Dipl.-Ing. Martin Dohr

Automotive Network Optimization

PhD Thesis

submitted in partial fulfillment of

the requirements for the degree of

Doctor of Technical Sciences

at the

Graz University of Technology

Supervisor:

Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Pribyl

Institute of Electronics

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Bernd Deutschmann

Graz, August 2014

Eidesstaatliche Erklärung

Affidavit

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral dissertation.

Graz, _____ _____

Datum/Date Unterschrift/Signature

Abstract

Ongoing developments in the automotive industry have led to highly complex electronic systems and resulting communication requirements. Approaches like AUTOSAR were introduced to meet this challenge and assure reliable electronic solutions in a cost-sensitive environment. This is done through modularization of software components accompanied by top-down design flows. From the perspective of automotive system architects, new degrees of freedom in the design of electronic features and communication are now available.

As a first step, this thesis investigates the potentials of optimization derived from these new design paradigms. A new problem formulation is then developed, modeling optimization potentials as compositions of combinatorial problems. Starting from this foundation, stochastic search methods are refined to assist in architectural decisions for complex networked systems. Furthermore, the thesis explores goals of optimization in aspects such as system complexity, communication effort and monetary costs.

Intensive research in application-specific improvements was conducted in order to ensure a reliable and effective optimization behavior. The resulting techniques feature novel guided mutation operators to enhance the convergence of the studied heuristics. Improvements are further achieved regarding the many-objective optimization performance of evolutionary algorithms. Scientific studies in experimental as well as in industrial environments confirm the proposed advancements.

The thesis is completed by the implementation of a modular optimization framework for industrial case studies and experimentation purposes. Finally, an in-depth discussion is provided regarding modeling techniques applied in this work and in comparable automotive literature.

Acknowledgements

The presented work would not have been possible without the support of countless individuals. First and foremost, I want to thank Wolfgang Pribyl for his guidance and supervision during the last three years.

Furthermore, I want to mention my project leader Bernd Eichberger for his organizational and technical support as well as countless technical discussions.

At Magna Steyr Engineering, Gabriel Stabentheiner for initiating this project. Along with him, Johannes Till and Wolfgang Reinprecht on behalf of all other engineers and specialists for providing me with most valuable insights into the automotive industry.

Manuel Menghin for his excellent feedback on my research papers, pointing out argumentative flaws and inconsistencies which are bound to happen due to business-blindness. Johannes Max Löffler and Matthias Freiberger for endless discussions about clean and readable code, advantages and pitfalls of design patterns and valuable lessons from their long software development experience.

I also want to thank Bernd Deutschmann and all staff members at the Institute of Electronics for the friendly and productive work environment. Particularly Claudia Cargnel-Winter and Julia Schönfelder for taking so much administrative work out of my hands and all fellow diploma and PhD students in the lab.

Last but not least, I owe very special thanks to my family, my girlfriend and all friends for their continuous support, encouragement and patience.

Contents

Abstract v

Acknowledgements vii

1 Introduction 1
1.1 Automotive Communication Networks 1
1.1.1 History 2
1.1.2 Classification 3
1.1.3 Requirements 4
1.1.4 Common Bus Systems 5
1.1.5 Other Bus Systems 9
1.2 Automotive System Design 10
1.2.1 State of the Art 10
1.2.2 Challenges 12
1.2.3 AUTOSAR 13
1.3 Motivation 18
1.3.1 New Design Paradigms 18
1.3.2 Potential for Optimization 18
1.4 Evolutionary Computation 19
1.4.1 Basic Terminology 20
1.4.2 History 23
1.4.3 Encodings 29
1.4.4 Operators 30
1.4.5 Exploration vs. Exploitation 32
1.4.6 Multiobjective Optimization 33
1.4.7 Problem Complexity 41
1.4.8 Standard Combinatorial Problems 43

2 Automotive Network Optimization **47**
2.1 Requirements Analysis 47
2.1.1 Process Integration 47
2.1.2 Versatility 49
2.1.3 Robustness 50
2.2 Problem Definition 50
2.2.1 Mapping Optimization 50
2.2.2 Topology Optimization 51
2.2.3 Discussion 54
2.3 Challenges 54
2.3.1 Communication Requirements 55
2.3.2 Computation Resources 55
2.3.3 Wiring Harness 56
2.3.4 Reliability 56
2.4 Our Approach 57
2.4.1 Rationale 57
2.4.2 Problem-Specific Encoding 59
2.4.3 Feasibility Preservation 59
2.4.4 Advanced Operators 60
2.4.5 Enhancements for Multiobjective Optimization . . 60
2.5 Other Approaches 61
2.5.1 Mapping Optimization 61
2.5.2 Topology Optimization 63
2.5.3 Mapping and Topology Optimization 64

3 jNetOpt **67**
3.1 Software Architecture 67
3.1.1 General Workflows 69
3.2 Input Objects 72
3.2.1 Proof of Concept 73
3.3 Encodings 74
3.3.1 ApplicationMatrix 75
3.3.2 BusMatrix 75
3.3.3 CommunicationMatrix 76
3.4 Operators 76
3.4.1 Operators for Mapping Optimization 77
3.4.2 Operators for Topology Optimization 78

3.5 Objectives 80
3.6 Algorithms 81
3.6.1 OnePhaseSPEA2 82
3.6.2 MappingSPEA2 83
3.6.3 TopologySPEA2 84
3.7 Graphical User Interface 85
3.7.1 Controller 86
3.7.2 View 87

4 Experiments and Results **89**
4.1 Guided Mapping Mutation 89
4.2 Guided Topology Mutation and Routing 92
4.3 Many-Objective Optimization 94
4.4 Discussion 100

5 Conclusion **103**

List of Publications **106**

List of Figures **108**

List of Tables **110**

Bibliography **113**

1 Introduction

I want to begin this thesis by giving the reader an overview of automotive communication networks, focusing on their history, classifications and currently used techniques.

This topic is followed by automotive design methods, first focusing on most recent developments in automotive engineering. As electronic functionalities in the car are increasing, new design paradigms have to be introduced for EE-architectures. Therefore, particular focus is given in this chapter to AUTOSAR as a means of tackling current challenges in the automotive design flow.

These new design methodologies offer previously unavailable degrees of freedom for system architectures. Consequently, a novel optimization potential is found in the design of in-vehicle networks. This optimization potential is the main motivation for this thesis and will be discussed as the concluding part of the automotive theme complex.

To explore the mentioned optimization potential, this work utilizes evolutionary computation. In the final section of the introduction, I will give a brief historical overview and familiarize the reader with the most common aspects and features of this technique. The chapter also covers combinatorial optimization as a prime application, providing a brief discussion on decision problem complexity and well-known examples.

1.1 Automotive Communication Networks

First, automotive communication networks are explained. I will start by giving a brief history of electronic components in the car. Then I will

explain the need for communication between such components and the resulting requirements for automotive communication systems. Consequently, selected bus systems and their applications will be described in more detail. I will conclude the first section by looking at less common bus systems, their distinctions and typical applications.

1.1.1 History

Electronics have been a fundamental part of cars since the beginning of commercial vehicles. First applications were as simple as headlights and starter motors. The first electronic fuel injection system, formerly called "'Brain Box"', was introduced in the late 1950s[1] but suffered from inferior reliability of electronic components at that time. However, as transistors and capacitors became more reliable, the electronic injection made a comeback from the the german company Robert Bosch Gmbh. Called the Bosch D-Jetronic, it had the first series production in the Volkswagen VW 1600 LE/TLE. This was the first control system in a car realized by an Electronic Control Unit (ECU) in contrast to mechanical solutions. More applications for electronics followed due to more reliable hardware components and increased computing power in embedded systems. Today, the Electric/Electronic (E/E) domain is one of the most important sources of revenue for vehicle producers.

With the increasing applicability of electronic functions in the domains of engine management and comfort, the number of such ECUs for applications such as cruise control or anti-lock braking grew steadily. In order to operate, measurements and signals had to be exchanged between those ECUs using analog transmission, i.e. voltage levels on discrete wires. It was just a matter of time until the increasing communication demands justified an automotive bus system for digital signal transmission to reduce the wiring effort. The first series production of the Controller Area Network (CAN) [1], a standard automotive bus system still used today, was in 1991 and connected 5 ECUs.

[1] E.g. US Patent No. 2980090, "'Fuel injection system"', filed april 1957

Since then, the extent of electronic functions in cars has been increasing at a rapid rate. Likewise, communication demands have been growing drastically and soon were not be able to be served by one CAN bus. The first heterogeneous bus system topology emerged, connecting all components required for engine and transmission management on a distinct bus system while all comfort related functions communicated on a separate, slower bus. This separation between the so called power train and body bus system is still used today although extended by several other communication domains for infotainment or safety-related features. Additionally, cost effective bus systems are used for local applications where bandwidth is of no major concern. All these networks are connected by gateways to allow cross-domain communication and to provide a single point of entry for diagnostic capabilities.

1.1.2 Classification

Automotive bus systems are classified by their speed and typical application. A frequently used classification system was introduced by SAE [2] and defines 3 types of bus systems. With the increasing bandwidth demand of X by wire and infotainment applications, two further classes were added. An overview of this classification system can be seen in Table 1.1.

Table 1.1: Classifications of automotive bus systems

Class	**Bitrate / sec**	**Typical Application**	**Typical Bus System**
A	<25 kBit	Sensor/Button Interface	LIN
B	25 - 125 kBit	Body Electronics, Comfort	Low Speed CAN
C	125 - 1000 kBit	Powertrain	High Speed CAN
D	1 - 10 MBit	X By Wire, Active Safety	FlexRay
Infotainment	>10 MBit	Video Streaming, Internet Connectivity	MOST, Ethernet

1.1.3 Requirements

In contrast to other domains, automotive applications impose unique requirements on communication systems. Communication reliability and determinism arise from safety related applications within the car to support the driver and assure functionality of real-time dependent features. The ubiquitous cost pressure in the automotive industry also affects the design of efficient network topologies and choice of cheap and lightweight cables.

Reliability

Automotive bus systems are designed to work under harsh environmental conditions regarding temperature, vibration and humidity. Furthermore, they should be robust against electromagnetic interference and not cause disturbances based on their own radiated emissions. To detect transmission errors, most communication systems employ CRC checks and robust encoding of raw data. Another way to significantly increase transmission reliability is to transmit data by current modulation instead of voltage levels. However, corresponding hardware interfaces are costly and the power consumption of such bus systems is usually higher.

Determinism

Real-time control systems in cars are often distributed over several sensors, ECUs and actuators. To ensure a proper execution of such systems, stringent timing requirements from signal source to sink have to be guaranteed. Therefore, several automotive systems (e.g. TTCAN, FlexRay) employ time triggered communication modes, where a distinct and recurring time slot is assigned for each signal. However, this is not the case for the standard CAN bus although the worst case transmission time of this popular bus system is very well studied [3].

Costs

Due to the highly competitive market and large production volumes, engineers are eager to employ the cheapest possible solutions wherever possible. In terms of bus systems, typical cost factors are complexity of the communication controller, the type of cable used as well as required CPU processing power and oscillator quality for proper communication. This also resulted in a hierarchical communication structure of low-cost class A and B bus systems interconnected by a faster backbone network.

1.1.4 Common Bus Systems

This section introduces the most common bus systems found in vehicles today.

Controller Area Network

The CAN bus is the standard vehicular bus system with many other applications (for example in industrial automation or aerospace). After the first series application in 1991 [4], the bus was standardized in ISO 11898 and SAE J2284. Since 2008, CAN is the only interface allowed for diagnosis and emission testing using the on-board diagnostics (OBD) standard for newly introduced cars [5].

The physical layer is represented by two distinct specifications. Low Speed CAN (ISO 11898-3) is used for Class B applications with bit rates < 125 kbit/sec. For Class C applications and higher transmission rates, High Speed CAN (ISO 11898-2) is typically used at 250 or 500 kbit/sec. Both specifications transmit via unshielded twisted pair (UTP) cables where High Speed CAN requires 120 Ω termination resistors to reduce signal reflection.

The multi-master design implements a 'Carrier Sense, Multiple Access with Collision Resolution' (CSMA/CR) transmission scheme where bit-wise arbitration resolves collisions on the physical medium. Further

features include bitstuffing to force signal transitions after five consecutive bits. This allows network participants to remain synchronized using the bus communication as reference and utilize cheaper clock generators.

Other developments in CAN technology include TT-CAN, a time triggered and deterministic extension (ISO 11898-4). Further, the increased bandwidth requirements have lead to the introduction of the Flexible Data-Rate (FD-CAN) protocol. Here, the bit times of payload data are drastically decreased to transmit more information in a standard CAN frame. The major advantage of FD-CAN is that it can be included in existing CAN networks without disturbing standard communication [6].

Local Interconnect Network

The Local Interconnect Network (LIN) [7] was developed for cost sensitive networks with low bandwidth requirements. Typical applications for LIN include localized networks, for example the connectivity within a driver's door.

Bit transfer rates are usually standardized at 9.6 or 19.2 kBit/sec, which results in the Class A categorization of the bus. The similarity to well-known UART bit rates is intentional in order to re-use common and very cheap clock generators. In fact, the common bit rate amongst other similarities to UART allowed chip designers to integrate LIN into existing serial communication modules in a very cost efficient way.

Each network consists of one dedicated master, usually with gateway functionality to a higher level CAN network. A transmission is initiated by this master while each slave node matches its internal clock using a sequence of synchronization bytes. Therefore, the reference clock of slave nodes does not require high precision and can be realized using cheap RC oscillators.

All these features allowed the design of an extremely cost efficient network, resulting in a firm positioning on the market next to the established CAN bus.

FlexRay

The development of FlexRay [8] began in 1999 with the goal of a deterministic and robust bus system for high bandwidth requirements. Those requirements resulted from new applications in the area of vehicle dynamics such as adaptive dampening systems for each wheel. The main driver behind the technology was BMW, which also developed the preceding Byteflight bus system and produced the first series vehicle utilizing FlexRay in 2006.

The physical layer uses one unshielded twisted pair of cables (Channel A) with an optional second pair (Channel B) for fault tolerance or bandwidth enhancement. The maximum data transfer speed is specified at 10 Mbit/sec for each channel using differential voltage levels for transmission.

FlexRay uses a mixed synchronous and asynchronous TDMA scheme in a multi-master architecture. This allows transmission of periodic signals with stringent real-time requirements and hard deadlines in the synchronous or static segment of a communication cycle. On the other hand, event triggered signals can be placed efficiently in the dynamic segment.

Although the system clock is always synchronizing between network participants, the extremely high data rate requires precise reference clocks with very low drift for each node. Therefore, the resulting hardware costs are high compared to CAN networks. As a consequence, FlexRay is currently only utilized in premium segment cars.

MOST

The Media Oriented Systems Transport (MOST) [9] bus was developed for high-bandwidth infotainment applications in vehicles. Supervised by the MOST Cooperation, a consortium of vehicle manufactures and supplier companies, the bus system quickly became the standard interface for multimedia and navigation purposes in the car.

MOST implements a synchronous ring topology with data rates specified at 25, 50 and 150 Mbit/sec. In contrast to all other automotive network technologies, MOST primarily uses optical fiber as the transmission medium. While this medium is immune to electromagnetic interference, wiring costs are considerably higher compared to standard copper technologies. However, MOST50 also specifies an electric physical layer using unshielded twisted pair cables.

The MOST Cooperation provides all standards and network services for all OSI layers. Therefore, MOST had very quick market penetration compared to other bus systems and is used in nearly all prominent car brands.

Ethernet

Being the standard communication technology for personal computers for decades, Ethernet was not considered as an automotive bus system for a long time. The first use case for Ethernet-based communication emerged due to the vast amount of ECU-software in premium cars. In 2008, the process of programming all ECUs with the newest firmware version was accelerated considerably by BMW, using Ethernet-capable embedded systems. However, the network was only intended for usage in a production facility or repair shop. Robust data transmission in driving conditions could not be provided with standard Ethernet technology due to EMC reasons.

A new approach is now conducted by the OPEN Alliance SIG (One-Pair Ether-Net, Special Interest Group)[2], a consortium of car manufacturers and automotive suppliers. The technology BroadR-Reach [10] is based on a 100 Mbit/sec, full duplex physical layer via unshielded twisted pair wires. This standard was invented with respect to automotive requirements concerning EMC and transmission reliability. Software protocols are being implemented to support the capability of real-time communication. Due to the high bandwidth and low costs, Ethernet

[2] http://www.opensig.org/

is now a serious competitor to the established FlexRay and MOST technologies for future in-car communication.

1.1.5 Other Bus Systems

This section gives a short overview of automotive bus systems for specific purposes. Currently, these networks are not considered for optimization as they are usually part of functional packages that communicate over separate connections.

PSI5

The Peripheral Sensor Interface 5 (PSI5) [11] is used for communication with intelligent automotive sensors. It supports synchronous and asynchronous communication modes in a point to point topology. A bus topology with one master and multiple slaves is only possible in a synchronous configuration. Sensor data is transmitted using current modulation over power supply lines at a usual speed of 125 and up to 189 kbit/sec. This modulation technique leads to a very robust communication and high immunity to interfering electromagnetic fields, which is why PSI5 is an interface typically used for airbag crash sensors. The robustness comes of course at the price of increased power consumption and complex bus coupling hardware.

SENT

Another commonly used sensor interface is the Single Edge Nibble Transmission (SENT) protocol, standardized as SAE J2716 [12]. In contrast to PSI5, it specifies a unidirectional pulse width modulation over a discrete wire. Due to the pulse width modulation, the effective data rate depends on the data values to be transmitted and can go as low as 30 kbit/sec. In each message frame, a synchronization impulse followed by two 12-bit measurement values is transmitted. The receiver measures the

length of each synchronization impulse and the following data pulses, which allows the utilization of very cheap RC oscillators at the sensor.

DSI

The Distributed Systems Interface (DSI) [13] specifies a synchronous communication technique for up to 15 sensor nodes. Command data is transferred from the master by modulating the voltage of the power supply line. Sensors can respond by modulating their current consumption using one of three defined levels. The communication offers transfer speeds between 125 and 200 kbit/sec, comparable to PSI5.

1.2 Automotive System Design

Literature states that a modern premium class car contains between 70 and 100 ECUs [14, 15], realizing a manifold of electronic functions. These ECUs are connected by a variety of different bus systems to transport and exchange data. Another trend is that a lot of these features and functionalities are no longer reserved for luxury vehicles. Currently, customers of medium class cars are also demanding features related to safety and driver assistance. This market demand leads to a challenging size and complexity of in-vehicle networks in cost sensitive markets. Therefore, new design methods for efficient realization have to be applied while focusing on the stringent reliability requirements of the automotive industry.

1.2.1 State of the Art

The design of a new vehicle's electric and electronic architecture begins by defining the functional and nonfunctional requirements and features. This is done by the vehicle manufacturer (OEM) and is usually based on previous car series and current market demands.

These requirements are divided into subsystems and components. Specifications for each functionality are then defined or adapted from similar car models. Tier-one suppliers are assigned to deliver the most components. Only some leading edge technologies are developed exclusively by the OEM to distinguish the own product from competitors on a technical level. However, the main roles of the OEM are the specification of the product, integration of each component and verification of interoperability at a system level before the start of production (SoP).

Using the current approach, subsystems are often defined as all functions that are executed on one specific ECU. Due to this partitioning, the supplier delivers a subsystem as a set of sensors, actuators and dedicated ECU. The automotive operating system and software components executed on this ECU reflect the know-how of the supplier and are often delivered as closed source or "black box" [16]. This also implies that the OEM is not able to accommodate additional functionality on such ECUs without support from the supplier.

This design approach led to the development of highly specialized and reliable control systems as for example engine control or 'Electronic Stability Programs' (ESP). On the other hand, the electronic architecture is also very inflexible because the required effort to adapt one ECU to accommodate additional functions is significant. Therefore, it is often cheaper to realize new functions as separate ECUs. This is one of the reasons for the large amount of ECUs installed in cars today.

Another reason lies in the heterogeneity of car variants and features that are configurable by the customer. For example, if the customer orders a car without the feature 'Park Distance Control' (PDC), the costs of production can be reduced by omitting the park distance sensors and ECU. This is only possible when the electronic architecture of the car includes one ECU solely dedicated to the feature PDC.

As the amount of ECUs grows, the communication requirements between them also increase. Therefore, new bus systems had to be added to the existing communication topology while ensuring cross-network communication using a central gateway connected to all bus systems. Next to the established - usually CAN based - buses dedicated to power train and comfort functions, new communication networks appeared

in modern cars. One prominent representative is the MOST bus for the infotainment domain. Also, several networks based on LIN were introduced and usually integrated as subsystems of the body/comfort CAN bus.

Concluding, the introduction of new electronic features led to an organic growth of new ECUs and bus systems. To ensure cross-network communication, all bus systems are interconnected by a central gateway. Due to different suppliers and closed source software, the global communication architecture of a modern car is very inflexible. For example, changes in an existing CAN communication matrix would have to be communicated to all suppliers of related network participants. Further, the re-usage of existing software solutions in different projects often requires manual adaptations of the software.

1.2.2 Challenges

Considering the previously described trends, several challenges to the automotive design methods, component suppliers and OEMs can be identified. As this is just a short overview of some aspects, the interested reader is referred to [17] and [18] for a more detailed analysis.

Reliability

New electronic features rely on sophisticated software algorithms. A fully equipped premium car can include up to 100 million lines of code [19] and software errors in safety-related components are a serious issue for OEMs. Therefore, measures to ensure robust generation and execution of software have to be taken. This also includes software testing procedures and coding quality standards.

Common Software Standards

Automotive software features compete on a functionality level that is experienced by the customer. This includes, for example, the visual representation of user interfaces or the response quality of stability control systems. The underlying operating systems and interfaces for flashing, configuring and diagnosis are not prone to competition and the relevant requirements are similar for most car manufacturers. Therefore, several initiatives to standardize such interfaces are successfully employed throughout the automotive software domain.

Cost Pressure

The shift of feature availability from premium segment to the whole portfolio of a car manufacturer also impacts on the cost pressure of new developments. The production volumes of medium class cars are much higher compared to the premium class. Therefore, every saving in unit production costs results in a higher revenue for the suppliers and OEMs.

Complexity

The vast amount of functions, different configurations and cross-function communication leads to very high system complexity. Consequently, a lot of OEMs have adopted model driven paradigms in their architecture process to handle the manifold of requirements and design challenges [20, 21, 22].

1.2.3 AUTOSAR

The most prominent initiative for handling the challenges mentioned in this context is the ”AUTomotive Open System ARchitecture” (AUTOSAR) [23]. It is a software architecture standard supported by the majority of global car manufacturers, suppliers and tool developers [24].

The goals are to provide a uniform development process for automotive systems and defined interfaces between a layered software architecture. AUTOSAR provides the standards and corresponding meta-models for model driven development. The actual implementation of operating systems and functional software is left to the automotive companies. Therefore, the AUTOSAR principle is also to "Cooperate on standards, compete on implementation" [23].

Development Process

The AUTOSAR development process is influenced by Model Driven Architecture (MDA) and meets the requirements for the design of highly complex systems. Further, it reflects the challenges of outsourcing the development of interconnected components to various suppliers.

The E/E architecture is defined by a standardized system configuration, consisting of functional software descriptions, hardware resources and system constraints. From this system configuration, several ECU-specific descriptions can be extracted. Such an extract contains all the information required to develop the hardware and firmware of one ECU. It can be seen as an XML-based interface between the OEM as system developer and tier-one suppliers.

Based on this interface, the ECU supplier is then tasked with the development of hardware components and embedded software realized by several layers of abstraction. A hardware abstraction layer (MCAL) represents the bottom of the AUTOSAR software stack and is usually provided by the chip vendor. The actual operating system and basic software modules for tasks such as communication or memory services are generated by according development tools. This is also the case for the Runtime Environment (RTE) which represents the AUTOSAR middle-ware.

Functional Software Components (SWCs) can be developed or generated independently of the hardware and basic software development. They represent the high-level features and actual functionality or application of the developed ECU. The interfaces defined in AUTOSAR allow the

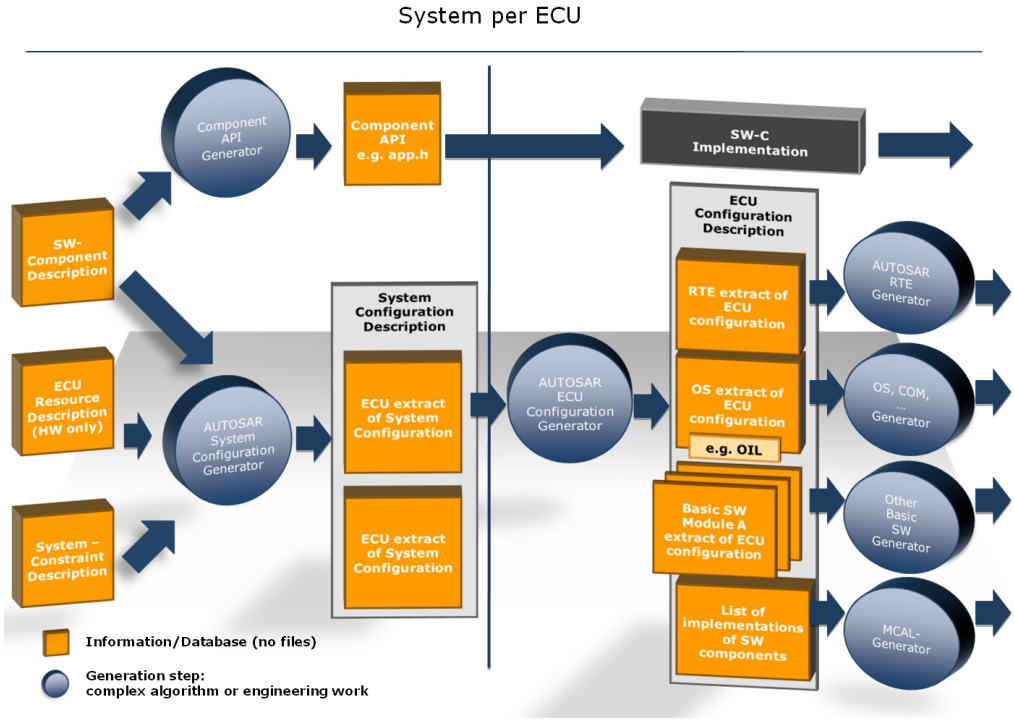


Figure 1.1: AUTOSAR development methodology [23]

execution of such software components without detailed knowledge of the underlying hardware layers. Furthermore, this development process allows a distinction between the development of functional applications and underlying embedded software and hardware. A chart of the whole development methodology is depicted in Figure 1.1 while the layered architecture of ECU software can be seen in Figure 1.2.

Modularization and Virtual Functional Bus

An important feature of AUTOSAR is the modular design of software components. Due to the model driven approach, functional components and their interconnections can be designed on a logical level. This allows a top-down development for system architecture; starting from

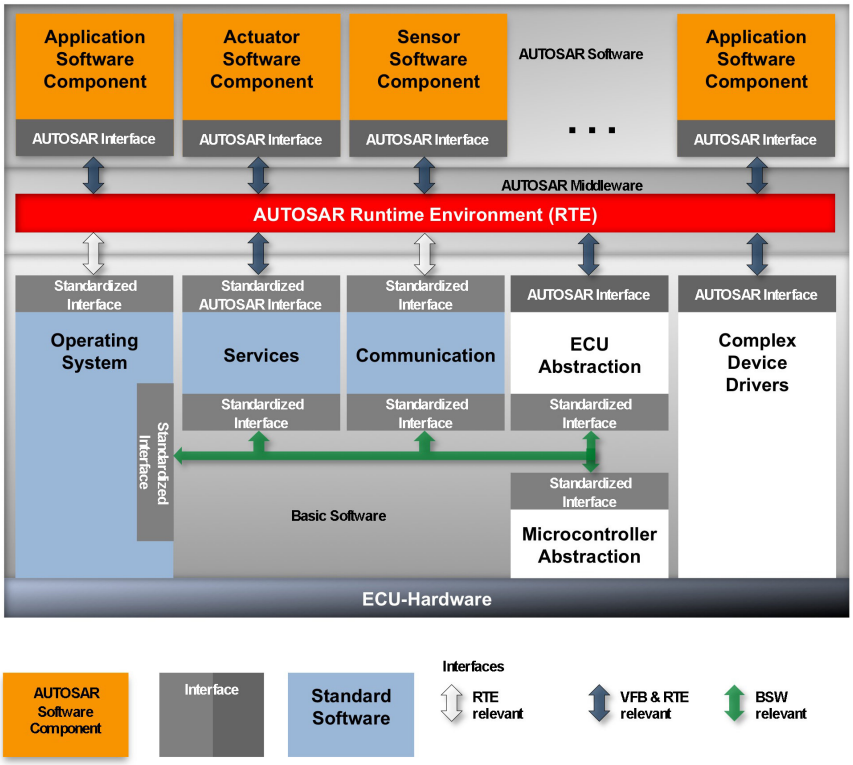


Figure 1.2: AUTOSAR ECU software architecture [23]

requirements and features, over realization by software components, to actual deployment on hardware units.

Automotive features often require communication between software components throughout the car. During the system configuration process, such communication requirements are modeled and specified. From the perspective of an application developer, all communication with other SWCs is handled by the Virtual Functional Bus (VFB). This abstract interface defines the exchange of messages with other applications. At this level of abstraction, it is irrelevant if the communicating software components are executed on the same ECU or connected via bus systems and gateways.

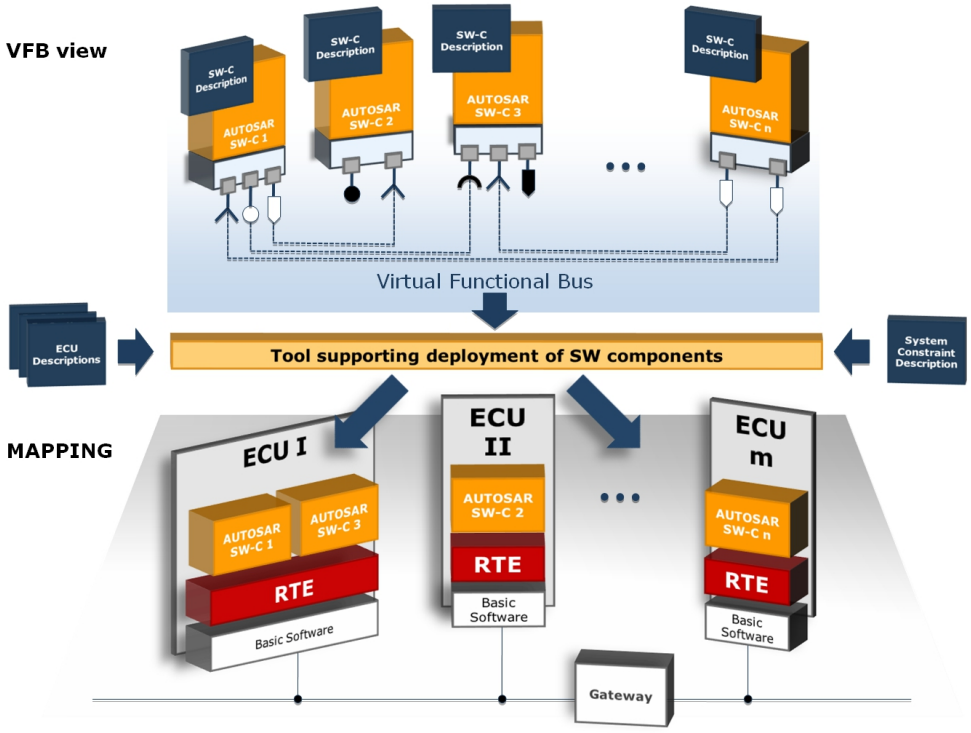


Figure 1.3: AUTOSAR basic approach [23]

The benefits of modular software components and an abstract communication model are manifold. Firstly, software components can be deployed and executed on various ECUs without altering the application source code. This leads to higher flexibility for the system architecture. Secondly, functional applications can be re-used on other hardware platforms and other vehicle series. Therefore, the testing effort is reduced and more robust software can be delivered at lower development costs. Thirdly, applications from various suppliers can be integrated on the same ECU. This revolutionizes the current development process where the combination of ECU and embedded firmware is usually provided by the same supplier. An overview of this approach is given in Figure 1.3.

1.3 Motivation

Due to the changes in automotive design methods, new paradigms and circumstances influence the automotive EE architecture. This affects all hierarchy levels, from ECU software development to network topology design.

1.3.1 New Design Paradigms

Automotive network topologies and distributed functions have been established historically. The partition of networks and assignment of functions to ECUs was based on best practices. Changes to this composition were difficult to impose as suppliers also had their portfolio optimized for these architectures.

This is not the case anymore as initiatives like AUTOSAR impacted several elements of the design of electronic architectures:

- Functional software is not necessarily bound to be executed on a specific ECU
- The design of a logical system-wide architecture between software components is possible
- Mapping functional software onto different ECUs influences network communication requirements
- Suppliers can focus on their core competences, e.g. hardware, basic or functional software
- Code-generation from behavioral modeling languages is feasible

1.3.2 Potential for Optimization

The new design paradigms that have been described lead away from classical architecture approaches. As pointed out in [17]: “A systematic top down design was never used. If we would not go in evolutionary steps but re-design the hardware/software systems in cars from scratch today, we would certainly come up with a quite different solution.” This

aspect promises new optimization potential for the system architecture. Some optimization goals that can be formulated in this area are:

- Omitting ECUs by moving all previously mapped functional software to other ECUs
- Reducing weight and length of the cable harness by optimal placement of gateways and topology layout
- Decreasing inter-ECU communication by mapping closely related software components onto the same ECU
- Automatically exploring design alternatives based on quality metrics
- Assisting design engineers by suggesting possible system architectures

In this thesis, the design of automotive network architectures in the electric/electronic domain is interpreted as a combinatorial optimization problem. I want to provide a reliable resolution to this problem in order to support automotive engineers in their architecture design process. Due to the complexity of distributed electronic features and their heterogeneous communication requirements, I want to use evolutionary computation as an optimization technique.

1.4 Evolutionary Computation

There are several motivations to why the scientific community is interested in evolution and ways of simulating evolutionary processes. The most important reason for this work is the optimizing behavior of evolution for complex problems. Other motivations are the development of machine intelligence, robust adaptation or as a tool to gain more knowledge of natural evolution processes.

In connection with this thesis, we want to view Evolutionary Computation (EC) as a subset of stochastic optimization methods. The main usage is for complex optimization problems where deterministic methods are not feasible due to one or several of the following reasons:

- The dimensionality of the search space is too high to be covered in reasonable time
- Non-linearities in objective functions prevent the calculation of derivatives and therefore the application of derivative-based deterministic optimization techniques
- The optimization problem contains several local optima (multimodal)
- Objectives are distorted by random noise

1.4.1 Basic Terminology

Evolutionary computation methods are inspired by the concept of Neo-Darwinism, which is a modern interpretation of Charles Darwin's work on natural selection [25], extended by principles of genetics to model mutation and inheritance. According to this concept, the development of life can be deduced from the processes of reproduction, mutation, competition and selection. In nature, the evolution of species is a flexible and creative task with a high degree of parallelism. As engineers, we want to make use of those characteristics to solve complex problems as "Darwinian evolution is intrinsically a robust search and optimization mechanism" [26].

Optimization

In our field, optimization is the search for the best solution to an optimization problem from a set of available solution candidates. The optimization problem maps an outcome for each possible solution candidate, which can be seen as a number of decisions. Each decision that has to be made for a problem represents one dimension in the solution space. Consequently, a problem that requires n independent decisions to be made leads to solutions consisting of n decision variables. The set of all solution candidates is then a n-dimensional search space.

From a mathematical point of view, an optimization problem is defined as the minimization of an objective function or cost function $f : \mathbb{R}^n \rightarrow \mathbb{R}$

in the following form: $min(f(\mathbf{x}))$ with $\mathbf{x} \in \mathbb{R}^n$ being a n-dimensional decision vector or solution within the search space $\mathbb{R}^n$.

Most publications throughout literature deal with minimization problems (e.g. minimum cost or weight). However, without loss of generality, an objective function given as maximization problem (e.g. maximise facility productivity) can be transformed into a minimization problem by stating $f_{min}(\mathbf{x}) = -f_{max}(\mathbf{x})$.

Depending on the problem, decision variables can be subject to equality and inequality constraints, which limit the search space to a feasible region A:

$$\begin{aligned} A = \{\mathbf{x} \in \mathbb{R}^n \mid c_i(\mathbf{x}) \geq 0 \, \forall \, i \in \{1, \ldots, p\} \wedge \\ c_i(\mathbf{x}) = 0 \, \forall \, i \in \{p+1, \ldots, q\}\} \end{aligned} \tag{1.1}$$

The goal of the optimization is to find the global optimum f^* with solution vector $\mathbf{x}^*$ such that:

$$\forall \mathbf{x} \in A : f^* = f(\mathbf{x}^*) \leq f(\mathbf{x}) \tag{1.2}$$

For some problems one might not only be interested in the global optimal solution but also in other satisfactory solutions. Such multimodal problems include a set of local optima f' in an ϵ-neighborhood N_ϵ around $\mathbf{x}'$:

$$f(\mathbf{x}') \leq f(\mathbf{x}) \; \forall \mathbf{x} \in N_\epsilon(\mathbf{x}') \tag{1.3}$$

$$N_\epsilon(\mathbf{x}') = \{\mathbf{x} \in A \mid 0 < \|\mathbf{x} - \mathbf{x}'\| < \epsilon\} \tag{1.4}$$

Further, some optimization problems might have more than one and most likely contradicting optimization goals and therefore more than one fitness function. This leads us to the field of Multi Objective Evolutionary Algorithms (MOEA, subsection 1.4.6).

Evolutionary Methods

In order to mimic an evolutionary process on a computer and apply it to an optimization problem, several abstraction steps have to be taken.

First, the optimization problem has to be formulated in a way that it may be processed by a computer. This involves the representation of a solution candidate $\mathbf{x}$ in a computable format. Such a representation might be a set of variables that directly encode decisions for the optimization problem (phenotype). However, some evolutionary algorithms like genetic algorithms use binary representations as simplified search space for inheritable information. This simplification (genotype) cannot be directly applied to the fitness function as it only represents a building plan for the actual decision variables. A genotype-phenotype decoding or mapping is then performed to evaluate the quality of the proposed solution.

Following genetic terminology, the entity of all decision variables of one solution is called an individual or chromosome. A set of individuals is then known as a population. Further, each decision variable in a chromosome is called a gene and the actual value of this variable is an allele. Usually, the chromosome initialization is done with random values but problem-specific knowledge might also be utilized if available.

To realize competition within a population and ultimately reach an optimization goal, a quality function has to be defined. Based on this function, fitness values can be assigned to competing solutions and a selection mechanism will choose promising individuals for reproduction and mutation (survival of the fittest).

These basic steps of a generic evolutionary algorithm are stated in algorithm 1, with nomenclature taken from [27, 28]. It is an iterative process where an offspring population is generated from parent individuals using crossover and random mutation. The most fit individuals are then selected to form the next parent population. One iteration of this process is called a generation.

Algorithm 1 Basic Evolutionary Algorithm

1: t := 0;
2: Generate initial population $P(t)$
3: Evaluate fitness for each individual in $P(t)$
4: **repeat**
5: Create offspring population $P'(t)$ from $P(t)$ using variation operators
6: Evaluate fitness for $P'(t)$
7: Select individuals for next generation $P(t+1)$
8: t := t+1;
9: **until** Stop condition is met

1.4.2 History

First efforts to utilize evolutionary theories for optimization problems can be dated back to 1957 [29] with other influencing works a few years later [30], [31]. A more detailed review of the early approaches can be found in [32]. However, the publications received little attention as the success of the proposed methods was greatly dependent on available computer platforms and calculating power. It took another 15 years of computer development for three new approaches to emerge:

- Evolution Strategies (ES) by Rechenberg [33] and Schwefel [34] in Berlin, Germany
- Genetic Algorithms (GA) by Holland [35] and Goldberg [36] in Ann Arbor, Michigan
- Evolutionary Programming (EP) by Fogel [37, 38] in San Diego, California

While all 3 paradigms try to emulate natural evolution in some way, they differ in the usage of crossover and mutation operators as well as encoding of solutions.

Evolution Strategies

One of the first prominent applications of evolution strategies was the design of a two-phase nozzle [39] with the result depicted in Figure 1.4 and described in [40]:

> *The perhaps optimal, at least unexpectedly good and so far best-known shape of the nozzle was counterintuitively strange, and it took a while, until the one-component two-phase supersonic flow phenomena far from thermodynamic equilibrium, involved in achieving such good result, were understood.*

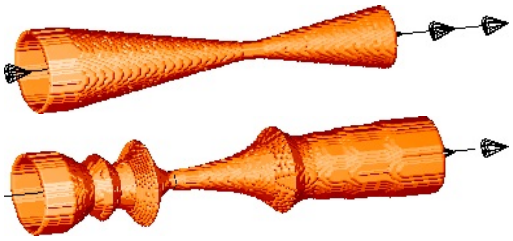


Figure 1.4: Starting solution and final nozzle shape optimized by evolutionary strategies[3]

Evolution strategies use a vector of n real-valued genes as representation to optimize fitness functions of the form:

$$f : \mathbb{R}^n \rightarrow \mathbb{R} \tag{1.5}$$

The primary operation is mutation, which is realized by adding a random value to each entry of the vector. This value has a Gaussian distribution with mean value of zero and variable standard deviation σ, which is an indicator for the mutation strength. In its basic form, evolutionary strategies create only one offspring from one parent using mutation, which is known as the (1+1)-ES. If the created child has a superior fitness value over the parent, it will replace it and act as a new parent in the next generation (truncation).

[3] Source: http://ls11-www.cs.uni-dortmund.de/people/schwefel/EADemos/

When keeping track of the recent successful and unsuccessful mutations, one can use their relation to each other as a guiding function to adjust the mutation strength. This was observed very early on [41] and became known as the "One-fifth success rule" [42]. It states that the success rate of an ES should be around $1/5$. If the success rate is higher then the search is too local and therefore σ should be increased. On the other hand, if the success rate is lower than $1/5$ then the mutation strength should be decreased as better solutions are more likely to be found near the current best. This self-adaptive behavior is a very powerful strength of ES.

Another important aspect of ES is the selection strategy and population size. As stated before, the basic (1+1)-ES creates one offspring from one parent and selects the best of those two solutions. The first algorithm to introduce a population was the "(μ+1)- or steady-state ES"[40]. Here, one offspring is usually created from one of μ parents. All parents are equally likely to be chosen for reproduction while other strategies might prefer individuals with a high level of fitness. Finally, the least fit solution from all parents and offspring is removed from the population and the next generation starts again with μ parents.

The next logical step was the creation of more than one offspring per generation which leads us to the (μ+λ)-ES, where λ offspring are created from μ parents. Again, the best individuals from all parents and offspring solutions are selected for the next generation. This is also known as an elitist technique when superior parents can outlast the offspring. In contrast to this, the (μ,λ)-ES only selects μ parents from the produced offspring; even if some parents might have a better fitness value. According to [40] and [43], the comma selection is the preferred method for unbounded search spaces while the elitist selection variant should be used for discrete optimization problems with limited search space. Further, elitist selection tends to converge into local optima which makes the comma selection better suited for multimodal optimization and changing environments or quality functions.

Another motivation for multimembered ES was the improvement of self-adaptation. Instead of one global mutation strength σ, each individual in the population inherited a distinct step width σ_i for each gene from

its parent solutions using recombination. In other words, the ES adapted the mutation strength for each variable or dimension in the search space separately. Further techniques like the $(\mu/\rho+\lambda)$-ES define a mating pool where ρ parents are selected for reproduction. Since then, a lot of research has been done on optimal values for ρ, μ and λ as well as different selection schemes [44, 45].

Genetic Algorithms

In contrast to evolution strategies, genetic algorithms were designed to use a genotypic binary representation instead of real-value variables. This representation is usually a bit string with defined length l and the according optimization problem is in the form of:

$$f : \{0,1\}^l \rightarrow \mathbb{R} \tag{1.6}$$

The genotype form is especially suitable for combinatorial problems consisting of discrete on/off decisions. For continuous parameter optimization with decision vectors $\mathbf{x} \in \mathbb{R}^n$, such an approach is not feasible. Therefore, it is common to split the bit string into n segments and decode each segment with an according integer representation [27]. This is a very basic and common genotype-phenotype mapping. All variation operations in genetic algorithms are performed on the genotype while fitness evaluations are calculated using the decoded phenotype. Further, GAs always use a population of several chromosomes compared to the early two-membered approaches in ES.

The reason for the string representation lies in the schemata theorem [46]. This theorem states that subsets of successful bit combinations have a higher probability of surviving through recombination and mutation for several generations. In other words, "Highly fit, short-defining-length schemata (we call them building blocks) are propagated generation to generation by giving exponentially increasing samples to the observed best;"[36].

The selection of reproduction candidates is done with respect to the fitness of the corresponding phenotypes which is also known as roulette wheel selection. Consider a genotype-phenotype mapping function $h : \{0,1\}^l \rightarrow \mathbb{R}^n$ and a resulting fitness value $f_i = f(\mathbf{x}_i) = f(h(\mathbf{g}_i))$ where $\mathbf{x}_i$ is the real-valued phenotype and $\mathbf{g}_i$ the according binary genotype of one individual. Each individual in the population P with the size of s can be selected for reproduction with the probability:

$$p_i = \frac{f_i}{\sum_{j=1}^{s} f_j} \tag{1.7}$$

Caution must be taken when working with high fitness values and small relative differences within the population. The roulette wheel selection would choose between the individuals with nearly equal probability [26]. Scaling functions or ranking of individual fitness values can be applied to solve such problems.

The main variation operator in genetic algorithms is crossover. This is in contrast to evolution strategies where mutation is the main operation. The first implementation was the single point crossover, where the bit string of two parent chromosomes $\mathbf{g}_1$ and $\mathbf{g}_2$ is cut off at the same random location x with $\mathbf{g}_1, \mathbf{g}_2 \in \{0,1\}^l$ and $0 \leq x \leq l$. Two offspring chromosomes are created by merging the first part of $\mathbf{g}_1$ with the last part of $\mathbf{g}_2$ and vice versa (see section 1.4.4). Other crossover techniques are two-point crossover where the bit string is cut into three parts and uniform crossover [47] where each bit is chosen randomly from one of the parent chromosomes.

Evolutionary Programming

The goal of evolutionary programming was to develop artificial intelligence using simulated evolution. Intelligence is defined here as "(1) to predict one's environment, coupled with (2) a translation of the predictions into a suitable response in light of the given goal" [48]. In the

first applications, finite state machines were evolved to predict the next symbol from a sequence based on previous symbols.

Evolutionary programming differs from genetic algorithms in aspects of encoding and reproduction. Typically, the encoding is always directly related to the optimization problem (e.g. finite state machines). This differs from standard GAs, where a genotype-phenotype mapping is performed to link a binary representation to the real-valued problem description. Further, the mutation operator works similarly to ES, with Gaussian distributed changes for each decision variable.

Despite their individual development, the differences between EP and ES are not so obvious. For instance, the mutation operator is the same for both techniques. However, EP prefers a stochastic selection for new populations compared to the strongly deterministic and elitist method used in evolution strategies. The stochastic selection used to form a new population from current parents and offspring is realized by a tournament function. In the simplest case, 2 solutions from the union of parent and offspring sets are picked at random. Their fitness values are compared and the better solution is copied into the next generation. Further, EP considers solutions as species and therefore typically does not implement a recombination or crossover operator. On the other hand, ES treats solutions as a population of individuals where specific traits of several parents can be found in offspring individuals.

Further Development

According to [49], the three techniques were applied separately to each other until the 1990s, which is surprising given the obvious similarities and roots in Darwinian evolution. In 1993, the first journal on Evolutionary Computation [50] was published, stating that the arrival of the digital computer finally made the applicability of evolutionary algorithms feasible. Also the term Evolutionary Computation became a collective title for all previously mentioned techniques.

Today, the distinctions between those three techniques are very blurry or non-existent. Modern, hybrid algorithms often use different characteristics from all approaches.

1.4.3 Encodings

Evolutionary Algorithms can use a variety of encodings to represent the optimization decisions in a computable form. Usually, the engineer prefers an encoding scheme that reflects the most important properties of the problem to be solved. The most common schemes are listed below:

Binary

A string of binary values is the preferred method for genetic algorithms. It is also used for combinatorial optimization problems, that require a sequence of discrete on/off decisions. An example for such problems according to the typology of [51] would be the Single Knapsack Problem (see section 1.4.8).

Integer

Using a set of integer values is the preferred method for problems where the solution reflects discrete, countable decisions. An example would be the Cutting-Stock Problem [52]. Very often, such integer values are encoded into binary strings when using genetic algorithms. This genotype-phenotype mapping can use standard binary coded decimals (BCD) or Gray code. The advantages of Gray code are more homogenous changes when applying mutation operators and transitions. For example the transition between the values 7 and 8 ($0111_{BCD} \rightarrow 1000_{BCD}$ equals $0100_{Gray} \rightarrow 1100_{Gray}$) can be done using only one bit flip. This feature is also known as reduced Hamming cliff.

Continuous

Continuous encodings are the standard representation in evolution strategies. They are used for any kind of problem consisting of real-value objective functions. When applying evolutionary algorithms, it is virtually mandatory to restrict the search space from $\mathbb{R}^n$ to a smaller region by defining upper and lower bounds for each decision variable.

Permutation

Permutations are primarily used for combinatorial optimization. Depending on the problem, we distinguish two kinds of permutation.

Permutations without Repetition use a fixed set of values for which the order of those values represents the decision. A popular example is the Traveling Salesman Problem (TSP, section 1.4.8) where a number of cities have to be visited in the shortest possible way without visiting any city twice. Given n cities, the possible solution space equals $n!$. In the special case where only k out of n cities have to be visited, the resulting solution space reduces to $\frac{n!}{(n-k)!}$.

Permutations with Repetition are used, for example, in the task assignment problem. A number of n tasks have to be executed by k processor cores. The permutation variable is a vector of the length n and each entry equals a processor assignment $\{1 \dots k\}$. The solution space is n^k.

1.4.4 Operators

As stated in the beginning, Neo-Darwinism relies on the processes of reproduction, mutation, competition and selection. In evolutionary algorithms, these processes are implemented by specific operators: crossover, mutation and selection. Competition can be seen as the fight over resources represented by limited population sizes. It is performed either intrinsically through the selection operator or as part of the population handling mechanism in the evolutionary algorithm. The implementation

of mutation and crossover is highly dependent on the chosen representation while selection operators reflect the overall strategy of the evolutionary algorithm used. A lot of research has been done into each of these operators and this work can only scratch the surface by pointing out the most used implementations. More detailed information can be found for example in [53].

Crossover

The goal of the crossover operator is to produce offspring individuals reflecting some qualities of their parents to achieve overall higher fitness ratings. The most basic implementation is the single point crossover, depicted in Figure 1.5. This operator works for all decision variables

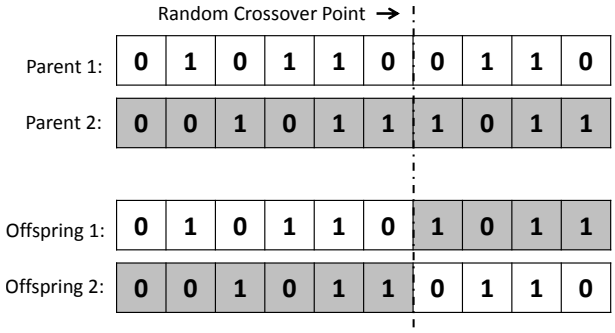


Figure 1.5: Single point crossover

based on strings of values. However, when using permutation as representation technique, repair heuristics have to be applied to ensure valid offspring permutations. The most prominent example is the Partial-Mapped Crossover (PMX) [54].

Mutation

This operator alters individuals by randomly perturbing their decision variables. Due to this randomly induced new information, the operator

can explore areas that would not be reached using only crossover operators. Again, the simplest method can be found for binary representations, where one or more randomly chosen bits in the string are inverted. For permutation encodings, swapping two bit positions offers a very fast and simple method for mutation. Adding $(0,\sigma)$-Gaussian distributed random values to real-valued decision variables is the standard mutation method for evolution strategies. Mutation strength can be controlled by adapting σ.

Selection

This operator determines which individuals from a current population are allowed to reproduce and form the next generation in an evolutionary algorithm. To induce competition, the selection is usually determined by fitness values of current solutions. This can be done deterministically, allowing only the best individuals to form the new generation as in Evolution Strategies. However, in multimodal or otherwise rugged objective functions, a deterministic selection procedure might lead to premature convergence on a local optimum. Therefore, probabilistic selection operators might reject a superior individual over another solution to extend the search out of such local optimal regions. The most common and simplest methods are roulette wheel and tournament selection.

1.4.5 Exploration vs. Exploitation

Optimization Algorithms in complex fitness landscapes need to find a balance between exploiting a good, local area and exploring the rest of the objective space in order to find even better solutions. In Evolutionary Computation, every operator can influence this behavior. Usually, mutation is used for exploration purposes while crossover improves individuals within a set of local solutions. As stated in [55], "Mutation serves to create random diversity in the population, while crossover serves as an accelerator that promotes emergent behavior from components". One has to note however, that both operators are strongly dependent on their implementation, used representation and effective

utilization. Therefore, it is also possible, for example, to examine local optima using only slight mutations of individuals represented in Gray encoding.

For selection operators, the feature describing explorative or exploitative behavior is known as selective pressure or selection intensity. Selective pressure[56, 57] examines the takeover time, meaning the number of repeated applications of the selection operator required, until the best solution of an initial population would occupy every slot in this population. A short takeover time equals a high selective pressure and would lead to faster convergence. Vice versa, a long takeover time corresponds to an explorative behavior. Another idea is to measure the average fitness of a population before and after selection as done in [58]. This fitness differential is then divided by the mean variance of the population fitness to obtain a dimension-less selection intensity measure.

1.4.6 Multiobjective Optimization

In a lot of real-world optimization problems, conflicting goals need to be satisfied. Such goals could be, for example, quality vs. cost for a product or transmission reliability vs. speed for a communication network. Countless other examples could be found but they all share the same attribute of a negative correlation between objective functions. This means, that increasing the quality of one objective leads to a degradation of one or more objectives. Given these characteristics, we can conclude that multiobjective optimization problems do not contain a single ideal solution. We can however provide a set of possible good solutions, from which a decision maker can pick the most suitable one.

A very good and in-depth introduction to this topic can be found in [59] while the following section only introduces the basic notations which are necessary for this thesis.

Pareto Dominance

The goal of multiobjective optimization is to find a set of solutions that are superior with respect to the underlying objective functions. This attribute, better known as Pareto optimality, implies that each solution in the optimized set outperforms or dominates all other solutions in a unique way. Mathematically, the definition of Pareto dominance is as follows:

Given an optimization problem with k minimization objectives:

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T \tag{1.8}$$

a solution $\mathbf{x}_a$ is said to dominate another solution $\mathbf{x}_b$ when $\mathbf{x}_a$ is at least equally good as $\mathbf{x}_b$ in all objectives and better in at least one objective:

$$\begin{aligned} &\forall i \in \{1, \ldots, k\}: \quad f_i(\mathbf{x}_a) \leq f_i(\mathbf{x}_b) \quad \text{and} \\ &\exists i \in \{1, \ldots, k\}: \quad f_i(\mathbf{x}_a) < f_i(\mathbf{x}_b) \\ &\text{denoted}^4\text{as}: \qquad \mathbf{x}_a \prec \mathbf{x}_b \end{aligned} \tag{1.9}$$

Example

We assume a 2-objective optimization problem with five possible solutions $\{\mathbf{x}_a, \ldots, \mathbf{x}_e\}$ and their corresponding fitness values $\{\mathbf{a}, \ldots, \mathbf{e}\} = \mathbf{f}(\mathbf{x}_a, \ldots, \mathbf{x}_e)$ given in Table 1.2. Plotting those solutions leads to the 2-

	a	b	c	d	e
f_1	2	2	1	4	5
f_2	2	3	4	3	1

Table 1.2: Resulting fitness values for the example multiobjective problem

dimensional representation in objective space as shown in Figure 1.6.

[4] The symbols $\prec\prec$ and $\preceq$ are also common to indicate strong and weak Pareto dominance, e.g. used in [60]

Using the definition of Pareto dominance, it can be seen that solutions b and d are clearly inferior to solution a. Further, solutions c and e are not better or worse than a. We therefore say, that solutions a, c and e are Pareto optimal, meaning that there is no known solution in the feasible decision space which dominates them. The union of all nondominated solutions in decision space is then called the Pareto optimal set P^* and the representation in objective space is known as the Pareto front PF^*. In our example, the Pareto set consists of the decision variables $\mathbf{x}_a, \mathbf{x}_c$ and $\mathbf{x}_e$ with the corresponding Pareto front a, c and e in objective space. The grayed area in Figure 1.6 indicates the part of objective space dominated by our three Pareto optimal solutions.

$$\begin{aligned} P^* &= \{\mathbf{x} \in A | \neg \exists \mathbf{x}' \in A \text{ with } \mathbf{f}(\mathbf{x}') \prec \mathbf{f}(\mathbf{x})\} \\ PF^* &= \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in P^*\} \end{aligned} \tag{1.10}$$

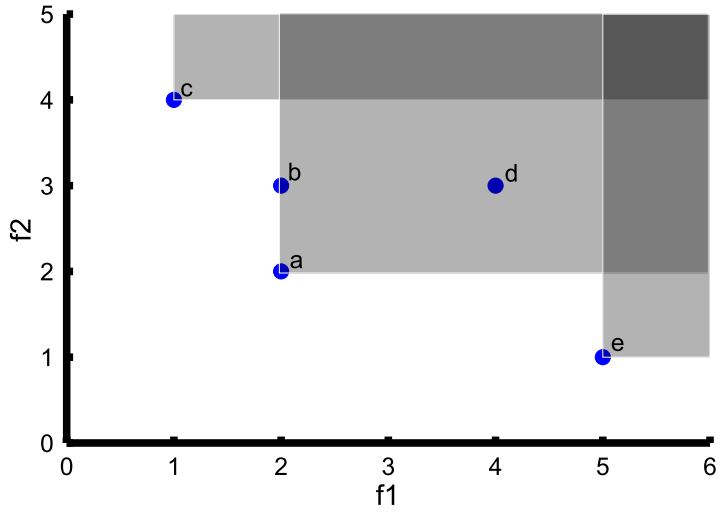


Figure 1.6: Plot of Pareto optimal solutions and dominated objective space

Performance Metrics

The goal of multiobjective optimization algorithms is to find a set of solutions PF^{known}, which approximates the - usually unknown - Pareto

optimal front PF^{true} of the problem. In comparison to single objective optimization, the performance of multiobjective optimization algorithms can be measured according to several characteristics:

- Efficiency in terms of computational effort required by the algorithm for each generation. Complex fitness assignment and ranking techniques can significantly slow an algorithm down, especially for higher-dimensional optimization problems
- Efficiency in terms of required function evaluations
- Scalability of the algorithm onto larger decision spaces and objective dimensions
- Convergence of the resulting population towards PF^{true}
- Divergence as a measure of how well the resulting populations are distributed among the front and how well the corner areas of PF^{true} are explored

Several performance metrics have been proposed to measure these characteristics and are used to compare different algorithms. Some of them require knowledge of PF^{true}, which is generally only known for benchmark problems. If performance metrics for real-world applications are required, one method is to do several independent runs of an algorithm and use the resulting global Pareto front of all runs as a substitute. Another method is to search the whole decision space although this becomes infeasible for large optimization problems.

In this work, I want to point out three metrics for multiobjective optimization algorithms: Hypervolume [61], (Inverse) Generational Distance [62] and ϵ-Indicator [61].

The Hypervolume [61] is probably the best-known performance indicator for multiobjective optimization. It represents the volume of objective space dominated by a Pareto front with respect to some arbitrary reference point. Choosing this reference point is a critical decision when implementing the Hypervolume indicator [63]. Methods of choice are the worst objective point of a solution set, the nadir point [64] or the extreme objective values of PF^{true}. The disadvantage of this indicator is a high computational complexity when calculating the hypervolume for many dimensions. However, there are efficient Monte-Carlo based

sampling methods to estimate the Hypervolume for many-objective optimization [65].

Let us assume a k-dimensional objective space $\mathbf{Z} \in \mathbb{R}_+^k$ and a reference point $\mathbf{r} \in \mathbb{R}_+^k$ which is dominated by all solutions of PF^{known}. In an adaptation from [66], the hypervolume indicator for a Pareto front $\mathbf{A}$ is defined as an attainment function $\alpha_{\mathbf{A}}(\mathbf{z})$ and an integral over the objective space bounded by $\mathbf{r}$:

$$\alpha_{\mathbf{A}}(\mathbf{z}) := \begin{cases} 1 & \text{if } \exists \mathbf{a} \in \mathbf{A} : \mathbf{a} \prec \mathbf{z} \\ 0 & \text{else} \end{cases} \tag{1.11}$$

$$I_H(\mathbf{A}, \mathbf{r}) := \int_{0,\ldots,0}^{r_1,\ldots,r_k} \alpha_{\mathbf{A}}(\mathbf{z})\, \mathrm{d}\mathbf{z}$$

An example plot of the Hypervolume indicator for 2-dimensional objective spaces can be seen in Figure 1.7, where the Pareto front $\mathbf{A}$ consists of 4 objective vectors and the reference point is chosen at $[5.5,\ 4.5]$. The gray area represents the Hypervolume between each point in $\mathbf{A}$ and the reference point.

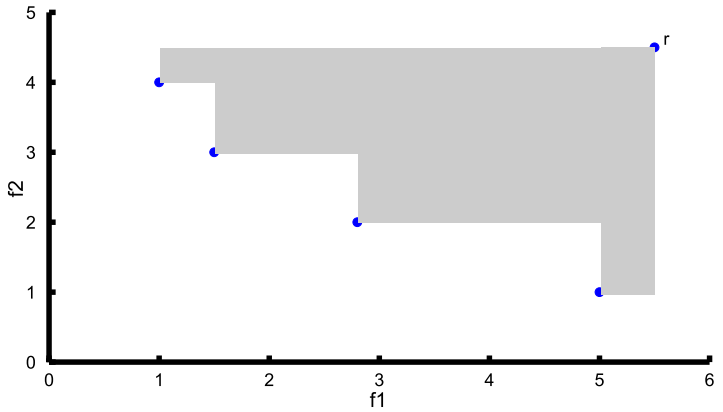


Figure 1.7: Example hypervolume for 2-dimensional objective space

Another metric often used is the Generational Distance indicator [62]. This indicator measures the average distance from a Pareto front $\mathbf{A}$ to

PF^{true} in objective space. The definition is adapted from [62]:

$$GD = \frac{1}{|\mathbf{A}|} \times \sqrt{\sum_{i=1}^{|\mathbf{A}|} d_i^2} \tag{1.12}$$

$|\mathbf{A}|$ represents the number of objective vectors in the known Pareto front $\mathbf{A}$ and d_i is the euclidean distance from point $\mathbf{a}_i \in \mathbf{A}$ to the nearest point in PF^{true}. A GD of zero means that each point in $\mathbf{A}$ lies exactly on a point in PF^{true}. While this metric reflects the convergence behavior of an evolutionary algorithm very well, one obvious drawback is that it provides no information about the distribution of $\mathbf{A}$ on PF^{true}. As an example, the Pareto front in Figure 1.8 would result in a Generational Distance value of zero although large portions of PF^{true} were left unexplored. To overcome this drawback, the metric was extended to

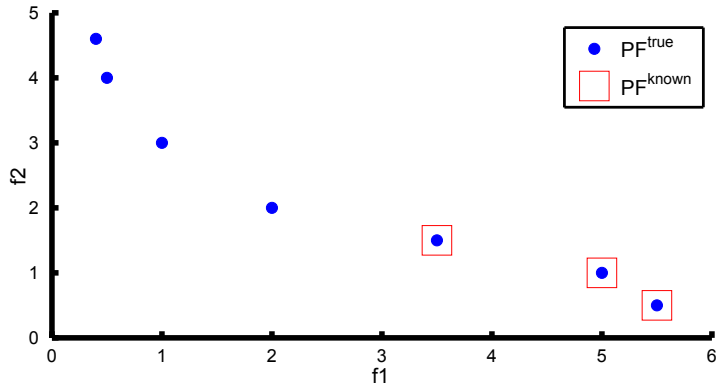


Figure 1.8: Example for poor divergence of PF^{known}

the Inverse Generational Distance (IGD) [67], measuring the distance between each point in PF^{true} and the nearest solution in PF^{known}. Hence, $n = |PF^{true}|$ and d_i is the euclidean distance between a point $p_i \; in PF^{true}$ and the nearest point in PF^{known}.

$$IGD = \frac{1}{n} \times \sqrt{\sum_{i=1}^{n} d_i^2} \tag{1.13}$$

The third metric I want to discuss is the additive ϵ-Indicator [61]. It defines a distance in objective space that lies between two approximation sets or one approximation set and PF^{true}. The definition of the ϵ-indicator relies on the notation of ϵ-dominance [68] that goes as follows:

Consider a k-dimensional, strictly positive objective space $\mathbf{Z} \in \mathbb{R}_+^k$. A point $z_1 \in \mathbb{R}_+^k$ is said to ϵ-dominate another point z_2 *iff* each of the objective values in z_1 dominate the corresponding objective values in z_2 plus $\epsilon \geq 0$. In case of a minimization problem we denote[5]:

$$z_1 \prec_\epsilon z_2 := \; z_1^i \prec (z_2^i + \epsilon) \;,\; \forall i \in \{1, \ldots, k\} \tag{1.14}$$

The ϵ-Indicator $I_\epsilon(\mathbf{A}, \mathbf{B})$ is then the smallest ϵ, that has to be added to a Pareto front $\mathbf{B}$, in order for $\mathbf{A}$ to dominate all solutions in $\mathbf{B}$. The smaller the value of the ϵ-indicator, the closer the distance between the two approximation sets.

$$I_\epsilon(\mathbf{A}, \mathbf{B}) := \min_{\epsilon \in \mathbb{R}_+} \{\forall \mathbf{b} \in \mathbf{B} \; \exists \mathbf{a} \in \mathbf{A} : \mathbf{a} \prec_\epsilon \mathbf{b}\} \tag{1.15}$$

In this work, we only use the additive ϵ-indicator. However, there also exists a multiplicative implementation where ϵ-dominance is defined as in Equation 1.14 but with $\ldots z_1^i \prec (z_2^i \times \epsilon) \ldots$.

Algorithms

Multiobjective optimization has been an active field of research over the last two decades. Since the first algorithms were proposed in the mid-1980s [69], a vast amount of conferences and journals have emerged. Moreover, countless multiobjective algorithms have been published[6] and tested on benchmark problems.

[5] The definition in [61] uses other relation operators for pareto dominance. However, this work uses a notation consistent with [60]

[6] See a distribution of publications per year at `http://delta.cs.cinvestav.mx/~ccoello/EMOO/EMOOstatistics.html`

As stated before, the main goal of multiobjective algorithms is to find a good approximation PF^{known} of the true Pareto front PF^{true} for a multiobjective problem. Proposed algorithms mainly differ in their usage of variation operators, their representation technique and, most importantly, in their selection strategy for reproduction. Most algorithms use dominance-based selection, ranking individuals which dominate most other individuals in objective space. These techniques are most often accompanied by some kind of density estimation such as niching, to spread the resulting points evenly on the Pareto front.

One famous representative is the 'Nondominated Sorting Genetic Algorithm II' (NSGA-II) [70]. For each individual in a current population, the algorithm defines a rank based on the the level of nonN-domination. Niching is implemented by adding a crowding distance for each individual. Parent solutions for the next generation are then selected by domination rank and crowding distance to ensure convergence toward PF^{true} while exploring the fitness landscape. According to [60], this is currently the most used MOEA in comparisons.

Another very prominent dominance-based MOEA is the 'Strength Pareto Evolutionary Algorithm 2' (SPEA2) [71]. In contrast to NSGA-II, it uses an external archive to store nondominated individuals during search. The fitness assignment for reproduction considers the number of dominated solutions as well as the number of solutions that dominate any individual. It achieves diversity using an advanced density estimation technique and archive truncation.

While those dominance-based algorithms perform very well for 2- and 3-objective problems, increasing the number of objectives leads to significant deterioration [72, 73]. The main reason for this is that the number of nondominated solutions, even in random samples, increases with the number of dimensions. Therefore, selection pressure towards PF^{true} decreases. In scientific literature, a problem is considered 'many-objective' if it features 4 to 20 objectives [74]. Several techniques have been proposed to improve the scalability of MOEAs onto many-objective problems. One proposal is to use indicator based evolutionary algorithms (IBEA) [75] and calculate the contribution of each solution to a quality indicator such as Hypervolume for example [76]. Individuals, that

contribute a larger volume are preferred in the selection for the next generation.

Another approach that performs exceptionally well for many-objective optimization is the multiobjective evolutionary algorithm based on decomposition, MOEA/D [77]. It transforms a multiobjective problem into a set of single objective sub-problems using weighted aggregation of objective functions. Each sub-problem is handled as an individual and can share information with neighboring sub-problems. The weight vectors can be set a-priori or adjusted during the optimization using an adaptive strategy [78].

1.4.7 Problem Complexity

The computational complexity of a decision problem is defined by complexity classes. This classification depends on how fast a problem can be solved on a Turing machine as a function of the input size n. Decision problems that can be solved using a deterministic algorithm within an polynomial time $O(n^k)$, are considered as tractable. If the run time is limited by a exponential function $O(k^n)$, it is considered as intractable and can only be solved for small instances of n in a reasonable computation time. An example for such run times can be seen in Table 1.3.

	polynomial, tractable			exponential, intractable		
n	$O(n)$	$O(n\ log(n^2))$	$O(n^3)$	$O(2^n)$	$O(n^n)$	$O(n!)$
10	10	20	1000	1024	3.6×10^6	10^{10}
120	120	499	1.4×10^6	1.3×10^{36}	$\sim 10^{198}$	$\sim 10^{249}$
150	150	653	3.4×10^6	1.4×10^{45}	$\sim 10^{262}$	??
1000	1000	6×10^3	10^9	$\sim 10^{301}$	??	??
5000	5000	3.7×10^4	1.3×10^{11}	??	??	??

Table 1.3: Example run times for polynomial and exponential complexity

Complexity classes are only specified for decision problems having a result $\in \{0,1\}$ or {Yes/No}. It is however easy to model an optimization problem as a decision problem by asking whether or not a possible solution s is the best solution for a given problem.

Class P

A decision problem belongs to the class P if and only if there exists an algorithm for a deterministic Turing machine that can solve the problem within a polynomial run time. The class name P denotes "'deterministic polynomial-time"' and a deterministic touring machine is identified by always producing the same output given a particular input sequence.

Class NP

The "'nondeterministic, polynomial-time"' class consists of decision problems which can be solved on a nondeterministic touring machine in polynomial time. Such a machine is only a theoretical construct and can be thought of as an all knowing oracle that guides the algorithm towards the correct solution. The verification of a solution can however be computed on a deterministic Turing machine in polynomial time.

Classes NP-complete and NP-hard

Decision problems can be transformed into other decision problems of equal or higher complexity. For example, the traveling salesman problem (section 1.4.8) can be stated as a quadratic assignment problem by using two assumptions:

1. Cities are represented by locations.
2. The flow between two consecutive facilities f_i and f_{i+1} is one and zero to all other facilities.

This transformation is called polynomial reduction. A problem C is said to be NP-complete (NPC) if it is in NP and reducible to any other problem $L \in NP$. Since NP only contains decision problems, a optimization problem cannot be NP-complete.

A problem H is said to be NP-hard if at least one problem exists in NP that can be reduced to H. Finding a polynomial algorithm to solve H would also imply that each problem in $L \in NP$ that can be be reduced to H is also solvable by this algorithm in polynomial time.

A Millennium Problem

It is easy to prove that each problem in P is also in class NP such that $P \subseteq NP$. However, it is still unknown whether P is only a subset of NP or $P = NP$. This is one of the seven unsolved millennium problems stated, for example, in [79]. Consequences of an answer to this question would be the following:

- If $P = NP$ then problems in NP would also be solvable in polynomial time by a deterministic Turing machine.
- As many applications e.g. in cryptography rely on the high complexity of mathematical problems, $P = NP$ would mean that many modern encryption methods are at least in theory breakable within polynomial complexity.
- However, $P = NP$ does not state anything about the computational complexity of NP-hard problems
- If $P \neq NP$, NP-hard problems would not be solvable in polynomial time by a deterministic Turing machine.

1.4.8 Standard Combinatorial Problems

This section introduces some well-known combinatorial optimization problems. Due to their usually discrete encoding, high complexity and rugged fitness landscapes, they represent a classic application for evolutionary algorithms.

Traveling Salesman Problem

Due to the simple and intuitive description, the traveling salesman problem (TSP) is maybe the best known optimization problem. It is defined by a set of locations (cities) and the distance between each pair of them. The problem is to find the shortest possible path between each location. The path optimization problem (Find the shortest route) is NP-hard while the corresponding decision problem (Is a given route the shortest one?) is NP-complete.

Several variants of the TSP exist with problem-specific constraints:

The standard TSP represents cities as coordinates in a 2- or 3-dimensional space and assumes euclidean distances between them. The most important implication of this model is, that the shortest route cannot visit a city twice due to the triangular inequality.

The symmetric TSP models distances as undirected cost functions. In contrast to that, an asymmetric TSP considers different costs for the route i and j compared to the route j to i. This can be used to model one-way streets and effectively doubles the search space.

Several agents are cooperating in the multiple traveling salesman problem (mTSP) [80], where each city has to be visited once by one of m individuals.

The time dependent variant (TDTSP) [81] considers different costs for each route depending on the respective position on the tour. A further increase in complexity is represented by the dynamic TSP [82] where even cities can appear and disappear dependent on the current time. These are one of the hardest to solve generalizations of the TSP.

Knapsack / Bin-Packing Problem

Packing problems model the filling process of containers of given sizes with items of given volume and value. Goals could be to minimize the amount of containers needed or to maximize the packed value for a given number of containers. In the special case of only one container, the

problem is known as a knapsack problem. Applications range from classical packing problems over task scheduling in multiprocessor systems to mapping functions onto blocks in 'Field Programmable Gate Array' (FPGA) design.

Without significant constraints, packing problems are known to be NP-hard. A methodical typology of cutting and packing problems has been proposed in [83] and further refined in [51].

Quadratic Assignment Problem

The quadratic assignment problem (QAP) was first introduced in 1957 [84] and represents a facility location optimization. Given a set of n facilities and n locations, the problem is defined by two matrices A and B of size $n \times n$. The entry $a_{i,j}$ represents the spacial distance between location i and j while $b_{i,j}$ is a measure for the required product flow between the facilities i and j. The goal is to find a mapping of facilities onto locations, usually encoded as bijective permutation $\pi : n \rightarrow n$ in such a way as to minimize the cost function:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} \times b_{\pi(i),\pi(j)} \tag{1.16}$$

The search space is defined by all possible permutations of $\{1...n\}$, which equals $n!$ combinations. The QAP has been proven to be NP-hard in [85] and due to the exponential search space increase, instances with more than 30 locations and facilities are considered as hard to solve, even with modern computer architectures [86].

2 Automotive Network Optimization

This chapter explores the use case of automotive network optimization which led to the development of jNetOpt. Building on the motivation from the previous chapter, I want to analyze general requirements for the successful application of optimization methods in an automotive setting. This will be followed by a formal problem definition based on well-known combinatorial problems. Given those environmental aspects, a top-down description and corresponding rationale of the automotive network optimization tool jNetOpt will be presented. Finally, I will compare this approach to other related optimization techniques found in relevant literature.

2.1 Requirements Analysis

Before designing jNetOpt, several requirements were defined to assure the acceptance of the optimization process in an automotive design environment. These requirements can be subdivided into three topics regarding process integration, versatility and robustness of the optimization tool.

2.1.1 Process Integration

Each automotive manufacturer and supplier company follows a preferred development process and supporting software toolchain. Network design and optimization represents one step within this global electronic

architecture process. Therefore, jNetOpt has to be seen as one component of this toolchain. Consequently, it is imperative to provide generic interfaces that allow a seamless integration of jNetOpt into an established development platform (Figure 2.1).

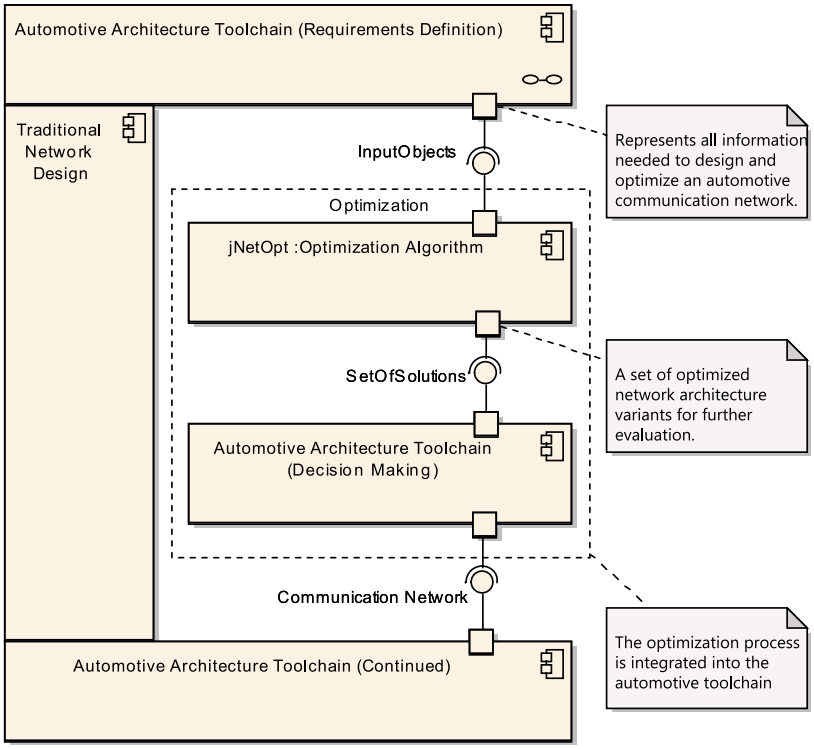


Figure 2.1: Integration of network optimization in E/E architecture development process

For this thesis, the development tool PREEVision [87] was chosen as a reference platform. It is a piece of software used for the model based development of automotive E/E architectures. Further, it supports the integration of custom modules due to the open-source Eclipse [88] platform and Java as the common programming language.

Integrating our optimization process into this environment requires defined interfaces. The artifact `InputObjects` contains all the information necessary for the design and optimization of an in-vehicle network. We define those objects as follows:

- A list of AUTOSAR-based atomic software components which have to be deployed onto hardware nodes within the car
- The corresponding communication requirements between those software components
- A set of available hardware platforms for code-execution
- A list of usable bus systems

Based on these objects, jNetOpt can explore and optimize application mapping and network topology variants. We expect at least one but usually several proposed architectures as a result of this exploration. As some quality preferences cannot be modeled with reasonable effort, it is the responsibility of the E/E design engineers to further evaluate those solutions. A final network topology can then be chosen from the suggested variants and the standard E/E architecture process continues with the optimized network.

2.1.2 Versatility

Each OEM has different preferences when it comes to the level of detail in E/E modeling. In some cases, this is also true for different product lines within the OEM's portfolio. A highly detailed model requires a lot of effort but enables a clean top-down development. On the other hand, modeling just the essential aspects and developing everything else in a bottom-up approach might lead to a faster time to market. A balance has to be found for each car model that depends on, among other things, target price and production volume. Therefore, the optimization framework needs to be adaptable towards different levels of detailism.

In the present form, this adaption is realized by the definition of optimization goals. The better a project is modeled in the development process, the more information can be used to state quality functions. This reflects the detailism of the underlying E/E model. Therefore, it is also necessary to define optimization goals based on the current project and the optimization framework needs to have a clean separation between objective evaluation and optimization.

2.1.3 Robustness

The optimization framework is intended to be used by automotive engineers. Detailed knowledge of optimization processes based on evolutionary computation is not expected. Therefore, all parameters related to the optimization behavior need to be encapsulated. Self-adaption of optimization parameters within reasonable bounds is required to ensure a robust optimization for varying problem instances. As the optimization objectives are also prone to adaptation, the framework has to be able to handle problem instances of variable dimensionality.

2.2 Problem Definition

In this section I will propose a general definition of the network architecture problem based on combinatorial problems. This formalization will point out potential synergy effects, where algebraic techniques and structures can be applied to automotive design space exploration. Along with several other authors, I also propose to break the process down into a mapping and topology optimization phase.

2.2.1 Mapping Optimization

The goal of this phase is to find a feasible mapping for all functional components on a set of distributed hardware units. In the first place, we consider a set of N ECUs and a given distance matrix D of the dimension $[N \times N]^{\mathbb{R}_+}$ where d_{ij} corresponds to the spacial distance between ECU i and j. Further, we consider a set of C software components and a bandwidth requirement matrix $B = [C \times C]^{\mathbb{R}_+}$ where b_{ij} refers to an abstract communication requirement between software components i and j. Note that entries in the main diagonal of both matrices are zero and they are symmetrical because $d_{ij} = d_{ji} \, \forall (i,j) \in N$ and $b_{ij} = b_{ji} \, \forall (i,j) \in C$. In other words, the model is simplified by stating that the direction of communication is not relevant for the network architecture. Finally, we will model installation costs when mapping a software

component onto an ECU in a matrix $E = [C, N]^{\mathbb{R}+}$. Here, it is also possible to define mapping constraints by setting installation costs to $e_{ij} = \infty$ for infeasible mappings.

Our goal is to find a mapping function $m : C \rightarrow N$. This function is not injective as several software components can be mapped onto the same hardware node. It is also not surjective as some nodes might not be utilized at all.

In order to minimize the communication effort within the car, we define a quadratic assignment problem (QAP) [89]:

$$\min_{m:C\rightarrow N} \sum_{i=1}^{C}(\sum_{j=i}^{C} b_{ij} \times d_{m(i)m(j)}) + e_{i\,m(i)} \tag{2.1}$$

Let us further consider a vector for required computational resources $R^R = [C \times 1]^{\mathbb{R}+}$ for each software component and two vectors for provided computational resources $R^P = [N \times 1]^{\mathbb{R}+}$ and monetary prices $P^{ECU} = [N \times 1]^{\mathbb{R}+}$ for each ECU. According to the typology in [51], we define a "Multiple Bin-Size Bin Packing Problem" where the bins are provided resources on ECUs and the items to be packed are required computational resources of software components. In order to minimize the set of ECUs used $A \subseteq N$ and therefore maximize the utilization, we define the problem as follows:

$$\begin{aligned} &\min_{m:C\rightarrow N} \sum_{i\in A} p_i^{ECU} \\ &\text{where} \quad A = \{i \in N \wedge (\exists j \in C \mid m(j) = i)\} \\ &\text{subject to} \quad r_i^P \geq \sum_{j\in C|m(j)=i} r_j^R \;,\; \forall i \in N \end{aligned} \tag{2.2}$$

2.2.2 Topology Optimization

After obtaining feasible mapping solutions, we redefine N as the set of all currently used ECUs and D as the distance matrix for every ECU in

N. Further, B shall represent the communication requirements between all software components mapped onto those ECUs. Unused ECUs are omitted from further optimization efforts. The topology optimization can now be defined as a bi-objective partitioning problem given the graph $G = (N, D)$:

Find a topology $T = [N \times k]^{\mathbb{B}}$ that partitions G into up to k clusters or sub-networks $G' = (N', D')$ where $t_{ij} = 1$ if and only if ECU i is part of cluster j. Only ECUs capable of acting as gateways $N_{GW} \subseteq N$ can be connected to two or more clusters.

The first objective is to minimize the cumulative cable length of each cluster by calculating the minimum spanning trees:

$$\begin{aligned} &\min_T \sum_{i=1}^{k} MST(G'_i) \\ &\text{with } G'_i = (N'_i, D'_i) \\ &\text{where } N'_i = \{j \in N \mid t_{ji} = 1\} \\ &\text{subject to } \sum_{i=1}^{k} t_{ji} \begin{cases} \geq 1 & \forall j \in N_{GW} \\ = 1 & \forall j \in \{N \setminus N_{GW}\} \end{cases} \end{aligned} \tag{2.3}$$

This equation forces the topology $T : G \to \{G'_1 \dots G'_k\}$ to consist of local sub-networks G' where N'_i equals the set of all nodes connected to sub-network i. The constraint ensures that all ECUs are connected to at least one network.

Secondly, we want to minimize the cross-cluster communication which equals all signals which have to be transmitted through a gateway. In order to do this, we first define all ECUs directly connected to an arbitrary ECU $i \in N$ as:

$$\begin{aligned} N^c(i) = \{a \in N \mid &(t_{ib} = 1) \wedge \\ &(t_{ab} = 1) \wedge \\ &(1 \leq b \leq k)\ \} \end{aligned} \tag{2.4}$$

Note that $N^c(i)$ can span over multiple sub-networks if node i acts as a gateway. The minimization goal of cross-cluster communication can now be written as:

$$\begin{aligned} \min_{T} \sum_{i=1}^{N} \sum_{j=i}^{N} b_{ij} \qquad & |N^c(i) \neq N^c(j) \\ \text{subject to } N^c(i) \cap N^c(j) \neq 0 \quad & \forall (i,j) \in N \mid b_{ij} \neq 0 \end{aligned} \tag{2.5}$$

The constraint in Equation 2.5 guarantees that an overlap between two clusters exists if a communication requirement between connected ECUs also exists. This assures that both clusters are connected by a gateway and each communication requirement can be fulfilled.

Both objectives would lead to the result of one global network. When looking at Equation 2.3, the minimum spanning tree of G is always shorter than the sum of connected sub-graphs G':

$$MST(G) \leq \sum_{i=1}^{k} MST(G'_i) \tag{2.6}$$

The only exception would be if G consists of a sub-network which is (a) locally separate and (b) has no communication requirements from/to the rest of the car. However, such a distinct sub-network is highly unlikely in a modern and interconnected vehicle. In Equation 2.5 it is obvious that a network consisting of only one large cluster does not require any gateways and therefore also no cross-cluster communication.

However, realizing the network topology as one large network would require a very fast and expensive bus system like FlexRay or Ethernet. It is obvious that such a network is overdimensioned for most applications within the car. Consequently, we need to model bus utilization and costs as additional optimization goals to ensure more realistic results.

Let us assume that a specific bus technology (CAN, FlexRay, Ethernet,...) is assigned to each cluster in $T : G \rightarrow \{G'_1 \dots G'_k\}$. Similar to the mapping optimization phase, we define a $[k \times 1]$ vector BW^T, containing the targeted communication bandwidth for each cluster based on the assigned

network technology. Further, connecting an ECU to a bus is modeled by a bus coupler with the associated price in the $[k \times 1]$-dimensional vector P. The according minimization goals for bus utilization and costs of bus systems can now be stated as:

$$\min_{T} \sum_{i=1}^{k} \alpha_i \frac{|BW_i^T - BW_i^R|}{BW_i^T} \Bigg| BW_i^R \neq 0 \tag{2.7}$$

Where BW_i^T equals the targeted bandwidth consumption for the sub-network i and BW_i^R the accumulated communication requirements for the cluster. The arbitrary α_i can be used to prioritize the utilization of a specific bus system. The actual costs of connecting ECUs to bus systems are finally defined by:

$$\min_{T} \|T \times P\| \tag{2.8}$$

2.2.3 Discussion

The proposed general problem formulation contains a quadratic assignment problem, a bin-packing problem and a graph partitioning problem with four goals. All these problems are known to be NP-hard [90]. Equations 2.1 and 2.2 might have dissimilar minima due to a high relocation effort of components and the layout of available computational resources. Further, it is clear that equations 2.3 and 2.5 prefer one high-speed bus system while equations 2.7 and 2.8 are leading towards a fine-grained topology of many low-cost networks. Therefore, we can expect a Pareto-set of solutions for each phase.

2.3 Challenges

The general formulation assumes abstract input parameters which are hard to determine in an ongoing design process. However, the quality of

optimized solutions depends greatly on exact models and predictions. In this chapter, I want to point out some of those parameters and why it is challenging to find appropriate models for them.

2.3.1 Communication Requirements

Current automotive design processes create a database and schedule of all signals and related timing constraints for each bus system. These schedules are partly distributed to suppliers for integrating their components into the global communication table. Therefore, alterations of bus topologies or message schedules require a huge amount of communication between all involved parties, which makes them very cost intensive. Furthermore, CAN, being the de-facto standard bus system in vehicles, is by definition incapable of real-time communication except for the highest priority frame. While estimations of worst-case transmission times can be modeled very exactly for cyclic messages [91, 3], event triggered message bursts are still hard to model and can lead to missed deadlines for signals with lower priority. Therefore, a lot of effort is put into timing verification for current bus networks, which would need to be repeated for topology optimizations. The challenges here are:

- To provide all relevant timing information at an appropriate level of detail at the point of design
- Estimate effective bus utilization for event and time triggered communications
- Find a flexible design process to allow topology changes without inducing extensive costs for timing verification

2.3.2 Computation Resources

Automotive software is usually optimized for a specific processing platform to reduce code size, required CPU time and therefore cost per produced unit [18]. With the possibility of relocating software components onto different ECUs, abstract resource metrics for heterogeneous

platforms need to be developed. While memory requirements for different CPU architectures vary only slightly, timing verification is a very complex task. To efficiently apply design space exploration techniques, an estimation of the amount of software that can be executed on a specific ECU has to be done. AUTOSAR does not yet provide such abstraction levels and relies on consistent timing analysis and verification [92]. However, a detailed model of task execution is usually not available during the phase of network architecture. Approximating it for each possible variant would be too time consuming to efficiently explore design alternatives using evolutionary algorithms. Therefore, an appropriate level of abstraction needs to be defined for this task, considering single- and also multi-core processors.

2.3.3 Wiring Harness

Very few design optimizations consider construction details and constraints with regards to the wiring harness. However, the resulting wire lengths are greatly dependent on the layout of cable tunnels and available drills in the bulkhead between engine and passenger compartment. In [93], I use the *Manhattan distance* as simplification, determining cable lengths as the sum of their coordinate distances in a 3-dimensional Cartesian coordinate system. The author of [94] goes even further and models wiring as a stub from a component to the nearest common cable ducts which feature a H-form due to production requirements. Additional EMC-related constraints are forthcoming due to high voltage wiring in electric and hybrid vehicles. For exact estimations of resulting cable lengths and weight, these aspects have to be considered during topology optimization.

2.3.4 Reliability

The proposed formalization does not consider communication reliability as an objective as in some previously analyzed publications [95, 96]. We assume that the reliability of automotive networks mainly depends

on the quality of the hardware used (physical layer, connectors, EMC measures,...). Further, functional safety requirements are directly derived from respective standards (e.g. ISO 26262). Both need to satisfy a minimum standard aspired by the OEM, which reflects the quality image of the brand. Therefore, the realization of reliability can be seen as a limiting constraint when choosing hardware components, or as a logical input parameter for system architecture. From our point of view, this is not an objective of mapping and topology optimization and relevant safety-critical deployments can be realized by according mapping constraints.

2.4 Our Approach

After analyzing requirements and the formal optimization environment, I now want to introduce the main contribution of this thesis, the optimization framework jNetOpt. I will start by giving a rationale as to why evolutionary computation was chosen as a means of optimizing automotive networks. This is followed by a more detailed discussion of important features unique to this work.

2.4.1 Rationale

Search Space

As stated in the previous discussion, the formal problem consists of several NP-hard optimization problems. The search space is primarily defined by two decision variables representing the optimization phases:

- The mapping function $m : C \rightarrow N$ which maps software components onto ECUs.
- The topology assignment $T : G \rightarrow \{G'_1 \dots G'_k\}$ which clusters the logical architecture into sub-networks.

Depending on the preferred modeling effort, let us assume there are between 100 and several thousand defined software components and 50 to 100 ECUs in a modern vehicle.

For the mapping optimization phase, we can approximate the lower bound of the search space as a *Permutation with repetition* by mapping 100 software components onto 50 ECUs. According to section 1.4.3, this equals 100^{50} possible permutations without considering mapping constraints.

The topology assignment can be interpreted as k-way graph partitioning problem [97], assigning N ECUs to k clusters. Considering a topology of 5 bus networks and omitting gateway capabilities, the search space can be approximated as k^N or 5^{50} for our example.

The combination of those two phases leads again to an additional increase of the search dimensions. In conclusion, it is safe to say that the search space of the mentioned problems is too large to be efficiently solved by exact optimization methods. This is one reason for the application of stochastic methods.

Constraints

The previous discussion does not include constraining aspects for optimization decisions. While they have the beneficial behavior of reducing the search space, constraints also disrupt homogenous areas and induce non-linearities and sharp edges in the search landscape. This can even lead to several disjunct areas of feasible solutions, which cannot be explored using standard deterministic methods.

However, given the right encoding and constraint handling method, evolutionary algorithms are known to perform well on such constrained problems [98]. Their stochastic nature can find disjunct areas not reachable by deterministic methods. This is another argument for the utilization of evolutionary computation.

Multiobjective and Multimodal Aspects

The formal problem definition consists of a total of 6 minimization goals which result in a 6-dimensional objective space. It has also been shown in the previous section, that most objectives have contradicting goals. Therefore, we can assume a Pareto front of feasible solutions as optimization result. Population-based algorithms are a popular approach to handle such problems with scalable effort.

Further, we can expect several local optima for each dimension in objective space. Again, evolutionary multiobjective optimization has been proven to work well for such problems, especially without a prior knowledge of the Pareto front (e.g. [99]). This is the third reason and concludes our rationale for the use of evolutionary computation.

2.4.2 Problem-Specific Encoding

In contrast to existing approaches, we want to employ sophisticated decision encodings based on the actual problems. This means to choose the encoding in such a way that problem-specific knowledge can be utilized during optimization. The advantages of this encoding can be the avoidance of constraint violations or inducing preferences during stochastic processes. A disadvantage is of course the limited applicability of existing mutation and crossover operators, which rely on standard genotypic representations.

2.4.3 Feasibility Preservation

Our optimization problem is represented by a highly constrained search space. The performance of evolutionary computation can therefore suffer under a high amount of infeasible solutions being generated during the search process.

Typical remedies are the introduction of penalty functions to reflect constraint violations and to avoid discarding infeasible individuals.

Another approach is the introduction of repair algorithms to find feasible alternatives within a small neighborhood in the search space. While such and other methods are proven to perform well, the idea of avoiding infeasible solutions by design is pursued in this work. I have presented and published these ideas in more detail in [100].

2.4.4 Advanced Operators

Due to the custom encoding techniques, we were able to develop guidance mechanisms for mutation operators. This feature lead to improvements in the convergence behavior of our optimization. The impact of guided mutation operators was further studied in [93, 101].

The basic idea is to avoid the degradation of fitness during the mutation of an individual. This is done by splitting the mutation operation into two steps. First, the random mutation is performed as usual. Then, a refining step is conducted, aiming to overlay the stochastic process towards an improvement of a specific fitness quality. This can also be done in multiobjective optimization using one mutation operator for each objective dimension.

Care must be taken not to exaggerate the usage of such guidance functions because the explorative behavior of optimization algorithms can deteriorate. Other studies regarding guided evolutionary operators can be found e.g. in [102, 103, 104].

2.4.5 Enhancements for Multiobjective Optimization

Our framework consists of two phases of optimization. The resulting set of solutions from the mapping optimization phase represents the initial population of the topology counterpart. This structure inspired the generalization of the approach towards many-objective optimization. In [105], I presented a methodology to improve the optimization behavior of well-known MOEAs for a large number of objectives.

This was done by decomposing a many-objective problem into 2- or 3-dimensional sub-problems, where common MOEAs are known to perform well. Optimizing the decomposed problems using a fracture of the overall computation budget allowed the algorithm to explore the search space more efficiently. After this exploration phase, all results were accumulated and represented the initial population of the many-objective optimization algorithm.

2.5 Other Approaches

When analyzing different optimization approaches, it is useful to classify them by their goals. We distinguish between techniques for:

1. mapping functional software on a fixed network of hardware nodes (mapping optimization)
2. finding optimal bus topologies for a fixed set of communication requirements (topology optimization)
3. optimizing both tasks either consecutively or in parallel (mapping and topology optimization)

In the following sections we will analyze selected methods using this form of classification. Concluding, we will provide a short summary of the number of optimization objectives, used heuristics and modeling approaches in tables 2.1, 2.2 and 2.3 respectively.

2.5.1 Mapping Optimization

Due to the advance of AUTOSAR, the process of mapping or deploying functional components onto a distributed network of hardware nodes is an emerging issue for automotive designers. Whilst this task is very well known in computer science, the stringent real-time requirements and various embedded hardware platforms within the car usually result in a very constrained optimization problem.

In [94], the author describes the artifacts to be mapped as "an aggregation of software and hardware components"[94]. This view assumes, that hardware components like sensors or actuators can also be mapped onto a set of possible ECUs. The author models abstract *resources* like RAM/ROM, CPU time, I/O Pins or available space on a printed circuit board, which are provided by an ECU and consumed by mapped components. Further, objective functions for aspects such as weight and cost of the wiring harness, estimated bus load and energy consumption are also defined. The bus load estimation assumes prior knowledge of all transmitted signals including their length, update rate and deadline. It calculates transmission time and prioritization for each signal depending on the bus used and optimal frame utilization. Further, a supplier complexity objective attempts to model the integration costs of components on a specific ECU. The author developed an evolutionary algorithm (EA) and an ant colony algorithm (ACO) and extended the process by including variant optimization.

In comparison to that, the authors of [106] model the problem as bi-objective, with communication overhead and data transmission reliability as quality metrics. Accordingly, they assume more detailed knowledge of network bandwidth and channel reliability as input parameters. Further, their definition of a component is restricted only to software with fixed hardware topology and locations of sensors and actuators. In order to avoid the congregation of all software components onto one ECU, a constraining memory model is defined. Further implemented constraints are the stringent localization of software components to a specific hardware node and the co-localization of 2 software components to different hosts. The rationale for these constraints are safety-related functions which are designed to continue operating even if one ECU fails. Similar to [94], the authors also implemented an ACO and compared the performance to a multi-objective genetic algorithm (MOGA).

Table 2.1: Summary of mapping optimizations

Ref.	Obj.	Heuristic	Models
[94]	5	EA, ACO	Cost, weight, bus load, energy consumption, supplier complexity, ECU resources
[106]	2	MOGA, ACO	Memory, bandwidth, communication reliability

2.5.2 Topology Optimization

It is common practice for new car developments to re-use earlier network topologies from previous car series and adapt to new communication demands when needed. This practice lead to the previously mentioned bus systems dedicated to power train, comfort functions and entertainment, which are well-established but also very inflexible.

The authors of [107] model the topology optimization as a graph partitioning problem. Input parameters like cycle time and size of messages are automatically imported from an automotive architecture tool. Further, this approach models costs for each bus system and participant as well as additional gateway costs and communication overheads. The sum of all these costs is the single objective function in this optimization. The graph partitioning problem is built with ECUs as nodes and communication requirements as edges. First, it is hierarchically clustered according to several "nearness functions"[107]. Each cluster represents a bus system yet the bandwidth utilization of each network might not be optimal. Therefore, the authors define a bin-packing problem to reach better bus utilizations. As the number of bus systems in a car is relatively low, this can be done with an exact dynamic programming approach. The presented work is the only deterministic optimization approach without any stochastic extension in this survey. It does not consider spacial distances between ECUs and resulting wiring costs.

On the contrary, the author of [108] provides a very detailed model of spacial distances based on a cubical grid throughout the car. It is

used to determine wiring costs for an Ethernet-based backbone network, connecting heterogeneous sub-networks. Due to the fixed topology of Ethernet, further objectives are the number of ports and energy consumption of network switches. Optimal utilization of bus systems is not a concern in this work as the focus lies on efficiently interconnecting local networks using the high bandwidth Ethernet backbone. The optimization is done using a genetic algorithm (GA) in combination with local repair strategies.

Table 2.2: Summary of topology optimizations

Ref.	Obj.	Heuristic	Models
[107]	1	HC + DP	Bus-, participant- and gateway-costs, nearness functions
[108]	3	GA, local repair	3-D clustering, wiring costs, port costs, energy consumption

2.5.3 Mapping and Topology Optimization

The next logical step is to combine both optimization tasks in one framework.

The optimization approach in [109] models components as functional software or sensor-/actuator hardware. For each ECU, a set of possible mounting spaces is assumed while resources such as memory or CPU speed are chosen according to the mapped components. Therefore, ECU resources are not considered as constraints to the optimization. The representation of a possible network is hierarchically structured, where the higher level represents the assignment of ECUs to bus systems and the lower level the clustering of components to ECUs. The approach uses an EA to optimize those two levels as well as the placement of ECUs in the car. Assisting deterministic functions are used to select proper bus types, wiring topologies and the placement of gateways. To enhance

the performance of the EA, problem-specific operators are used for the variation of solutions. The objectives of optimization are a cost model of the ECUs/cables used and a metric for ECU complexity, corresponding to the number of components assigned to each ECU. Due to the structure of this approach, the tasks of mapping and topology optimization are done simultaneously.

In contrast to that, the design approach in [110] uses one EA loop for component mapping and a nested EA loop for topology optimization. For both algorithms, the very well known SPEA2 was implemented and assisted by custom repair algorithms. Input parameters are retrieved from a customized FIBEX format, extended by a detailed model of communication signal properties. The author has identified five optimization objectives: cost efficiency, reliability, variant management efficiency, testability and extendability. Each objective is defined by an according metric or model. In this work as well as in [109], gateways are treated as distinct hardware units, able to connect two bus systems but not capable of hosting functional components. Hardware resources such as available memory or CPU time are not considered in this approach. However, a detailed verification of schedulability and response time is performed for each signal and bus system.

Table 2.3: Summary of topology and routing optimizations

Ref.	Obj.	Heuristic	Models
[109]	2	EA	ECU costs, mounting spaces
[110]	5	EA + nested EA	Costs, reliability, variant efficiency, testability, extendability
[111]	1+1	Repeated matching + SA	ECU costs, task installation cost, wiring stubs

Similarly to [109], the authors of [111] also define three problems for the network design procedure: component mapping, ECU positioning and network assignment. The first two tasks are implemented as single objective optimizations and done consecutively. A cost model for ECUs

and mapping components onto a specific node is assumed. Further, computing resources are given for each ECU and handled as constraints during the optimization. For the ECU positioning problem, a set of possible installation locations is modeled for each hardware node. The authors assume a common cable path in the car where the main bus lines reside. To connect ECUs, stub lines or alterations to this common cable path have to be added to the wiring harness. When optimizing ECU placement, the cost of extra wiring can be reduced. Finally, a network assignment task is conducted which verifies that all communication signals can be scheduled and routed over the actual bus topology. This is done by the well-known model used in [91] and assumes prior knowledge of all messages, cycle times and priorities. Surprisingly, the optimization of gateways is not mentioned in this work but modeled as a direct connection between two bus systems without any additional costs. The authors developed a repeated matching method assisted by simulated annealing (SA) to optimize the three problems.

3 jNetOpt

This chapter introduces the implementation of jNetOpt. I will start with an overview of the principle design patterns and anticipated workflows of the software. This is followed by a more detailed description of implemented classes and their usage.

JNetOpt is fully developed in Java under Java Runtime Environment 1.7. I decided to use two software products as the development toolchain:

- Sparx Enterprise Architect [112] for architecture modeling based on the 'Unified Modeling Language' (UML).
- Eclipse [88] as Java editor and environment for execution, debugging and unit tests

When using model driven architecture as a design approach, the developer is confronted with a manifold of different domain perspectives. Due to limited resources, I decided to use only a subset of the available models. Further, I allowed some simplifications while modeling interactions between elements and model transformations.

Despite those necessary simplifications, the model driven architecture allowed a very flexible top-down design throughout the development process. In my opinion, this selective usage of parts of the UML specification was very important to achieving a well-structured final software while keeping design effort within reasonable limits.

3.1 Software Architecture

In the previous chapter, I argued as to why jNetOpt should be easily integrable in an automotive development toolchain. On the other hand,

the software has an experimental character and is also used for scientific studies. Both use cases utilize the same optimization algorithms but in different applications. Therefore, I have chosen to apply the 'Model - View - Controller' (MVC)[113] design pattern as the main guideline for the software architecture.

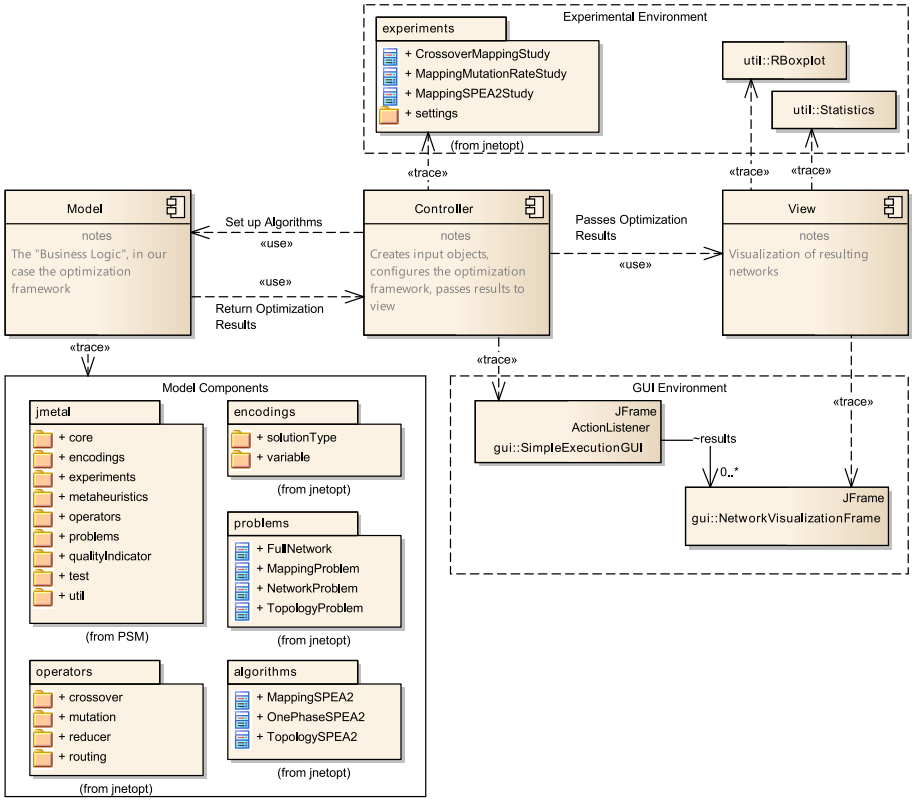


Figure 3.1: The Model-View-Controller design pattern applied in jNetOpt

The focus of this design pattern is a strict encapsulation of the business logic, the 'Model'. Several execution and visualization variants can be developed for specific use cases and interact with the same model. In this work, I developed an experimental environment for scientific studies and a GUI-based environment with a focus on automotive network optimization as seen in Figure 3.1.

The 'Model' component consists of all objects related to optimization. It is based on the Java-based metaheuristic algorithms framework, called

jMetal [114]. Several extensions have been implemented to improve the performance of automotive-related network optimizations.

The optimization can be configured using two different 'Controller' objects. One is based on the `Experiment` class in jMetal and uses generic test networks to evaluate optimization performance under different algorithm settings. The other one focuses on the automotive use case and provides a GUI-based representation as well as interfaces for industrial applications.

The 'View' objects are realized by statistical evaluation of optimization performance or as a graphical representation of the resulting networks.

3.1.1 General Workflows

For each use case, a different optimization workflow is provided. As automotive engineer, input objects can be imported via the `SimpleExecutionGUI` class. Optimization parameters such as number of total evaluations and used operators can be set here. The result of one optimization can then be examined in the `NetworkVisualizationFrame` or passed on to the automotive architecture toolchain.

The second workflow concerns performance evaluations using different algorithms, operators and settings. Here, the focus is laid on statistical analysis of several independent optimization runs.

Figure 3.2 depicts the general optimization workflow for both use cases. It further shows some relevant classes which will be discussed in the following sections.

jNetOpt supports the optimization of software mapping and network topology as discussed in section 2.2. This is realized using two kinds of optimization approaches:

1. A one-phase approach which optimizes either one of the optimization tasks or a combination of both in one iteration.
2. The separation of both tasks into two distinct optimization iterations, here labeled as a two-phase approach.

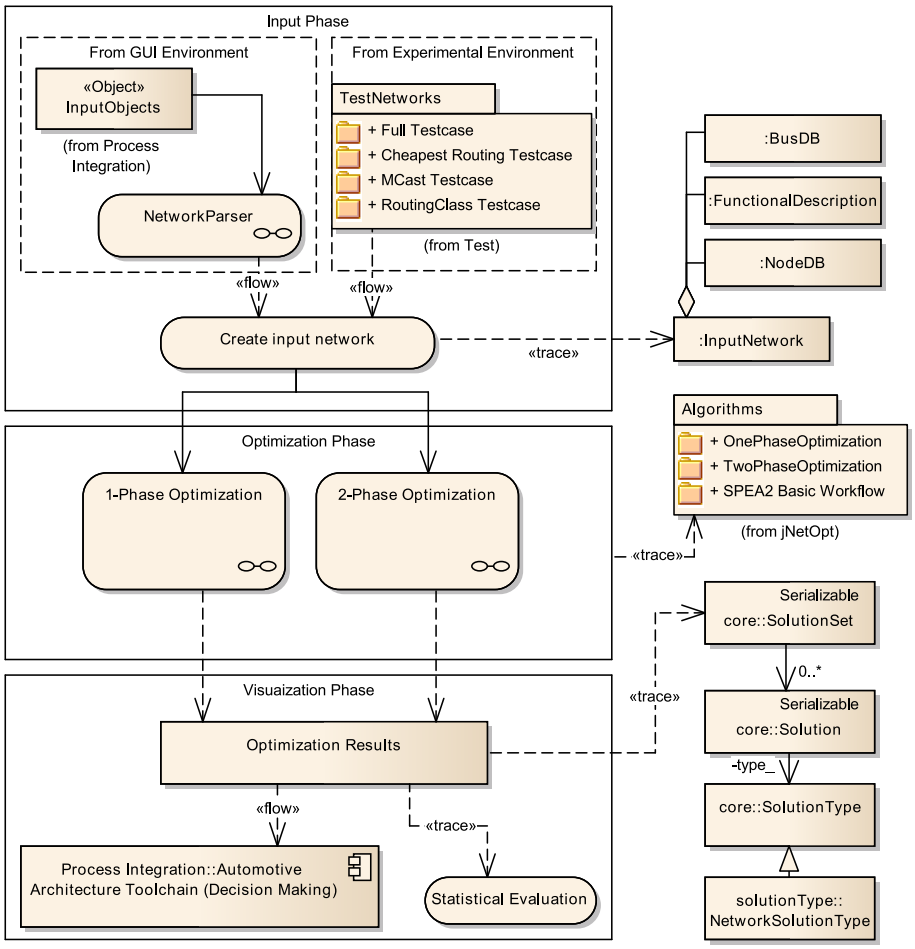


Figure 3.2: The proposed workflows of jNetOpt

Overviews of the one-phase and two-phase optimization approaches that have been referred to are provided in Figure 3.3 and Figure 3.4 respectively. These models also trace the optimization tasks through several MDA-abstraction levels and to the underlying combinatorial problems defined in section 2.2.

The one-phase approach is especially suitable for topology optimization of a network with fixed software mapping. It further allows easy benchmarking of algorithms which is very important for performance studies.

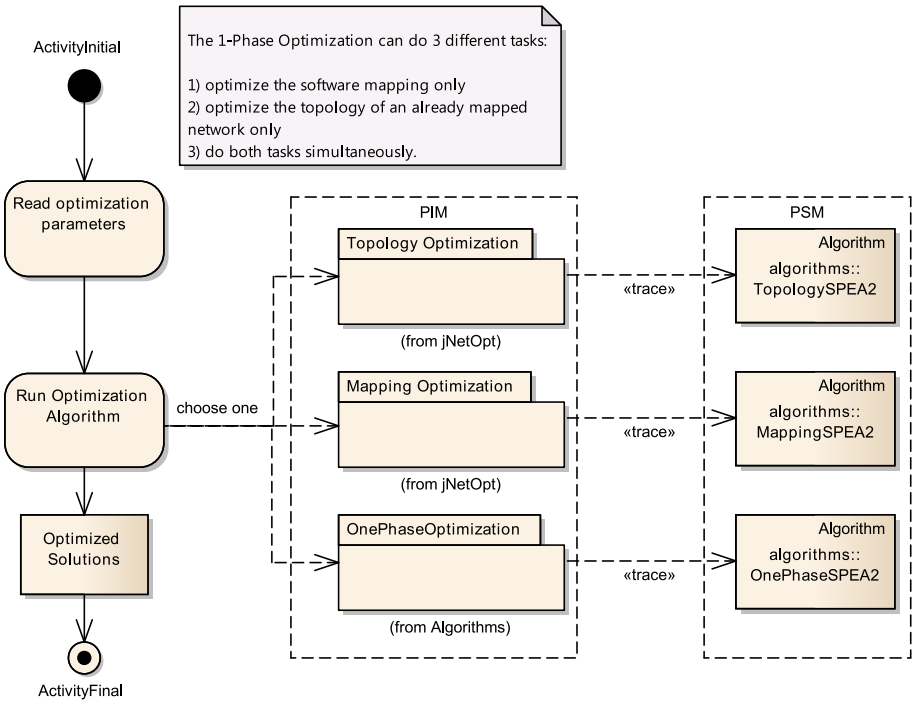


Figure 3.3: Overview of the one-phase optimization approach

However, the performance of the combined approach is naturally inferior to comparable two-phase optimizations.

The two-phase optimization utilizes the same algorithms available to the one-phase counterpart. The difference is that this approach populates the second algorithm with mapping results from the first optimization.

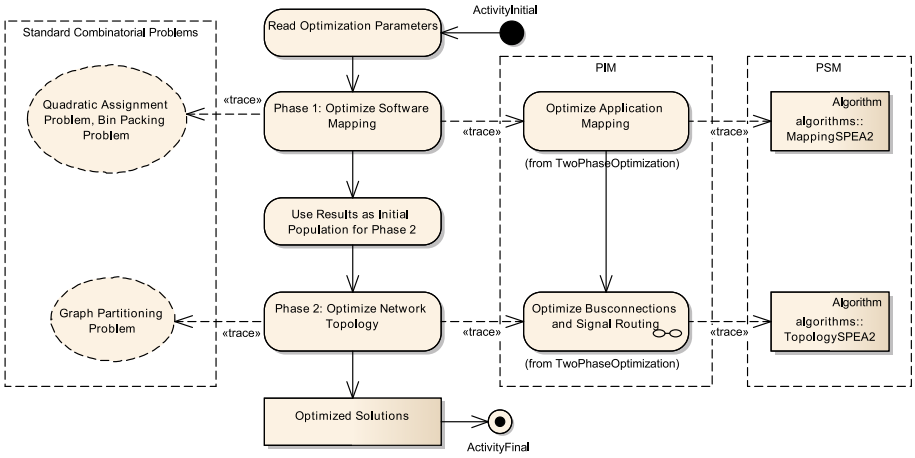


Figure 3.4: Overview of the two-phase optimization approach

3.2 Input Objects

Based on the discussion in subsection 2.1.1, the optimization framework requires input data from the automotive architecture process. This data describes the network to be optimized as well as constraining boundaries. jNetOpt uses hierarchical structures to store this data in several classes:

- The `NodeDB` is a database containing all possible ECUs within the car. Each ECU is represented by an instance of jNetOpt's `Node` class.
- All bus systems available for optimization are stored in the `BusDB` class.
- The logical communication architecture between functions is represented by the `FunctionalDescription` class. It consists of several `AtomicSoftwareComponents` which reflect the AUTOSAR modeling approach. Communication requirements (signals) to other software components are mapped as ports in the class `CommPort`. To support multicast communication, an instance of `CommPort` can also have more than one receiver.

All these elements are summarized in the class `InputNetwork`. This class and all related classes are serializeable using standard Java methods. This is an easy way to store and recall network configurations or provide

an interface to other Java-based tools. A class diagram of all relations as well as some example test network instances is shown in Figure 3.5.

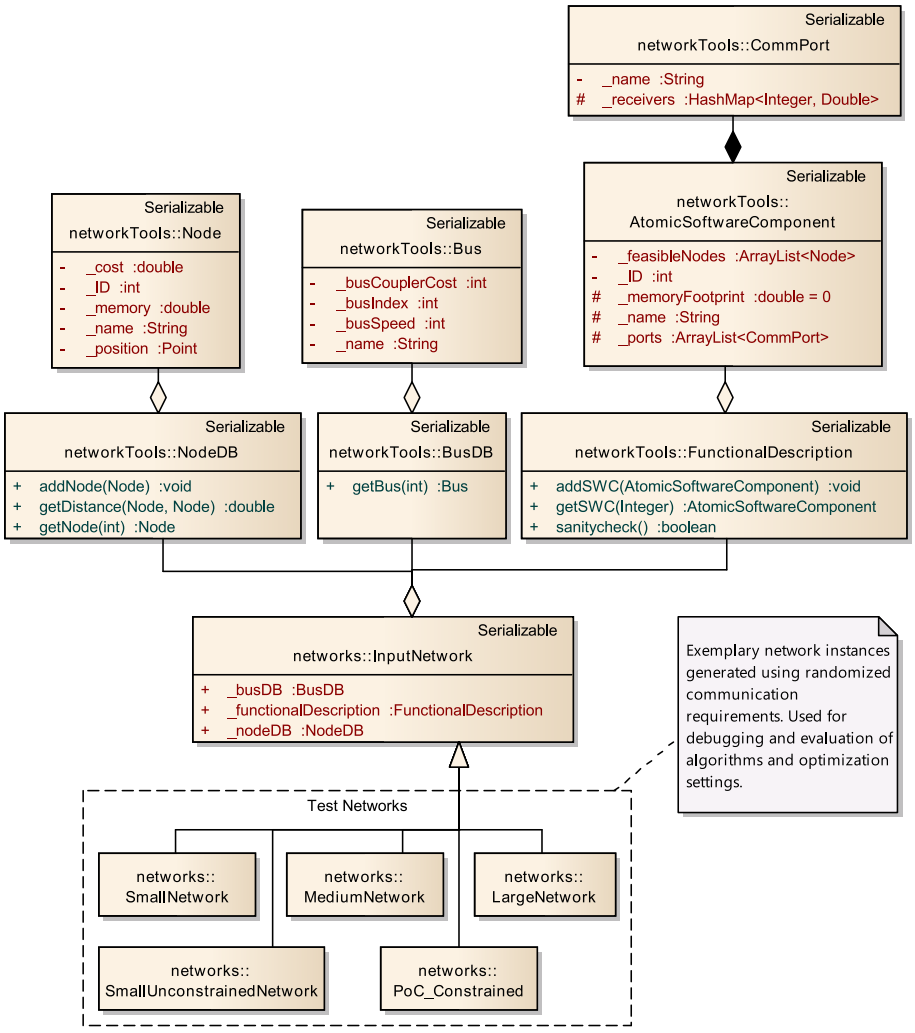


Figure 3.5: Class diagram of `InputNetwork` and example test networks

3.2.1 Proof of Concept

During the development of jNetOpt, a 'Proof of Concept' of the interface between the optimization framework and PREEVision as an automotive

architecture tool was required. This PoC was realized by extracting all communication requirements from an actual automotive project as comma separated values (.csv) file. The layout for each modeled signal was as follows:

```
SendingECUName;NameOfSWC;NameOfSignal;SignalBandwidth;
ReceivingSWC1;ReceivingSWC2;....;ReceivingSWCn
```

jNetOpt provides a file parser to create a functional description out of this data. The PoC was done using a network with fixed mappings of SWCs onto ECUs. However, extending this process to allow the import of a set of feasible sending ECUs for each software component is a straightforward task.

3.3 Encodings

The jNetOpt representation of a complete network consists of four elements.

1. A global instance of `InputNetwork` containing all software components, logical communication requirements and ECUs
2. The mapping of software components onto ECUs represented by the variable `ApplicationMatrix`
3. Connections of ECUs to bus systems stored in the variable `BusMatrix`
4. In case of multiple possible routes for each signal, routing paths are stored in an instance of `CommunicationMatrix`

The instance of `InputNetwork` is considered as a static input parameter. The other three objects are problem specific encodings and unique for each candidate solution. Therefore, they represent the optimization variables and are subject to variation using mutation and crossover operators during optimization.

SWC	1	2	3	4	5	6	7
ECU	2	3	5	4	1	4	1

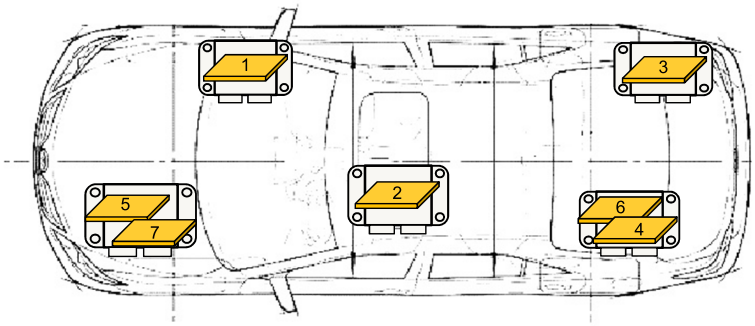


Figure 3.6: An example of mapping of software components onto ECUs

3.3.1 ApplicationMatrix

The class `ApplicationMatrix` stores the mapping of functional and gateway software components onto ECUs. By accessing the `FunctionalDescription` and `NodeDB`, the class can also provide methods to evaluate memory utilization of nodes and functional distances between mapped SWCs. The encoding is implemented as an array with the length of `_numberOfSWCs`. Each entry represents the currently mapped ECU of a software component. An example using seven software components and five ECUs is shown in Figure 3.6.

3.3.2 BusMatrix

The `BusMatrix` encodes connectivity between bus systems and ECUs which equals the topology of a candidate network. This is done in a binary matrix encompassing all available ECUs and bus networks. A logical '1' means that the corresponding ECU is connected to the bus system as demonstrated in Figure 3.7.

ECU	1	2	3	4	5
BUS 1	1	0	1	1	0
BUS 2	0	1	0	0	1
BUS 3	0	0	0	1	1

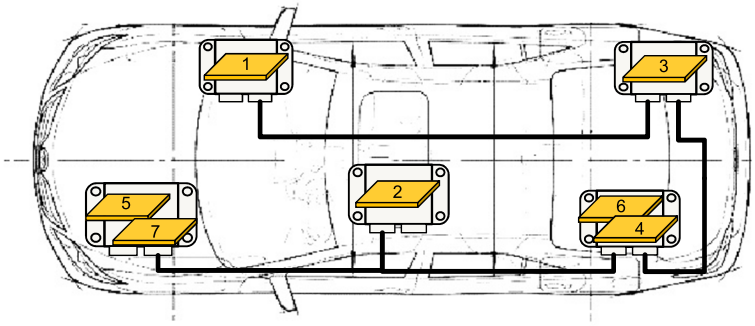


Figure 3.7: An example of connections of ECUs and bus networks

3.3.3 CommunicationMatrix

This variable maps the routing of signals if more than one connection from source to destination is possible. This is the case when two ECUs are connected by more than one bus system or gateway. In this case, the bus utilization of one solution depends on the selected routes. Further, this class monitors the utilization of bus connections for each node. Signal transmissions via gateways are stored as instances of the class `SignalRoute`.

3.4 Operators

The optimization algorithms use several kinds of operators to manipulate candidate solutions. I want to classify three groups:

- The variation operators crossover and mutation for software mapping and topology alterations

- Selection operators to pick candidate solutions for the next generation of the evolutionary algorithm
- Network-specific operators for routing signal paths over the network and cleaning up unused connections

Due to the problem-specific encoding used in this work, custom operators for crossover and mutation had to be developed. Although, the selection operators only use fitness values of evaluated solutions. Therefore, the proposed mechanisms for the respective algorithms could be applied here. The network-specific signal routing is required to finalize and validate a new candidate solution. In order to provide a better overview, all operators are introduced by their respective usage in the optimization process.

3.4.1 Operators for Mapping Optimization

All variations for mapping optimization are performed on the instance of `ApplicationMatrix` as seen in Figure 3.8. This is also the only variable required to evaluate a solution in terms of mapping quality.

jNetOpt provides a mating operator based on the one-point crossover, which can be applied on two parent solutions to create two offspring solutions. Provided there are two feasible mappings as parents, the offspring solutions will also map every software component onto a feasible node. However, the crossover operation can lead to an over-utilization if too many software components are mapped onto the same node.

Apart from the standard random mapping mutation operator, jNetOpt also provides two guided counterparts. The class `FuncMappingMutator` alters software mappings with a bias towards a better functional distance. In a similar way, the `MemoryMappingMutator` focuses on improving the memory utilization of nodes. All mutation operators provide a unified interface and are interchangeable throughout the optimization process.

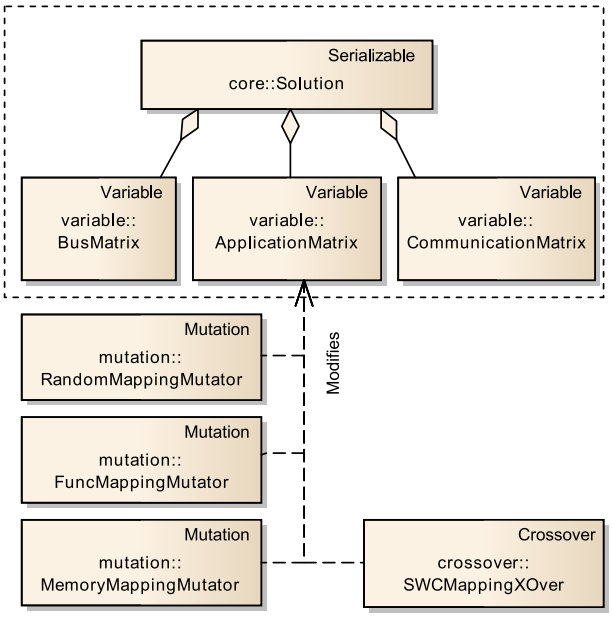


Figure 3.8: Operators relevant to mapping optimization

3.4.2 Operators for Topology Optimization

The topology optimization performs variations on the `BusMatrix` through crossover and mutation. Then, a routing algorithm is applied to ensure that every signal can be transmitted over the candidate network.

The crossover operation is implemented in a two-point fashion using two solutions called parents A and B. Two bus systems, i.e. two 'rows' in the `BusMatrix`, are then chosen randomly. The offspring solution A is created by cloning all variables of parent A and replacing all entries between those two randomly chosen rows in the `BusMatrix` with corresponding information from parent B. Offspring B is created in the same fashion using the mirrored combination of parents.

The resulting offspring solutions then undergo the process of mutation. Similar to the mapping optimization, the user can again choose between random mutation and a guided counterpart. Random Mutation is implemented as an probabilistic bit flip operation on the `BusMatrix`. In the `GuidedBusMutator` class, connections between bus systems and nodes

depend to some degree on the actual bus utilization. This means that there is a slightly higher chance of disconnecting a node from an already highly utilized bus system and connect it to a better suited network. Further, a `RandomGatewayMutator` class was implemented to alter the deployment of gateway functionalities within the network.

Based on the new topology, feasible routing paths for each signal are determined and stored in the `CommunicationMatrix`. In case no feasible route can be found or available bus systems are already fully utilized, the router can also add new connections to the `BusMatrix`. As a consequence, the sequence in which signals are picked and routed affects the resulting communication paths. In this work, we distinguish three cases:

- Signals are routed in the descending order of their required communication bandwidth. This procedure is realized in the 'Largest Load First' - `LLFRouter` class. The behavior is deterministic, which means that given an identical `InputNetwork` and `BusMatrix`, this router will always produce the same `CommunicationMatrix`.
- As an extension, the routing sequence itself can be seen as an encoded variable and undergoes optimization through evolutionary methods. This is done in the `DefineSequenceRouter` in collaboration with a `RoutingSequenceMutator`. Again, the behavior is deterministic given identical topologies and routing sequences.
- In contrast to the above cases, a 'Random Sequence Router' `RSRouter` class provides the non-deterministic counterpart to the other routing operators. Applying this operator onto the same candidate solution might not always produce the same resulting network utilization.

After all signals are routed, some connections in the `BusMatrix` might not be required at all. Therefore, the candidate solution is trimmed of all unnecessary connections using the `Reducer` class. An overview of all mentioned classes is given in Figure 3.9.

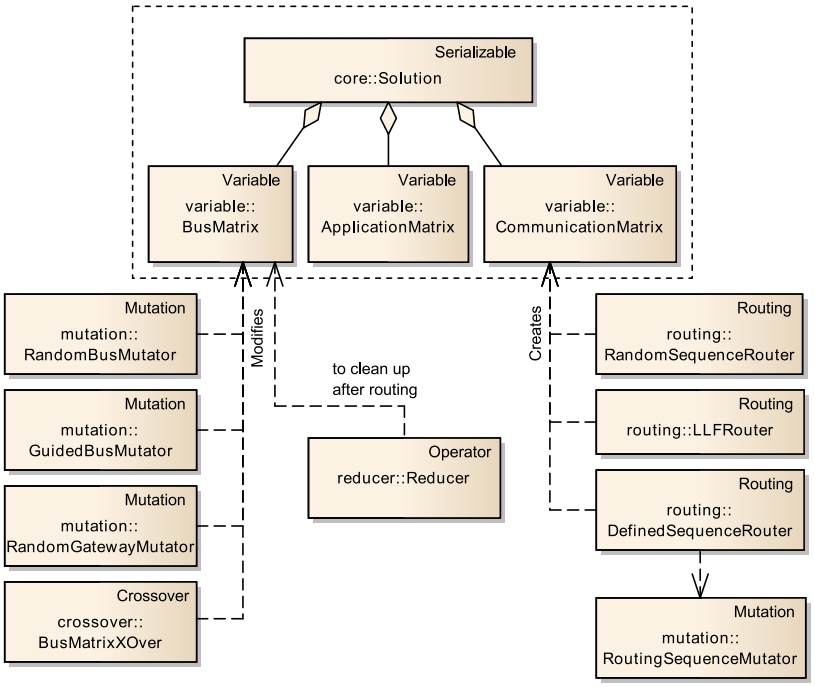


Figure 3.9: Operators relevant to topology optimization

3.5 Objectives

The evaluation of candidate solutions is done using quality metrics or objective functions. jNetOpt provides a set of predefined objective functions for network optimization. It is quite obvious that the quality of optimized solutions depends on the underlying automotive model and practicability of chosen objective functions. Further, the degree of detail in the model that is assumed in this work might not reflect all aspects of a real-world architecture. Therefore, it is strongly suggested that for industrial applications objective functions are adapted based on the actual degree of detail used in the automotive network model.

In the jMetal software architecture, such functions are implemented in the `Problem` class. Based on this architecture, jNetOpt provides a specialization of this class, called `NetworkProblem`. This class of problem implies

that the candidate solutions are of the type `NetworkSolutionType` and therefore contain all necessary objects to evaluate a network.

For our purposes, jNetOpt provides three specialized implementations of the class `NetworkProblem`:

- A `MappingProblem`, which only evaluates the deployment of software components in terms of functional distance and utilization of ECUs. This problem is used for the mapping phase of the two-phase optimization approach. The relevant objectives are node utilization and functional distance, further explained in [93]
- A corresponding `TopologyProblem` to evaluate the utilization of bus systems and monetary costs of wires and bus couplers. The three objectives refer to the second part of the two-phase approach. A more detailed description is again found in the corresponding publication [101].
- Complementary, a combination of both problems to optimize a network using the one-phase optimization approach. In order to keep the overall dimensionality of the objective space within reasonable limits, the cost objectives for wires and bus couplers are combined in this implementation. Therefore, this problem has four objectives.

3.6 Algorithms

In accordance with the one- and two-phase optimization approaches described in subsection 3.1.1, jNetOpt provides three different algorithms for network optimization. For all experiments, we used the well known SPEA2 multiobjective algorithm as a basic framework. The fitness assignment and environmental selection required by this metaheuristic were inherited from the jMetal implementation. Further, each algorithm was designed to work with interchangeable operators for crossover, mutation and routing to allow a further range of experimental opportunities.

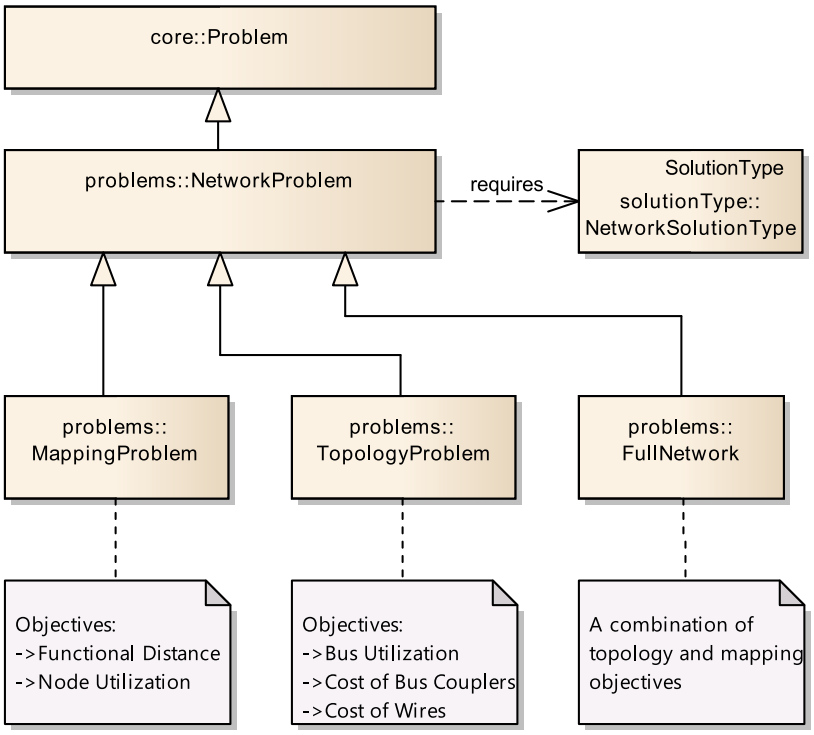


Figure 3.10: Provided implementations of the `NetworkProblem` class

3.6.1 OnePhaseSPEA2

This is the first implementation of a network optimization heuristic. In contrast to later approaches, this algorithm optimizes application mapping and network topology at the same time. It does this by randomly selecting one of three variation paths for each offspring created as shown in Figure 3.11. The flow labels 1/3, 2/3 and 1/2 refer to the probability of choosing the respective variation path. For the sake of clarity, the variation process is only depicted for one offspring from the crossover process.

The paths of variation focus on distinct targets. One path optimizes the mapping of software components onto ECUs by evolutionary variation of the `ApplicationMatrix`. If a new solution outperforms others in mapping-related objectives, it will be passed on to the next generation.

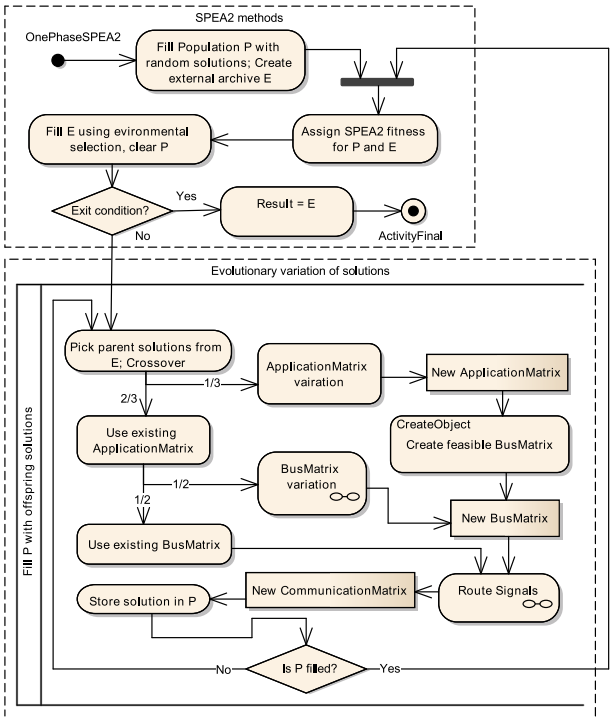


Figure 3.11: Flow diagram of the OnePhaseSPEA2 heuristic

Another path keeps the current mapping and alters the bus topology. Again, distinct quality metrics will reveal an improved solution. The third path does not alter mapping or topology variables. It only applies a new signal routing on an existing network. Depending on the router used, this can also include a variation of the routing sequence.

3.6.2 MappingSPEA2

This algorithm is designed to optimize only the software mapping of a network. Bus variations as well as evaluations regarding topology qualities are completely omitted as seen in Figure 3.12. Therefore, the algorithm is also suitable for investigating the effects of guided mapping mutation operators in detail.

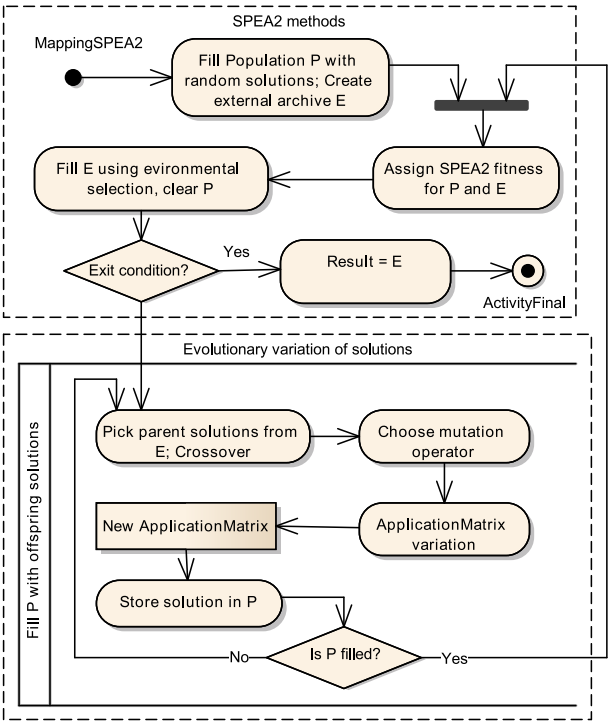


Figure 3.12: Flow diagram of the MappingSPEA2 heuristic

The result of this algorithm is a set of Pareto-optimal mapping solutions. In contrast to all other algorithms, those solutions do not represent a fully functional network and a second phase of topology optimization is required.

3.6.3 TopologySPEA2

This algorithm can be utilized for two use cases. First, as a standalone topology optimization for a network of fixed software mappings. The second case is as subsequent optimization on a set of possible mapping solutions. Accordingly, the initial set of solutions is composed of one or many possible software deployments.

Care must be taken when choosing parent solutions for the crossover operator. A crossover of different mapping solutions could lead to infeasible offspring individuals. Consequently, the implemented operator checks for equality before performing the mating operation.

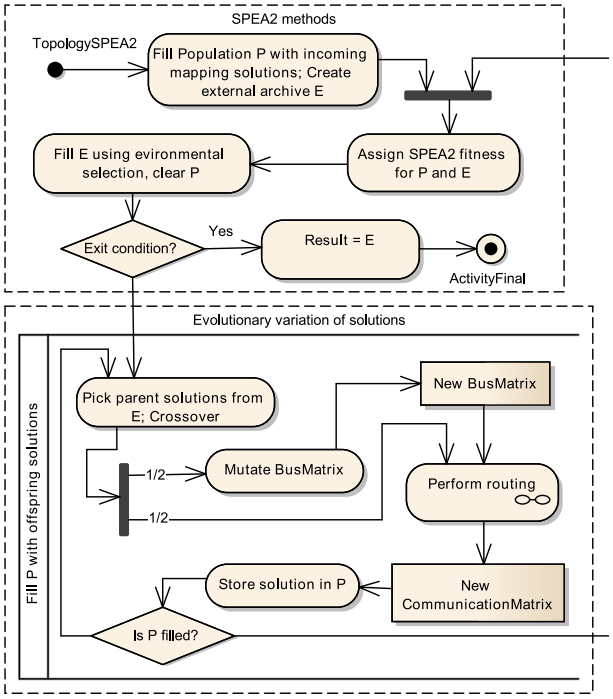


Figure 3.13: Flow diagram of the TopologySPEA2 Heuristic

Similarly to the `OnePhaseSPEA2`, the algorithm performs a mutation operation on the `BusMatrix` with the probability of 1/2. Then, all signals are routed on the resulting bus topology. Mutation and routing operators are again interchangeable for experimental purposes.

3.7 Graphical User Interface

While the `Experiment` class in jMetal provides all features for scientific studies, a graphical user interface for demonstration and automotive test purposes was also a requirement. According to the MVC design pattern,

two Java-based frames were developed for controlling optimization parameters and visualization of resulting networks.

3.7.1 Controller

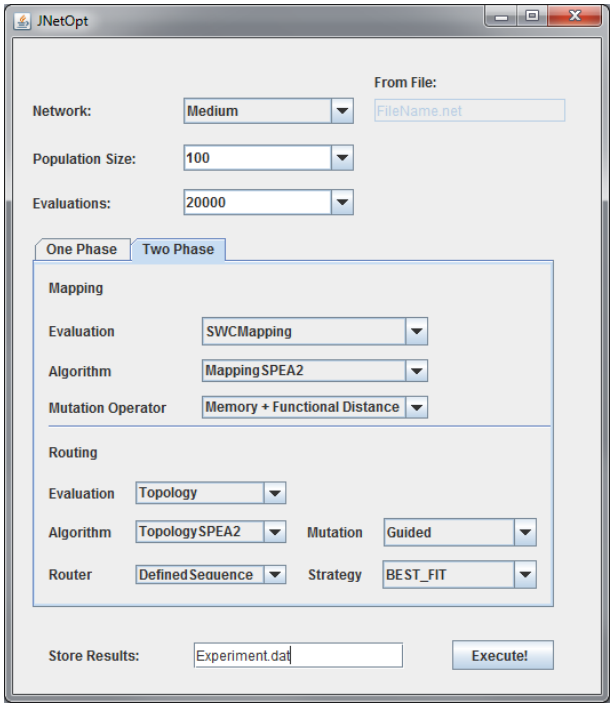


Figure 3.14: Graphical user interface for execution of the optimization

Figure 3.14 shows the execution environment of jNetOpt. Input networks are generated or loaded from external files using the `NetworkParser` class. General settings of population sizes and number of total evaluations are assumed equal for all algorithms. After these global settings, the user can choose between a one-phase and two-phase optimization approach and the relevant algorithms and problem instances. Optional parameters such as the kind of mutation and routing operators are also customizable.

jNetOpt provides continuous logging capabilities during optimizations. This information as well as all optimization results are optionally stored in a separate file after all algorithms are finished.

3.7.2 View

After a successful optimization, the resulting network solutions can be displayed as shown in Figure 3.15. In the top left list, one solution out of the Pareto-optimal set of solutions can be chosen for further evaluation. Below, rudimentary Pareto-diagrams can be displayed. More detailed information for a selected solution as well as a graphical visualization is shown on the right hand side.

Additional information can be extracted in two ways. The instances of `ApplicationMapping` and `CommunicationMatrix` can be displayed on a console environment in Eclipse. Also, each ECU in the graphical network representation shows additional information about mapped software components when moving the mouse pointer over the respective ECU.

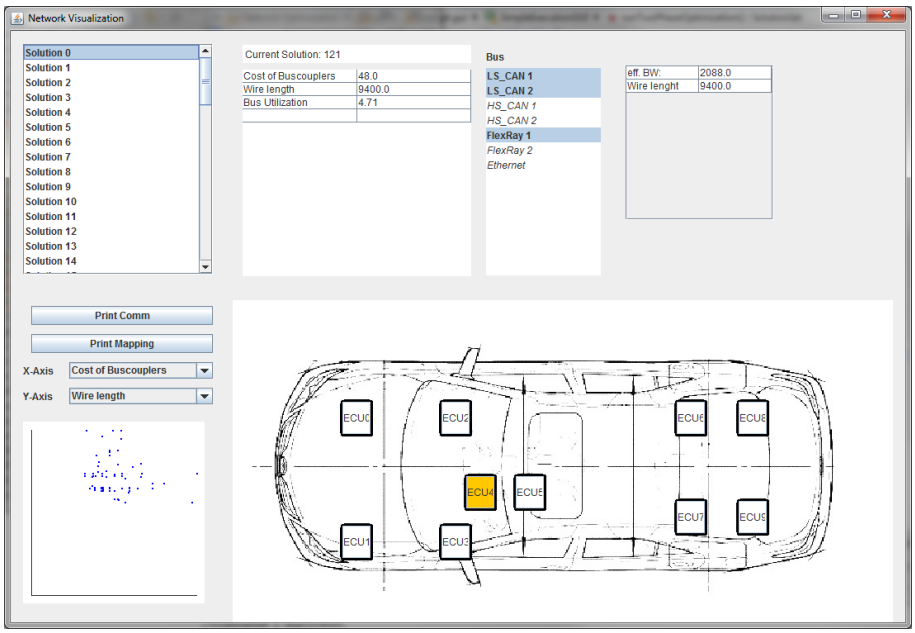


Figure 3.15: Graphical user interface of an example network visualization

4 Experiments and Results

During the development of this thesis, several studies have been performed to support the scientific aspects and design considerations. The main topics were improvements of the optimization performance due to guided mutation operators as pointed out in [93, 101]. Further, the separation of the presented optimization problem into two phases proved to be very effective. This work-flow was transformed into a generalized form and further studied in [105]. In this chapter, I will give a summary of the experiments performed and the resulting insights.

Evolutionary computation is a stochastic procedure. Therefore, each optimization run will provide a slightly different result. As a consequence, comparisons between different algorithms and operator setups require several independent runs for each setup and according statistical evaluation. It is common practice to publish such evaluations as 'box plots' in order to show the distribution of quality indicators over a set of independently executed experiments.

4.1 Guided Mapping Mutation

The idea of guided mutation was introduced in subsection 2.4.4. The first realization was the development of two advanced mutation operators for mapping optimization as shown in subsection 3.4.1. They represent a counterpart to the standard stochastic mutation operator.

In order to evaluate the performance of these operators, five different setups were chosen. The first three setups solely utilize one mutation operator during optimization. They are used to verify the functionality of the guided operators in comparison to a random mutation. The

fourth setup utilizes both guided operators during optimization. For each evolutionary iteration, i.e. each variation of a candidate solution, one guided mutation operator is chosen randomly. The operator setup 5 utilizes all three operators in the same fashion. A summary of these setups is given in Table 4.1.

Table 4.1: Usage of mutation operators in different operator setups [93]

	Operator Setup				
	1	**2**	**3**	**4**	**5**
Functional Distance Mutation	x			x	x
Node Utilization Mutation		x		x	x
Random Mutation			x		x

For each setup, the mapping optimization algorithm `MappingSPEA2` (subsection 3.6.2) was executed for 100 independent runs. The test network consists of 40 ECUs and 60 software components with varying memory and communication requirements. Each optimization run was stopped after 20.000 evaluations. After each run, the objectives functional distance and node utilization (see `MappingProblem`, section 3.5) were evaluated. A subsequent topology and routing optimization was not conducted as it has no effects on the mapping optimization goals.

For a better visual representation, representative Pareto fronts of each operator setup are depicted in Figure 4.1. It can be seen that the exclusive usage of either guided mutation operator already outperforms the random reference. However, the resulting fronts also converge towards their respective optimization objective and do not cover the whole search space efficiently. Interchanging the operators, as done in setups 4 and 5, leads to the overall best results.

Finally, the quality indicators Hypervolume and Epsilon were calculated for each result. Figure 4.2 shows box plots for these indicators. Setups 2 and 3 are clearly inferior in terms of covered Hypervolume which is not surprising after reviewing the representative Pareto fronts in Figure 4.1. But only the ϵ-Indicator reveals the better distributed solutions of the

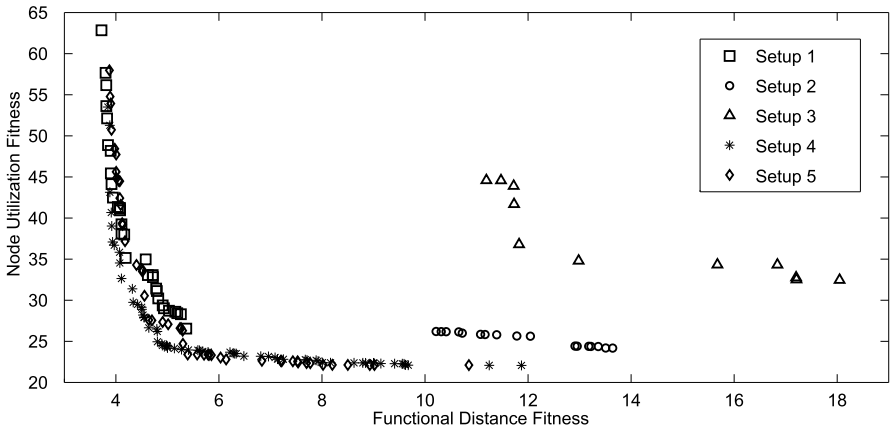
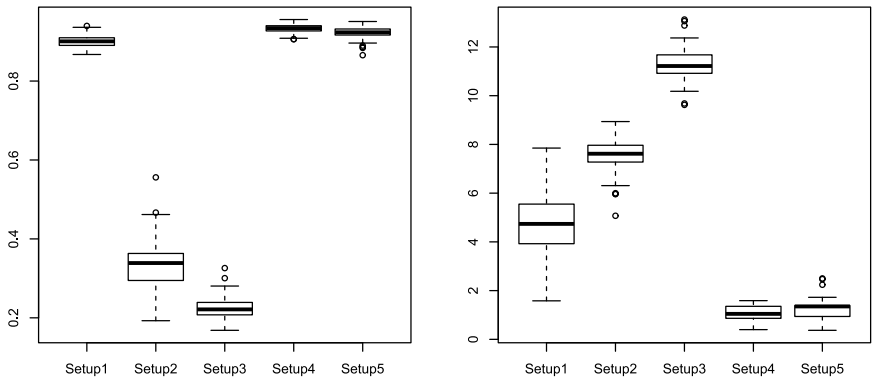


Figure 4.1: Exemplary Pareto fronts for each mapping optimization setup [93]

combined setups 4 and 5 in comparison to the exclusively used guided operator in setup 1.



(a) Hypervolume (b) Epsilon

Figure 4.2: Distribution of quality indicators for mapping optimization [93]

Concluding, the experiments have shown that the guidance developed for mapping mutation operators clearly outperforms random mutation in every aspect. However, they tend to show a local behavior towards their respective optimization goals. This can be prevented by interchanging the guided operators as done in setup 4. The difference to setup 5,

which also utilizes the random mutation operator next to both guided variants, is not statistically significant. It might be suitable to utilize random mutation in order to ensure a better explorative behavior of the algorithm. This is of special importance for constrained and multimodal optimization problems.

4.2 Guided Topology Mutation and Routing

For the topology optimization phase, jNetOpt provides two mutation operators and three kinds of routing algorithms as described in subsection 3.4.2. Next to the random mutation operator, a guided variant is influenced by the current bus utilization for each bus system individually. Routing operators are distinguished by the sequence in which they process the signals to be routed. jNetOpt provides mechanics for a randomized sequence, routing the signals with the largest bus load first (LLF) and a defined sequence of signals. This defined sequence is again subject to mutation during optimization.

Again, the advanced guided mutation process of the `BusMatrix` is compared to standard randomized mutation. Further, the different routing algorithms are applied in combination with each mutation strategy. This leads to 6 different operator setups summarized in Table 4.2. Preliminary studies have shown that interchanging operators, as done in the previous experiment, does not yield any improvements which is why this feature was omitted here.

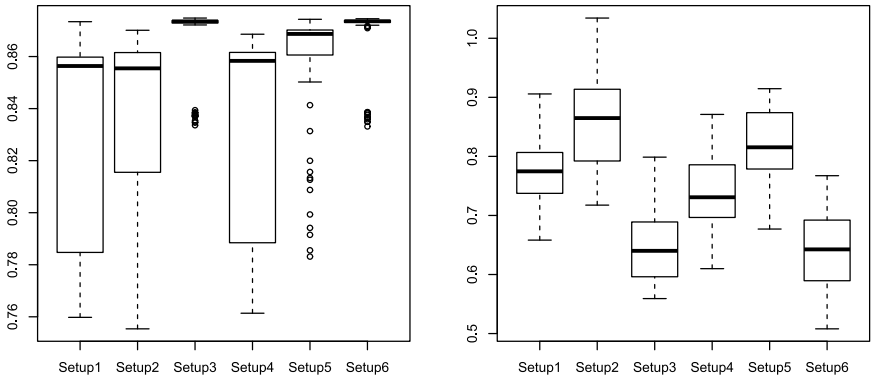
Table 4.2: Operator setups for topology optimization experiments

	Operator Setup					
	1	**2**	**3**	**4**	**5**	**6**
Mutation Operator	Random			Guided		
Router	Random	LLF	Def. Seq.	Random	LLF	Def. Seq

The test network consists of 40 ECUs and 200 signals that need to be transmitted over up to 8 available bus systems. Each operator setup

was tested by executing the `TopologySPEA2` (subsection 3.6.3) algorithm for 100 independent runs. Each optimization run was terminated after 20.000 evaluations. For this phase, the mapping of software components onto ECUs was set as static. Consequently, the objectives from `TopologyProblem` were utilized to calculate quality indicators for each run.

As in the last experiment, the box plots of Hypervolume and Spread [70] indicators are plotted in Figure 4.3. The hypervolume indicator clearly shows that setups 3 and 6, which utilize the defined sequence router, provide the best performance. This is also confirmed when evaluating the spread indicator. Overall, only a slight improvement of guided mutation (setups 4, 5 and 6) can be seen, particularly for the `LLFRouter`. This behavior is again indicated when looking at the Inverse Generational Distance indicator in Figure 4.4.



(a) Hypervolume (b) Spread

Figure 4.3: Distribution of quality indicators for topology optimization

Summarizing, it can be stated that the combination of mutation and routing strategies has a large influence on the optimization result. The improvement of guided mutation has the greatest impact when using the purely deterministic `LLFRouter`. When adding non-deterministic elements into the routing process, as seen in the random sequence router, the guided mutation effect diminishes. However, the best results where achieved with the defined sequence router. Therefore it can be deduced that it is beneficial to treat the actual signal routing sequence

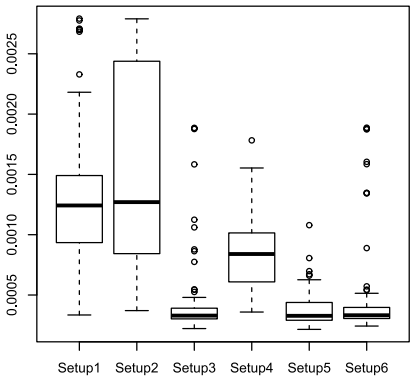


Figure 4.4: IDG indicator for topology optimization

as evolutionary variable and let it evolve in combination with other topology information stored in the `BusMatrix`.

4.3 Many-Objective Optimization

The 2-phase optimization approach in jNetOpt features a novelty in evolutionary computation as the resulting set of solutions from the mapping optimization represents the initial population of the topology optimization. Based on this deconstruction, an increased optimization performance was observed. However, the application of this technique is limited to the very specific structure of the optimization problem presented in this thesis.

As stated in subsection 1.4.6, the performance of multiobjective optimization algorithms decreases with the number of objectives. Therefore, it is desirable to apply the technique used onto other many-objective applications. In order to utilize the positive effects of problem decomposition and sequential usage of MOEAs for general optimization, two questions arise:

- How can the deconstruction technique be applied to uncorrelated multiobjective problems?
- How are different kinds of evolutionary algorithms affected?

To generalize this approach, let us assume a k-dimensional, many-objective optimization problem $\mathbf{f}(\mathbf{x})$ separated into a set of 2- or 3-dimensional sub-problems:

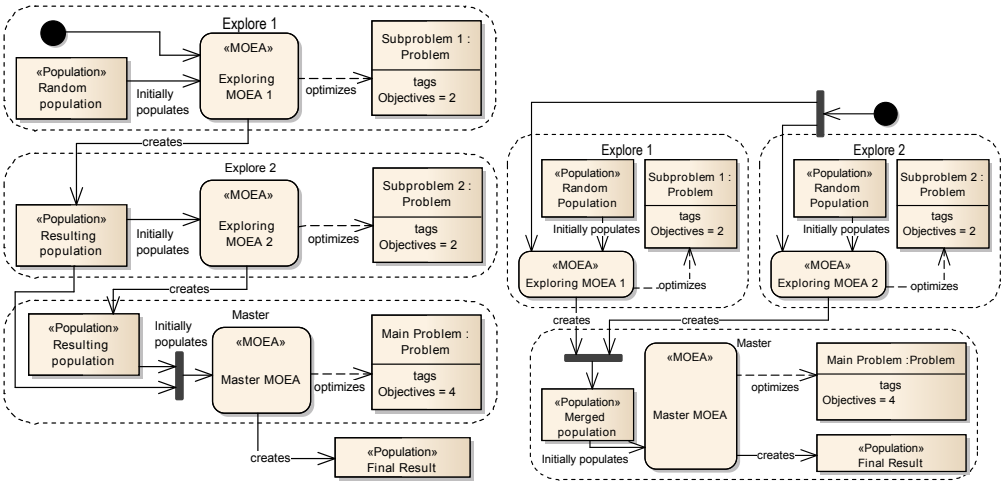
$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= [f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T \\ \mathbf{f}(\mathbf{x}) &= \{\mathbf{f}_{s1}(\mathbf{x}), \ldots, \mathbf{f}_{sn}(\mathbf{x})\} \\ 2 \times n \leq k &\leq 3 \times n \end{aligned} \tag{4.1}$$

As a subsequent step, I propose the optimization of the sub-problems using 'exploring' MOEAs. The resulting solutions, representing good approximations in some areas of the global PF^{true}, are then consolidated to the initial population of the many-objective 'master' algorithm. This technique can be applied to any kind of traditional evolutionary algorithm. For this experiment, four representative MOEAs were chosen:

- NSGAII [70] and SPEA2 [71] representing Pareto-dominance based EAs
- The hypervolume-indicator based IBEA [76]
- MOEA/D as a state of the art MOEA used in multiobjective optimization [77]

The cooperation between exploring and master algorithms can be implemented in two ways. In both approaches, the optimization of the first sub-problem is done using a randomly initialized starting population. In the 'sequential' approach, the resulting set of solutions from the first optimization is used as an initial population for the following sub-problem. In contrast to that, the 'parallel' approach begins each exploring optimization with a randomly initialized population.

After all sub-problems have been optimized, the 'master' algorithm is initiated with resulting solutions from each exploring algorithm. In order to easily distinguish between those two approaches, the prefixes 'S' and 'P' have been added to the algorithm under test to indicate the sequential and parallel method respectively. Further, the flow diagram for both approaches for an example 4-dimensional problem are depicted in Figure 4.5.



(a) Sequential (b) Parallel

Figure 4.5: Workflows for enhanced many-objective optimization [105]

As a benchmark problem, the multiobjective Quadratic Assignment Problem [115] was chosen. Three instances consisting of 4, 6 and 12 objectives were created using the problem generator from [116]. All relevant parameters and dimensions of the respective sub-problems are summarized in Table 4.3.

Table 4.3: Parameters for mQAP benchmark problems [105]

Name	**20-4fl-uni**	**10-6fl-uni**	**15-12fl-uni**
Facilities	20	10	15
Objectives	4	6	12
Subproblems	2 × 2 Obj.	2 × 3 Obj.	4 × 3 Obj.
maxFlow	100		
Correlation	0		
Seed	23453464		

For each evolutionary algorithm under test, the performance of sequential and parallel approaches was compared to the unaltered variant as reference. To allow fair comparison, two total evaluation budgets (small

and large) were chosen for each setup. The corresponding allocation of evaluations between master and explorer algorithms as well as according population sizes are given in Table 4.4.

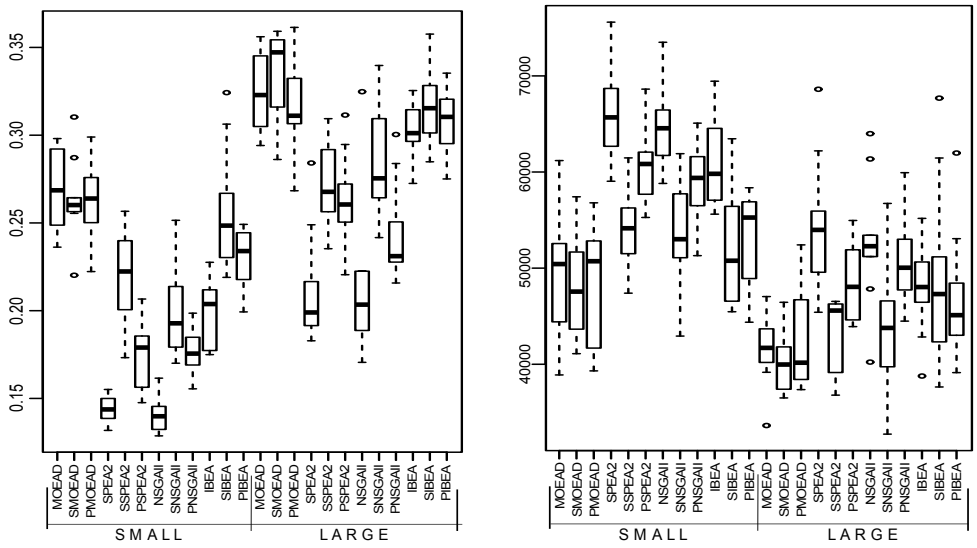
Table 4.4: Population sizes and evaluation budgets of sequential and parallel methods compared to unaltered reference algorithms [105]

		SMOEA/PMOEA			Reference
20-4fl-uni	Population Size		Explorer:	100	400
			Master:	400	
	Evaluation Budget	Small	Explorer:	2×2500	15000
			Master:	10000	
		Large	Explorer:	2×3000	30000
			Master:	24000	
10-6fl-uni	Population Size		Explorer:	120	500
			Master:	500	
	Evaluation Budget	Small	Explorer:	2×2500	15000
			Master:	10000	
		Large	Explorer:	2×10000	75000
			Master:	55000	
15-12fl-uni	Population Size		Explorer:	120	1200
			Master:	1200	
	Evaluation Budget	Small	Explorer:	4×5000	120000
			Master:	100000	
		Large	Explorer:	4×15000	300000
			Master:	240000	

The quality indicators Hypervolume and Epsilon were calculated for each result of the 4- and 6-dimensional problem instances. The corresponding box plots for 100 independent runs are depicted in Figure 4.6 and Figure 4.7.

When looking at the 4-objective problem, the improvements for SPEA and NSGAII are evident in all cases. The highest increase in performance is achieved when applying the sequential method, labeled SSPEA2

and SNSGAII respectively. Also, IBEA does profit from the method when only using a small evaluation budget. However, this advantage diminishes when a higher number of total evaluations is allowed. The MOEA/D algorithm does not significantly benefit from any kind of sequential usage in this study.

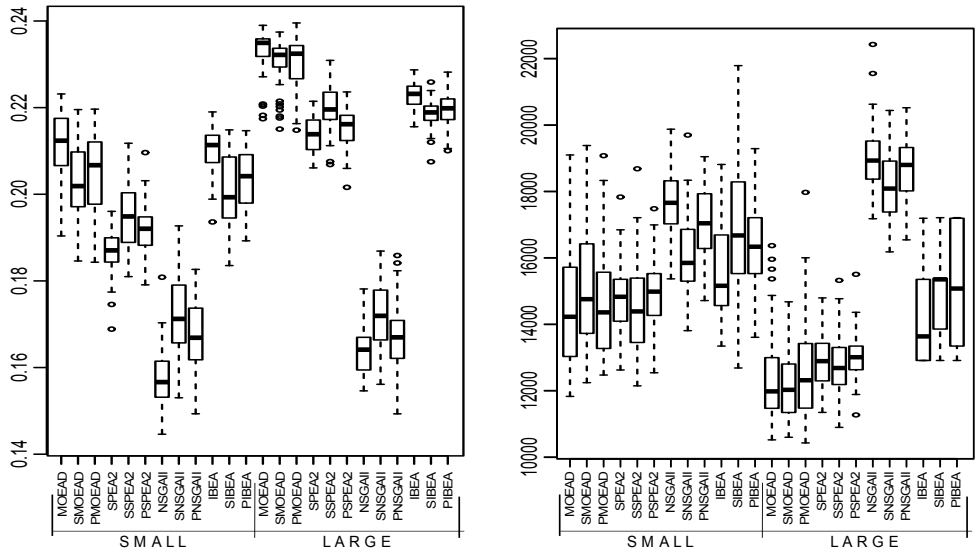


(a) Hypervolume (b) Epsilon

Figure 4.6: Resulting quality indicators for the 4-objective problem [105]

When applying the proposed methods onto the 6-dimensional problem instance, the results vary. The Pareto-based SPEA2 and especially NSGAII algorithms still benefit from the sequential approach. But as the number of objectives increases, their overall performance is clearly inferior to the modern MOEA/D and IBEA heuristics. However, as pointed out in the corresponding research paper [105], the evaluations of the exploring algorithms are much faster due to the lower-dimensional sub-problems. Therefore, the execution time of sequential and parallel optimization methods is reduced by 10 to 20% compared to the unaltered algorithms.

The computational complexity of the 12-dimensional problem instance required some changes to the experiment setup. First, the execution time of SPEA2 and the hypervolume-based IBEA were too high to finish the

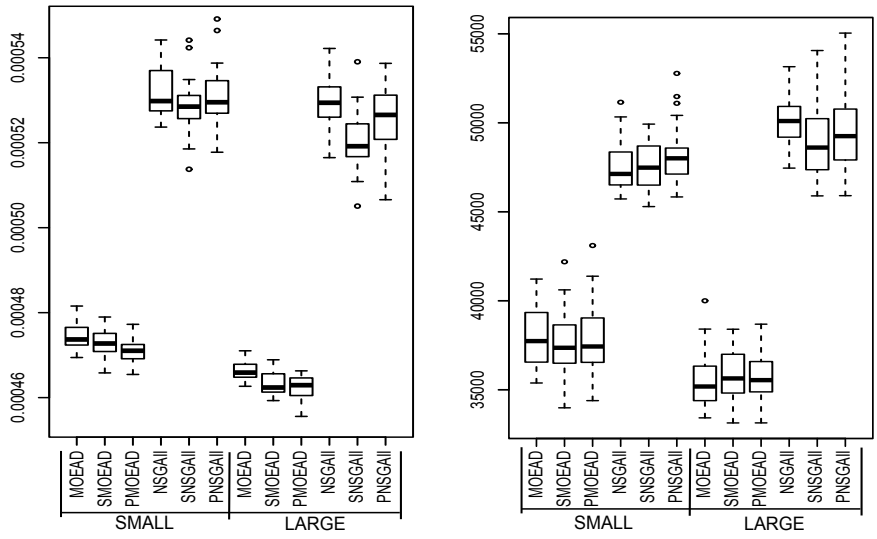


(a) Hypervolume (b) Epsilon

Figure 4.7: Resulting quality indicators for the 6-objective problem [105]

experiment in a reasonable time. Therefore, the comparison was only made between NSGAII and MOEA/D with a reduced number of only 30 independent runs. Further, the calculation of the hypervolume indicator for all resulting Pareto fronts was also impracticable. The much faster to calculate Inverted Generation Distance was used as a surrogate.

The results are shown in Figure 4.8. The expected superior performance of MOEA/D over NSGAII is obvious. Nevertheless, NSGAII does still benefit from the sequential application of exploring algorithms. The IGD indicator also reveals a performance increase for sequential and parallel approaches in combination with MOEA/D. This increase, combined with a faster execution time of up to 25%, underlines the performance and applicability of the proposed method for this problem instance.



(a) IGD (b) Epsilon

Figure 4.8: Resulting quality indicators for the 12-objective problem [105]

4.4 Discussion

In this chapter, the key advantages of jNetOpt have been studied.

First, the effects of mutation operators guided by objective functions have been demonstrated on the mapping optimization algorithm. Using a guided mutation operator for each objective results in a higher convergence towards the respective optimization goal. The negative effect of premature convergence and poor coverage of the whole Pareto front can be averted by interchanging the mutation operators during optimization.

Secondly, the effects of different routing strategies and guided mutation has been evaluated during topology optimization. Here, the optimization structure consists of evolutionary alterations of the bus topology followed by a routing mechanism for each signal. The key finding is that guided mutation is best suited when the following routing sequence is deterministic. Another result is that the best overall routing performance has been achieved using the defined sequence router. Therefore, it is

advisable to view the actual routing sequence as part of the evolutionary variables.

The final improvement was observed when using the two-phase optimization approach in comparison to the inferior one-phase algorithm. Based on this insight, a general form of this deconstruction has been proposed: Partitioning a many-objective optimization problem into 2- or 3-dimensional sub-problems. Apply efficient algorithms to optimize those sub-problems in exploratory phases and merge the resulting solutions as the initial population of the many-objective master algorithm. Experiments have shown that this method is very suitable for Pareto-based MOEAs. At higher objective dimensions, the exploratory phased also improve the performance of the very modern MOEA/D algorithm.

5 Conclusion

In this thesis, I presented a framework for the optimal architecture of automotive networks. The main focus is laid upon two tasks:

- The deployment of functional software components onto ECUs.
- Optimal layout of network topologies and gateways.

The motivation for the first task arises mainly from the exponential increase of software functionalities in modern cars. New design paradigms have to be found to ensure reliable operation especially for safety-related components in a networked environment. Further, the market demand for novel electronic features is no longer limited to the premium segment of vehicles. This induces a massive cost pressure for OEMs and tier-one suppliers.

One major driving force in this area is the AUTOSAR initiative, providing new standards in automotive software architecture. Important aspects of this architecture are the re-usability of software and decoupling of functional components from the underlying embedded operating system and hardware layers. This leads to new degrees of freedom for system architects as they can now choose the executing hardware unit for software components more independently. A new potential for optimization is discovered here in terms of communication requirements and utilization of ECUs.

The second task deals with topology optimization using state of the art network technologies. Due to the introduction of hybrid and purely electric vehicle concepts, the suitability of traditional network designs is questionable. Further, higher communication bandwidths for applications like multimedia streaming or driving assistance require communication over new network technologies such as automotive Ethernet.

For both tasks, an interpretation based on combinatorial problems is stated. This interpretation is one of several reasons to choose evolutionary computation as a suitable heuristic and optimization approach.

In contrast to existing works, this thesis discusses several methods to improve the optimization process. First, the two tasks were realized in the framework as two distinct optimization phases. For both phases, a common representation scheme was developed to efficiently encode network solutions and given constraints. Additionally, advanced genetic operators have been introduced for each phase, utilizing the application-specific encoding to improve the optimization performance.

A concluding step is the generalization of the two-phase approach to enhance many-objective optimization. This resulted in a hierarchical optimization approach consisting of several lower-dimensional MOEAs to explore the search space in certain regions. The results of those exploring phases represent the initial solution of the master optimization algorithm. Performance improvements in the field of computationally intensive many-objective optimization have been shown for four representative evolutionary algorithms in terms of convergence and run time. This technique can be applied to any kind of population-based metaheuristic.

The software architecture has been constructed using modular design patterns. This allows an efficient implementation of further extensions for the network model and interfaces to various automotive tool-chains. Therefore, the integration of a network optimization step into an existing automotive design process is feasible. This was also shown in a 'Proof of Concept' using the architecture tool PREEVision.

Another important aspect is the clear partitioning between optimization logic and quality metrics from the automotive industry. Studies and feedback from the automotive industry have shown that the effort put into model driven development of EE architectures varies between OEMs, system suppliers and also different vehicle platforms. Therefore, the implementation of quality metrics is encapsulated to allow application-specific levels of detail.

Finally, the quality of optimization always depends on the quality of the underlying model. Today, AUTOSAR has reached a level of maturity and acceptance that allows the efficient model driven development of software-based features in modern cars. This encourages the application of optimization in the automotive industry as a solution to the imminent cost pressure and reliability requirements.

List of Publications

Martin Dohr and Bernd Eichberger, "An Application-Specific Approach in Automotive Network Optimization" in *Proceedings of the 2012 International Conference on Genetic and Evolutionary Methods (GEM)*, Las Vegas, USA, July 2012

Martin Dohr and Bernd Eichberger, "Guided Mutation Strategies for Multi-Objective Automotive Network Architecture" in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC)*, Cancún, Mexico, June 2013

Martin Dohr and Bernd Eichberger, "Evolutionary Routing Strategies for Automotive Networks" in *Proceedings of the 2013 International Conference on Genetic and Evolutionary Methods (GEM)*, Las Vegas, USA, July 2013

Martin Dohr and Bernd Eichberger, "Improving Many-Objective Optimization Performance by Sequencing Evolutionary Algorithms" in *Proceedings of the 14th annual conference on Genetic and Evolutionary Computation (GECCO 2014)*, Vancouver, BC, Canada, July 2014.

List of Figures

1.1 AUTOSAR development methodology 15
1.2 AUTOSAR ECU software architecture 16
1.3 AUTOSAR basic approach 17
1.4 Starting solution and final nozzle shape optimized by evolutionary strategies 24
1.5 Single point crossover 31
1.6 Plot of Pareto optimal solutions and dominated objective space 35
1.7 Example hypervolume for 2-dimensional objective space . 37
1.8 Example for poor divergence of PF^{known} 38

2.1 Integration of network optimization in E/E architecture development process 48

3.1 The Model-View-Controller design pattern applied in jNetOpt 68
3.2 The proposed workflows of jNetOpt 70
3.3 Overview of the one-phase optimization approach 71
3.4 Overview of the two-phase optimization approach 72
3.5 Class diagram of `InputNetwork` and example test networks 73
3.6 An example of mapping of software components onto ECUs 75
3.7 An example of connections of ECUs and bus networks . . 76
3.8 Operators relevant to mapping optimization 78
3.9 Operators relevant to topology optimization 80
3.10 Provided implementations of the `NetworkProblem` class . . 82
3.11 Flow diagram of the OnePhaseSPEA2 heuristic 83
3.12 Flow diagram of the MappingSPEA2 heuristic 84
3.13 Flow diagram of the TopologySPEA2 Heuristic 85
3.14 Graphical user interface for execution of the optimization 86

3.15 Graphical user interface of an example network visualization 88

4.1 Representative Pareto fronts for each mapping optimization setup 91
4.2 Distribution of quality indicators for mapping optimization 91
4.3 Distribution of quality indicators for topology optimization 93
4.4 IDG indicator for topology optimization 94
4.5 Workflows for enhanced many-objective optimization . . . 96
4.6 Resulting quality indicators for the 4-objective problem . . 98
4.7 Resulting quality indicators for the 6-objective problem . . 99
4.8 Resulting quality indicators for the 12-objective problem . 100

List of Tables

1.1 Classifications of automotive bus systems 3
1.2 Resulting fitness values for the example multiobjective problem 34
1.3 Example run times for polynomial and exponential complexity 41

2.1 Summary of mapping optimizations 63
2.2 Summary of topology optimizations 64
2.3 Summary of topology and routing optimizations 65

4.1 Usage of mutation operators in different operator setups . 90
4.2 Operator setups for topology optimization experiments . 92
4.3 Parameters for mQAP benchmark problems 96
4.4 Population sizes and evaluation budgets of sequential and parallel methods compared to unaltered reference algorithms 97

Bibliography

[1] Robert Bosch GmbH. (2014, July) CAN - Controller Area Network. [Online]. Available: http://www.semiconductors.bosch.de/en/ipmodules/can/can.asp

[2] *SAE J2057/1*, Society of Automotive Engineers Std., 2006.

[3] R. I. Davis and A. Burns, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Refuted, Revisited and Revised. Real-Time Systems*, vol. 35, pp. 239–272, 2007.

[4] O. Hartkopp, "Programmierschnittstellen für eingebettete Netzwerke in Mehrbenutzerbetriebssystemen am Beispiel des Controller Area Network," Ph.D. dissertation, Otto-von-Guericke-Universität Magdeburg, 2011.

[5] W. Zimmermann and R. Schmidgall, *Bussysteme in der Fahrzeugtechnik: Protokolle und Standards*, 3rd ed. Wiesbaden: Vieweg, 2008.

[6] *CAN with Flexible Data-Rate*, Robert Bosch GmbH Std., March 2014. [Online]. Available: http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can_fd_spec.pdf

[7] LIN Administration. (2013, April) LIN - Local Interconnect Network. [Online]. Available: http://www.lin-subbus.org

[8] FlexRay Consortium. (2013, May) FlexRay Communications System. [Online]. Available: http://www.flexray.com

[9] MOST Cooperation. (2014, April) MOST - Media Oriented Systems Transport. [Online]. Available: http://www.mostcooperation.com

[10] P. Hank, T. Suermann, and S. Müller, "Automotive Ethernet, a Holistic Approach for a Next Generation In-Vehicle Networking Standard," in *Advanced Microsystems for Automotive Applications 2012*, G. Meyer, Ed. Springer Berlin Heidelberg, 2012, pp. 79–89. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29673-4_8

[11] Robert Bosch GmbH. (2014, April) Peripheral sensor interface 5. [Online]. Available: http://psi5.org/

[12] *SAE J2716 - SENT - Single Edge Nibble Transmission for Automotive Applications.*, Society of Automotive Engineers Std., 2010.

[13] DSI Consortium. (2013, September) DSI - Distributed Systems Interface. [Online]. Available: http://www.dsiconsortium.org/

[14] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," *Embedded World*, vol. 2004, pp. 235–252, 2004.

[15] D. Goswami, M. Lukasiewycz, M. Kauer, S. Steinhorst, A. Masrur, S. Chakraborty, and S. Ramesh, "Model-based development and verification of control software for electric vehicles," in *Proceedings of the Design and Automation Conference (DAC)*, Austin, USA, 2013.

[16] B. Hardung, T. Kölzow, and A. Krüger, "Reuse of Software in Distributed Embedded Automotive Systems," in *Proceedings of the 4th ACM International Conference on Embedded Software*, ser. EMSOFT '04. New York, NY, USA: ACM, 2004, pp. 203–210. [Online]. Available: http://doi.acm.org/10.1145/1017753.1017787

[17] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th international conference on Software engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/1134285.1134292

[18] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *2007 Future of Software Engineering*, ser. FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 55–71. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.22

[19] R. N. Charette, "This Car Runs on Code," *IEEE Spectrum*, Feb. 2009.

[20] M. Conrad, I. Fey, M. Grochtmann, and T. Klein, "Modellbasierte entwicklung eingebetteter fahrzeugsoftware bei daimlerchrysler," *Informatik - Forschung und Entwicklung*, vol. 20, no. 1-2, pp. 3–10, 2005. [Online]. Available: http://dx.doi.org/10.1007/s00450-005-0197-5

[21] N. Mellegård and M. Staron, "Characterizing model usage in embedded software engineering: a case study." in *ECSA Companion Volume*, ser. ACM International Conference Proceeding Series, I. Gorton, C. E. Cuesta, and M. A. Babar, Eds. ACM, 2010, pp. 245–252. [Online]. Available: http://dblp.uni-trier.de/db/conf/ecsa/ecsa2010c.html#MellegardS10

[22] C. Bodenstein, F. Lohse, and A. Zimmermann, "Executable specifications for model-based development of automotive software," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, Oct 2010, pp. 727–732.

[23] AUTOSAR Consortium. (2013, November) Automotive Open System Architecture. [Online]. Available: http://www.autosar.org

[24] S. Furst, "Challenges in the design of automotive software," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, March 2010, pp. 256–258.

[25] C. Darwin, *On the Origin of the Species by Means of Natural Selection: Or, The Preservation of Favoured Races in the Struggle for Life*. John Murray, 1859. [Online]. Available: http://books.google.at/books?id=jTZbAAAAQAAJ

[26] D. Fogel, "An introduction to simulated evolutionary optimization," *Neural Networks, IEEE Transactions on*, vol. 5, no. 1, pp. 3–14, 1994.

[27] T. Bäck and H.-P. Schwefel, "Evolutionary computation: an overview," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, 1996, pp. 20–29.

[28] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 1, pp. 3–17, 1997.

[29] G. E. P. Box, "Evolutionary Operation: A Method for Increasing Industrial Productivity," *Applied Statistics*, vol. 6, no. 2, pp. 81–101, Jun. 1957.

[30] R. M. Friedberg, "A learning machine: Part i," *IBM Journal of Research and Development*, vol. 2, no. 1, pp. 2–13, 1958.

[31] H. J. Bremermann, "Optimization through evolution and recombination," in *Proceedings of the Conference on Self-Organizing Systems – 1962*, M. C. Yovits, G. T. Jacobi, and G. D. Golstine, Eds. Washington, DC: Spartan Books, 1962, pp. 93–106.

[32] D. B. Fogel, *Evolutionary Computation: The Fossil Record*, 1st ed. Wiley-IEEE Press, 1998.

[33] I. Rechenberg, "Cybernetic solution path of an experimental problem," in *Royal Aircraft Establishment Translation No. 1122, B. F. Toms, Trans.* Farnborough Hants: Ministry of Aviation, Royal Aircraft Establishment, Aug. 1965.

[34] H.-P. Schwefel, "Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik," Diplomarbeit, Technische Universität Berlin, Hermann Föttinger–Institut für Strömungstechnik, März 1965.

[35] J. H. Holland, "Outline for a logical theory of adaptive systems," *J. ACM*, vol. 9, no. 3, pp. 297–314, Jul. 1962. [Online]. Available: http://doi.acm.org/10.1145/321127.321128

[36] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[37] L. J. Fogel, "Autonomous automata," *Industrial Research*, vol. 4, no. 2, pp. 14–19, 1962.

[38] ——, "On the organization of intellect," Ph.D. dissertation, University of California, Los Angeles-Engineering, 1964.

[39] J. Klockgether and H. Schwefel, "Two-phase nozzle and hollow core jet experiments." *Journal Name: pp 141-8 of Engineering Aspects of Magnetohydrodynamics. /Elliott, D. G. (ed.). University, Miss. Univ. of Mississippi (1970).*, Jan 1970.

[40] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002. [Online]. Available: http://dx.doi.org/10.1023/A%3A1015059928466

[41] M. Schumer and K. Steiglitz, "Adaptive step size random search," *Automatic Control, IEEE Transactions on*, vol. 13, no. 3, pp. 270–276, 1968.

[42] A. Auger, "Benchmarking the (1+1) Evolution Strategy with One-Fifth Success Rule on the BBOB-2009 Function Testbed," in *ACM-GECCO Genetic and Evolutionary Computation Conference*, Montreal, Canada, 2009. [Online]. Available: http://hal.inria.fr/inria-00430515

[43] M. Herdy, "Application of the 'evolutionsstrategie' to discrete optimization problems," in *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, ser. PPSN I. London, UK, UK: Springer-Verlag, 1991, pp. 188–192. [Online]. Available: http://dl.acm.org/citation.cfm?id=645821.670344

[44] H. Beyer, *The Theory of Evolution Strategies*. Springer, Berlin, Heidelberg, New York, 2001.

[45] E. Mezura-Montes, "Alternative Techniques to Handle Constraints in Evolutionary Optimization," Ph.D. dissertation, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2004.

[46] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[47] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 2–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=645512.657265

[48] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence*. Piscataway, NJ, USA: IEEE Press, 1995.

[49] ——, "Foundations of evolutionary computation," in *Proceedings of the SPIE, Volume 6228*, 2006, p. 622801.

[50] *Evolutionary Computation*, vol. 1, no. 1, 1993.

[51] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109 – 1130, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722170600292X

[52] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the Cutting-Stock problem," *Operations Research*, vol. 9, no. 6, pp. 849–859, 1961. [Online]. Available: http://dx.doi.org/10.2307/167051

[53] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, 1st ed., T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Bristol, UK, UK: IOP Publishing Ltd., 1997.

[54] D. E. Goldberg and R. Lingle, Jr., "Alleleslociand the traveling salesman problem," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 154–159. [Online]. Available: http://dl.acm.org/citation.cfm?id=645511.657095

[55] W. M. Spears, "Crossover or mutation?" in *FOUNDATIONS OF GENETIC ALGORITHMS 2*. Morgan Kaufmann, 1992, pp. 221–237.

[56] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Urbana*, vol. 51, pp. 61 801–2996, 1991.

[57] T. Bäck, G. Rudolph, and H. Paul Schwefel, "Evolutionary programming and evolution strategies: Similarities and differences," in *In Proceedings of the Second Annual Conference on Evolutionary Programming*, 1994, pp. 11–22.

[58] T. Blickle and L. Thiele, "A comparison of selection schemes used in genetic algorithms," Gloriastrasse 35, CH-8092 Zurich: Swiss Federal Institute of Technology (ETH) Zurich, Computer Engineering and Communications Networks Lab (TIK, Tech. Rep., 1995.

[59] M. Ehrgott, *Multicriteria Optimization (2. ed.)*. Springer, 2005.

[60] C. A. Coello Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[61] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 117–132, 2002.

[62] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Ph.D. dissertation, Air Force Institute of Technology, Wright Patterson AFB, OH, USA, 1999.

[63] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point," in *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, ser. FOGA '09. New York, NY, USA: ACM, 2009, pp. 87–102. [Online]. Available: http://doi.acm.org/10.1145/1527125.1527138

[64] K. Deb, K. Miettinen, and D. Sharma, "A hybrid integrated multi-objective optimization procedure for estimating nadir point," in *EMO*, 2009, pp. 569–583.

[65] J. Bader, K. Deb, and E. Zitzler, "Faster hypervolume-based search using monte carlo sampling," in *MCDM*, 2008, pp. 313–326.

[66] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration," in *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, ser. EMO'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 862–876. [Online]. Available: http://dl.acm.org/citation.cfm?id=1762545.1762618

[67] M. N. Le, Y.-S. Ong, S. Menzel, C.-W. Seah, and B. Sendhoff, "Multi co-objective evolutionary optimization: Cross surrogate augmentation for computationally expensive problems," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, June 2012, pp. 1–8.

[68] S. Helbig and D. Pateva, "On several concepts for ϵ-efficiency," *Operations-Research-Spektrum*, vol. 16, no. 3, pp. 179–186, 1994. [Online]. Available: http://dx.doi.org/10.1007/BF01720705

[69] J. D. Schaffer, "Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition)," Ph.D. dissertation, Vanderbilt University, Nashville, TN, USA, 1984, aAI8522492.

[70] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[71] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation, and Control*. CIMNE, Barcelona, Spain, 2002, pp. 95–100.

[72] V. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, and K. Deb, Eds. Springer Berlin Heidelberg, 2003, vol. 2632, pp. 376–390. [Online]. Available: http://dx.doi.org/10.1007/3-540-36970-8_27

[73] E. Hughes, "Evolutionary many-objective optimisation: many once or one many?" in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1, 2005, pp. 222–227 Vol.1.

[74] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe, "Many-objective optimization: An engineering design perspective," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, ser. EMO'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 14–32. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31880-4_2

[75] J. Bader, "Hypervolume-Based Search for Multiobjective Optimization: Theory and Methods," Ph.D. dissertation, ETH Zurich, Switzerland, 2010.

[76] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII*. Springer, 2004, pp. 832–842.

[77] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 712–731, 2007.

[78] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "Moea/d with adaptive weight adjustment," *Evolutionary Computation*, 2013.

[79] K. Evlin, *The Millennium Problems: The Seven Greatest Unsolved Mathematical Puzzles of Our Time*. Basic Books, 2002. [Online]. Available: http://books.google.at/books?id=Id5QAAAAMAAJ

[80] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209 – 219, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305048304001550

[81] H. Abeledo, R. Fukasawa, A. Pessoa, and E. Uchoa, "The time dependent traveling salesman problem: polyhedra and algorithm," *Mathematical Programming Computation*, vol. 5, no. 1, pp. 27–55, 2013. [Online]. Available: http://dx.doi.org/10.1007/s12532-012-0047-y

[82] A. Zhou, L. Kang, and Z. Yan, "Solving dynamic tsp with evolutionary approach in real time," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 2, Dec 2003, pp. 951–957 Vol.2.

[83] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145 – 159, 1990, cutting and Packing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037722179090350K

[84] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica: journal of the Econometric Society*, pp. 53–76, 1957.

[85] S. Sahni and T. Gonzalez, "P-complete approximation problems," *J. ACM*, vol. 23, no. 3, pp. 555–565, Jul. 1976. [Online]. Available: http://doi.acm.org/10.1145/321958.321975

[86] S. Gabrielsson, "A parallel tabu search alglorithm for the quadratic assignment problem," Master's thesis, Luleå University of Technology, 2007.

[87] Vector Informatik GmbH. (2014, April) PREEVision - Develop E/E Architectures. [Online]. Available: http://vector.com/vi_preevision_en.html

[88] Eclipse Foundation. (2014, April) The Eclipse Foundation open source community website. [Online]. Available: https://www.eclipse.org/

[89] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, no. 2, pp. 657 – 690, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221705008337

[90] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[91] K. Tindell and A. Burns, "Guaranteed message latencies for distributed safety-critical hard real-time control networks," *Report YCS229, Department of Computer Science, University of York, May 1994*, 1994.

[92] K. Klobedanz, C. Kuznik, A. Thuy, and W. Müller, "Timing modeling and analysis for autosar-based software development: A case study," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '10. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2010, pp. 642–645. [Online]. Available: http://dl.acm.org/citation.cfm?id=1870926.1871078

[93] M. Dohr and B. Eichberger, "Guided mutation strategies for multi-objective automotive network architecture," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 2473–2479.

[94] Bernd Hardung, "Optimisation of the Allocation of Functions in Vehicle Networks," Dissertation, Universität, Erlangen, 2006.

[95] I. Meedeniya, B. Buhnova, A. Aleti, and L. Grunske, "Reliability-driven deployment optimization for embedded systems," *J. Syst. Softw.*, vol. 84, no. 5, pp. 835–846, May 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2011.01.004

[96] M. Glass, M. Lukasiewycz, F. Reimann, C. Haubelt, and J. Teich, "Symbolic reliability analysis and optimization of ecu networks," in *Design, Automation and Test in Europe, 2008. DATE '08*, March 2008, pp. 158–163.

[97] C. Bichot and P. Siarry, *Graph Partitioning*, ser. ISTE. Wiley, 2011. [Online]. Available: http://books.google.at/books?id=9Q25cQAACAAJ

[98] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, no. 1, pp. 1–32, Mar. 1996. [Online]. Available: http://dx.doi.org/10.1162/evco.1996.4.1.1

[99] R. Kumar and P. Rockett, "Effective evolutionary multimodal optimization by multiobjective reformulation without explicit niching/sharing," in *Applied Computing*, ser. Lecture Notes in Computer Science, S. Manandhar, J. Austin, U. Desai, Y. Oyanagi, and A. Talukder, Eds. Springer Berlin Heidelberg, 2004, vol. 3285, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30176-9_1

[100] M. Dohr and B. Eichberger, "An application-specific approach in automotive network optimization," in *Proceedings of the 2012 international conference on genetic and evolutionary methods*, jul 2012, pp. 62–67.

[101] ——, "Evolutionary routing strategies for automotive networks," in *Proceedings of the 2013 international conference on genetic and evolutionary methods*, jul 2013, pp. 63–68.

[102] W. Peng and Q. Zhang, "Network topology planning using moea/d with objective-guided operators," in *Parallel Problem Solving from Nature - PPSN XII*, ser. Lecture Notes in Computer Science, C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds. Springer Berlin Heidelberg, 2012, vol. 7492, pp. 62–71. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32964-7_7

[103] C.-M. Chen, Y. ping Chen, and Q. Zhang, "Enhancing moea/d with guided mutation and priority update for multi-objective optimization," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, May 2009, pp. 209–216.

[104] T.-C. Chiang and L.-C. Fu, "An improved multiobjective memetic algorithm for permutation flow shop scheduling," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, July 2010, pp. 1–8.

[105] M. Dohr and B. Eichberger, "Improving many-objective optimization performance by sequencing evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2014)*, Vancouver, BC, Canada, July 2014.

[106] A. Aleti, L. Grunske, I. Meedeniya, and I. Moser, "Let the Ants Deploy Your Software - An ACO Based Deployment Optimisation Strategy," in *ASE*, 2009, pp. 505–509.

[107] M. Heinz, M. Hillenbrand, K. Klindworth, and K.-D. Müller-Glaser, "Rapid automotive bus system synthesis based on communication requirements," in *Rapid System Prototyping (RSP), 2011 22nd IEEE International Symposium on*, may 2011, pp. 53 –58.

[108] B. Müller-Rathgeber, "Bordnetze für verteilte heterogene subsysteme." Ph.D. dissertation, Technical University Munich, 2010, http://d-nb.info/1010478303. [Online]. Available: http://mediatum.ub.tum.de/node?id=972996

[109] R. Moritz, T. Ulrich, and L. Thiele, "Evolutionary Exploration of E/E-Architectures in Automotive Design," in *Proceedings of the International Conference on Operations Research*, Zurich, Switzerland, 2011, pp. 361–366.

[110] M. Limam, *A New Approach for Gateway-based Automotive Network Architectures*, ser. Schriftenreihe des Lehrstuhls Fahrzeugmechatronik der TU Dresden. expert-Verlag, 2010. [Online]. Available: http://books.google.com/books?id=h87LYgEACAAJ

[111] S. Kim, E. Lee, M. Choi, H. Jeong, and S. Seo, "Design optimization of vehicle control networks," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 7, pp. 3002 –3016, sept. 2011.

[112] Sparx Systems. (2014, April) Enterprise Architect UML Modeling Tool. [Online]. Available: http://www.sparxsystems.com

[113] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Chichester, UK: Wiley, 1996.

[114] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997811001219

[115] J. D. Knowles and D. Corne, "Towards landscape analyses to inform the design of hybrid local search for the multiobjective quadratic assignment problem." in *HIS*, ser. Frontiers in Artificial Intelligence and Applications, A. Abraham, J. R. del Solar, and M. Köppen, Eds., vol. 87. IOS Press, 2002, pp. 271–279.

[116] J. Knowles and D. Corne, "Instance generators and test suites for the multiobjective quadratic assignment problem," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, and K. Deb, Eds. Springer Berlin Heidelberg, 2003, vol. 2632, pp. 295–310. [Online]. Available: http://dx.doi.org/10.1007/3-540-36970-8_21

This document is set in Palatino, compiled with pdfLaTeX2e and BibTeX.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template