathrm{d}v. \qquad (3.8)
\end{aligned}
$$

Furthermore, we define a projection $\text{proj}_d(\mathbf{u}) = u_d$ to produce the expected utility of priority d for the utility vector $\mathbf{u} = (u_1, \ldots, u_{N_D})$.

For each KPI, i.e., attribute, the utility of the effects is based on two criteria: first the probabilistic satisfaction of the priority ranges and, second, the probabilistic KPI values within each range. Figure 3.20 visualizes the calculation of the expected utility for different priorities for one KPI. Thereby, the colored areas represent the expected utility for each priority.

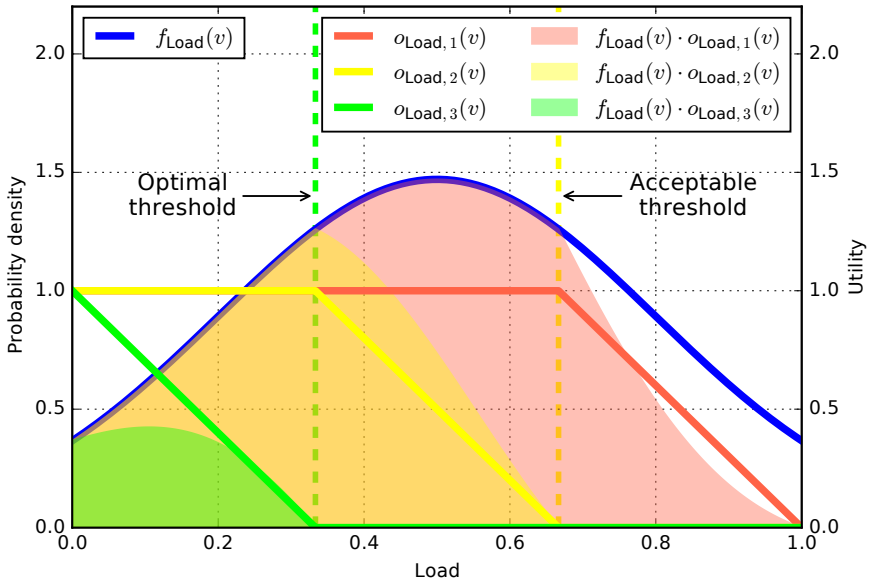


Figure 3.20: Visualization of the calculation of the utility. The expected utility for each priority is the area below the utility graph with the same color.

According to utility theory, an action a_1 is preferred to another action a_2 if the expected utility of the effect of a_1, $\mathbf{u}_1$, is higher or equal than the expected utility of the effect of a_2, $\mathbf{u}_2$. That is, $a_1 \succsim a_2 \iff \mathbf{u}_1 \geq \mathbf{u}_2$. However, we define an expected utility per priority range whereas the expected utility of a higher priority is infinitely more important than the expected utility of a lower priority, i.e., the operator concentrates first on the high priority KPI values and then on the lower priority KPI values. Hence, the preferences over the actions is defined by a lexicographical order over the expected utility vectors of the outcomes of the actions. In this way, the presented priority-based multi-criteria decision making problem aims at maximizing the expected utility along the priority ranges from high to low.

Definition 3.19 (Ordering of expected utility vectors)**.** Two utility vectors $\mathbf{u}' = (u'_1, u'_2, u'_3)$, $\mathbf{u}'' = (u''_1, u''_2, u''_3)$ are lexicographically ordered with $\geq_D$: $(\mathbb{R})_{d \in D} \times (\mathbb{R})_{d \in D}$ according to the priority as

$$\mathbf{u}' \geq_D \mathbf{u}'' \iff \exists d_i \in D. \forall d_j \in D, d_j < d_i. u'_{d_j} = u''_{d_j} \wedge u'_{d_i} \geq u''_{d_i}.$$

Definition 3.20 (Action preference)**.** The preference of an action a' with the corresponding utility vector $\mathbf{u}'$ over an action a'' with the utility vector $\mathbf{u}''$, is defined as

$$a' \succsim a'' \iff \mathbf{u}' \geq_D \mathbf{u}''.$$

Figure 3.21 exemplifies the preference ordering for the example introduced in Figure 3.15. Due to the formal definitions, it is now possible to totally order the system states A, B, C, and D according to the operator objectives. Let $\mathbf{u}^A = (1.0, 1.0, 0.25)$, $\mathbf{u}^B = (1.0, 0.75, 0.42)$, $\mathbf{u}^C = (1.0, 0.95, 0.0)$, and $\mathbf{u}^D = (0.8, 0.42, 0.0)$ be the utility vectors of the system states A to D. Consequently, $A \succsim C \succsim B \succsim D$ because $(1.0, 1.0, 0.25) \geq_D (1.0, 0.95, 0.0) \geq_D (1.0, 0.75, 0.42) \geq_D (0.8, 0.42, 0.0)$. Specifically, $C \succsim D$ since $1.0 = 1.0 \wedge 0.95 \geq 0.75$.

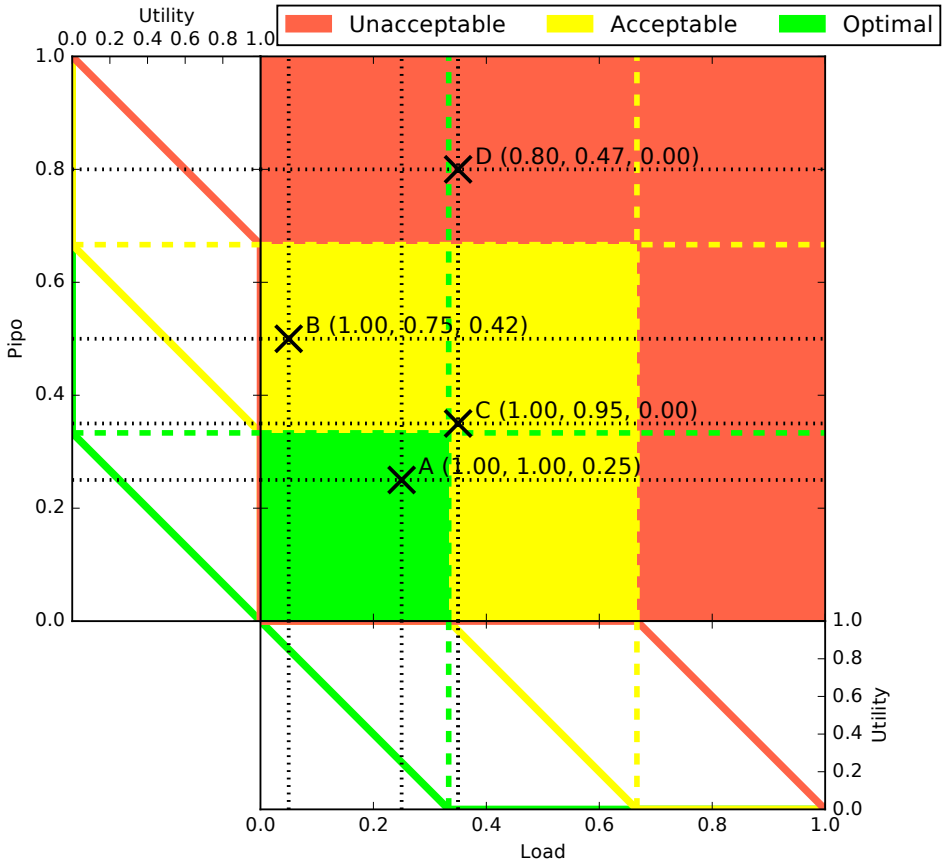


Figure 3.21: Example utilities for three priority ranges and two KPIs.

3.5 Operations Process

The process of operating a SON changes with the introduction of ODSO. In manual SON operations mode, the human operator has to manually adapt the SON configuration, i.e., perform a SON change, if either a context change, an objective change,

or a technical change happens as outlined in Chapter 3.1.3. Thereby, the former typically requires the adaptation of single cells whereas the latter two generally affect the whole network. This leads to the automation and dynamics gaps.

The ODSO concept overcomes the automation gap with the introduction of separated, formalized models for the operational objectives and the technical expertise. This loosens the tight integration between both types of knowledge and enables their separate evolution as depicted in Figure 3.22. If the operational objectives change (indicated by the yellow marks on the time line), e.g., due to a new marketing campaign, the operator only needs to adapt the objective model independently of the technical model. Hence, the operator controls the SON through the objectives. Conversely, changes in the technical expertise (indicated by the blue marks on the time line), e.g., due to the deployment of a new SON function, require only the adaptation of the technical models independently of the objective model. As a result, both models can evolve independently without interference which simplifies their maintenance.

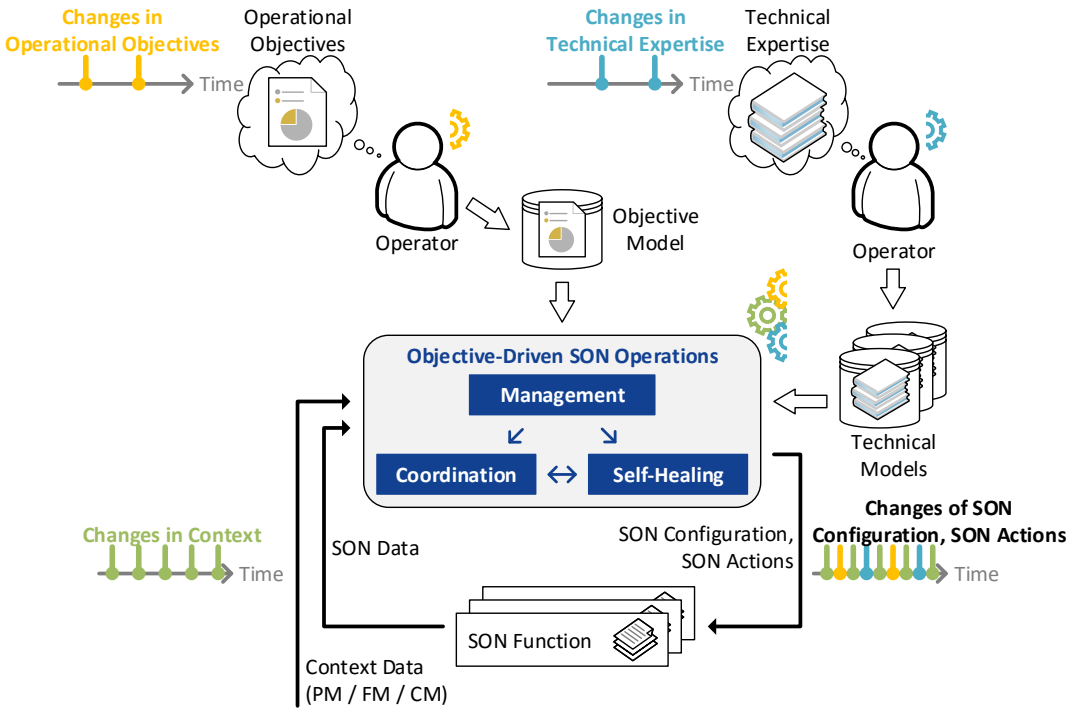


Figure 3.22: The ODSO concept decouples the different changes such that the objective model and the technical models can evolve independently of each other. The main efforts for SON operation are performed automatically by the ODSO system.

Although the formalization of the operational objectives and technical expertise into the respective models requires some manual efforts (indicated by the partial yellow and blue cogwheels next to the operators in Figure 3.22), the main part of

the automation gap, i.e., bringing together the objectives and the expertise to derive the SON configuration, is performed automatically by the ODSO system (indicated by the partial yellow and blue cogwheels next to the ODSO system). Furthermore, ODSO enables overcoming the dynamics gap since the automatic derivation of the SON changes can be performed whenever a change in the context (indicated by the green marks on the time line), operational objectives or technical expertise occurs without any human involvement (indicated by the green cogwheel next to the ODSO system). This allows adapting the SON to the concrete characteristics of the network, thereby, lifting formerly wasted potentials for optimization.

Although the two model types can evolve independently of each other, they need to satisfy a property in order to enable the automated combination of both: the effects determined by the action proposal based on the technical models must fit to the utility functions defined in the objective model. In the end, this boils down to the fact that both must be defined over the exact same set of KPIs K with the same domain $\mathrm{Dom}(k)$ and semantics for each KPI $k \in K$. In the presentation so far, this property was implicitly assumed. This is not a large assumption since a lot of the basic KPIs are standardized, e.g., [3GP15g] defines common KPIs for LTE. If this assumption cannot be fulfilled, it is necessary to establish a translation between the KPI effects and KPI objectives, i.e., a transformation from a KPI value in a technical model to KPI values in the objective model. Thereby, the mapping between technical and objective KPIs can be $1:1$, $1:n$, $m:1$, or $m:n$. This approach, however, is not new and commonly known in the research area of data integration [Len02].

3.6 Related Work

The contributions of ODSO for overcoming the manual gap of SON operations can be divided into two aspects: on the one hand, it defines a PBM system based on a utility-function policy that enable autonomic operations of a SON and, on the other hand, it provides an architecture with three distinctive components that perform all tasks related to SON operations. Thus, this section presents the related work according to these two aspects.

3.6.1 Policy-Based Management

In this chapter, we present other PBM approaches capable of overcoming the manual gap without a focus on SON. Thereby, they are sorted along the employed policy types as in Chapter 2.2.

3.6.1.1 Action Policy

There is a considerable body of work regarding the use of action policies in network management. This approach is even standardized by the Internet Engineering Task Force (IETF) under the term PBM [Moo+01][Ver02]. Further noticeable is the work under the term PBNM that has been coined in [Str03]. Other examples of

this category can be found in [Pha+08][Rom+10]. A detailed presentation of these approaches is, however, omitted since, in principle, they do not allow autonomic operations as ODSO. This is shown in the following.

Operations Process As outlined in Chapter 2.2.1, an action policy-based system, in principle, captures the results of a complex decision performed by a human operator in rules. Figure 3.23 depicts a possible operations process induced by the usage of an action policy for overcoming the manual gap of SON operations. The policy is a set of IF context condition THEN action rules manually created by the operator based on the operational objectives and the technical expertise (indicated by the yellow and blue cogwheels next to the operator). In principle, these capture the result of the operator's decision making, i.e., action, for each and every operational context identified by the context condition. That means, for fixed operational objectives and technical expertise, the operator needs to project his thoughts into each possible operational state the network might be in, decide which action to perform, and record it as a rule. This can be seen as a manual pre-compilation of the autonomic decision making performed by the ODSO components. Of course, the operator practically just thinks through some distinctive operational situations since, theoretically, the number of states is often infinite. The complete process needs to be reiterated whenever one of the inputs, i.e., objectives or expertise, changes (indicated by the yellow and blue marks on the time lines). This is because the action policy intertwines both inputs such that effects of the objectives and the technical expertise on the rules of the policy are hardly distinguishable.

As depicted in Figure 3.23, an action policy system allows overcoming the dynamics gap, i.e., it can automatically adapt the SON configuration (indicated by the green cogwheel next to the action policy system) according to variations in the operational context (indicated by the green marks on the time line). However, the automation gap, i.e., the adaptation of the SON to new operational objectives or technical expertise, is left open and still needs to be performed manually.

An action policy system does not fully satisfy the requirements for autonomic SON operations. This has two main disadvantages: First, the maintenance of the action policy is costly since it needs to be completely recreated if either the operational objectives or the technical expertise changes. Second, it is hard to consider uncertainty in such a rule system. Nevertheless, it also has two main advantages: First, it is a simple, easy to understand model that is broadly used and well-known in network management. Second, the direct control via the IF context condition THEN action leads to a comprehensible behavior of the system and enables the operator to verify operational decisions in advance.

Policy Refinement The widespread adoption of PBNM lead to a number of approaches to ease the creation of the action policy. Most common are *policy refinement* approaches which allow an abstraction of the concepts used in an action policy [Mee+06][Gal+12][Rom12]. As depicted in Figure 3.24, these approaches start with a business policy, i.e., a set of action rules with operator concepts. The rules describe under which conditions an actions needs to be executed, however, the terms

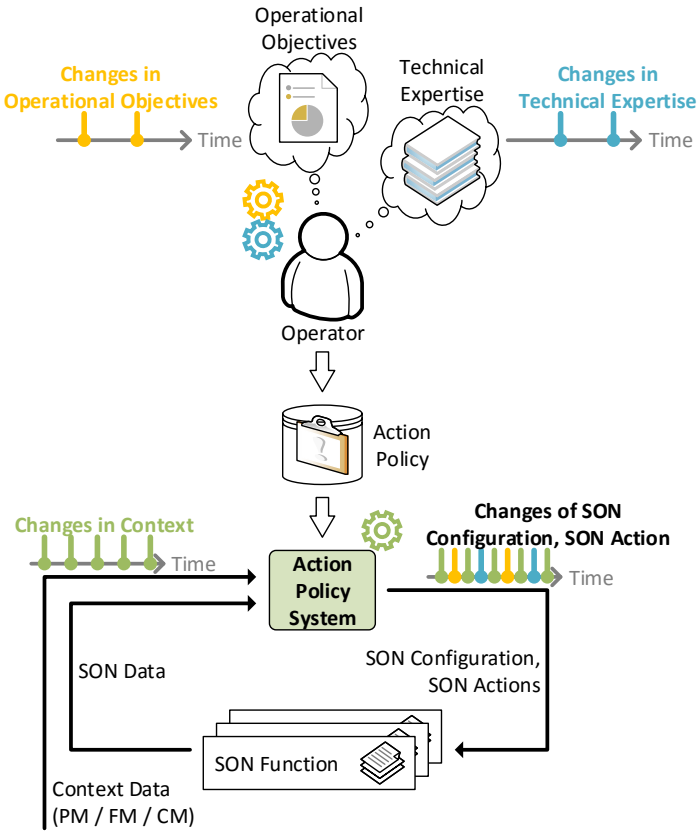


Figure 3.23: An action policy system enables automated SON operations in response to context changes. The action policy still needs to be manually derived from the objectives and the expertise, though.

used in the condition and action parts are business objects and business properties which do not directly correspond to real systems, e.g., "time = business_hours" refers to the time period when businesses are typically open. That allows the operator to define the behavior of the network in abstract operator concepts without any knowledge of the concrete technical implementation. In order to be executable, the business policy needs to be refined into a *technical policy*, i.e., a set of action rules with technical concepts, e.g., "time = [08:00, 17:59]" refers to the concrete time period between 8 am and 6 pm. Therefore, policy refinement approaches require some kind of concept model which maps the operator concepts to technical concepts, e.g., "time ↦ time" and "business_hours ↦ [08:00, 17:59]". Note that the mapping can be more complex, e.g., a business concept can be mapped to several technical concepts.

It is important to notice that, although the business policy allows the abstract definition of the action rules, it is still a direct definition of the behavior of the network in terms of actions that need to be taken under some condition. In principle, policy refinement can be seen as a translation of action rules between two

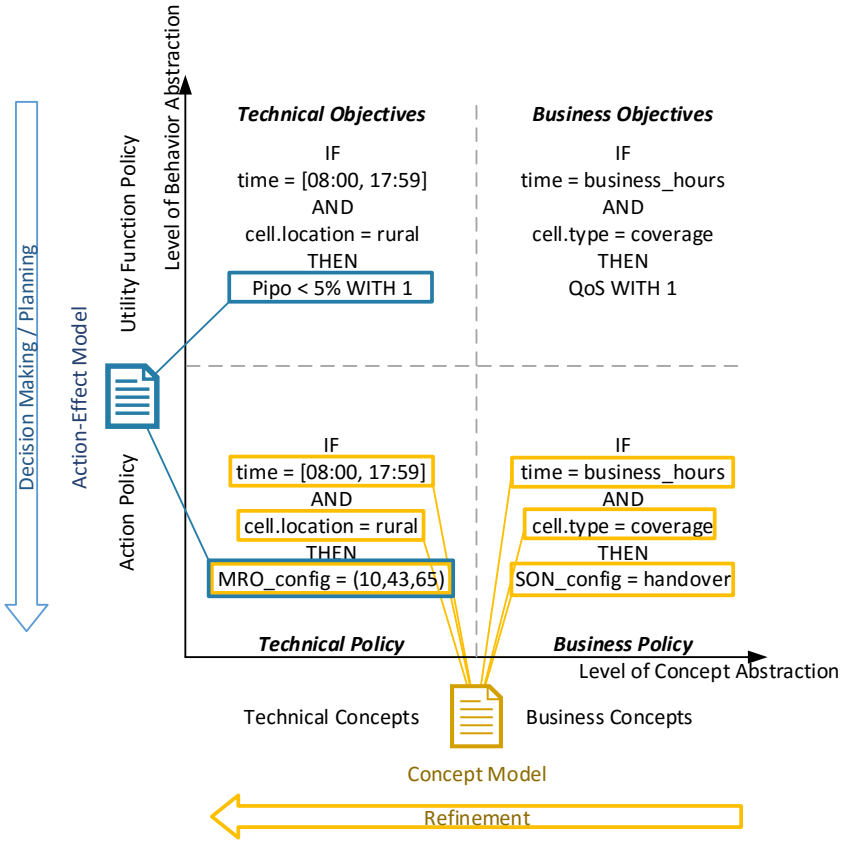


Figure 3.24: Comparison between refinement of action policies (yellow) and planning for utility function policies (blue).

languages [Rom12]. Hence, the business policy still mixes up the objectives with the technical expertise.

In contrast to policy refinement, ODSO is based on a utility function policy that directly expresses the technical objectives as depicted in Figure 3.24. As can be seen, this behavioral abstraction is orthogonal to the conceptual abstraction performed by policy refinement. Since the objectives of the MNO can be context-dependent (see Chapter 3.1.1.3), the technical objectives may also be a set of rules. However, they do not define an action to be taken in the conclusion but instead a technical objective, i.e., prioritized utility functions and weights for the different KPIs. For instance, "Pipo $<$ 5% WITH 1" represents a simple, one priority utility function on the KPI handover ping-pong rate with the weight 1. In order to determine the actual action to be taken, the system must perform a decision making or planning process in order to determine an action or a set of actions, respectively, that achieves the objectives. Therefore, the system requires an action-effect model which defines the effects of the actions. In ODSO these are provided by the technical models. Note that ODSO does not create the technical policy but instead can perform the decision making in each decision situation for the concrete context.

The objects in the rules of the technical objectives are defined using technical concepts, i.e., they are defined on the same level of abstraction with respect to the rule concepts. Since the two presented dimensions of abstractions are orthogonal, the two abstraction approaches can also be combined to allow the definition of *business objectives*, i.e., a utility function policy with business concepts. Note that the business objectives can be similar to the business goals introduced in Chapter 3.1.1 though.

3.6.1.2 Goal Policy

A PBM approach based on a goal policy already allows autonomic behavior and, thus, overcoming the manual gap of SON operations. However, goals are less expressive than utility functions as outlined in Chapter 2.2.2. Hence, a goal policy would only allow to define satisficing objectives as they are typically defined in SLAs but no maximization objectives going beyond that.

An often cited approach for PBM based on a goal policy is presented in [Ban+05]. They present a two-step approach comprised of a goal refinement and action policy creation: First, high-level goals, i.e., desired system states, in terms of business concepts are manually refined into low-level goals in terms of technical concepts. Thereby, they propose the usage of refinement patterns that provide some process guidance. Second, an action policy is derived from the technical goals. Although, the authors call this step also policy refinement, according to the classification used in this work, it is a decision making step. Thereby, the proposed policy actions are actually action sequences that need to be performed to achieve the goal. These are planned using abductive reasoning and event calculus based on a detailed model of the action effects and the environment.

In [AB07], the authors present a PBM system that combines a conflicting action policy with a goal policy for conflict resolution. Thereby, the goals are KPI targets derived from SLAs for data center management. Consequently, the technical model is a set of conflicting action rules that propose actions in response to specific events. In order to resolve the conflicts the system employs a mathematical queuing model to simulate the effects of an action on the KPIs. Such a mathematical formulation is very difficult to create for RAT operations, though. Finally, the action which maximizes the weighted sum of satisfied KPI targets is selected.

A PBM approach for Information Technology (IT) service management based on SLAs is presented in [BST06]. The approach aims at enabling the control along business objectives while requiring as little effort as possible to model the involved IT systems. The business objectives are target values for KPIs derived from SLAs that can be weighted to represent their importance. In contrast to the ODSO objectives, however, the objectives can be defined as aggregates over time periods, e.g., "the aggregate service revenue generated over the current three-month period must be above 100,000 $" [BST06, p. 47]. Such future objectives are not seen relevant for SON management here, though, since the KPI objectives are not defined as aggregates. Instead, they should be satisfied in every granularity period instantly. The technical model allows estimating the probability of the KPI values in the future for a specific

course of action given the current system state. This definition is quite similar to the technical models required for ODSO. Based on that, the approach can estimate the utility of each course of action as the weighted sum of the probabilities that the objectives might not be satisfied in the future.

In [Bal+08], the authors present a very similar concept for self-optimization in mobile networks[4]. Thereby, the operator defines an overall goal over the KPIs based on thresholds. The technical model is a set of conflicting action rules that propose possible optimization actions and a self-learned dynamic Bayesian net that allows estimating the probability of satisfying the goals for each action. These ambiguities about the possible courses of actions are resolved by selecting the action that maximizes the probability that the goal is achieved.

3.6.1.3 Utility Policy

The use of utility policies in PBM is not a new idea. However, most of the related work utilize simple utility functions to represent the objectives. Putting this in the context of SON operation, this would mean that the operator cannot prioritize specific value ranges for the KPIs. As a result, this leads to a more complicated objective elicitation and less predictable system behavior.

A common approach to create simple utility functions is to compute and use the negative distance to target values for KPIs. At first sight these approaches seem like goal policy PBM system, but they actually use linear utility functions with possibly negative values. For instance, in [SK05], the authors propose a network management system that autonomically configures routers in a network to provide different user classes with their required QoS in terms of target values for KPIs like bandwidth and delay. The technical model is represented as a set of forecast functions that are learned during execution. These enable the system to predict the effects of some configuration change on the KPIs. Based on this, the system can predict and maximize the expected utility as the weighted sum of the KPI utilities for different configurations. Similarly, [KD07] presents an autonomic data center management system called Unity. The manager has the task to allocate servers either to web server node groups for load balancing or to a free pool. The decision-making is based on simple, linear utility functions for the response times of each web server group and the number of unassigned servers in the pool. Thereby, the response time utility is defined as the negative distance to a target response time. Based on this, the manager aims at maximizing the sum of the utilities. The estimations of the response times for a specific allocation of servers were estimated using a queuing model which corresponds to a technical model. This is comparable to the queuing model used in [AB07].

[NGA08] present a comprehensive study on utility theory for access network selection. Therefore, they analyzed different parameterized utility functions and proposed the usage of an adapted sigmoid function. Additionally, they also proposed the usage of a product function (multiplicative multi-criteria utility) as utility aggre-

[4]The authors actually refer to self-healing, however, their definition is more related to self-configuration and self-optimization as used in this thesis.

gation since, in their use case, the overall utility must be 0 if one component utility is 0, i.e., the attributes are not utility-independent if one attribute utility is 0. Based on this rather theoretical work, [Mot+14] presents a framework called OppLite for load-balancing (off-loading) for mobile networks. The management application that is running on the UEs decides whether it normally connects to a BS, acts as a relay node, or opportunistically connects to a relay. This decision is based on the criteria number of neighbors, battery lifetime, and link quality. They also use sigmoidal utility functions and multiplicative multi-criteria utility aggregation. The decision making is based on a fixed threshold on the utility of the current situation. Hence, this approach does not predict the effects of the decisions making and a technical model is not necessary.

[Ant+99][Cra+03] present a framework for multi-criteria optimization of routing paths in fixed networks. Instead of requiring the MNO to define a utility function, the approach is based on thresholds for acceptable (reservation) and requested (aspiration) values for each criterion, i.e., KPI. These thresholds, referred to as soft-constraints, define four types of priority ranges: range A where all requested thresholds are satisfied; B ranges where at least for one KPI the requested threshold is satisfied and the acceptable thresholds for the other KPIs; C ranges where all acceptable thresholds are satisfied; and D ranges where no threshold is met. Based on this, the optimization algorithm aims at finding solutions, i.e., routing paths, which are in range A. Thereby, the selection in the range is guided by a linear utility function that is constructed from the thresholds. If there is no solution in range A, search continues in B ranges and so forth. In this work, the technical model is a model of the network that allows the estimation of the criteria values. The idea of the soft-constraints is similar to the prioritized utility functions in ODSO. However, the latter can actually be seen as an extension of the former providing several advantages: on the one hand, it allows a fine-grained definition of the utility functions for each priority range, and, on the other hand, it enables decision-making with stochastic effects of the actions which might probabilistically violate the thresholds.

As stated in Chapter 3.4, ODSO can be seen as PBM system based on an action policy for action proposal and utility-based conflict resolution. The authors of [BB08] propose a system that also adopts this idea. The starting point is an action policy with many conflicts, i.e., its evaluation proposes a number of actions whereby only one action is allowed. From the rules, a state space is derived based on the used metrics and thresholds in the condition parts, e.g., a condition like $CQI > 0.6$ would result in two metric ranges, one for values greater than 0.6 and another for values less than 0.6, that are finally cross-multiplied with other metric ranges to form the states. That is, the objectives are defined over discrete KPI domains. Based on a linear utility function and weight for each metric, the utility of a state is a kind of weighted sum of the metric utilities. Compared to ODSO, the objective definition is rather simple. However, the focus of this work is that the system learns the best action to take in each system state using reinforcement learning. That is, it learns an estimation of the expected utility of the state that the system will transition to if an action is executed. As a result, the technical model, i.e. the effects of an action, are learned at run time. This is similar to the forecast

functions presented in [SK05], as outlined above.

ACCENT [TC09] is a similar approach which, however, is based on an explicit formulation of the action effects in advance. Hence, it is closely related to the approach in ODSO. The objective model is a set of context-dependent objectives that define linear utility functions on KPIs. The technical model is a set of action rules which define actions to perform together with an estimation of their effects. Based on these models, the system can determine the applicable actions and their respective effects in the current context, and resolve the conflicts among the actions based on their estimated utility. Compared to ODSO, the approach does neither consider stochastic actions with probabilistic effects nor prioritized objectives ranges.

3.6.2 SON Operations

In the following, related work focusing specifically on the operation of a SON is presented. In general, most of this work has been done within funded research projects aiming at developing SON. Therefore, the respective projects and their contribution to an architecture for SON operations are presented. Related details regarding the tasks of each specific component, i.e., management, coordination, and self-healing, are presented in the respective chapters later.

3.6.2.1 3GPP Standardization

The 3GPP does not provide an architecture for the operation of a SON, specifically not an autonomic approach as represented by ODSO. Nevertheless, [3GP13, Annex A] provides a simple target achievement evaluation to assess the result of the optimization by some SON function. Although this is supposed to support monitoring of a SON, it can be seen as operational objectives. The target achievement is based on the definition of thresholds and weight for the network KPIs. Using this, the total target achievement is defined as the weighted sum of the differences between target values and actual performance for each KPI. Similar to the approach in [SK05] presented above, this can be seen as unnormalized, linear utility functions.

3.6.2.2 SOCRATES Project

The Self-Optimisation and self-ConfiguRATion in wirelEss networkS (SOCRATES) research project [SOC11] was running from 2008 until 2010 and supported by the European Union under the 7th Framework Program. It involved academic and industrial partners from six European countries. The project aimed at "the development, evaluation and demonstration of concepts, methods and algorithms for self-configuration, self-optimisation and self-healing in LTE networks" [Kür+10, p. 17]. Hence, it was a project specifically targeting SON. Starting with 24 identified use cases, the project developed a holistic view on SON and elaborated nine use cases in detail. Although the project focused on LTE, the developed methods might also be applied to other technologies like 3G systems.

With respect to SON operations, SOCRATES was among the first to acknowledge that the execution of independent SON functions, i.e., SON functions that do not

know about each other, needs to be "harmonized", i.e., controlled such that the SON functions jointly work towards the same goal [Sch+11]. Therefore, the project introduces a SON coordinator which performs three main tasks: first, the harmonization of the configuration of the SON functions, here referred to as policies, second, the harmonization of the actions of the SON functions at run time, and, third, the detection and correction of undesired behavior by the SON functions at run time. As can be seen, these tasks nicely resemble the three SON operations tasks presented in this work: SON management, SON coordination, and SON self-healing.

The common goal of network operations towards which the SON functions should work, is provided as an operator policy containing high-level performance objectives like target user performance and cell performance values and coordination-specific aspects [Kür+10]. Thereby, the performance objectives can depend on the cell type, location, and other factors. Furthermore, the policy defines trade-offs between competing objectives, e.g., weights between metrics or priorities between functions. The operator policy is refined into cell-specific policies, each capturing the performance objectives relevant for the specific cell. Each cell-specific policy is further translated into SON function-specific policies that control the behavior of the SON functions. Besides that, the operator policy is also refined into a SON coordinator-specific policy which controls conflict detection and resolution between the SON functions at run time as well as the detection of undesired behavior. This translation and harmonization process is performed by a component referred to as policy function. As can be seen, this PBM concept enables an autonomic operations of a SON along operator objectives.

In summary, the interesting ideas presented in SOCRATES have the potential to overcome the manual gap of SON operations. However, the project did not elaborate the concept beyond these rather abstract ideas. This is not surprising since the project's focus was the development of the SON functions themselves. In particular, the policy function, i.e., the derivation of the SON function-specific and coordination policies, is only identified as a task that is not straightforward and an open issue [Sch+11][Kür+10]. Although [Cru+11] presents some additional abstract implementation considerations, a lot of questions regarding the accomplishment of efficient SON operations where not answered [Ban11]. The ODSO concept was inspired by their work, picked up some ideas, and refined them into a holistic framework for autonomic SON operations.

3.6.2.3 UniverSelf Project

The UniverSelf research project [Uni10] was running from 2010 until 2013 with the support of the European Union under the 7th Framework Program. It brought together 17 partners from industry and academia from 10 countries with the "aim of overcoming the growing management complexity of future networking systems, and to reduce the barriers that complexity and ossification pose to further growth" [Uni10, "About UniverSelf"]. Therefore, the project developed the Unified Management Framework (UMF) and applied it to six use cases including "SON and SON Collaboration according to Operator Policies" [Uni12b, p. 1] which comprised the devel-

opment of SON functions as well as an operations approach for their management and coordination [Uni12b].

UMF [Tsa+13] is a framework for network management based on Network Empowerment Mechanisms (NEMs), i.e., automatic functions that solve specific problems. An NEM can be seen as a generalization of a SON function beyond mobile networks. The NEMs are controlled, i.e., operated, by the UMF core comprising of a governance component that configures the NEMs with a policy, the coordination component that coordinates the concurrent execution of the NEMs, and a knowledge component that collects, reasons, and disseminates information about the network. Between these three components, there is a well-defined exchange of information and control: the governance component sets the coordination policy as well as the information collection policy, the coordination informs the governance if it experiences abnormal behavior, and the knowledge component provides information to all components. Obviously, these components can be mapped to the ODSO components SON management and SON coordination, as well as the technical models and the operational context. Furthermore, some kind of self-healing as envisioned in ODSO is performed by the UMF coordination component as well.

The governance component provides a human-to-network interface allowing the operator to easily define high-level policies for the operation of the network. These policies are action policies defined over abstract concepts. Based on that, the governance component performs a policy refinement over three layers of abstraction in order to derive policies that specifically control the NEMs, i.e., SON function configurations. Thereby, possible conflicts in the policies are resolved as well.

In comparison with ODSO, UMF provides a more complex framework that can be applied to a broad spectrum of use cases. It can be fitted to SON operations and provides a similar architecture to ODSO. The latter, however, has been specifically developed for SON and, so, accounts for the specific characteristics of SON operations. Actually, [Uni12d][Alt+13] outline proof of concepts for SON management and SON coordination. However, the utilized action policies inherently do not allow an autonomic PBM as it is envisioned by ODSO.

3.6.2.4 COMMUNE Project

The COMMUNE research project [COM11] was running from 2011 until 2014 with the goal to develop a solution for network management under uncertainty "due to the inherent complexity of the network" [COM11, "Scope of COMMUNE"]. It involved 12 partners from industry from 5 different European countries and was supported by Celtic-Plus. As a result, the project developed a generic autonomic management framework and applied it in 4 scenarios: SON, machine-to-machine communications, fiber-to-the-home, and peer-to-peer overlay streaming.

The autonomic framework developed in COMMUNE is called Generic ARchitecture of Self-Organized Networks (GARSON) [Luo+13]. It divides cognitive functions into six parts: the Autonomic Monitoring Plane monitors the network and computes statistics, the Knowledge Plane contains the logic to make decisions based on the monitored data, the Autonomic Actuating Plane executes the decisions, the Cogni-

tive Plane performs machine learning and adapts the logic of the Knowledge Plane, and the Policy Control Plane enables management of all parts. As can be seen, these planes to some extent overlap with the steps of the MAPE loops. The policy for controlling the GARSON functions is an action policy that either triggers specific actions or set some thresholds of the functions in response to some events. Since GARSON foresees a hierarchical architecture of the autonomic network, comprising domain level, inter-domain level, and top level, the policy may need to be refined from a high-level policy to a technical policy.

Despite the scenario of autonomic management of SON, COMMUNE did not provide a consistent approach for SON operations as ODSO does. Instead, the focus laid on developing cognitive, i.e., self-learning, SON functions that fit into the general GARSON framework. Regarding RAN operations, COMMUNE developed a self-healing, coordination, and energy savings function [Bar+13a]. However, it remains unclear how these functions are operated.

3.6.2.5 SEMAFOUR Project

The SEMAFOUR research project [SEM12], running from 2012 to 2015, aimed to develop a "unified self-management system, which enables the MNOs to holistically manage and operate their complex heterogeneous mobile networks" [SEM12]. It involved eight partners from industry and academia from six different European countries and was supported by the European Union under the 7th Framework Program. SEMAFOUR had two main outcomes: on the one hand, develop a set of multi-RAT / multi-layer SON functions for HetNets, and, on the other hand, conceive an integrated SON operations system, referred to as integrated SON management.

Although a considerable part of the idea of the ODSO approach where brought into the SEMAFOUR project, the resulting architecture evolved differently. SEMAFOUR defines four components for SON operations [Cam+15]: Policy-based SON Management (PBSM) that corresponds to ODSO management, SON coordination that corresponds to ODSO coordination and self-healing, monitoring and diagnosis that corresponds partially to the ODSO operational context, and the decision support system that performs network planning and has no correspondence in ODSO. In contrast to ODSO, only the PBSM component is controlled with a policy. The coordination component, however, is supposed to receive priorities for the SON functions from the PBSM component which derived them from the policy. However, this process is has not been investigated.

The policy for controlling the PBSM is a goal policy defined on network cell-level KPIs [Göt+15]. Thereby, the KPI objectives can be context dependent and weighted according to their importance. The context is reduced to a predefined number of context classes that combine the values for several possible context properties. The technical model to derive the SON configuration is similar to the ODSO technical models, with the exception that they are deterministic. This makes them to some extent incorrect in the probabilistic environment. In contrast to ODSO the goal policy approach used in SEMAFOUR does not provide fully autonomic control as

the utility policy approach adopted in ODSO. Furthermore, ODSO provides a more consistent architecture in which all operations components are controlled by the objectives instead of a derived configuration.

4 SON Management

This chapter introduces the ODSO SON management component. It performs the task of configuring the SON functions in a way that they optimize the network configuration such that the satisfaction of the operator objectives is maximized. Therefore, we first outline the problem that manual SON management requires considerable efforts due to the limited knowledge of the operator about the SON functions and their complex interactions. Based on this, we provide a detailed description of Objective 2 that aims for an autonomic SON management. We describe how the generic ODSO component design can be applied to the SON management task of SON operations. Thereby, it is shown how several technical models, referred to as SON function models, from different SON function vendors can be integrated into a consistent SON configuration. This design makes up Solution and Contribution 2.

4.1 Problem and Motivation

Chapter 3.1.1 has shown that the primary goal of MNOs is to operate the SON such that the satisfaction of a set of KPI-related, weighted, and context-dependent objectives is maximized. However, SON functions do not understand objectives, i.e., they cannot be configured with these objectives. Instead, their behavior is defined through SON function-specific configuration parameters. On the one hand, these parameters typically comprise thresholds on performance measurements and indicators for the execution of the function, referred to as targets [3GP12]. On the other hand, operators need to adjust the algorithms of SON functions via parameters or policies to their operational context [NGM10], as well as to other SON functions to improve their joint optimization [NGM14]. [Ben+13b] has shown that different SON Function Configurations (SFCs)[1], which refers to a specific tuple of values for these parameters, can lead to different behavior of SON functions resulting in clearly distinguishable network performance in terms of KPI values. In other words, the SON functions need to be configured in the right way in order to produce the network performance that the operator desires.

SON management "refers [...] to setting targets, configuring the SON function behaviour at a high level, and monitoring SON function results rather than directly changing the low-level configuration and monitor low-level performance indicators for the SON function." [Wal+11, p. 61] That is, it is the process of transforming the operator objectives into SFCs for all SON functions such that the satisfaction of the

[1] In this report, an SFC is termed SON Function Configuration Parameter Value (SCV) set.

objectives is maximized. As explained in Chapter 3.1, this is currently done by a human operator which leads to a special form of the manual gap of SON operations, referred to as *manual gap of SON management*.

In order to configure the SON functions correctly, human operators need to estimate the expected network performance of a specific SFC which requires precise technical knowledge about the SON algorithms and configuration parameters. Although the MNO typically knows his network, the knowledge of the algorithmic details of a SON function is typically limited. This is caused by the fact that the algorithms and configuration parameters of SON functions are not standardized but defined by each vendor. However, this information is generally considered as intellectual property of the vendor and they are reluctant to disclose all details of their algorithms since the development requires considerable research efforts. Even worse, the behavior of a SON function with respect to its configuration is typically non-linear and non-monotonic. As a result, the MNOs often have a limited knowledge about what is exactly going on in a SON function. In other words, the SON functions are often considered as *black-boxes* by the operators (see Chapter 3.1.2).

A SON is not supposed to be a monolithic system with fixed features dubbed SON functions. Instead, it is thought of as a flexible framework in which a set of independent and self-contained SON functions automatically optimize the network. For instance, any subset of the SON functions CCO, MRO, MLB, and ESM introduced in Chapter 3.3.2 may be deployed and active. Thereby, it is even possible to combine SON functions by different vendors together. Especially these multi-vendor SON environments can be complex to operate manually as the different SON functions interact with each other. However, this raises another problem for SON management: even if the MNO would know the algorithmic details of the SON functions, the fact that several, interacting SON functions are executed in parallel in the network requires the even more difficult estimation of their combined performance.

Hence, SON management is faced with the problem to combine the technical expertise of the deployed SON functions in order to come up with an overall configuration of all SON functions that satisfies the operator objectives. This SON configuration needs to be consistent, i.e., the SFCs must be conflict-free. Two SFCs might be conflicting if the respective SON functions are configured with conflicting or even opposite goals. For instance, consider an MLB function that aggressively performs load balancing by drastically tuning the CIO which might deteriorate the handover performance. If an MRO function is configured to improve the handover performance aggressively as well, then this may lead to an instable network configuration since MLB and MRO are changing the CIO back and forth. If the goals of two SON functions defined by their SFCs do not match, this is referred to as a conflict in SON management. Consequently, it must be ensured that the effects of the SON functions match together. This specific SON function conflict detection is sometimes referred to as SON co-design or harmonization in related work (see Chapter 2.1.5). The NGMN already recognized this problem and created a set of guidelines for developing co-designed SON functions [NGM14].

Because of the difficulty of predicting the result of a change in the SON configuration, MNOs often resort to a default SFC for each SON function that is created

by the vendors according to unified, network-wide MNO requirements during a trial in the network (see Chapter 3.1.2). Consequently, each SON function is configured in a uniform, context-independent way over time and location. This uniform and static configuration leads to non-optimal network performance with respect to the real, context-specific objectives due to the automation and dynamics gap as outlined in Chapter 3.1.3.

4.2 Goals and Requirements

The main goal of objective-driven SON management is to close the manual gap by applying the generic ODSO component design as described in Objective 1. The automation gap is closed by defining a model of the SON function behavior, referred to as SON function model, which allows automatic reasoning by the system. Since the required technical knowledge is usually not available to the MNO, the technical expertise for SON management is supposed to be provided by the SON function vendors. However, since they are reluctant to disclose all their intellectual property, SON management must be able to work with as little information as possible about the SON functions. As a result, ODSO SON management needs to define a standardized, minimal form for SON function models that allows them to keep the SON function algorithms undisclosed while providing the operators with the necessary knowledge to operate the functions. This approach shifts the efforts for building up the technical expertise about the SON functions from the MNOs to the vendors, thus, saving the MNOs huge costs. Fortunately, the vendors need to have the technical knowledge for the SON function development anyway. Since the flexibility of SON in terms of options to combine actual functions, the technical expertise regarding the SON functions cannot be provided for a specific set of functions. Instead, there must be a SON function model for each single SON function.

Given the vendor-provided SON function models, the operator does not need any detailed technical knowledge for managing the SON functions. Instead, ODSO SON management provides a common interface for the operator to manage the whole SON, i.e., all SON functions together, via the context-dependent operator objectives. Since these are provided in form of a formalized objective model as presented in Chapter 3.4.3, the dynamic gap can be closed. Furthermore, this concept separates the knowledge domains of operator and vendor and enables both to evolve independently (see Chapter 3.5).

It is the task of ODSO SON management to combine the separate SON function models and determine a consistent SON configuration that maximizes the satisfaction of the operator objectives. Thereby, it needs to ensure that the different SFCs do not conflict with each other. Furthermore, SON management must be integrated into the overall ODSO architecture as outlined in Chapter 3.3.3. Hence, it must compute and provide the expected SON function effects to SON coordination and SON self-healing.

4.3 Component Design

This section presents an objective-driven SON management approach based on the generic ODSO component design in order to enable SON management through operator objectives. Instead of a uniform and static configuration, the approach provides capabilities for a differentiated and dynamic configuration of SON functions whilst ensuring a conflict-free overall configuration of the SON.

Figure 4.1 depicts an overview of the three-step, computational process in the SON management component. As outlined in Chapter 3.3.3 and depicted in Figure 3.10, the process is triggered by some external event. It produces a set of SFCs for configuring the SON functions, and the SON function effects for the SON coordination and SON self-healing component. The whole process, as presented here, is performed in the context of a specific network cell that might need to be reconfigured. This cell is given by trigger. Therefore, if several cells, or the whole network, needs to be reconfigured, this process is executed multiple times, once for each cell.

The first step, the SON function model combination, is started by the external trigger event which indicates the cell to configure. The SON function model combination corresponds to the action proposal step in the generic ODSO component design. Hence, it requires a technical model which allows deriving possible actions and their effects. In the case of SON management, a possible action corresponds to a set of SFCs, exactly one for each SON function that is deployed in the network, referred to as SON configuration. The technical model is actually a set of SON function models, exactly one for each deployed SON function. Each SON function model describes one SON function and provides information about possible SFCs and respective effects on the KPIs in a context-dependent manner. Based on these models and the operational context, the SON function model combination can determine the feasible SFCs from the different SON function models, combine them to conflict-free SON configurations, and estimate the resulting KPI effects of each SON configuration.

The second step of the process, the SON configuration selection, corresponds the action selection phase in the generic ODSO component design. It evaluates the effects of the SON configurations with respect to the objective model and selects the most preferred SON configuration in the operational context of the considered cell.

In the third step of the process, the selected SON configuration needs to be enforced, i.e., deployed to the network. Besides that, the SON function effect computation creates a mapping for the expected effects of the SON function with their new configuration that is provided to the other ODSO components.

4.3.1 Trigger

A trigger event indicates that the SON functions for a specific cell $c \in C$, which is contained in the event, might need to be reconfigured by the SON management component. This can happen for three reasons:

- The operational context of c has changed. As stated in Chapter 3.3.1.3, this

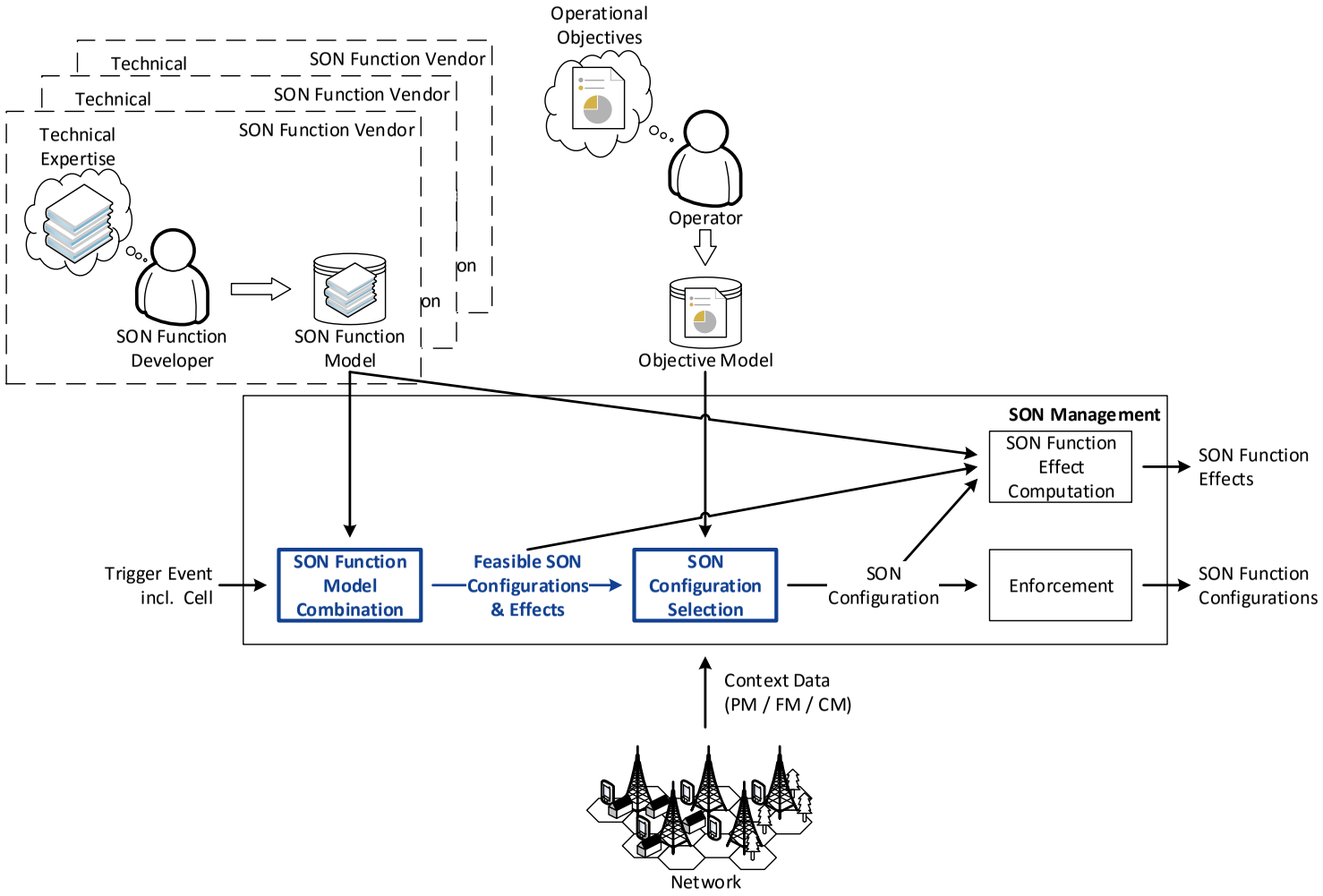


Figure 4.1: The decision making process of the SON management component. The steps of the generic ODSO component design are drawn in blue.

comprises changes in the PM, CM, FM, and operational data. A change can result in different feasible SFCs for the SON functions, different KPI effects of the SFCs, different applicable KPI objectives, and, finally, a different SON configuration. A common example is the transition from the busy hours period of the day to the non-busy period.

- c is a newly activated network cell and the SON needs to be initially configured for it. This can be seen as a special self-configuration use case. However, this case may also apply to network cells that are turned on after being off for energy saving. Since SON functions may be disabled for them, they need to be reconfigured according to the new operational context of the network cell.

- If a new SON function is deployed to the SON then it needs to be initially configured for all cells $c \in C$. Therefore, the SON function model for the new function is added to the system and SON management needs to be triggered for all network cells.

The generation of the trigger events needs to be aligned with the operational procedures of the MNO as well as the computational capabilities of the SON system. Thereby, three cases can be distinguished:

- The insertion of a new network cell and the addition of a new SON function are distinctive and rare events. So, it is possible to trigger the system for each of them immediately. This results in one trigger event for a new network cell. In case a new SON function is deployed, it requires the creation of several trigger events, one for each cell in the network. However, the operational procedures can require that changes in the configuration of the SON are only performed at specific maintenance times, e.g., every full hour. Then it is necessary to delay the trigger event generation till the next maintenance time.

- The operational context of a network cell is changing continuously, e.g., the time is progressing. Hence, each network cell potentially requires a reconfiguration at every time instant. This is neither reasonable nor feasible due to the required computational efforts. If the MNO has defined specific maintenance times, the configuration of every cell can be updated then. Therefore, it is necessary to create a trigger event for each cell in the network at each maintenance time. If the operational procedures allow for continuous configuration changes, it is still reasonable to limit the reconfiguration intervals to, e.g., one hour. Anyway, the lower limit for the reconfiguration interval is given by the granularity period of the network, since this is the smallest interval between two activations of a SON function as described in Chapter 3.3.2. Notice that this approach limits the worst case adaptation time to a change in the operational context to the reconfiguration interval duration.

- The reactivation of a network cell from energy saving might happen quite often. Especially small cells are often considered to be turned on and off whenever the network load requires it, i.e., several times an hour [Las+11]. However,

if the SON is configured in fixed time intervals then it does not make sense to compute a configuration twice within one reconfiguration interval. Instead, the SFCs determined within one interval should be cached and reused in that same interval.

The generation of the trigger events provides optimization potential along two contrary dimensions: reducing the number of generated trigger events leading to no change in the SON configuration of a network cell whilst also reducing the worst case reaction time to changes in the operational context.

4.3.2 SON Function Model

A SON function model is a part of the technical model for SON management: it describes the effects of one specific SON function. In principle, given the algorithm of a SON function, the operator would be able to predict the effect of a SON function quite accurately using real network simulations. However, this approach requires the vendors of a SON function to fully reveal their intellectual property, thereby, potentially enabling others to copy the algorithms. Hence, vendors are typically reluctant to disclose such information so that the SON functions appear to the operator like black boxes. As a result, the SON function model is designed to be a minimal description of how a SON function works: it provides as little information as possible to keep the vendor's intellectual property closed, while also providing as much information as necessary to manage a SON function. For SON management, it is necessary to know at least a set of reasonable SFCs for the SON function that are distinctive with respect to their effects, and an indication of their resulting network performance. That is, the SON function does not model the behavior of a SON function but its effects in terms of expected KPI values under some SFC.

Definition 4.1 (SON function configuration)**.** A SON Function Configuration (SFC) $g_s \in G_s$ is a collection of values for the configuration parameters of the SON function $s \in S$. That is, it configures the SON function and completely determines its behavior. G_s is the set of all valid SFCs for s.

The applicability as well as the effect of an SFC can be dependent on the operational context. This enables, e.g., a specifically adapted SFC for MRO in a high mobility area like a highway. Actually, this context-dependency is one reason that SON functions need to be configured at all. The dependency must be reflected in the SON function model. The expected network performance for an SFC is defined as a partial effect (see Definition 3.9). It defines the expected, probabilistic KPI values that the execution of the SON function with the SFC in the given operational context yields. Thereby, it is possible to express that a SON function does only influence a subset of all KPIs, e.g., that MRO is only affecting the handover ping-pong rate.

Definition 4.2 (SON function model)**.** The SON function model $\mathrm{SFM}_s : C \times \mathbf{X} \to \mathcal{P}(G_s \times \mathbf{F}^{\perp})$ for the SON function $s \in S$ is a mapping from a network cell $c \in C$ and an operational context $\mathbf{x} \in \mathbf{X}$ to a set of pairs $(g_s, \mathbf{f}^{\perp}) \in G_s \times \mathbf{F}^{\perp}$ representing the applicable SFC $g_s \in G_s$ and its partial effect $\mathbf{f}^{\perp} \in \mathbf{F}^{\perp}$ on c.

The SON function model SFM_s for a SON function s is assumed to contain one special SFC: the deactivation of s. Consequently, this SFC $g_s^{\mathrm{off}} \in G_s$ should be feasible independently of the context, and its assigned effect $\mathbf{f}^{\perp}$ should have no defined KPI effect, i.e., $\forall k \in K.\, \mathbf{f}^{\perp}(k) = \perp$.

One way to represent a context dependent SON function model is through an action policy (see Chapter 3.4.1.3), leading to a SON function model consisting of rules like:

```
IF condition(cell, context) THEN sfc YIELDS effect
```

whereby condition(cell, context) is some condition on the cell $c \in C$ and the current operational context $\mathbf{x} \in \mathbf{X}$, sfc is an SFC $g_s \in G_s$ for the SON function $s \in S$, and effect is a partial effect $\mathbf{f}^{\perp} \in \mathbf{F}^{\perp}$. As in each rule system, conflicts can arise meaning that two rules are proposing different effects for the same SFC (see Chapter 2.2.1). Considering the semantics of a SON function model, it seems clear that a conflict resolution based on the specificity of the rules is most appropriate: the rule that has the most specific context condition should be accepted.

The generation of the SON function model requires detailed information about the algorithm of the SON function. Since this knowledge is usually kept secret by the vendor of the function, it is the duty of the manufacturer of the SON function to create the SON function model and provide it to the MNO. Thus, the model can be seen as part of the documentation of the SON function which provides a number of example SFCs for the SON function and the resulting KPI expectations. There are several approaches how the vendor can create the model:

Expert knowledge: The SON function model can be created based on the knowledge of the experts that develop the SON function. This is possible since the developers often have specific goals in terms of KPI values in mind when they conceive the algorithm. Furthermore, they can precisely state in which operational contexts an SFCs should not be used. However, the expert knowledge that can be gathered is typically not quantitative but qualitative. That is, the developers are often not able to define exact probability distributions over the KPI values. Instead, they are able to state value ranges in which the KPI values are predicted to be or that a KPI value will be maximized or minimized [FLS14a]. Hence, the model is not supposed to be perfectly accurate. Consequently, the SON function model derived from these statements may contain effects with uniform and triangular probability distributions (see Figure 3.12).

Simulations: The simulation of a SON function in different situations, i.e., operational contexts, is part of the common development process of SON functions. Drawing on this, it is possible to create a detailed SON function model by simulating the execution of the SON function with a number of different SFCs in different operational contexts. In [Hah+14][HK14][Ben+13b][Göt+15], the authors provide results of an evaluation of this approach. Thereby, they outline the difficulties that need to be overcome, specifically the need for realistic simulations to expose subtle effects and the usage of smart search strategies

to make the model creation computationally tractable. Although, the work on the approach is ongoing and not finalized yet, we believe that this approach can, at least, provide a reasonable starting point. Anyway, it is supposed to be more accurate than a model taken from expert knowledge. The result of the simulations are quantitative expected KPI values. Although they could, in principle, be reduced to qualitative statements as the expert knowledge, it is reasonable to capture the more detailed data. Hence, the SON function model can contain probability distributions that are fitted to the simulation results, e.g., a normal or an arbitrary distribution as shown in Figure 3.12.

Real Data: The most reliable source of information about the expected KPI values are measurements in a real network. However, it is often impossible to create a SON function model solely on this since it is impracticable to test and measure the KPIs for a huge number of SFCs in all interesting operational contexts: on the one hand, this would require a huge number of trials taking months to execute and, on the other hand, MNOs are not willing to test drive potentially harmful configured SON functions in their network since this might lead to inferior user satisfaction. However, given that a SON function is configured by a SON function model created with one of the former approaches, it is possible to monitor the execution of the SON function, collect the actual KPI values, and use this data to improve the accuracy of the SON function model. [LSH16] outlines this approach in more detail. Given the data from the real network, either the operator improves the SON function model himself by adapting the vendor-provided one, or the data is used by the vendor to create a new version of the SON function model which is then distributed to the customers.

4.3.3 SON Function Model Combination

A SON function model defines the behavior of one SON function in terms of the expected KPI values produced by the SON function with a specific SFC in a specific operational context. The task of the SON function model combination is to evaluate the different SON function models, determine non-conflicting combinations of SFCs for all SON functions, and estimate the effects of these SON configurations. This step of the SON management process is related to the action proposal in the generic ODSO component design. Consequently, a SON configuration corresponds to an action in the generic ODSO component design (see Definition 3.5). Hence, the symbol a is reused and, in this chapter, always refers to a SON configuration.

Definition 4.3 (SON configuration)**.** In the context of SON management, an action as defined in Definition 3.5 is refined to a SON configuration. A SON configuration $a : S \to \bigcup_{s \in S} G_s$ is a mapping from each SON function $s \in S$ to a valid SFC $g_s \in G_s$. The respective network cell for the SON configuration is not explicitly encoded in a, but instead implicitly given by the trigger event. Notice that the SFC for a SON function is always from the respective domain, i.e., $\forall s \in S. a(s) \in G_s$. The set of all possible SON configurations is A.

The final result of the SON function model combination is the creation of a set of feasible SON configurations and their effects. Therefore, the formalization presented in Definition 3.10 is adopted:

Definition 4.4 (Feasible SON configurations and effects)**.** In the context of SON management, the action-effect set as defined in Definition 3.10 is refined to the set of feasible SON configurations and effects. The set of pairs of feasible SON configurations a and their respective complete effects $\mathbf{f}$ is defined as

$$\begin{aligned}\mathrm{AF} = \{(a, \mathbf{f}) \mid & a \in A \text{ is applicable and conflict-free,} \\ & \mathbf{f} \in \mathbf{F} \text{ is the expected combined effect of } a\}.\end{aligned}$$

For a pair $(a, \mathbf{f}) \in \mathrm{AF}$, $\mathbf{f}$ is the expected resulting effect on the network cell $c \in C$ that the SON function model combination is executed for.

4.3.3.1 Conflict Detection

The set of all valid SON configurations can be easily created as the cross product of the actions proposed by all SON function models. However, this set may contain SON configurations that have conflicts, i.e., two or more SFCs in the configuration are conflicting. The term conflict among SON functions is mainly known from SON coordination and refers to "negative interactions between SON functions" [Ban+11b, p. 322] which may decrease the network system performance. The determination of such conflicts from a SON configuration is difficult. Most of the conflicts considered by SON coordination are concurrency-related and caused by parallel execution of several SON functions. Since these are often not caused by the SON configuration, the SON management component has no means to detect and avoid them. However, another, subtle type of conflict is that SON functions are working competitively against each other instead of cooperatively with each other. Specifically in the ODSO framework this means that two or more SON functions optimize the same KPI to different target values. Such a setting can cause oscillations, i.e., the SON functions are alternately optimizing the KPI to their targeted value [Sch+11][NGM14]. Since the SON function models encode the effects of the different SFCs, the SON function model combination is particularly able to detect this type of conflict. This may ensure that the SON is configured in such way that the optimization targets for all SON functions are harmonized.

The determination of possible conflicts caused by the SON configuration is a complex task given the hardly predictable physical environment and the dynamics of concurrently executed SON functions. Given the predicted effects of an SFC by the SON function model, SON function model combination may, at least, make an estimation for conflicts in order to avoid the most obvious misconfigurations. The idea is that SFCs for different SON functions are conflicting if they cause the respective SON functions to optimize the same KPI to different values, i.e., their effects for at least one KPI do not agree sufficiently. In the following, this idea is outlined for two SON functions, s_1 and s_2, that are in conflict with respect to a KPI $k \in K$:

1. The effects of the SON functions on k, $f^{\perp}_{k,s_1} \neq \perp$ and $f^{\perp}_{k,s_2} \neq \perp$, are seen as independent probabilistic processes that influence the actual value $v \in \mathrm{Dom}(k)$ of k. For each $k \in K$, V_{k,s_1} and V_{k,s_2} represent random variables for the future values of k according to the effects $f^{\perp}_{k,s_1}$ and $f^{\perp}_{k,s_2}$, respectively. Hence, the probability that a SON function produces a KPI value within a value range Y is $\Pr(V_{k,s_n} \in Y) = \int_Y f^{\perp}_{k,s_n}(v)\,\mathrm{d}v$ for each $n = 1, 2$.

2. Based on this, a KPI value v is not accepted by any one of the SON functions, say s_1, if v is not a possible value of its effect, i.e., $f^{\perp}_{k,s_1}(v) = 0$. The reasoning behind this idea is the following: if v is not a possible outcome of a SON function that affects k then this is caused by the fact that the SON function avoids v since it optimizes k towards other values. In other words, v is not an element of the goal of the SON function.

3. The set of mutually accepted KPI values by s_1 and s_2 is then
$$\{v \mid v \in \mathrm{Dom}(k), f^{\perp}_{k,s_1}(v) > 0, f^{\perp}_{k,s_2}(v) > 0\}. \tag{4.1}$$

Definition 4.5 (Agreed values of partial KPI effects)**.** Given a set $\mathcal{F}_k \subseteq F_k$ of partial KPI effects $f^{\perp}_k \in \mathcal{F}_k$ for one KPI $k \in K$. The set of agreed values for the partial effects is defined as

$$V_{\mathrm{agree}}(\mathcal{F}_k) = \{v \mid v \in \mathrm{Dom}(k), \forall f^{\perp}_k \in \mathcal{F}_k.\, f^{\perp}_k = \perp \vee f^{\perp}_k(v) > 0\}.$$

Notice that $V_{\mathrm{agree}}(\{f^{\perp}_{k,s}\})$ produces the set of KPI values $v \in \mathrm{Dom}(k)$ of k for which the effect of s defines a non-zero probability. This set is always non-empty: On the one hand, if the partial KPI effect $f^{\perp}_{k,s} = \perp$ then it agrees on all values, i.e., $V_{\mathrm{agree}}(\{\perp\}) = \mathrm{Dom}(k)$. On the other hand, if the partial KPI effect $f^{\perp}_{k,s} \neq \perp$ then $\int_{\mathrm{Dom}(k)} f^{\perp}_{k,s}(v)\,\mathrm{d}v = 1$ (see Definition 3.6) and, thus, at least one KPI value must have a non-zero probability, i.e., $\exists v \in \mathrm{Dom}(k).\, f^{\perp}_{k,s}(v) > 0$.

A simple approach to determine an agreement based on $V_{\mathrm{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\})$ is to calculate the ratio of the cardinality of agreed to all possible KPI values, i.e.,

$$\frac{\left|V_{\mathrm{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\})\right|}{\left|V_{\mathrm{agree}}(\{f^{\perp}_{k,s_1}\}) \cup V_{\mathrm{agree}}(\{f^{\perp}_{k,s_2}\})\right|}. \tag{4.2}$$

However, this would ignore the probabilities of the KPI values. Consider that a small ratio of values may be the outcome of s_1 but not accepted by s_2. If the probability that s_1 produces such a value is low then this small disagreement may still be fine for SON management. However, if the probability that s_1 produces such a value is high then this small disagreement is definitely considered as a conflict. Hence, we pursue another approach and continue in the following way:

4. The agreement can be calculated as the probability, that the outcome of each random process is accepted by the other process. The probability that $f^{\perp}_{k,s_1}$ produces an agreed value v is given by:
$$\Pr(V_{k,s_1} \in V_{\mathrm{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\})) = \int\limits_{V_{\mathrm{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\})} f^{\perp}_{k,s_1}(v)\,\mathrm{d}v. \tag{4.3}$$

5. Since the two random processes are supposed to be independent, the overall probability that the outcome of $f^{\perp}_{k,s_1}$ and $f^{\perp}_{k,s_2}$ is in $V_{\text{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\})$, is then the product of the marginal probabilities

$$\begin{aligned}&\Pr(V_{k,s_1} \in V_{\text{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\}), V_{k,s_2} \in V_{\text{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\}))\\ &= \Pr(V_{k,s_1} \in V_{\text{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\})) \cdot \Pr(V_{k,s_2} \in V_{\text{agree}}(\{f^{\perp}_{k,s_1}, f^{\perp}_{k,s_2}\})). \quad (4.4)\end{aligned}$$

Definition 4.6 (Agreement of SFCs)**.** The agreement of a set $\mathbb{F} \subseteq \mathbf{F}^{\perp}$ of partial effects $\mathbf{f}^{\perp} \in \mathbb{F}$ is the minimum of the agreements of all respective KPI effects, i.e.,

$$\text{agree}(\mathbb{F}) = \min_{k \in K} \text{agree}_k(\{\mathbf{f}^{\perp}(k) \mid \mathbf{f}^{\perp} \in \mathbb{F}\}).$$

A KPI agreement for a set $\mathcal{F}_k \subseteq F^{\perp}_k$ of the partial KPI effects $f^{\perp}_k \in \mathcal{F}_k$ on the KPI k is calculated as the probability that each outcome is in $V_{\text{agree}}(\mathcal{F}_k)$, i.e.,

$$\text{agree}_k(\mathcal{F}_k) = \prod_{f^{\perp}_k \in \mathcal{F}_k, f^{\perp}_k \neq \perp} \Pr(V_k \in V_{\text{agree}}(\mathcal{F}_k)) = \prod_{f^{\perp}_k \in \mathcal{F}_k, f^{\perp}_k \neq \perp} \int_{V_{\text{agree}}(\mathcal{F}_k)} f^{\perp}_k(v)\,\mathrm{d}v$$

with V_k representing the random variable for the probability distribution $f^{\perp}_k$. This implies that a partial KPI effect $f^{\perp}_k = \perp$ is supposed to produce a $v \in V_{\text{agree}}(\mathcal{F}_k)$ with probability 1.

This conflict detection approach is visualized in Figure 4.2 and Figure 4.3: Figure 4.2 depicts two exemplary KPI effects $f^{\perp}_1$ and $f^{\perp}_2$ as well as their set of mutually accepted KPI values, and Figure 4.3a and Figure 4.3b show the respective probabilities $\Pr(V_1 \in V_{\text{agree}}(\{f^{\perp}_1, f^{\perp}_2\})) = 0.86$ and $\Pr(V_2 \in V_{\text{agree}}(\{f^{\perp}_1, f^{\perp}_2\})) = 0.70$ as the shaded areas. The resulting KPI agreement is $\text{agree}_k(\{f^{\perp}_{k,1}, f^{\perp}_{k,2}\}) = 0.60$. Note that this probability is 1 if both effects are fully overlapping, and it is 0 if the probability distributions are not overlapping. Furthermore, if one SON function has no effect on a KPI k, i.e., $\mathbf{f}^{\perp}_1(k) = \perp$, then the agreement is 1. Given the agreements per KPI, the overall agreement is their minimum, i.e., the minimum probability that the two effects have an equal value for one KPI. Note that it is valid to define a minimum probability density greater 0 for the calculation of $V_{\text{agree}}(\{f^{\perp}_{k,1}, f^{\perp}_{k,2}\})$ which might be useful for comparing, e.g., a normal distributed KPI effect (see Figure 3.12) since such a distribution $f(v) > 0$ for all $v \in \text{Dom}(k)$.

Based on the agreement of the effects, $\mathbf{f}^{\perp}_1$ and $\mathbf{f}^{\perp}_2$, of two SFCs for two SON functions, a potential conflict can arise if $\text{agree}(\{\mathbf{f}^{\perp}_1, \mathbf{f}^{\perp}_2\}) \leq \rho_{\text{agree}}$. Here, ρ_{agree} defines a minimal probability that the two SFCs produce a mutually agreed KPI value. A broad conflict definition would be $\rho_{\text{agree}} = 0$ meaning that the SFCs have no overlap for any KPI at all. On the one hand, this causes only fully contradictory configurations to be rejected. On the other hand, it also avoids to mark too many SON configurations as having conflicts. For instance, consider an aggressive SFC for an MRO function with the simple effects $V_{\text{agree}}(\{\mathbf{f}^{\perp}_{\text{MRO}}(\text{Pipo})\}) = [0.0, 0.02]$ and $V_{\text{agree}}(\{\mathbf{f}^{\perp}_{\text{MRO}}(\text{Load})\}) = [0.8, 0.9]$, and, an aggressive SFC for an MLB function with the simple effects $V_{\text{agree}}(\{\mathbf{f}^{\perp}_{\text{MLB}}(\text{Load})\}) = [0.4, 0.6]$ and $V_{\text{agree}}(\{\mathbf{f}^{\perp}_{\text{MLB}}(\text{Pipo})\}) =$

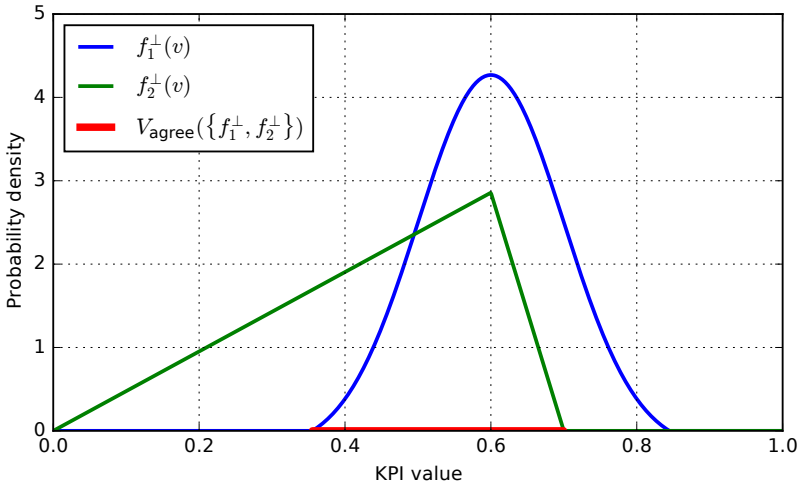


Figure 4.2: The two KPI effects and the agreed KPI values.

$[0.0, 0.2]$. In this example, the effects of both functions are conflicting since their agreement is 0. Particularly the KPI effects on the cell load are not overlapping. Hence, a SON configuration with these SFCs would be rejected.

Based on this, Definition 4.4 of the set of pairs of feasible SON configurations a and their respective complete effects $\mathbf{f}$ produced by SON function combination can be refined.

Definition 4.7 (Applicable and conflict-free actions)**.** A SON configuration $a \in A$ for a cell $c \in C$ in context $\mathbf{x} \in \mathbf{X}$, with $g_s = a(s)$ for all SON functions $s \in S$, is *applicable and conflict-free* if and only if

- a is a valid combination of SFCs defined by the SON function models, i.e.,

  $$\forall s \in S. \exists \mathbf{f}^{\perp} \in \mathbf{F}^{\perp}. (g_s, \mathbf{f}^{\perp}) \in \text{SFM}_s(c, \mathbf{x})$$

  and

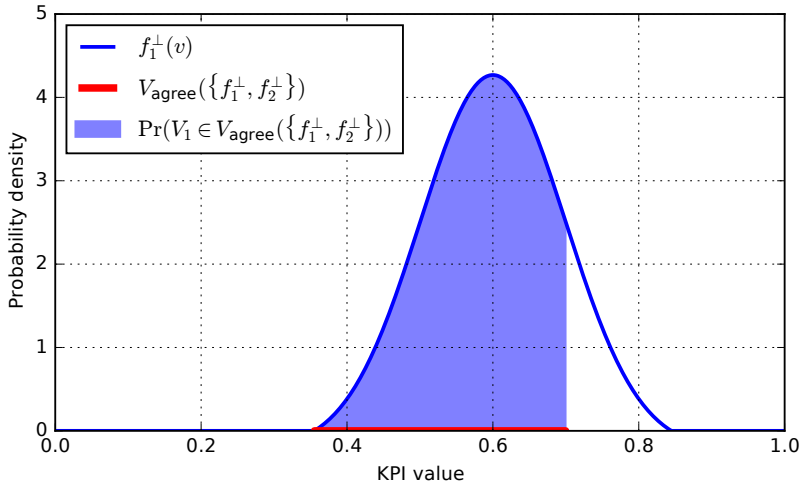
- the effects by all SFCs in a, i.e., the set

  $$\mathbb{F} = \{\mathbf{f}^{\perp} \mid s \in S, (g_s, \mathbf{f}^{\perp}) \in \text{SFM}_s(c, \mathbf{x})\},$$

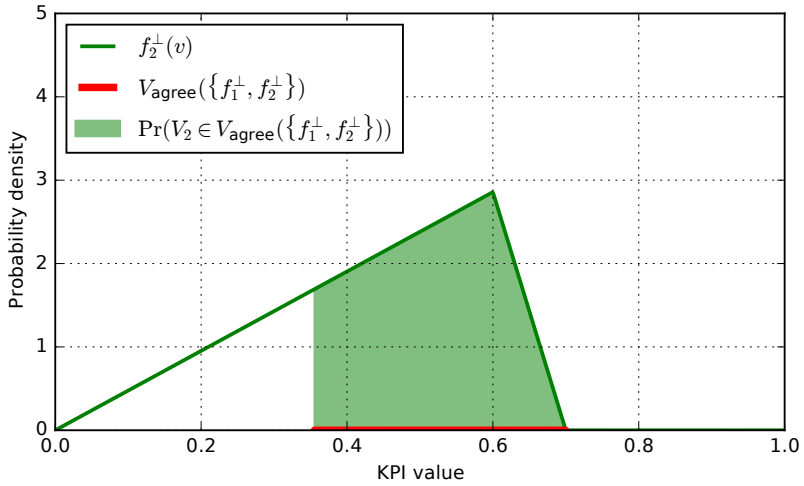
  are not conflicting, i.e.,

  $$\text{agree}(\mathbb{F}) > \rho_{\text{agree}}.$$

The presented conflict detection is solely based on the SON function models without any further information. Therefore, it can be seen as a rough prediction that cancels out the most obvious conflicts. As outlined in [Hah+14][HK14], the interaction between SON functions might be, however, more complex than predicted. Hence, even non-conflicting SON configurations might turn out to cause problems



(a) The probability that f_1 will produce an accepted outcome.



(b) The probability that f_2 will produce an accepted outcome.

Figure 4.3: Visualization of the agreement of two KPI effects.

in the network. Therefore, monitoring the SON behavior could be beneficial to detect and avoid other conflicts. Chapter 6.3.3.1 presents how a detection of ineffective SON functions might be added into the ODSO architecture. Since, this work does not dive into the details of conflict detection, the reader may refer to [Iac+15] for further information regarding this topic.

4.3.3.2 Combined Effect Estimation

Each SON function model SFM_s provides an estimation of the effects of an SFC for the respective SON function s. This estimation is based on the assumption that the SON function is the only function in the network. Since there are several SON functions active in the network, the SON function model combination has to estimate the effects of the combined SON configuration. As with the conflict detection for

SON configurations, the accurate prediction of the effects if several SON functions are executed in parallel is difficult due to the physical environment and the complex interactions between the SON functions. However, with a similar idea, it is possible to provide a rough estimation of the expected combined effect based on the single SON function effects.

Let $a \in A$, with $g_s = a(s)$ for all SON functions $s \in S$, for a cell $c \in C$ in context $\mathbf{x} \in \mathbf{X}$ be an applicable and conflict-free SON configuration for the SON functions S. Since all KPIs are assumed to be independent of each other (see Chapter 3.4.1.2), it is possible to estimate the combined KPI effect $f_k^{\perp}$ for each single KPI $k \in K$ independently of the other KPIs. As for conflict detection, each SON function $s \in S$ with the SFC g_s is assumed to be an independent probabilistic process with the outcome $f_{k,s}^{\perp}$ given by the SON function model. Based on this, it is necessary to define a mapping from the outcomes of all $f_{k,s}^{\perp}$ to a combined effect for the SON configuration $f_{k,a}^{\perp}$. In principle, this mapping could express the complex interactions between the SON functions. However, this would require a lot of effort for the definition, especially considering the possible need to bring together knowledge from different SON function vendors. Hence, a generic approach requiring only the SON function models is adopted.

For the generic effect combination, the basic assumption is that any one of the SON functions that have an effect on k, may determining the combined KPI effect. In other words, one of the functions $s \in S$ affecting k is solely dominating and determining the resulting KPI effect. Let S_k be the set of SON functions that have an effect on k, i.e., $f_{k,s}^{\perp} \neq \perp$. Each function may be dominating with equal probability $\rho = 1/|S_k|$. Hence, the dominating SON function $s_{\text{dominating}} \in S_k$ may produce any KPI value $v \in \text{Dom}(k)$ according to its effect $f_{k,s_{\text{dominating}}}^{\perp}$. That is, the probability that the combined effect produces a KPI value in some value range $W \subseteq \text{Dom}(k)$ is then

$$\Pr(V_{k,a} \in W) = \Pr(V_{k,s_{\text{dominating}}} \in W) = \int_W f_{k,s_{\text{dominating}}}^{\perp}(v)\,\mathrm{d}v \tag{4.5}$$

with $V_{k,a}$ and $V_{k,s_{\text{dominating}}}$ representing the random variables with the resulting probability distribution and $f_{k,s_{\text{dominating}}}^{\perp}$, respectively. Putting this result together with the probability that a SON function is dominating gives

$$\begin{aligned}
\Pr(V_{k,a} \in W) &= \sum_{s \in S} \rho \cdot \Pr(V_{k,s} \in W) \\
&= \sum_{s \in S} \rho \cdot \int_W f_{k,s}^{\perp}(v)\,\mathrm{d}v \\
&= \rho \int_W \sum_{s \in S} \cdot f_{k,s}^{\perp}(v)\,\mathrm{d}v.
\end{aligned} \tag{4.6}$$

Figure 4.4 visualizes this combination approach for the example given in Figure 4.2: the blue and green line depict two KPI effects $f_1^{\perp}$ and $f_2^{\perp}$ and the cyan plot shows the resulting combined KPI effect $f_a^{\perp}$. As can be seen, all KPI values that

are possible for any of the input KPI effects is also possible in the combined effect. Thereby, $\int_{\mathrm{Dom}(k)} f_a^{\perp}(v)\,\mathrm{d}v = 1$.

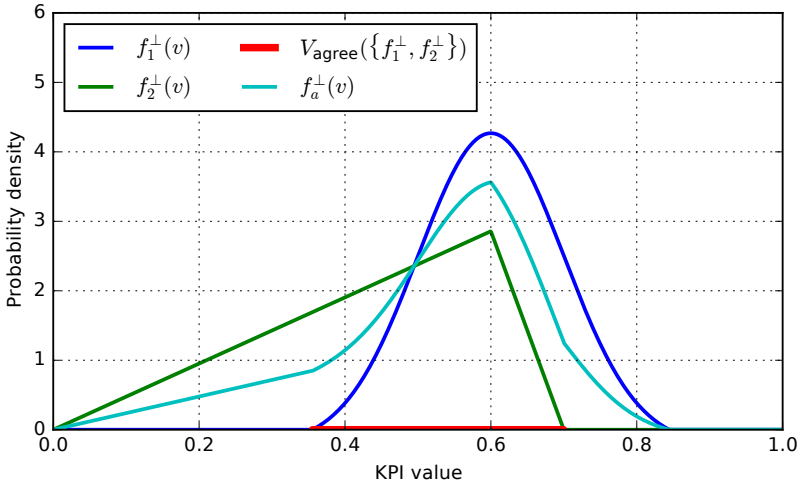


Figure 4.4: Visualization of the combined KPI effect for two KPI effects.

Definition 4.8 (Combined KPI effect)**.** For a SON configuration a and a KPI $k \in K$, let $f_{k,s}^{\perp}$ be the KPI effect for k of the SFC $a(s)$ for a SON function $s \in S$. The set of KPI effects by the SFCs in a, which affect k, is given by $\mathcal{F}_k = \{f_{k,s}^{\perp} \mid s \in S, f_{k,s}^{\perp} \neq \perp\}$. The combined KPI effect of a is given as

$$f_{k,a}^{\perp}(v) = \begin{cases} \dfrac{1}{|\mathcal{F}_k|} \cdot \displaystyle\sum_{f^{\perp} \in \mathcal{F}_k} f^{\perp}(v) & \text{if } \mathcal{F}_k \neq \emptyset \\ \perp & \text{otherwise.} \end{cases}$$

It can be the case, that one KPI is not affected by any SON function, e.g., if the SON configuration contains an SFC which represents that the respective SON function is disabled. In this case, it is reasonable to assume that the effect of the SON configuration on the KPI is equal to the currently discrete, deterministic value of the KPI in the mobile network. This can be formalized using the merging function defined in Definition 3.11. As reasonably expected, a SON configuration with all SON functions turned off (see g_s^{off} in Chapter 4.3.2) would have a combined effect with the current network performance. Alternatively, the KPI effect f_k on KPI k that is not affected by a SON configuration may also be a uniform probability density function with the constant value $f_k(v) = {}^1/_{|\mathrm{Dom}(k)|}$ for all $v \in \mathrm{Dom}(k)$. In other words, every KPI value is expected with equal probability because there is no information at all about k. This approach is suitable if the current network performance is not available for some reason.

Based on this, Definition 4.4 of the set of pairs of feasible SON configurations a and their respective complete effects $\mathbf{f}$ produced by SON function combination can be refined.

Definition 4.9 (Combined effect)**.** A feasible and conflict-free SON configuration a, for a cell $c \in C$ in context $\mathbf{x} \in \mathbf{X}$ has the expected combined effect

$$\begin{aligned}\mathbf{f} =& \mu(\mathbf{f}^{\perp}, c, \mathbf{x}, 1.0) \\ & \text{such that } \mathbf{f}^{\perp}(k) = f^{\perp}_{k,a} \text{ for all KPIs } k \in K\end{aligned}$$

with μ being the effect merging function introduced in Definition 3.11, and $f^{\perp}_{k,a}$ being the combined KPI effect for KPI k as defined in Definition 4.8.

This combined effect estimation is solely based on the SON function models without any further information. Just like the presented conflict detection, it can be seen as a rough prediction. In parallel to the discussion in Chapter 4.3.3.1, this prediction might be improved by learning the actual effects of a SON configuration in the MNO's network. [LSH16] describes a related learning approach.

4.3.4 SON Configuration Selection

The set of feasible SON configurations computed by the SON function model combination represents the options that SON management can choose from. The configuration selection phase decides which option is the best SON configuration with respect to the operational objectives. This step of the SON management process is related to the action selection in the generic ODSO component design.

First, the objective model (see Chapter 3.4.3) is evaluated with respect to the current operational context of the cell $c \in C$ under consideration $\mathbf{x} \in \mathbf{X}$, i.e., $\mathrm{OM}(c, \mathbf{x})$. This provides all applicable KPI objectives and their weights in the context. Using this result, the SON configuration selection can calculate the expected utility vector $\mathbf{u}_a$ for each feasible SON configuration and effect pair $(a, \mathbf{f}) \in \mathrm{AF}$. Since only one SON configuration can be applied to the network cell, the one with the highest satisfaction of the operational objectives is selected.

Definition 4.10 (SON configuration selection)**.** Given the set of feasible SON configurations and effects AF for a network cell $c \in C$ in the context $\mathbf{x} \in \mathbf{X}$, let $\mathbf{u}_{a_i}$ be the utility vector of a_i with the expected combined effect $\mathbf{f}_i$, i.e., $(a_i, \mathbf{f}_i) \in \mathrm{AF}$. SON Configuration Selection selected the most preferred SON configuration a_i such that $\forall (a_j, \mathbf{f}_j) \in \mathrm{AF}.\, a_i \succsim a_j$, i.e., $\forall (a_j, \mathbf{f}_j) \in \mathrm{AF}.\, \mathbf{u}_i \geq_D \mathbf{u}_j$, with $\mathbf{u}_j$ being the utility vector of a_j.

4.3.5 Enforcement

The best SON configuration for a specific network cell needs to be deployed to the SON system. However, in order to avoid unnecessary overhead, this may only be done, if the best SON configuration differs from the current configuration of the cell.

4.3.6 SON Function Effect Computation

As described in Chapter 3.3.3, SON management does not solely configure the SON functions but also provides the other two ODSO components with a model of the

expected effects of each SON function with the current SON configuration for a network cell, referred to as SON function effects. This allows, e.g., SON coordination to estimate the effects on the network performance of two conflicting SON functions. It is necessary to update this technical model with every change of a cell's SON configuration since the effects of the execution of a SON function depend on its configuration. Note that the objective model is not specialized for each cell but instead the general model is used by the other components.

The SON function effects are supposed to represent the long-term effects of the execution of each SON function in a network cell. For a specific SON function with a given SFC, the effect provided by the SON function model is an estimation of the long-term KPI effects if only this function would be active in the network. In contrast to that, the combined effect of the deployed SON configuration for a cell contains the expected effects of all SON functions together. Hence, it also defines KPI effects for KPIs that a specific SON function does not affect. Consequently, the SON function effects need to combine both models in order to give an estimation of the long-term effects of a specific SON function in combination with the other SON functions. Specifically, the KPI effect on KPI k of a SON function s must be the combined KPI effect on k by the SON configuration if and only if s is supposed to affect k according to its SON function model. Otherwise the KPI effect on k must be undefined, i.e., $\perp$.

Definition 4.11 (SON function effects)**.** The SON function effects $\mathrm{SFE} : C \times S \to \mathbf{F}^{\perp}$ is a mapping from a network cell $c \in C$ and a SON function $s \in S$ to the expected long-term partial effect $\mathbf{f}^{\perp}_{\mathrm{SFE}} \in \mathbf{F}^{\perp}$ that the execution of s with the current SFC g_s in c holds. With every reconfiguration of a SON function $s \in S$ for cell $c \in C$ in context $\mathbf{x} \in \mathbf{X}$, this mapping is updated such that

$$\mathrm{SFE}(c, s) = \mathbf{f}^{\perp}_{\mathrm{SFE}} \text{ such that for all KPIs } k \in K$$

$$\mathbf{f}^{\perp}_{\mathrm{SFE}}(k) = \begin{cases} \mathbf{f}(k) & \text{if } (g_s, \mathbf{f}^{\perp}_{\mathrm{SFM}}) \in \mathrm{SFM}_s(c, \mathbf{x}) \wedge \mathbf{f}^{\perp}_{\mathrm{SFM}}(k) \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

for the updated SON configuration a with $g_s = a(s)$, and the expected combined effect $\mathbf{f}$ for a in c according to Definition 4.9. Additionally, we define the shorthand form $\mathrm{SFE}(c, s; k) = \mathrm{SFE}(c, s)(k)$.

4.4 Related Work

The following presentation of related work focuses specifically on automated or autonomic SON management, i.e., the configuration of SON functions. An overview of the research regarding SON operations and PBM is given in Chapter 3.6.

A general survey of the literature regarding SON reveals that the management of SON functions has not been in focus of SON research, yet. On the one hand, it is noticeable that most of the efforts are dedicated to the development of new and improved SON functions. On the other hand, some publications mentioning

"SON management" only cover SON coordination (see [Ban+11b] and Chapter 5.5). Nevertheless, there is a small body of work that touches SON management as defined in this thesis.

4.4.1 SOCRATES Project

The SOCRATES project, introduced in Chapter 3.6.2.2, was among the first that acknowledged the need for SON management: it is "essential as SON grows, to automate the mapping and distribution of high-level performance objectives to low-level policies specific to each SON function" [Sch+11, p. 194]. This includes an alignment of the SON function configurations, referred to as "heading harmonization" [Sch+11, p. 195], in order to minimize the risk for conflicts.

These goals are supposed to be achieved through a policy function which is a part of the developed SON coordination framework [Kür+10]. That is, from SOCRATES's point of view, SON management is not a distinct SON operations task but a constituent part of SON coordination. Besides transforming high-level objectives into SON function specific policies, the policy function is also supposed to configure the other components of the SON coordination framework. However, apart from these rather abstract requirements for the policy function, SOCRATES did not provide any clue about how these might be achieved and implemented. Instead, they acknowledge that "the transformation of high-level performance requirements into SON (Coordinator) function specific policies is not straightforward but requires a detailed understanding of the network and operational experience" [Sch+11, p. 196]. Nevertheless, the SOCRATES project has been an important source of requirements for the ODSO SON management approach.

4.4.2 UniverSelf Project

The UniverSelf project, introduced in Chapter 3.6.2.3, was not specifically targeting SON but aimed at developing a generic self-management framework called UMF. However, the management of SON functions, implemented in the UniverSelf framework, was among its case studies [Uni12d][Uni12c][Uni12a]. Thereby, the governance block is in focus since it enables controlling the SON through high-level business goals by translating them into low-level policies, i.e., SFCs, for the SONs functions.

The governance block is divided into two main functions [Tsa+13]: the human to network function provides a Graphical User Interface (GUI) for the operator that allows the definition of high-level policies and objectives, and the policy derivation and management function translates these high-level goals into low-level configurations while also ensuring that they are conflict-free. The latter is a PBNM-inspired, action policy-based framework which employs the policy continuum [Mee+06]. Although the publicly available documents do not provide much information on the exact transformation process, the examples given in [Gal+12][Kar+13] lead to the conclusion that the policy transformation is an instance of a policy refinement (see Chapter 3.6.1.1) based on semantic technologies. In order to make the framework flexible, the transformation is supposed to be based on meta-data provided by the

SON functions. The governance block also configures the coordination component. The conflict detection aims to detect conflicts between configuration rules on different abstraction levels. In its simplest form, it can detect conflicts like two rules may be triggered together and set two different SFCs for the same SON function instance. In a more complex form, it can also detect more subtle conflicts like contradicting SFCs for two different SON functions. However, this requires the operator to provide a complex conflict detection model, which links the two SFCs, requiring detailed knowledge of the SON functions and their interactions.

The UniverSelf project provides a comprehensive framework for managing autonomic networks. Although the case study for the application of UMF to SON provides a nice show case, details about the implementation are unclear. This regards specifically the question how realistic SON functions, which have not particularly been developed for UMF, may be included in the management concept. In contrast to that, ODSO does not provide a generic framework but focuses on the problem of SON management and provides a more concrete solution. Furthermore, the behavioral abstraction provided by ODSO allows a higher degree of automation and real autonomic behavior in contrast to the conceptual abstraction in UMF. Finally, due to the vendor-provided SON function models, ODSO conflict detection and combined effect estimation can reason for potential conflicts and expected effects for several SON functions, without requiring the operators to specify additional information about the SON functions that they may not have.

4.4.3 COMMUNE Project

The COMMUNE project, introduced in Chapter 3.6.2.4, provided a framework for cognitive management under uncertainty. However, based on the available documents [Luo+13], it is obvious that the focus of the project was on the development of cognitive functions that may operate in uncertain, realistic environments. Hence, the suggested management concept is based on a simple action rule-based PBM system using an operator-provided policy proposing concrete SFCs for the SON functions given some context.

4.4.4 SEMAFOUR Project

The policy-based SON management proposed by the SEMAFOUR project, introduced in Chapter 3.6.2.5, is closely related to the ODSO approach, since a lot of the early ODSO concepts have been developed in conjunction with SEMAFOUR. Based on these basic ideas, ODSO and SEMAFOUR followed different goals: ODSO, on the one hand, concentrates on extending the initial ideas by founding it on MAUT and probability theory, and provides an integrated framework for SON operations. SEMAFOUR, on the other hand, focuses more on the SON function models and provides an initial approach for automatically learning them [LSH16].

In contrast to ODSO, SEMAFOUR is based on a goal policy that defines a weighted acceptable threshold for each KPI [Göt+15][Cam+15]. The effects of SFCs are also defined in SON function models. [Göt+15] describes several approaches for

determining and modeling the effects, however, none of them provides a consistent, probabilistic concept like ODSO, which may lead to a less accurate effect evaluation. This might be one reason why SEMAFOUR finally adopted an automatic learning approach for the creation of the SON function models. However, it should be noted that a probabilistic ODSO SON function model seems also better suited for this machine learning solution.

5 SON Coordination

SON coordination ensures that the concurrent execution of SON functions does not lead to negative performance impacts due to conflicts caused by unwanted interactions between the functions. The task of SON operations has already been identified by research as an important ingredient for a SON system as outlined in Chapter 2.1.5. This chapter introduces a new, autonomic SON coordination approach that performs conflict resolution according to operational objectives. We first outline the problem of classical SON conflict resolution approaches and describe the expected behavior of ODSO SON coordination according to Objective 3. The developed solution enables on-line, pre-action coordination of SON functions along technical constraints as well as operator objectives by applying the generic ODSO component design. Thereby, it ensures that the accepted SON function requests maximize the operator satisfaction even in complex conflict situation. The focus of this approach is not the detection of potential conflicts between SON functions but the decision making for resolving such conflicts. This component makes up Solution and Contribution 3. Notice that a concept for off-line coordination is presented in Chapter 4.3.3.1 as part of the SON management task.

5.1 Problem and Motivation

Independently of each other, SON function continuously monitor measurements from the network and, upon detection of a problem, request the right to optimize the network configuration (see Chapter 3.3.2). If pre-action SON coordination detects a conflict between two or more concurrent SON function requests then this conflict must be resolved in order to avoid negative system performance. The possible resolution options considered here are to accept or reject, i.e., execute or block, the changes requested by a SON function. Based on that, the decision problem of SON coordination is to accept the best SON function requests for execution. However, this decision depends on several aspects.

[3GP13] presents two example scenarios for SON function conflicts and a resolution approach based on priorities:

- In one scenario, the SON functions COC[1], CCO, and ESM[2] request to change the RET of one cell to different values at the same time. Typically, COC would

[1] COC is a self-healing function that aims to compensate faulty cells.
[2] The SON function algorithms in this scenario differ from the ones considered in this work.

have the highest priority since it compensates a network failure and, consequently, would be executed. The reasoning behind this is that self-healing functions recover the network from an abnormal system state to a normal state and, hence, are a prerequisite for self-optimization functions which are only reasonable under normal network conditions. As a result, COC should be executed before any optimization function like CCO or ESM. Similarly, consider the example of a conflict between CCO and MRO. The typical workflow is that after CCO changes the size of a cell, MRO is active in order to adjust the handover settings to the new physical size of the cell. As a result, CCO should be executed before MRO because otherwise, the likely execution order would be MRO, CCO, and MRO again which is inefficient.

- In another scenario, the SON functions MRO and MLB are in conflict since both request the adjustment of the handover parameters within one cell at the same time. In this case, the priorities of the SON functions are highly dependent on the MNO's preferences, i.e., the main network optimization goal being either the handover performance or the throughput performance. A similar decision problem is given by two MLB function instances that request changes on different network cells and which are in conflict due to overlapping effects. In this case, the priorities might depend on which cell is more overloaded.

These examples outline that the decision problem of SON coordination depends on both technical knowledge and the operator objectives. Some conflicts need to be resolved in a specific manner due to technical constraints whereas others need to be resolved based on the operator objectives. Furthermore, the SON coordination needs to consider that specific requests, e.g., triggered by the human operator, also need to be coordinated in the sense that they are always preferred compared to all other conflicting requests. Note that conflicts may not solely exists between requests by different SON functions but also between requests by the same SON function which target different cells.

Typical approaches for conflict resolution in SON coordination based on priorities [3GP13] or fixed rules [Ban13][GSB14], mix up the knowledge about technical constraints and operator objectives. This does not allow the operator to control SON coordination with his specific objectives. Furthermore, they do not consider the current system state in order to dynamically adjust the priorities of the SON functions according to the severity of the performance degradation and the functions' predicted effects.

As an example, consider a conflict between two SON functions, SON function 1 and SON function 2, which should be resolved based on the operator objectives. That is, there is no technical constraint on the execution order of both functions. The resulting decision problem the SON coordinator is faced with is visualized in Figure 5.1. Before the coordination decision, the network is supposed to be in an acceptable state regarding the operator objectives with a utility of 0.5. Due to the conflict, the SON functions are expected to interfere with each other in a negative way. Hence, their concurrent execution is expected to decrease the network performance, i.e., the utility of the predicted state is 0.2. This would also be the

result of an uncoordinated SON function execution. If one of the two SON functions is executed alone, then the utility is expected to increases to either 0.6 for SON function 1 or 0.7 for SON function 2. However, depending on the context, the effects in terms of utility gain could also be switched. A fixed priority-based or rule-based coordination decision would not consider the expected utilities of both SON function requests. Instead, it would always reject either SON function 1 or SON function 2. Hence, depending on the context, this decision may not be optimal.

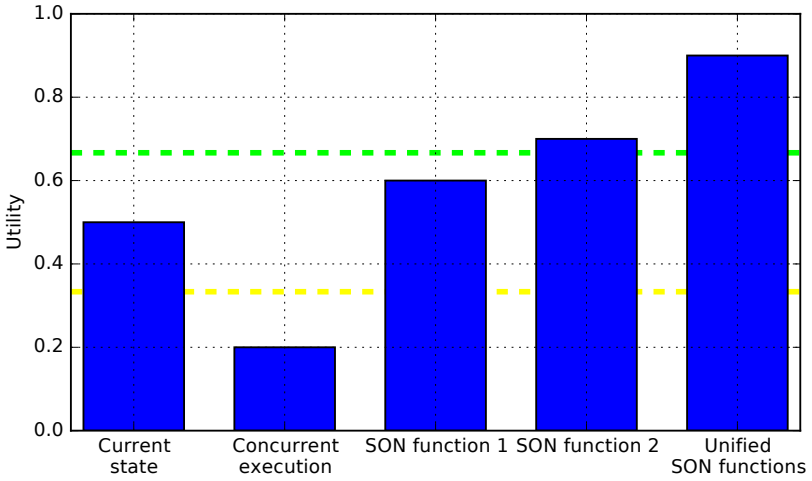


Figure 5.1: Expected utilities of the execution of SON function 1 and SON function 2.

Figure 5.1 also depicts a coordination approach that attempts to "unify both requests into one" [Sch+11, p. 196]. Therefore, SON coordination needs to analyze the requests and merges them into a new, different configuration change. Although this type of coordination promises the highest resulting utility, it requires in depth knowledge of the SON functions and the physical environment. As pointed out in [Ban11], this actually leads to a duplication of functionality, finally making SON functions obsolete. For this reason and the basic assumption that SON functions are black boxes (see Chapter 3.1.2), this coordination option is considered neither reasonable nor feasible by us.

5.2 Goals and Requirements

The idea of objective-driven SON coordination is to estimate and consider the expected utilities of the SON function requests for decision making. The goal is to always accept the optimal set of SON function requests, i.e., accept SON function 2 in Figure 5.1. Therefore, it is necessary to separate the knowledge for technical-driven conflict resolution from the knowledge for objective-driven conflict resolution. This enables the operator to control SON coordination with the operational objectives as required by Objective 3. Based on this, the SON coordinator should acknowledge the set of requests that satisfy the technical constraints and maximizes the expected

system performance with respect to the operator objectives.

The case that the coordination decision depends on the MNO's preferences is especially interesting in the context of ODSO. If two SON functions are in conflict the decision depends on the operator objectives, the current system state, and the predicted effects of the execution of the SON functions. Figure 5.2 depicts a simple decision scenario for SON coordination with respect to operator objectives as defined in the ODSO framework. Consider the SON functions MRO, MLB $\in S$ which request to change the CIO of the same network cell and, thus, are in conflict. The estimated effects of the actions are depicted as arrows, i.e., MRO optimizes the performance of the KPI handover ping-pong rate and MLB the performance of the KPI cell load, respectively. The decision which SON function to execute, strongly depends on the cell's context. If the cell is currently in State A, MLB should be preferred over MRO since it is more important to leave the unacceptable value region of the cell load than to satisfy the optimal threshold of the handover ping-pong rate. However, in system State B, the preference is the other way around since it is more important to improve the performance with respect to the handover ping-pong rate. It is the goal of the ODSO SON coordination to automate such objective-driven decision making.

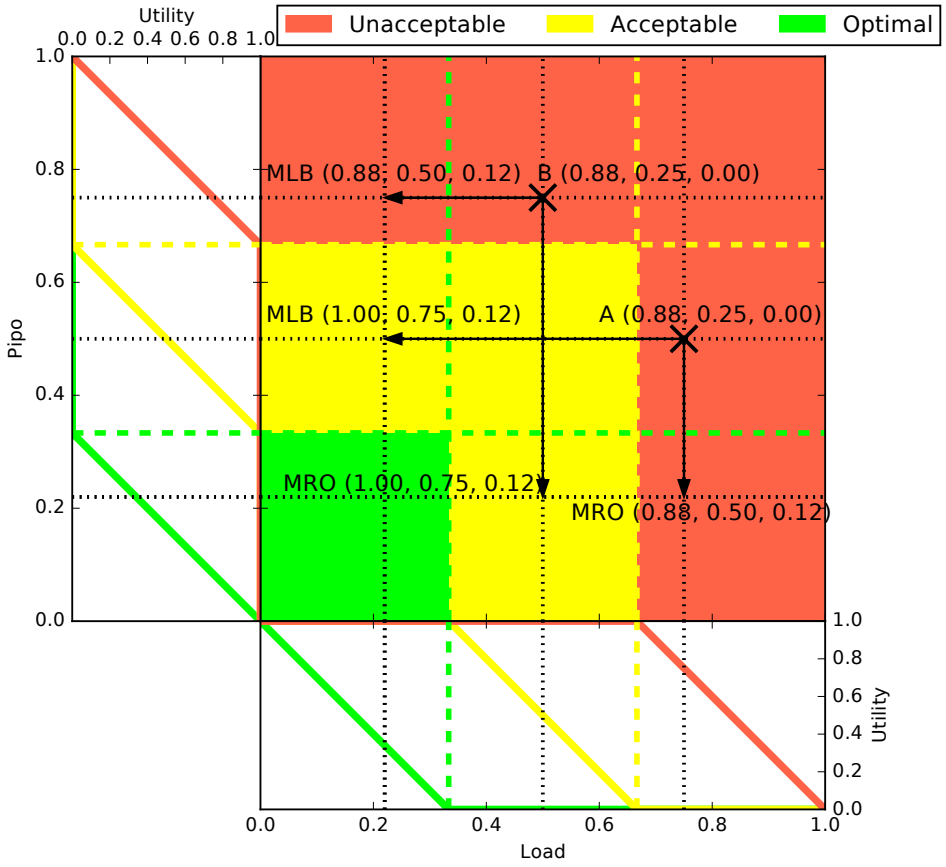


Figure 5.2: Example for the decision problem of the objective-driven conflict resolution between two SON functions in two different system states.

After rejecting one SON function in a granularity period, it may be the case that the very same SON function requests again a change in the next granularity period. Actually, this is the expected behavior since the function was not able to solve the performance problem and, thus, it will likely still be present. Following this thought, it becomes clear that SON coordination performs a kind of scheduling of the SON functions, i.e., it decides which SON function should be executed first, second, and so on. Implicitly, objective-driven SON coordination aims to execute those configuration changes first that improve the satisfaction of operator objectives, i.e., the utility, the most. This leads to a behavior such that it tries to achieve an acceptable or optimal system state, with respect to the operator objectives, as soon as possible.

The example above outlines the decision making for solely two requests. However, in a real network, there are maybe many requests at the same time. This complicates the decision making even more due to the complex conflict relations between the requests. As a result, it is not enough to simply accept the best action for each conflict, but instead it is necessary to accept a conflict-free set of SON function requests that maximizes the overall satisfaction of the operator objectives.

It is important to note a principle assumption of the presented approach: all SON function requests are expected to be correct in the sense that their execution may improve the performance. Specifically, this means that SON coordination should not actively block requests that are not in conflict because the SON function may be erroneous. Such a behavior could be necessary to avoid oscillations between SON functions that change a cell's configuration back and forth. However, the detection and recovery of such behavior is handled by ODSO SON self-healing in Chapter 6.

The detection and resolution of conflicts is a complex task that requires considerable knowledge about the SON. However, ODSO defines an architecture in which SON management provides the other components with information, specifically the SON function effects as described in Chapter 3.3.3. The SON coordination presented in the following will be integrated into this framework and, thus, is enabled to reuse this knowledge in order to reduce the necessary additional information required by the MNO.

5.3 Component Design

The design of objective-driven SON coordination is based on the generic ODSO component design. Its computational process is divided into three parts as depicted in Figure 5.3. The first step, referred to as conflict detection and technical resolution, evaluates the technical knowledge and is related to the action proposal of the generic ODSO component design as described in Chapter 3.4. The second step, referred to as SON function request selection, evaluates the operator objectives and corresponds to the action selection. The third step, referred to as enforcement, distributes the coordination decision to the SON functions as replies to their requests and to SON self-healing as the SON activity (see Chapter 6.3.1.2). Apart from the respective knowledge models, both steps rely on the operational context for computation. The

process is triggered by an event that is raised at the end of the execution phase of the batch coordination. This event contains all requests for action execution by the SON functions collected during the SON function execution phase (see Chapter 3.3.3).

The technical knowledge, which is evaluated by the conflict detection and technical resolution, comprises three parts: the SON function effects provided by the SON management component, a conflict detection model describing the possible conflicts between the SON functions, and a coordination constraint model containing information about the technical constraints for coordination. Based on these models, the first step estimates the possible effects of the requests with respect to the KPIs, determines the conflicts among the SON function requests, and partially resolves them by rejecting requests based on technical constraints.

The remaining conflicts are supposed to be resolved according to the operator objectives which is done by the SON function request selection. Therefore, it analyzes the unresolved conflicts, and acknowledges or declines SON function requests such that the satisfaction of the operator objectives, i.e., the expected utility, is maximized. Similar to the generic ODSO component design, the expected utility is, thereby, computed based on the estimated effects of the SON function requests and the objectives provided as the ODSO objective model. Finally, the enforcement step executes the decision by sending the respective acknowledgment or reject messages to the SON functions that requested an action execution.

SON coordination has to make a decision between two options for a request: acknowledge or reject. Specifically, rescheduling of a request and pre-emption of a running SON function are not considered (see [Ban+11b]). On the one hand, this allows the description to concentrate on the core idea of the objective-driven SON coordination. On the other hand, these actions can be made unnecessary with some assumptions: First, the granularity period is at least as long as the longest completion time (see execution and impact-time presented in Chapter 5.3.1.2) of the SON functions in the SON. Hence, there cannot be any concurrently executed SON functions when the coordination is triggered and, so, no running SON functions needs to be pre-empted. Second, SON functions continue requesting actions after a previous rejection of the same action if the problem they attempt to solve still exists. In this way, the actions do not need to be rescheduled since they will be requested again by the SON function in the next coordination phase.

As outlined in Chapter 3.3.3, ODSO assumes a SON that is synchronized with the granularity period: after the collection of measurements, all SON function are triggered, their requests collected and processed by SON coordination, and finally the accepted changes to the network configuration are deployed. Chapter 5.4 discusses a how the presented concept might be implemented in an asynchronous SON where SON functions may be triggered at any point in time requesting changes of the network configuration.

5.3.1 Conflict Detection and Technical Resolution

Conflict detection and technical resolution, depicted in Figure 5.4, is performed in three steps that are presented in the following. The input to the process is a set

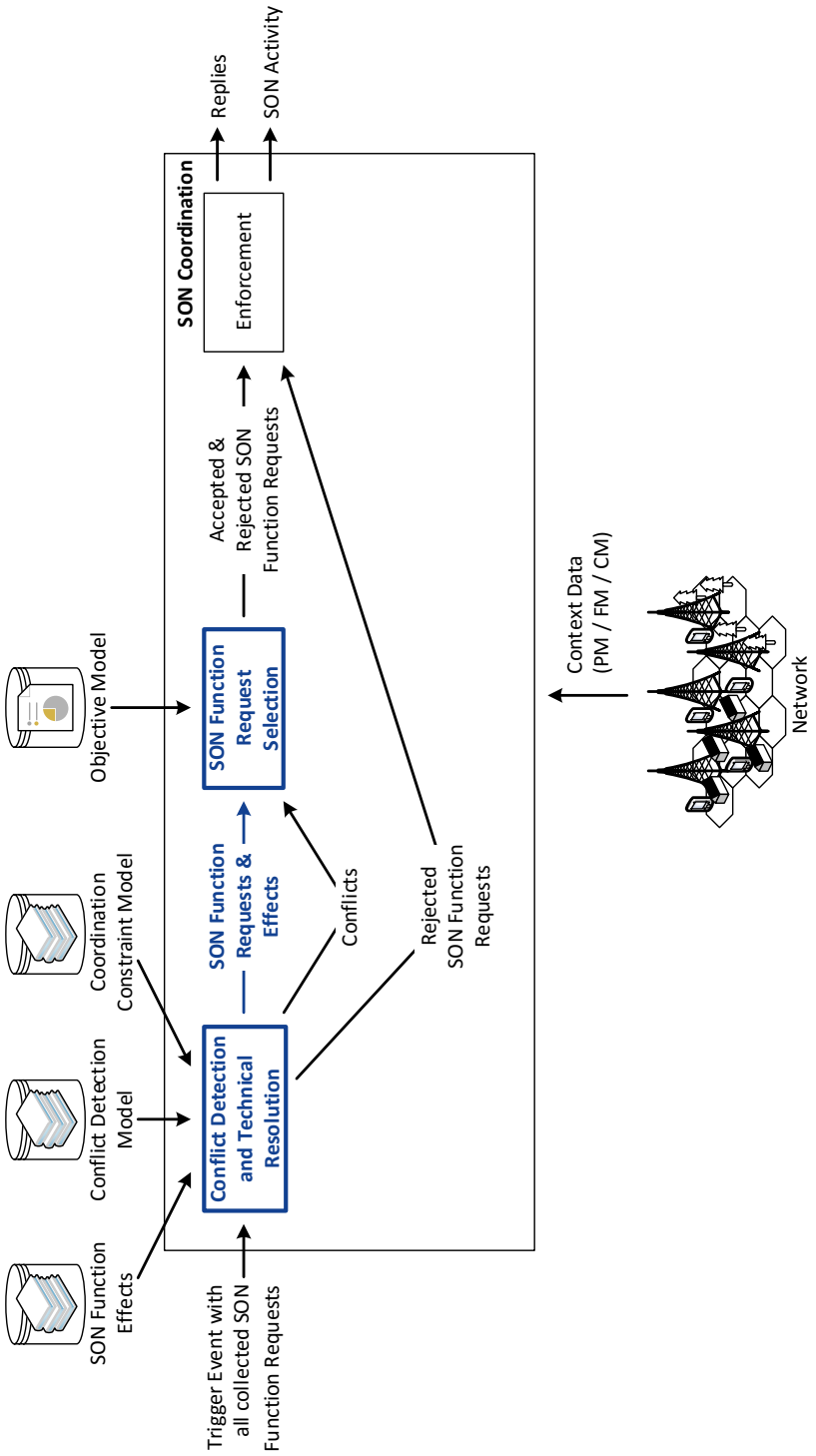


Figure 5.3: Computation process of the SON coordination component. The steps of the generic ODSO component design are drawn in blue.

of all SON function requests that were collected during the execution of the SON function instances in the current granularity period as described in Chapter 3.3.3. From an ODSO point of view, a SON function request can be seen as an action (see Definition 3.5) which SON coordination may select to accept. Consequently, a not selected SON function request is implicitly rejected. Due to this meaning, the symbol a is reused and, in this chapter, always refers to a SON function request.

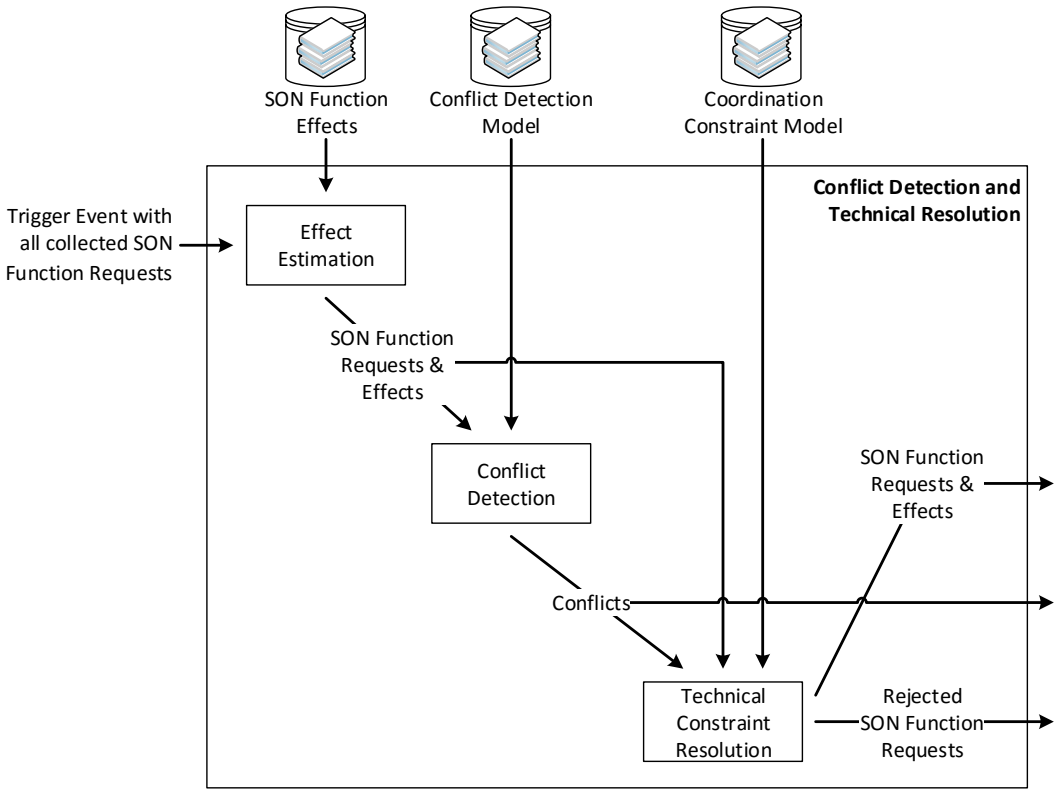


Figure 5.4: Computation process of the conflict detection and technical resolution.

Definition 5.1 (SON function requests)**.** In the context of SON coordination, an action as defined in Definition 3.5 is refined to be a SON function request. A SON function request $a = (s, c)$ with $a \in A$ is a request for the execution of some configuration change by the SON function $s \in S$ running on cell $c \in C$. The set $A = S \times C$ is the domain of all possible SON function requests. $\mathcal{A} \subseteq A$ is the set of SON function requests that are collected during SON function execution phase and need to be coordinated.

Apart from this definition, the request might also comprise additional information like the attempted changes of the network parameters which are, however, not of specific interest in this following.

5.3.1.1 Effect Estimation

The first step of conflict detection and technical resolution is the estimation of the effects of the collected SON function requests. This means, it has to predict the effect on the network cell c of the acceptance of a SON function request $a = (s, c)$, i.e., the one-time execution of the requesting SON function s. The idea of ODSO SON coordination is to reuse the SON function effects SFE provided by the SON management component as described in Chapter 4.3.6 for that.

The SON function effects are derived from the combined effects of the current SON configuration and, thus, represent the long-term expectations for the KPI values of a SON function. However, SON coordination need to make a decision for a single execution of a SON function which produces short-term effects. Although the consideration of the long-term effects may sometimes lead to non-optimal decisions by SON coordination, there are three reasons to still do so:

- The approach allows to reuse the information from the SON function models provided by the vendors of the SON functions. This reduces the efforts for modeling the necessary technical expertise for autonomic SON operations considerably.
- The prediction of the short-term effects of a configuration change of the network is very complex since the peculiarities of the environment are dominating the resulting KPI values. Since it is impossible to observe all properties of the environment, it is often also impossible to accurately predict the resulting network performance. This is one reason why quite a lot of SON functions actually adopt a trial-and-error approach for optimization (see the SON functions described in [HSS11][RH12]). As a result, the long-term effects may be the best effect estimation possible, anyway.
- It may be the case that a SON function first reduces the performance of a cell just to improve it considerably in a consecutive optimization step. In order to handle such workflow-like SON functions, SON coordination needs to look some granularity periods into the future. As a result, SON coordination should actually consider the long-term effects of the SON functions.

The estimated effect of a SON function request $a = (s, c)$ from the SON function effects SFE needs to be combined with the current performance state $\mathbf{x}_c$ of c, i.e., the current values of the KPIs. This is because the execution of a SON function does not necessarily affect all KPIs of c. Specifically, if the execution of s is not expected to have an impact on KPI k, i.e., $\text{SFE}(c, s; k) = \perp$, then the value of k is expected to be the same before and after the execution of s.

Definition 5.2 (Complete effect of SON function request)**.** The estimated complete effect $\mathbf{f}$ of a SON function request $a = (s, c)$ is determined based on the SON function effects $\text{SFE}(c, s)$ (see Definition 4.11) by using the effect merging function $\mathbf{f} = \mu(\text{SFE}(c, s), c, \mathbf{x}, 1.0)$ (see Definition 3.11). As can be seen, we suppose that the probability of the SON function execution changing the KPIs as 1.0, i.e., it is for sure.

The presented estimation approach focuses on reusing knowledge already provided, i.e., the SON function models. In a complicated and dynamic environment, this estimation might show the general direction of optimization by a SON function but may not be accurate regarding the actual expected KPI values. In principle, machine learning seems to be a promising approach to improve the accuracy by learning the network performance of previous coordination decisions. However, this is not trivial since the state space of this problem is enormous even for a simplified formalization [Iac+14a]. In contrast to that, in some cases the effects of a SON function request may be deterministic and computable. For instance, consider an ESM request that attempts to switch off a network cell: the effect on the KPI energy consumption is accurately predictable without consulting the SON function effects. Actually, the direct computation may be more accurate than the probabilistic prediction. However, this example shows nicely the trade-off that an MNO needs to make between accuracy by detailed models and the required human effort for the creation of such models.

Multiple Similar SON Function Requests per Cell A case for which the proposed estimation of the effects of the SON functions may also not be accurate are functions that request several independent configuration changes for one cell, each handling different configuration parameters (cf. cell-focused SON functions in Chapter 3.3.2). Although such non-cell-focused SON functions are not in the primary scope of ODSO, we would like to discuss their handling since they are not uncommon. An example for such a SON function is MRO which optimizes the KPI handover ping-pong rate by adjusting the CIO: the ping-pong rate can be determined for each handover relation to a neighboring cell and, similarly, the CIO is configured for each neighbor relation separately. Consequently, there are implementations of the MRO functionality, e.g., in [Ban13], that optimize each and every neighbor relation of a cell independently. This means, however, that the effect estimation presented above is too optimistic as it estimates that each individual request for a neighbor relation achieves the overall, expected cell effect given by the SON function effects. In such case, it is more realistic that each request achieves a part of the expected cell effect such that the effects of all requests together accumulate to the overall cell effect. Again, the accurate estimation of the effects of multiple SON function requests per cell requires considerable human effort or sophisticated machine learning algorithms for creating the technical models. Alternatively, the partial effect of such a SON function request may be approximated by assuming that each request may result in the overall, expected cell effect given by the SON function effects with the same probability.

Consider that SON function $s \in S$ is triggering n SON function requests $a_1 \dots a_n$ for the same network cell c in a specific granularity period, i.e., $\forall i = 1, \dots, n.\, a_i = (s, c)$. Then each of the n requests may be effective with the uniform probability ρ leading to the effect $\mathbf{f} = \mu\left(\mathrm{SFE}(c, s), c, \mathbf{x}, 1.0\right)$ given by Definition 5.2. Since $n \cdot \rho = 1$, $\rho = 1/n$ and the expected effect of a single request is $1/n \cdot \mathbf{f} = \mu\left(\mathrm{SFE}(c, s), c, \mathbf{x}, 1/n\right)$. This approach can be interpreted as each request may be effective with the probability $1/n$. Chapter 5.3.2.1 shows that this approach leads to a SON function request

utility during SON function request selection that is equivalent to a cell-focused SON function. Notice that for non-cell-focused functions, the set of collected SON function requests $\mathcal{A}$ (see Definition 5.1) needs to be redefined as a multiset containing several formally equal requests (s, c) which may differ in other, non-formalized properties.

5.3.1.2 Conflict Detection

The second step of SON coordination is the determination of conflicts between the SON function requests. A SON function conflict is defined as negative interactions between two executed instances of the same or different SON functions [Ban13]. Whether two SON function instances are in conflict depends on the following characteristics [3GP13][LIA13][Ban13]:

- Interactions between the SON functions and their actions, e.g., the measurement conflict between an RET change by a CCO function and the measured handover performance by an MRO function, since the RET affects the cell size and, so, the handover performance. [Ban13] provides a comprehensive classification of different conflicts.

- Spatial characteristics of the SON functions, often referred to as impact-area. For instance, a RET change by a CCO function may influence the handover performance in neighboring cells. Consequently, conflict detection needs to consider overlapping impact-areas.

- Temporal characteristics of the SON functions, often referred to as execution time and impact-time. This comprises, e.g., the time to enforce a network configuration change and the time for this change to materialize in the PM data. As noted in Chapter 5.3, the impact-time is not relevant for the synchronous SON execution presented here.

As can be seen, conflict detection requires in-depth technical knowledge about the SON functions which is captured in the conflict detection model.

Definition 5.3 (Conflict detection model)**.** The conflict detection model $\mathrm{CDM} : A \times A \times \mathbf{X} \to \mathbb{B}$ defines possible conflicts between SON function requests as a mapping from a pair of SON function requests $(a_1, a_2) \in A \times A$ and the current operational context $\mathbf{x} \in \mathbf{X}$ to a Boolean value $0, 1 \in \mathbb{B}$ indicating whether a_1 and a_2 are in conflict. That is $\mathrm{CDM}(a_1, a_2, \mathbf{x}) = 1$ if and only if a_1 and a_2 are in conflict in $\mathbf{x}$.

Based on this, the set of conflicts for a set of requested SON function executions can be computed.

Definition 5.4 (Conflict set)**.** The result of conflict detection is the set of conflicts for the collected SON function requests $\mathcal{A}$ in the current operational context $\mathbf{x}$ is defined as a set of pairs of conflicting requests:

$$\kappa = \{(a_1, a_2) \mid a_1, a_2 \in \mathcal{A}, a_1 \neq a_2, \mathrm{CDM}(a_1, a_2, \mathbf{x}) = 1\}.$$

The relation defined by $(a_1, a_2) \in \kappa$ is irreflexive, symmetric, but not necessarily transitive.

The objective-driven SON coordination presented in this thesis focuses on conflict resolution. Fortunately, there is numerous related work that describes conflict detection approaches, e.g., [Sch+11][Cia+12][Ban13][Tsa+13][Iac+15]. Hence, we draw on these results. As simple approach outlined in [Ban13] explicitly defines the conflict detection model as an action policy containing rules of the form:

```
IF condition(request_1, request_2, context) THEN true
```

If for a pair of SON function requests any of the rules produces true, a conflict is recorded. For instance, the following rules detects conflicts between CCO and MRO SON function requests if their impact-areas, determined using the operational context, overlap:

```
IF request_1.sonFunction = CCO AND
   request_2.sonFunction = MRO AND
   overlap(request_1.impactArea(context),
           request_2.impactArea(context))
 THEN true
```

In [RT11] and [Tsa+13], two separate groups of authors present a conflict detection approach based on ontological models. Thereby, the possibly affected network configuration parameters by a SON function request as well as the possibly affected KPIs by a configuration change are defined in a semantic model based on ontologies. Using automated reasoning, it is possible to compute possible conflicts by searching for SON function requests that affect the same KPI in one cell. Although this idea promises some degree of automation by using automated reasoning components, the models require considerable manual efforts to be created, especially for an operator who has little insight into the SON functions. However, by reusing the SON function effects, it is possible to implement a similar idea in the ODSO framework. The SON function effects can be used to detect characteristic conflicts [Ban13] which occur if two SON functions adapt different network parameters that affect the same KPIs. Following this argument, two SON function requests are in conflict if they target the same cell and their predicted effects indicate a KPI effect on a common KPI:

$$\mathrm{CDM}(a_1, a_2, \mathbf{x}) = \begin{cases} 1 & \text{if } a_1 = (s_1, c_1), a_2 = (s_2, c_2), c_1 = c_2, s_1 \neq s_2, \\ & \quad \exists k \in K. \mathrm{SFE}(c_1, s_1; k) \neq \bot \wedge \mathrm{SFE}(c_2, s_2; k) \neq \bot \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

5.3.1.3 Technical Constraint Resolution

The third step of SON coordination is imposing technical constraints for conflict resolution provided by the coordination constraint model. Such constraints can be necessary for two reasons:

- There are technical reasons for resolving conflicts in a specific manner, e.g., because they need to be executed in a specific workflow [Ban+11b]. For instance, a self-healing function should be executed before an optimization function since the former restores the failure-free operation of a network cell that is a prerequisite for optimization (see Chapter 5.1 and 6.3.4.2).

- The operator must be able to influence coordination, e.g., in order to enforce the execution of a manually triggered SON functions or to block unwanted SON functions.

Violations of the constraints are resolved by rejecting some SON function requests. Therefore, the constraints must be prioritized in order to accurately control the decision making. Consider the case that a self-healing COC request and a CCO function request are in conflict (see Chapter 5.1), and the operator requested the execution of the CCO request. Hence, there are two technical constraints for their coordination: first the COC request should be executed and the CCO rejected due to general technical reasons and, second, the CCO should be executed and every conflicting request should be rejected due to the manual intervention by the operator. This shows that, of course, violated operator constraints need to be resolved before general technical constraints.

A simple and well-known way to implement such a prioritized decision making is by using a mature action rule system like JBoss Drools [The13]. These allow assigning a "salience value" [The13, Ch. 6.2.2.2] to the if-then action rules which perfectly resembles the desired priorities. So, coming back to the example presented before, the generic technical constraint that a COC request should be preferred to a CCO request may be expressed as

```
IF request_1.sonFunction = COC AND
   request_2.sonFunction = CCO AND
   (request_1, request_2) in conflictSet
  THEN reject(request_2) PRIORITY 2
```

For the sake of the example, this rule has a priority of 2 which is not the highest priority. Such conflict rules must explicitly define transitivity, i.e., if a COC request is preferred to a CCO request which itself may be preferred to an MRO request then there must be a rule for checking COC against the conflict with CCO, COC against MRO, and CCO against MRO. The operator enforced execution of the CCO function in the example can be expressed as

```
IF operator_requested(request_1) = true AND
   operator_requested(request_2) = false AND
   (request_1, request_2) in conflictSet
  THEN reject(request_2) PRIORITY 1
```

The rule analyzes a SON function request request_2 if it is in conflict with the operator-enforced request request_1 and, consequently, rejects it. As a result, this rule removes all SON function request that are in conflict with operator-enforced requests, thereby, ensuring their execution. This rule assumes that a function operator_requested( ) provides an indication that a request was triggered by the operator. Note that the rule has a priority of 1 and, hence, is evaluated before the generic technical constraints.

It is important to notice that the technical constraint resolution is performed on all SON function requests and not solely the requests that are in conflict. The reason is that the operator, or even SON self-healing (see Chapter 6.3.4.2), may need to block the execution of a SON function even if it is not in conflict with any other requests.

Definition 5.5 (Technical constraint resolution)**.** Independent of the actual implementation, the technical constraint resolution computes a set of rejected SON function requests $\overline{\mathcal{A}}^{\mathrm{TCR}} \subseteq \mathcal{A}$ based on the coordination constraint model and the collected SON function request $\mathcal{A}$, and a set of technically accepted SON function request $\mathcal{A}^{\mathrm{TCR}} = \mathcal{A} \setminus \overline{\mathcal{A}}^{\mathrm{TCR}}$.

Definition 5.6 (SON function requests and effects)**.** In the context of SON coordination, the action-effect set as defined in Definition 3.10 is refined to the set of SON function requests and effects. Based on the technical constraint resolution, the set of SON function requests and effects $\mathrm{AF} \in A \times \mathbf{F}$ contains pairs $(a, \mathbf{f})$ for all technically accepted SON function request $a \in \mathcal{A}^{\mathrm{TCR}}$ and their respective complete effects $\mathbf{f} \in \mathbf{F}$ as determined by effect estimation.

5.3.2 SON Function Request Selection

After conflict detection and technical resolution, the SON function request selection step resolves the remaining conflicts between the SON function requests based on the operator objectives. Therefore, it first calculates the expected utility for the execution of each request based on its expected effects. Second, it resolves the conflicts by determining a conflict-free subset of all SON function requests which maximizes the sum of the expected utilities.

5.3.2.1 Expected Utility for SON Function Requests

Single SON Function Request The calculation of the expected utility for each SON function request is performed based on the expected effects provided by the conflict detection and technical resolution, the objective model, and the operational context. In principle, a conflict between two SON function requests a_1 and a_2 is based on the preference relation $\succsim$, i.e., a_1 should be selected if and only if $a_1 \succsim a_2$. Definition 3.20 presented that this decision has to be based on the expected utility vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ of the complete effects $\mathbf{f}_1$ and $\mathbf{f}_2$ of both actions, i.e., $a_1 \succsim a_2 \iff \mathbf{u}_1 \geq_D \mathbf{u}_2$.

If only SON function requests for the same cell could be in conflict, this comparison would be correct since the effects on one network cell are commensurable. However, since conflicting requests a_1, a_2 with the effects $\mathbf{f}_1, \mathbf{f}_2$ and the utilities $\mathbf{u}_1, \mathbf{u}_2$ can also target different network cells, i.e., $\text{proj}_C(a_1) \neq \text{proj}_C(a_1)$, $\mathbf{u}_1 \geq_D \mathbf{u}_2$ does consider incomparable effects $\mathbf{f}_1$ and $\mathbf{f}_2$ on different network cells. The Figure 5.5 outlines this with an example. Consider Cell A and Cell B with the marked system states before the execution of any SON function request. Obviously, Cell B is performing worse than Cell A. For both cells, the SON function MRO requested a configuration change but these requests are in conflict. Considering solely the final expected performance and utility, the request on Cell A should be preferred. However, this is obviously not the behavior expected by an MNO. Instead, the request on Cell B should be accepted since the initial state is much worse and, thus, the overall improvement of the performance and the satisfaction of the objectives is higher.

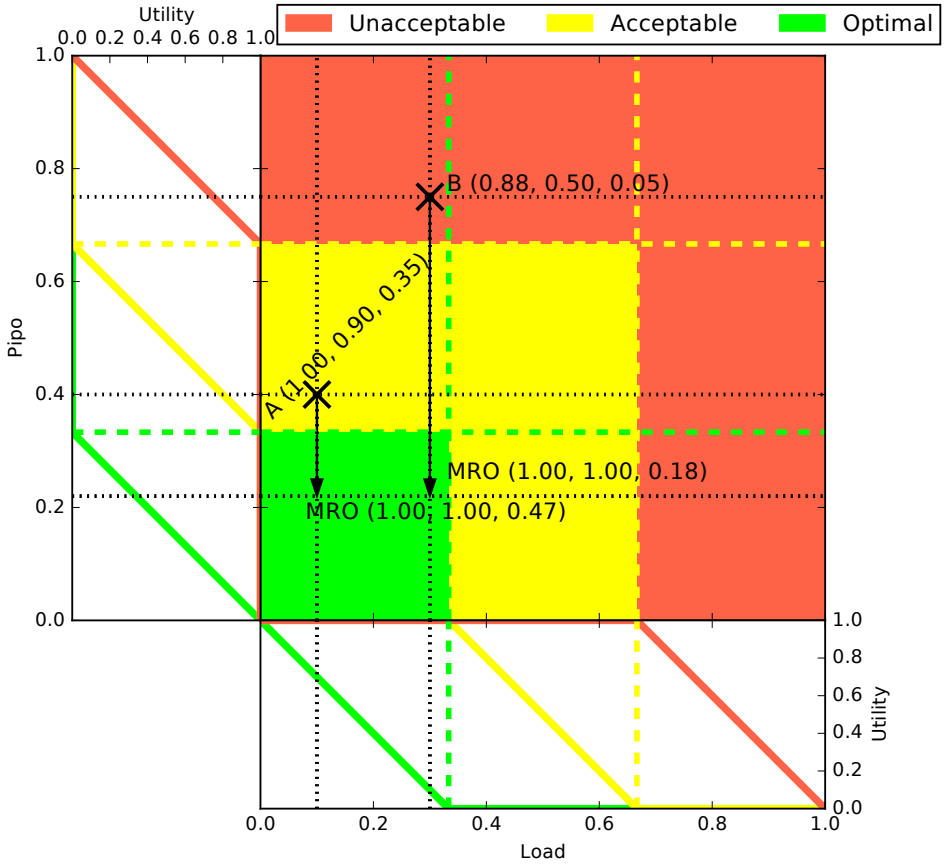


Figure 5.5: Example outlining the importance to consider the utility improvement for the selection of the SON function requests.

The preference relation $a_1 \succsim a_2$ is actually based on the state of the whole network after the execution of either request. Let $\mathbf{u}_1$ be the expected utility vector for the effect of a_1 on cell c_1, $\mathbf{u}_2$ be the utility vector for the effect of a_2 on cell c_2, C be the set of all network cells, and $\mathbf{u}_{\mathbf{x}_c}$ be the utility vector of the current performance of the network cell c. Based on this, the preference between a_1 and a_2 can be calculated

using the expected utility of all cells of the network after the execution of either SON function request:

$$a_1 \gtrsim a_2 \iff \mathbf{u}_1 + \sum_{c \in C, c \neq c_1} \mathbf{u}_{\mathbf{x}_c} \geq_D \mathbf{u}_2 + \sum_{c \in C, c \neq c_2} \mathbf{u}_{\mathbf{x}_c}. \tag{5.2}$$

This calculation shows that the selection of a SON function request is a MAUT decision problem over attributes being the satisfaction of the MNO preferences for each cell. Hence, Equation 5.2 requires some assumptions in order to be valid: First, the shown additive aggregation of the utility per cell requires that the preferences regarding the satisfaction of the objectives for each cell are additive independent. In parallel to the discussion for KPI preferences in Chapter 3.4.3.3, we see this as a reasonable assumption for the cells that simplifies to the preference model considerably. Second, the satisfaction of each cell's objectives is equally important for all cells, i.e., each cell utility has the same weight. This assumption simplifies the following presentation of the design. However, if the MNO sees some network cells more important than others then Equation 5.2 could be extended to a weighted sum of the cell utilities similarly to the weighted KPI utilities in Equation 3.6.

Note that Equation 5.2 shows a vector addition over the utility vectors. In order to keep the semantics of the priorities as defined in Definition 3.19, these operations must be performed element-wise.

Definition 5.7 (Utility vector addition and subtraction)**.** Two utility vectors $\mathbf{u}' = (u'_1, \ldots, u'_{N_D})$ and $\mathbf{u}'' = (u''_1, \ldots, u''_{N_D})$ are added and subtracted element-wise, i.e., $\mathbf{u}' + \mathbf{u}'' = (u'_1 + u''_1, \ldots, u'_{N_D} + u''_{N_D})$ and $\mathbf{u}' - \mathbf{u}'' = (u'_1 - u''_1, \ldots, u'_{N_D} - u''_{N_D})$.

Using some mathematical transformations, one can simplify Equation 5.2 for the overall network state to the following

$$\begin{aligned} &\mathbf{u}_1 + \sum_{c \in C, c \neq c_1} \mathbf{u}_{\mathbf{x}_c} \geq_D \mathbf{u}_2 + \sum_{c \in C, c \neq c_2} \mathbf{u}_{\mathbf{x}_c} \\ \iff\ &\mathbf{u}_1 + \mathbf{u}_{\mathbf{x}_{c_2}} \geq_D \mathbf{u}_2 + \mathbf{u}_{\mathbf{x}_{c_1}} \\ \iff\ &\mathbf{u}_1 - \mathbf{u}_{\mathbf{x}_{c_1}} \geq_D \mathbf{u}_2 - \mathbf{u}_{\mathbf{x}_{c_2}}. \end{aligned} \tag{5.3}$$

This result can be interpreted such that the preference of a SON function request a on cell c is based on the difference between the cell performance utility before its execution $\mathbf{u}_{\mathbf{x}_c}$ and the expected utility after its execution $\mathbf{u}_1$. In other words, the preference depends on the expected improvement of the utility.

Definition 5.8 (Expected utility improvement of a SON function request)**.** The expected utility improvement $\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}_{\mathbf{x}_c}$ of a SON function request a with the effect $\mathbf{f}$ on cell $c = \text{proj}_C(a)$ is the difference of the expected utility vector $\mathbf{u}$ of $\mathbf{f}$ and the utility of the current performance $\mathbf{u}_{\mathbf{x}_c}$ of c. Thereby, $\mathbf{u}_{\mathbf{x}_c} = \mathrm{E}\left[o_{*,*}(\mathbf{f}_c)\right]$ is the utility vector for the effect $\mathbf{f}_c \in \mathbf{F}$ such that $\mathbf{f}_c(k) = \delta_{\mathbf{x}(c,k)}$ is representing the current performance of c as a tuple of Dirac distributions (see Definition 3.11) for all KPIs $k \in K$.

Definition 5.9 (Preference of SON function requests)**.** A SON function request a_1 with the expected utility improvement $\Delta\mathbf{u}_1$ is preferred to a request a_2 with $\Delta\mathbf{u}_1$ if and only if $\Delta\mathbf{u}_1 \geq_D \Delta\mathbf{u}_2$.

The fact that $\Delta\mathbf{u}$ is representing the expected improvement of the network performance can also be used to reject actions with negative expected effects. That is, a SON function request a with $\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}_{\mathbf{x}_c}$ is rejected if $(0.0, 0.0, 0.0) \geq_D \Delta\mathbf{u}$ even if a is not in conflict with any other request. This is motivated by the fact that the current network performance is preferred to the expected network performance after the execution of the SON function request, $\mathbf{u}_{\mathbf{x}_c} \geq_D \mathbf{u}$. This decision logic has one main advantage and one main disadvantage:

Advantage: SON functions optimize one specific or a small set of KPIs. Thereby, they neglect other KPIs in order to be computationally efficient. Hence, the execution of SON function s_1 might improve the performance of one optimized KPI k_1 while reducing the performance of other another KPIs k_2. The decision which performance state is preferred depends on the concrete operator objectives and can only be evaluated by the SON coordinator. Continuing this thought, the reduced performance of k_2 might trigger another SON function s_2 whose execution eventually reduces the performance of k_1 again. It can be seen that this scenario can lead to an oscillating execution between s_1 and s_2. Such a situation can happen between MRO, which improves the handover performance while potentially worsening the cell load, and MLB, which improves the cell load while potentially worsening the handover performance, as shown in Figure 5.6. If the expected utilities of both SON function requests would be compared with the current system performance, then one state, either the good handover performance or load, would be preferred and the request by the SON function that worsens this situation would be rejected.

Disadvantage: The effects of the execution of a SON function are only estimations and depend on complex interrelations between the configuration of the cell and the environment. This complexity requires sophisticated algorithms within the SON functions to determine when and how to request a change in the network configuration. The SON function models can provide only a rough estimation of the effects of the execution of a SON function which is much less accurate than the algorithms in the SON function. This is because these algorithms should actually be kept secret as the intellectual property of the vendor. Furthermore, the effect estimation, as presented here, does not consider the actually requested changes to the network configuration. As a result, rejecting conflict-free requests based on these rough effect estimations can block a lot of SON functions that might actually improve the network performance. Of course, the effect estimation in the SON coordination can be made more accurate. However, this would eventually lead to the duplication of the SON function algorithms in the SON coordination component making the coordination very complex and the SON function obsolete [Ban11].

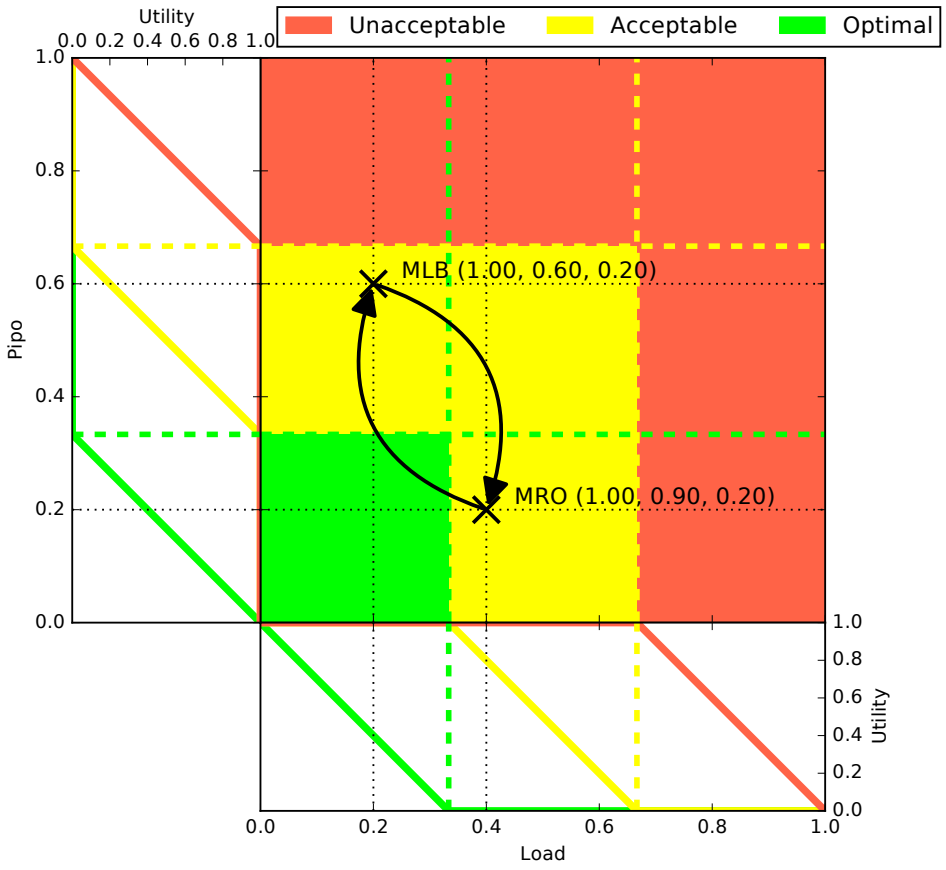


Figure 5.6: Example of an oscillation between MRO and MLB.

As already indicated in Chapter 5.2, we opt for following the second argument and, hence, would not advice to reject non-conflicting actions that may reduce the network performance. Instead, the ODSO framework allows to detect and handle such problems by monitoring the activity of the SON functions using SON self-healing (see Chapter 6). However, this assessment may change in the future if more accurate effect models, potentially created using machine learning, are available (see Chapter 5.3.1.1).

Multiple Similar SON Function Requests per Cell Chapter 5.3.1.1 discussed the issues that arise from SON functions that trigger several independent SON function requests for the same network cell. From the perspective of SON function request selection, the decision making should be independent of this issue. That means, an operator is indifferent regarding the operational preferences between selecting the single SON function request a' of a cell-focused SON function $s' \in S$ on cell $c \in C$ and selecting all n triggered requests $a_1, \ldots, a_n$ of a non-cell-focused SON function $s \in S$ on the same cell c with $s' \neq s$, if and only if the expected cell effects, $\mathrm{SFE}(c, s')$ and $\mathrm{SFE}(c, s)$, are equal. In other words, s' and s are similar SON functions that solely differ in the fact that s' puts all desired changes into a single request and s requests every change independently. The indifference can be formally expressed as $a' \sim \{a_1, \ldots, a_n\}$. Consequently, $\Delta\mathbf{u}' = \sum_{i=1}^{n} \Delta\mathbf{u}_i$ must hold, with $\Delta\mathbf{u}'$ being the utility improvement by a' and $\Delta\mathbf{u}_i$ being the utility improvement of a_i. In the following, we show that this holds for the proposed estimation of the expected effects presented in Chapter 5.3.1.1.

With no loss in generality, we solely focus on the priority $d \in D$. Hence, for each d it must hold that $\Delta\mathbf{u}'_d = \sum_{i=1}^{n} \Delta\mathbf{u}_{i,d}$ with $\Delta\mathbf{u}'_d = \mathrm{proj}_d(\Delta\mathbf{u}')$ and $\Delta\mathbf{u}_{i,d} = \mathrm{proj}_d(\Delta\mathbf{u}_i)$. The utility for the priority can be converted to:

$$
\begin{aligned}
\Delta\mathbf{u}'_d &= \mathbf{u}'_d - \mathbf{u}_{\mathbf{x}_{c,d}} && \text{(Definition 5.8)} \\
&= \left(\sum_{k \in K} w_k \cdot \int_{\mathrm{Dom}(k)} f'_k(v) \cdot o_{k,d}(v)\, \mathrm{d}v \right) - \\
&\quad \left(\sum_{k \in K} w_k \cdot \int_{\mathrm{Dom}(k)} \delta_{\mathbf{x}(c,k)}(v) \cdot o_{k,d}(v)\, \mathrm{d}v \right) && \text{(Definition 5.8, Definition 3.17)} \\
&= \sum_{k \in K} w_k \cdot \int_{\mathrm{Dom}(k)} f'_k(v) \cdot o_{k,d}(v) - \delta_{\mathbf{x}(c,k)}(v) \cdot o_{k,d}(v)\, \mathrm{d}v \\
&= \sum_{k \in K} w_k \cdot \int_{\mathrm{Dom}(k)} (f'_k(v) - \delta_{\mathbf{x}(c,k)}(v)) \cdot o_{k,d}(v)\, \mathrm{d}v, && (5.4)
\end{aligned}
$$

with f'_k denoting the expected KPI effect of a' on KPI k as determined by effect

estimation. With similar conversions, we get

$$\sum_{i=1}^{n} \Delta \mathbf{u}_{i,d} = \sum_{i=1}^{n} \sum_{k \in K} w_k \cdot \int_{\mathrm{Dom}(k)} (f_{i,k}(v) - \delta_{\mathbf{x}(c,k)}(v)) \cdot o_{k,d}(v)\, \mathrm{d}v$$
$$= \sum_{k \in K} w_k \cdot \int_{\mathrm{Dom}(k)} (\sum_{i=1}^{n} f_{i,k}(v) - \delta_{\mathbf{x}(c,k)}(v)) \cdot o_{k,d}(v)\, \mathrm{d}v, \tag{5.5}$$

with $f_{i,k}$ denoting the expected KPI effect of a_i on KPI k as determined by effect estimation.

Based on Equation 5.4 and Equation 5.5, it is sufficient to show that for each $k \in K$,

$$f'_k(v) - \delta_{\mathbf{x}(c,k)}(v) = \sum_{i=1}^{n} f_{i,k}(v) - \delta_{\mathbf{x}(c,k)}(v). \tag{5.6}$$

Note that the SON function effects for both SON functions, $\mathrm{SFE}(c, s')$ and $\mathrm{SFE}(c, s)$, are equal. Regarding the estimated effects by the effect estimation step described in Chapter 5.3.1.1, two cases have to be distinguished:

- If $\mathrm{SFE}(c, s; k) = \mathrm{SFE}(c, s'; k) = \perp$, i.e., the SON functions do not affect k, then $f'_k(v) = \mu_k(\perp, c, \mathbf{x}, 1.0)(v) = \delta_{\mathbf{x}(c,k)}(v)$ and $f_{i,k}(v) = \mu_k(\perp, c, \mathbf{x}, 1/n)(v) = \delta_{\mathbf{x}(c,k)}(v)$ according to Definition 3.11. Consequently, Equation 5.6 becomes

$$\delta_{\mathbf{x}(c,k)}(v) - \delta_{\mathbf{x}(c,k)}(v) = \sum_{i=1}^{n} \delta_{\mathbf{x}(c,k)}(v) - \delta_{\mathbf{x}(c,k)}(v)$$
$$0 = \sum_{i=1}^{n} 0. \tag{5.7}$$

- If $\mathrm{SFE}(c, s; k) = \mathrm{SFE}(c, s'; k) = f_k$, i.e., the SON functions affect k according to the KPI effect $f_k \neq \perp$, then $f'_k(v) = \mu_k(f_k, c, \mathbf{x}, 1.0)(v) = 1 \cdot f_k(v) + (1 - 1) \cdot \delta_{\mathbf{x}(c,k)}(v)$ and $f_{i,k}(v) = \mu_k(f_k, c, \mathbf{x}, 1/n)(v) = 1/n \cdot f_k(v) + (1 - 1/n) \cdot \delta_{\mathbf{x}(c,k)}(v)$ according to Definition 3.11. Consequently, Equation 5.6 becomes

$$f_k(v) - \delta_{\mathbf{x}(c,k)}(v) = \sum_{i=1}^{n} \frac{1}{n} \cdot f_k(v) + \left(1 - \frac{1}{n}\right) \cdot \delta_{\mathbf{x}(c,k)}(v) - \delta_{\mathbf{x}(c,k)}(v)$$
$$= \sum_{i=1}^{n} \frac{1}{n} \cdot f_k(v) + \frac{1}{n} \cdot \delta_{\mathbf{x}(c,k)}(v)$$
$$= f_k(v) - \delta_{\mathbf{x}(c,k)}(v). \tag{5.8}$$

As a result of Equation 5.7 and Equation 5.8, we have shown that $\Delta \mathbf{u}' = \sum_{i=1}^{n} \Delta \mathbf{u}_i$ holds for the proposed estimation of the expected effects presented in Chapter 5.3.1.1.

Set of SON Function Requests The goal of the conflict resolution based on the operator objectives is to resolve the remaining conflicts between SON function requests $\mathcal{A}^{\mathrm{TCR}}$, which are accepted by conflict detection and technical resolution, by selecting or rejecting requests such that

- the set of finally selected requests $\mathcal{A}^{\mathrm{OM}} \subseteq \mathcal{A}^{\mathrm{TCR}}$ is conflict-free and
- the set is preferred over all other conflict-free subsets of $R \subseteq \mathcal{A}^{\mathrm{TCR}}$, i.e., $\forall R \subseteq \mathcal{A}^{\mathrm{TCR}}, R$ is conflict-free. $\mathcal{A}^{\mathrm{OM}} \succsim R$.

Therefore, the relation $\succsim$ needs to be defined for sets of actions. Based on Equation 5.2, one can see a set of SON function requests as a decision option where each request changes the utility of a cell. The utility improvement vectors provide an abstraction of the expected utility of an SON function request on a particular network cell from the performance of the rest of the network cells. In other words, the preference of a SON function request a_1 over a request a_2 is completely independent of the performance of the network cells $c \in C \setminus \{\mathrm{proj}_C(a_1), \mathrm{proj}_C(a_2)\}$ that are not affected by a_1 or a_2. This particularly also holds if a non-conflicting SON function request a_3 is executed in parallel with $\mathrm{proj}_C(a_3) \neq \mathrm{proj}_C(a_1)$ and $\mathrm{proj}_C(a_3) \neq \mathrm{proj}_C(a_2)$. Based on this, the preference of a set of accepted SON function requests can be based on MAUT with independent attributes being the accepted requests. Again, we assume additive independence between the accepted the SON function requests (see Chapter 5.3.2.1) allowing to define a utility measure over the sum of the utilities of the accepted SON function requests. Hence, the preference of a set of accepted SON function requests R_1 to another set R_2 is based on the sum of the utility improvements of the requests in R_1 and R_2 as

$$R_1 \succsim R_2 \iff \sum_{a \in R_1} \Delta\mathbf{u}_a \geq_D \sum_{a \in R_2} \Delta\mathbf{u}_a \tag{5.9}$$

with $\Delta\mathbf{u}_a$ being the expected utility improvement by SON function request a.

Unfortunately, this simple definition would not accept SON function requests with a negative expected utility improvement $\Delta\mathbf{u}$, i.e., $(0.0, 0.0, 0.0) \geq_D \Delta\mathbf{u}$. For example, consider two conflicting request a_1 and a_2 with $\Delta\mathbf{u}_1 = (0.0, 0.0, -0.5)$ and $\Delta\mathbf{u}_2 = (0.0, 0.0, -0.4)$ respectively. In this case, the outlined approach would select no action since, the empty set of accepted SON function requests $\emptyset$ has a higher sum of utility improvements $\sum_{a \in \emptyset} \Delta\mathbf{u}_a = (0.0, 0.0, 0.0)$ as both other options $\{a_1\}$ or $\{a_2\}$. Note this problem also occurs if there are additional requests with positive utility improvements involved. In order to overcome this problem in a generic way, it is necessary to uniformly lift the utility improvements by a constant such that no SON function request has a utility improvement $(0.0, 0.0, 0.0) \geq_D \Delta\mathbf{u}$. This is a positive linear transformation regarding the utility (see Chapter 2.3.2) and, hence, does not change the resulting preferences regarding the requests. Note that a lift is only necessary if at least one utility improvement is negative.

Definition 5.10 (Lifted expected utility improvement of a SON function request)**.** Given the set of technically accepted SON function requests $\mathcal{A}^{\text{TCR}}$, let $\Delta\mathbf{u}_a$ denote the expected utility improvement for a request $a \in \mathcal{A}^{\text{TCR}}$. Let $\Delta U = \{\Delta\mathbf{u}_a \mid a \in \mathcal{A}^{\text{TCR}}, (0.0, 0.0, 0.0) \geq_D \Delta\mathbf{u}_a\}$ be the set of all negative expected utility improvements, and let $\Delta\mathbf{u}_{\text{min}}$ be the smallest, negative expected utility improvement by all requests, i.e., $\Delta\mathbf{u}_{\text{min}} = \min_{\Delta\mathbf{u}\in\Delta U} \Delta\mathbf{u}$. The lifted utility improvement $\widehat{\Delta\mathbf{u}}$ of a SON function request with the utility improvement $\Delta\mathbf{u}$ is calculated as

$$\widehat{\Delta\mathbf{u}} = \begin{cases} \Delta\mathbf{u} + \Delta\mathbf{u}_{\text{min}} + (0.0, 0.0, 0.1) & \text{if } \Delta U \neq \emptyset \\ \Delta\mathbf{u} & \text{otherwise.} \end{cases}$$

The additional, arbitrary $(0.0, 0.0, 0.1)$ ensures that there is no $\widehat{\Delta\mathbf{u}} = (0.0, 0.0, 0.0)$.

Based on these discussions, it is possible to define a preference relation over sets of accepted SON function requests.

Definition 5.11 (Preference over sets of accepted SON function requests)**.** The preference of a set of accepted SON function requests R_1 to another set R_2 is based on the sum of the lifted expected utility improvements of the requests in R_1 and R_2 as

$$R_1 \gtrsim R_2 \iff \sum_{a\in R_1} \widehat{\Delta\mathbf{u}}_a \geq_D \sum_{a\in R_2} \widehat{\Delta\mathbf{u}}_a$$

with $\widehat{\Delta\mathbf{u}}_a$ being the lifted expected utility improvement by SON function request a.

The summation of the lifted expected utility improvements assumes that the requests within a set R are independent of each other. Specifically, there must be no two requests which have an effect on the same KPI in the same network cell. Otherwise, the improvement of that KPI would be counted twice and, so, falsify the calculated overall expected utility vector. Consider an MLB function that reduces the load in a cell from 90% to 70% and another load optimization function like traffic steering [Las+11] that reduces the load to 60%. It is obvious that the execution of both functions will not necessary lead to a load of 40%. Hence, the expected utilities of both requests cannot be summed up. Instead, it would be necessary to combine their effects similarly to the approach that has been presented in Chapter 4.3.3.2 for the SON management component, and calculate the utility based on the combined effect. However, this should, in principle, not happen for the sets of SON function request considered in SON coordination since they must be conflict-free. As outlined in Chapter 5.3.1.2, this means that two requests affecting the same KPI for the same network cell are typically seen as in conflict and, hence, will not appear in one set. Notice that this discussion does not apply to the previously discussed case of SON functions that trigger several independent requests for the same network cell: the introduced handling ensures that the adapted effects of requests for the same cell can be summed up as described in Chapter 5.3.2.1.

5.3.2.2 Objective-Driven Conflict Resolution

A simple heuristic to resolve the conflicts in the set of technically accepted SON function requests $\mathcal{A}^{\mathrm{TCR}}$ is to evaluate each single conflict and reject the SON function request with the smaller lifted expected utility improvement. That is, a conflict between two SON function requests $a_1, a_2 \in \mathcal{A}^{\mathrm{TCR}}$ is resolved by selecting the request that is preferred and rejecting the other request, i.e., select a_1 if and only if $a_1 \succsim a_2$. However, this heuristic is not optimal.

Since a SON function request is not necessary in conflict with only one other request, the conflicts can form complex non-transitive dependencies. This can be visualized as a graph with the SON function requests being the nodes and the conflicts between the requests being the edges. Figure 5.7 depicts an exemplary graph for three SON function requests a_1, a_2, and a_3 with conflicts (a_1, a_2) and (a_1, a_3). The conflicts can be resolved in five ways: the conflict-free sets of selected SON functions are $\{\}$, $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, and $\{a_2, a_3\}$. Thereby, only $\{a_1\}$ and $\{a_2, a_3\}$ are interesting since they are maximal in the sense that no request can be added without producing a conflict. Comparing these two sets, $\{a_2, a_3\}$ is preferred, i.e., $\{a_2, a_3\} \succsim \{a_1\}$, since

$$\begin{aligned} &\widehat{\Delta \mathbf{u}}_{a_2} + \widehat{\Delta \mathbf{u}}_{a_3} \geq_D \widehat{\Delta \mathbf{u}}_{a_1} \\ &\iff (0.0, 0.4, 0.5) + (0.0, 0.4, 0.9) \geq_D (0.0, 0.6, 0.7) \\ &\iff (0.0, 0.8, 1.4) \geq_D (0.0, 0.6, 0.7). \end{aligned} \tag{5.10}$$

However, the comparison of single SON function requests would lead to the selection of a_1 since it is preferred to each single other request, i.e., $\widehat{\Delta \mathbf{u}}_{a_1} \geq_D \widehat{\Delta \mathbf{u}}_{a_2}$ and $\widehat{\Delta \mathbf{u}}_{a_1} \geq_D \widehat{\Delta \mathbf{u}}_{a_3}$. Note that, since the expected utility improvement for the highest priority region, i.e., the unacceptable region, is equally 0.0, the decision is based on the comparison of the expected utility improvements of the acceptable region, i.e., $0.8 \geq 0.6$.

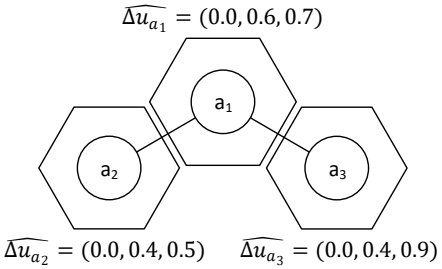


Figure 5.7: Exemplary graph of the conflicts between the SON function requests a_1, a_2, and a_3.

The task of objective-driven conflict resolution is in essence a constrained multi-criteria decision problem: select a set of SON function requests (= decision) that is conflict-free (= constraints) and maximizes the overall utility (= criteria). Such

problems are common in the field of operations research and Integer Programming (IP) has been developed as an efficient solution approach [Win04, Ch. 9]. Additionally, there are numerous off-the-shelf IP solvers available. IP is a declarative solution paradigm which represents the problem as a mathematical optimization model that comprises a set of decision variables, constraints over the variables, and a single objective function over the variables that needs to be maximized or minimized. In the ODSO concept, however, there are three utility functions that need to be maximized in a lexicographical order according to the priorities (see Definition 3.19). The Goal Programming (GP) approach extends IP to handle multiple utility functions in order to make a globally best decision [JT10]. The key extension of GP is the combination of the different utility functions. Among the numerous approaches, preemptive or lexicographic GP aims to optimize the utility functions according to a lexicographic order [HL05, "Supplement to Chapter 7"]. Therefore, it fits naturally for SON function request selections.

Definition 5.12 (Goal Programming problem)**.** SON function request selection is modeled as the following lexicographic Goal Programming (GP) problem regarding the priority domain D:

$$\operatorname{lex}_D \max \mathbf{u} = \sum_{a \in \mathcal{A}^{\mathrm{TCR}}} \mathrm{var}_a \cdot \widehat{\Delta \mathbf{u}}_a$$

subject to

$$\forall (a_1, a_2) \in \kappa.\, \mathrm{var}_{a_1} + \mathrm{var}_{a_2} < 2$$

and

$$\forall a \in \mathcal{A}^{\mathrm{TCR}}.\, \mathrm{var}_a \in \mathbb{B}.$$

The last line defines the variables of the program: one Boolean variable var_a for each SON function request a in the set of technically accepted requests $\mathcal{A}^{\mathrm{TCR}}$ which indicates whether the request is selected ($\mathrm{var}_a = 1$) or rejected ($\mathrm{var}_a = 0$). The second line defines the constraints: for each conflict from the set of conflicts κ, it formulates the constraint that at most one of the variables for the conflicting requests can be 1. Finally, the first line defines the objective: the lexicographical maximization of the sum of the expected utility of the selected requests with respect to the priorities.

Based on the variable assignment that maximizes this GP problem, the set of selected, conflict-free SON function requests $\mathcal{A}^{\mathrm{OM}}$ and the set of rejected request $\overline{\mathcal{A}}^{\mathrm{OM}}$ is defined as

$$\mathcal{A}^{\mathrm{OM}} = \{a \in \mathcal{A}^{\mathrm{TCR}} \mid \mathrm{var}_a = 1\}$$
$$\overline{\mathcal{A}}^{\mathrm{OM}} = \{a \in \mathcal{A}^{\mathrm{TCR}} \mid \mathrm{var}_a = 0\}.$$

A preemptive GP problem can be solved in a sequential procedure that involves the solution of a series of nonpreemptive IP problems [HL05, "Supplement to Chapter 7"]: the maximization of the lexicographic objective $\mathbf{u} = (u_1, \ldots, u_{N_D})$ is split into

N_D optimization problems, one for each expected utility for each priority. Beginning with the highest priority utility u_1, this nonpreemptive problem is solved and the resulting best value for the single priority objective saved. Finally, the constraint that the single priority objective must be equal to the best value is added to the problem and the next process with the next lower priority continued. For objective-driven conflict resolution with $N_D = 3$ and $\mathbf{u} = (u_1, u_2, u_3)$, this means that, first, the sum of the expected utility for the priority d_1 must be maximized:

$$\max u_1 = \sum_{a \in \mathcal{A}^{\mathrm{TCR}}} \mathrm{var}_a \cdot \mathrm{proj}_1(\widehat{\Delta \mathbf{u}}_a)$$

subject to

$$\forall (a_1, a_2) \in \kappa . \, \mathrm{var}_{a_1} + \mathrm{var}_{a_2} < 2$$

and

$$\forall a \in \mathcal{A}^{\mathrm{TCR}} . \, \mathrm{var}_a \in \mathbb{B}. \tag{5.11}$$

The resulting best value for u_1 is saved and fixed as an additional constraint in the next iteration:

$$\max u_2 = \sum_{a \in \mathcal{A}^{\mathrm{TCR}}} \mathrm{var}_a \cdot \mathrm{proj}_2(\widehat{\Delta \mathbf{u}}_a)$$

subject to

$$\forall (a_1, a_2) \in \kappa . \, \mathrm{var}_{a_1} + \mathrm{var}_{a_2} < 2$$

$$u_1 = \sum_{a \in \mathcal{A}^{\mathrm{TCR}}} \mathrm{var}_a \cdot \mathrm{proj}_1(\widehat{\Delta \mathbf{u}}_a)$$

and

$$\forall a \in \mathcal{A}^{\mathrm{TCR}} . \, \mathrm{var}_a \in \mathbb{B}. \tag{5.12}$$

The best value u_2 is also saved and fixed as a constraint in the final iteration:

$$\max u_3 = \sum_{a \in \mathcal{A}^{\mathrm{TCR}}} \mathrm{var}_a \cdot \mathrm{proj}_3(\widehat{\Delta \mathbf{u}}_a)$$

subject to

$$\forall (a_1, a_2) \in \kappa . \, \mathrm{var}_{a_1} + \mathrm{var}_{a_2} < 2$$

$$u_1 = \sum_{a \in \mathcal{A}^{\mathrm{TCR}}} \mathrm{var}_a \cdot \mathrm{proj}_1(\widehat{\Delta \mathbf{u}}_a)$$

$$u_2 = \sum_{a \in \mathcal{A}^{\mathrm{TCR}}} \mathrm{var}_a \cdot \mathrm{proj}_2(\widehat{\Delta \mathbf{u}}_a)$$

and

$$\forall a \in \mathcal{A}^{\mathrm{TCR}} . \, \mathrm{var}_a \in \mathbb{B}. \tag{5.13}$$

Equation 5.13 yields the maximal utility with respect to the lexicographical ordering of the utility vectors, $\mathbf{u} = (u_1, u_2, u_3)$.

5.4 Approach for Asynchronous Coordination

The description of SON coordination so far is focused on a synchronous execution of the SON functions, i.e., the SON function requests are collected for each granularity period and processed in a batch. Hence, the input to SON coordination is a set $\mathcal{A}$ of collected SON function requests that need to be coordinated. However, if the SON functions are executed asynchronously, each request needs to be processed individually. The presented ODSO SON coordination approach can also be applied to this setting.

The simplest form of an asynchronous objective-driven SON coordination solely requires that the ODSO SON coordination keeps track of the execution of the SON function requests. When a SON function requests the execution of a configuration change, SON coordination has to check whether this request is in conflict with any currently executed SON function request. Thereby, a request is considered running if the "execution time" [Ban13, p. 72] and "impact-time" [Ban13, p. 72] of the running SON functions are not elapsed [Ban13]. If there is no conflict, then the requested change can be accepted. Otherwise, the request needs to be buffered or rejected. Once a SON function finished its execution, SON coordination can collect all requests that were buffered due to a conflict with the finished function. For this set, it can perform the coordination process as presented in the previous chapters, and trigger the accepted SON functions. Of course, it needs to consider other, currently executed SON functions.

This simple approach, however, is not optimal since it does not consider the execution and impact-times of the SON functions during the decision making. In the synchronous SON coordination, the SON functions are treated as is they all have the same impact-time which is less or equal to the granularity period (see Chapter 3.3.2). However, in asynchronous mode, this assumption is not justified anymore since SON functions may have diverse impact-times [Ban13]. In order to exemplify this consider two SON function requests $a_1, a_2 \in A$ that produce the expected utility improvements $\widehat{\Delta\mathbf{u}}_{a_1}, \widehat{\Delta\mathbf{u}}_{a_2}$. If the expected utility improvements by a_1 and a_2 are equal, i.e., $\widehat{\Delta\mathbf{u}}_{a_1} = \widehat{\Delta\mathbf{u}}_{a_2}$, ceteris paribus, the MNO will surely prefer to execute the request with the shorter impact-time since it produces the expected improvement in the network performance earlier. More interesting is the situation that $\widehat{\Delta\mathbf{u}}_{a_1} \geq_D \widehat{\Delta\mathbf{u}}_{a_2}$ but the impact-time of a_2 it greater than the impact-time of a_2. In this case, the operator needs to make a trade-off decision between the expected utility improvement and the time to produce it, i.e., does the operator prefer either a_1 which improves the network performance considerably but requires a long time for that, or a_2 which quickly improves the network performance slightly. This reasoning may even lead to the case that an operator prefers to buffer acceptable SON function requests in order to wait until another buffered but still conflicting SON function request can be executed in the future.

Such decision problems can be solved using the concept of net present value which enables comparison of utilities that materialize at different points in time [YTP93]. [Ein13] shows the application of this idea to SON coordination[3]. In principle, it

[3]This work has been supervised by the author of this thesis.

reduces the expected utility improvement of an action with respect to its impact-time based on an operator defined discount rate. Given the expected utility improvement $\widehat{\Delta\mathbf{u}} = (u_1, \ldots, u_{N_D})$ of an action with the impact-time t, the discounted expected utility improvement $\widehat{\Delta\mathbf{u}}_{\text{discounted}} = (u_{1,\text{discounted}}, \ldots, u_{N_D,\text{discounted}})$ is calculated such that

$$u_{n,\text{discounted}} = \frac{u_n}{(1+i)^t} \tag{5.14}$$

with $n = 1, \ldots, N_D$ and $i \in [0, 1]$ being the operator defined discount rate. Thereby, the i acts as a weighting factor for the time: the higher i, the more important it is for the operator to achieve even small expected utility improvements in the short run.

5.5 Related Work

The following presentation of related work focuses on automated or autonomic conflict resolution for SON coordination. An overview of the research regarding SON operations and PBM has been presented in Chapter 3.6.

5.5.1 Off-line Coordination

In contrast to SON management, there is a considerable body of research regarding SON coordination. However, a huge proportion is focused on off-line coordination, i.e., the development of SON functions that do not interfere with each other.

A common strategy for that is to see SON functions as control loops and use control theory to ensure properties like stability. For instance, [GSB11] describes a probabilistic framework for analyzing the interactions between SON functions based on a model that defines SON functions as discrete-time Markov chains. Furthermore, [Com+13] presents a coordination framework that requires modeling the algorithms of SON functions as ordinary differential equations. Based on that, the authors were able to prove typical control theoretic guarantees like stability for their approach. Another approach of this category is to design the execution intervals of conflicting SON functions such that they work on different time scales. For instance, [KK13] describes implementations of an MRO function and an MLB function whereby the former is seldom active whereas the latter changes the network configuration very often. Since the trigger times are set such they never request a change together, they are considered non-conflicting. [GBK11][Tsa+13, Ch. "Hierarchical optimization"] provide a detailed explanation why this holds even if the impact-times of the SON functions overlap.

Another strategy is to align the requests of possibly conflicting SON function through a control hierarchy. The most simple case is exemplified in [Liu+10] where an MRO function directly controls an MLB function by setting constraints on the allowed network configurations for MLB. Similarly, [RB11] and [LIA13] present approaches which assume that the internal logic of SON functions can be directly defined as a policy. This enables the MNO to design the SON functions such that

they interfere with each other as little as possible. However, this requires the manufacturers to provide a detailed action model and reveal the details of their SON Functions. [GBK11] generalizes these approaches to a "coordination by information" [GBK11, p. 1] and "coordination by control" [GBK11, p. 2]. In the same category falls the "Centralized multi-objective optimization" [Tsa+13, p. 136] approach which assumes that the SON functions send their goal function, i.e., they algorithm, to a central coordinator which performs a multi-objective optimization regarding all goal functions and determines the best network configuration. In other words, the coordinator becomes a kind of super SON function that comprises the behavior of all regular SON functions. The SEMAFOUR project [Alt+14] provide a set of general design principles for SON functions based on the approaches above.

These approaches show impressive results and, actually, can be seen as the theoretically optimal approach. However, in reality they set very tough constraints on the SON functions that are quite far from reality. On the one hand, some approaches require a lot of information about the SON functions. As a matter of fact, the algorithm details are often required. This requirement stands against the basic assumption of this thesis that the SON functions are seen as black boxes since the vendors aim to keep their algorithms secret. On the other hand, most approaches assume that all SON functions adhere to some framework in order to be usable, thereby limiting the design space for SON functions considerably. This does not solely require all SON function vendors to agree on such a constrained framework, but also limits their ability to separate themselves from competitors. As a result, such approaches are only possible if the whole SON system, i.e., all SON functions, is provided by one single vendor since the vendor has the knowledge and control to enforce the satisfaction of the requirements outlined above. However, such a system stands against the idea of SON as a flexible network optimization system.

5.5.2 On-line Coordination

There is some related work regarding on-line coordination, i.e., the coordination of SON functions at run time that have not been developed to be executed together. That is, the SON function developers assumed that their SON function is executed in isolation in the network. As a result, these coordination approaches intrinsically enable the operation of functions by multiple vendors. Along the ODSO SON coordination component design, the related work can be classified into two types: first, approaches that focus on the effect estimation for the SON functions and perform a simple decision making, and, second, approaches that perform a simple effect estimation (if at all) and focus on an elaborate SON function request selection.

5.5.2.1 Effect Estimation

The SOCRATES project was one of the first big research activities regarding SON operation as introduced in Chapter 3.6.2.2. It acknowledged the need for a coordinated execution of SON functions [Sch+11], referred to a "tailing harmonization" [Sch+11, p. 195]. The presented SON coordination concept is a great source

of interesting ideas, e.g., the SON function requests may be evaluated based on their expected effects and SON functions may provide the KPI value predictions. However, there is not even a conceptual outline how this might be implemented. Hence, the actual coordination concept is unclear.

There are also more concrete approaches for effect estimation published. They have in common to utilize machine learning to build a model allowing the prediction of the effects. For instance, [DA10] shows an approach to coordinate different instances of one SON function[4]. Thereby, fuzzy reinforcement learning is used to estimate the effects and learn an optimal selection strategy.

An approach developed in the UniverSelf project (see Chapter 3.6.2.3), dubbed "Self-orchestration through Utility Predicates" [Ben+13c, p. 36], requires the SON functions to predict the utility of their actions and send it to a central coordinator. Conflicts are resolved by selecting the request with the highest prediction. After execution, the actual network performance is measured and compared to the prediction. Using reinforcement learning, the approach can learn when SON functions make bad predictions and, thus, optimize the internal prediction. Another point of view is that the system learns when it can trust the predictions [Cia+12]. Unfortunately, it is not presented how the SON functions may come to their predictions. In [Iac+14b], a similar concept that has been developed in the SEMAFOUR project (see Chapter 3.6.2.5) is presented. It requires SON functions to report a "happiness" [Iac+14b, p. 198], a simple indication of the current network performance with respect to their optimization goal. Using reinforcement learning, the approach learns which combination of accepted requests yields the highest expected improvement in the happiness.

Unfortunately, the latter two concepts cannot be used without adaptation in ODSO coordination for two reasons: First, the learning of a single performance indicator, i.e., utility or happiness, builds a model of the combined performance prediction over several KPIs instead of a prediction for each single KPI. Hence, the learned knowledge is rendered useless if the objectives change. Second, the SON function-internal effect estimation requires the functions to understand the objectives. In other words, they must be able to evaluate the objective model of the operator. This leads to a concept called cognitive functions which is a next step in the research on SON (see Chapter 8.2).

All presented concepts have in common that they do not consider operator-defined, SON function-independent KPI objectives for their decision making. Furthermore, only simple, transitive SON function conflicts are considered. Hence, their selection mechanisms do not provide an optimal solution for complex conflicts spanning several network cells.

5.5.2.2 SON Function Request Selection

The need for a dedicated objective-driven conflict resolution between SON function requests has not been identified by related work yet to the best of our knowledge.

[4]The authors present Inter-Cell Interference Coordination (ICIC), however, the actual functionality is not relevant.

Consequently, the published approaches for SON function request selection mix up technical constraint resolution, effect estimation, and objective-driven conflict resolution. As outlined in Chapter 3.5, this increases the manual efforts for maintaining the respective models since the technical knowledge cannot evolve independently of the operational objectives.

The probably most simple approach for conflict resolution is to use a "random token" [Tsa+13, p. 132] that selects one request of a conflict based on pure chance. However, even the 3GPP acknowledges in their standards [3GP13] that the operator might need to control the process. Therefore, the usages of priorities or rules is suggested.

As already stated, the SOCRATES project [Sch+11] was among the first to present a SON coordination concept (see Chapter 5.5.2.1). The idea is that a coordination policy is derived from the overall operator objectives. This policy defines constraints for the operation of the SON functions that ensure the satisfaction of the objectives. For instance, it may define priorities for the SON functions, or enable to evaluation of the effect prediction by the SON function regarding a cell-specific operator policy. However, besides the general statement that this task is not straightforward, the SOCRATES projects does not provide any further details how this may be accomplished.

The most common approach for operator-controlled conflict resolution is the usage of a rules system, often also referred to as a policy. [Ban13] uses a set of Event-Condition-Action (ECA) rules to detect and resolve conflicts between requests, e.g., on the event that an MLB function request occurs, if there is a conflicting MRO request then reject the MLB request. Thereby, the reason for this decision logic can be a technical constraint, an operator objective, or both. The COMMUNE project (see Chapter 3.6.2.4) describes, in principle, the implementation of this approach within the GARSON framework [Bar+13a]. Also [LIA13] proposes a coordination scheme with ECA-rules, which the authors refer to as Trigger-Condition-Action (TCA) rules. Among the numerous generic coordination approaches proposed by the UniverSelf project, the "Multi-priority event driven separation in time" [Tsa+13, p. 139] resembles a policy-driven concept based on rules. [GSB14] propose a rule-based coordination framework that is based on a thorough analysis of the monitored measurements and the adapted network configuration parameters of SON functions.

Compared with the ODSO SON coordination approach, all rule-based approaches, in principle, assign dynamic priorities to the SON function requests. This requires detailed knowledge how each SON function affects the network in order to determine a prioritization that satisfies the operator objectives. Furthermore, the priorities are fixed. Although the policy allows some adaptation to the operational context, this is typically limited to CM data, e.g., the cell's location, and does not consider the cell's performance in terms of KPI values. In contrast, ODSO SON coordination allows the operator to define objectives regarding the network performance and let the system automatically determine the best coordination decision. Note that it is theoretically possible to emulate objective-driven coordination with a set of rules. However, this requires to pre-calculate the best actions for all possible combinations of SON function requests in all possible KPI value combinations. The result of this

non-trivial and probably non-tractable computation would make up the set of rules.

Apart from this, none of the related conflict-resolution approaches considers complex, non-transitive conflict relations. Hence, they do not provide an optimal solution for a conflict situation as presented in Figure 5.7.

6 SON Self-Healing

SON management and SON coordination are controlling a SON-enabled network according to the operator objectives. Both are effective if the network and the SON are functioning normally because this is the assumption for the technical models they use. Consequently, if the network encounters some problem that prevents normal operations, both cannot effectively react to that. "The purpose of [SON] Self-healing is to solve or mitigate the faults which could be solved automatically by triggering appropriate recovery actions." [3GP14c, p. 7]

This chapter introduces the ODSO SON self-healing component that determines an optimal recovery strategy for a problem with respect to its severity and the recovery action costs. Therefore, we first outline the two problems of current SON self-healing approaches: the neglect of the SON itself for problem detection and diagnosis, and the lack of an autonomic degradation recovery. Based on this, we present the desired behavior of the ODSO SON self-healing component according to Objective 4. The two parts of Solution and Contribution 2 are described thereafter: on the on hand, the integration of run-time data from the SON into degradation detection and diagnosis, and, on the other hand, the application of the generic ODSO component design to the task of degradation recovery. Thereby, it is shown how the preferences regarding the recovery actions can be considered in decision making.

6.1 Problem and Motivation

SON self-healing aims at detecting and automatically mitigating faults in the network. In mobile network operations, the operational personnel is faced with two types of failures [3GP14c]: On the one hand, common, known issues that are relatively easy to detect and diagnose, and can be automatically recovered. Examples of this category are failing hardware components for which a standby replacement exists, software bugs in a new software version, or an erroneous configuration of the BS. These can be recovered with a simple self-healing workflow, i.e., a well-defined process of recovery steps. For the given examples, the recovery workflows might be switching to a spare, standby hardware component, reverting a software update to a previous version, or undoing a configuration change. On the other hand, there are unknown, rare failures that are very difficult to detect or not automatically recoverable. These errors require a lot of human intuition, knowledge and creativity to handle which makes them very difficult to automate [NGM08]. As a consequence, SON self-healing's focus is on the first type of failures.

Figure 6.1 depicts the generic design of SON self-healing as a four step process: First, failures are detected directly in the network, e.g., by a BS, or through cell degradation detection[1] that analyzes the network data. Second, once a problem is detected, root cause diagnosis is informed via an alarm and determines the reason for it. Third, degradation recovery determines recovery actions for the diagnosed problems which may be automated self-healing workflows or the escalation to the operator through a trouble ticket. Fourth, action enforcement controls the execution of the recovery actions.

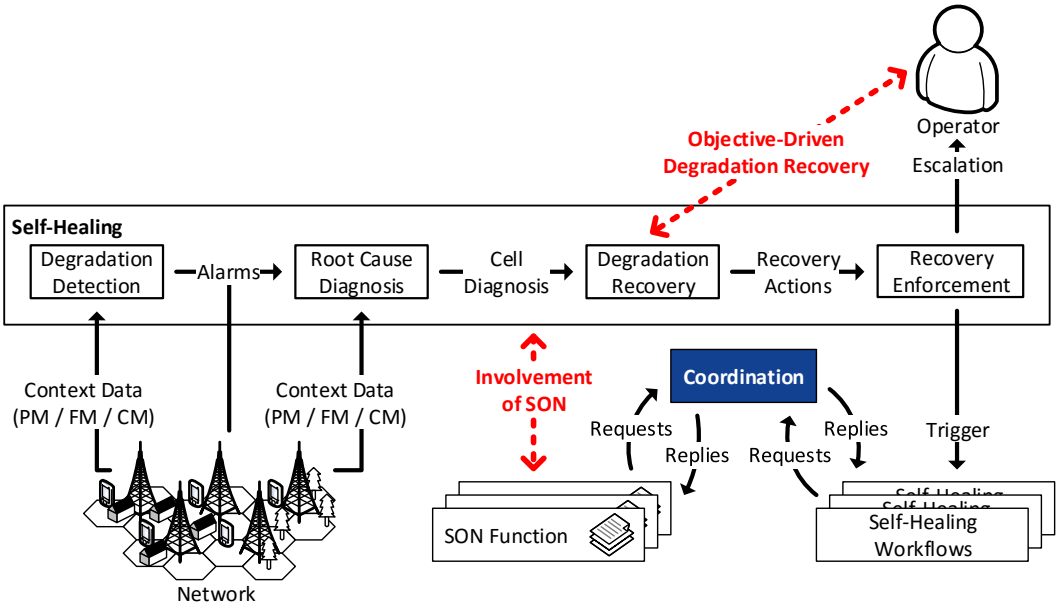


Figure 6.1: State-of-the-art SON self-healing process and identified problems in red.

This design has two shortcomings: On the one hand, SON self-healing does not consider the SON, and specifically the SON functions. This makes the self-healing concept incomplete since SON functions can also be seen as a piece of software that can produce problems that need to be mitigated. Hence, Figure 6.1 is missing a concept for involving the SON. On the other hand, while there has been significant research on cell degradation detection and root cause diagnosis, an automatic degradation recovery approach is missing. In other words, most SON self-healing approaches solely aim to present a human operator with an accurate diagnosis. Therefore, self-healing cannot be fully automated since the operator still has to select a recovery action even in simple cases.

[1] Often related literature considers solely the more specific Cell Outage Detection (COD). Following [Nov+11], we focus on the broader term degradation here.

6.1.1 Involvement of SON

SON functions are typically closed-loop control systems that monitor the network performance and, if necessary, adapt the configuration of the network. The decision algorithms are based on several assumptions about the network and its operating point. For instance, an MRO function assumes that there is no coverage hole between two neighboring cells that might cause handover problems, an MLB function assumes that the network has sufficient capacity to cope with the traffic, and a CCO function assumes that the cells in the network are functional and able to cover the area without holes. Typically, these assumptions are not checked by the SON function since they are hard to verify. If an assumption does not hold at run time, the respective SON function might be rendered ineffective, i.e., it is not able to achieve its target. Specifically this means that, e.g., MRO is not able to improve the handover performance such that the operator objectives are satisfied. Such a situation can occur if, e.g., newly erected buildings lead to coverage holes due to shadowing or a public demonstration leads to an unforeseen hot-spot. As a result, an ineffective SON function can also be an indication of problems in the network.

A subtle problem about an ineffective SON function is that it can also affect other SON functions, leading to a non-functional SON in the worst case. This is due to the coordination of interacting SON functions. An ineffective SON function may, e.g., produce configuration oscillations, i.e., it continuously requests changes of the cell configuration at SON coordination. This is due to the state-less decision algorithms in the SON functions that often do not analyze an execution history. If the requests are always accepted by SON coordination, then other, conflicting SON functions are never executed. This situation, referred to as monopolization, can be seen as a deadlock in the SON that blocks the optimization of the network performance to satisfy the operator objectives. As an example, consider an MRO function that attempts to improve the handover performance in a cell by adjusting the cell's configuration. However, the high number of handover problems is caused by a small coverage hole due to shadowing. Although the CCO function detects the coverage problem, the handover problems might induce a more severe performance degradation and, thus, SON coordination accepts only requests by MRO. As a result, MRO continuously changes the handover configuration without any performance improvement and, at the same time, blocks the execution of CCO that might be able to resolve the problem.

Regarding SON self-healing, SON functions cannot only be an indicator or the cause of a problem but also the recovery, though. As shown in the MRO-CCO example above, the execution of a SON function, specifically CCO, may resolve problems that render other SON functions ineffective. Furthermore, it might also be the case that a SON function is not able to detect the problem but self-healing is. Hence, the active execution of a SON function by SON self-healing should be considered as a valid recovery action.

6.1.2 Objective-Driven Degradation Recovery

The 3GPP SON self-healing distinguishes two steps [3GP14c]: monitoring the network to detect outages and healing them. The term healing refers to the analysis of the detected problem using additional information, the determination of appropriate recovery actions, and their execution. The first step specifically means that the root cause of a detected cell outage or degradation in general needs to be determined. This is a complicated problem that attracted considerable research (see [Nov+11] for an overview), including similarity-based [SN12], Bayesian net-based [Bar07][Kha+08], and self-organizing map-based approaches [Lai+05]. The majority of them aims to come up with a diagnosis result showing the likeliness of possible root causes. For instance, the root cause diagnosis can determine that a detected performance degradation in a cell is due to a coverage hole with 45%, a software problem with 35%, or broken hardware with 20% probability. This research is partially so advanced that there are even products implementing these concepts available [Dez14][2op10].

However, most research on healing neglects the recovery and execution part. Hence, it is not clear how to use a root cause diagnosis result in order to automatically determine the most rational recovery action. That is why we explicitly distinguish between the three tasks in Figure 6.1. A simple solution could be to map each possible root cause with one automatic recovery workflow. Based on this, the action for the most likely problem is executed. This approach, however, has two shortcomings: First, the mapping between root causes and recovery actions is typically many-to-many, i.e., a root cause might be mitigated by several self-healing actions and an action might recover the network from several possible root causes. For instance, a trouble ticket, i.e., triggering a manual inspection of a degradation, can mitigate every root cause. So, a problem caused by a crashed BS software might be recovered by, e.g., restarting the BS or creating a trouble ticket. Second, this solution does not consider the operator preferences for scoring possible recovery actions. Thereby, the application of the preferences is twofold: on the one hand, they can be used to assess the severity of a degradation in order to determine how important the recovery is for the MNO, and, on the other hand, they allow to rank the actions depending on their efforts and impacts. For instance, an operator will try to avoid the creation of a trouble ticket since it requires human effort to process it. Similarly, the execution of a SON function is preferred to a restart of a BS since the latter may lead to a temporary outage. In general, an automatic recovery should be attempted for moderate degradations though it might not be effective. However, if the degradation is severe and the diagnosis is inconclusive then a trouble ticket should be created immediately. Furthermore, if several cells need to be recovered then the most severe cells should be recovered first.

6.2 Goals and Requirements

The goal of objective-driven SON self-healing, as stated in Objective 4, is to provide autonomic self-healing for a SON based on the ODSO approach. Going beyond trou-

bleshooting of hardware, software, and configuration failures in the BSs, it should also involve the SON. Thereby, the SON functions may be considered in three ways:

- as probes for the detection of possible problems, e.g., an MRO function detects a coverage hole,
- as root causes of problems, e.g., a SON function has a bug and affects the network negatively, and
- as recovery actions for problems, e.g., a CCO function can mitigate a detected coverage hole. This also includes blocking the execution of SON functions.

The consideration of the operation of the SON as part of SON self-healing enables MNOs to gain trust in the automatic SON system, since abnormal behavior of the SON functions can be detected and automatically mitigated.

There is already a considerable amount of related work regarding degradation detection and root cause diagnosis in SON self-healing. Therefore, the presented approach does not aim to extend this field of research but, instead, draws on this work and focuses on the merely investigated topic of degradation recovery. The goal is to automatically mitigate simple and known problems based on the operator objectives. It is clear, that an automatic recovery is only possible for well-known problems with automatic recovery workflows. That means, for each problem, there must be an automatic detection procedure and automatic recovery actions that can work without human intervention. The operator should decide which concrete problems these are since they depend on the degree of trust by the operator in the SON system. For instance, in an early phase of SON, the MNO might be reluctant to automatically recover any failure since the personnel does not trust the automatic procedures. Consequently, most diagnosed problems will be escalated as trouble tickets. However, in a mature phase of SON, the MNO might feel much more confident in autonomic SON self-healing and lets the system mitigate most problems without intervention.

Degradation recovery needs to integrate several sources of information for its decision making in order to select the best recovery action. Specifically, it should consider the following:

- The probability that an action is effective, i.e., it resolves the degradation. This depends on the probabilities of the possible root causes for the degradation given by the root cause diagnosis and the effects of the action.
- The severity of the degradation, particularly with respect to the operator objectives on the KPIs.
- The reluctance of the operator to execute the action due to the required efforts or its performance impact. This is often referred to as cost but, here, goes beyond monetary measures.

Based on this, the most rational action should be executed, i.e., either an automatic recovery action or the creation of a trouble ticket. In general, the decision making

should select automatic actions if it is very likely that it recovers the problem and the operator is not reluctant to use it. However, an ambiguous root cause diagnosis result should preferably be manually inspected.

The automatic execution of recovery actions by SON self-healing must, of course, be integrated with regular SON operations. Hence, the execution of recovery workflows must be coordinated with the execution of regular SON functions. Therefore, in line with ODSO SON coordination (see Chapter 5.3), we assume a synchronized execution of SON self-healing, i.e., the detection and diagnosis is triggered in regular intervals together with the SON functions as described in Chapter 3.3.3.

In summary, the specific contributions of this thesis to SON self-healing are:

- a concept for the integration of SON functions into self-healing as probes for the detection of problems, as possible root causes of problems, and as recovery actions for the mitigation of problems,
- an objective-driven degradation recovery that automatically triggers countermeasures for operator-defined problems guided by the goal to maximize the satisfaction of operator objectives, and
- an integration of this concept into the overall ODSO architecture, particularly SON coordination.

6.3 Component Design

In order to achieve the goals for SON self-healing, the function design depicted in Figure 6.1 is extended as shown in Figure 6.2. On the one hand, the involvement of SON is achieved by considering the SON functions for self-healing: first, their activity is monitored through SON coordination in order to detect problems, second, they may raise alarms by themselves if they detect a degradation, and, third, their coordinated execution may be triggered and controlled by self-healing. On the other hand, a degradation recovery component based on the generic ODSO component design is introduced. It analyzes the possible root causes and, guided by a recovery model and the SON function effects, determines a recovery action that maximizes the satisfaction of the operator objectives. Thereby, it weighs the preferences regarding the recovery actions, provided as a cost model, with the operational objectives. In the following, the building blocks of ODSO SON self-healing are described with a focus on the newly introduced concepts.

This design can be aligned with the two-step 3GPP general self-healing procedure [3GP14c]: the degradation detection and the SON functions that raise alarms corresponds to the "monitoring part" [3GP14c, p. 8] and the other three components to the "healing process part" [3GP14c, p. 8]. Consequently, degradation detection can be seen as an implementation of the "Self-healing Input Monitoring Function" [3GP14c, p. 11]. Furthermore, root cause diagnosis and degradation recovery are related to the "Self-healing Diagnosis Function" [3GP14c, p. 11]. Finally, recovery enforcement is an implementation of the "Triggering Recovery Action/s Function" [3GP14c, p. 11], the "Self-healing Evaluating Function" [3GP14c, p. 11],

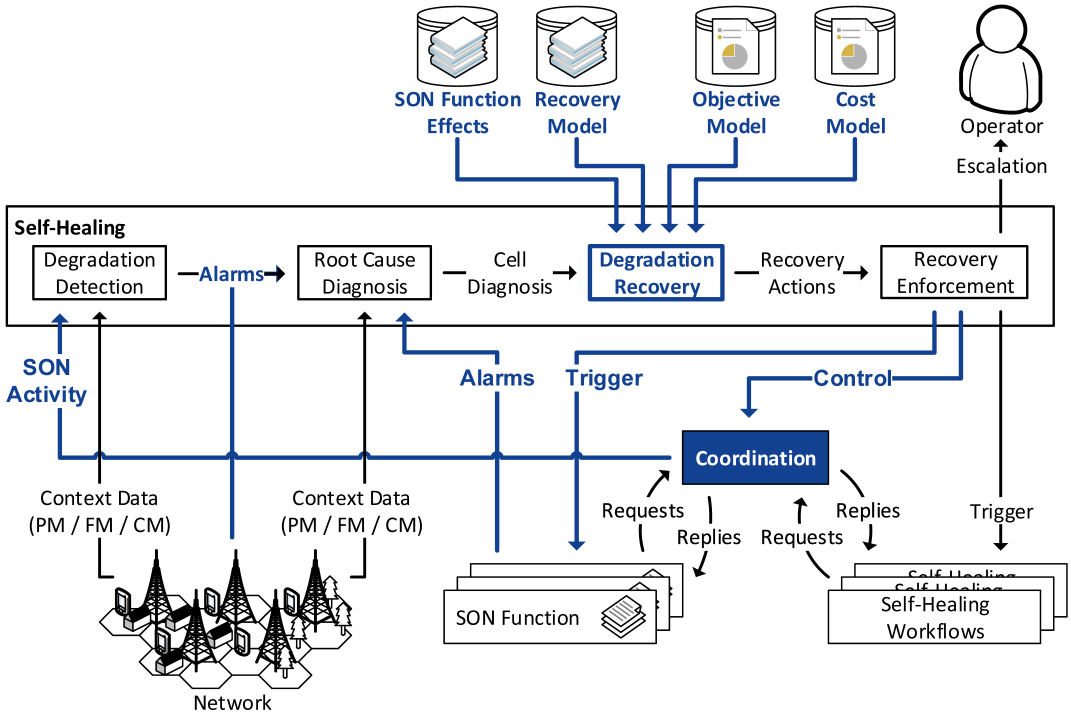


Figure 6.2: Design of the extended SON self-healing for ODSO. New or adapted components are blue and bold.

and the "Self-healing Fallback Function" [3GP14c, p. 11] from the respective 3GPP standard.

6.3.1 Detection of Degradations

There are several entities that continuously monitor the network for abnormal situations in the network, i.e., a "Trigger Condition of Self-Healing" [3GP14c, p. 7]. In the following, we first present traditional, network-focused degradation detection approaches before we introduce the new approach to integrate the SON into degradation detection. For both cases, an alarm is created once a problem is detected that notifies the root cause diagnosis about the incident. An alarm is, hence, seen as a general indication that some abnormal situation occurred. It should contain additional information which provide more details about the detected abnormality. Besides the impacted BSs or network cells, this information can comprise, e.g., abnormal KPI values or an indication of the severity of the abnormal situation [SN12]. In the following, the entities that might produce such alarms are presented in more detail.

6.3.1.1 Network-focused

Traditionally, the focus of the self-healing monitoring part lies on the detection of network-focused, abnormal situations in a BS or network cell. Often the BSs are monitoring their operation by themselves and can notify the self-healing function about a problem as depicted in Figure 6.2. This is the classical meaning of an alarm, e.g., in the "Alarm Triggered Self-healing" [3GP14c, p. 13] use case. These alarms are typically indicating hardware or software problems, e.g., a problem with the installation of a new software version or a reduction of the main power [3GP14c].

The challenging task is to detect problems that do not cause alarms, e.g., because a hardware component failed that cannot be actively monitored or the software has some bug. This is often referred to as sleeping cell and one of the focus areas of related research [Nov+11]. The principal solution approach is to monitor and analyzes the PM and CM data produced in the network for abnormalities, i.e., unexpected values. Thereby, normal variations due to the seasonal or random environment need to be considered. This task is performed by the degradation detection component, shown in Figure 6.2, which utilizes sophisticated statistical algorithms [Nov+11][2] for that, e.g., Kolmogorov-Smirnov test [Nov13] or merge growing neural gas algorithm [GNM15]. There are even indirect detection approaches that analyze Twitter data [TYN15]. In this way, degradation detection is able to detect, e.g., sleeping cells that are in outage without producing an alarm, or cells that are experiencing some abnormal KPI values. The KPIs as well as the degree of deviation from the normal behavior can be important information for root cause diagnosis and, so, should be attached to the alarm.

6.3.1.2 SON-focused

A neglected source of information for detecting anomalies is the SON itself. However, particular problems may be detectable by analyzing the execution of the SON functions. Although they are not particularly developed to look for such things, it may happen that some problems cause a SON function to behave anomalous. The reason for this is that developers of SON functions typically assume a failure-free network and, so, a network problem may put the function into an unforeseen state. In such situations, the SON algorithms may not be effective anymore. Consequently, the alarms generated through this data source indicate that a SON function is ineffective and not able to achieve its target. Additional details about the ineffectiveness can be useful for the following self-healing part.

The ineffectiveness of a SON function can, in principle, be detected with two approaches: state-based and history-based detection. The former focuses its analysis on the current network state, i.e., PM, CM, and FM data, and the reaction of the SON functions in the current granularity period. The latter performs a similar analysis, however, over a sequence of recent network states and activities. This enables a more complex temporal analysis that allows extrapolating trends. This way it is possible to detect, e.g., a SON function that continuously modifies network

[2] [CBK09] provides a general, domain-agnostic overview of anomaly detection methods.

parameters without any performance improvement, i.e., monopolization.

Apart from the detection of otherwise unnoticed degradations, monitoring the execution of the SON functions also provides the subsequent root cause diagnosis with an additional source of information. Hence, these alarms may confirm or refute some assumed root causes for a degradation which, consequently, increases the accuracy and trust in the cell diagnosis results. In the following, we present two approaches for monitoring a SON in order to detect a degradation.

SON Function Alarms A SON function is designed for the detection of specific performance issues based on KPIs from the network, and the determination and execution of corrective changes of network parameters. Therefore, they employ sophisticated algorithms [Las+11]. By integrating the SON functions as probes into SON self-healing, it is possible to exploit these algorithms in two ways: On the one hand, reuse the complex mathematical monitoring and analysis procedures to directly detect anomalies, i.e., unexpected system states. For instance, an advanced MRO function may create an alarm if there are lots of too-early and too-late handovers which result in an inconclusive algorithm state. On the other hand, detect irregularities during the execution of the algorithms. This is not possible from the outside since the functions are considered as black-boxes. For instance, MLB may create an alarm if it cannot offload any more user in a cell due to a limit for the CIO parameter. In summary, a SON function may raise alarms if it detects problematic situations that it cannot correct by itself. Note that such an alarm concept is currently not considered by the respective standards, e.g., [3GP13].

The complexity of a SON function should not increase due to the degradation detection. Instead, it is seen as a side product of the actual optimization algorithm. Hence, the type of detection approach depends on the optimization algorithm. State-based detection approaches, in general, allow detecting problems from anomalous PM data that is not foreseen by the algorithm so that it runs into an exception. History-based detection methods may detect deviations and trends in the network behavior. For instance, a SON function that learns how a configuration change affects the KPIs may raise an alarm if, at some point in time, the learned model differs significantly from the actual reaction.

SON Degradation Detection In a mobile network there will always be a mixture of alarming-enabled and traditional SON functions. Nevertheless, legacy SON functions may still be exploited indirectly for degradation detection by monitoring and analyzing their external behavior, i.e., their activity. A centralized degradation detection additionally has a broader scope of monitoring by integrating data from all SON functions and all network cells. This may allow detecting problems beyond the focus of a single SON function instance since a wider range of PM, CM, and FM data, e.g., the performance of a group of BSs in a specific area, system-level KPIs or Minimization of Drive Tests (MDT) [Tom+11] data, may be correlated with the activity of SON functions.

The disadvantage of indirect monitoring is that degradation detection has no information about the algorithm or the internal status of the SON function. Hence,

its analysis must always be based on assumptions about the logic of the functions. If these are not correct, this can lead to false inferences and, consequently, false alarms or unnoticed problems. For instance, a continuously active CCO function might indicate a coverage hole produced by a BS failure. However, if an MLB function is often executed, this does not necessarily indicate a problem because MLB strongly dependents on the user behavior that might change regularly. As a result, knowledge and history-based inference mechanisms may be required.

As an example, a simple, generic approach for detecting ineffective SON functions may be based on the assumption that an active SON function attempts to improve the KPI values. Conversely, if a SON function does not achieve any improvement over several consecutive executions it may be considered ineffective. This generic approach may be extended to consider the SON function effects provided by SON management, which provide an indication of the expected system that the SON achieves. Hence, if the actual system state deviates from the expected system state for a considerable amount of time, this indicates that the SON functions might not be able to achieve this desired state. Therefore, it may be either the case that SON functions are actively changing the network configuration without positive effects, or that SON functions are not active due to an error or a wrong SFC.

The best entity to provide data about SON activity, i.e., configuration changes by the SON functions, is SON coordination since it already monitors and controls the request by SON functions for conflict resolution. Besides that, it may also report the actual set of SON function requests and the detected conflicts to the degradation detection. This enables the identification of repeatedly occurring conflicts which might indicate a non-optimal SON configuration with respect to the avoidance of conflicts (see Chapter 4.3.3.1).

6.3.2 Root Cause Diagnosis

An alarm represents an anomaly in the network, e.g., a failure of a BS, unexpected KPI values, or abnormal behavior of a SON function. As such, they indicate symptoms of possible degradations, i.e., the visible result of a problem. It is the task of root cause diagnosis to analyze all detected anomalies in the operational context and derive possible root causes for the problems. This step is important since, just like in medicine, the root causes must be treated in order to thoroughly resolve the symptoms.

Root cause diagnosis in mobile networks is a difficult matter since one symptom may be produced by different root causes and one root cause may generate several symptoms. Furthermore, this $n : m$ relation is probabilistic, i.e., two instances of the same root cause may produce different subsets of the possible symptoms. For instance, an MRO function alarm may be caused by a BS failure or coverage hole. Although root cause diagnosis is complex, there has been considerable research on the topic that produced a set of common approaches: [Nov+11] presents an overview over rule-based systems, Bayesian nets, case-based reasoning systems, and neural networks. Due to the stochastic nature of the relation between root causes and symptoms, however, the most promising research is focusing on probabilistic

approaches like Bayesian nets [Kha+06][Bar07][Kha+08][2op10][Ben+12][Dez14].

For ODSO SON self-healing, we draw on previously presented diagnosis approaches. Thereby we focus on probabilistic approaches since, on the one hand, they provide a profound theory and, on the other hand, match nicely to the ODSO concept. Consequently, given a set of alarms and additional context information, root cause diagnosis derives a cell diagnosis, i.e., a set of possible root causes and their probabilities that may cause a degradation in a network cell. Thereby, a single fault assumption is adopted, i.e., it is assumed that at only one of the probable root causes is truly present and not more. This assumption is commonly adopted for root diagnosis since it reduces the necessary amount of knowledge for inference [Bar+06]. Furthermore, [Bar+06] also shows that the results derived under single fault and multiple fault assumption typically do not differ significantly.

Definition 6.1 (Cell diagnosis)**.** Root cause diagnosis produces a cell diagnosis (c, ρ_T), i.e., a pair of a degraded cell $c \in C$ and a probability distribution over all root causes T. $\rho_T : T \to [0, 1]$ is a function that maps a root cause $t \in T$ to its likelihood, i.e., $\rho_T(t) = \Pr(t \text{ is the root cause of the problem})$. Thereby, the root cause probabilities must be complete such that $\sum_{t \in T} \rho_T(t) = 1$.

Some root cause diagnosis approaches do not necessarily provide a probability distribution that sums up to 1. The reason is that the problem may be produced by some unknown cause. In that case, the probability distribution can be extended with a root cause unknown $\in T$ that is assigned the missing probability portion. Furthermore, it is of course possible that the diagnosis actually detects several problems given a set of alarms. In this case, each problem is seen as a single recovery case that is handled independently. Consequently, root cause diagnosis creates a cell diagnosis for each problem, and the following degradation recovery needs to be triggered for each case separately.

6.3.3 Degradation Recovery

Degradation recovery is an often neglected part of SON self-healing. It refers to the task of creating a recovery strategy for a probabilistic cell diagnosis determined by root cause diagnosis. Thereby, the system has to consider both technical knowledge, i.e., the likeliness of the root causes, the effectiveness of the recovery actions and the $n : m$ relation between the causes and actions; as well as operational objectives, i.e., the severity of the degradation regarding the operational objectives and action preferences. Due to this two-fold information, the ODSO is a well-suited approach for implementing degradation recovery.

The design of objective-driven degradation recovery is depicted in Figure 6.3. In line with the adopted generic ODSO component design (see Chapter 3.4), it first performs a recovery action proposal for the cell diagnosis that is provided by root cause diagnosis and comes up with possible recovery actions including their probabilistic effect. This step corresponds to ODSO action proposal and is based on a recovery model, the SON function effects, and the current operational context.

During recovery action valuation, the actions are scored and sorted along the utility of their effects and the MNO preferences regarding the actions. This step is related to ODSO action selection and based on the objective model by the operator as well as an additional cost model that represents the action preferences. The result of this step is a list of the possible recovery actions that is sorted along the MNO preferences.

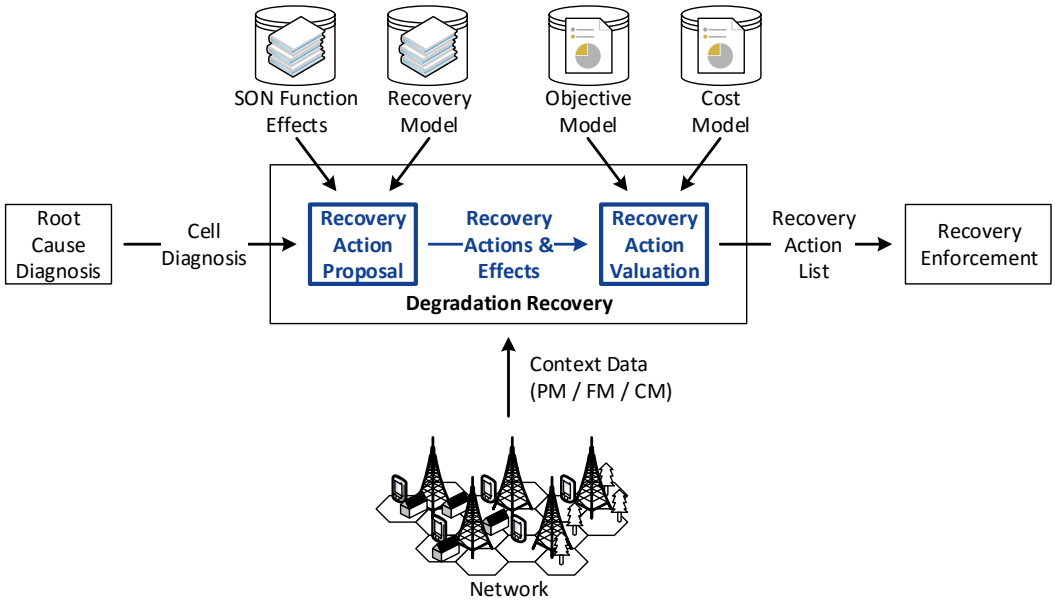


Figure 6.3: Design of degradation recovery. The steps of the generic ODSO component design are drawn in blue.

6.3.3.1 Action Types

In principle, there are two types of recovery actions that can be triggered by SON self-healing:

Trigger actions are automated procedures that can be executed without any human intervention. They are attempts of SON self-healing to mitigate a problem automatically. After their execution, SON self-healing is able to monitor the results of the action and, if it was not effective, trigger some other action. Traditionally, these actions comprise automatic self-healing workflows like restarting a BS or triggering COC. However, with the inclusion of SON into self-healing, it may also trigger or block the execution of a regular SON function like CCO. Notice that all these automatic actions need to be integrated with SON coordination.

Escalation actions are often non-automated procedures to handle a problem. They can be seen as alarms by SON self-healing itself, indicating that it cannot au-

tomatically mitigate a problem automatically. These actions cannot be monitored and, hence, an escalation action is the end of the recovery attempt. The most common escalation action is the creation of a trouble ticket that triggers a manual inspection of a problem. Another example for such an action is triggering SON management to adapt the SON configuration. This can be necessary if, e.g., SON coordination detects a lot of conflicting SON function requests that might be due to a non-optimal configuration of the SON. Though this feedback is not presented in this work, [Cam+15] provides some corresponding ideas for adapting the SON function models. Similarly, SON coordination may be informed about newly detected conflicts between SON functions that are currently not prevented and need to be added to the conflict detection model.

Independent of its type, an action is seen as a sequence of steps that need to be executed. Hence, an action must not necessarily be exactly one command. For instance, a recovery action for an ineffective SON function can comprise two steps: blocking the function and triggering some other SON function. Thereby, automatic and non-automatic steps can also be mixed. For instance, an escalation action for an ineffective SON function can comprise the steps: block the function and create a trouble ticket. A recovery action corresponds to an action in the generic ODSO component design.

Definition 6.2 (Recovery actions)**.** In the context of SON self-healing, an action as defined in Definition 3.5 is refined to a recovery action. A recovery action $a \in A$ is an action to resolve a root cause of a network problem, i.e., either a trigger action or an escalation action. Thereby, A is the set of all possible recovery actions. A recovery action is targeting the network cell $c = \text{proj}_C(a)$ in the sense that its execution affects the KPIs of $c \in C$.

6.3.3.2 Decision-Making Approach

This sections presents the abstract formulation of the decision making problem that degradation recovery attempts to solve. Therefore, the principle structure of the decision is visualized in Figure 6.4 in form of an influence diagram (see Chapter 2.3.4). The input variables shown at the top are, on the one hand, the deterministic operational context $\mathbf{x} \in \mathbf{X}$ including the current values of the KPIs for the cell $c \in C$ which are known, and, on the other hand, a probability distribution $\rho_T(t)$ over the possible root causes $t \in T$ given by the cell diagnosis (c, ρ_T). In the following presentation, V_T denotes a probabilistic variable for the true root cause and V_k denotes a probabilistic variable for the future value of the KPI $k \in K$.

The recovery action proposal determines the applicable actions and predicts their expected effect on each KPI for the cell c. Figure 6.4 highlights the variables involved for the prediction regarding the KPI $k = 1$. So, degradation detection first determines the complete KPI effect on k based on an assumed root cause $t \in T$, the given current KPI value $\mathbf{x}(c, k) \in \text{Dom}(k)$, a selected recovery action $a \in A$, and the operational context $\mathbf{x} \in \mathbf{X}$. This probability distribution can be denoted

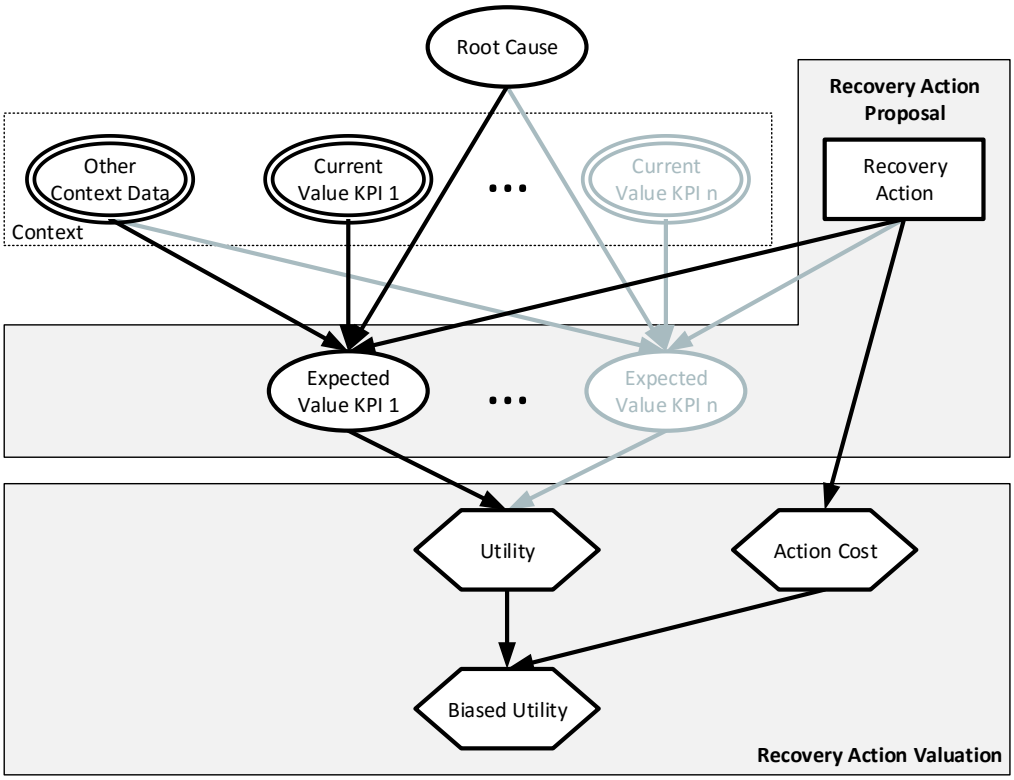


Figure 6.4: Influence diagram formalizing the decision problem of degradation recovery.

as $\Pr(V_k \mid V_T = t, \mathbf{x}(c,k), a, \mathbf{x})$. This prediction is formalized in the recovery model RM. Based on this, recovery action proposal determines the expected, complete KPI effect $\Pr(V_k \mid V_T, \mathbf{x}(c,k), a, \mathbf{x})$ for the probabilistic cell diagnosis V_T as

$$\Pr(V_k \mid V_T, \mathbf{x}(c,k), a, \mathbf{x}) = \sum_{t \in T} \rho_T(t) \cdot \Pr(V_k \mid V_T = t, \mathbf{x}(c,k), a, \mathbf{x}). \tag{6.1}$$

This probability distribution is represented by, e.g., the chance node *Expected Value KPI 1* in Figure 6.4. In other words, the expected KPI value after the execution of the recovery action a is the weighted sum of the KPI effects of a under the condition that a root cause t is present weighted with the probability that t is present. This calculation is based on the assumption that the KPIs are independent (see Chapter 3.4.1.2).

In recovery action valuation, the expected KPI effects $\Pr(V_k \mid V_T, \mathbf{x}(c,k), a, \mathbf{x})$ of an action a are evaluated against the operator objectives in order to determine the expected utility of a as shown on Chapter 3.4.4. However, this expected utility is not suitable for selecting the recovery action to perform. For instance, consider the creation of a trouble ticket: a manual inspection of a problem will likely resolve every possible root cause, so, the expected effects will be quite positive regarding the operational objectives. However, the operator does not want to inspect every simple

problem since this requires a lot of manual efforts. Therefore, the expected utility is finally evaluated against the action cost to get the biased utility. Note that it is not necessary to consider the current performance of the cell for the action selection as it is for SON coordination (see Chapter 5.3.2.2). The reason is that SON coordination compared actions with effects on different cells, however, in degradation detection, it is assumed that the proposed actions for a cell diagnosis (c, ρ_T) affect the same cell c.

The influence diagram shown in Figure 6.4 is based on the single failure assumption: the root cause variable in the influence diagram can only take on one value, i.e., one root cause. Consequently, the degradation in a problem case can only be caused by a single root cause. Figure 6.5 shows and adaptation of recovery action proposal that considers multiple root causes. There is a probabilistic variable for each root cause with two states: present or absent. So, several root causes can be present. This requires the determination of the expected effect on each KPI over the Boolean states $t_1, \ldots, t_n$ of all n root causes, i.e., $\Pr(V_k \mid V_{T_1} = t_1, \ldots, V_{T_n} = t_n, \mathbf{x}(c,k), a, \mathbf{x})$ which is considerably more complex than the probability distribution $\Pr(V_k \mid V_T = t, \mathbf{x}(c,k), a, \mathbf{x})$. That is, instead of solely determining the effect of an action for each root cause, it is necessary to determine the action effect for each combination of present and absent root causes. Hence, instead of defining $|T|$ effects, it is necessary to define $2^{|T|}$ effects. Explicitly capturing this knowledge in the recovery model is barely possible since the extensive knowledge acquisition would overwhelm a human operator. This is similar to the problem of multiple fault assumption in root cause diagnosis presented in Chapter 6.3.2. Aligned with this discussion, the following presentation is based on the single fault assumption.

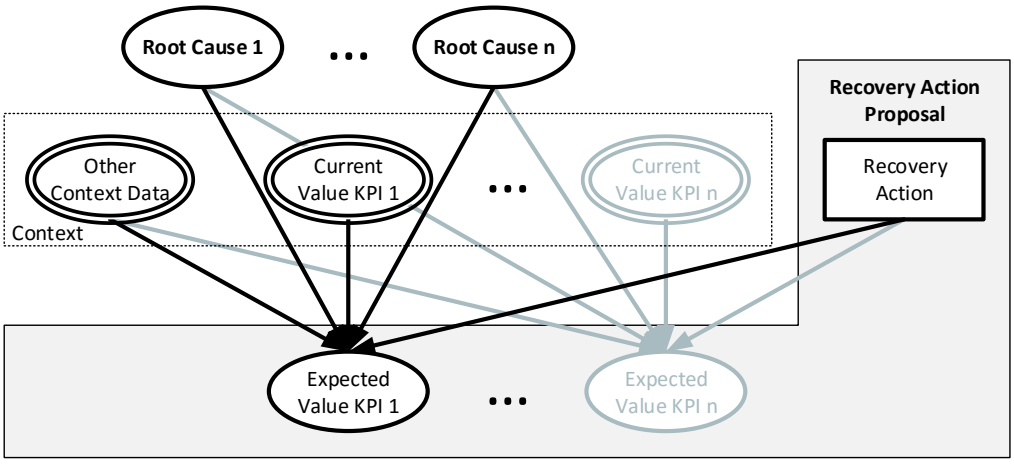


Figure 6.5: Influence diagram formalizing the multiple failure recovery action proposal.

6.3.3.3 Recovery Action Proposal

Recovery action proposal determines possible countermeasures for a cell diagnosis and estimates the effect of each action on the KPIs.

Recovery Model ODSO self-healing is based on the assumption that, given an operational context and a possible root cause, the operator can provide a set of possible recovery actions, each with an expected effect and the action effectiveness. In the conceptual view in Chapter 6.3.3.2, this has been formalized that the operator provides the conditional probabilities $\Pr(V_k \mid V_T = t, \mathbf{x}(c,k), a, \mathbf{x})$ for each KPI. For the implementation, this is formalized in parallel to the KPI effects as a probability distribution function.

Definition 6.3 (Recovery Model)**.** The recovery model is a mapping

$$\mathrm{RM} : C \times T \times \mathbf{X} \times (C \times S \to \mathbf{F}^{\perp}) \to \mathcal{P}(A \times \mathbf{F}^{\perp} \times [0,1]).$$

That is, based on the operational context $\mathbf{x} \in \mathbf{X}$, which comprises the current KPI values, and the SON function effects $\mathrm{SFE} : C \times S \to \mathbf{F}^{\perp}$ by SON management (see Definition 4.11), the recovery model defines a set of tuples $(a, \mathbf{f}^{\perp}, \rho) \in \mathcal{P}(A \times \mathbf{F}^{\perp} \times [0,1])$, each defining a recovery action a, its expected partial effect $\mathbf{f}^{\perp}$, and the probability ρ that a will be effective, for a given, assumed root cause $t \in T$ in the network cell $c \in C$.

Although the recovery models does look complicated at first sight, it simply represents the knowledge of the operator which action resolves which root causes as well as how likely they are to be effective. Various formalisms can be used to represent the recovery model, e.g., logic-based, case-based, or decision tree-based approaches. Similar to the previous components (see Chapter 4.3.2, Chapter 5.3.1.2, and Chapter 5.3.1.3), we propose an action rule-based approach here. This allows the simplified acquisition of the information for building the recovery model from automated, typically rule-based systems which are already in place, troubleshooting handbooks, or expert knowledge. The rules that make up the recovery model have the generic form:

```
ROOTCAUSE root cause IF condition(cell, context)
  THEN recovery action YIELDS effects WITH effectiveness
```

Thereby, each rule is proposing exactly one recovery action for a single root cause for a condition condition(cell, context) on the diagnosed cell and the current operational context, and estimates the partial effects and effectiveness of this action. However, there might be several rules for the same root cause, hence, a root cause can be recovered by several actions and an action can treat multiple root causes. Nevertheless, there must be at most one defined effect for a recovery action and a root cause.

The estimated partial effects that the execution of recovery action yields allows evaluating the action against the operator objectives. The effects are defined as probability distributions as described in Chapter 3.4.1.2. Although it might be

possible to leverage existing troubleshooting knowledge for determining the actions that may resolve a root cause, the estimation of the effects on the KPIs is still difficult. However, the operator can draw on the SON function effects SFE for the cell c which contains the expected effects of the SON functions with the current SON configuration as described in Chapter 4.3.6. In order to simplify the model elicitation, it is possible to consider two basic cases of effects:

Restoration of a cell's performance is based on the assumption that the current KPI values in the cell are solely caused by the detected problem. As a result, the recovery action, that resolves the problem, is expected to restore the cell performance to normal which is determined by the SON configuration for that cell. Thus, the effect can be determined as the combination of the effects of all SON functions given by SFE, i.e.,

$$\begin{aligned} &\mathrm{SFE}(c, *) = \mathbf{f}^{\perp} \text{ such that for all KPIs } k \in K \\ &\mathbf{f}^{\perp}(k) = \begin{cases} f_k & \text{if there exists a } f_k \in \{\mathrm{SFE}(c, s; k) \mid s \in S\} \\ & \qquad\qquad \text{such that } f_k \neq \perp \\ \perp & \text{otherwise.} \end{cases} \end{aligned} \tag{6.2}$$

Note that the defined, non-$\perp$ partial KPI effect of the SON functions all are equal, i.e.,

$$\begin{aligned} \forall s_1, s_2 \in S, k \in K.\, (\mathrm{SFE}(c, s_1; k) \neq \perp \wedge \mathrm{SFE}(c, s_2; k) \neq \perp) \\ \implies \mathrm{SFE}(c, s_1; k) = \mathrm{SFE}(c, s_2; k). \end{aligned} \tag{6.3}$$

This case is focusing on traditional recovery actions, e.g., automatic restart of a BS or restoration of an old software version.

Optimization of the cell performance assumes an action optimizes specific KPIs that are known by the operator. Of course, not all KPIs must be impacted, i.e., some partial KPI effect $f_k^{\perp}$ can be $\perp$. This case is focusing on triggering SON functions by SON self-healing. Hence, the treated issue is typically an unnoticed performance problem that can be overcome by the execution of a SON function. So, the effects of the action can be directly taken from SFE. i.e., the expected effects of triggering the SON function s are the SON function effects $\mathrm{SFE}(c, s)$.

The effectiveness of the action is the probability ρ that the action will produce effects given the root cause under condition(cell, context). In other words, it is the probability that the action will recover the problem with the given root cause, i.e., $\rho = \Pr(\text{effects will materialize} \mid \text{root cause, condition(cell, context), action})$. Consequently, with the probability $1 - \rho$ the effects will not materialize, hence, the cell's KPI values will stay the same. The following three rules are examples for this idea:

- `ROOTCAUSE sleeping_cell IF context(time) = night THEN restartBs(cell) YIELDS sfe(cell, *) WITH 0.5`

states that a sleeping cell cell might be recovered by restarting the BS. However, due to the expected impact on the network, this shall only be considered during low-traffic at night. The restart of a BS is a typical restoration action, i.e., the operator cannot explicitly define the impacted KPIs. Instead, the restart is expected to recover the normal performance of the cell which is given by the SON function effects sfe(cell , *) representing SFE(cell, ∗). However, a restart might not recover all sleeping cells and, hence, it is expected to be effective in only 50% of the cases.

- `ROOTCAUSE coverage_problem IF true THEN cco(cell) YIELDS sfe(cell, cco) WITH 0.7`

  proposes the execution of the CCO function if a coverage hole has been identified regardless of the operational context. The effect of a CCO action can be easily predicted using the SON function effects as sfe(cell , cco), i.e., SFE(cell, CCO). Since it is not clear what caused the coverage hole, e.g., a new construction site, the CCO might not be able to close the hole and, hence, it is expected to be effective in only 70% of the cases.

- `ROOTCAUSE * IF true THEN troubleTicket(cell) YIELDS sfe(cell, *) WITH 0.9`

  is the creation of a trouble ticket for any problem with any root cause. It is expected to recover the normal cell performance almost certainly with an effectiveness of 90%. This rule should be always present in the recovery model as the escalation rule if SON self-healing cannot recover a problem.

Recovery Actions and Effects For a cell diagnosis (c, ρ_T), recovery action proposal can determine the expected complete effect $\mathbf{f}$ of a recovery action a based on the recovery model RM. As shown in Equation 6.1, the idea is that the action effects for one root cause can be probabilistically combined to the action effects for all root causes by calculating the sum over the effects for each possible root cause weighted with its probability. Therefore, the partial effects defined by the recovery model must first be turned into complete effects using the merging function μ described in Definition 3.11. As outlined in Chapter 6.3.3.2, the KPIs are considered independent of each other. Consequently, the following description of the effect combination can focus on each single KPI effect first.

Definition 6.4 (Complete KPI effect for an action given a root cause)**.** Given the operational context $\mathbf{x} \in \mathbf{X}$, the diagnosed cell $c \in C$, and the recovery model RM, the complete KPI effects $f_{k,a,t} \in F_k$ on the KPI $k \in K$ of an recovery action $a \in A$ for each root cause $t \in T$ is defined as

$$f_{k,a,t}(v) = \mu_k(\mathbf{f}^{\perp}(k), c, \mathbf{x}, \rho)(v)$$

with $(a, \mathbf{f}^{\perp}, \rho) \in \mathrm{RM}(c, t, \mathbf{x}, \mathrm{SFE})$ being the expected partial effect of a for t as defined in the recovery model.

This probabilistic merging of the partial KPI effect of a with the current KPI value and can be interpreted as the resulting expected complete KPI effect if t is the true root cause of the problem. Based on this definition, the effects can be combined for all root causes.

Definition 6.5 (Combined complete KPI effect for an action)**.** The combined complete KPI effect $f_{k,a} \in F_k$ on the KPI $k \in K$ of a recovery action $a \in A$ for the cell diagnosis (c, ρ_T) is defined based on the

$$f_{k,a}(v) = \sum_{t \in T} \rho_T(t) \cdot f_{k,a,t}(v)$$

with $f_{k,a,t}$ being the complete KPI effect for a given t as defined in Definition 6.4. Note that $\sum_{t \in T} \rho_T(t) = 1$.

In the end, the recovery action proposal determines the recovery actions and effects.

Definition 6.6 (Recovery actions and effects)**.** In the context of SON self-healing, the action-effect set as defined in Definition 3.10 is refined to the set of recovery actions and effects. The set of pairs of recovery actions a and their respective combined complete effects $\mathbf{f}$ for a cell diagnosis (c, ρ_T) is defined as

$$\begin{aligned} \mathrm{AF} = \{(a, \mathbf{f}) \mid & a \in A, \\ & \exists t \in T.\, \rho_T(t) > 0 \wedge (a, \cdot, \cdot) \in \mathrm{RM}(c, t, \mathbf{x}, \mathrm{SFE}), \\ & \forall k \in K.\, \mathbf{f}(k) = f_{k,a}\} \end{aligned}$$

with $f_{k,a}$ being the combined complete KPI effect on k for the recovery action a as defined in Definition 6.5. That is, AF contains an action-effect pair for every action that potentially recovers at least one of the possible root causes.

6.3.3.4 Recovery Action Valuation

Recovery action valuation evaluates the proposed recovery actions and effects, and sorts them according to the MNO's preferences. The preferences for an action consider, on the one hand, its expected effects and, on the other hand, the aversion to execute the action due to involved efforts or performance impacts.

Preferences for Recovery Actions The satisfaction of the MNO's preferences regarding a proposed action and effect $(a, \mathbf{f}) \in \mathrm{AF}$ can be evaluated based on the objective model by comparing the expected utility vector for the effect as shown in Chapter 3.4.4: an action a_1 with effect $\mathbf{f}_1$ and expected utility vector $\mathbf{u}_1$ is preferred to an action a_2 with effect $\mathbf{f}_2$ and expected utility vector $\mathbf{u}_2$ if and only if $\mathbf{u}_1 \geq_D \mathbf{u}_2$. However, this would neglect the efforts that are necessary to execute the recovery action. For instance, consider a recovery escalation action that creates a trouble ticket. Since the manual inspection is expected to recover all possible root causes,

this action will return the problematic cell into a satisfactory state with a high probability. Consequently, the expected effect satisfies the objectives quite well. As a result, the system will always create a trouble ticket for each cell diagnosis solely due to the high effectiveness of this action. Therefore, it is necessary to also consider a preference to execute an action.

In utility theory, the concept of action cost serves as an action-dependent, deterministic amount that represents the effort to execute an action. By comparing the action cost with the utility of the outcome, i.e., building the difference of utility and cost, the decision maker is able to capture both in one number. Consequently, the decision between actions is not based on the absolute expected utility vectors for their effects, but on their biased utility vectors as shown in Figure 6.4.

Definition 6.7 (Preference for recovery actions)**.** The preference regarding two recovery actions a_1 and a_2 depends on their respective biased utility vectors $\Delta\mathbf{u}_1$ and $\Delta\mathbf{u}_2$ which capture the expected performance improvement and the cost of the actions. Consequently,

$$a_1 \succsim a_2 \iff \Delta\mathbf{u}_1 \geq_D \Delta\mathbf{u}_2.$$

Unfortunately, it is not easy to define the cost of a recovery action for SON self-healing. If the utility would be a monetary measure, e.g., the actual revenue from the network, the action cost could be its actual monetary cost, e.g., the labor cost for handling a trouble ticket or the lost revenue due to an outage. However, the utility is an abstract preference measure defined over the KPI values. Hence, the monetary cost needs to be translated into this abstract measure using a mapping that describes how much money the improvement of KPIs by a specific degree is worth the MNO. Although it is possible, though not easy, to define this, the creation of the mapping goes beyond the scope of this thesis since such a mapping can be seen as the derivation of objectives from high-level goals as described in Chapter 3.1.1.

KPI-based Action Cost An alternative option is to describe the action cost based on KPI values. This approach is not unrealistic under the assumption that the technical personnel operating the network does not know the exact monetary costs of, e.g., the restart of a BS or a trouble ticket causing an in-depth failure inspection. Instead, it seems likely that they base their action selection on some desired differences in the expected improvements of a cell's performance for different recovery actions. For instance, a trouble ticket should be executed if the performance is severely degraded and no other action is expected to improve the performance acceptably. Such decision making may be formalized in the ODSO framework in two ways:

- The improvement of the expected utility $\mathbf{u} - \mathbf{u}_{\mathbf{x}_c}$, i.e., the difference between the expected utility of the action effect and the utility of the current cell performance (see Chapter 5.3.2.1), for the trouble ticket must be by a factor higher than the improvement by the restart. Thereby, the factors can be seen as weights for the actions. Consequently, an action a with the expected utility

vector $\mathbf{u}$ for the current utility $\mathbf{u}_{\mathbf{x}_c}$ and weight $n \in \mathbb{R}$ would have the biased expected utility

$$\Delta\mathbf{u} = n \cdot (\mathbf{u} - \mathbf{u}_{\mathbf{x}_c}), \tag{6.4}$$

where n is multiplied with the expected utility for every priority. The drawback of this approach is that an expensive action like a trouble ticket creation could be triggered even if the cell performance is not severely degraded just because other actions are expected to solely have small improvements.

- The improvement of the expected utility for the trouble ticket must be at least some fixed amount higher than the improvement by the restart. This results in the relative expected utility $\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}_{\mathbf{x}_c} - \mathbf{q}$ for an action a with the expected utility vector $\mathbf{u}$ for the current utility $\mathbf{u}_{\mathbf{x}_c}$ and action cost $\mathbf{q}$ for a. Since $\mathbf{u}_{\mathbf{x}_c}$ is equal for all proposed action for one problem as they are all focused on the same cell c, this can be simplified to:

  $$\Delta\mathbf{u} = \mathbf{u} - \mathbf{q}. \tag{6.5}$$

  This approach fits smoothly into MAUT: it can be seen as adding another attribute, i.e., the cost, to the attributes representing the KPIs whereby the weight of the cost is equal to the sum of the weights of the KPIs. Thereby, the cost is a deterministic value that does solely depend on the action and not the action outcome. In the concrete example, the cost of a trouble ticket should be higher than the cost of a restart, of course.

In the following, we concentrate on the latter approach for the inclusion of the operator's action preferences into recovery action valuation. On the one hand, this is motivated by the fact that it matches with the theoretical foundations of ODSO. On the other hand, it turns out that this representation together with the priority regions for the utility allow a quite comprehensible cost definition on a coarse scale.

Definition 6.8 (Biased utility vector)**.** The biased utility vector for an action a with the expected utility vector $\mathbf{u}$ and action cost $\mathbf{q}$ is

$$\Delta\mathbf{u} = \mathbf{u} - \mathbf{q}.$$

Definition 6.9 (Action cost vector)**.** An action cost vector $\mathbf{q} = (q_1, \ldots, q_{N_D})$ is a vector of the priority region-specific costs $q_d \in [0, 1]$ for all priority regions $d \in D$.

An action cost vector $\mathbf{q} = (q_1, q_2, q_3)$ reduces the expected utility of an action $\mathbf{u} = (u_1, u_2, u_3)$ for each priority d separately. It can be seen as a disadvantage of the action a compared to other actions, such that a needs to be better with respect to the expected utility, in order to be selected. Thereby, the expected utility represents the probability that a is effective for the cell diagnosis, as well as a's predicted effects on the KPIs. Still, it is difficult to determine specific values for the rather abstract concept cost.

In order to support the operators, we present a heuristic that enables the definition of the action cost on a coarse scale. It is based on the idea that the action costs, in principle, represent that an expensive action a should only be considered for execution if no other action is able to improve the cell's performance beyond a priority region d with certainty. Consequently, this also means that a should only be considered if the current cell performance is not better than d. For instance, if a trouble ticket should only be issued if no other action is certainly able to improve a cell's performance (from the unacceptable priority region) to at least the acceptable priority region then $d = 1$. The priority region-specific action costs of a, $q_{d'}$ for all priorities $d' > d$, are set to 1 which results in a biased utility for these priorities of $\text{proj}_{d'}(\Delta\mathbf{u}) \leq 0$. Consequently, if at least one other action a' is expected to improve the cell's performance beyond the priority region d with a probability of 1 and partially satisfies any d', then the biased utility for this priority region d' is greater 0 and a is less preferred then a'. In summary, the cost $\mathbf{q}_a = (q_1, q_2, q_3)$ for action a must satisfy: if a should only be considered if no other action is expected to fully satisfy the objectives of priority $d \in D$, then $q_{d'} = 1$ for all $d' > d$.

In order to exemplify this heuristic, consider cell c having a problem which might be recovered with one of the following actions: a_{TT} to escalate via a trouble ticket, a_{RC} to reset a cell, and a_{CCO} to trigger the CCO SON function. The creation of a trouble ticket should only be considered if no other action satisfies the unacceptable utility range, i.e., at least one KPI is expected to stay in the unacceptable range for all other actions. Hence, $\mathbf{q}_{\text{TT}} = (0, 1, 1)$. The negative impact on the network of a cell reset, i.e., the temporary cell outage, should be counted in by allowing its execution only if no other action satisfied the acceptable utility range, i.e., $\mathbf{q}_{\text{RC}} = (0, 0, 1)$. Finally, the execution of a SON function like MRO has a low impact on the network and, so, they should always be considered. Consequently, $\mathbf{q}_{\text{CCO}} = (0, 0, 0)$. Table 6.1 shows the resulting decisions made by objective-driven self-healing in different situations. Thereby, two possible root causes for the problem in c are considered: t_{CH} refers to a coverage hole that may be recovered by a_{CCO}, and t_{SC} refers to a sleeping cell problem that is supposed to be recovered by a_{RC}. Furthermore, a_{TT} is able to recover both problems. All actions have the same expected effect with a utility of $(1.00, 1.00, 0.33)$ and an effectiveness of 90%.

Based on this, Table 6.1 shows the utilities $\mathbf{u}$ and biased utilities $\Delta\mathbf{u}$ for the recovery actions for two cell diagnose, represented by the root cause probabilities $\rho_T(\cdot)$, and three initial performance states, represented by their utility. Thereby, the entries for a cell diagnosis and a performance state is sorted according to the biases utility such that the top action is the most preferred one and would be executed. The first cell diagnosis expresses a higher likelihood that a degradation is caused by t_{CH} and in the second case, t_{SC} is more likely. Three different initial performance states $\mathbf{u}_{\mathbf{x}_c}$ for c are considered: unacceptable, acceptable, and optimal. As can be seen, the utility of a_{TT} is always higher than for the other actions since the trouble ticket potentially resolves both root causes. The reason that it is not always selected is its cost which makes it desired only in the unacceptable cell state, and, in this case, it is always selected. Conversely, a_{CCO} is always selected in the optimal system state since the other actions are just not desired in these cases. In the acceptable

Table 6.1: Example for the decision making by SON self-healing considering the action costs.

Initial Utility $\mathbf{u}_{\mathbf{x}_c}$	Cell Diagnosis for cell c	
	$\rho_T(t_{\text{CH}}) = 0.7, \rho_T(t_{\text{SC}}) = 0.3$	$\rho_T(t_{\text{CH}}) = 0.3, \rho_T(t_{\text{SC}}) = 0.7$
$(0.77, 0.00, 0.00)$	a_{TT} $\mathbf{u} = (0.98, 0.90, 0.30)$, $\Delta\mathbf{u} = (0.98, -0.10, -0.70)$ a_{CCO} $\mathbf{u} = (0.92, 0.63, 0.21)$, $\Delta\mathbf{u} = (0.92, 0.63, 0.21)$ a_{RC} $\mathbf{u} = (0.83, 0.27, 0.09)$, $\Delta\mathbf{u} = (0.83, 0.27, -0.91)$	a_{TT} $\mathbf{u} = (0.98, 0.90, 0.30)$, $\Delta\mathbf{u} = (0.98, -0.10, -0.70)$ a_{RC} $\mathbf{u} = (0.92, 0.63, 0.21)$, $\Delta\mathbf{u} = (0.92, 0.63, -0.79)$ a_{CCO} $\mathbf{u} = (0.83, 0.27, 0.09)$, $\Delta\mathbf{u} = (0.83, 0.27, 0.09)$
$(1.00, 0.50, 0.00)$	a_{CCO} $\mathbf{u} = (1.00, 0.82, 0.21)$, $\Delta\mathbf{u} = (1.00, 0.82, 0.21)$ a_{RC} $\mathbf{u} = (1.00, 0.64, 0.09)$, $\Delta\mathbf{u} = (1.00, 0.64, -0.91)$ a_{TT} $\mathbf{u} = (1.00, 0.95, 0.30)$, $\Delta\mathbf{u} = (1.00, -0.05, -0.70)$	a_{RC} $\mathbf{u} = (1.00, 0.82, 0.21)$, $\Delta\mathbf{u} = (1.00, 0.82, -0.79)$ a_{CCO} $\mathbf{u} = (1.00, 0.64, 0.09)$, $\Delta\mathbf{u} = (1.00, 0.64, 0.09)$ a_{TT} $\mathbf{u} = (1.00, 0.95, 0.30)$, $\Delta\mathbf{u} = (1.00, -0.05, -0.70)$
$(1.00, 1.00, 0.23)$	a_{CCO} $\mathbf{u} = (1.00, 1.00, 0.30)$, $\Delta\mathbf{u} = (1.00, 1.00, 0.30)$ a_{RC} $\mathbf{u} = (1.00, 1.00, 0.25)$, $\Delta\mathbf{u} = (1.00, 1.00, -0.75)$ a_{TT} $\mathbf{u} = (1.00, 1.00, 0.32)$, $\Delta\mathbf{u} = (1.00, 0.00, -0.68)$	a_{CCO} $\mathbf{u} = (1.00, 1.00, 0.25)$, $\Delta\mathbf{u} = (1.00, 1.00, 0.25)$ a_{RC} $\mathbf{u} = (1.00, 1.00, 0.29)$, $\Delta\mathbf{u} = (1.00, 1.00, -0.71)$ a_{TT} $\mathbf{u} = (1.00, 1.00, 0.32)$, $\Delta\mathbf{u} = (1.00, 0.00, -0.68)$

performance state, the selection depends on the cell diagnosis: if a_{RC} is more likely to be effective since t_{SC} is more probable then it has a higher utility and biased utility than a_{CCO} and vice versa. This selection is performed since the cost for priority 2 is the same, i.e., 0, for both actions.

In parallel to the operator objectives, the costs of the operator need to be captured in a machine-readable cost model. Thereby, the cost might be context-dependent which allows expressing that a high-priority trouble ticket might be even less preferred at night than it is during the day. The cost model can be represented in rules just like the objective model (see Chapter 3.4.3.5).

Definition 6.10 (Cost Model)**.** The cost model $\text{CM} : A \times C \times \mathbf{X} \to ([0,1])_{d \in D}$ is a function that determines an action cost vector $\mathbf{q} \in ([0,1])_{d \in D}$ for a recovery action $a \in A$ for a given network cell $c \in C$ in a context $\mathbf{x} \in \mathbf{X}$.

Recovery Action List The result of recovery action valuation is a sorted list of the recovery actions and their effects according to Definition 6.7.

Definition 6.11 (Recovery action list)**.** The recovery action list RAL $= ((a_1, \mathbf{f}_1), (a_2, \mathbf{f}_2), (a_3, \mathbf{f}_3), \dots)$ is a sorted list of tuples of an recovery action a_i and the respective expected complete effect $\mathbf{f}_i$ for $i = 1, 2, 3, \dots$. The list is sorted according to the biased utility vectors of the actions in descending order, i.e., for $(a_i, \mathbf{f}_i) \in$ RAL with the biased utility vector $\Delta\mathbf{u}_i$ and $(a_j, \mathbf{f}_j) \in$ RAL with the biased utility vector $\Delta\mathbf{u}_j$ is holds $i \leq j \iff \Delta\mathbf{u}_i \geq_D \Delta\mathbf{u}_j$.

6.3.4 Recovery Enforcement

Recovery enforcement finally performs the actual recovery by executing recovery actions and evaluating their results. Thereby, the execution of the actions has to be aligned with SON coordination.

6.3.4.1 Execute-Evaluate Loop

In principle, recovery enforcement is an execute-evaluate loop that can be performed in two manners.

- The enforcement has some information on how to evaluate the results. Hence, it can consecutively go through the list of proposed actions:
  1. Take the first, i.e., most preferred, action-effect pair from the sorted action list, execute it, and evaluate the results.
  2. If the problem is resolved or an escalation action is triggered, recovery enforcement is finished.
  3. If the problem is still persistent, continue with Step 1.
- The enforcement has no information on how to evaluate the results. Therefore, enforcement simply executes the most preferred action. If the action is not an escalation action, then the evaluation of the results has to be performed as a repetition of the whole SON self-healing process: the problem needs to be detected, its root causes must be diagnosed, and possible recovery actions must be planned again. Thereby, it is possible that the root causes as well as the recovery action may differ from the initial process run, e.g., if multiple root causes had been present and one of them has been resolved. Since the degradation recovery does not consider already executed actions, recovery enforcement needs to keep track of the recovery process in order to not select an already executed action again.

6.3.4.2 Execution of Trigger Actions

Trigger actions comprise self-healing workflows or regular SON functions as presented in Chapter 6.3.3.1. In contrast to escalation actions, they directly affect a network cell: upon being triggered, they analyze the network and attempt to perform some configuration changes to the network cell under consideration. Of course, these changes have to be coordinated with the activity of normally running SON functions in the network. Specifically, the following issues need to be considered:

- A recovery action like reverting a cell's configuration has impact on all other SON functions running on that cell and maybe even beyond that. Hence, this self-healing workflow should be executed isolated from them. Therefore, all other SON functions in the cell either need to be turned off or their requests need to be ignored, i.e., blocked, by SON coordination. The same applies if a recovery action comprises the temporary blocking of the execution of a SON function.

- If several problems have been diagnosed close to each other, it might be necessary to coordinate these recovery actions if they impact each other, i.e., SON coordination needs to determine which problem to recover first. This should be done based on the expected performance improvements by the actions. SON self-healing does not itself request a configuration change at SON coordination but instead the triggered workflow or SON function. Hence, SON coordination needs to be informed to handle such requests with the SON self-healing priority.

The execution request by a recovery action should, in principle, be prioritized against regular SON function requests that aim to optimize the performance. Consequently, the recovery actions must always be preferred in SON function conflicts because SON self-healing resolves exceptional situations in which normally running SON functions are likely to be neither effective nor efficient: One the one hand, the SON functions are likely to misinterpret the situation and request the wrong changes since they have been conceived for a non-erroneous network. One the other hand, the effect predictions for the SON functions, which are the basis of objective-driven SON coordination (see Chapter 5.3.1.1), are likely to be inaccurate in the presence of a failure.

The prioritization of recovery actions independent of the expected utilities is actually an instance of a coordination constraint that can be enforced through the technical constraint resolution of SON coordination presented in Chapter 5.3.1.3. Similar to the example for operator-defined requests, a generic conflict resolution rule for self-healing may be added:

```
IF recovery_action(request_1) = true AND
   recovery_action(request_2) = false AND
   (request_1, request_2) in conflictSet
  THEN reject(request_2) PRIORITY 2
```

Here, request_1 is a SON function request by a recovery action that was triggered by SON self-healing and request_2 is a non-self-healing function request. If request_1 and request_2 are in conflict then request_2 should be rejected. Thereby, the rule is given an assumed priority of 2 which is lower than the operator priority 1. In order to evaluate the predicate recovery_action(), SON coordination is provided with a set of all triggered actions in each granularity period. Its idea is that any request_2 which is not send by a recovery action but is in conflict with a recovery action request_1 must be rejected.

However, if root cause diagnosis identified two or more problems close to each other, it may happen that the recovery actions for these problems are in conflict

with each other. Consequently, the coordination constraint does not apply in this case. Instead, these conflicts need to be resolved based on the severity of the degradations and the expected improvement of the network performance. In other words, conflicts among self-healing triggered recovery actions should be resolved like conflicts between regular SON function requests, i.e., based on the expected utility of the recovery actions. Therefore, SON self-healing enforcement provides SON coordination with the expected effects of the recovery actions as determined by the recovery action proposal. Consequently, given that the recovery action a with $(a, \mathbf{f}) \in \text{RAL}$ is triggered by recovery enforcement, then the effect estimation of the conflict detection and technical resolution (see Chapter 5.3.1.1) takes $\mathbf{f}$ as the estimation of the request corresponding to a. Later on, SON function request selection will base its decision on the expected utility improvement by $\mathbf{f}$. This approach has two advantages: on the one hand, SON coordination is not able to estimate the effects of self-healing workflows since they are not contained in the SON function effects provided by SON management (see Definition 4.11), and, on the other hand, the effect estimation using the recovery model is supposed to be more accurate in failure situations than the estimations by SON coordination. Figure 6.6 depicts the interaction between SON self-healing recovery enforcement, the recovery actions, and SON coordination in form of a sequence diagram.

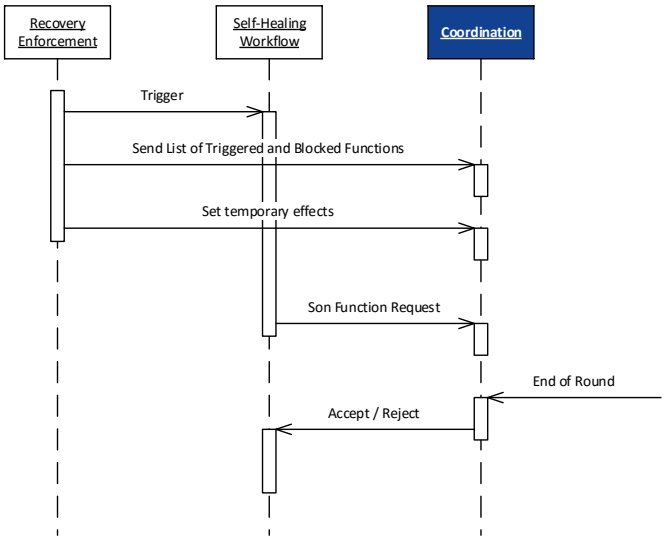


Figure 6.6: Sequence diagram of the interactions between SON self-healing enforcement, the recovery actions, and SON coordination

6.3.4.3 Execution of Escalation Actions

Escalation actions are non-automated recovery actions, e.g., the manual inspection of a problem by a human engineer. Therefore, recovery enforcement can solely request their execution, e.g., via issuing a trouble ticket, but cannot monitor their results. Therefore, they are final actions that end the execute-evaluate loop. Of

course, the requests need to provide further information about the problem, e.g., the cell diagnosis, which help the operators solving the degradation.

6.4 Related Work

This chapter presents related work to SON self-healing. It has already been noted that there is considerable work on degradation detection and root cause diagnosis. Actually, there are even sophisticated products beyond academic research available, e.g., [2op10][Dez14]. However, the ODSO self-healing approach concentrated on two different aspects. Hence, we present related work regarding the involvement of SON and degradation recovery in the following.

6.4.1 Involvement of SON

It is obvious that a complex, modular system like a SON is prone to unforeseeable problems at run time. Hence, it was early acknowledged that the execution of SON functions needs to be monitored in order to detect ineffective or undesired behavior. By the 3GPP [3GP13] and most of the related work, this is seen as a task of SON coordination.

The SON coordination concept developed in the SOCRATES project (see Chapter 3.6.2.2) was one of the first comprehensive ideas for SON coordination. Consequently, it also featured a function to detect undesired behavior of the SON functions, namely the guard function [Kür+10][Sch+11]. This undesired behavior comprises, among others, oscillations of the network configuration, unexpected KPI value combinations, and extreme KPI values. Furthermore, the authors particularly mention the detection of poor absolute performance of the network, i.e., not achieved objectives by the SON functions. Upon detection of a problem, the alignment function is triggered which analyzes the behavior and determines the erroneous SON function. As a recovery, it may undo previous changes of network parameters, block the execution of SON functions, or change its SFC. These recovery measures are then validated through the guard function by verifying that the detected problem disappears. However, as already described for the other ODSO components, the SOCRATES coordination is a rather abstract idea [Ban+11a]. Thus, it provides no hint how this functionality may be achieved. Nevertheless, the authors stress the importance of involving the SON in order to enable operators to gradually develop trust in automatic SON systems.

The UniverSelf project (see Chapter 3.6.2.3) developed a comprehensive vision for establishing operator trust in the operation of a SON [Cia+12][Cia+13]. Their framework distinguishes three levels of trust: trust in the reliable operation of a single SON function, trustworthy interworking of several SON functions, and seamless deployment of SON functions in a plug-and-play manner. The authors describe an extended policy-based approach in which an action policy, e.g., a set of ECA rules, defines the obligations of a SON function, i.e., what it should do, and the allowed actions that they may perform. This allows the operator do restrict the execution of SON functions to trustworthy behavior. In the ODSO approach, this concrete

control of the SON function behavior is seen as a task of either SON management for disabling the SON functions or SON coordination for rejecting the SON function requests during technical constraint resolution. In order to determine when a SON function is trustworthy or not, the abstract measure trust may be computed by monitoring the deviation of the actual KPI performance from the MNO goals, the frequency of network reconfigurations, and the duration of the continuous activation of a SON function. As can be seen, trust is directly related to the effectiveness of a SON function. Using reinforcement learning, the system is able to create a model of the expected trust of a SON function in different contexts. This model may be used by SON functions to improve their decision making, or to alert the UniverSelf coordination or governance block about a "situation where they are not able to fulfil the specified goals" [Tsa+13, p. 44]. In the latter case, the operational problem is delegated to higher-layer decision makers or a human operator. Eventually, this may lead to a new rule that blocks the erroneous SON function in the specific situation. In principle, this approach is closely related to the SON-based degradation detection: the execution of the SON functions is monitored and a detected ineffectiveness escalated.

In [RSB13], the authors present a coordination approach with dynamic priorities which prevents the monopolization of the network by a SON function, i.e., that a SON function with a high priority permanently issues requests which block other functions. However, the reason for this abnormal behavior of the function is not particularly analyzed and, consequently, specific countermeasures like triggering other SON functions not proposed. Instead, the authors simply block the SON function in order to allow another SON function to be accepted and, hopefully, resolve the problem that caused the ineffectiveness.

The authors of [Iac+15] analyze the execution of SON functions and detect, through a simple threshold, whether a SON is able to achieve the MNO's goals. If fails to do so, e.g., due to oscillations, a Bayesian net is used to determine the root cause of this ineffectiveness. Since the authors assume the cause to be an undetected conflict between concurrently executed SON functions, the diagnosis approach creates a new SON function conflict that may be incorporated into SON coordination.

All of these approaches have identified the problem that SON functions may become ineffective in specific operational situations and that this must be overcome, typically by blocking the execution of the problematic function. This shows that they see an uncommon, ineffective behavior of a SON function as the symptom of a problem in the SON itself. In contrast to that, the ODSO approach considers the possibility that such a behavior may also be caused by SON-external factors, specifically failures in the network. This is the reason why ODSO considers problems of the SON in SON self-healing instead of SON coordination.

6.4.2 Degradation Recovery

Decision theory provides a solid foundation for failure recovery. Consequently, there is a comprehensive body of research investigating the properties of decision-theoretic

troubleshooting approaches. Thereby, most of the approaches are based on a probabilistic diagnosis of a problem and aim at determining a sequence of repair actions, each recovering a specific root cause, such that the overall cost of the actions is minimized. For instance, the early work [HBR95] presents a simple decision criteria for ordering the recovery actions that has been derived from the analysis of decision trees. [SJK00] shows how this concept may be applied to troubleshooting printers. Both approaches are based on a simple cost measure and do not consider, e.g., the severity of the failure, which might limit the acceptable actions. In fact, the applicability of actions cannot be constrained in any way. Furthermore, an action is assumed to solely recover one possible problem, in fact, all example actions are the repair of different parts of the failed system. However, these approaches determine a sequence of recovery actions and observations for an uncertain diagnosis result, which minimizes the overall expected cost. This is referred to as a sequential decision process. In contrast to one-shot decision processes, they do not solely consider the utility of the current action but also include subsequent actions. Therefore, it is typically assumed that the system stays constant which is, however, barely the case in dynamic mobile networks.

The authors of [Lit+03][RJW10], present a more general approach based on partial observable Markov decision processes. They are also based on decision-theory, however, the chosen representation and reasoning approach allows for a more complex system description with a dynamic environment and actions that recover multiple faults. Again, they determine a sequence of recovery actions but do not consider the severity of the problems. Unfortunately, these approaches, though more expressive, are typically computationally much more complex than the former ones. This makes them often not usable in an on-line, reactive fashion.

Apart from such generic troubleshooting approaches, there is actually little related work regarding degradation recovery specifically in SON. Most of the research entitled with self-healing or alike actually treats either some self-configuration or self-optimization function, or the algorithm of some self-healing workflow that recovers a specific failure. In the former category, there are [TSA10] presenting inter-cell interference coordination, [Tiw+10] showing an MRO/MLB-related approach, and [Bal+08] with an automatic neighbor relationship setup (see Chapter 3.6.1.2 for an analysis of the general decision procedure). A member of the latter category is, for instance, the COC approach presented by the SOCRATES project (see Chapter 3.6.2.2) [Ami+09][Kür+10]. Thereby the compensation of broken cells is considering different, operator defined targets which need to be weighted using a cost function. Again, though, the available documentation provides no details about the implementation of this idea.

The authors of [Wil+04] are one of the few and among the first to acknowledge the need for rational degradation recovery based on operator objectives. They discuss the selection of recovery actions based on root cause probabilities and action costs, thereby, referring to the above mentioned framework in [HBR95]. Unfortunately, the presentation does not go into much more detail and does not provide solutions for the above mentioned problems that might occur in the context of SON self-healing. Instead, this is still presented as a partially open question in a later paper [BLM12].

In the recent work [Zam+15], a comprehensive recommender system for networks is presented. It aims to analyze network data for known event patterns that indicate some problem and need to be mitigated by executing some action. Drawing on a library of known, contextual event pattern-action pairs, it matches the pairs to the monitored sequence and recommends the best matching actions to be executed. By recording the reactions of the operator, the system is also able to extend the library. The focus of the work, however, lies on the efficient analysis of extensive data streams in order to detect the patterns, i.e., complex event processing. In contrast to the ODSO approach, this system can be seen as a fuzzy action policy system which solely replays what its rules, i.e., the event pattern-action pairs, say.

7 Evaluation

This chapter presents a qualitative evaluation of the ODSO approach. The focus lies on showing the feasibility of objective-driven SON management, coordination, and self-healing by simulating specific network scenarios for each operational task using an LTE network simulator. The analysis of the results reveals the complexity of the decision making and shows that ODSO enables autonomic steering the network through operational objectives in order to improve the network performance.

7.1 Scope and Approach Overview

The evaluation of ODSO is performed by analyzing each operational task separately. The reason for this is that their focus differs considerably: SON management aims to control the overall, long-term network performance, SON coordination focuses on specific, short-term settings in which SON functions compete with each other, and SON self-healing is focused on dedicated failure situations. So, a scenario which highlights the advantages of, e.g., objective-driven SON coordination does not necessarily show interesting behavior of, e.g., SON management. In order to provide some comparability, we use the same network simulation for all tasks, however, the concrete scenario for each operational task is adapted in order to highlight the specific effects of the respective objective-driven decision making.

In order to clarify the scope of the evaluation, notice that this chapter does not provide an in-depth quantitative evaluation of the network performance that can be achieved through ODSO. On the one hand, the achieved network performance is largely dependent on the correct technical models, specifically the SON function models. Hence, such an evaluation mainly assesses the accuracy of the elicitation approach for the technical models (see Chapter 4.3.2) which is not the focus of this thesis. On the other hand, the idea of ODSO is not to optimize the performance of the network, which is done by specialized SON functions, but to use these functions more efficiently with respect to the diverse operator objectives. This feature, however, is shown in the following evaluation.

Another valid assessment of the ODSO approach is to empirically measure the gained advantages in SON operations to the operational personnel. This could be done in terms of, e.g., a simplification of SON management leads to less configuration efforts, or an extended automation of SON coordination and self-healing leads to fewer manual interventions. Such an evaluation would require the deployment of the ODSO system in live networks and, subsequently, interviewing the MNOs. This,

however, is barely possible since an operational network and the time of human operators is crucial for an MNO. Nevertheless, this chapter still provides a both valid and viable evaluation of the ODSO system.

7.2 Simulation Setup

This section describes interesting features of the simulation setup which comprises a system-level network simulator, a set of SON functions, and a prototypical implementation of the three ODSO components. The simulation system is an extension of the experimental SON systems presented in [BR12] and [TSC14].

Figure 7.1 shows the static structure of the simulation environment which closely resembles the vision depicted in Figure 3.7. The network simulator computes the behavior of a 3GPP LTE network and provides PM data, i.e., KPI values, to the SON function engine. This engine executes the SON functions and deploys the results of their algorithms, i.e., a new network configuration, back to the network simulator. The algorithm of a SON function is controlled by the SON management component which configures it prior to its execution. Additionally, SON management provides the current SON function effects and the objective model to the other two components. The SON functions send network configuration change requests to the SON coordination component, which accepts or rejects them. Depending on the SON coordination result, the SON functions finally deploy the changes to the network. Finally, SON self-healing monitors the activity of the SON functions and PM data and may trigger the execution of SON functions and enforce their acceptance at the SON coordinator.

Figure 7.2 depicts the execution sequence of the different simulation components. As can be seen, this closely resembles the conceptual, synchronous execution of the ODSO architecture along granularity periods shown in Figure 3.10. After each granularity period, the network simulator provides the SON function engine with the collected PM data, i.e., KPI values collected during the last granularity period. Every time the SON function engine receives new PM data, it performs four steps:

1. It triggers the SON management component for all cells, which configures the SON functions according to the applicable operational objectives and provides SON coordination and SON self-healing with the necessary models.

2. It synchronously triggers the execution of each SON function for all cells, which may send a SON function request to SON coordination.

3. It triggers SON self-healing, which detects problems in the PM data and the SON function activity and may execute corresponding countermeasures, i.e., triggers a SON function and enforces its execution at the SON coordinator.

4. It triggers SON coordination which resolves the conflicts in the collected requests and sends replies to the SON functions. If a request is accepted, the function changes the network configuration in the SON function engine which propagates it to the network simulator.

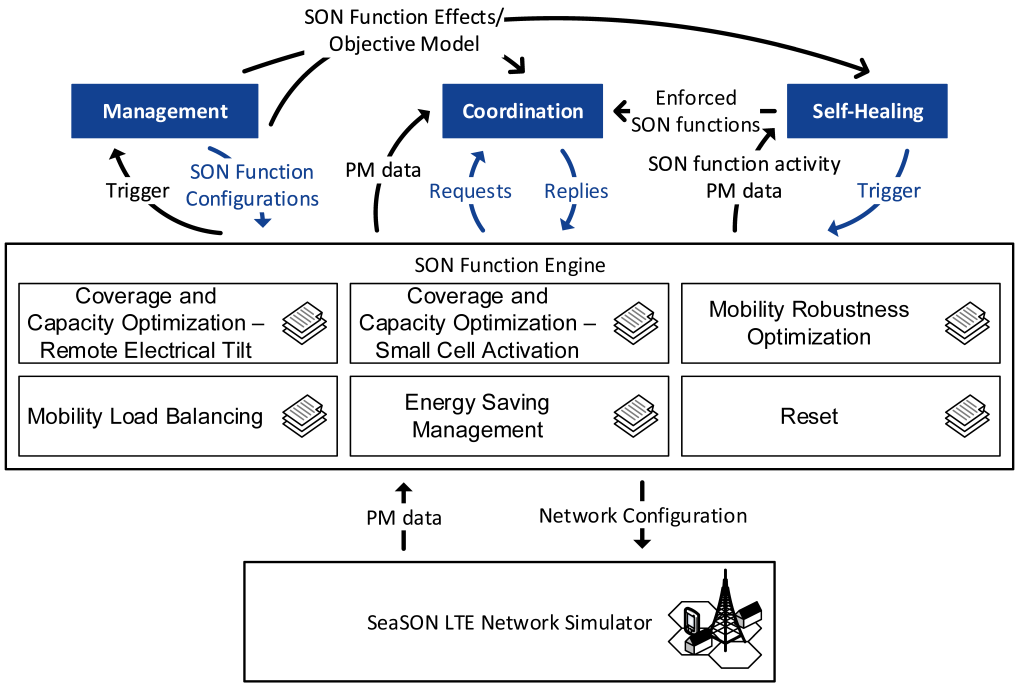


Figure 7.1: Overview of the components of the simulation system.

7.2.1 LTE Network Simulator

The evaluation uses a state-of-the-art, system-level LTE network simulator by Nokia Networks called System Experience of Advanced SON (SEASON) [Nok09]. It simulates an urban LTE network in the city center of Helsinki, Finland. Figure 7.3 depicts a screenshot of the network from the simulator and Table 7.1 specifies the simulation details.

As can be seen, the urban HetNet consists of 12 BSs that are spanning up 32 LTE macro cells in an area of roughly 50 km^2. The initial configuration of the network cells, e.g., the RET, is manually optimized to provide a good network performance. Three additional small cells, namely Cell 33, Cell 34, and Cell 35, are located within the coverage area of the macro Cell 6 in order to simulate a HetNet. As described in Definition 3.2, these small cells are assigned to the macro Cell 6 and will be used in the SON management scenario (see Chapter 7.3.1). We consider 1000, initially uniformly distributed users that are randomly walking in a predefined street pattern with 6 km/h, thereby using their UEs with a Constant Bit Rate (CBR) of 128 kbps, e.g., for streaming music. In the SON management (see Chapter 7.3.1) and the SON coordination scenario (see Chapter 7.4.1), we additionally consider 60 users with 128 kbps CBR in Cell 6 to emulate busy hours traffic, and, in the SON management scenario, 50 more users with 256 kbps CBR in a portion of Cell 6 to emulate a hot spot area, e.g., due to a demonstration. The granularity period, i.e., the time interval for collecting and providing the PM data, corresponds to 5400 simulated

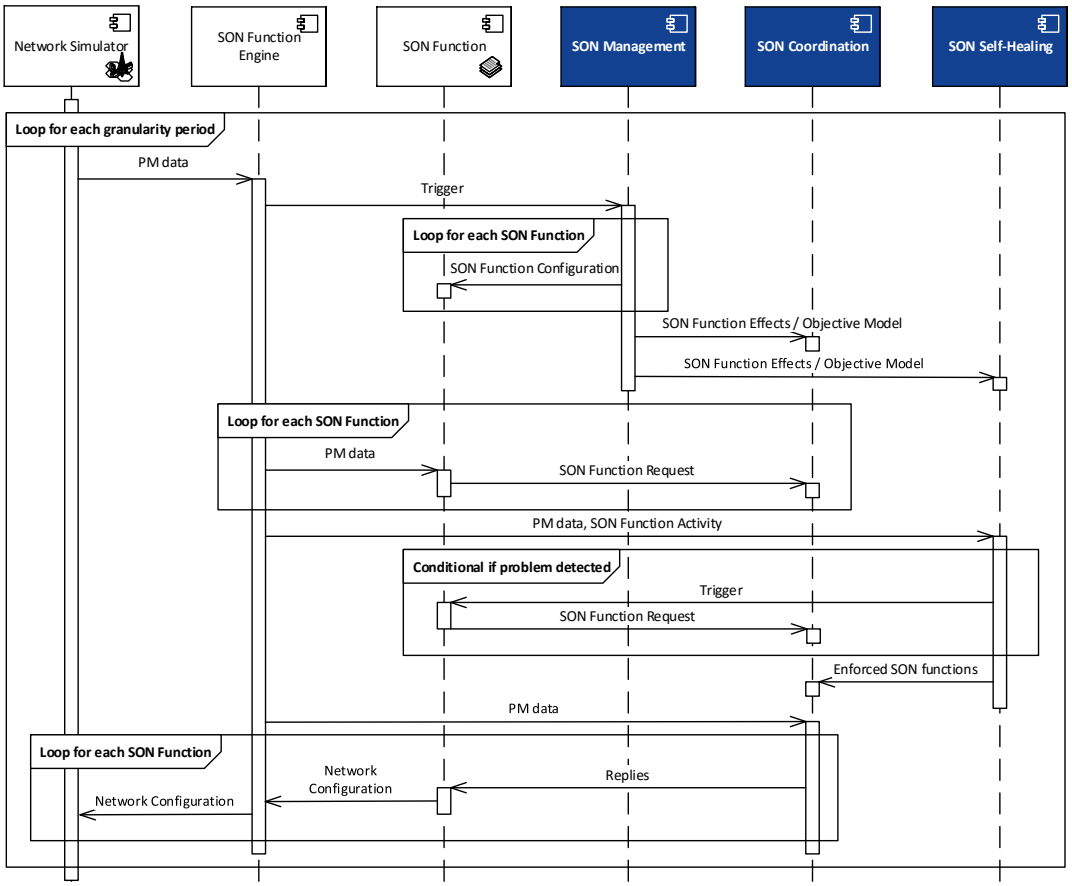


Figure 7.2: Overview of the execution sequence of the simulation system.

seconds, i.e., 1.5 hours.

7.2.2 SON Function Engine

The SON function engine is a run-time environment for SON functions written in Java. It has two major features: first, it represents the interface to the network simulator and manages all data exchange, and, second, it manages the life cycle of the SON functions.

For the evaluation, however, the employed SON functions are of particular interest. In the scenarios for the three ODSO components, we specify six SON functions based on the examples presented in Chapter 3.3.2:

Coverage and Capacity Optimization - Remote Electrical Tilt (CCO-RET) aims at improving the signal quality for the UEs of a network cell. It corresponds partially to the CCO function introduced in Chapter 3.3.2 as it optimizes the CQI of a macro cell by adapting its RET. The SFC of a CCO-RET instance $\mathsf{CCO{-}RET}(\mathsf{enabled}, \mathsf{threshold}, \mathsf{stepSize})$ comprises three parameters, namely a Boolean $\mathsf{enabled}$ whether it is enabled, an activation threshold

Table 7.1: Simulation parameters

Category	Parameter	Value
Radio	Carrier frequency	2 GHz
	Carrier bandwidth	20 MHz
	Number of PRBs	100
	Thermal noise	−114.447 dBm
	Shannon gap	−1.0 dBm
	Path loss model	UMTS 30.03 [3GP98, B.1.4.1.3]
	Other losses	20.0 dB
	Scheduler Model	CBR mode
	Shadowing correlation distance	50.0 m
	Shadowing standard deviation	8.0 dB
	Handover hysteresis threshold	2.0 dB
	RLF threshold	−6.0 dB
Network	Type	Urban HetNet
	Area	50 km^2
	Granularity Period	5400 s
	Macro cell layer	
	Number of cells	32
	Height	17 - 20 m
	Antenna gain	14 dB
	TXP	26 dB
	RET	0.0° - 3.5°
	Beam angles	65° horizontal, 9° vertical
	Small cell layer	
	Number of cells	3
	Positions	Macro Cell 6, see Figure 7.3
	Height	10 m
	Antenna gain	14 dB
	TXP	10 dB
	RET	10°
	Beam angles	360° horizontal, 9° vertical
Users	User mobility model	6 km/h random walk
	Regular users	
	Number and distribution	1500 random in whole network
	Data rate	128 kbps CBR
	Busy hours users	
	Number and distribution	60 in macro Cell 6, see Figure 7.8
	Data rate	128 kbps CBR
	Hot spot users	
	Number and distribution	50 in macro Cell 6, see Figure 7.8
	Data rate	256 kbps CBR

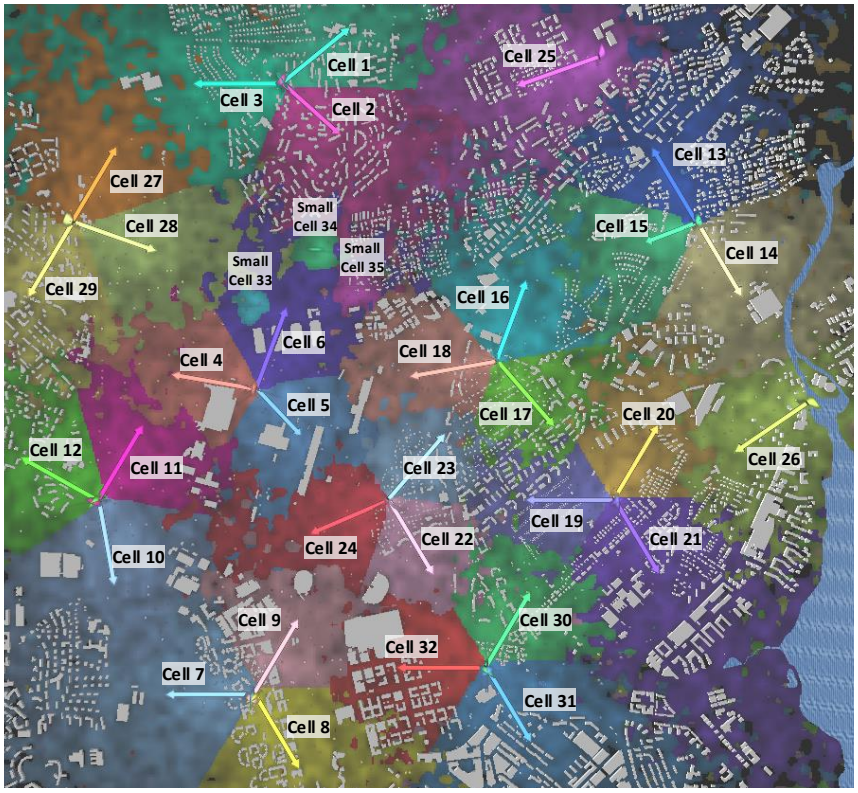


Figure 7.3: Screenshot from the network simulator showing the considered LTE network. The colored patches indicate the coverage areas and the arrows show the main sending direction of the network cells' antennas.

value threshold for the CQI, and a step size stepSize for the RET adaption. If the value of the CQI drops below threshold, then CCO-RET adapts the RET of the cell's antenna by stepSize, i.e., it tilts the beam up or down in a trial-and-error manner.

Coverage and Capacity Optimization - Small Cell Activation (CCO-SCA) aims at improving the capacity for the UEs of a network cell. It corresponds partially to the CCO function introduced in Chapter 3.3.2 as it optimizes the cell load of a macro cell by activating the small cells assigned to the macro cell. Therefore, it is solely executed on macro cells. Notice that this function has no effect on macro cells without associated small cells. The SFC of a CCO-SCA instance CCO−SCA(enabled, threshold) comprises two parameters, namely a Boolean enabled whether it is enabled, and an activation threshold value threshold on the macro cell's throughput. If the throughput rises above threshold, then CCO-SCA attempts to activate all assigned small cells such that they can take over some UE generated load. Note that the throughput is indirectly related to the cell load [Las+11].

Mobility Robustness Optimization (MRO) minimizes unnecessary handovers to

neighbor cells, specifically so-called ping-pongs, by adapting the CIO parameters. It corresponds to the MRO function introduced in Chapter 3.3.2. The SFC of an MRO instance MRO(enabled, threshold, stepSize) comprises three parameters, namely a Boolean enabled whether it is enabled, an activation threshold value threshold for the handover ping-pong rate, and a step size stepSize for the CIO adaptation. MRO monitors the KPI handover ping-pong rate for each neighbor relation individually and, if the ping-pong rate is above threshold, adjusts the virtual cell border by increasing the CIO to the problematic neighbor by stepSize. Thereby, one MRO instance may trigger several SON function requests as it request a change for each neighbor pair separately. Notice that the MRO function does also optimize the too-early and too-late handover rate [Las+11]. However, the simulated scenarios do not show such problems and, hence, these KPIs are omitted.

Mobility Load Balancing (MLB) attempts to reduce the load within a cell by adapting its CIO such that UEs are handed over to neighboring cells. It corresponds to the MLB function introduced in Chapter 3.3.2. The SFC of an MLB instance MLB(enabled, threshold, stepSize) comprises three parameters, namely a Boolean enabled whether it is enabled, an activation threshold value threshold for the load, and a step size stepSize for the CIO adaptation. Therefore, it monitors the cell load and, if it is above threshold, adjusts the CIO to all neighboring cells which are not overloaded by stepSize. Thereby the CIO is adapted symmetrically, i.e., the CIO from the overloaded cell to the neighbor is reduced and the CIO for the opposite direction is increased.

Energy Saving Management (ESM) is turning off small cells in order to save energy if they are not needed. This function corresponds to the ESM function introduced in Chapter 3.3.2. It is solely executed on macro cells and controls their assigned small cells. Consequently, ESM has no effect in a macro cell without any small cells. The SFC of an ESM instance ESM(enabled, threshold) comprises two parameters, namely a Boolean enabled whether it is enabled, and a threshold value threshold on the throughput of a small cell for turning it off. If the throughput of a small cell drops below threshold, then ESM deactivates this small cell and, thus, reduces the KPI energy consumption of the assigned macro cell (see Chapter 3.3.1.2). Note that the throughput is indirectly related to the cell load [Las+11].

Reset is not a continuously executed optimization SON function but an on-demand triggered self-healing workflow. Consequently, it does not monitor any KPI but solely resets and restarts the network cell for which it is triggered. Hence, there is no specific SFC for reset.

For each one of the 35 cells, one instance of the CCO-RET, MRO, and MLB function is created and can be configured independently of the others. The CCO-SCA and ESM functions are similarly instantiated for the 32 macro cells. Each SON function instance solely focuses on the optimization of the cell it is executed on. Thereby, depending on the configuration of the particular scenario, only a subset

of the SON functions might be enabled. These SON functions provide reasonable algorithms for solving the targeted problems. It is important to notice that an MNO may only get little more information about the SON functions than provided here. It is obvious that the MNO, so, is barely able to accurately predict the behavior of the SON functions.

The SON functions CCO-RET and MRO have been used before in other research work [BR12], whereas the others have been specifically developed for this thesis with inspiration from [Las+11], and from [Iac+14b] for the MLB function in particular. This is also the reason for the distribution of the CCO functionality over two SON functions CCO-RET and CCO-SCA: as CCO-RET was already available from previous work, we were able to simply reuse the function. This case may be common in real networks in the future: as SON functions are continuously modified, enhanced, or replaced, it is likely that the new functionality will not be deployed as new monolithic SON functions but as a bunch of small, focused SON functions. This allows the MNOs to buy and deploy the functionality they particularly need.

The development of the CCO-SCA and ESM functions revealed a specific feature of the ODSO concept that needs to be taken into account. The effects of an action, e.g., the configuration of a SON function, are supposed to apply solely to the network cell that the action is executed on. Particularly, an action effect cannot define an influence on neighboring cells of the action's cell. Suppose the ESM function would be executed on the small cells: besides the used energy of the small cell, the SFC of the function, i.e., whether the small cell is more aggressively turned off or not, has a huge impact on the KPI cell load of the respective macro cell. However, this cannot be modeled. For that reason, the ESM function has been developed such that the macro cell completely controls the small cells in a master-slave manner (see [Göt+15, Ch. 3.1.3] for an example involving this common approach for a different set of SON functions). In this way, the effect of a specific SFC for the macro cell ESM function can model the impact on the macro cell load as well as the additionally used energy of the small cells. Although this is an elegant solution, the consideration of side effects of actions could become an interesting future extension of the ODSO approach (see Chapter 8.2).

7.2.3 Objective-Driven SON Operations

The ODSO components have been implemented according to the presented concepts in Java. Therefore, we solely describe some interesting details of the implementation here.

In order to efficiently calculate the expected utility, all functions over KPI values, i.e., probability density functions f_k and utility functions $o_{k,d}$ for all KPIs $k \in K$ and priorities $d \in D$, are discretized. That is, the domain of each KPI is split into 1001 discrete bins and each function is transformed into a list of samples for all bins. For instance, the effect $f_k : \mathrm{Dom}(k) \to \mathbb{R}^+$ is represented as the list $\left(f_k\left({}^{0}/_{1000} \cdot |\mathrm{Dom}(k)|\right), f_k\left({}^{1}/_{1000} \cdot |\mathrm{Dom}(k)|\right), \ldots, f_k\left({}^{1000}/_{1000} \cdot |\mathrm{Dom}(k)|\right)\right)$, with $|\mathrm{Dom}(k)|$ being the cardinality of k. This representation enables an efficient calculation of integrals for the expected utility of a KPI effect.

The SON coordination component uses an efficient, publicly available solver named "Choco" [PFL14] for finding the optimal solution for the GP problem presented in Definition 5.12. Since it is preferably working with integer variables, the expected utilities of the SON function requests are transformed from real values $r \in \mathbb{R}$ to integer values $i \in \mathbb{Z}$ by $i = \text{round}(100 \cdot r)$.

In order to simplify the presentation of the scenarios and to ease comprehensibility of the behavior of the ODSO components, all KPI objectives are represented with linear utility that are derived from two thresholds, one for the acceptable region and one for the optimal region, as exemplified in Chapter 3.4.3.5. For instance, the objective for the KPI CQI $\mathsf{CQI} = \mathsf{linear}(0.6,\ 0.8)$ represents the KPI objective $\mathbf{o}_{\text{CQI}} = (\text{linear}(0.0, 0.6), \text{linear}(0.6, 0.8), \text{linear}(0.8, 5.5547))$ using Equation 3.4.

The evaluation considers the four KPIs introduced in Chapter 3.3.1.2 for the objectives, which are calculated by the network simulator and provided as part of the PM data:

Channel Quality Indicator (CQI) indicates the average signal quality in a cell.

Handover ping-pong rate (short form used in formula, code, and figures: Pipo) indicates the worst handover performance, i.e., highest handover ping-pong rate, of all neighbor relations in a cell. Note that the MRO function, however, does not monitor this KPI but the separated handover ping-pong rate for each neighbor relation.

Cell load (short form used in formula, code, and figures: Load) indicates the utilized capacity of the cell and, so, also the spare capacity for additional UEs.

Energy consumption (short form used in formula, code, and figures: Energy) indicates the additional energy consumption by active small cells in a macro cell's coverage area.

In order to plot the following graphs, we have defined a reduced form of the expected utility vector. Such a representation is very useful for comparison and plotting of the utility vector since only one number needs to be compared or drawn. However, this representation reduces the information of the utility vector and, thus, cannot be used for calculations with the utility vectors.

Definition 7.1 (Reduced form of expected utility vector)**.** A utility vector $\mathbf{u} = (u_1, u_2, u_3)$ can be reduced to a single dimension scalar in the domain $[0, 1]$ as

$$\langle \mathbf{u} \rangle_{\mathbf{u}} = \begin{cases} \frac{u_1+u_2+u_3}{3} & u_1 = 1.0 \wedge u_2 = 1.0 \\ \frac{u_1+u_2}{3} & u_1 = 1.0 \\ \frac{u_1}{3} & \text{otherwise.} \end{cases}$$

This reduction focuses on the utility of the highest, not fully satisfied priority region. Hence, $\langle \mathbf{u} \rangle_{\mathbf{u}} \in [0, 1/3]$ if the utility of the unacceptable region $u_1 < 1.0$, $\langle \mathbf{u} \rangle_{\mathbf{u}} \in [1/3, 2/3]$ if the utility of the unacceptable region $u_1 = 1.0$ and the utility of the acceptable region $u_2 < 1.0$, and $\langle \mathbf{u} \rangle_{\mathbf{u}} \in [2/3, 1]$ if the utility of the unacceptable

region $u_1 = 1.0$ and the utility of the acceptable region $u_2 = 1.0$. Considering the utility vectors $\mathbf{u}^A = (1.0, 1.0, 0.25)$, $\mathbf{u}^B = (1.0, 0.75, 0.42)$, $\mathbf{u}^C = (1.0, 0.95, 0.0)$, and $\mathbf{u}^D = (0.8, 0.42, 0.0)$ depicted in Figure 3.21, the respective reduced representations are $\langle \mathbf{u}^A \rangle_{\mathbf{u}} = 0.75$, $\langle \mathbf{u}^B \rangle_{\mathbf{u}} = 0.58$, $\langle \mathbf{u}^C \rangle_{\mathbf{u}} = 0.65$, and $\langle \mathbf{u}^D \rangle_{\mathbf{u}} = 0.27$.

7.2.4 Common Scenario

The simulations presented in the following chapters adopt a common scenario as the basis for their specific scenarios which are simulated for a specific number of granularity periods. This approach enables the comparison of the results from different simulations as the principle behavior of the network and the SON functions would lead to similar KPI values. Of course, this generic scenario is adapted for each simulation in order to show the specific ODSO task that should be highlighted.

The basic scenario comprises a single, macro cell, network layer, i.e., the three small cells shown in Figure 7.3 are turned off. Furthermore, only the regular user population described in Table 7.1 is active by default.

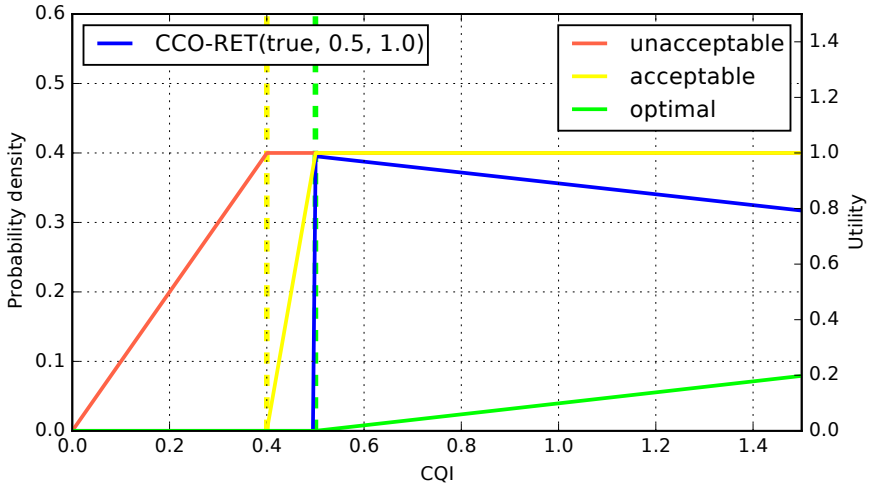
The three most common SON functions are deployed by default: CCO-RET, MRO, and MLB. The CCO-SCA and ESM functions are turned off by default, and the reset SON function does not need any configuration but is actively triggered externally. The basic SON function models for these functions defines one SFC with one KPI effect for each SON function (see Chapter 7.2.2 for an explanation of the SFCs):

```
IF true THEN CCO-RET(true, 0.5, 1.0)
  YIELDS effect(CQI, triangular(0.5, 0.5, 5.5547))
IF true THEN MRO(true, 0.1, 0.5)
  YIELDS effect(Pipo, triangular(0.0, 0.1, 0.1))
IF true THEN MLB(true, 0.6, 0.5)
  YIELDS effect(Load, triangular(0.0, 0.6, 0.6))
```

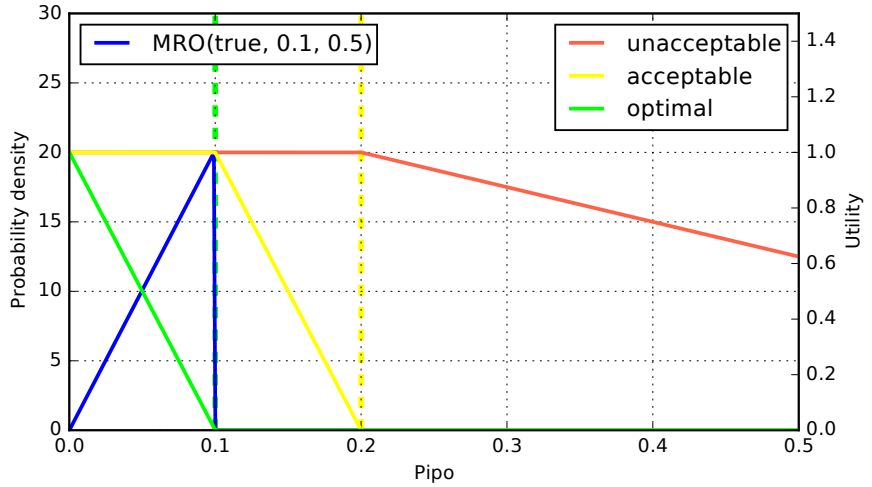
As we are the vendors of the SON functions, we have defined the SFCs based on our own expertise in the simulation scenario. The SON function effects were manually derived from the respective SFCs by inspecting the performance of the simulated network. Therefore, we used triangular KPI effects as explained in Chapter 4.3.2. The KPI effect probability distributions are depicted as the blue lines in Figure 7.4: CCO-RET is expected to solely affect the CQI and keep it above 0.5; MRO is expected to solely affect the handover ping-pong rate and keeps it below 0.1; and MLB is expected to solely affect the cell load and keeps it below 0.6. Of course, these are quite simple effects, e.g., one would typically expect that MLB also affects the handover performance. However, on the one hand, the elicitation of such complex models is not in the focus of this work (see [Göt+15] for some more complex models). As the SON function models propose only one SFC for each SON function, the whole network is configured uniformly.

The objective model defines operator objectives for the three KPIs that the default SON functions affect:

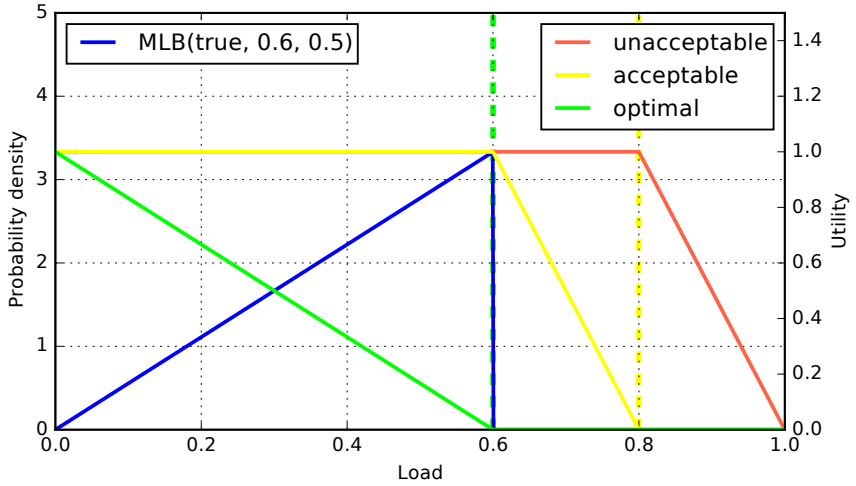
```
IF true THEN CQI  = linear(0.4, 0.5) WITH 0.33
             Pipo = linear(0.2, 0.1) WITH 0.33
             Load = linear(0.8, 0.6) WITH 0.33
```



(a) KPI CQI (not showing full domain $[0.0, 5.5547]$).



(b) KPI handover ping-pong rate (not showing full domain $[0.0, 1.0]$).



(c) KPI cell load.

Figure 7.4: KPI effects of the SFCs CCO−RET(true, 0.5, 1.0), MRO(true, 0.1, 0.5) and MLB(true, 0.6, 0.5), as well as the respective KPI objectives.

These KPI objectives are also visualized in Figure 7.4 as the red, yellow and green lines. Furthermore, the yellow and green dashed lines visualize the thresholds for the acceptable and optimal priority region. Thereby, the KPI objectives have the same weight and, thus, are equally important. The KPI objective for energy consumption, however, is weighted with 0 and, thus, ignored. Without any real world operator objectives available, this objective model has been derived from an inspection of the achievable performance in the simulated network. That is, the initially quite optimal configured network has been simulated and the resulting performance was taken as the lower bound for the optimal region. The acceptable region is then defined based on our expertise. Notice that more challenging objectives and more aggressive SFCs would lead to a lot of ineffective SON function activity without improving the performance due to the network dimensioning.

Figure 7.5 shows the results of a simulation for the default network configuration. The upper three graphs show the mean values of the KPIs CQI, handover ping-pong rate, and cell load as blue lines over 40 granularity periods. Additionally, the blue area indicates the value range over all cells, i.e., the top and bottom data points of the area represent the maximum and minimal KPI values within the granularity period. The dashed, yellow and green lines show the threshold for the acceptable and optimal region respectively. The bars represent the number of accepted SON function requests for the CCO-RET, MRO, and MLB function respectively. The scale for these bars is drawn on the right side of the graphs. The graph at the bottom shows the mean of the reduced utilities (see Definition 7.1) of the cells as well as the utility value range. In principle, the variations in the KPI values are caused by randomness in the simulation.

As can be seen, the default configuration is quite optimal and, so, the SON functions are barely active and do only occasionally send a single SON function request during the simulation. Actually, only MRO is sometimes active in the presented simulation. This is on purpose: by adapting this optimal setup, i.e., introducing specific problems, it is ensured that, in principle, the SON functions are able to improve the specific scenario configuration to some optimal configuration and do not face performance problems they cannot solve. This allows the evaluation to concentrate on the operations of the SON functions without much consideration of the specifics of their algorithms.

7.3 SON Management

The advantages of objective-driven SON management will be shown in a scenario that primarily focuses on different configurations of the CCO-SCA and ESM functions. It will show that the ODSO approach adapts the SON configuration to variable operator objectives and, thus, satisfies them more than a fixed SON configuration.

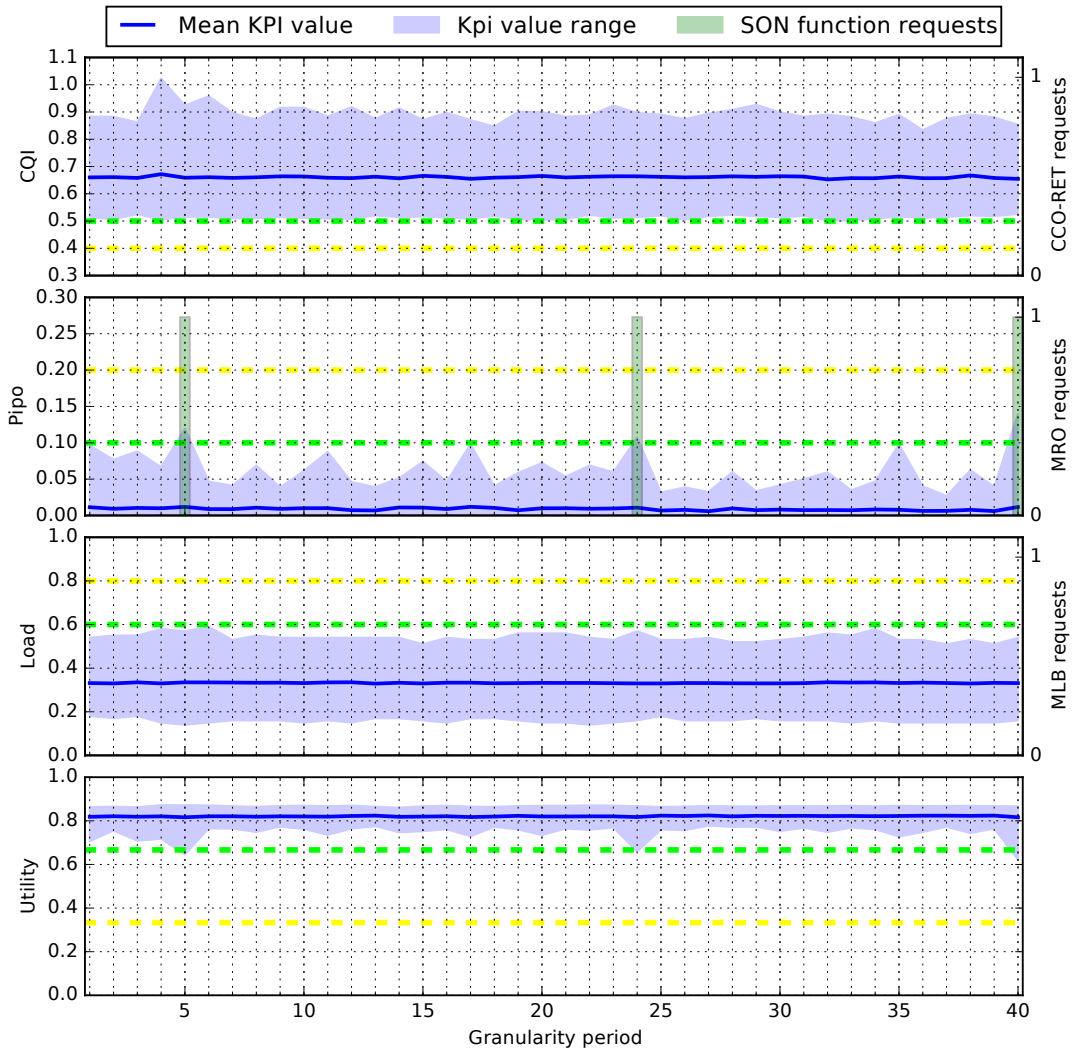


Figure 7.5: Simulation result of the common default scenario.

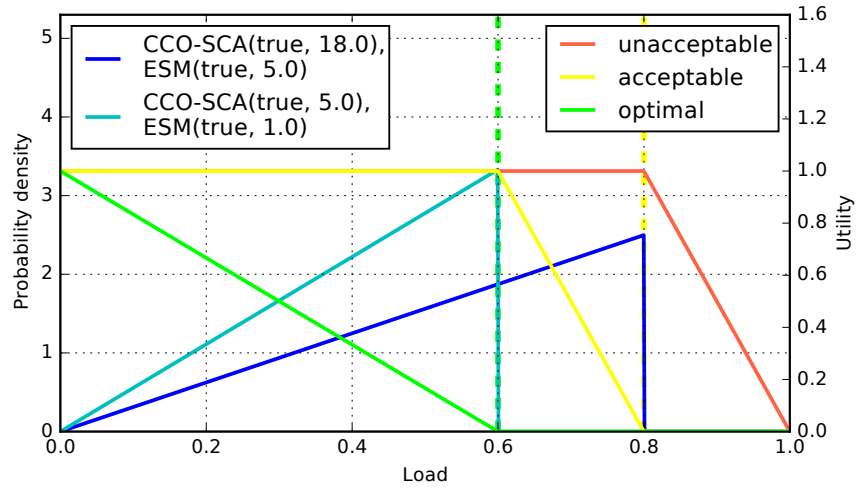
7.3.1 Scenario

The scenario is supposed to show the trade-off decision that objective-driven SON management makes for determining the SON configuration. Specifically, the two competing KPIs are energy consumption and cell load, the latter representing the user satisfaction with respect to the throughput offered by the mobile network. Both KPIs are controlled by the CCO-SCA and ESM functions based on their SFCs. On the one hand, if the CCO-SCA function is configured to aggressively reduce the cell load, then the small cells in a macro cell will be turned on very often, thus, increasing the energy consumption. Vice versa, a relaxed CCO-SCA function configuration will switch on the small cells seldom which yields a higher cell load and reduced energy consumption. On the other hand, if the ESM function is configured to aggressively save energy, then the small cells in a macro cell will be turned off more often in order to decrease the energy consumption. Of course, this increases the cell load leading to possibly more unsatisfied users. In contrast to that, a relaxed ESM configuration will cause the function to turn the small cells off less frequently. In that way, the small cells reduce the cell load of the macro cell more often, however, at the cost of a higher energy consumption. In the simulation, this is expressed in two SFCs for both the CCO-SCA and ESM SON function model (see Chapter 7.2.2 for an explanation of the SFCs):

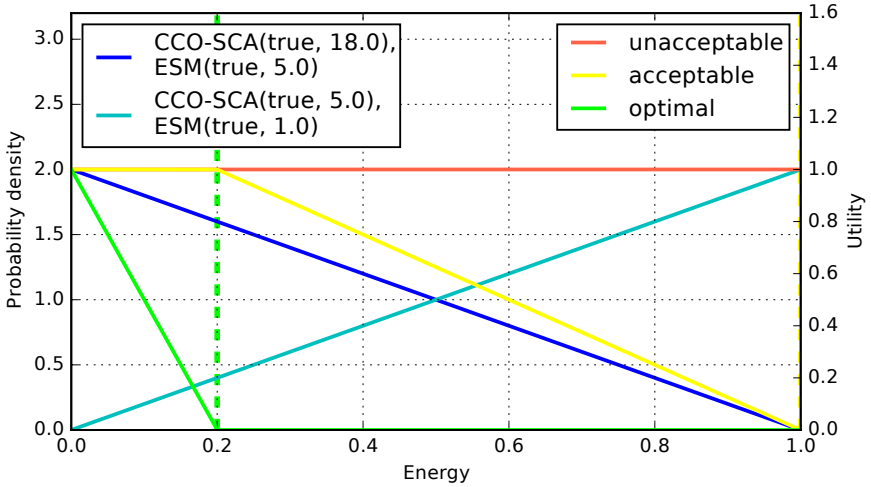
```
IF true THEN CCO-SCA(true, 5.0)
  YIELDS effect(load,   triangular(0.0, 0.6, 0.6)),
         effect(energy, triangular(0.0, 1.0, 1.0))
IF true THEN CCO-SCA(true, 18.0)
  YIELDS effect(load,   triangular(0.0, 0.8, 0.8)),
         effect(energy, triangular(0.0, 0.0, 1.0))

IF true THEN ESM(true, 5.0)
  YIELDS effect(load,   triangular(0.0, 0.8, 0.8)),
         effect(energy, triangular(0.0, 0.0, 1.0))
IF true THEN ESM(true, 1.0)
  YIELDS effect(load,   triangular(0.0, 0.6, 0.6)),
         effect(energy, triangular(0.0, 1.0, 1.0))
```

As before, we, as the vendors of the SON functions, have defined the SFCs and derived the SON function effects based on our expertise and inspection of the simulation scenario. Conceptually, both are expected to be provided by the vendors of the SON functions. The SON function model can be interpreted in the following way. The aggressive CCO-SCA SFC CCO−SCA(true, 5.0) is expected to keep the cell load below 0.6 and the energy consumption close to 1.0, i.e., all small cells are expected to be turned on. The relaxed CCO-SCA SFC CCO−SCA(true, 18.0) is expected to keep the cell load below 0.8 and the energy consumption close to 0.0, i.e., the small cells are expected to be turned off. The aggressive ESM SFC ESM(true, 5.0) is expected to keep the cell load below 0.8 and the energy consumption close to 0.0, i.e., only the macro cell is turned on. The relaxed ESM SFC ESM(true, 1.0) is expected to keep the cell load below 0.6 and the energy consumption close to 1.0, i.e., the macro cell and three small cells are turned on. Notice that these configurations are symmetric. The KPI effects of all four SFCs, which are defined using triangular

probability distributions, are shown as blue lines in Figure 7.6.



(a) KPI effects and KPI objective on cell load.



(b) KPI effects and KPI objective on energy consumption.

Figure 7.6: Visualization of KPI effects of the CCO-SCA SFCs CCO−SCA(true, 18.0) and CCO−SCA(true, 5.0), and ESM SFCs ESM(true, 1.0) and ESM(true, 5.0), as well as the KPI objectives on cell load and energy consumption.

Besides the CCO-SCA and ESM functions, there are also a CCO-RET and an MRO function deployed. However, both are barely active and have little impact on the network performance since the initial network configuration is already quite good. The MLB function is not activated since CCO-SCA is considered to reduce the cell load.

Typically, the focus of the MNO regarding the two KPIs is shifting during a day: In the busy business hours, it is of utmost importance to satisfy the bandwidth requirements of the users. Hence, it is very important to not only provide each UE with as much throughput as possible but also to keep radio resources free to quickly satisfy the needs of additional UEs. At night, however, fewer people are

actively using the network and, thus, the MNO does not want all cells to be active since this would be a waste of resources, especially in dense HetNets in urban areas. Consequently, the focus shifts towards saving energy.

The simulation will be executed for 40 granularity periods. The objective model defines the following objectives for the cell load and energy consumption:

```
IF granularity period in [1 - 6, 15 - 20, 29 - 34]
  THEN load   = linear(0.8, 0.6) WITH 0.48
       energy = linear(1.0, 0.2) WITH 0.05

IF granularity period in [7 - 14, 21 - 28, 35 - 40]
  THEN load   = linear(0.8, 0.6) WITH 0.05
       energy = linear(1.0, 0.2) WITH 0.48
```

As can be seen, the defined target values for the KPIs cell load and energy consumption are constant: the cell load has an acceptable threshold of 0.8 and an optimal threshold of 0.6, and the energy consumption has an acceptable threshold of 1.0 and an optimal threshold of 0.2. That means energy consumption is never considered unacceptable which resembles the common understanding of ESM as a non-critical feature. The respective objective functions are depicted as red, yellow and green lines in Figure 7.6. Furthermore, the yellow and green dashed lines visualize the thresholds for the acceptable and optimal priority region. However, the weight of the KPI objectives changes every 6 to 8 granularity periods in order to emulate the change in the operator's focus over time. Figure 7.7 depicts a radar chart of the KPI weights for the different time periods.

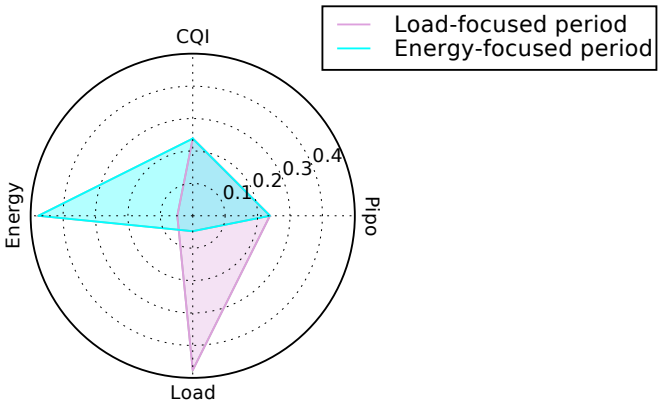


Figure 7.7: KPI weights during the cell load-focused and energy consumption-focused periods.

The scenario simulates the "eNB overlaid use case" [3GP14a, p. 23] in which a macro cell is constantly providing coverage whereas small cells may be switched on to provide additional capacity in busy areas. The considered network comprises only one macro cell that has some assigned small cells, namely Cell 6 with the small cells Cell 33, Cell 34, and Cell 35 as depicted in Figure 7.8. For that reason, the evaluation concentrates on showing the behavior of the SON in that particular area. Thereby, we also consider three different load situations in the network as depicted

in Figure 7.9. In the first 13 granularity periods, the cell load is produced solely by the regular user population, i.e., $^{1500}/_{32} \approx 47$ users in Cell 6 with 128 kbps CBR on average. From approximately granularity period 14 on, there are additionally 60 busy hours users in Cell 6 requiring 128 kbps CBR. Finally, in granularity period 28, we deploy an additional hot spot area in the Cell 6 consisting of 50 users requiring 256 kbps. Please refer to Table 7.1 for the details of the simulation setup.

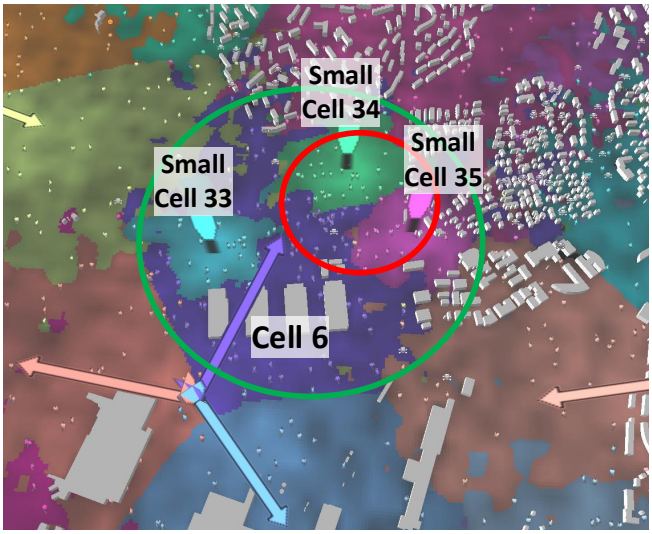


Figure 7.8: The considered Cell 6 and the area with the busy hour users (green) and the additional hot spot (red).

It is obvious that this setup is not intended to simulate realistic variations occurring in mobile network, i.e., the shift in traffic over the course of a day and week. Unfortunately, this simplification is necessary due to a limitation of the utilized LTE network simulator. Nevertheless, the proposed setup allows the simulation and analysis of different situations which may occur in a real network. Notably, there are six different situations of interest, shown as the colored areas in Figure 7.9:

RL is a situation with regular users (low cell load), and cell load-focused objectives.

RE is a situation with regular users (low cell load), and energy consumption-focused objectives.

BL is a situation with regular and busy hours users (medium cell load), and cell load-focused objectives.

BE is a situation with regular and busy hours users (medium cell load), and energy consumption-focused objectives.

HL is a situation with regular, busy hours and hot spot users (high cell load), and cell load-focused objectives.

HE is a situation with regular, busy hours and hot spot users (high cell load), and energy consumption-focused objectives.

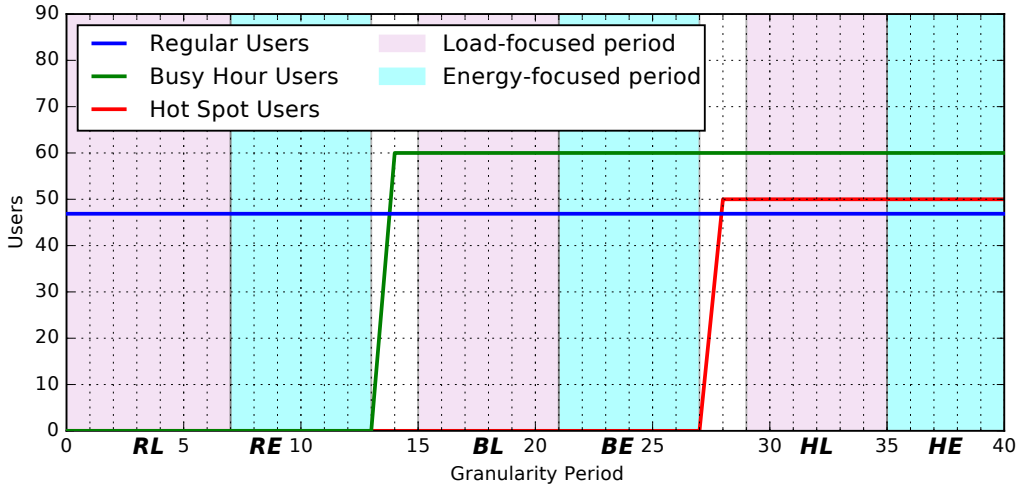


Figure 7.9: The course of the simulation: the lines visualize the number of UEs from the specific user group and the colored areas indicate the objective KPI the operator is focusing on.

Notice that the granularity periods 13-15 and 27-29 are not marked as an interesting scenario. The reason is that the LTE network simulator does not allow to automatically add the user groups. As a result, we may get inaccurate PM data during these periods when we add the users manually. Therefore, we decided to ignore these transition phases in the analysis of the results.

7.3.2 Results

Before analyzing the results of the simulation, it is worth taking a closer look at the CCO-SCA and ESM SON function models. Both functions are configured by thresholds on the measurement throughput. However, this is not an objective KPI for the operator. Thus, the SON function model needs to translate the configuration regarding the throughput thresholds into an expectation for the value of the cell load KPI. This translation is far from trivial since the achieved throughput just partially influences the cell load. Another important factor is, e.g., the average signal quality. Therefore, is can be expected that in reality, the SON function model is much more complex than the presented one, especially, the context description that determines the KPI effects might be more specific. In fact, the SON function model used here has been specifically computed for Cell 6 and, thus, intrinsically considers the actual context of that cell.

Notice that the SFCs are just one possible implementation of a configuration interface for a CCO-SCA or ESM function. For instance, it is just as likely that an ESM function is solely controlled trough a Boolean value which determines whether the function should be aggressive or not. This example shows even more that SON functions and their configuration can be non-transparent to the operator.

The CCO-SCA and ESM SON function models nicely outline a key advantage of the ODSO concept: the abstraction of the details of the configuration of SON

Table 7.2: The unweighted utilities per KPI and the KPI agreement for cell load and energy consumption of the possible SON configurations.

		Cell load	Energy consumption
$a_{\text{CCO-SCA,ESM}}$	utility	$0.63 = \langle(1.00, 0.88, 0.26)\rangle_{\mathbf{u}}$	$0.53 = \langle(1.00, 0.60, 0.10)\rangle_{\mathbf{u}}$
	agreement	0.56	1.00
$a_{\text{CCO-SCA},\overline{\text{ESM}}}$	utility	$0.78 = \langle(1.00, 1.00, 0.33)\rangle_{\mathbf{u}}$	$0.47 = \langle(1.00, 0.41, 0.01)\rangle_{\mathbf{u}}$
	agreement	1.00	1.00
$a_{\overline{\text{CCO-SCA}},\text{ESM}}$	utility	$0.59 = \langle(1.00, 0.77, 0.19)\rangle_{\mathbf{u}}$	$0.60 = \langle(1.00, 0.79, 0.19)\rangle_{\mathbf{u}}$
	agreement	1.00	1.00
$a_{\overline{\text{CCO-SCA}},\overline{\text{ESM}}}$	utility	$0.63 = \langle(1.00, 0.88, 0.26)\rangle_{\mathbf{u}}$	$0.53 = \langle(1.00, 0.60, 0.10)\rangle_{\mathbf{u}}$
	agreement	0.56	1.00

functions towards general objective KPIs. That is, the operator does not need to figure out how the objectives can be expressed in terms of the SON function configuration. Instead, this is provided by the vendors of SON functions who have the available resources and knowledge to create the respective models. So, the operational personnel can manage a SON through a harmonized and standardized interface in form of KPI objectives.

In SON management, the SON configuration is determined based on the operational objectives encoded in the objective model and the expected effects of the SFCs on the network performance as described in the SON function models. Consequently, SON management distinguishes just two situations in the given scenario: the periods in which the KPI objective cell load has a high weight and the periods in which the KPI objective energy consumption has a high weight. In other words, the actual network performance, i.e., the cell load situation, is not considered.

Given the SON function models, the management component needs to make a decision between four possible SON configurations. These differ solely in the configuration of the CCO-SCA and ESM functions, hence, we omit the SFCs for CCO-RET and MRO, namely CCO−RET(true, 0.5, 1.0) and MRO(true, 0.1, 0.5):

$$
\begin{aligned}
a_{\text{CCO-SCA,ESM}} &= (\mathsf{CCO{-}SCA}(\mathsf{true},\, 5.0), \mathsf{ESM}(\mathsf{true},\, 5.0), \dots) \\
a_{\text{CCO-SCA},\overline{\text{ESM}}} &= (\mathsf{CCO{-}SCA}(\mathsf{true},\, 5.0), \mathsf{ESM}(\mathsf{true},\, 1.0), \dots) \\
a_{\overline{\text{CCO-SCA}},\text{ESM}} &= (\mathsf{CCO{-}SCA}(\mathsf{true},\, 18.0), \mathsf{ESM}(\mathsf{true},\, 5.0), \dots) \\
a_{\overline{\text{CCO-SCA}},\overline{\text{ESM}}} &= (\mathsf{CCO{-}SCA}(\mathsf{true},\, 18.0), \mathsf{ESM}(\mathsf{true},\, 1.0), \dots),
\end{aligned} \tag{7.1}
$$

where $\bar{\cdot}$ indicates a relaxed SFC.

The cell load-focused and energy consumption-focused KPI objectives define the same utility functions but different weights. Hence, the unweighted utilities of the SON configurations for each KPI are the same. Table 7.2 outlines the resulting utilities and agreements for the possible configurations. Thereby, it focuses on the relevant KPIs cell load and energy consumption. Notice that the utility for the KPI CQI is always $\langle(1.00, 1.00, 0.33)\rangle_{\mathbf{u}} = 0.78$ and for the handover ping-pong rate $\langle(1.00, 1.00, 0.33)\rangle_{\mathbf{u}} = 0.78$ and their KPI agreement is consistently 1.0, since the SFCs of the CCO-SCA and ESM function do not affect these two KPIs.

Given the KPI utilities, it is possible to analyze the resulting expected behavior of the SON functions under a specific SON configuration:

$a_{\text{CCO-SCA,ESM}}$ configures CCO-SCA to aggressively reduce the load and ESM to aggressively reduce the energy consumption. This contradictory SON configuration is expected to achieve neither a high satisfaction of the cell load objective nor of the energy consumption. The conflict in this configuration is shown in the low agreement of 0.56. Hence, by setting the agreement threshold to, e.g., $\rho_{\text{agree}} = 0.6$, this configuration would be rejected as conflicting.

$a_{\text{CCO-SCA},\overline{\text{ESM}}}$ configures CCO-SCA to aggressively reduce the load and ESM to be relaxed. Overall, this configuration is expected to lead to a low cell load at the cost of a high energy consumption.

$a_{\overline{\text{CCO-SCA}},\text{ESM}}$ configures CCO-SCA to be relaxed and ESM to aggressively reduce the energy consumption. Overall, this configuration is expected to lead to a low energy consumption at the cost of a high cell load.

$a_{\overline{\text{CCO-SCA}},\overline{\text{ESM}}}$ configures CCO-SCA to be relaxed and ESM to be relaxed. This configuration is expected to achieve neither a high satisfaction of the cell load objective nor of the energy consumption. The conflict in this SON configuration is shown in the low agreement of 0.56. Hence, by setting the agreement threshold to, e.g., $\rho_{\text{agree}} = 0.6$, this configuration would be rejected as a conflicting.

Based on these unweighted utilities, it is possible to reconstruct the decision making of objective-driven SON management in the concrete scenario. Table 7.3 shows the resulting overall utilities for the different SON configurations in the different periods. In the cell load-focused periods, i.e., RL, BL, HL, $a_{\text{CCO-SCA},\overline{\text{ESM}}}$ has the highest utility and, thus, ODSO SON management will configure all cells with this SON configuration during these periods. In the energy consumption-focused periods, i.e., RE, BE, HE, $a_{\overline{\text{CCO-SCA}},\text{ESM}}$ has the highest utility and, thus, ODSO SON management will configure all cells with this SON configuration during these periods. The behavior nicely resembles the expectations of the MNO that in cell load-focused periods CCO-SCA is aggressive and ESM is relaxed, whereas in energy consumption-focused periods ESM is aggressive and CCO-SCA is relaxed. Notice that the other two, contradictory SON configurations would never be selected due to their consistently low objective satisfaction.

Figure 7.10 shows the results of the simulation of the scenario for the considered Cell 6. The upper four graphs depict the measured KPI values for the 40 granularity periods. The dashed, yellow and green lines show the thresholds on the KPIs for the acceptable and optimal region respectively. Furthermore, the number of requests per granularity period by the SON functions are shown as bars with the scale on the right. Thereby, the SON functions requests are aligned with the KPIs they are monitoring: CCO-RET requests are shown in the CQI chart, MRO requests in the handover ping-pong rate chart, CCO-SCA requests in the cell load chart, and ESM requests in the energy consumption chart. The colors of the bars correspond to

Table 7.3: The utilities of the possible SON configurations during cell load-focused periods, i.e., RL, BL, HL, and energy consumption-focused periods, i.e., RE, BE, HE.

	RL, BL, HL	RE, BE, HE
$a_{\text{CCO-SCA,ESM}}$	$0.64 = \langle(1.00, 0.93, 0.29)\rangle_{\mathbf{u}}$	$0.60 = \langle(1.00, 0.80, 0.18)\rangle_{\mathbf{u}}$
$a_{\text{CCO-SCA,}\overline{\text{ESM}}}$	$\mathbf{0.66} = \langle(1.00, 0.97, 0.32)\rangle_{\mathbf{u}}$	$0.57 = \langle(1.00, 0.72, 0.18)\rangle_{\mathbf{u}}$
$a_{\overline{\text{CCO-SCA}},\text{ESM}}$	$0.63 = \langle(1.00, 0.88, 0.26)\rangle_{\mathbf{u}}$	$\mathbf{0.63} = \langle(1.00, 0.89, 0.26)\rangle_{\mathbf{u}}$
$a_{\overline{\text{CCO-SCA}},\overline{\text{ESM}}}$	$0.64 = \langle(1.00, 0.93, 0.29)\rangle_{\mathbf{u}}$	$0.60 = \langle(1.00, 0.80, 0.18)\rangle_{\mathbf{u}}$

the colors of the SON configurations. The graph at the bottom depicts the overall utility in the reduced form (see Definition 7.1). The dashed, yellow and green lines show the threshold on the utility for the acceptable and optimal region respectively.

The simulation has been executed in three different configurations in order to enable a comparison of objective-driven SON management with fixed SON configurations that are not changed during the simulation. Thereby, we concentrated on the conflict-free SON configurations $a_{\overline{\text{CCO-SCA}},\text{ESM}}$ and $a_{\text{CCO-SCA},\overline{\text{ESM}}}$. Since the two different SON configurations mainly affect the cell load and the energy consumption, the following analysis of the behavior in the different periods focuses on these KPIs. The variations in the CQI and handover ping-pong rate are primarily due to random variations in the simulation. In the analysis of the simulation results, it may seem that ODSO has a lag of 1 granularity period, e.g., from Period 7 to Period 8. However, this can be explained by the fact the utility in Period 7 is calculated already with the new objectives in the current granularity period. However, the results of the actions of ODSO in Period 7 due to the new objectives can only be monitored in the following Period 8.

RL: ODSO management deploys $a_{\text{CCO-SCA},\overline{\text{ESM}}}$, i.e., the small cells are switched on. Consequently, the cell load is lower than the constant configuration $a_{\overline{\text{CCO-SCA}},\text{ESM}}$, however, the energy consumption is higher. Unfortunately, this is not the optimal decision in this scenario, since the overall utility of $a_{\overline{\text{CCO-SCA}},\text{ESM}}$ is considerably higher. This is due to the fact that in this specific situation the actual cell load for $a_{\overline{\text{CCO-SCA}},\text{ESM}}$ is better than the expected cell load stated in the SON function model. This outlines the importance of accurate SON function models for the ODSO approach.

RE: ODSO management deploys $a_{\overline{\text{CCO-SCA}},\text{ESM}}$, i.e., the small cells are switched off. Consequently, the cell load is higher than the constant configuration $a_{\text{CCO-SCA},\overline{\text{ESM}}}$, however, the energy consumption is lower. As can be seen, this change of the SON configuration yields a considerable improvement of the satisfaction of the operator objectives since the overall utility is about 0.4 higher than the overall utility of $a_{\text{CCO-SCA},\overline{\text{ESM}}}$. Since the load situation did not change, this is solely due to the change in the KPI weights, i.e., the focus of the operator objectives.

BL: ODSO management deploys $a_{\text{CCO-SCA},\overline{\text{ESM}}}$. Consequently, the cell load is lower

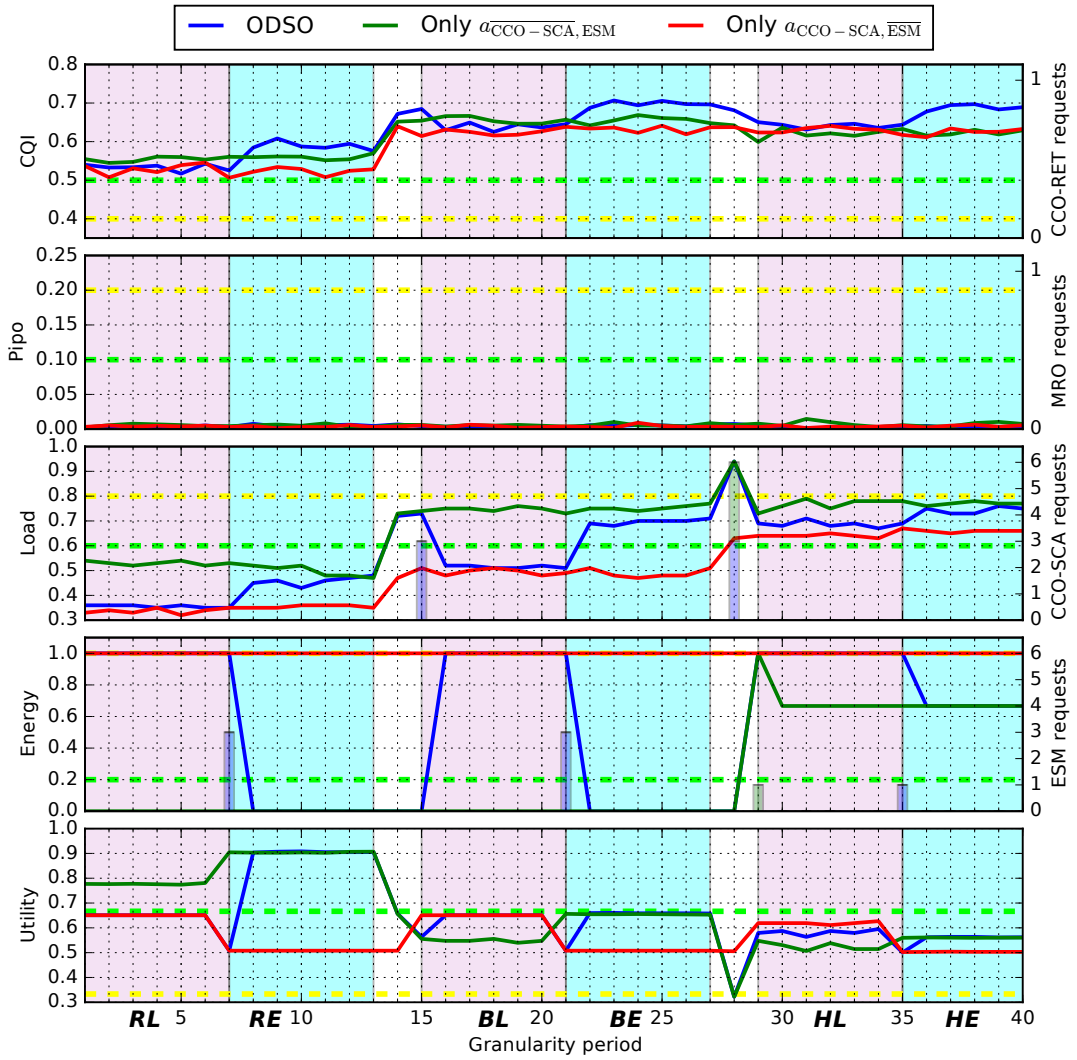


Figure 7.10: The simulation results for the SON management scenario comparing the performance of Cell 6 for the configurations ODSO, $a_{\overline{\text{CCO-SCA}},\text{ESM}}$, and $a_{\text{CCO-SCA},\overline{\text{ESM}}}$.

than the constant configuration $a_{\overline{\text{CCO-SCA}},\text{ESM}}$, however, the energy consumption is higher. Comparing the overall utility of this configuration with the overall utility of $a_{\overline{\text{CCO-SCA}},\text{ESM}}$, this yields a higher satisfaction of the operator objectives. Hence, this change in the SON configuration can be seen as a rational decision.

BE: ODSO management deploys $a_{\overline{\text{CCO-SCA}},\text{ESM}}$. Consequently, the cell load is higher than the constant configuration $a_{\text{CCO-SCA},\overline{\text{ESM}}}$, however, the energy consumption is lower. Comparing the overall utility of this configuration with the overall utility of $a_{\text{CCO-SCA},\overline{\text{ESM}}}$, this yields again a higher satisfaction of the operator objectives. Since the load situation did not change, this is solely due to the change in the KPI weights, i.e., the focus of the operator objectives.

HL: ODSO management deploys $a_{\text{CCO-SCA},\overline{\text{ESM}}}$. Since the cell load in this period is very high due to the hot spot users, even the configuration $a_{\overline{\text{CCO-SCA}},\text{ESM}}$ switches on two of the three small cells. The reason for this is that the hot spot is present in only two of the three small cells. Therefore, the CCO-SCA function first turns on all three small cells and then ESM turns off the cell that does not cover the hot spot area. Comparing the overall utility of the ODSO configuration with the overall utility of $a_{\overline{\text{CCO-SCA}},\text{ESM}}$, the former still yields a higher satisfaction of the operator objectives.

HE: ODSO management deploys $a_{\overline{\text{CCO-SCA}},\text{ESM}}$. Again, this causes two small cells to stay switched on. As can be seen, this increases the cell load and reduced the energy consumption. Since energy savings is considered important, the ODSO selected SON configuration results in a higher overall utility compared with $a_{\text{CCO-SCA},\overline{\text{ESM}}}$. Since the load situation did not change, this is solely due to the change in the KPI weights, i.e., the focus of the operator objectives.

The key insight to gain from the analysis of the scenario is that the ODSO approach for SON management dynamically adapts the SON configuration such that the expected satisfaction of the utilities is maximized. This does not mean, that the selected configuration is always optimal, especially in those situations where the SON function models are not accurate. However, most of the time, the selected configuration is optimal and, actually, resembles the decision that a human operator would make. This can be seen in Figure 7.10 since the overall utility of ODSO is in most periods as high as the best constant configuration. This shows that even the adaptation of the configuration of quite dynamic and intelligent SON functions like the presented CCO-SCA and ESM functions, promises better alignment to the actual operator objectives.

For completeness, Figure 7.11 provides the simulation results of the SON configuration $a_{\text{CCO-SCA,ESM}}$. As can be seen, the conflict in the latter configuration leads to oscillations in the network configuration, specifically, the small cells are turned on and off alternately. Besides the low utility, such varying network configuration may also lead to instabilities and unpredictable problems in the network that the MNO aims to avoid.

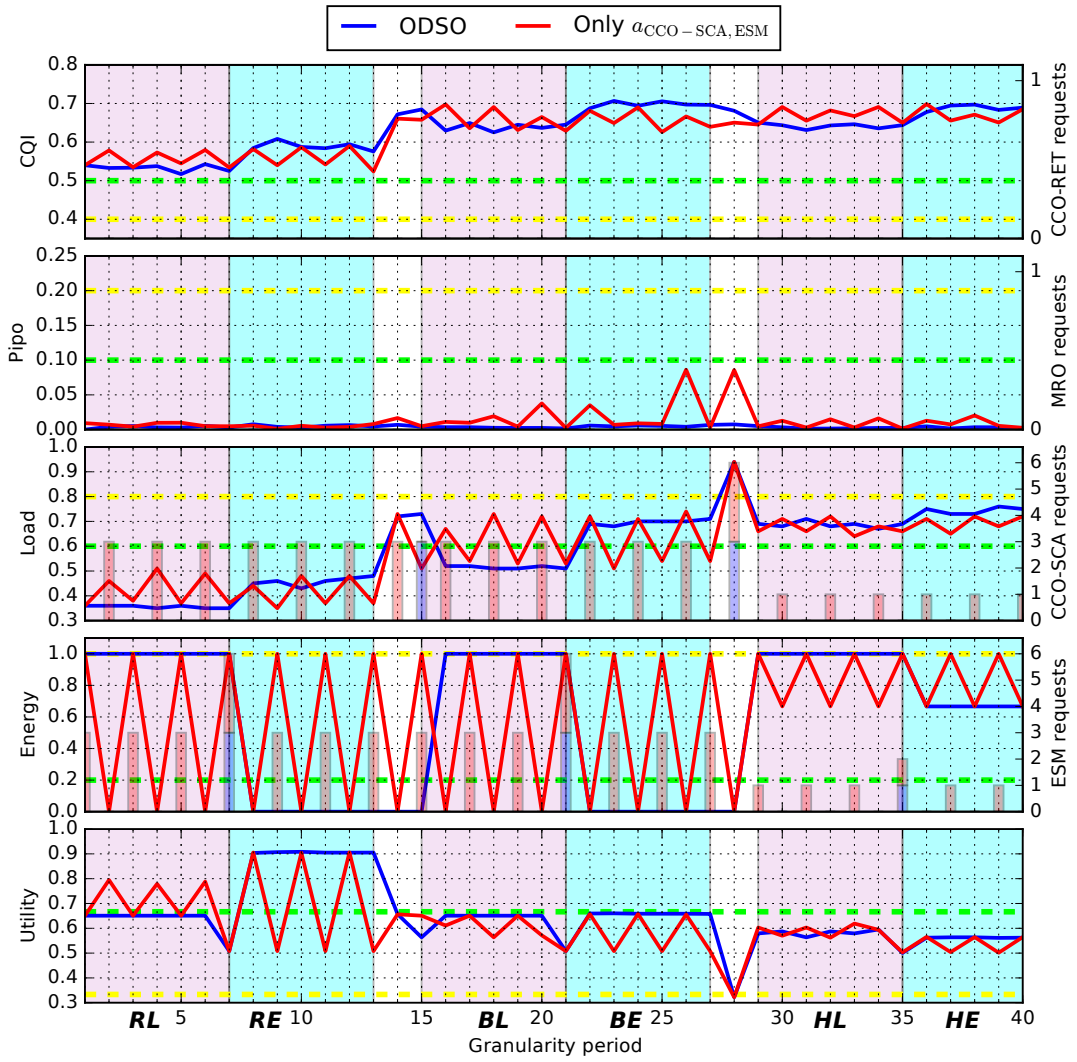


Figure 7.11: The simulation results for the SON management scenario comparing the performance of Cell 6 for the configurations ODSO, and the conflictful $a_{\mathrm{CCO-SCA,ESM}}$.

7.4 SON Coordination

The goal of objective-driven SON coordination is to schedule the sequential execution of conflicting actions such that the satisfaction of the operator objectives is maximized as quickly as possible. Therefore, the evaluation of the ODSO coordination component is based on a scenario that requires a decision between several conflicting SON function requests that aim at overcoming different performance problems. In order to show the advantages of the objective-driven decision making, we compare the ODSO approach with a state-of-the-art, policy-based SON coordination approach.

7.4.1 Scenario

The scenario focuses on the three-sectored BS that hosts Cell 4, Cell 5, and Cell 6 shown in Figure 7.12. The simulation starts with a non-optimal configuration of the cells, which introduces performance problems that, in turn, trigger the respective SON functions to solve them. Although this setup seems to be artificially chosen to exemplify the advantage of objective-driven coordination, such a case can also happen in a real network, e.g., if a new BS is deployed into the network for which an optimal configuration has not been determined before. Specifically, the following problems are present: the CIO of Cell 4 is too low for the actual network layout, resulting in an increased handover ping-pong rate; the RET of Cell 5 is too high which leads to only a small area of good reception, resulting in an reduced CQI for the UEs; and Cell 6 experiences an overload situation due to 60 additional users during busy hours. Apart from these problems, the network configuration corresponds to the details shown in Table 7.1 with the regular user population and the small cells turned off.

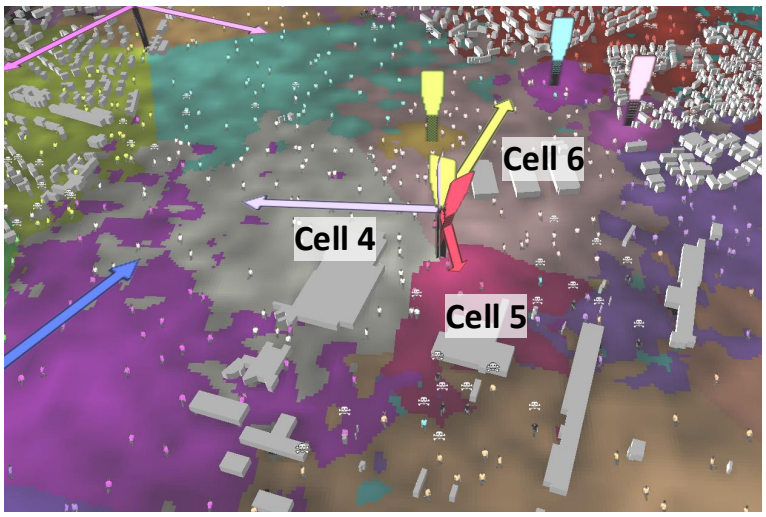


Figure 7.12: The problematic networks cells: Cell 4 has a misconfigured CIO, Cell 5 has a wrong RET, and in Cell 6 are unusually many UEs.

Table 7.4: SON function request conflicts

Category	Parameter	Value
Impact-area	CCO-RET	Target cell + direct neighbors
	MRO	Target cell pair
	MLB	Target cell pair
Conflict dependencies	CCO-RET - CCO-RET	Overlapping impact-area
	CCO-RET - MRO	Overlapping impact-area
	CCO-RET - MLB	Overlapping impact-area
	MRO - MRO	Equal impact-area
	MRO - MLB	Equal impact-area
	MLB - MLB	Equal impact-area

The three active SON functions, which are configured according to the common SON function model, aim to solve the introduced problems: MRO attempts to optimize the CIO of Cell 4, CCO-RET optimizes the RET of Cell 5, and MLB optimizes the CIO of Cell 6 to offload some UEs to its neighbors. There are no further SON functions active. Since the network configuration, apart from the considered three cells, is quite good, the SON functions are barely active in other cells.

The evaluation compares the ODSO coordination approach with a well-known policy-based coordination approach that has been published in [Ban13] (see Chapter 5.5.2.2). This approach uses rules to detect and resolve conflicts at the same time. Since the ODSO coordination is focusing on conflict resolution, the policy-based coordination approach is separated into an equivalent rule-based conflict detection and a priority-based conflict resolution. This allows utilizing the same conflict detection for both coordination approaches. As shown in Table 7.4, the conflict detection is based on predefined impact-areas of the SON function requests (see Chapter 5.3.1.2), and rules that define the dependencies. The conflict detection rules are derived from the case study for the policy-based coordination approach [Ban13] and experience. They can be summarized in the following way:

- RET changes by CCO-RET change the cell size of the targeted cell, which has a huge impact on the cell and its direct neighbors. Hence, no other SON function should be active in this impact-area.
- An MRO SON function request affects only one neighbor relationship without any effect on the other neighbor cells. Hence, there should be no other MRO or MLB instance active for the same cell pair.
- An MLB is similar to an MRO SON function request. Hence, there should be no other MRO or MLB instance active for the same cell pair.

The policy-based conflict resolution prioritizes the CCO-RET function before the MRO function before the MLB function. This is motivated by the idea that the CCO-RET affects the performance of a whole cell, whereas MRO and MLB typically optimize solely a specific neighbor relation. The objective-driven SON conflict resolution is guided by the operator objectives and the expected effects of a SON

function request. Both are not changed compared to the default scenario. Furthermore, the approach presented in Chapter 5.3.1.1 has been adopted to handle multiple SON function requests on the same cell by MRO and MLB.

7.4.2 Results

Figure 7.13 and Figure 7.14 show the results of the simulation for the policy-based and objective-driven coordination in the same manner in order to ease comparison. Both figures are divided into two parts. The upper three graphs show the values for the three KPIs, namely CQI, handover ping-pong rate, and cell load, for the Cell 4, Cell 5, and Cell 6 as lines over the 30 granularity periods of the simulation. The dashed, yellow and green lines show the threshold for the acceptable and optimal region respectively. Furthermore, these three graphs show the number of requests per granularity period by the SON functions as bars with the scale on the right. Thereby, the SON functions requests are aligned with the KPIs they are monitoring: CCO-RET requests are shown in the CQI chart, MRO requests in the handover ping-pong rate chart, and MLB requests in the cell load chart. The colors of the bars correspond to the colors of the cells. So, for instance, Figure 7.13 shows that in the first granularity period Cell 5 reports a CQI value of about 0.38 which triggers a SON function request by CCO-RET for Cell 5 in the same granularity period. The bottom graph of Figure 7.13 and Figure 7.14 shows the overall cell utility of Cell 4, Cell 5, and Cell 6 in the reduced form as lines. Additionally, the black line shows the average utility of the three cells in the reduced form (see Definition 7.1). This is calculated, similarly to the utility of a set of SON function requests presented in Definition 5.11, as the average utility for each priority region and then turned into the reduced form. The dashed, yellow line indicates the threshold of the acceptable region and the dashed, green line shows the threshold for the optimal region. Furthermore, the bars show the number of accepted requests for each SON function from the three cells with the scale on the right. As can be seen, the initial network configuration is equal for both simulations and, thus, also the performance.

The policy-based conflict resolution is driven by the function priorities. As shown in Figure 7.13, it is first accepting every CCO-RET function request until the CQI is good enough such that the CCO-RET function is not active anymore in granularity period 10. Notice that the little pause in period 7 is a random variation. All concurrent MRO and MLB requests are rejected. This improves the utility of the CQI objective. Starting in period 10, it accepts all MRO requests and rejects conflicting MLB requests. However, some MLB requests are not in conflict with the MRO requests and, thus, are accepted. Based on the KPIs, it can be seen that the SON finally adapted the network configuration to the new situation around period 16. Later requests are caused by random variations in the simulations.

The objective-driven coordination, shown in Figure 7.14, first analyses the possible improvements of each and every request, i.e., their effects. In this scenario, it is roughly the difference of each cell utility of the current system performance to the dashed, green line. Then, it determines which conflict-free subset of the requests improves the utility the most. Thus, it computes that the concurrent execution

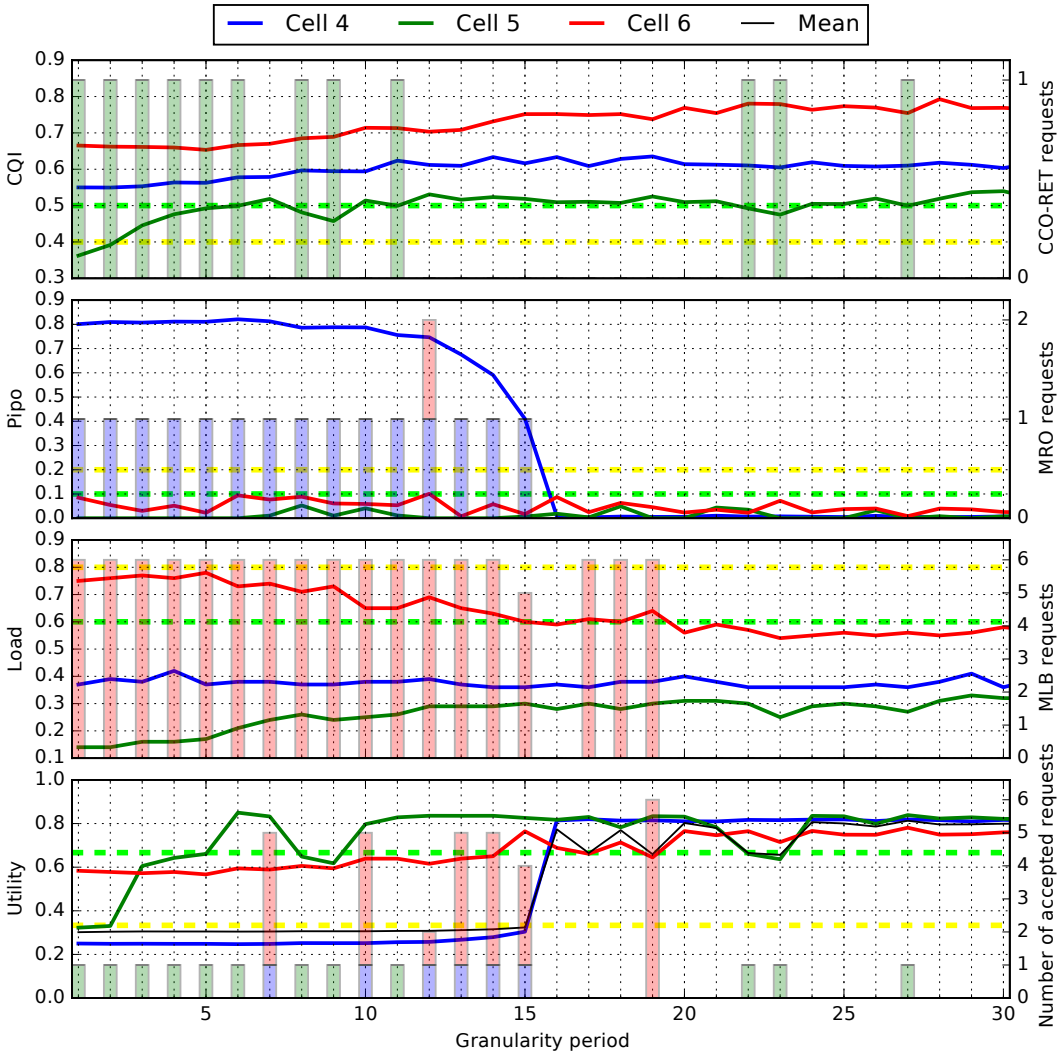


Figure 7.13: Simulation result of policy-based coordination.

of the MRO and MLB requests is expected to yield a higher system performance improvement than the execution of the CCO-RET request. Consequently, the MRO and MLB requests are accepted in the beginning of the simulation. Just later on, when the CIOs are close to optimal in granularity period 7, the CCO-RET requests are accepted and the CQI improves. Considering the KPIs, the SON finally adapted network configuration to the new situation around period 14. Again, the later requests are due to some random variations.

Comparing the overall utility, it can be seen that policy-based and objective-driven coordination achieve a stable and full satisfaction of the objectives in around granularity period 15. It is not surprising that both approaches require a similar amount of time, because the scenario, in principle, requires a fixed number of accepted SON function executions, i.e., configuration changes by the SON functions, to solve the problems. Furthermore, due to the shared conflict-detection, there is theoretically the same selection of conflict-free sets of SON function requests both

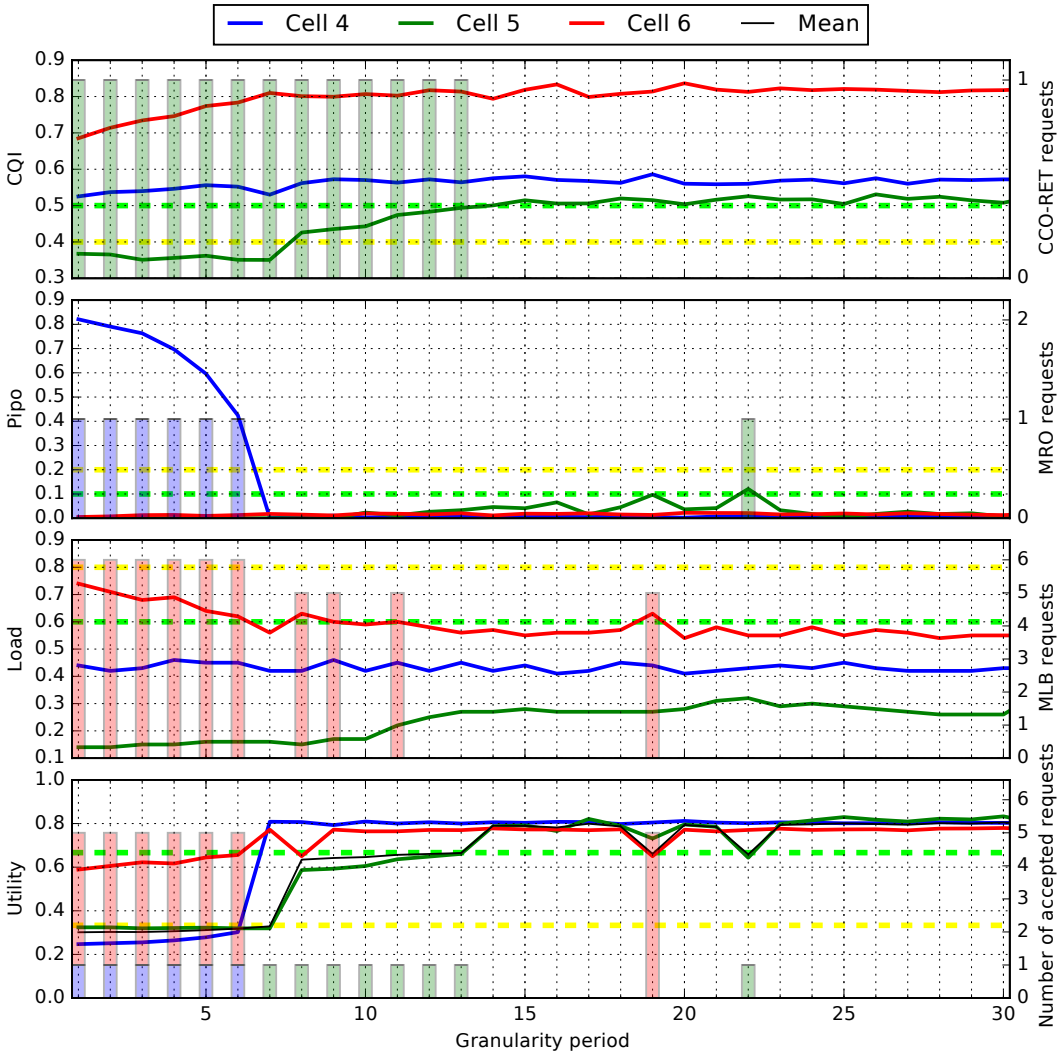


Figure 7.14: Simulation result of objective-driven coordination.

approaches can choose from. However, it can be seen that objective-driven SON coordination achieves a great utility increases already in the beginning of the simulation between period 7 and 8 because it concentrates first on the important problems with respect to the operator objectives. In contrast, the policy-driven coordination achieves these improvements later between Round 15 and 16. Of course, this result depends on the specific priorities of the SON functions in this scenario, and another order can produce a similar behavior to the objective-driven coordination. However, following this argument would require the operator to adapt the priorities for each and every coordination situation which, actually, is a manual implementation of the objective-driven conflict resolution. This nicely summarizes the general goal of objective-driven coordination: focus on the worst problems regarding the operator objectives that can be overcome quickly.

7.5 SON Self-Healing

The SON self-healing component of the ODSO architecture is supposed to solve mainly two problems (see Chapter 6.1): integrate the SON into the self-healing procedure in order to utilize SON functions for problem detection, root cause diagnosis and recovery, and provide an objective-driven decision making for the selection of the executed recovery measure. This evaluation, therefore, is twofold: first, we present the involvement of SON and, second, we focus on degradation recovery.

7.5.1 Involvement of SON

This chapter concentrates on showing the ability of the ODSO SON self-healing component to detect and diagnose problems in the network by monitoring the execution of SON functions. Specifically, a sleeping cell, which does not accept any traffic, is introduced into the simulated network. Sleeping cells are a serious problem in mobile networks since they are performing poorly without generating any specific failure alarms [Sch+11]. Additionally, they typically do not report any PM data. Although the absent data may indicate some problem in the cell, this is not a certain symptom. Missing KPI reports are not uncommon in operational networks (see [Ben+13a, Ch. 2.4]), e.g., due to overload in the core network. Hence, this indication is often ignored by the operational personnel and traditional SON self-healing if it is not underpinned by other anomalies. As a result, sleeping cells often remain unrecognized for hours or even days. A sleeping cell can be caused by software failures, in which case the remedy can be the reset of the cell's software configuration.

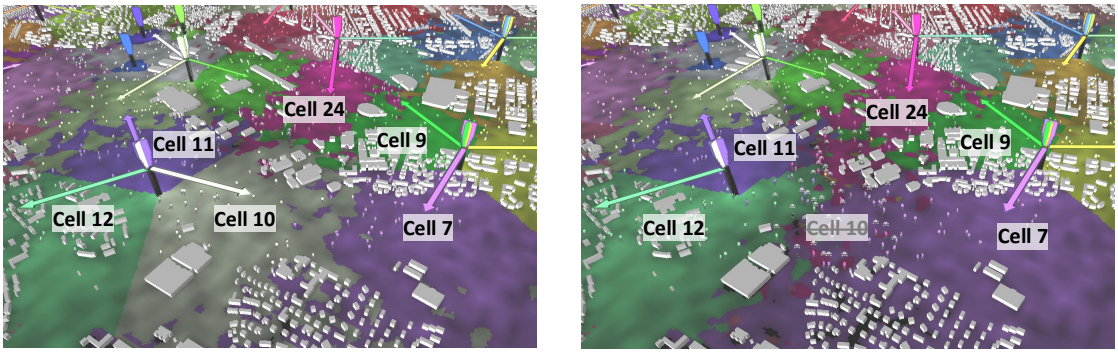
In this scenario, the ineffective SON functions in the neighborhood of the introduced sleeping cell provide additional evidence for correctly detecting and diagnosing the problem. Based on this, the problem can be easily recovered via triggering the reset SON function for the specific network area.

7.5.1.1 Scenario

The scenario focuses on the network area shown in Figure 7.15. The network is operational and, since the initial configuration is quite good, the deployed SON functions are barely active. However, in granularity period 5, Cell 10 will face an error that turns it into a sleeping cell. Particularly, this means that Cell 10 will not serve any UE and so, the users in its coverage area will need to be served, if possible, by the neighboring cells. However, since Cell 10 is sleeping, it is assumed that the BS does not raise any alarm and no KPIs will be reported.

In the scenario, there are three SON functions active, CCO-RET, MRO, and MLB which are configured as outlined in Chapter 7.2.4. Additionally, the reset SON function is deployed, however, it needs to be actively triggered.

In order to detect a problem in the network, SON self-healing monitors the activity of the SON functions and raises an alarm in case an ineffective SON function is detected. During operation, the degradation detection keeps smoothed values of all KPIs of all network cells, which minimizes random effects of the environment.



(a) Cell 10 operational.

(b) Cell 10 in failure state.

Figure 7.15: Visualization of the cells of the SON involvement scenario and the impact of Cell 10 in sleeping mode.

Thereby, exponential smoothing [Win04, Ch. 24.2] with $\alpha = 0.3$ is employed. After a SON function instance was active in a cell, the degradation detection evaluates the cell KPIs in the next granularity period: if the values of the KPIs did not improve by at least a factor of 1.2 above the smoothed value then the SON function instance is marked ineffective. Furthermore, if a SON function instance has been ineffective 5 times in a row then the degradation detection raises an alarm for that instance. This alarm can be seen as an indication that the SON function instance is not able to achieve its optimization goal. The employed mechanism can be considered as a history-based detection method.

The diagnosis of the raised alarms is quite simple: An alarm emitted by a CCO-RET SON function is considered as an indication for a sleeping cell in the neighborhood. Furthermore, the recovery planning in this scenario for SON self-healing is the following: If the diagnosis for a cell problem indicates a possible sleeping cell in the neighborhood then the reset SON function is triggered for all neighboring cells. This function is supposed to perform a basic check whether the cell, it is triggered for, is operational, and attempts to reset the cell if it is unavailable.

7.5.1.2 Results

Figure 7.16 shows the results of the simulation. The upper three graphs show the values for the three KPIs, namely CQI, handover ping-pong rate, and cell load, for the problematic Cell 10 and its neighbors Cell 7, Cell 9, Cell 11, Cell 12, and Cell 24 as lines over 15 granularity periods of the simulation. The dashed, yellow and green lines show the threshold for the acceptable and optimal region respectively. Furthermore, the number of requests per granularity period by the SON functions are shown as bars with the scale on the right. Thereby, the SON functions requests are aligned with the KPIs they are monitoring: CCO-RET requests are shown in the CQI chart, MRO requests in the handover ping-pong rate chart, and MLB requests in the cell load chart. The colors of the bars correspond to the colors of the cells.

Additionally, the light gray area shows the granularity periods 6 to 11 in which Cell 10 is sleeping. During that time, its KPI values are not reported any more, hence, there is a gap in the plot. The graph at the bottom shows no KPI but the requests by the reset SON function as bars similarly to the graphs above.

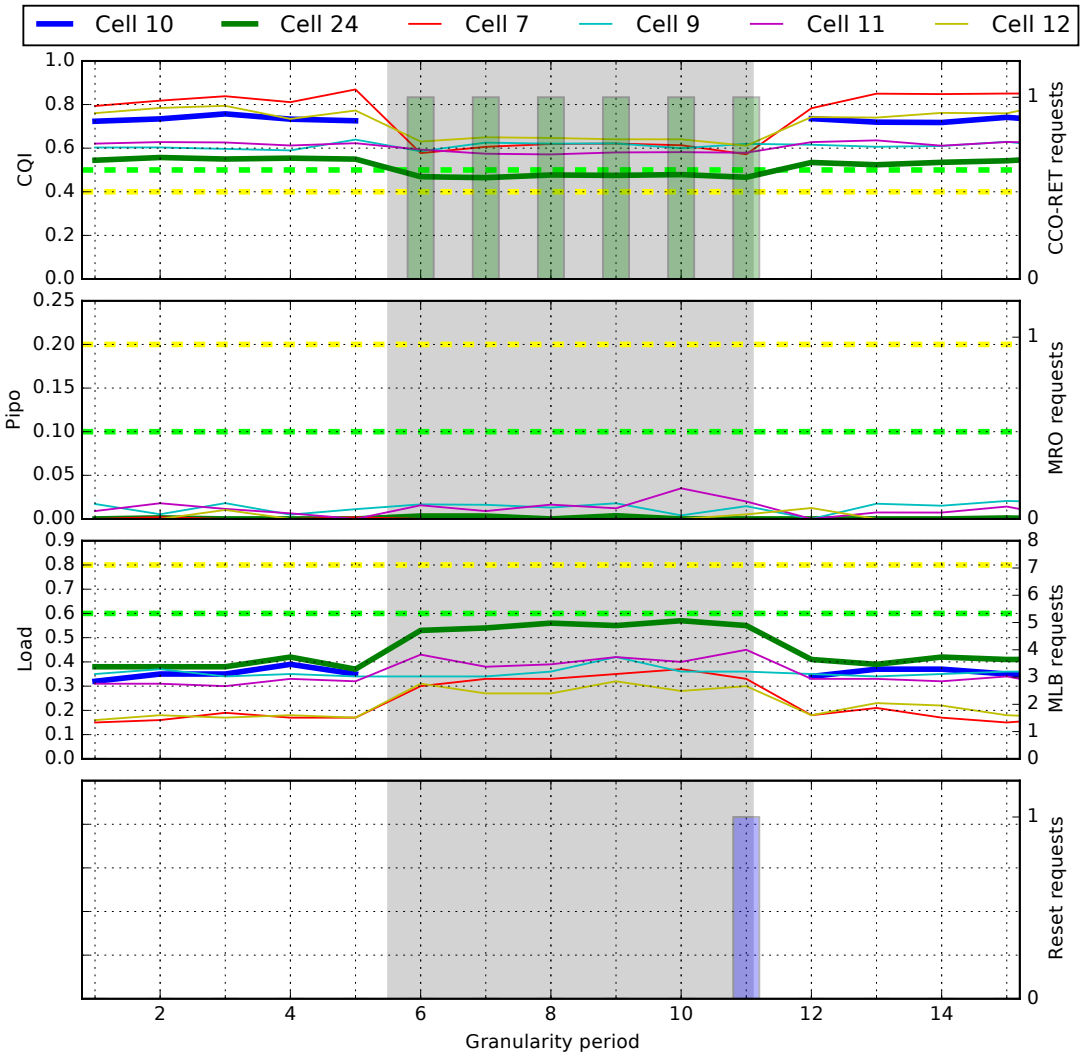


Figure 7.16: KPI values from the simulation of the detection of the erroneous Cell 10 by the SON-focused degradation detection.

Apart from the missing KPI reports, Cell 10 does also not accept any UEs. Hence, the neighboring cells, in particular Cell 24, have to take over these users. This inevitably leads to a very bad signal quality for the new UEs that are far away. Consequently, the CQI of Cell 24 is considerably reduced which triggers the CCO-RET function. The CCO-RET SON function attempts to improve the CQI value of Cell 24, but it is not able to improve the cell's performance even after 5 accepted SON function requests. This ineffectiveness is detected by the degradation detection which raises a respective alarm. This alarm is seen as a verification that the missing KPIs in Cell 10 are really the result of a severe problem and, thus, root cause

diagnosis is able to diagnose the sleeping cell condition. Based on this, degradation recovery triggers the reset SON function in granularity period 11 as a recovery action. This restores Cell 10 and immediately improves the CQI of Cell 24. As a result, the network is fully operational again.

7.5.2 Degradation Recovery

This chapter concentrates on showing the benefits of the objective-driven decision making by the autonomic degradation recovery of ODSO SON self-healing. Specifically, we outline its advantage compared to several other selection mechanisms by evaluating a number of simulated failure diagnosis results.

7.5.2.1 Scenario

The simulation focuses on the comparison of the different degradation recovery approaches based on artificially created cell diagnoses. Hence, this scenario does use neither the LTE network simulator nor the SON function engine. The general evaluation procedure consists of three steps:

1. The input for the degradation recovery is generated. This comprises the creation of an artificial, probabilistic cell diagnosis as well as an operational context including KPI values. Furthermore, the supposed true root cause of the diagnosed problem is selected. These inputs are referred to as diagnosis case.

2. The cell diagnosis and operational context are passed to all degradation recovery approaches which, in turn, compute a recovery actions to perform.

3. For each degradation recovery approach, the recovery action is evaluated with respect to its effectiveness to recover the true root cause of the diagnosis case. The result is logged.

In order to generate some statistical relevant results, we have performed the above process 1000 times and present the aggregated results.

Creation of Diagnosis Cases Each diagnosis case captures one concrete problem situation that the degradation recovery approaches needs to recover. It is created in the following process:

1. The operational context of the problematic cell is defined: Therefore, the values of the KPIs CQI, handover ping-pong rate, and cell load are sampled according to Gaussian distributions with $\mathcal{N}(\mu, \sigma^2)$ as shown in Figure 7.17. Furthermore, the context contains five CM and operational properties (see Chapter 3.3.1.3), shown in Table 7.5, that indicate the current time of the day, whether the cell is experiencing a special event, e.g., a sports game, the layer of the cell, and when the most recent change in the configuration or software update of the cell or BS happened. Each of them is sampled with the given probabilities.

Table 7.5: Context parameters for the creation of diagnosis cases.

Cell property	Domain / Probability
time	$\Pr(\text{day}) = 0.50$, $\Pr(\text{night}) = 0.50$
specialEvent	$\Pr(\text{yes}) = 0.13$, $\Pr(\text{no}) = 0.87$
cellLayer	$\Pr(\text{macro}) = 0.50$, $\Pr(\text{small}) = 0.50$
configurationChange	$\Pr(\text{recently}) = 0.25$, $\Pr(\text{awhile}) = 0.75$
softwareUpdate	$\Pr(\text{recently}) = 0.25$, $\Pr(\text{awhile}) = 0.75$

Table 7.6: Root causes for the creation of diagnosis cases.

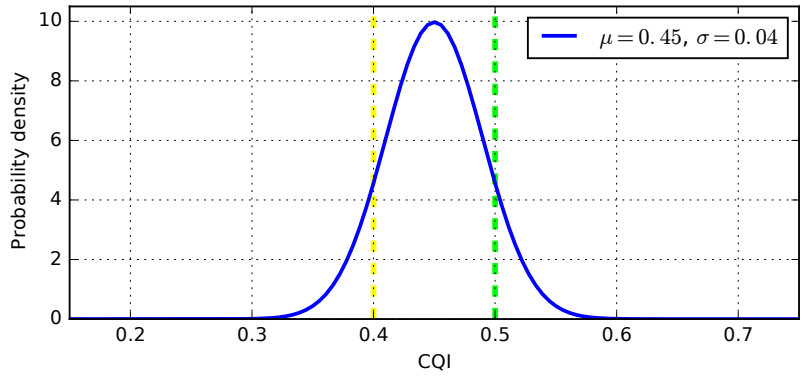
Root causes
hardwareFailure
softwareFailure
coverageHole
unknown

2. A cell diagnosis result is created based on the root causes given in Table 7.6: a hardware failure in the BS, a software failure in the BS, an erroneous configuration of the TXP, or it is an unknown, other root cause. The cell diagnosis result is drawn over these root causes such that the sum of the probabilities is 1.0.

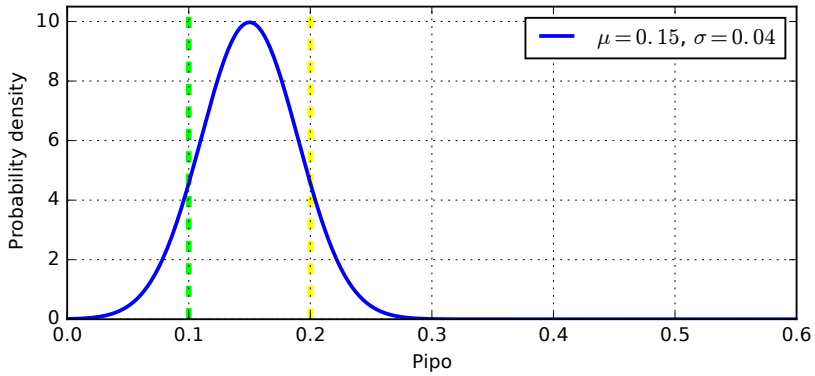
3. The true root cause of the degradation is sampled according to the cell diagnosis result. Consequently, the simulation is based on the assumption that the root cause diagnosis is accurate.

Degradation Recovery for Diagnosis Cases For each diagnosis case, the cell diagnosis and operation context is passed to each and every degradation recovery approach. In order to allow for a fair comparison, all degradation recovery approaches use the same recovery, cost and objective model, i.e., they have the same information about possible countermeasures for a root cause. Listing 7.1 shows the common recovery model using the syntax outlined in Chapter 6.3.3.3. As can be seen from the listing, the following recovery actions are considered: perform a restart of the BS, trigger the SON function CCO-RET, revert the last configuration change, undo the last software update, and triggering the manual inspection of a failure via a trouble ticket. The SON functions CCO-RET, MRO, and MLB are active with the SON function effects presented in Chapter 7.2.4.

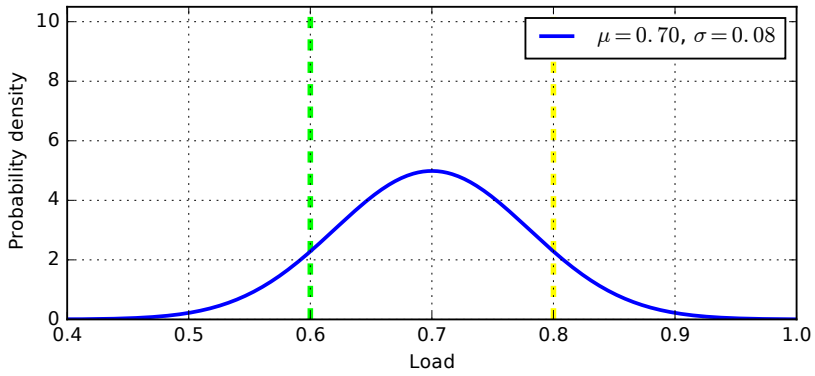
The common objective model is the same as presented in Chapter 7.2.4, and the cost model is shown in Listing 7.2. This cost model is artificially created by us based on the heuristic presented in Chapter 6.3.3.4 with the following reasoning: All recovery actions except triggering CCO-RET should only be considered if the KPIs are not optimal. This is because they either have actual monetary costs, e.g., a trouble ticket, or they may have a negative impact on the network performance. The reverting actions as well as the BS reset are equally preferred. We assume that



(a) CQI.



(b) Handover ping-pong rate.



(c) Cell load.

Figure 7.17: Probabilistic KPI value distributions for the creation of diagnosis cases.

```
ROOTCAUSE hardwareFailure IF true
  THEN restartBs                 YIELDS sfe(cell, *)   WITH 0.5
ROOTCAUSE hardwareFailure IF true
  THEN troubleTicket             YIELDS sfe(cell, *)   WITH 0.9

ROOTCAUSE softwareFailure IF softwareUpdate = recently
  THEN revertSoftwareUpdate      YIELDS sfe(cell, *)   WITH 0.7
ROOTCAUSE softwareFailure IF true
  THEN restartBs                 YIELDS sfe(cell, *)   WITH 0.5
ROOTCAUSE softwareFailure IF true
  THEN troubleTicket             YIELDS sfe(cell, *)   WITH 0.9

ROOTCAUSE coverageHole IF true
  THEN triggerCcoRet             YIELDS sfe(cell, cco) WITH 0.7
ROOTCAUSE coverageHole IF configurationChange = recently
  THEN revertConfigurationChange YIELDS sfe(cell, *)   WITH 0.7
ROOTCAUSE coverageHole IF softwareUpdate = recently
  THEN revertSoftwareUpdate      YIELDS sfe(cell, *)   WITH 0.7
ROOTCAUSE coverageHole IF true
  THEN restartBs                 YIELDS sfe(cell, *)   WITH 0.5
ROOTCAUSE coverageHole IF true
  THEN troubleTicket             YIELDS sfe(cell, *)   WITH 0.9
```

Listing 7.1: Recovery model.

all three action cause a temporary outage of the cell due to a restart of the BS. Therefore, these actions are less preferred for macro cells which ensure a mandatory coverage compared to small cells which provide optional capacity. The creation of a trouble ticket should only be considered if the network performance is degraded beyond the acceptable state. However, at night, less operational personnel is available and trouble tickets are less preferred. Notice that if a special event like a sports game is ongoing then providing a good QoS is of utmost importance so that the MNO accepts manual inspections more commonly. This is due to the visibility of network problems to numerous users. Of course, this reasoning may not be valid for every MNO, but, fortunately, a MNO can adapt the cost model to its specific needs.

For one diagnosis case, each degradation recovery approach determines one action $a \in A$ that should be executed. Thereby, the following four approaches are evaluated for the cell diagnosis (c, ρ_T) over the set of root causes T (see Definition 6.1), the context $\mathbf{x}$ (see Definition 3.3), the recovery model RM (see Definition 6.3), and the SON function effects SFE (see Definition 4.11):

Simple degradation recovery aims to resemble a traditional, action policy-based approach. It solely considers the most likely root cause and tries to resolve it while ignoring the other root causes. Thereby, it does not consider the objectives, action costs, or action effectiveness. Ties are broken at random. So, given the most likely root cause $t \in T$, it selects any proposed action from the recovery model $\mathrm{RM}(c, t, \mathbf{x}, \mathrm{SFE})$ at random.

Probabilistic degradation recovery aims to resemble an approach that is aware of

```
ACTION triggerCcoRet IF true
  THEN (0.0, 0.0, 0.0)

ACTION restartBs IF cellLayer = macro
  THEN (0.0, 0.5, 1.0)
ACTION restartBs IF cellLayer = small
  THEN (0.0, 0.0, 1.0)

ACTION revertConfigurationChange IF cellLayer = macro
  THEN (0.0, 0.5, 1.0)
ACTION revertConfigurationChange IF cellLayer = small
  THEN (0.0, 0.0, 1.0)

ACTION revertSoftwareUpdate IF cellLayer = macro
  THEN (0.0, 0.5, 1.0)
ACTION revertSoftwareUpdate IF cellLayer = small
  THEN (0.0, 0.0, 1.0)

ACTION troubleTicket IF specialEvent = yes
  THEN (0.0, 0.0, 1.0)
ACTION troubleTicket IF specialEvent = no AND time = day
  THEN (0.0, 1.0, 1.0)
ACTION troubleTicket IF specialEvent = no AND time = night
  THEN (0.2, 1.0, 1.0)
```

Listing 7.2: Cost model.

the root cause probabilities and action effectiveness. Hence, it selects the action with the highest probability to resolve the problem. Ties are broken at random. So, it selects the action $a \in A$ that maximizes $\sum_{t \in T} \rho_T(t) \cdot z(t, a)$ with

$$z(t,a) = \begin{cases} \rho & \text{if } (a, \mathbf{f}^{\perp}, \rho) \in \mathrm{RM}(c, t, \mathbf{x}, \mathrm{SFE}) \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

Valuating degradation recovery solely considers the operational objectives but is unaware of probabilities. Consequently, it neglects the root cause probabilities as well as the effectiveness of the actions. In principle, it can be seen as a ODSO degradation recovery, however, with the following effect estimation for the recovery action proposal: First, the complete KPI effect for an action given a root cause (see Definition 6.4) is defined as $f_{k,a,t}(v) = \mu_k(\mathbf{f}^{\perp}(k), c, \mathbf{x}, 1.0)(v)$, hence, the action effectiveness is always 1.0. Second, the probability distribution of the cell diagnosis result is always $\rho_T(t) = {}^1/_{|T|}$ for all $t \in T$, i.e., each root cause is equally likely.

ODSO degradation recovery selects the first recovery action from the recovery action list RAL (see Definition 6.11).

Evaluation of Recovery Actions Finally, the recovery action proposed by a degradation recovery approach is evaluated against the true root cause. If the recovery

action can resolve the problem according to the recovery model, then the action is counted as effective, i.e., a correct action. The resulting utility vector is calculated based on the probabilistic merged effect of the action effect, the action effectiveness, and the operational context of the diagnosis case (see Definition 6.4). However, if the recovery action is not able to resolve the underlying true root cause according to the recovery model, then the action is counted as ineffective, i.e., an incorrect action, and the resulting utility vector is calculated based on the operational context of the diagnosis case, i.e., the problematic KPI values. Furthermore, the cost of the selected recovery action is recorded.

7.5.2.2 Results

The 1000 randomly created recovery cases split into 253 cases with an initial system state that is unacceptable, i.e., the utility of the KPI values in the artificially created operational context is unacceptable. Following the definition of the utility vector, at least one KPI value must be unacceptable for that. Furthermore, the majority of 747 cases has an acceptable initial system state and only one case is optimal, i.e., all KPI values are optimal. Given the probability distributions for the initial values in Figure 7.17, such a result is expected.

Figure 7.18 shows the raw, key performance metrics for the evaluation of the degradation recovery approaches:

Correct action ratio is calculated between the number of correct actions and the number of all diagnosis cases. A recovery action is considered correct if the recovery model proposes it as a counter measure for the true root cause.

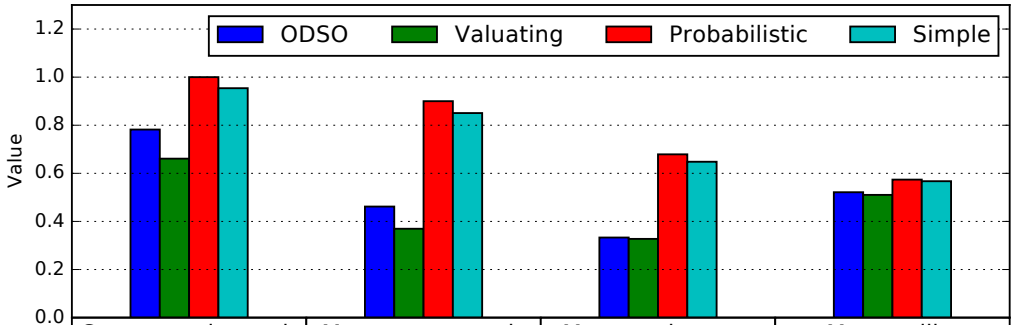
Mean recovery probability is calculated over the likeliness ρ of the selected recovery action a to recover the true root cause t in context $\mathbf{x}$ with $(a, \mathbf{f}^{\perp}, \rho) \in \mathrm{RM}(c, t, \mathbf{x}, \mathrm{SFE})$ for the common cell c, the recovery model RM, and the SON function effects SFE. In contrast to the correct action ration, this metric considers the action's effectiveness to recover the true root cause.

Mean action cost is the mean over a reduced form of the costs of the selected recovery actions. This reduced form cost is calculated as

$$\langle \mathbf{q} \rangle_{\mathbf{q}} = \begin{cases} \frac{q_3+q_2+q_1}{3} & q_3 = 1.0 \wedge q_2 = 1.0 \\ \frac{q_3+q_2}{3} & q_3 = 1.0 \\ \frac{q_3}{3} & \text{otherwise.} \end{cases}$$

for the action cost vector $\mathbf{q} = (q_1, q_2, q_3)$. As expected, the reduced form cost is higher the greater the cost for a high priority cost q_d with $d \in D$ is. For instance, $\langle (0.0, 0.0, 1.0) \rangle_{\mathbf{q}} = 0.33$, $\langle (0.0, 0.5, 1.0) \rangle_{\mathbf{q}} = 0.50$, and $\langle (0.2, 1.0, 1.0) \rangle_{\mathbf{q}} = 0.73$. As can be seen, it is inverse to the reduced utility defined in Definition 7.1.

Mean utility is the mean over the reduced form, expected utility vectors of the selected recovery actions.



	Correct action ratio	Mean recov. prob.	Mean action cost	Mean utility
ODSO	0.78	0.46	0.33	0.52
Valuating	0.66	0.37	0.33	0.51
Probabilistic	1.0	0.9	0.68	0.57
Simple	0.95	0.85	0.65	0.57

Figure 7.18: Correct action ratio, mean recovery probability, mean action cost, and mean utility of the simulation for the four degradation recovery approaches.

The inspection of the results show that the probabilistic degradation recovery yields the highest correct action ratio of 1.0 and mean recovery probability of 0.90. This is not surprising as this approach focuses solely on the recovery probability. An inspection of the cases reveals the logical result of this goal: it always triggers a trouble ticket since this can recover all root causes and has the highest effectiveness. Consequently, the mean action cost of 0.68 is also the highest.

The simple degradation recovery produces a similar behavior since it is focused on resolving solely the most likely root cause. Hence, it selected the trouble ticket in 914 cases and tried some other action in 86 cases. With this approach it was in 46 cases wrong and did not recover the problem, leading to a slightly reduced recovery probability 0.85 and minor decrease of the action cost to 0.65.

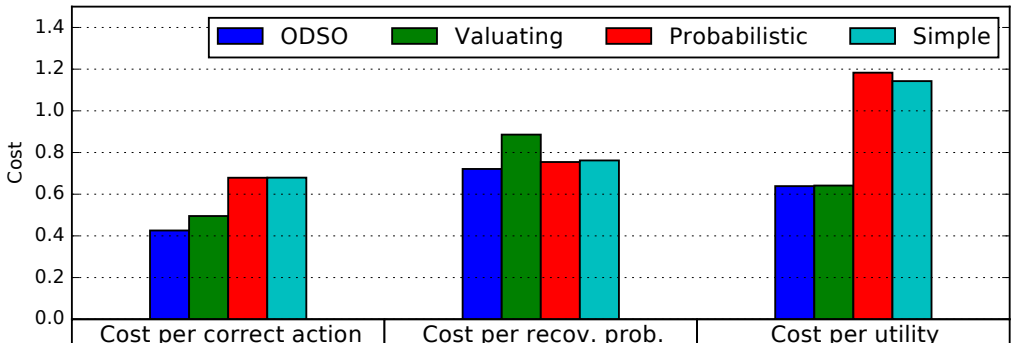
In contrast to that, the valuating degradation recovery picks a possible action solely based on the optimal outcome without considering any probabilities. As a result, it often picks the cheapest action since their effects, expect for CCO-RET, are all the same. This resulted in 624 cases where it decided to restart the BS, 358 cases where it reverted a software update, and 18 triggered CCO-RETs. Notice that a trouble ticket was never issued. This produced a small mean cost of 0.33, however, also a very low correct action ratio and recovery probability of 0.66 and 0.37 respectively.

The probabilistic and valuating degradation recovery represent two extreme recovery approaches: recover the problem at all cost versus recover as cheap as possible. The ODSO approach aims to find a middle way. It is interesting to see that the simulation results underpin this. The correct action ratio is almost halfway between both approaches at 0.78 and the recovery probability is at 0.46. Interestingly, the mean cost of 0.33 is about the same as the expected minimal cost by the valuating approach. This is caused by an intelligent mix of recovery actions: 741 restarts, 136 trouble tickets and 115 reverted software updates, but also some reverted configu-

rations and triggered CCO-RETs.

Interestingly, the achieved expected utility of the recovery actions does not differ that much. The reason for this is that no recovery action is able to recover a problem with certainty and, hence, the utility stays in the same priority before and after the action execution. So, if the utility of the initial system state is unacceptable then the expected system state of the trouble ticket is still unacceptable since there is a slight probability that it is not effective. The fact that it is most likely that the system state is optimal after the execution of a trouble ticket is only weakly represented by the reduced form of the utility used here. Later, we present the results of an adapted scenario where all action have an effectiveness of 1.0. Nevertheless, the result shows the expected distribution that the probabilistic degradation recovery produces the highest utility, the valuation approach the lowest utility, and ODSO is in between.

Degradation recovery has to make a trade-off between maximizing the recovery probability and minimizing the action cost. Of course, every operator has to decide what is more important. In ODSO, this operational goal is represented by the cost model. In order to show that for the given cost model, the ODSO degradation recovery makes rational decisions, Figure 7.19 shows the ratios between the mean action cost, and the three antagonist performance metrics correct action ratio, mean recovery probability and mean utility. In principle, they can be interpreted as the mean cost of selecting a correct recovery action, the mean cost of generating 100% mean recovery probability, and the mean cost of generating a mean utility of 1 respectively. These are artificial metrics, however, they give an indication of how much each approach invests (in terms of action costs) for a return (in terms of the other three performance metrics). Consequently, the ratios shown in Figure 7.19 should be as low as possible for a rational decision maker.



	Cost per correct action	Cost per recov. prob.	Cost per utility
ODSO	0.43	0.72	0.64
Valuating	0.5	0.89	0.64
Probabilistic	0.68	0.75	1.18
Simple	0.68	0.76	1.14

Figure 7.19: Cost per correct action ratio, cost per recovery probability ratio, and cost per utility ratio of the simulation for the four degradation recovery approaches.

As can be seen, for all three ratios, the ODSO degradation recovery achieves the lowest value of the four decision making approaches. This shows that it, indeed,

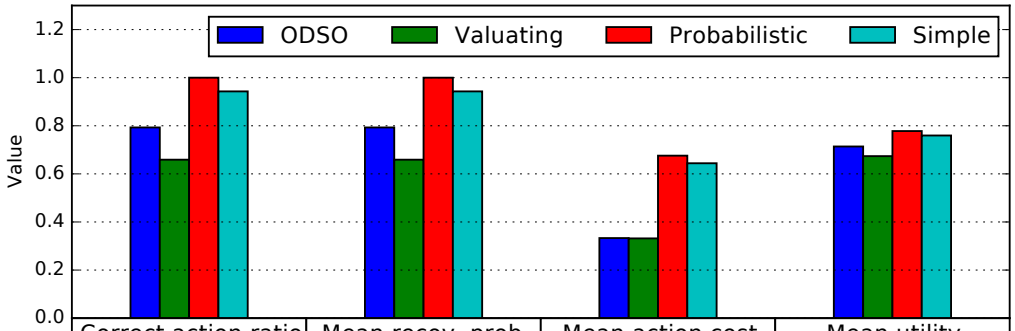
is a rational, autonomic decision maker. As expected, the probabilistic approach performs best of the rest regarding the cost per recovery probability as it focuses on achieving a high return in this discipline. Consequently, the valuating approach is best of the rest regarding the cost per utility for the same reason. In both cases, the difference to the ODSO approach is not much. However, considering all three ratios, the ODSO is on average 13% better than the simple approach, 50% better than the probabilistic approach, and 48% better than the valuating approach (calculated as the arithmetic mean over the ratios for all three cost per performance metrics).

In order to complete the previous discussion about the low achieved utility, Figure 7.20 depicts the results for the very same scenario, however, all actions have an effectiveness of one. As can be seen, the utility of the probabilistic agent increases to a value greater than 0.66 which is optimal. This value is actually the expected utility of the SON configuration. Nevertheless, the general message of the evaluation, particularly that ODSO degradation recovery achieves the lowest values for the cost per return ratios, is similar to the previous simulation.

7.6 Summary and Discussion

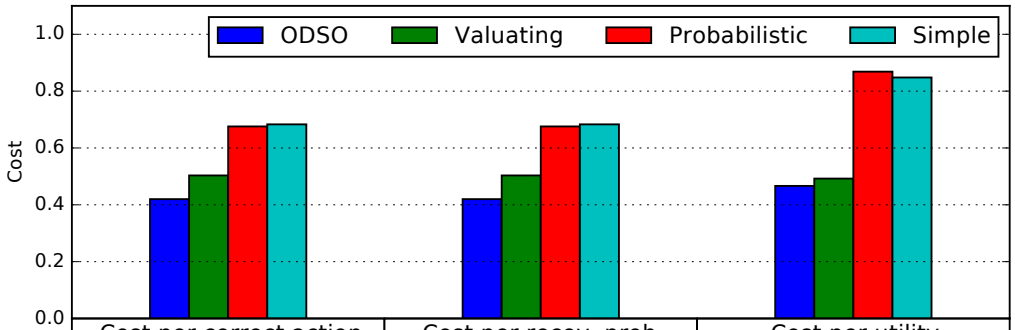
This chapter showed the feasibility of the ODSO approach for SON operations and its advantages in terms of achievable performance improvements compared to manual or partially automated SON operations. For each considered SON operations task, we have simulated a specific scenario that allowed us to show the differences in the decision making and the efficiency of the ODSO approach. Thereby, it became obvious that the performance improvements by ODSO did not emerge from new, improved SON functions but instead from a better utilization of the given SON functions. That is, ODSO often resembled what a human operator would have also achieved in a similar situation. However, this manual operations requires enormous efforts which are typically avoided and, hence, this potential is not lifted. ODSO automates this decision making, solely controlled by the actual operational objectives formalized in the objective model. As a result, we have shown that the automatic adoption of the SON configuration, conflict resolution and degradation recovery according to the actual operational objectives promises considerable performance improvements in a mobile network. Notice, that these results also validate to some extent the simplifications made for keeping the ODSO approach computationally tractable, i.e., specifically the probabilistic independence of the KPIs effects (see Chapter 3.4.1.2), the probabilistic combination of the effects of the SFCs (see Chapter 4.3.3.2), and the estimation of the short-term effects of a SON function request from the long-term SON function effects (see Chapter 5.3.1.1).

The evaluation of SON management shows that the ODSO component is able to adapt the SON configuration to varying operational objectives. Thus, it achieves an overall higher satisfaction of the operational objectives, i.e., better network performance, in different network contexts than any fixed SON configuration. Thereby, the operator solely defines different weights for the KPIs that represent the focus of network operations at different points in time. This shows that even adaptive



	Correct action ratio	Mean recov. prob.	Mean action cost	Mean utility
ODSO	0.79	0.79	0.33	0.71
Valuating	0.66	0.66	0.33	0.67
Probabilistic	1.0	1.0	0.68	0.78
Simple	0.94	0.94	0.64	0.76

(a) Correct action ratio, mean recovery probability, mean action cost, and mean utility.



	Cost per correct action	Cost per recov. prob.	Cost per utility
ODSO	0.42	0.42	0.47
Valuating	0.5	0.5	0.49
Probabilistic	0.68	0.68	0.87
Simple	0.68	0.68	0.85

(b) Cost per correct action ratio, cost per recovery probability ratio, and cost per utility ratio.

Figure 7.20: Results of the adapted simulation with common action effectiveness of 1.0 for the four degradation recovery approaches.

and sophisticated SON functions like CCO-SCA and ESM need to be constantly aligned with the operator objectives to achieve the best results. The evaluation of the ODSO SON coordination component showed that rational scheduling of conflicting SON function requests may lead to a faster achievement of a specific level of objective satisfaction. Although the actual schedule barely impacts the time necessary to overcome a set of performance issues in the network, we showed that ODSO focuses particularly on the worst problems regarding the operator objectives that can be overcome quickly, and orders the SON function execution accordingly. Finally, the two-fold evaluation of ODSO self-healing showed two aspects: First, we have shown that information about the SON, particularly the activity of the SON functions, provides valuable information to support the detection and diagnosis of failures in the network, e.g., ambiguous problems like a sleeping cell. Second, we outlined that the ODSO approach for degradation recovery selects the most rational recovery actions in a probabilistic and stochastic environment in comparison to simpler and often more intuitive approaches. This advantage manifests in the fact that it recovered a high number of simulated failure cases while also achieving a low cost of troubleshooting.

Although the evaluation results are promising, it is also important to notice the limits of the approach: it can only make accurate decisions if the underlying knowledge, i.e., the technical models, is accurate. In other words, the performance benefits of ODSO depend to a considerable degree on the quality of the SON function models, coordination models, and self-healing models. Inaccurate technical models may lead to non-optimal or, in the worst case, even degraded network performance. This has been shown in the evaluation of SON management (see Chapter 7.3.2): in the load-focused period with a low user population (RL), the ODSO SON management component selected a load-focused configuration for the SON functions CCO-SCA and ESM. However, in the specific situation, this turned out to be non-optimal, as the user throughput could have also been provided with an energy-focused SON configuration which achieved a higher operational objective satisfaction. Furthermore, SON management can just select from the SON function configurations provided in the SON function models. Hence, the basic question is whether the optimal SON function configuration for a specific situation is actually contained in the model and its effects accurately defined? As the SON function effects are also utilized by the ODSO SON coordination and SON self-healing components, inaccurate SON function models also affect their decisions. Especially regarding SON coordination, Chapter 5.3.1.1 already discussed the issue that the SON function effects represent long-term effects whereas SON coordination makes short-term decisions. Consequently, if the long-term and short-term effects diverge considerably, SON coordination will make non-optimal decisions. In the evaluation, however, we did not encounter such a situation. Nevertheless, if this fact would be known by the operator or SON function vendor, it is possible to adapt the SON function effects prior to their utilization. Similar to the SON function models, the accuracy of the recovery model strongly affects the performance of ODSO SON self-healing. Unfortunately, in this case it might be even more difficult to provide better models as such failures in the network seldom happen and are hardly foreseeable.

The evaluation also revealed the need for a future extension of the ODSO concept: the effects of an action may affect not solely a single network cell but a group of cells. This became apparent for the CCO and ESM functions as discussed in Chapter 7.2.2: both SON functions focus on small cells, however, they mainly affect KPIs in the covering macro cell. Although we presented an elegant solution in the evaluation, this problem is more fundamental and needs to be reflected in the general concept accordingly. A similar situation may occur regarding MLB as it reduces the cell load in its target cell by actually increasing the cell load in the neighboring cells.

These limitations of the ODSO approach, however, are not surprising and we have identified them during the development of the concept. Decision making, in general, depends on the accuracy of the available information as well as the conscious and structured evaluation of this knowledge in the light of explicit objectives. In this work, we have focused on the latter evaluation of the models while also structuring and designing the system to allow further extensions for improving the former part. Specifically, the presented ODSO approach is agnostic regarding the creation and representation of the technical models. As mentioned in Chapter 3.4.1.3, Chapter 4.3.2, and Chapter 6.3.3.3, the presented action-policy based technical models are one option to model the expertise. Future research may, e.g., develop new concepts that involve the utilization of machine learning techniques (see Chapter 8.2). These can be easily integrated into the ODSO approach as long as the formalized interfaces to the models, e.g., Definition 4.2, are adopted.

8 Conclusion and Outlook

This thesis introduced the Objective-Driven SON Operations (ODSO) concept that enables *autonomic* Self-Organizing Network (SON) operations, i.e., the control of network operations through operational objectives on network Key Performance Indicators (KPIs). This chapter summarizes the presented contributions and results, and discusses possible future work.

8.1 Summary

This thesis presented a concept for objective-driven, autonomic operations of SON systems regarding the tasks SON management, SON coordination, and SON self-healing. In summary, objectives for its development have been (see Chapter 1.1):

- overcoming the manual gap of SON operations by enabling an integrated, objective-driven control of a SON (Objective 1),
- autonomically configure the SON functions according to the operational objectives by using SON function vendor-provided technical models (Objective 2),
- autonomically resolve SON function conflicts at run-time such that the operational objectives are satisfied (Objective 3), and
- involving the SON in SON self-healing as well as selecting recovery actions for degradations with respect to the operational objectives and action preferences (Objective 4).

By solving these objectives, the development of ODSO aims to, on the one hand, reduce the manual efforts and Operational Expenditure (OPEX) necessary for the operation of a mobile network, and, on the other hand, lift the full optimization potential of SON by aligning it closer with the operational objectives. In the following, we summarize our solutions for these objectives and discuss our contributions and results that go beyond the state of the art in mobile network operations.

Objective-Driven SON Operations: Architecture and Component Design

Chapter 3 presented a holistic, integrated architecture for autonomic SON operations controlled through a single, high-level interface in form of an objective model. This

architecture extends a SON by introducing objective-driven SON management, SON coordination, and SON self-healing components which are all based on a generic component design for autonomic decision making (see Solution and Contribution 1). Together, these contributions are able to close the manual gap of SON operations and set ODSO apart from related work that either focuses on automating single SON functions or single operations tasks, or providing automated operations frameworks that do not consider operational objectives.

The introduced ODSO architecture assigns each of the three SON operations tasks to a dedicated component. The components are integrated into a framework that allows them to interact with each other in order to share information, e.g., about SON function effects which represent the expected effects of the currently active SON functions. The most important feature of the framework is, however, that the three components are all controlled through a common, network-wide objective model that encodes the operational objectives regarding the KPIs of the Mobile Network Operator (MNO) in a formalized, vendor-independent manner.

Each ODSO component is based on the generic component design for objective-driven operations. It is a two-step decision making process that separates the computation of possible action options and their probabilistic effects from the assessment of these options based on the common objective model. The former is based on the task-specific processing of technical models. The latter is based on Multiattribute Utility Theory (MAUT) which provides a theoretically-founded, valid and favorable decision making approach that enables a high degree of autonomic behavior. Thereby, we extend classical MAUT to enable the objective model expressing context-dependent priority regions and utility functions for the KPI values as well as trade-offs between them.

The ODSO concept focuses on decision making based on technical models that formalize the technical expertise of human operators. The evaluation in Chapter 7 showed that it enables an automation of manual SON operation and facilitates a control of the SON-enabled mobile network through the objective model. Thereby, the achievable improvements in network performance are not due to improved SON functions or network optimization algorithms but due to a better alignment of the SON function execution with the objectives. This approach lifts additional optimization potential whose manual achievement would require considerable manual efforts. However, it is also shown that the results of ODSO depend on the accuracy of the technical models and their processing, which are strongly connected to the specific SON operations tasks. This may require additional research on the creation and validation of these models. Nevertheless, the architecture and design of the ODSO concept has been created to support such extensions.

SON Management

Chapter 4 presented the details of the SON management ODSO component that autonomically configures the SON functions according to the objective model (see Solution and Contribution 2). Based on the generic ODSO component design it, first, computes feasible SON configurations end their expected effects, and, second,

evaluates the configurations against the objective model in order to determine the optimal configuration.

In order to compute the feasible SON configurations, we define a semantics for the technical models describing diverse SON functions. It allows the vendors to protect their intellectual property by keeping the algorithms closed while the operators are enabled to estimate the expected KPI values for each SON Function Configurations (SFCs). The result is the definition of a SON function model for each SON function that provides possible SFCs and their expected, probabilistic effects on the KPIs. SON management combines the independent SFCs for each SON function to a SON configuration and, thereby, analyzes the SFCs for possible conflicts, i.e., SON functions with incompatible optimization goals. Furthermore, it computes the overall expected KPI effects based on a probabilistic estimation without requiring any additional input by the human operator, thus, keeping the manual efforts for the operation of ODSO low.

In Chapter 7.3, we have evaluated this approach in a simulation of a Heterogeneous Network (HetNet) scenario that involved different settings for a specialized Coverage and Capacity Optimization (CCO) and an Energy Saving Management (ESM) function. We were able to show that the continuous adaptation of the SFCs by ODSO SON management, which is driven by varying operational objectives, is able to gain a higher satisfaction of the objectives compared to a fixed uniform SON configuration. However, the evaluation also revealed that inaccurate SON function models may lead to non-optimal or even degraded network performance.

SON Coordination

Chapter 5 introduced the SON coordination ODSO component that focuses on autonomic conflict resolution between SON function requests according to the objective model (Solution and Contribution 3). Based on the generic ODSO component design it, first, estimates the expected performance effects of each SON function request, determines the conflicts between the requests, and performs a technical conflict resolution. Thereby, the component reuses information about the expected effects of the SON configuration provided by the SON management component in order to estimate the effects of each single SON function execution without requiring additional manual input. Second, the remaining SON function conflicts are resolved based on a valuation of the effects according to the objective model. In order to ensure an optimal decision making even if the detected conflicts are complex and not transitive, we outline an approach to model the final selection problem with Goal Programming (GP) such that it can be computed using standard solvers. As a result, ODSO SON coordination orders conflicting SON function requests, i.e., it schedules them, such that the operator objectives are satisfied as quickly as possible.

In Chapter 7.4, we have evaluated this approach in a simulation of a coordination problem involving several conflicting SON functions. We were able to show that the objective-driven resolution of conflicts leads to a schedule that achieves an acceptable level of objective satisfaction faster than a state-of-the-art, rule-based coordination approach. Thereby, the overall number of executed actions by both

approaches is similar, but their execution order differs. Nevertheless, inaccurate technical models or incorrect effect estimations in complex scenarios may lead to non-optimal schedules. However, this was not revealed in the evaluation.

SON Self-Healing

Chapter 6 presented the SON self-healing ODSO component. Our contributions to the state of the art are twofold (Solution and Contribution 4): on the one hand, we propose an approach for involving the SON functions in the detection of degradations, and, on the other hand, we present a concept for autonomic degradation recovery based on a probabilistic diagnosis of the detected problems.

The involvement of the SON is achieved by either utilizing SON functions as active probes that can raise alarms once they detect an abnormal situation, or by monitoring and analyzing the execution of SON functions for indications of abnormal behavior. In this way, it is possible to detect a wider range of network degradations and to provide the subsequent root cause diagnosis with additional symptom data that may increase its accuracy.

Autonomic degradation recovery is based on the generic ODSO component design. First, it computes applicable recovery actions for a diagnosed problem and estimates their effects based on a recovery model and the expected normal network performance provided by SON management. Thereby, it also considers the likeliness of the different root causes for the problem. Second, the effects are evaluated with respect to the operator objectives on the KPIs, but also considering the MNO's preferences regarding the recovery actions as expressed in a cost model. In summary, this can be seen as the first comprehensive approach for objective-driven degradation recovery in SON.

In Chapter 7.5.1, we have shown that it is possible to support the detection and diagnosis of sleeping cells in a realistic network by monitoring the execution of SON functions. In Chapter 7.5.2, we showed that objective-driven decision making for a recovery action in response to a probabilistic failure diagnosis, as it is performed by ODSO-based degradation recovery, is able to optimally trade-off the recovery effectiveness with the recovery efforts and problem severity. This has been shown by comparing the ODSO approach with several related degradation recovery methods. It must be noted, though, that the achievable results depends on the quality of the technical models. As a result of the autonomic self-healing approach, the time from the occurrence of a failure to its recovery can be reduced.

8.2 Future Work

ODSO can be seen as a new approach in network management that goes beyond a pure evolution of established approaches. Consequently, the concept requires a multitude of new processes and artifacts that have not been there before. In this thesis, we focused on the development of the overall concept for autonomic decision making and, thus, introduced assumptions in order to keep the scope of the thesis

consistent. In the following, we present some ideas for future work in the context of the ODSO concept that may substantiate these assumptions.

Increasing Accuracy of Technical Models

Chapter 7.6 describes a main limitation for the applicability of the ODSO approach: the resulting network performance strongly depends on the accuracy of the technical models. ODSO as an autonomic decision making approach makes optimal decisions based on the formalized technical expertise in these models. However, if the models are not correct, then ODSO is likely to configure, coordinate, and recover a SON poorly leading to inferior network performance or even performance degradation.

Therefore, we see the topic of knowledge elicitation and technical model creation as a focus area in future research on SON operations. Thereby, the utilization of machine learning methods [RN10] is a very promising approach as it enables the automation of this task without great manual efforts (see Chapter 4.3.2). Especially, Reinforcement Learning (RL) [SB12] has shown a wide range of application for problems like this. Thereby, the system follows a trial-and-error method, i.e., it selects some action to execute, monitors the resulting effects in the concrete environment, and learns a technical model from this. This model could be very accurate as it specifically contains the effects in the real mobile network. There have already been some initial studies on the application of this method for SON management in the SElf-MAnagement FOr Unified heterogeneous Radio access networks (SEMAFOUR) project [Cam+15][Kür+15] which are expected to be continued in the future. Furthermore, the project developed a detailed concept for using RL in SON coordination [Sch+14a][Göt+15] which could be adapted to fit into the ODSO approach. Apart from the promises of RL for the creation of the technical model, there is a major open question that needs to be answered for a real application. The trial-and-error approach introduces some risk that the tried action might be wrong and deteriorate the network performance. Hence, it is necessary to develop a concept that allows the operator to trade-off the risk of inferior decisions and the potential gains from more accurate knowledge. The concept of trust developed in the UniverSelf project [Cia+13] may be a starting point for this.

It is also worth to invest some efforts on research for methods for the elicitation of technical knowledge from human engineers and operators for the manual creation of technical models. Manually created technical model are useful for two reasons: First, they provide a starting point for the ODSO system, i.e., a rough estimation of the action effects, for the initial deployment when machine-learned knowledge is not available yet. Second, some actions may be rarely executed, e.g., some self-healing workflows as the respective failure may be uncommon, which makes learning the effects very difficult. In this case the estimation by a knowledgeable expert human operator is a very valuable input.

Another direction to increase the accuracy of the technical models is to extend the scope of the effects of an action. In the presented ODSO concept, an action is tight to a network cell and its effects only apply to the same cell. However, as outlined in Chapter 7.6, some SON functions may also affect neighboring cells. Such

an approach increases the complexity of SON management considerably as it is not possible to determine a SON configuration for each network cell separately. Instead, it must find an overall optimal configuration similarly to SON coordination which needs to find an overall best subset of SON function requests to execute. Future work will need to show whether this problem is computationally tractable and, if it is intractable, how a good solution may be approximated.

Elicitation of Objective Model

Besides the technical models, the ODSO system requires an objective model which represents the preferences of the operator. Similar to a manually elicited technical model, it is an open research question which method is best to create the objective model. As shown in Chapter 2.3.3.1, there is substantial related work on this topic, e.g., the Analytic Hierarchy Process (AHP). However, it still needs to be determined which one of the numerous approaches fits the decision problems in SON operation and the background of the operational personnel best.

New Application Areas

We have presented ODSO as an approach for making SON autonomic by automating SON operations. Although the ODSO architecture with the three components SON management, SON coordination, and SON self-healing as well as their interactions is quite specific to SON, the generic component design, i.e., the general structure of the objective and technical models as well as the decision making is not SON dependent. Therefore, it seems natural to attempt applying the generic component design to other application areas in mobile networks management.

A current trending topic, specifically in combination with the upcoming Fifth Generation (5G) mobile networks, is network virtualization [BDW14]. Thereby, MNOs attempt to virtualize their Radio Access Network (RAN) and core network and create a cloud-based infrastructure that dynamically provides network services when and where they are needed. Thereby, the network functions are virtual instances that are not tied to a specific server but that can be deployed dynamically to any data center that the MNO can access. The difficulty in this approach is to decide which network functions should be deployed and where they should be running. The decision making could be automatically controlled using an action policy, however, just like for SON, an autonomic approach, which is controlled through an objective model, promises a higher degree of automation and possibly better network performance. The generic ODSO component design may provide a blueprint for future research on this topic.

Beyond SON: Cognitive Network Operations

Currently, SON functions implement the operational procedures often in form of fixed rules and calculations. Hence, the operators need to adapt those SON functions to the specifics of the actual mobile network and the operational objectives.

However, there are some attempts to make the SON functions more intelligent by employing machine learning methods [RN10] such that SON functions are enabled to adapt themselves to the network characteristics. In order to guide such learning, the SON functions need to be given a target that they aim to achieve. This development leads to an evolution of the SON concept that is currently referred to as *cognitive functions* [Kür+15]. In other words, cognitive functions can be seen as an extension of SON functions that, first, understand the operational objectives of the MNO, and, second, are able to adapt themselves to the network characteristics in order to achieve these objectives autonomically.

Figure 8.1 depicts a characterization of manual network operations, SON, ODSO, and cognitive functions. As can be seen, manual network operations, SON and ODSO (without the extensions discussed in this chapter) rely on hard-coded knowledge in form of manuals, rules, or technical models. However, SON increases the level of automation compared to manual operations by introducing automatic decision making, and ODSO extends this to autonomic network management. As can be seen, cognitive functions are also considered an autonomic mobile network operations approach, but with adaptive knowledge that is automatically created using machine learning methods.

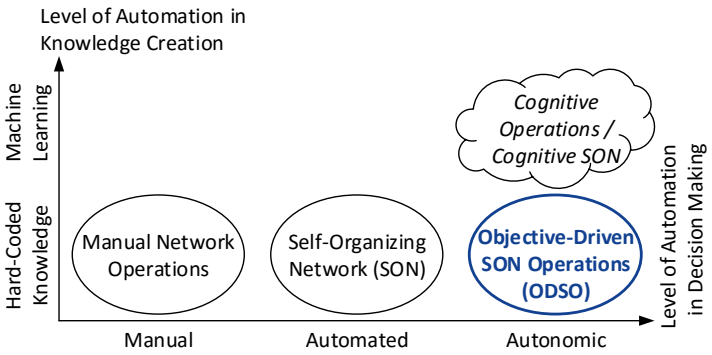


Figure 8.1: Characterization of manual network operations, SON, ODSO, and cognitive functions with respect to automation of decision making and knowledge creation.

The generic ODSO component design may provide a valuable blueprint for the construction of an autonomic decision making procedure in cognitive functions. However, the architecture of the operational tasks will change considerably as the centralized control performed in ODSO is substituted with distributed control. Related to SON management, this means that the objectives are directly provided to the cognitive functions. Based on this, the cognitive functions may need to exchange information with each other in order to detect possibly competing objectives and align their behavior such that conflicts are avoided and the satisfaction of the operational objectives is maximized. Similarly, the cognitive functions may inform each other about the actions they are going to execute in order to detect and resolve potential conflicts among those actions in a distributed manner. The same applies

to self-healing. However, research on the cognitive function concept just started and it is not clear how these tasks will be performed.

Bibliography

[2op10] 2operate. *2solve - Troubleshooting Process Automation.* White Paper. 2operate, 2010.

[3GP11] 3GPP. *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions (Release 9).* Technical report TR 36.902 V9.3.1. 3rd Generation Partnership Project, 2011.

[3GP12] 3GPP. *Technical Specification Group Services and System Aspects; Telecommunication Management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Requirements (Release 11).* Technical specification TS 32.521 V11.1.0. 3rd Generation Partnership Project, 2012.

[3GP13] 3GPP. *Technical Specification Group Services and System Aspects; Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS) (Release 11).* Technical specification TS 32.522 V11.7.0. 3rd Generation Partnership Project, 2013.

[3GP14a] 3GPP. *Technical Specification Group Services and System Aspects; Telecommunication management; Energy Saving Management (ESM); Concepts and requirements (Release 12).* Technical specification TS 32.551 V12.0.0. 3rd Generation Partnership Project, 2014.

[3GP14b] 3GPP. *Technical Specification Group Services and System Aspects; Telecommunication Management; Self-Organizing Networks (SON); Concepts and requirements (Release 12).* Technical specification TS 32.500 V12.1.0. 3rd Generation Partnership Project, 2014.

[3GP14c] 3GPP. *Technical Specification Group Services and System Aspects; Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements (Release 12).* Technical specification TS 32.541 V12.0.0. 3rd Generation Partnership Project, 2014.

[3GP15a] 3GPP. *Technical Specification Group GSM/EDGE Radio Access Network; Overall description; Stage 2 (Release 13).* Technical specification TS 43.051 V13.0.0. 3rd Generation Partnership Project, 2015.

[3GP15b] 3GPP. *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2 (Release 13).* Technical specification TS 36.300 V13.1.0. 3rd Generation Partnership Project, 2015.

[3GP15c] 3GPP. *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (Release 12).* Technical specification TS 36.213 V12.7.0. 3rd Generation Partnership Project, 2015.

[3GP15d] 3GPP. *Technical Specification Group Radio Access Network; UTRAN overall description (Release 13).* Technical specification TS 25.401 V13.0.0. 3rd Generation Partnership Project, 2015.

[3GP15e] 3GPP. *Technical Specification Group Services and System Aspects; Network architecture (Release 13).* Technical specification TS 23.002 V13.4.0. 3rd Generation Partnership Project, 2015.

[3GP15f] 3GPP. *Technical Specification Group Services and System Aspects; Telecommunication management; Performance Management (PM); Concept and requirements (Release 13).* Technical specification TS 32.401 V13.0.0. 3rd Generation Partnership Project, 2015.

[3GP15g] 3GPP. *Technical Specification Group Services and System Aspects; Telecommunication management; Performance Management (PM); Performance measurements Evolved Universal Terrestrial Radio Access Network (E-UTRAN) (Release 13).* Technical specification TS 32.425 V13.2.0. 3rd Generation Partnership Project, 2015.

[3GP16] 3GPP. *3GPP homepage.* 2016. URL: `http://www.3gpp.org` (visited on 02/01/2016).

[3GP98] 3GPP. *Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.2.0).* Technical report TR 101 112 V3.2.0. 3rd Generation Partnership Project, 1998.

[AB07] Issam Aib and Raouf Boutaba. "Business-Driven Optimization of Policy-Based Management solutions". In: *2007 10th IFIP/IEEE International Symposium on Integrated Network Management.* IEEE, May 2007, pp. 254–263.

[Ago11] Nazim Agoulmine, ed. *Autonomic Network Management Principles: From Concepts to Applications.* Burlington, USA: Elsevier, 2011.

[Ali+13] Osianoh Glenn Aliu et al. "A Survey of Self Organisation in Future Cellular Networks". In: *IEEE Communications Surveys & Tutorials* 15.1 (Jan. 2013), pp. 336–361.

[Alt+13] Zwi Altman et al. *Deliverable D4.11: Leaflet of the third prototype of a use-case.* Deliverable. UniverSelf Project, June 2013.

[Alt+14] Zwi Altman et al. "On design principles for self-organizing network functions". In: *2014 11th International Symposium on Wireless Communications Systems (ISWCS).* IEEE, Aug. 2014, pp. 454–459.

[Ami+09] Mehdi Amirijoo et al. "Cell outage management in LTE networks". In: *2009 6th International Symposium on Wireless Communication Systems.* IEEE, Sept. 2009, pp. 600–604.

[Ant+99] C. Henggeler Antunes et al. "A multiple objective routing algorithm for integrated communication networks". In: *International Teletraffic Congress (ITC-16)*. Elsevier, June 1999.

[ARC15a] ARCEP. *Couverture Mobile: le suivi des déploiements des opérateurs*. 2015. URL: `http://www.arcep.fr/index.php?id=8161%7B%5C&%7DL=1` (visited on 08/20/2015).

[ARC15b] ARCEP. *Observatory / Couverture et qualité des services mobiles*. 2015. URL: `http://www.arcep.fr/index.php?id=12849%7B%5C&%7DL=1` (visited on 08/20/2015).

[AT16] Janne Ali-Tolppa and Tsvetko Tsvetkov. "Optimistic Concurrency Control in Self-Organizing Networks Using Automatic Coordination and Verification". In: *2016 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, Apr. 2016. accepted paper.

[BA07] Raouf Boutaba and Issam Aib. "Policy-based Management: A Historical Perspective". In: *Journal of Network and Systems Management* 15.4 (2007), pp. 447–480.

[Bal+08] Javier Baliosian et al. "Policy-based self-healing for radio access networks". In: *Proc. IEEE Network Operations and Management Symposium (NOMS 2008)*. Salvador, Bahia, Brazil, Apr. 2008, pp. 1007–1010.

[Ban+04] Arosha K. Bandara et al. "A goal-based approach to policy refinement". In: *5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*. Yorktown Heights, NY, USA: IEEE, June 2004, pp. 229–239.

[Ban+05] Arosha K. Bandara et al. "Policy refinement for diffserv quality of service management". In: *9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005)*. Nice, France: IEEE, May 2005, pp. 469–482.

[Ban+11a] Tobias Bandh et al. "Policy-based coordination and management of SON functions". In: *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011)*. Dublin, Ireland: IEEE, May 2011, pp. 827–840.

[Ban+11b] Tobias Bandh et al. "SON Operation". In: *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, 2011. Chap. 9, pp. 322–356.

[Ban11] Tobias Bandh. *The SOCRATES SON-Function Coordination Concept*. Technical Report. Nov. 2011. URL: `http://net.in.tum.de/fileadmin/bibtex/publications/papers/socrates.pdf` (visited on 02/15/2016).

[Ban13] Tobias Bandh. "Coordination of autonomic function execution in Self-Organizing Networks". PhD Thesis. Technische Universität München, Apr. 2013.

[Bar+06] Raquel Barco et al. "Comparison of probabilistic models used for diagnosis in cellular networks". In: *2006 IEEE 63rd Vehicular Technology Conference* 2 (2006).

[Bar+13a] Enda Barrett et al. *Specification of cognitive network management framework and functions.* Deliverable D3.1. COMMUNE Project, Nov. 2013.

[Bar+13b] Enda Barrett et al. *Specification of knowledge-based reasoning algorithms Project.* Deliverable D4.1. COMMUNE Project, Dec. 2013.

[Bar07] Raquel Barco. "Bayesian modelling of fault diagnosis in mobile communication networks". PhD thesis. 2007.

[Bau+12] Bernhard Bauer et al. "Resource-oriented Consistency Analysis of Engineering Processes". In: *Proceedings of the 14th International Conference on Enterprise Information Systems.* SciTePress, June 2012, pp. 206–211.

[BB08] Raphael M. Bahati and Michael A. Bauer. "Reinforcement learning in policy-driven autonomic management". In: *Proc. IEEE Network Operations and Management Symposium (NOMS 2008).* Salvador, Bahia, Brazil, Apr. 2008, pp. 899–902.

[BD09] Ronen I. Brafman and Carmel Domshlak. "Preference Handling - An Introductory Tutorial". In: *AI Magazine* 30.1 (2009), pp. 58–86.

[BDW14] Henrik Basilier, Marian Darula, and Joe Wilke. "Virtualizing network services - the telecom cloud". In: *Ericsson Review* 91 (Mar. 2014), pp. 2–9.

[Ben+12] Leila Bennacer et al. "Optimization of fault diagnosis based on the combination of Bayesian Networks and Case-Based Reasoning". In: *2012 IEEE Network Operations and Management Symposium.* IEEE, Apr. 2012, pp. 619–622.

[Ben+13a] Sana Ben Jemaa et al. *Definition of Requirements for a Unified Self-Management System.* Deliverable D2.2. SEMAFOUR Project, July 2013.

[Ben+13b] Sana Ben Jemaa et al. *Integrated SON Management - Requirements and Basic Concepts.* Deliverable D5.1. SEMAFOUR Project, Dec. 2013.

[Ben+13c] Leila Bennacer et al. *Deliverable D3.9: Handbook on optimization, learning, operation and cooperation methods.* Tech. rep. UniverSelf Project, Aug. 2013.

[BLM12] Raquel Barco, Pedro Lazaro, and Pablo Munoz. "A unified framework for self-healing in wireless networks". In: *IEEE Communications Magazine* 50.12 (Dec. 2012), pp. 134–142.

[BR12] Tobias Bandh and Raphael Romeikat. "An Integrated SON Experimental System for Self-Optimization and SON Coordination". In: *Proc. 2nd International Workshop on Self-Organizing Networks (IWSON 2012)*. Paris, France, Aug. 2012.

[BST06] Claudio Bartolini, M. Salle, and David Trastour. "IT service management driven by business objectives An application to incident management". In: *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*. Vancouver, Canada: IEEE, Apr. 2006, pp. 45–55.

[Bun15] Bundesnetzagentur. *Jahresbericht 2014*. Tech. rep. Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, 2015.

[Cam+15] Luis Miguel Campoy Cervera et al. *Integrated SON Management Implementation Recommendations*. Deliverable D5.4. SEMAFOUR Project, Aug. 2015.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection". In: *ACM Computing Surveys* 41.3 (July 2009), pp. 1–58.

[Cha11] Marinos Charalambides. *Deliverable D3.4: Cooperation Strategies and Incentives*. Tech. rep. UniverSelf Project, Dec. 2011.

[Cia+12] Laurent Ciavaglia et al. "Unifying Management of Future Networks With Trust". In: *Bell Labs Technical Journal* 17.3 (Dec. 2012), pp. 193–212.

[Cia+13] Laurent Ciavaglia et al. *Deliverable D4.16: Assessment Results of Trust in Autonomics Release 2*. Tech. rep. UniverSelf Project, Nov. 2013.

[Com+13] Richard Combes et al. "Coordination of autonomic functionalities in communications networks". In: *11th International Symposium on Modeling & Optimization in Mobile, Ad Hoc & Wireless Networks (WiOpt)*. Tsukuba Science City, Japan, 2013, pp. 364–371. arXiv: `arXiv:1209.1236v1`.

[COM11] COMMUNE. *COgnitive network ManageMent under UNcErtainty*. 2011. URL: `http://projects.celtic-initiative.org/commune` (visited on 02/15/2016).

[Con14] Connect. *Mobilfunk-Netztest 2014/2015*. 2014. URL: `http://www.connect.de/vergleichstest/netztest-2014-bestes-handynetz-connect-telekom-o2-vodafone-eplus-2744431.html` (visited on 02/15/2016).

[CP04] Li Chen and Pearl Pu. *Survey of Preference Elicitation Methods*. Tech. rep. Ecole Politechnique Federale de Lausanne, 2004.

[Cra+03] José Craveirinha et al. "A new multiple objective dynamic routing method using implied costs". In: *Journal of Telecommunications and Information Technology* 3 (2003), pp. 50–59.

[Cru+11] Rubén Cruz et al. "Multi-Technology SON". In: *Self-Organizing Networks: Self-Planning, Self-Optimization and Self-Healing for GSM, UMTS and LTE*. Ed. by Juan Ramiro and Khalid Hamied. Chichester, UK: John Wiley & Sons, 2011. Chap. 3, pp. 47–64.

[CSW99] Eng U. Choo, Bertram Schoner, and William C. Wedley. "Interpretation of criteria weights in multicriteria decision making". In: *Computers & Industrial Engineering* 37.3 (Nov. 1999), pp. 527–541.

[DA10] Mariana Dirani and Zwi Altman. "A cooperative Reinforcement Learning approach for Inter-Cell Interference Coordination in OFDMA cellular networks". In: *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 2010)*. Avignon, France, May 2010, pp. 170–176.

[Dah+14] Erik Dahlman et al. "5G Radio Access". In: *Ericsson Review* 6 (June 2014).

[DAS11] Mariana Dirani, Zwi Altman, and Mikael Salaun. "Autonomics in Radio Access Networks". In: *Autonomic Network Management Principles: From Concepts to Applications*. Ed. by Nazim Agoulmine. Elsevier, 2011. Chap. 7, pp. 141–166.

[Dez14] Dezide. *Dezide & Bayesian Networks*. 2014. URL: `http://www.dezide.com/technology/dezide-bayesian-networks/` (visited on 12/02/2015).

[DL97] Philippe Delquié and Ming Luo. "A simple trade-off condition for additive multiattribute utility". In: *Journal of Multi-Criteria Decision Analysis* 6.5 (Sept. 1997), pp. 248–252.

[Dom+11] Carmel Domshlak et al. "Preferences in AI: An overview". In: *Artificial Intelligence* 175.7-8 (May 2011), pp. 1037–1052.

[DT99] Jon Doyle and Richmond H. Thomason. "Background to Qualitative Decision Theory". In: *AI Magazine* 20.2 (1999).

[Dye05] James S. Dyer. "Maut - Multiattribute Utility Theory". In: *Multiple Criteria Decision Analysis: State of the Art Surveys*. Ed. by José Figueira, Salvatore Greco, and Matthias Ehrogott. New York, USA: Springer-Verlag, 2005. Chap. 7, pp. 265–292.

[Ebe+09] Jörg Eberspächer et al. *GSM– Architecture, Protocols and Services*. 3rd. Chichester, UK: John Wiley & Sons, Ltd, 2009.

[Ein13] Sabrina Einberger. "Nutzenbasierte Konfliktlösung für Batch Request und Single Request in SON". Bachelor thesis. University of Augsburg, 2013. URL: `http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:384-opus4-24792`.

[Eri15] Ericsson. *Ericsson Mobility Report - November 2015.* Tech. rep. Nov. 2015.

[Fis70] Peter C. Fishburn. *Utility Theory for Decision Making.* John Wiley & Sons, 1970.

[Fis74] Peter C. Fishburn. "Exceptional Paper–Lexicographic Orders, Utilities and Decision Rules: A Survey". In: *Management Science* 20.11 (1974), pp. 1442–1471.

[FK74] Peter C. Fishburn and Ralph L. Keeney. "Seven independence concepts and continuous multiattribute utility functions". In: *Journal of Mathematical Psychology* 11.3 (Aug. 1974), pp. 294–327.

[FLS14a] Christoph Frenzel, Simon Lohmüller, and Lars Christoph Schmelz. "Dynamic, context-specific SON management driven by operator objectives". In: *2014 IEEE Network Operations and Management Symposium (NOMS).* IEEE, May 2014, pp. 1–8.

[FLS14b] Christoph Frenzel, Simon Lohmüller, and Lars Christoph Schmelz. "SON management based on weighted objectives and combined SON Function models". In: *2014 11th International Symposium on Wireless Communications Systems (ISWCS).* IEEE, Aug. 2014, pp. 149–153.

[Fre+14] Christoph Frenzel et al. "Detection and resolution of ineffective function behavior in Self-Organizing Networks". In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014.* IEEE, June 2014, pp. 1–3.

[Fre+15a] Christoph Frenzel et al. "Objective-driven coordination in self-organizing networks". In: *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC).* IEEE, Aug. 2015, pp. 1453–1458.

[Fre+15b] Christoph Frenzel et al. "Operational Troubleshooting-Enabled Coordination in Self-Organizing Networks". In: *Mobile Networks and Management - 6th International Conference, MONAMI 2014, Würzburg, Germany, September 22-24, 2014, Revised Selected Papers.* Springer, Feb. 2015, pp. 149–162.

[FSB12] Christoph Frenzel, Henning Sanneck, and Bernhard Bauer. "Automated rational recovery selection for self-healing in mobile networks". In: *2012 International Symposium on Wireless Communication Systems (ISWCS).* IEEE, Aug. 2012, pp. 41–45.

[FSB13a] Christoph Frenzel, Henning Sanneck, and Bernhard Bauer. "A Fuzzy, Utility-Based Approach for Proactive Policy-Based Management". In: *Proceedings of the 7th International Symposium on Theory, Practice, and Applications of Rules on the Web - 7th International Symposium, RuleML 2013, Seattle, WA, USA, July 11-13, 2013.* Springer, July 2013, pp. 84–98.

[FSB13b] Christoph Frenzel, Henning Sanneck, and Bernhard Bauer. "Rational Policy System for Network Management". In: *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, May 2013, pp. 776–779.

[FSH11] Christoph Frenzel, Henning Sanneck, and Seppo Hämäläinen. "Future Research Topics". In: *LTE Self-Organising Networks (SON)*. Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, Dec. 2011. Chap. 11, pp. 379–390.

[Gal+12] Aristi Galani et al. "A policy based framework for governing Future networks". In: *Proc. 2012 IEEE Globecom Workshops*. Anaheim, CA, USA, Dec. 2012, pp. 802–806.

[GBK11] Markus Gruber, Sem Borst, and Edgar Kuehn. "Stable Interaction of Self-Optimization Processes in Wireless Networks". In: *2011 IEEE International Conference on Communications Workshops (ICC)*. IEEE, June 2011, pp. 1–6.

[Geo13] Hans-Otto Georgii. *Stochastics: Introduction to Probability and Statistics*. 2nd. Berlin, Germany: Walter de Gruyter, 2013.

[GH03] Carlos Gershenson and Francis Heylighen. "When Can We Call a System Self-Organizing?" In: 2003, pp. 606–614.

[GNM15] Borislava Gajic, Szabolcs Nováczki, and Stephen S. Mwanje. "An improved anomaly detection in mobile networks by using incremental time-aware clustering". In: *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*. IEEE, 2015, pp. 1286–1291.

[Göt+15] Dario Götz et al. *Integrated SON Management - Policy Transformation and Operational SON Coordination (final results)*. Deliverable D5.3. SEMAFOUR Project, Feb. 2015.

[GSB11] Xavier Gelabert, Berna Sayrac, and Sana Ben Jemaa. "A performance evaluation framework for control loop interaction in Self Organizing Networks". In: *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, Sept. 2011, pp. 263–267.

[GSB14] Xavier Gelabert, Berna Sayrac, and Sana Ben Jemaa. "A Heuristic Coordination Framework for Self-Optimizing Mechanisms in LTE HetNets". In: *IEEE Transactions on Vehicular Technology* 63.3 (Mar. 2014), pp. 1320–1334.

[GSM15] GSM Association. *The Mobile Economy 2015*. Tech. rep. 2015.

[Hah+14] Sören Hahn et al. "SON management simulator implementation and findings". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. Krakow, Poland: IEEE, May 2014, pp. 1–15.

[Hah+15] Sören Hahn et al. "Classification of Cells Based on Mobile Network Context Information for the Management of SON Systems". In: *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE, May 2015, pp. 1–5.

[Han05] Sven Ove Hansson. *Decision Theory: a brief introduction*. 2005. URL: http://elibrary.kiu.ac.ug:8080/jspui/handle/1/1620 (visited on 03/28/2016).

[Hau14] Duncan Haughey. *SMART Goals*. 2014. URL: https://cdn.projectsmart.co.uk/pdf/smart-goals.pdf (visited on 08/12/2015).

[HBH91] Max Henrion, John S. Breese, and Eric J Horvitz. "Decision Analysis and Expert Systems". In: *AI Magazine* 12.4 (1991), pp. 64–91.

[HBR95] David Heckerman, John S. Breese, and Koos Rommelse. "Decision-theoretic troubleshooting". In: *Communications of the ACM* 38.3 (Mar. 1995), pp. 49–57.

[HK14] Sören Hahn and Thomas Kürner. "Managing and altering mobile radio networks by using SON function performance models". In: *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. Barcelona, Spain, Aug. 2014, pp. 214–218.

[HL05] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to operations research*. 8th ed. Boston, USA: McGraw-Hill Higher Education, 2005.

[HSS11] Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori, eds. *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Chichester, UK: John Wiley & Sons, Dec. 2011.

[HT10] Harri Holma and Antti Toskala, eds. *WCDMA for UMTS: HSPA Evolution and LTE*. 5th. Chichester, UK: John Wiley & Sons, Ltd, 2010.

[HT11] Harri Holma and Antti Toskala, eds. *LTE for UMTS: Evolution to LTE-Advanced*. 2nd. Chichester, UK: John Wiley & Sons, Ltd, Mar. 2011.

[Iac+14a] Ovidiu Iacoboaiea et al. "Coordinating SON instances: Reinforcement learning with distributed value function". In: *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*. Washington, DC, USA: IEEE, Sept. 2014, pp. 1642–1646.

[Iac+14b] Ovidiu Iacoboaiea et al. "SON Coordination for parameter conflict resolution: A reinforcement learning framework". In: *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. Istanbul, Turkey, Apr. 2014, pp. 196–201.

[Iac+15] Ovidiu Iacoboaiea et al. "SON Conflict Diagnosis in Heterogeneous Networks". In: *IEEE 26th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2015.

[IS89] Yannis E Ioannidis and Timos K Sellis. "Conflict resolution of rules assigning values to virtual attributes". In: *ACM SIGMOD Record* 18.2 (June 1989), pp. 205–214.

[ITU15] ITU. *Measuring the Information Society Report*. Tech. rep. International Telecommunication Union, 2015.

[IZ14] Ali Imran and Ahmed Zoha. "Challenges in 5G: how to empower SON with big data for enabling 5G". In: *IEEE Network* 28 (2014), pp. 27–33.

[Jen+07] Brendan Jennings et al. "Towards autonomic management of communications networks". In: *IEEE Communications Magazine* 45.10 (Oct. 2007), pp. 112–121.

[Jor+14] Ljupco Jorguseski et al. "Self-organizing networks in 3GPP: standardization and future trends". In: *IEEE Communications Magazine* 52.12 (Dec. 2014), pp. 28–34.

[JT10] Dylan Jones and Mehrdad Tamiz. *Practical Goal Programming*. New York, USA: Springer, 2010.

[Kar+13] Ingo Karla et al. *Deliverable D4.15: Synthetic analysis of deployment results and impacts*. Tech. rep. Oct. 2013.

[Kar14] Edi Karni. "Axiomatic Foundations of Expected Utility and Subjective Probability". In: *Handbook of the Economics of Risk and Uncertainty*. Ed. by Mark J. Machina and W. Kip Viscusi. 2014, pp. 1–39.

[KC03] Jeffrey O. Kephart and D.M. Chess. "The vision of autonomic computing". In: *Computer* 36.1 (Jan. 2003), pp. 41–50.

[KD07] Jeffrey O. Kephart and Rajarshi Das. "Achieving Self-Management via Utility Functions". In: *IEEE Internet Computing* 11.1 (2007), pp. 40–48.

[Kee74] Ralph L. Keeney. "Multiplicative Utility Functions". In: *Operations Research* 22.1 (1974).

[Kee77] Ralph L. Keeney. "The art of assessing multiattribute utility functions". In: *Organizational Behavior and Human Performance* 19.2 (Aug. 1977), pp. 267–310.

[Kee82] Ralph L. Keeney. "Decision Analysis: An Overview". In: *Operations Research* 30.5 (Oct. 1982), pp. 803–838.

[Kha+06] Rana M. Khanafer et al. *Automating Diagnosis in Troubleshooting*. Tech. rep. Celtic Gandalf, 2006.

[Kha+08] Rana M. Khanafer et al. "Automated diagnosis for UMTS networks using Bayesian network approach". In: *IEEE Transactions on Vehicular Technology* 57.4 (2008), pp. 2451–2461.

[KK13] Khoa Truong Dinh and Sławomir Kukliński. "Joint implementation of several LTE-SON functions". In: *2013 IEEE Globecom Workshops (GC Wkshps)*. Atlanta, USA, Dec. 2013, pp. 953–957.

[KR93] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives - Preferences and Value Trade-Offs*. Cambridge University Press, 1993.

[Kür+10] Thomas Kürner et al. *D5.9: Final Report on Self-Organisation and its Implications in Wireless Access Networks*. Deliverable D5.9. SOCRATES Project, Jan. 2010.

[Kür+15] Thomas Kürner et al. *SON for future Networks*. Deliverable D4.4. SEMAFOUR Project, May 2015.

[KW04] Jeffrey O. Kephart and William E. Walsh. "An artificial intelligence perspective on autonomic computing policies". In: *Proceedings. Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, 2004. POLICY 2004.* Yorktown Heights, USA: IEEE, June 2004, pp. 3–12.

[Lai+05] J. Laiho et al. "Advanced analysis methods for 3G cellular networks". In: *IEEE Transactions on Wireless Communications* 4.3 (May 2005), pp. 930–942.

[Las+11] Daniela Laselva et al. "Self-Optimisation". In: *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, 2011. Chap. 5, pp. 135–234.

[Len02] Maurizio Lenzerini. "Data integration: A Theoretical Perspective". In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*. New York, New York, USA: ACM Press, 2002, p. 233.

[LIA13] Hafiz Yasar Lateef, Ali Imran, and Adnan Abu-Dayya. "A framework for classification of Self-Organising network conflicts and coordination algorithms". In: *2013 IEEE 24th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. London, UK, 2013, pp. 2898–2903.

[Lii+12] Marja Liinasuo et al. "Human Operator Perspective to Autonomic Network Management". In: *5th International Conference on Advances in Computer-Human Interactions (ACHI 2012)*. c. Valencia, Spain, Jan. 2012, pp. 128–134.

[Lit+03] Michael L Littman et al. "Cost-Sensitive Fault Remediation for Autonomic Computing". In: *IJCAI workshop on AI and autonomic computing: developing a research agenda for self-managing computer systems, Acapulco, Mexico*. 2003.

[Liu+10] Zhiqiang Liu et al. "Conflict Avoidance between Mobility Robustness Optimization and Mobility Load Balancing". In: *Proceedings of the Global Communications Conference, 2010. GLOBECOM 2010, 6-10 December 2010, Miami, Florida, USA*. IEEE, 2010, pp. 1–5.

[Loh+15] Simon Lohmüller et al. "Policy-Based SON Management Demonstrator". In: *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE, May 2015, pp. 1–2.

[LSH16] Simon Lohmüller, Lars Christoph Schmelz, and Sören Hahn. "Adaptive SON Management Using KPI Measurements". In: *2016 IEEE Network Operations and Management Symposium (NOMS)*. Apr. 2016. accepted paper.

[Luo+13] Markus Luoto et al. *Architecture for Cognitive Networks*. Tech. rep. COMMUNE project, 2013.

[Lup13] Emil C. Lupu. *Ponder2*. 2013. URL: `http://erats.net/wp/ponder2/` (visited on 08/12/2015).

[LWN07] Jaana Laiho, Achim Wacker, and Tomá?? Novosad, eds. *Radio Network Planning and Optimisation for UMTS*. 2nd. Chichester, UK: John Wiley & Sons, Ltd, Dec. 2007.

[Mee+06] Sven van der Meer et al. "Autonomic Networking: Prototype Implementation of the Policy Continuum". In: *Proc. 1st IEEE International Workshop on Broadband Convergence Networks (BcN 2006)*. Vancouver, Canada, Apr. 2006, pp. 1–10.

[Miy95] Hajime Miyazaki. *On Lexicographic (Dictionary) Preference*. 1995. URL: `www.econ.ohio-state.edu/miyazaki/econ804/LEXICO.pdf`.

[Moo+01] B. Moore et al. *Policy Core Information Model – Version 1 Specification*. RFC 3060. IETF, 2001.

[Mot+14] Vinicius F. S. Mota et al. "Managing the decision-making process for opportunistic mobile data offloading". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, May 2014, pp. 1–8.

[Mov+12] Zeinab Movahedi et al. "A Survey of Autonomic Network Architectures and Evaluation Criteria". In: *IEEE Communications Surveys & Tutorials* 14.2 (Jan. 2012), pp. 464–490.

[NGA08] Quoc-Thinh Nguyen-Vuong, Yacine Ghamri-Doudane, and Nazim Agoulmine. "On utility models for access network selection in wireless heterogeneous networks". In: *2008 IEEE Network Operations and Management Symposium (NOMS)*. Salvador, Brazil, Apr. 2008, pp. 144–151.

[NGM07a] NGMN Alliance. *NGMN Informative List of SON Use Cases*. Tech. rep. Next Generation Mobile Networks Alliance, 2007.

[NGM07b] NGMN Alliance. *Use Cases related to Self Organising Network, Overall Description*. Tech. rep. Next Generation Mobile Networks Alliance, 2007.

[NGM08] NGMN Alliance. *NGMN Recommendation on SON and O&M Requirements*. Tech. rep. Next Generation Mobile Networks Alliance, 2008.

[NGM10] NGMN Alliance. *NGMN Top OPE Recommendations*. Whitepaper. Next Generation Mobile Networks Alliance, Sept. 2010.

[NGM14] NGMN Alliance. *Recommended Practices for Multi-vendor SON Deployment*. Tech. rep. Next Generation Mobile Networks Alliance, 2014.

[NGM15] NGMN Alliance. *5G White Paper*. White paper. Next Generation Mobile Networks Alliance, 2015.

[NM53] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. 3rd. Princeton University Press, 1953.

[Nok09] Nokia Siemens Networks. *Self-Organizing Network (SON): Introducing the Nokia Siemens Networks SON Suite - an efficient, future-proof platform for SON*. 2009.

[Nok14] Nokia. *Looking ahead to 5G*. Tech. rep. Nokia Solutions and Networks, 2014.

[Nov+11] Szabolcs Nováczki et al. "Self-Healing". In: *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, 2011. Chap. 6, pp. 235–266.

[Nov13] Szabolcs Nováczki. "An improved anomaly detection and diagnosis framework for mobile network operators". In: *9th International Conference on the Design of Reliable Communication Networks, DRCN 2013, Budapest, Hungary, March 4-7, 2013*. IEEE, 2013, pp. 234–241.

[OMG15] OMG. *Decision Model and Notation*. Tech. rep. Object Management Group, 2015.

[Ope13] OpenRules. *Solving Rule Conflicts*. 2013. URL: `https://openrules.wordpress.com/2013/08/14/solving-rule-conflicts-part-1/` (visited on 08/12/2015).

[PFL14] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco3 Documentation*. 2014. URL: `http://www.choco-solver.org` (visited on 02/15/2016).

[Pha+08] Tan Phan et al. "A survey of policy-based management approaches for Service Oriented Systems Swinburne University of Technology". In: (2008), pp. 392–401.

[RAH11] Juan Ramiro, Mark Austin, and Khalid Hamied. "Return on Investment (ROI) for Multi-Technology SON". In: *Self-Organizing Networks: Self-Planning, Self-Optimization and Self-Healing for GSM, UMTS and LTE*. Ed. by Juan Ramiro and Khalid Hamied. Chichester, UK: John Wiley & Sons, 2011. Chap. 7, pp. 231–261.

[RB11] Raphael Romeikat and Bernhard Bauer. "Automated Refinement of Policies for Network Management". In: *Proc. 17th Asia-Pacific Conference on Communications (APCC 2011)*. Kota Kinabalu, Malaysia, Oct. 2011, pp. 439–444.

[RG95] Anand S. Rao and Michael P. Georgeff. "BDI Agents: From Theory to Practice". In: *First International Conference on Multiagent Systems (ICMAS)*. San Francisco, CA, USA, June 1995.

[RH12] Juan Ramiro and Khalid Hamied, eds. *Self-organizing networks: self-planning, self-optimization and self-healing for GSM, UMTS and LTE*. Chichester, UK: John Wiley & Sons, 2012.

[RJW10] Thomas Reidemeister, Miao Jiang, and Paul a.S. Ward. "An extensible framework for repair-driven monitoring". In: *2010 International Conference on Network and Service Management*. IEEE, Oct. 2010, pp. 142–149.

[RN10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[Rom+10] Raphael Romeikat et al. "Policy-driven workflows for mobile network management automation". In: *6th International Wireless Communications and Mobile Computing Conference (IWCMC)*. Caen, France, 2010, pp. 1111–1115.

[Rom12] Raphael Romeikat. "Domain-Specific Development of Event Condition Action Policies". PhD Thesis. 2012.

[RSB13] Raphael Romeikat, Henning Sanneck, and Tobias Bandh. "Efficient, dynamic coordination of request batches in C-SON systems". In: *2013 IEEE 77th Vehicular Technology Conference (VTC Spring)*. Dresden, Germany, June 2013, pp. 1–6.

[RT11] Vilho Räisänen and Haitao Tang. "Knowledge modeling for conflict detection in self-organized networks". In: *2011 3rd Internactional Conferernce on Mobile Networks and Management (MONAMI)*. Aveiro, Portugal, 2011, pp. 107–119.

[Saa08] Thomas L. Saaty. "Decision making with the analytic hierarchy process". In: *International Journal of Services Sciences* 1.1 (2008), p. 83.

[Sam15] Samsung Electronics. *5G Vision*. White Paper. 2015.

[San+11] Henning Sanneck et al. "Self-Configuration ('Plug-and-Play')". In: *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, 2011. Chap. 4, pp. 81–134.

[Sar+11] Cinzia Sartori et al. "SON for Heterogeneous Networks (HetNet)". In: *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, 2011. Chap. 10, pp. 357–378.

[SB12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd. Cambridge, MA, USA: MIT Press, 2012.

[Sch+08] Lars Christoph Schmelz et al. *D2.4: Framework for the development of self-organising methods*. Deliverable. SOCRATES Project, 2008.

[Sch+11] Lars Christoph Schmelz et al. "A Coordination Framework for Self-Organisation in LTE Networks". In: *Proc. 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. Dublin, Ireland, May 2011, pp. 193–200.

[Sch+13] Lars Christoph Schmelz et al. *State-of-the-art in Radio Network Management Feedback from the Advisory Board Members*. Deliverable D2.3. SEMAFOUR Project, Mar. 2013.

[Sch+14a] Lars Christoph Schmelz et al. *Integrated SON Management - Policy Transformation and Operational SON Coordination (first results)*. Deliverable D5.2. SEMAFOUR Project, June 2014.

[Sch+14b] Lars Christoph Schmelz et al. "SON management demonstrator". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, May 2014, pp. 1–2.

[Sch12] Lars Christoph Schmelz. *Self-Management for Unified Heterogeneous Radio Access Networks*. Presentation at Future Networks 10th FP7 Concertation Meeting. Brussels, Belgium, 2012.

[SEM12] SEMAFOUR. *SEMAFOUR website*. 2012. URL: `www.fp7-semafour.eu` (visited on 03/28/2016).

[SFL14] Lars Christoph Schmelz, Christoph Frenzel, and Simon Lohmüller. *Network Entity and Method for Controlling a SON-Function*. WIPO Pub. No. WO 2014/191469 A1. Dec. 2014.

[SGM05] Yannis Siskos, Evangelos Grigoroudis, and Nikolaos F. Matsatsinis. "UTA Methods". In: *Multiple Criteria Decision Analysis: State of the Art Surveys*. Vol. 78. International Series in Operations Research & Management Science. New York: Springer-Verlag, 2005.

[SJK00] Claus Skaanning, Finn V. Jensen, and Uffe Kjærulff. "Printer Troubleshooting Using Bayesian Networks". In: 2000, pp. 367–380.

[SK05] Nancy Samaan and Ahmed Karmouch. "An automated policy-based management framework for differentiated communication systems". In: *IEEE Journal on Selected Areas in Communications* 23.12 (2005), pp. 2236–2247.

[SK09] Nancy Samaan and Ahmed Karmouch. "Towards Autonomic Network Management: an Analysis of Current and Future Research Directions". In: *IEEE Communications Surveys & Tutorials* 11.3 (2009), pp. 22–36.

[SN12] Péter Szilágyi and Szabolcs Nováczki. "An Automatic Detection and Diagnosis Framework for Mobile Communication Systems". In: *IEEE Transactions on Network and Service Management* 9.2 (June 2012), pp. 184–197.

[SOC11] SOCRATES. *The Socrates Project Website*. 2011. URL: `http://www.fp7-socrates.eu/` (visited on 02/01/2016).

[SSF13] Henning Sanneck, Péter Szilágyi, and Christoph Frenzel. *Sub-cell Level, Multi-layer Degradation Detection, Diagnosis and Recovery*. WIPO Pub. No. WO 2013/143572 A1. Oct. 2013.

[SSF14] Henning Sanneck, Péter Szilágyi, and Christoph Frenzel. *Self Organizing Network Operation Diagnosis Function*. WIPO Pub. No. WO 2014/023347 A1. Feb. 2014.

[SSH06] S. Schmid, M. Sifalakis, and David Hutchison. "Towards Autonomic Networks". In: *Autonomic Networking: First International IFIP TC6 Conference, AN 2006*. Springer, 2006, pp. 1–11.

[STB11] Stefania Sesia, Issam Toufik, and Matthew Baker, eds. *LTE– The UMTS Long Term Evolution*. 2nd. Chichester, UK: John Wiley & Sons, Ltd, Feb. 2011.

[Ste05] Theodor J. Stewart. "Dealing with Uncertainties in MCDA". In: *Multiple Criteria Decision Analysis: State of the Art Surveys*. Vol. 78. International Series in Operations Research & Management Science. New York: Springer-Verlag, 2005. Chap. 11, pp. 448–466.

[Ste95] Theodor J. Stewart. "Simplified approaches for multicriteria decision making under uncertainty". In: *Journal of Multi-Criteria Decision Analysis* 4.4 (Dec. 1995), pp. 246–258.

[Ste96] Theodor J. Stewart. "Robustness of Additive Value Function Methods in MCDM". In: *Journal of Multi-Criteria Decision Analysis* 5.4 (Dec. 1996), pp. 301–309.

[Str03] John Strassner. *Policy-Based Network Management: Solutions for the Next Generation*. Amsterdam, Netherlands: Morgan Kaufmann, Aug. 2003.

[TC09] Kenneth J. Turner and Gavin A. Campbell. "Goals and Conflicts in Telephony". In: *Proc. International Conference on Feature Interactions in Software and Communication Systems (ICFI 2009)*. June. Lisbon, Portugal, June 2009, pp. 3–18.

[Tel04] TeleManagement Forum. *Wireless Service Measurements Handbook*. 2004.

[Tel12] TeleManagement Forum. *SLA Management Handbook*. Tech. rep. 2012.

[The13] The JBoss Drools team. *Drools Documentation*. 2013. URL: https://docs.jboss.org/drools/release/6.2.0.Final/drools-docs/html/ (visited on 08/12/2015).

[Tiw+10] Moazzam Islam Tiwana et al. "Statistical Learning-based Automated Healing: Application to mobility in 3G LTE networks". In: *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, Sept. 2010, pp. 1746–1751.

[Tom+11] Malgorzata Tomala et al. "Supporting Function: Minimisation of Drive Tests (MDT)". In: *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, Dec. 2011. Chap. 7, pp. 267–310.

[Tsa+13] Kostas Tsagkaris et al. *Deliverable D2.4: Unified Management Framework (UMF) Specifications Release 3*. Tech. rep. UniverSelf Project, Nov. 2013.

[TSA10] Moazzam Islam Tiwana, Berna Sayrac, and Zwi Altman. "Statistical Learning in Automated Troubleshooting: Application to LTE Interference Mitigation". In: *IEEE Transactions on Vehicular Technology* 59.7 (Sept. 2010), pp. 3651–3656.

[TSC14] Tsvetko Tsvetkov, Henning Sanneck, and Georg Carle. "An experimental system for SON verification". In: *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. Barcelona, Spain, Aug. 2014, pp. 343–344.

[Tsv+14] Tsvetko Tsvetkov et al. "A Post-Action Verification Approach for Automatic Configuration Parameter Changes in Self-Organizing Networks". In: *6th International Conference on Mobile Networks and Management (MONAMI 2014)*. Würzburg, Germany, 2014, pp. 135–148.

[Tsv+15] Tsvetko Tsvetkov et al. "A Constraint Optimization-Based Resolution of Verification Collisions in Self-Organizing Networks". In: *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2015.

[TYN15] Kei Takeshita, Masahiro Yokota, and Ken Nishimatsu. "Early network failure detection system by analyzing Twitter data". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, May 2015, pp. 279–286.

[Uni10] UniverSelf. *Univerself website*. 2010. URL: http://www.univerself-project.eu (visited on 02/15/2016).

[Uni12a] UniverSelf. *Case Study - Part I: Operator-governed, end-to-end, autonomic, joint network and service management*. Tech. rep. UniverSelf Project, 2012, pp. 1–15.

[Uni12b] UniverSelf. *Case Study - Part I: SON and SON collaboration according to operator policies*. Tech. rep. UniverSelf Project, 2012.

[Uni12c] UniverSelf. *Proof of Concept: Governance and coordination of autonomic LTE wireless access and MPLS core network segments.* Tech. rep. UniverSelf Project, 2012.

[Uni12d] UniverSelf. *Proof of Concept: Governance Framework for Network Optimisation.* Tech. rep. UniverSelf Project, Oct. 2012.

[Ver02] D.C. Verma. "Simplifying network administration using policy-based management". In: *IEEE Network* 16.2 (2002), pp. 20–26.

[Wal+11] Richard Waldhauser et al. "Self-Organising Networks (SON)". In: *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency.* Ed. by Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. Chichester, UK: John Wiley & Sons, Ltd, 2011. Chap. 3, pp. 39–80.

[WD92] Michael P. Wellman and Jon Doyle. "Modular Utility Representation for Decision-Theoretic Planning". In: *First International Conference on AI Planning Systems (AIPS-92).* 1992, pp. 236–242.

[Wei00] Stefan Weinzierl. *Introduction to Monte Carlo methods.* June 2000. arXiv: `0006269 [hep-ph]`.

[Wei04] Paul Weirich. *Realistic Decision Theory: Rules for Nonideal Agents in Nonideal Circumstances.* Oxford University Press, 2004.

[Wil+04] Volker Wille et al. "Automation and Optimisation". In: *GSM, GPRS and EDGE Performance.* Chichester, UK: John Wiley & Sons, 2004, pp. 467–511.

[Win04] Wayne L. Winston. *Operations Research: Applications and Algorithms.* 4th ed. Cengage Learning, 2004.

[YTP93] Kum Khiong Yang, F.Brian Talbot, and James H. Patterson. "Scheduling a project to maximize its net present value: An integer programming approach". In: *European Journal of Operational Research* 64.2 (Jan. 1993), pp. 188–198.

[Zam+15] Faisal Zaman et al. "A recommender system architecture for predictive telecom network management". In: *IEEE Communications Magazine* 53 (2015), pp. 286–293.

List of Acronyms

3GPP	3rd Generation Partnership Project
2G	Second Generation
3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
AC	Autonomic Computing
AHP	Analytic Hierarchy Process
AI	Artificial Intelligence
ANR	Automatic Neighbor Relationship setup
ARCEP	Autorité de Régulation des Communications Électroniques et des Postes
BDI	Believes, Desires, Intentions
BS	Base Station
CAPEX	Capital Expenditure
CBR	Constant Bit Rate
CCO	Coverage and Capacity Optimization
CCO-RET	Coverage and Capacity Optimization - Remote Electrical Tilt
CCO-SCA	Coverage and Capacity Optimization - Small Cell Activation
CIO	Cell Individual Offset
CM	Configuration Management
COC	Cell Outage Compensation
COD	Cell Outage Detection
COMMUNE	COgnitive network ManageMent under UNcErtainty
CQI	Channel Quality Indicator
DCR	Dropped Call Rate
E-UTRAN	Evolved Universal Terrestrial Radio Access Network

ECA	Event-Condition-Action
ESM	Energy Saving Management
EU	European Union
FM	Failure Management
GARSON	Generic ARchitecture of Self-Organized Networks
Gbps	gigabits per second
GERAN	GSM EDGE Radio Access Network
GP	Goal Programming
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HetNet	Heterogeneous Network
ICIC	Inter-Cell Interference Coordination
ICT	Information and Communications Technology
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Integer Programming
IT	Information Technology
ITU	International Telecommunication Union
KPI	Key Performance Indicator
KQI	Key Quality Indicator
LTE	Long Term Evolution
M2M	Machine-to-Machine
MAPE	Monitor-Analyze-Plan-Execute
MAUT	Multiattribute Utility Theory
Mbps	megabits per second
MDT	Minimization of Drive Tests
MLB	Mobility Load Balancing

MNO	Mobile Network Operator
MRO	Mobility Robustness Optimization
NEM	Network Empowerment Mechanism
NGMN	Next Generation Mobile Networks Alliance
OAM	Operations, Administration, and Maintenance
ODSO	Objective-Driven SON Operations
OMG	Object Management Group
OPEX	Operational Expenditure
PBM	Policy-Based Management
PBNM	Policy-Based Network Management
PBSM	Policy-based SON Management
PCI	Physical Cell ID
PM	Performance Management
PRB	Physical Resource Block
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
RET	Remote Electrical Tilt
RL	Reinforcement Learning
RLF	Radio Link Failures
SFC	SON Function Configuration
SEASON	System Experience of Advanced SON
SEMAFOUR	SElf-MAnagement FOr Unified heterogeneous Radio access networks
SLA	Service Level Agreement
SOCRATES	Self-Optimisation and self-ConfiguRATion in wirelEss networkS
SON	Self-Organizing Network

TXP	Transmission Power
UE	User Equipment
UMF	Unified Management Framework
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Terrestrial Radio Access Network

List of Symbols

$\succsim$	A preference relation (see Chapter 2.3.1).
$\geq_D$	An ordering relation over expected utility vectors considering the priorities (see Definition 3.19).
a	An action in the generic component design (see Definition 3.5); specifically, a cell-specific SON configuration in the context of SON management (see Definition 4.3), a SON function request in the context of SON coordination (see Definition 5.1), or a recovery action (see Definition 6.2) in the context of SON self-healing.
$\tilde{a}$	A procedure in the generic component design (see Definition 3.5).
$\tilde{A}$	The set of all procedures in the generic component design (see Definition 3.5).
A	The set of all actions in the generic component design (see Definition 3.5); specifically, all cell-specific SON configurations in the context of SON management (see Definition 4.3), all SON function requests in the context of SON coordination (see Definition 5.1), or all recovery actions for a problem in the context of SON self-healing (see Definition 6.2).
$\mathcal{A}$	The set of all SON function requests for a granularity period (see Definition 5.1).
AF	The action-effect set in the generic component design (see Definition 3.10); specifically, the feasible SON configurations and their effects in the context of SON management (see Definition 4.4), the SON function requests and their effects in the context of SON coordination (see Definition 5.6), or the recovery actions and their effects in the context of SON self-healing (see Definition 6.6).
agree($\mathbb{F}$)	The agreement for the set of partial effects $\mathbb{F}$ (see Definition 4.6).
$\mathcal{A}^{\mathrm{TCR}}$	The set of accepted SON function requests by technical conflict resolution (see Definition 5.5).
$\overline{\mathcal{A}}^{\mathrm{TCR}}$	The set of rejected SON function requests by technical conflict resolution (see Definition 5.5).
$\mathcal{A}^{\mathrm{OM}}$	The set of accepted SON function requests by SON function request selection (see Definition 5.12).

$\overline{\mathcal{A}}^{\mathrm{OM}}$	The set of rejected SON function requests by SON function request selection (see Definition 5.12).
$\mathbb{B}$	The set of Boolean numbers, i.e., $\mathbb{B}= \{0, 1\}$.
$\perp$	The empty KPI effect (see Definition 3.8).
c	A network cell (see Definition 3.1).
C	The set of all network cells (see Definition 3.1).
CDM	The conflict detection model (see Definition 5.3).
CM	The cost model (see Definition 6.10).
δ_i	The Dirac distribution for i (see Chapter 3.4.2).
d	A priority (see Definition 3.13).
D	The set of all priorities (see Definition 3.13).
$\mathrm{Dom}(k)$	The domain of a KPI k (see Definition 3.2).
$\mathrm{E}[j(i)]$	The expected value of the function j regarding the probability distribution i (see Chapter 3.4.4).
$\mathbf{f}$	A complete effect (see Definition 3.7).
$\mathbf{F}$	The set of all complete effects (see Definition 3.7).
F_k	The function space of all possible KPI effects for KPI k (see Definition 3.6).
f_k	A KPI effect for KPI k (see Definition 3.6).
$\mathbf{f}^{\perp}$	A partial effect (see Definition 3.9).
$\mathbf{F}^{\perp}$	The set of all partial effects (see Definition 3.9).
$F_k^{\perp}$	The function space of all possible partial KPI effects for KPI k (see Definition 3.8).
$f_k^{\perp}$	A partial KPI effect for KPI k (see Definition 3.8).
μ	The effect merging function (see Definition 3.11).
μ_k	The KPI effect merging function for KPI k (see Definition 3.11).
G_s	The set of all SFCs for the SON function s (see Definition 4.1).
g_s	An SFC for the SON function s (see Definition 4.1).

κ	The set of detected SON function request conflicts (see Definition 5.4).
k	A KPI (see Definition 3.2).
K	The set of all cell KPIs (see Definition 3.2).
$\text{lex}_D \max$	The lexicographic maximization operation (see Definition 5.12).
N_D	The number of priorities (see Definition 3.13).
$o_{k,d}$	A utility function for KPI k and priority d (see Definition 3.14).
$\mathbf{O}_k$	The set of all KPI objectives for KPI k (see Definition 3.14).
$\mathbf{o}_k$	A KPI objective for KPI k (see Definition 3.14).
$o_{*,*}$	A vector of vectors of utility functions with the same priority for all KPIs and all priorities (see Definition 3.18).
$o_{*,d}$	A mapping from a KPI k to the utility function with the priority d (see Definition 3.17).
o	A utility function (see Definition 3.12).
OM	The objective model (see Definition 3.16).
$\mathcal{P}(X)$	The power set of a set X.
ρ	A probability function, $\rho \in [0, 1]$ (see, e.g., Definition 3.11).
ρ_{agree}	The threshold probability for SFC conflict detection (see Definition 4.7).
$\Pr(X)$	The probability distribution over the random variable X (see, e.g., Definition 3.6).
$\text{proj}_i(j)$	The projection on the i-th component of the tuple j (see, e.g., Chapter 3.4.1.1).
$\mathbf{q}$	An action cost vector (see Definition 6.9).
q_d	An action cost regarding priority d (see Definition 6.9).
$\langle \mathbf{q} \rangle_{\mathbf{q}}$	The reduced representation of the action cost vector $\mathbf{q}$ (see Chapter 7.5.2.2).
$\mathbb{N}$	The set of natural numbers.
$\mathbb{R}$	The set of real numbers.
R	A set of SON function requests (see, e.g., Definition 5.11).

RAL	A recovery action list (see Definition 6.11).
RM	The recovery model (see Definition 6.3).
ρ_T	A probability distribution over the possible root causes (see Definition 6.1).
s	A SON function (see Definition 3.4).
S	The set of all SON functions (see Definition 3.4).
SFE	The SON function effects (see Definition 4.11).
SFM_s	The SON function model for SON function s (see Definition 4.2).
t	A possible degradation root cause (see Definition 6.1).
T	The set of all possible degradation root causes (see Definition 6.1).
TM	The technical model in the generic component design (see Chapter 3.4.1.3).
$\mathbf{u}$	An expected utility vector (see Definition 3.18).
$\measuredangle\mathbf{u}$	The biased utility of a recovery action (see Definition 6.8).
$\Delta\mathbf{u}$	An expected utility improvement of a SON function request (see Definition 5.8).
$\widehat{\Delta\mathbf{u}}$	A lifted expected utility improvement of a SON function request (see Definition 5.10).
u_d	A utility for priority d (see Definition 3.17).
$\langle\mathbf{u}\rangle_{\mathbf{u}}$	The reduced representation of the expected utility vector $\mathbf{u}$ (see Definition 7.1).
v	A value of a KPI (see Definition 3.2).
$V_{\text{agree}}(\mathcal{F}_k)$	The set of agreed KPI values for the KPI k for the set of KPI effects $\mathcal{F}_k$ (see Definition 4.5).
var_x	A GP variable for the term x (see Definition 5.12).
V_k	A random variable for the future value of the KPIs k (see Definition 3.6).
W_k	The set of all KPI weights for the KPI k (see Definition 3.15).
w_k	A KPI weight for KPI k (see Definition 3.15).
$\mathbf{x}$	An operational context (see Definition 3.3).

$\mathbf{x}_c$	The operational context of the network cell c in the operational context $\mathbf{x}$ (see Definition 3.3).
$\mathbf{X}$	The set of all possible operational contexts (see Definition 3.3).
$\mathbf{x}(c, k)$	The value of KPI k in the network cell c in the operational context $\mathbf{x}$ (see Definition 3.3).
$\mathbb{Z}$	The set of integer numbers.

List of Definitions

3.1 Definition: Network cell . . . 55
3.2 Definition: KPI . . . 55
3.3 Definition: Operational context . . . 57
3.4 Definition: SON function . . . 59
3.5 Definition: Action . . . 65
3.6 Definition: KPI effect . . . 66
3.7 Definition: Effect . . . 66
3.8 Definition: Partial KPI effect . . . 68
3.9 Definition: Partial effect . . . 68
3.10 Definition: Action-effect set . . . 69
3.11 Definition: Effect merging function . . . 70
3.12 Definition: Utility function . . . 72
3.13 Definition: Priority . . . 74
3.14 Definition: KPI objective . . . 77
3.15 Definition: KPI weight . . . 78
3.16 Definition: Objective model . . . 79
3.17 Definition: Expected utility for one priority . . . 82
3.18 Definition: Expected utility vector . . . 82
3.19 Definition: Ordering of expected utility vectors . . . 84
3.20 Definition: Action preference . . . 85

4.1 Definition: SON function configuration . . . 105
4.2 Definition: SON function model . . . 105
4.3 Definition: SON configuration . . . 107
4.4 Definition: Feasible SON configurations and effects . . . 108
4.5 Definition: Agreed values of partial KPI effects . . . 109
4.6 Definition: Agreement of SFCs . . . 110
4.7 Definition: Applicable and conflict-free actions . . . 111
4.8 Definition: Combined KPI effect . . . 114
4.9 Definition: Combined effect . . . 115
4.10 Definition: SON configuration selection . . . 115
4.11 Definition: SON function effects . . . 116

5.1 Definition: SON function requests . . . 128
5.2 Definition: Complete effect of SON function request . . . 129
5.3 Definition: Conflict detection model . . . 131
5.4 Definition: Conflict set . . . 132
5.5 Definition: Technical constraint resolution . . . 134
5.6 Definition: SON function requests and effects . . . 134
5.7 Definition: Utility vector addition and subtraction . . . 136
5.8 Definition: Expected utility improvement of a SON function request . 136
5.9 Definition: Preference of SON function requests . . . 137

5.10 Definition: Lifted expected utility improvement of a SON function request . . . 142
5.11 Definition: Preference over sets of accepted SON function requests . . 142
5.12 Definition: Goal Programming problem . . . 144

6.1 Definition: Cell diagnosis . . . 163
6.2 Definition: Recovery actions . . . 165
6.3 Definition: Recovery Model . . . 168
6.4 Definition: Complete KPI effect for an action given a root cause . . . 170
6.5 Definition: Combined complete KPI effect for an action . . . 171
6.6 Definition: Recovery actions and effects . . . 171
6.7 Definition: Preference for recovery actions . . . 172
6.8 Definition: Biased utility vector . . . 173
6.9 Definition: Action cost vector . . . 173
6.10 Definition: Cost Model . . . 175
6.11 Definition: Recovery action list . . . 176

7.1 Definition: Reduced form of expected utility vector . . . 191

List of Figures

1.1 Overview of SON operations. 3
1.2 Overview of objective-driven SON Operations 9
1.3 Outline of this thesis (the Ox or Cy show that the chapter covers Objective x or Solution and Contribution y respectively). 12

2.1 Exemplary Long Term Evolution (LTE) mobile network with mostly 3-sector Base Stations (BSs) taken from the simulation system presented in Chapter 7.2. 21
2.2 Overview of a HetNet [Sch12]. 21
2.3 Conceptual view on a SON function as a coordinated feedback loop. . 24
2.4 Potential control parameter conflicts [Ban+11a]. 25
2.5 The general Monitor-Analyze-Plan-Execute (MAPE) loop in Autonomic Computing (AC) (adapted from [KC03]). 27
2.6 Example of an influence diagram for the decision where to build a new airport (adapted from [RN10]) 36

3.1 Manual network operations: the human operator monitors the network data and reacts to detected problems by changing the network configuration. 40
3.2 SON-enabled network operations: the human operator monitors the network and SON data and reacts to detected problems by changing the SON configuration. 41
3.3 Manual SON operations means combining the operational context, operational objectives, and technical expertise in order to determine a good SON configuration. 43
3.4 Derivation of operational objectives from operator goals, and their application in different operational contexts as part of SON operations. 44
3.5 The two levels of indirection in SON operations, physical indirection and SON indirection, which must be overcome with technical expertise. 48
3.6 The frequency of changes in the context, the operational objectives, and the technical expertise, require very frequent adaptations of the SON configuration. 51
3.7 ODSO-enabled network operations: the human operator provides an objective model and technical models and the Objective-Driven SON Operations system performs autonomic SON operations accordingly. . 53
3.8 ODSO enables autonomic operations by allowing the operator to control the SON with utility-function policies (adapted from [KC03]). . . 54
3.9 Overview of the architecture of ODSO. 61
3.10 The run-time execution of ODSO (the arrow and component colors match Figure 3.9). 63
3.11 The two-step decision making process in ODSO. The steps of the generic ODSO component design are drawn in blue. 64

3.12 Exemplary uniform, triangular, normal, and arbitrary distributed effect of an action on the KPI cell load. 67
3.13 Contour plot of the three-dimensional, exemplary effect of an action on the KPIs cell load and handover ping-pong rate. 67
3.14 Visualization of the merging of an effect with the current context. . . 71
3.15 The four most common types of utility functions. 72
3.16 Example of three priority ranges for two KPIs. 75
3.17 Visualization of two KPI objectives, each with the three priority ranges. 76
3.18 Examples for KPI weights during busy hours and night-time. 79
3.19 Visualization of the example KPI objective for cell load. 80
3.20 Visualization of the calculation of the utility. The expected utility for each priority is the area below the utility graph with the same color. . 84
3.21 Example utilities for three priority ranges and two KPIs. 85
3.22 The ODSO concept decouples the different changes such that the objective model and the technical models can evolve independently of each other. The main efforts for SON operation are performed automatically by the ODSO system. 86
3.23 An action policy system enables automated SON operations in response to context changes. The action policy still needs to be manually derived from the objectives and the expertise, though. 89
3.24 Comparison between refinement of action policies (yellow) and planning for utility function policies (blue). 90

4.1 The decision making process of the SON management component. The steps of the generic ODSO component design are drawn in blue. 103
4.2 The two KPI effects and the agreed KPI values. 111
4.3 Visualization of the agreement of two KPI effects. 112
4.4 Visualization of the combined KPI effect for two KPI effects. 114

5.1 Expected utilities of the execution of SON function 1 and SON function 2. 123
5.2 Example for the decision problem of the objective-driven conflict resolution between two SON functions in two different system states. . . 124
5.3 Computation process of the SON coordination component. The steps of the generic ODSO component design are drawn in blue. 127
5.4 Computation process of the conflict detection and technical resolution. 128
5.5 Example outlining the importance to consider the utility improvement for the selection of the SON function requests. 135
5.6 Example of an oscillation between Mobility Robustness Optimization (MRO) and Mobility Load Balancing (MLB). 138
5.7 Exemplary graph of the conflicts between the SON function requests a_1, a_2, and a_3. 143

6.1 State-of-the-art SON self-healing process and identified problems in red. 154

6.2 Design of the extended SON self-healing for ODSO. New or adapted components are blue and bold. . . . 159
6.3 Design of degradation recovery. The steps of the generic ODSO component design are drawn in blue. . . . 164
6.4 Influence diagram formalizing the decision problem of degradation recovery. . . . 166
6.5 Influence diagram formalizing the multiple failure recovery action proposal. . . . 167
6.6 Sequence diagram of the interactions between SON self-healing enforcement, the recovery actions, and SON coordination 178

7.1 Overview of the components of the simulation system. . . . 185
7.2 Overview of the execution sequence of the simulation system. . . . 186
7.3 Screenshot from the network simulator showing the considered LTE network. The colored patches indicate the coverage areas and the arrows show the main sending direction of the network cells' antennas. 188
7.4 KPI effects of the SFCs CCO−RET(true, 0.5, 1.0), MRO(true, 0.1, 0.5) and MLB(true, 0.6, 0.5), as well as the respective KPI objectives. . . . 193
7.5 Simulation result of the common default scenario. . . . 195
7.6 Visualization of KPI effects of the Coverage and Capacity Optimization - Small Cell Activation (CCO-SCA) SFCs CCO−SCA(true, 18.0) and CCO−SCA(true, 5.0), and ESM SFCs ESM(true, 1.0) and ESM(true, 5.0), as well as the KPI objectives on cell load and energy consumption. . . . 197
7.7 KPI weights during the cell load-focused and energy consumption-focused periods. . . . 198
7.8 The considered Cell 6 and the area with the busy hour users (green) and the additional hot spot (red). . . . 199
7.9 The course of the simulation: the lines visualize the number of User Equipments (UEs) from the specific user group and the colored areas indicate the objective KPI the operator is focusing on. . . . 200
7.10 The simulation results for the SON management scenario comparing the performance of Cell 6 for the configurations ODSO, $a_{\overline{\text{CCO−SCA}},\text{ESM}}$, and $a_{\text{CCO−SCA},\overline{\text{ESM}}}$. . . . 204
7.11 The simulation results for the SON management scenario comparing the performance of Cell 6 for the configurations ODSO, and the conflictful $a_{\text{CCO−SCA,ESM}}$. . . . 206
7.12 The problematic networks cells: Cell 4 has a misconfigured Cell Individual Offset (CIO), Cell 5 has a wrong Remote Electrical Tilt (RET), and in Cell 6 are unusually many UEs. . . . 207
7.13 Simulation result of policy-based coordination. . . . 210
7.14 Simulation result of objective-driven coordination. . . . 211
7.15 Visualization of the cells of the SON involvement scenario and the impact of Cell 10 in sleeping mode. . . . 213
7.16 KPI values from the simulation of the detection of the erroneous Cell 10 by the SON-focused degradation detection. . . . 214

7.17 Probabilistic KPI value distributions for the creation of diagnosis cases. 217
7.18 Correct action ratio, mean recovery probability, mean action cost, and mean utility of the simulation for the four degradation recovery approaches. 221
7.19 Cost per correct action ratio, cost per recovery probability ratio, and cost per utility ratio of the simulation for the four degradation recovery approaches. 222
7.20 Results of the adapted simulation with common action effectiveness of 1.0 for the four degradation recovery approaches. 224

8.1 Characterization of manual network operations, SON, ODSO, and cognitive functions with respect to automation of decision making and knowledge creation. 233

List of Tables

6.1 Example for the decision making by SON self-healing considering the action costs. 175

7.1 Simulation parameters 187
7.2 The unweighted utilities per KPI and the KPI agreement for cell load and energy consumption of the possible SON configurations. 201
7.3 The utilities of the possible SON configurations during cell load-focused periods, i.e., RL, BL, HL, and energy consumption-focused periods, i.e., RE, BE, HE. 203
7.4 SON function request conflicts 208
7.5 Context parameters for the creation of diagnosis cases. 216
7.6 Root causes for the creation of diagnosis cases. 216

List of Listings

7.1 Recovery model. 218
7.2 Cost model. 219