Electric/Electronic-Architectures

Automating and optimizing communication matrices

Felix Fikke

Master of Science Thesis

Embedded Software Department
Faculty of Electrical Engineering, Mathematics and Computer Science

Electric/Electronic-Architectures

Automating and optimizing communication matrices

by

Felix Fikke

in partial fulfillment of the requirements for the degree of

Master of Science

in Telecommunications Sensing

at the Delft University of Technology,

to be defended publicly on 22 April 2016 at 14:30.

Supervisor:	Dr. Przemysław Pawełczak	
Thesis committee:	Prof. Dr. Koen G. Langendoen,	TU Delft
	Dr.-Ing. Andreas Kern,	BMW Group

This thesis is confidential and cannot be made public until 22 April 2016 at 14:30

embedded software

The work in this research was supported by BMW Group. Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft University of Technology (eemcs)

The undersigned hereby certify that they have read and recommend to the Faculty of Embedded Software Group Department of Software Technology Faculty EEMCS, Delft University of Technology Delft, The Netherlands www.es.ewi.tudelft.nl for acceptance a thesis entitled

Electric/Electronic-Architectures

by

Felix Fikke

in partial fulfillment of the requirements for the degree of

Master of Science Telecommunications Sensing

Dated: 22 April 2016 at 14:30

Supervisor(s):

Dr. Przemysław Pawełczak

Dr.-Ing. Andreas Kern

Reader(s):

Prof. Dr. Koen G. Langendoen

Table of Contents

Chapter 1

Introduction

Designing a car used to be the job of mechanical engineers, the main differentiating factors of cars used to come with their mechanical properties; the chassis, suspension set-up, engine characteristics, etc. this has seen a mayor shift in the last decade. The level of sophistication of modern cars has reached such high levels that the common consumer has a hard time noticing the differences in the mechanical set-up of vehicles from different competitors [1, 2].

Attention of car manufacturers has shifted to adding features aimed at improving the safety and comfort of the driver and passengers. Spurred on by developments in micro-electronics that allowed the automotive industry to develop sophisticated electronic monitoring, regulating and controlling systems. These systems have evolved to mechatronic systems that are characterized by the integration of components (hardware) and signal-based functions (software) in embedded computer systems called Electronic Control Units (ECUs), capable of performing specific functionality autonomously [3]. Examples of these systems are digitally controlled fuel injection engines, electronic stability control (ESC), anti-lock brake system (ABS) and cruise control systems. At the same time, camera, radar and ultrasonic sensors are used to sense the environment around the vehicle.

Further development of these systems underwent a rapid increase in complexity and diversity. Implementing more and more sensors and processors in all parts of the vehicle, and in the future even considering data from other vehicles through vehicle-to-vehicle communications. The exchanges of information that are necessary to achieve this push the limits of the current day electrical communication systems in cars. And whilst implementing new technologies that allow for high data-rates seems an easy solution, the fixed costs of re-developing sensors and devices to include new standards that have not yet been broadly accepted by the industry are extremely high and risk full. A different approach is to look at the potential of the communication systems currently in place.

1-1 Automotive Electric/Electronic-Architectures

In the early days of the modern vehicle, vehicles contained only a few electrical systems. Most systems essential to the core task of driving the vehicle were done by hydraulics and mechanics. As ever more systems introduced comprised of electronic monitoring, regulating and controlling systems, the cabling required to interconnect these systems quickly increased. This not only lead to increased complexity on a system level, but also to decreased reliability due to the high number of connection points and increased weight and material costs. To handle this increased complexity, manufacturers needed to incorporate efficient network architectures in their design process.

The IEEE holds the following definition of architecture [4]: *"Architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution."*

In the automotive industry the components are mainly the ECUs. The relationships between them are the communication networks, whilst the environment can be seen as the physical placement of the components, defining the cabling requirements, and the power requirements. The design and evolution guidelines describe the process of life-cycle management that the architecture must enable. These combined aspects of the electrical infrastructure in vehicles is called the *Electric/Electronic-Architecture (E/E-A)*.

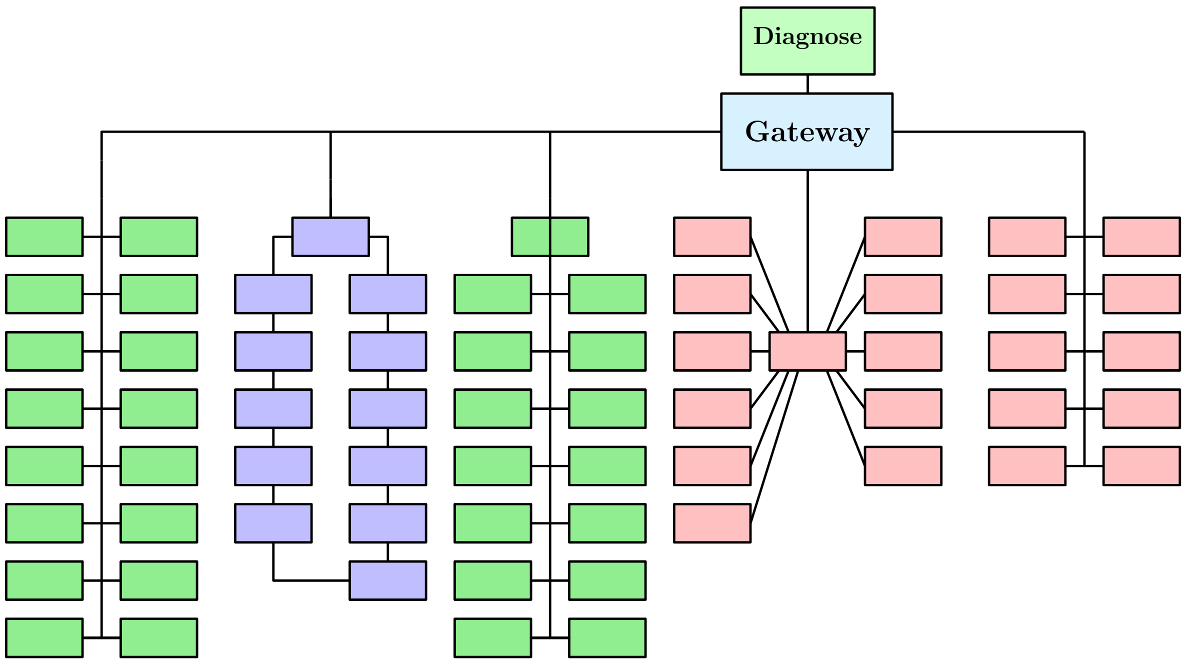


Figure 1-1: Example Electric/Electronic-Architecture of BMW 7-series. [5]

To deal with the rapidly increasing number of cables required to interconnect these ECUs, manufactures introduced serial field buses. By connecting ECUs on these buses, the number of dedicated cables could be drastically reduced; resulting in lower costs and weight of the total E/E-A. This also allowed for sensor data and other calculated values to be shared throughout the electrical systems of a vehicle.

Original Equipment Manufacturers (OEMs) initially started developing their own proprietary buses, suited to their specific needs. However, with the increased number of ECUs being sourced from external suppliers, standards were needed to meet goals related to:

- Cost reduction
- Integration of supplier components in vehicles
- Increased reliability
- Harmonization of tools used in development, diagnostics and after-sales.

Having standards in place also allowed suppliers to develop their own ECUs capable of specific features that they can then offer to OEMs as off-the-shelf products. The high production volumes bring down the costs whilst the broad use of these components throughout the industry ensures the devices are tested to a great extent under diverse conditions, which increases their reliability.

This rapidly lead to an increase in the amount of embedded systems in cars, being connected by a number of different bus technologies. This made efficient communication between multiple ECUs possible, allowing manufacturers to implemented evermore previously mechanical and hydraulic functions by digitally controlled ECUs and introduce new functions that were previously impossible to realize. The amount of ECUs sequentially saw a huge spike in the nineties, as can be seen in figure Figure 1-2. Recent advances in advanced driver assistance features, and increased connectivity of vehicles, have sparked another growth in the amount of ECUs and transmitted signals in vehicles (Figure 1-2).

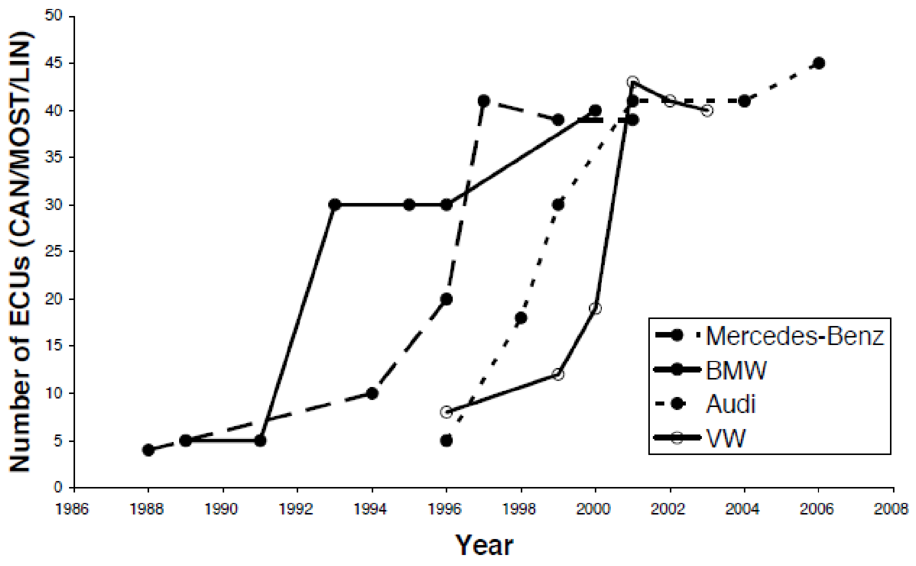


Figure 1-2: Increase in number of ECUs in modern cars. [5]

Dealing with the increased number of ECUs and bus systems has thereby become a mayor concern for manufacturers. Many of the new Advanced Driver Assistance System (ADAS) rely on Real Time Systems (RTS) that need to combine a wide range of sensor data within stringent timing requirements. The E/E-A needs to be designed in such a way that allows these flows of information to be transmitted quickly and reliably over buses with limited capacity.

Chapter 2

Fundamentals

2-1 Current Automotive E/E-Architectures

Modern day premium class vehicles incorporate a vast range of distributed Electronic Control Units (ECUs) that communicate with each other and different components throughout the vehicle. Most of these systems are interconnected via a common network infrastructure, the so-called Electric/Electronic-Architecture (E/E-A) of the vehicle. This architecture describes the physical location, the implemented communication technologies, the power distribution and the function mapping of different components. Commonly used communication technologies in the automotive industry to interconnect ECUs are Controller Area Network (CAN) [6], Local Interconnect Network (LIN) [7], FlexRay [8], Media Oriented Systems Transport (MOST) [9] and Ethernet [10]. To enable cross-network communication, gateways are implemented to exchange information between different communication protocols. The general benefits of using a common E/E-architecture are among others the possibility of reusing sensor data for different applications, the optimization of the wiring harness (for example the avoidance of parallel cabling in the same installation spaces), the simple scalability when adding new functions, and the support for having different expansion stages.

Current developments towards Advanced Driver Assistance System (ADAS) indicate that the number and complexity of such distributed applications will further increase [5] [11]

2-2 Current intra-vehicle communication systems

The amount of signals being transmitted in a modern vehicle have dramatically increased. The recently introduced 2016 BMW 7-Series is a prime example of such a modern vehicle that contains many state of the art in ADAS. This requires up to 50 ECUs, together distributing more than 15.000 values and signals (Figure 2-1) [12]. These signals can contain a wide range of information, that can be divided into two categories.

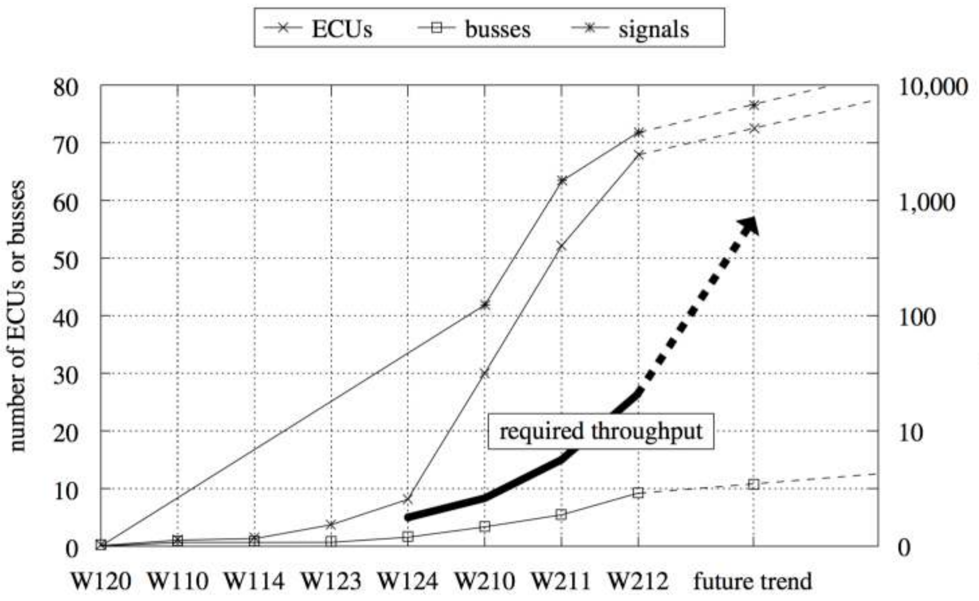


Figure 2-1: The evolution of the Electric/Electronic-Architecture (E/E-A) over Mercedes Benz E-class type series [12].

- Event-triggered: signals that can be send at any given time, usually following the occurrence of a specific event. They are further classified by their time-criticalness. An example of a event-triggered signal is the indicator light. When a driver activates the indicator-stalk, the indicator-lights have to be activated within 200 ms (legal requirement).
- Time-triggered signals have specific timing requirements. These are signals which need to be frequently updated as they change constantly or are critical to time-sensitive calculations. An example of a such a time-triggered signals are the wheel speeds, which are transmitted every 20 ms.

2-2-1 Controller Area Network (CAN)

CAN is an automotive-specific bus standard developed by Robert Bosch GmbH, which was released in 1986 [6, 11]. It defines layer-1 and layer-2 functionality of the Open Systems Interconnection (OSI) network model. CAN is typically used to transmit control traffic between ECUs within the vehicle. It uses twisted pair copper cables at lengths of up to 40 m. Messages are encapsulated in frames with a maximum data field size of 64 bits.

CAN is a broadcast bus which transmits messages in an event-triggered fashion using deterministic collision resolution to control access to the bus (Carrier Sense Multiple Access / Collision Resolution, CSMA/CR). Messages are transmitted in frames containing 0 to 8 bytes of payload data. These frames can be transmitted at speeds of 10 Kbps up to 1 Mbps, although 500 Kbps is the common implementation for non safety-critical x-by-wire systems. Speeds of 250 Kbps are used for control and diagnostics, and 125 Kbps and less are used typically for body and comfort electronics [5].

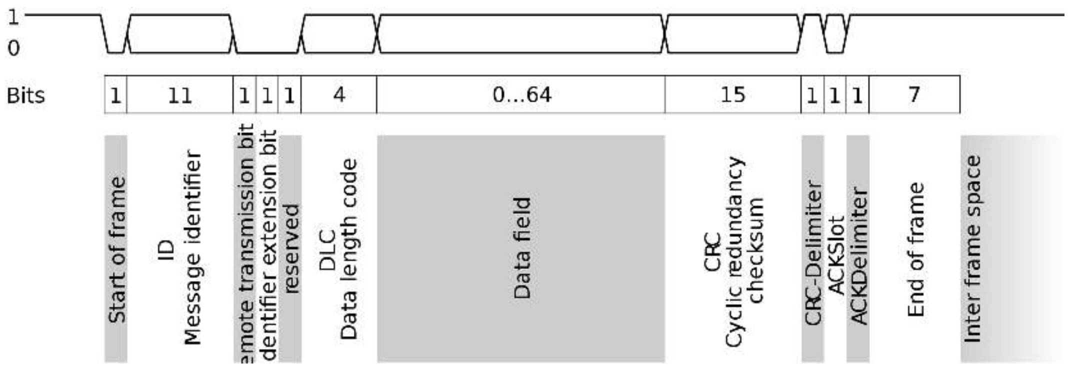


Figure 2-2: CAN Base Frame telegramm according to specification v2.0A [13]

A CAN bus can define up to 2048 unique ID's which are used for arbitration. Each unique frame is capable of transferring an 8-byte payload (Figure 2-2). The CAN-identifier determines the priority of a message, and thus the likelihood of the message successfully winning the arbitration on the medium, signals with stringent timing-requirements should be assigned to Protocol Data Units (PDUs) with high priority.

The bus load is determined by the bandwidth occupied by the signals being transmitted, divided by the total available bandwidth. The total size of a CAN frame consists of the *payload* (the PDU) and the *frame header*. The PDU can contain multiple signals up to a maximum size of 8 bytes (L_i). In addition to the payload there are a number of overhead bits. For a standard CAN message, the header contains 34 bits (g) that are exposed to bit-stuffing[1] and 10 bits that are not. Depending on the amount of stuffing bits needed, the maximum size of a CAN frame (S_{frame}) is given by (2-1) [5].

$$S_{frame} = g + 10 + 8 \cdot L_i \left\lfloor \frac{g + 8L_i - 1}{4} \right\rfloor \tag{2-1}$$

Each signal has a specific cycle time-requirement t_{cycl} that is determined by the engineering team that sets these requirements for each signal an ECU needs to receive or transmit. The signal with the highest cycle-time in a PDU specifies how often that message needs to be transmitted over the bus. The resulting bus load on the CAN bus is given by equation (2-2). Where r_B is the bit-rate of the bus.

$$Busload = \frac{1}{t_{cycl}} \cdot \frac{S_{frame}}{r_B} \tag{2-2}$$

The goal when designing the E/E-A of a vehicle is to keep this bus load as low as possible. As this increases the reliability of the system, whilst reducing the number of CAN buses required, and thus material costs and weight.

[1]Bit stuffing or bit padding technique, forces a complemented bit in the stream after 5 bits of the same type have been transmitted. Stuffing ensures that rising edges are available for on-going synchronization of the network. [14]

2-2-2 Local Interconnect Network (LIN)

LIN [7] is an inexpensive broadcast communication bus developed in the late nineties by the LIN consortium consisting of a number of automotive manufacturers including BMW and Daimler. It arose from a desire for a cheaper alternative to CAN for less important elements of the in-vehicle network [11].

LIN is typically used for infrequently used controls of body and comfort electronics. For instance, electric window control, seat adjustment control, light sensors and climate control. It consists of sub-systems, such as for example the door of a car with all its functionality such as window lifts, door locks etc. These subsystems are interconnected with a control unit that is connected to the vehicles CAN network via a LIN/CAN gateway. This reduces the number of devices that need to communicate with the CAN network directly, eliminating the costs of CAN controllers for each of these devices. LIN complements CAN by being much cheaper and simpler yet supporting the communications needed for typical non safety-related automotive subsystems [5].

LIN is a master/slave time-triggered type of field bus providing network speeds of up to 20 Kbps, where messages are sent in frames containing 2, 4 or 8 bytes. LIN is running on any UART/SCI, using a single wire, and the frame transmission is predictable in terms of timing, providing typical reaction times in the order of 200 ms.

2-2-3 FlexRay

In the late nineties, upcoming applications like x-by-wire systems that aim to replace mechanical or hydraulic systems required higher bandwidth and increased reliability compared to existing bus systems. To develop a suitable network technology BMW and Daimler, together with General Motors, Volkswagen and NXP Semiconductors founded the FlexRay Consortium [8]. FlexRay provides high speed fault-tolerant communications by combining time-triggered TDMA and the event-triggered friendly FTDMA.

For the physical layer, electrical and optical solutions are available relying on either single or dual channels, forming either a passive bus topology, or an active star topology. Using single channels between the nodes reduces the amount of wiring needed and therefore also the cost of the automotive system. The dual channel configuration can be implemented for increased fault tolerance in critical X-by-wire systems.

Messages are sent in frames containing 0 to 254 bytes of payload data and 5 bytes of header. Frames can be either statically scheduled with bounded communication latency, or they can be dynamically scheduled. Ding et al [15] have proposed a Genetic Algorithm (GA) based algorithm, optimizing the static scheduling of the FlexRay.

FlexRay provides network speeds of up to 10 Mbps and is the de-facto communication standard for high-speed automotive control applications interconnecting ECUs in modern automotive systems with special interest high-speed safety-critical automotive systems [5].

2-2-4 Media Oriented Systems Transport (MOST)

MOST was developed primarily to support the transmission of multimedia data. The maximum possible bandwidth as defined by the MOST150 standard is 150 Mbps, which makes it much more suitable than CAN for multimedia data transmission. It uses polymer optical fibers as communication medium, and supports interconnection of up to 64 nodes. While the MOST Cooperation published the MOST specification [9], details relating to the data link layer (OSI layer-2) are only available on a royalty basis.

2-2-5 Ethernet

Ethernet [10] is a widely used communication bus, especially for Internet due to its low cost, speed, and flexibility. A benefit of using Ethernet in vehicles lies with the increased bandwidth it offers. Technologies specifically developed for the automotive domain such as CAN and MOST are very well suited for intra-vehicle communications, but lacked the flexibility to scale with increasing bandwidth demand. Previously low-bandwidth control applications have turned into complex real-time ECUs. Ethernet however, has the benefit of having matured due to the massive use in other industries. It offers different throughputs from 10 Mbps up to 40 Gbps for non-automotive applications and packet sizes ranging from 64 bytes up to 1500 bytes.

Ethernet has seen its first implementation in vehicles for interfacing with diagnostic equipment. Due to its increased bandwidth with respect to CAN bus interfaces, diagnostic operations like flashing new firmware's can be achieved in fraction of the time previously needed. Using a CAN based network, the process of flashing a new firmware takes up to 10 h when flashing an 81 MB firmware update. Using an Ethernet network and a much larger 1 GB update, this procedure takes 20 min [16]. Allowing for significant reduced turnover time of vehicle maintenance services.

The increase in ADAS that merge data from a number of high-bandwidth sensors such as 24 GHz short-range or 77 GHz long-range RADAR sensors, ultrasonics, infrared cameras, and RGB optical video cameras [17] show the potential of Ethernet as a serious network solution for the automotive industry.

2-3 Wireless communication systems

Wireless communication systems are already present in modern vehicles. Apart from several wireless technologies to communicate with with consumer devices over GSM/3G/4G, Bluetooth and WiFi, radio frequency is used in the case of Tire Pressure Monitoring System (TPMS) to communicate with pressure and temperature sensors mounted in the tire [18]. For additional implementations of wireless interconnection of sensors and ECUs, Ultra-wideband and IEEE 802.x based solutions are being investigated [19, 20]. The main benefit of wireless systems is that they alleviate cabling requirements. As modern day vehicles carry kilometers of cabling weighing up to 30 kg [21] each cable saved is welcomed. However, this advantage is limited as each ECU or sensor still needs an electrical power source. Another vital requirement in the automotive domain is high reliability. As the wireless medium is often unpredictable in terms of the temporal behavior of message transmissions, the temporal guarantees that can be provided with a wireless network are usually not as reliable as for a wired link. Especially interference from other sources than the communications network itself are unpredictable. Wireless systems provide another vulnerability, as the need for a physical connection to the vehicles wiring harness is removed, the system is more susceptible to external manipulation. As Rouf et al. [22] have shown for the TPMS by demonstrating a method of eavesdropping, reverse engineering and injecting false data in a moving vehicle. This raises concerns about security in wireless networks, due to the absolute need for reliability and security in safety-critical systems, wired solutions are expected to dominate for the foreseeable future [11].

Chapter 3

Optimization of E/E-Architectures

3-1 Motivation of optimization

As we have seen in section 2-2, a number of different communication technologies are implemented in the vehicle. Each sending and receiving data in their specific manners. There are however a few commonalities throughout the different technologies. Despite their different frame structures, each has a specific payload of data in the data link layer it ultimately transmits. The contents of this Protocol Data Unit (PDU) range from measured data to calculated values and are commonly referred to as *signals*.

During the design process of an Electronic Control Unit (ECU), it is determined which specific signals it will need to receive from different ECUs, and which signals it will need to transmit as they are required by other ECUs. The data size of the signals differs, as does the size of data a PDU can contain for each communication technology. In general, however, a multiple of signals can be included in a PDU. As only one ECU is capable of filling a PDU with signals before the message is sent on its respective bus, deciding which signal is assigned to which PDU is evaluated per ECU for each signal it has to transmit. These assignments are stored in communication matrices that are flashed into the ECUs. The signals have several properties that influence the choice of PDU it is assigned to:

- Data-size
- Event-triggered or cyclic
- Timing-requirements
- List of senders
- List of targets

The procedure used to assign the signals to ECUs determines the performance of the resulting Electric/Electronic-Architecture (E/E-A). This is best illustrated by an example.

3-1-1 Example of a signal assignment

Let us consider 4 ECUs that share a common communication bus, Figure 3-1 shows a possible architecture connecting these ECUs using a Controller Area Network (CAN) bus.

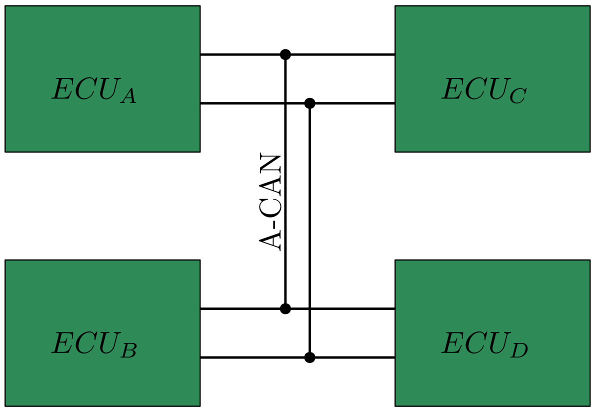


Figure 3-1: Example of a simple E/E-A.

Taking a closer look at ECU_A, Table 3-1 shows an example communication matrix that lists the signals this ECU needs to transmit, along with the respective receiving ECUs and timing-requirements. These signals will have to be assigned to PDUs with a capacity of 14 *bit*, which can be done using different strategies.

Signal	Receiver	Size (bit)	t_{cycl} (ms)
S_1	ECU_C	4	20
S_2	ECU_B	6	90
S_3	ECU_D	4	400
S_4	ECU_B	8	80
S_5	ECU_D	6	300
S_6	ECU_B	2	200
S_7	ECU_D	8	40
S_8	ECU_C	2	100
S_9	ECU_B	6	300
S_{10}	ECU_C	4	100

Table 3-1: Signal list of ECU_A.

Next we will evaluate three different examples of strategies to assign these signals to PDUs and evaluate the differences based on a set of metrics.

Strategy 1

The most straightforward way of assigning the signal is simply iterating all signals and add them to the first possible PDU that has enough space to contain the signal. This strategy is more or less comparable to the actual method applied today where signals are assigned to PDUs grouped by similar function. This would result in the assignment visualized in Table 3-2a.

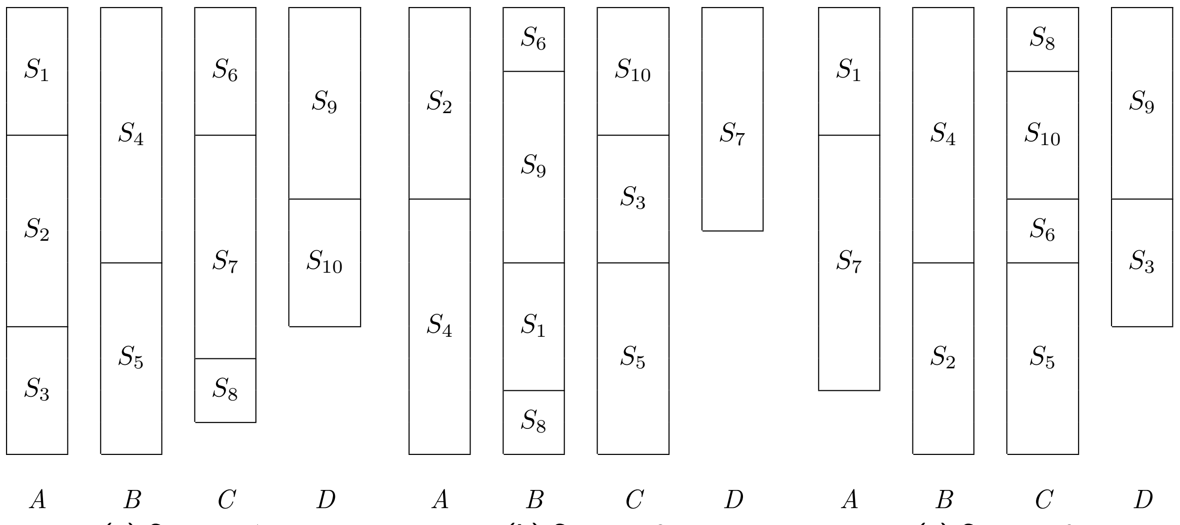
Strategy 2

Another strategy would be to group the signals according to their respective receivers. This has the advantage of minimizing buffer usage at the receivers as all relevant signals are received at once instead of spread out over multiple messages. This assignment is shown in Table 3-2b.

Strategy 3

The third example we will evaluate will assign signals based on their timing-requirements. This has the advantage of reducing the amount of unnecessary re-transmissions of signals. As the frequency with which PDUs need to be transmitted is dependent on the signal with the most stringent timing-requirement. An example of this strategy is shown in Table 3-2c.

Table 3-2: Assigning signals to PDUs with different strategies.



(a) Strategy 1 **(b)** Strategy 2 **(c)** Strategy 3

The specific signal assignment determines several metrics influencing the overall performance of the system. Such as the:

- Number of PDUs needed
- Fill-ratio the PDUs
- Generated bitrate

The amount of PDUs needed to transmit a certain set of signals is an important as some of the bus systems used in E/E-A have only a limited number of available PDUs. Unused PDUs allows the system to be ready for future addition of new signals. A high fill-ratio of PDUs improves the payload ratio of a packet and minimizes per packet overheads. The bitrate needed to transmit the signals is ultimately the metric with the highest impact, as the bitrate is in most cases the limiting factor of the system.

Looking at the three examples of signal to PDU assignments, we can evaluate the effects these different strategies have on the overall performance of the system. Table 3-3 shows the results of the before mentioned metrics[1] for each strategy.

Strategy	Nr. PDUs	Avg. fill ratio	Generated bus load
1	4	0.89	74%
2	4	0.89	100 %
3	4	0.89	55 %

Table 3-3: Evaluation of strategies shown in Table 3-2.

The results show that even with a very small set of signals in seemingly similar assignments, the required bus load to transmit these signals, varies significantly.

3-2 Significance of signal to PDU optimization

Until now, the process of assigning signals to PDUs is done manually, whereby the signals are often grouped together in PDUs according to their function. This leads to a mapping that is understandable and easily traceable for engineers needing to work with these signals, either to change assignments, or to determine which messages they need to listen to for their functions. This method however, in no way guarantees optimal performance of the discussed metrics. Which is troublesome since the increasing number of ECUs and new Advanced Driver Assistance Systems (ADAS) not only cause an increase in the number of signals being transmitted, their timing-requirements have also become more stringent. As more and more signals are transmitted ever more frequently, the loads on the communication buses increase. Studies and tests conducted by BMW have shown an increased rate of erroneous transmissions with higher bus loads. With bus loads over 50%, the reliable transmission of signals can no longer be guaranteed [23, 24]. This can lead to anything from severe system failures to unexpected behavior for the customer, both highly undesirable. To prevent this, the E/E-A is designed not to exceed certain thresholds of bus loads.

Limiting the maximum bus load also limits the amount of signals (and thus customer features) that can be implemented. Therefore reducing the bus load is an important field of interest within the topic op optimizing the E/E-A.

3-3 Optimizing the signal to PDU assignment

As shown in subsection 3-1-1, the manual assignment of signals to PDUs results in a sub-optimal assignment. A lot of PDUs have unused space and contain signals with different timing-requirements. Using all of the space a frame can contain is important in lowering the bus load, as well as matching signals with similar timing-requirements:

[1]PDU capacity of 14 bit, packet overhead of 2 bit, bit-rate 2 Kbps.

- A fully 'filled' PDU results in a lower overhead to payload ratio.
- Reducing the total number of messages needed to transmit, allows for additional signals to be sent over the same bus.
- Grouping signals with similar timing-requirements prevents unnecessary retransmission of signals.

If we take a look at an average ECU, that has to transmit a total of 300 signals with an accumulated size of 2300 *bit* on the CAN bus. These signals need to be divided over PDUs of 64 *bits*. Ideally this would require only 36 PDUs ($2300/64 = 35.94$). The number of different ways a PDU can be filled is given by the Partition Function $P(n) = P(64) = 1741630$. The total number of possible partitions is $1741630^{36} = 4.73 \cdot 10^{224}$. We can see that finding the optimal solution within this huge search space is a laborious task. If we take a step back to analyze a generalization of this problem, we can see why.

The process of optimally filling a PDU of size c with signals of size s_i is an equivalent of the Subset sum problem [25], which is defined as follows:

- Given a sequence of integers $a_1, \ldots, a_n$ and a parameter c,
- Decide whether there is a subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} s_i = c$

The Subset Sum Problem is a well known true decision problem that is NP-complete [26]. It is also a special case of the Bin Packing Problem (BPP).

3-4 Bin Packing Problem

The Bin Packing Problem (BPP) is a combinatorial optimization problem that is part of a large family of cutting and grouping problems, which consist in dividing sets of elements into separate subsets. It is a widely researched topic as it appears as a (component) part of many more complicated real world problems such as distribution of resources, operation scheduling, cutting materials, etc. A study analyzing the requests made to the Stony Brook Algorithm Repository showed that out of 75 algorithmic problems, the BPP ranks 3rd in the list of most needed algorithm implementations, after kd-trees and suffix-trees [27].

The BPP is defined as follows. A finite set of n items is to be packed into containers/bins with capacity c. The number of available containers is assumed to be unlimited, each with capacity $c > 0$. The size of items $i \in \{1, \ldots, n\}$ is $s_i > 0$, as shown in (3-1):

$$\forall k : \sum_{i \in bin(k)} s_i \leq c \tag{3-1}$$

The goal of the one-dimensional BPP is to minimize the number of containers needed for packing all n items [28, 25, 29]. The theoretical optimum number of bins used for any problem instance is given by (3-2). Were the accumulated size of all signals, subtracted from the accumulated capacity of bins is less than the single capacity of a bin.

$$opt = \left\lceil \left(\sum_{i=1}^{n} s_i \right) \Big/ c \right\rceil \tag{3-2}$$

As the BPP can be reduced from the Subset Sum problem, which we have seen in section 3-3 to be a NP-complete decision problem, this naturally gives rise to the associated NP-hard optimization problem [30, 31, 32], requiring extensive resources to compute.

3-4-1 Solving the Bin Packing Problem

In this section, we review some of the important solution approaches to the classical BPP, which form the basis for our method of solving the assignment of signals to PDUs. There are three main approaches to consider:

- Exact methods can typically only be applied to small instances of the BPP do to the exponential increase in computing power needed for large instances.
- Heuristics are usually general rules of thumb that, depending on the characteristics of the problem, provide good solutions with regards to quality and computation time.
- Metaheuristics are highly tailored hybrid algorithms that describe general search principals rather than specific rules. They typically combine mechanisms that diversify the search space, and intensification mechanisms that exploit previously found solutions [33]. Some Metaheuristics are inspired by optimization processes in nature, such as Genetic Algorithm (GA) and Ant Colony Optimization (ACO). These algorithms have proven to be amongst the most successful approaches to solving the BPP [34, 35, 36].

One-dimensional BPPs have been studied extensively, resulting in numerous heuristics and approximate methods for solving BPP. First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) are the two of the best known heuristics used to solve BPP, developed by Coffman et al. [31]. In these algorithms items are processed in decreasing order of size. The items are then assigned to the first bin that has a high enough residual capacity to hold that item in FFD, or to the bin that results in the smallest possible residual capacity in BFD. Johnson et al. [37] showed the asymptotic worst case performance bounds for FFD and BFD to be $11/9\,Opt$. These algorithms can be implemented in $O(n \log n)$ time.

Martello and Toth introduced a number of fundamental heuristics for solving BPPs, such as the reduction procedure (MTRP) and an exact an enumerative (branch-and-bound) method (MTP) [25]. MTP is one of the earliest methods for solving the one-dimensional bin packing problem. It is often used as a basic reference in comparative BPP studies. The MTP algorithm attempts to find bin assignments dominating others. After such a bin is found, the problem is reduced by removing the dominating bin. In order to prevent an exponential search, only dominating bins of at most three items are taken into account [38]. Making MTP less suitable for applications with large numbers of smaller items.

Gupta and Ho [28] introduced the Minimum Bin Slack (MBS) heuristic. MBS is bin-focused, at each iteration, an attempt is made to find a subset of the items that fills a bin whilst generating the least possible residual capacity.

Although BPPs have been extensively studied in the past, resulting in the approximation methods discussed above, studies have mainly focused on the classical single-objective one-dimensional BPP. The methods discussed earlier are all aimed at improving the accuracy and efficiency of solving the specific problem that only focusses on packing items in the least amount of bins. However, many real-world problems tend to have additional, sometimes even conflicting, objectives. Solving these so called Multi-Objective Combinatorial Optimization Problems (MOCOPs) requires finding several solutions that represent an optimal combination of the different objectives. Despite methods aimed at solving MOCOPs exactly, due to the inherent $\mathcal{NP}$-hardness of the majority of MOCOPs [39], exact solutions can only be applied to small-scale problems. For more complex problems, metaheuristics are required to find a solution in reasonable time. [40, 41].

3-4-2 Multi-Objective Combinatorial Optimization Problem (MOCOP)

A MOCOP is defined as:

$$\begin{aligned} &Optimize\ F(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \\ &Subject\ to\ :\ \ x \in \mathcal{D} \end{aligned} \tag{3-3}$$

Where $F(x)$ is the m-dimensional objective vector $f_i(x)$ is the i^{th} objective to be optimized (either minimized or maximized), $\mathcal{D}$ is the feasible solution space.

In contrast to single-objective optimization problems where only one objective needs to be optimized, to solve a MOCOP we need to find a suitable trade-off between the multiple objectives $f_i(x)$. Therefore, there exist no unique optimal solution but instead, a set of solutions representing the best possible trade-offs among the objectives. Such solutions are contained in the Pareto optimal set (PO). When plotting the objective function values corresponding to the solutions stored in the Pareto optimal set, we obtain the Pareto front of the problem. The objective of the NSGA algorithm is to improve the adaptive fit of a population of candidate solutions to a Pareto front constrained by a set of objective functions. The algorithm uses an evolutionary process with surrogates for evolutionary operators including selection, genetic crossover, and genetic mutation. The population is sorted into a hierarchy of sub-populations based on the ordering of Pareto dominance. Similarity between members of each sub-group is evaluated on the Pareto front, and the resulting groups and similarity measures are used to promote a diverse front of non-dominated solutions [42, 43].

3-5 Introduction to Genetic Algorithm

The Genetic Algorithm (GA) (see [44, 45, 46] is a metaheuristic that tries to find solutions to optimization and search problems. Guided by the principles of natural evolution, GAs attempt to find better solutions by evolving existing ones. Similarly, to the process of natural selection and genetics, desirable features from a generation are selected and combined to form the next generation.

Belonging to the wider group of Evolutionary algorithms (EA), GAs came into focus after the studies John Holland performed of cellular automata, where he introduced the Holland's schema theorem, also called the fundamental theorem of GA [45]. Which states that above

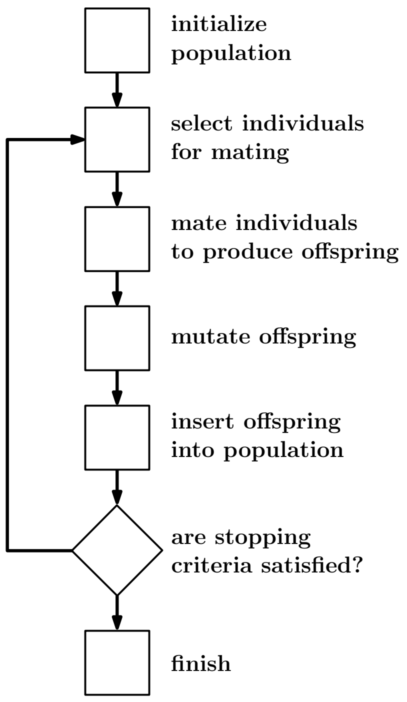


Figure 3-2: Standard GA process

average fitness schemata grow exponentially, and below average fitness schemata decay from generation to generation. This means that while exploring the search space, a member of an above-average schema is likely to produce off-spring of above-average fitness.

Figure 3-2 shows the basic steps of an GA. Exploration of the search-space is done by generating a set of solution candidates (individuals) that form the population. The quality of each individual is measured by a fitness function. Individuals are selected to be used to generate the next generation of individuals. This is done using a weighted probability function, to increase the chance of evolving good candidates. Genetic operations are applied to the selected individuals, such as recombination and mutation. Recombination takes multiple individuals (parents) to generate new solutions (offspring) with the goal that favorable properties from both parents are transferred to the new individuals. The mutation process randomly alters part of an individual.

Since GAs are inspired by biological genetics, most of the terminology comes from that field as well. In the context of GA, individuals are called *genotypes*, whereas the solutions that are encoded by individuals are called *phenotypes*. This is to differentiate between the representation of solutions and solutions themselves. A genotype contains a collection of *genes* that represent specific features. Genes have a specific location on the chromosome called *locus*, the value of the gene is called *alleles*.

3-5-1 Encoding

To apply the GA to a particular problem, the first step is to is to devise an encoding scheme to map a possible solution (phenotype) to a genotype (individual). The technique used to to this has a mayor influence on the effectiveness of the GA. Encodings should have the following properties [47]:

- Embody the fundamental building block that is significant to the problem [46]
- Is amenable to genetic operators that can propagate these building blocks from parents' genotype to offspring genotype [48]
- Allow a tractable mapping to the solution (phenotype) being encoded.

The most common representation techniques use n-bit binary strings $S \in \{0,1\}^n$. Where n represents the number of variables of a problem. This classic representation form is named after John Holland [44].

j	1	2	3	4	5	...	$n-1$	n
$S[j]$	1	1	0	1	0	...	0	1

Table 3-4: Holland-style binary representation.

Example of Problem: Knapsack problem
The problem: List of items with given value and size. The knapsack has given capacity. Select subset of items to maximize the value of items in the knapsack, without exceeding the capacity of the knapsack.
Encoding: Each bit says, if the corresponding item is in the knapsack.

When initializing the GA, the first generation of individuals is generated randomly. Depending on the characteristics of the problem, not all generated genotypes necessarily represent valid solutions. For instance, in the example of the Knapsack problem, there exist genotypes that represent solutions with more items in the knapsack than the capacity constraints allow for. There are a number of ways [49] to deal with invalid genotypes:

- using a representation that can only contain feasible solutions
- implementing a repair operator that transforms infeasible solutions to feasible solutions
- disregarding infeasible solutions that have been generated, in the selection process.
- applying a penalty in the evaluation process for infeasible solutions, decreasing their chance of 'surviving' to the next generation. [46]

The first option of limiting the genotype to feasible solutions, is not always possible. Repairing each invalid genotype can be a tedious effort for some problems, such as packing problems, whereby the job of fitting items in containers is actually the inherent problem the GA is trying to solve. The most straightforward approach is to just disregard any infeasible solution, limiting the search to only the feasible solution-space. However, applying a penalty term to infeasible solutions instead of disregarding them completely, can have its advantages too. It assures that those infeasible genotypes are less likely to be considered in the selection process. But genotypes with small 'overshoots' can still be accepted, allowing to guide the search for the optimal solution from two sides, which can increase the effectiveness of the algorithm. A dynamically increasing penalty-factor will purge all infeasible solutions with increasing generations.

3-5-2 Selection

After the first generation is created, the selection process evaluates all genotypes in the population using a fitness function. This fitness function depends on the specific problem, it could be a measure of the average fill of a bin, or for the knapsack problem, it would be a function measuring the value of the Individuals in the knapsack. After evaluating all the genotypes in a population, a proportion of the population is chosen to act as parents for the next generation. The most common selection method is the roulette-wheel selection [50, 44]. Whereby each individual is assigned a selection probability, proportional to its fitness. Suppose f_i is the fitness of an individual i, the chance it will be selected is $P[i] = f_i \Big/ \left(\sum_{j=1}^{n} f_j\right)$. Analogue to a roulette wheel that is being spun, with some individuals occupying more space on the wheel, making them more likely to be selected. Other selection strategies are stochastic universal sampling, tournament selection and rank-based selection [50].

3-5-3 Crossover

Having selected good candidates from a population, the crossover operation is used to invoke heredity. Favorable features, that have been selected from a previous generation, need to be transmitted from the parents to their offspring. During this process, offspring genotypes will become predisposed to the characteristics of its parents. As in nature, this involves an partial exchange of the parents' chromosomes.

The exact procedure of the crossover process, varies based on the type of encoding used. Many different crossover operators exist as two-point, uniform, partially mixed, and uniform order-based crossover. For the general case of a Holland style encoding an example one-point crossover is shown in Figure 3-3. In this case the operator randomly and uniformly selects an integer k between 1 and the genotype length less one $[1, l-1]$. Two new strings are created by swapping all chromosomes between positions k and l.

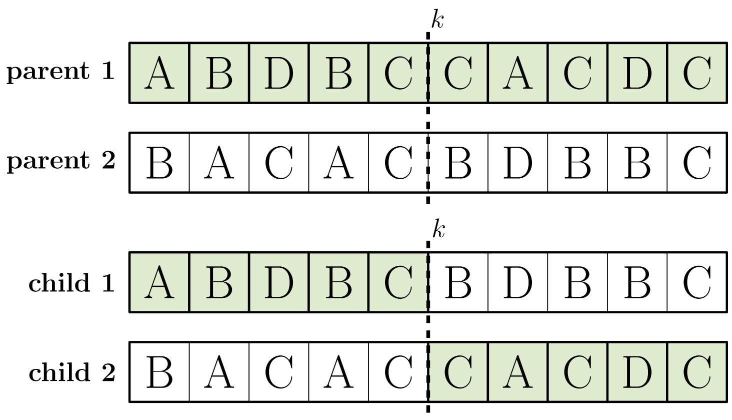


Figure 3-3: One-point crossover with Holland encoding.

Not all genotypes go through the crossover process, a small number (usually $2-5\%$) is excluded in order to preserve a possible 'elite' genotype.

3-5-4 Mutation

The final operator in the genetic algorithm is the mutation algorithm. The effect of mutation is to reintroduce divergence into a converging population. After a number of generations, the GA may be converging upon a local maximum. By mutating some chromosomes, it is possible to find a way past this local maximum. The biological inspiration behind this operator is the way in which a chance mutation in a natural chromosome can lead to the development of desirable traits which give the individual an advantage over its competitors. [51]. The mutating operator simply tosses a biased coin with very small probability P_{mut} at each bit and, according to that result, changes a 1 into a 0 and vice versa, as shown in Figure 3-4.

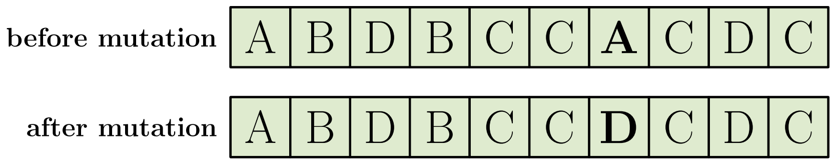


Figure 3-4: Mutation operation.

Chapter 4

Concept of optimizing E/E-Architectures using GA

Improving the Electric/Electronic-Architecture (E/E-A) by optimizing the signal to Protocol Data Unit (PDU) assignments is a complex $\mathcal{NP}$-Hard Bin Packing Problem (BPP) problem. In this chapter, several solutions are proposed based on the implementation of a Genetic Algorithm (GA). Firstly, the design choices of the algorithms will be explained, followed by their implementation and an overview of the results and a comparison of the algorithms and other heuristics.

In general, a GA has five basic components. All influencing the overall quality and efficiency of the GA:

1. An encoding method, which determines the translation of an Electronic Control Units (ECUs) communication matrix to a genetic representation (genotype) that can be input in the GA.
2. A way to create an initial population of individuals.
3. An evaluation function to determine the quality of the generated genotypes, and a selection mechanism.
4. Genetic operators (crossover and mutation) that alter the genetic composition of offspring during reproduction
5. Values for the parameters of the GA.

These components will be described in the following sections.

4-1 Choosing an encoding method

A possible solution to the problem has to be encoded into a genotype that serves as an input in the GA. This genotype travels through the various steps of the algorithm. Choosing the right encoding is critical to the quality of the result of the GA. The encoding that will function as the input to the GA should adhere to several properties, as described in subsection 3-5-1. The meaningful „building blocks” of the genotype should therefore encode the assignment of signals to a PDU.

Application of the classical Holland style encoding scheme is the first route that has been taken in the GA literature when dealing with grouping problems [30, 52, 53]. In our case, this leads to a genotype that has one gene for each signal an ECU transmits, describing the PDU that signal is assigned to. Figure 4-1 shows an example genotype using this encoding scheme and the example communication matrix of subsection 3-1-1.

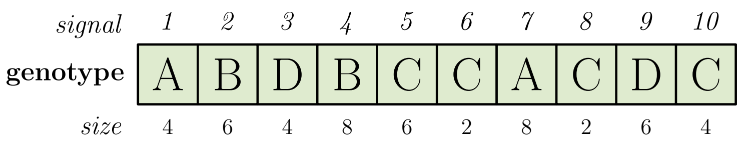


Figure 4-1: Holland-style encoding of Strategy 3 (see 3-1-1).

Despite the wide use of the Holland encoding scheme, it has some significant drawbacks [29, 30]. These can be explained when looking at Radcliffe’s six design principles for constructing useful representations [54] - each candidate solution in the search space (all valid signal to PDU assignments) should be represented by as few distinct chromosomes as possible (ideally exactly one), in order to reduce the size of the space the GA has to search. We can see that the Holland encoding does not adhere to the principle of minimal redundancy in the case of grouping problems. This is illustrated by an example of two encodings:

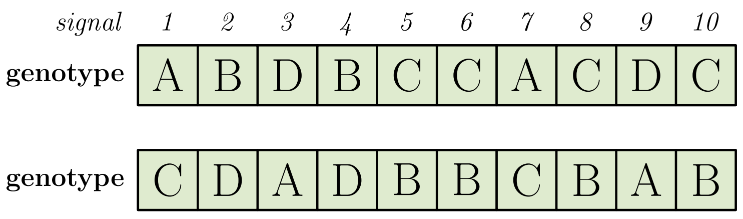


Figure 4-2: Redundancy in Holland encodings.

Both genotypes encode the same solution, namely that where the first and eight signal are assigned to one group, signals two, three and seven to another group, signal four to a single group, etc. Thus the encoding is highly redundant, since the fitness function of a grouping problem depends only on the grouping of the items, rather than the naming of the group. The degree of redundancy of this encoding grows exponentially with increasing problem size and thus increasing number of groups. This problem is called degeneration [54] which leads to inefficient coverage of the search space where the same configuration of groups is explored repeatedly. The minimization of degeneracy is a key component of a good design and improves the performance of the GA [30, 54].

The Holland style encoding creates another drawback; it casts context-dependent information out of context under the standard crossover. Generated offspring looses their meaning with respect to the problem to solve. Taking part of the genotype and combining it with another results in genotypes that have nothing in common with the parents and lose the knowledge with respect to the solution. This can be seen by revisiting the example of subsection 3-5-3. Figure 4-3 shows the same crossover process with the signals respective sizes listed. We recall that the maximum capacity in this example was 14. Both of the parent genotypes only contain groupings $S_{PDU} \leq 14$, however, group B in child 1 and group A and C in child 2 exceed the bin capacity.

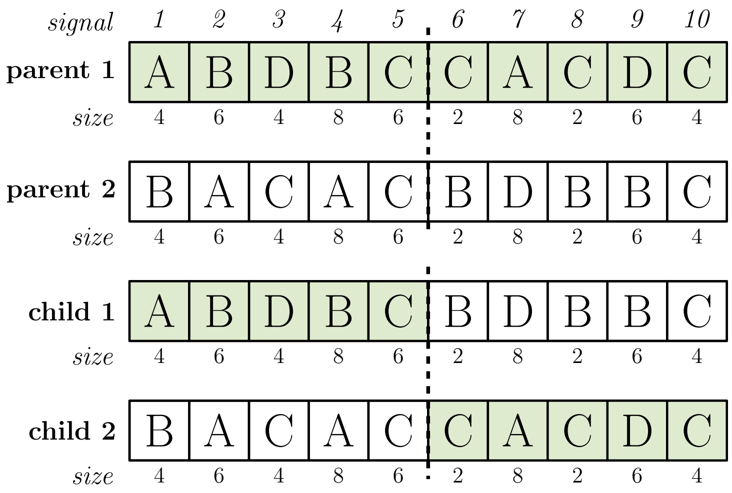


Figure 4-3: Holland style crossover generating invalid results.

Despite the straightforwardness of the Holland style encoding, making it easy to implement, the mentioned drawbacks keep this strategy from reaching an optimal algorithm and result. The reason for this is that the structure of this encoding is item oriented, whilst in grouping problems the groups are the meaningful building blocks. An encoding better suitable for the BPP was proposed by Falkenauer [30] that makes the actual grouping of the signals become the genes of the genotypes. This is done by augmenting the standard genotype by a grouping part, which encodes the groups on a one gene for one group basis.

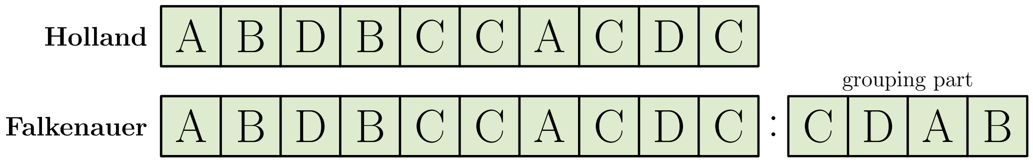


Figure 4-4: Falkenauer's genotype.

The important difference is that the genetic operators will work on the group part of the genotypes, the standard item part of the chromosomes merely serves as a lookup-table identifying which signals actually form which group. This changes the requirements to the genetic operators as well, as they also have to be adapted from being item based to working with groups. Starting with the fact they will have to be able to handle genotypes of variable length.

The crossover process is more extensive than for the Holland style encoding [30], it consists of the following steps:

1. Select at random two crossing sites, delimiting the crossing section, in each of the two parents.
2. Inject the contents of the crossing section of the first parent at the first crossing site of the second parent. As the crossover works with the group part of the genotype, this

means that complete groups of one genotype are added from the first parent into the second.

3. Since the groups added contain signals already contained in that genotype, the original groups containing these duplicates are eliminated, giving way to the 'new' injected groups. This however, also deletes some items not contained in the 'new' groups.
4. Reinsert these items to existing into the groups by using heuristics. Falkenauer suggests First Fit Decreasing (FFD) for the reinsertion of the remaining items [30]. In this study, we propose a hybrid variant of the FFD, adapted to better suit the hard constraints and the multi-objective cost function to optimize for this problem.
5. Apply the points 2. through 4. to the two parents with their roles permuted in order to generate the second child.

The difference of this approach is that the focus of the operator lies with promoting promising groups by inheritance, as is the idea behind the GA. It also adheres to Radcliffe's design principles. Figure 4-5 shows the crossover process referencing the steps described above. In the example, two crossover points are selected in the parents. The groups between these points in parent 1 ('A'), are then added at the place of crossover point 1 of parent 2. Group 'A' consists of signals 1 and 7, these signals are assigned to groups 'b' and 'd' in parent 2, therefore these groups are removed in step 3. This orphans signals 3, 6, 8 and 9. In step 4, these signals are reinserted in the solution.

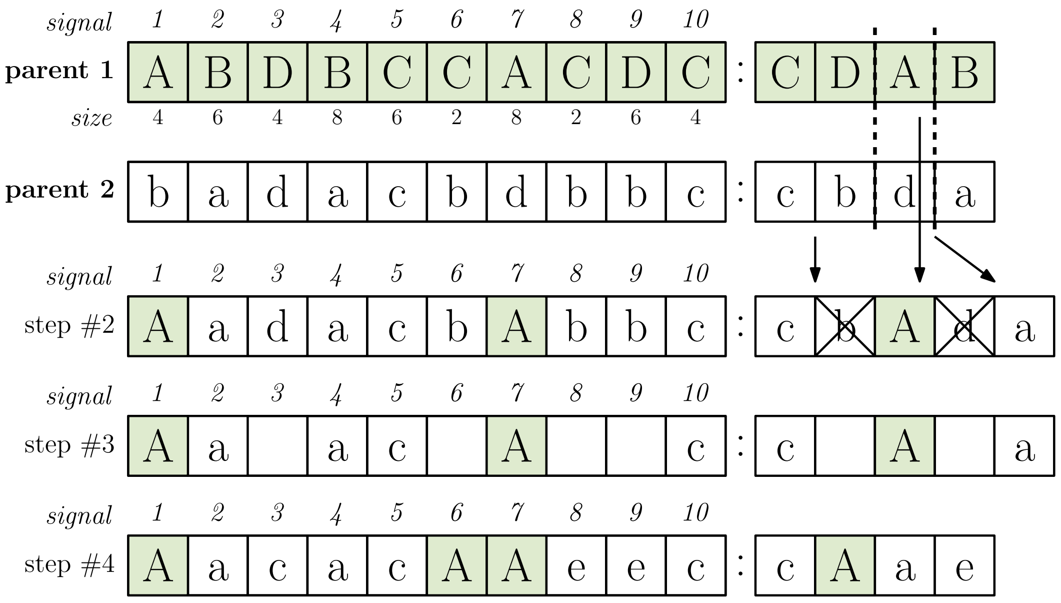

signal
1
2
3
4
5
6
7
8
9
10
parent 1
A B D B C C A C D C : C D A B
size
4 6 4 8 6 2 8 2 6 4
parent 2
b a d a c b d b b c : c b d a
step #2
A a d a c b A b b c : c b A d a
step #3
A a a c A c : c A a
step #4
A a c a c A A e e c : c A a e

Figure 4-5: Crossover process for Falkenauer genotype.

4-1-1 Reinserting items

Falkenauer suggests using the FFD, and others have suggested a range of heuristics for reinserting orphaned items back into groups. However, this leads to the same problem already discussed in subsection 3-4-1. Namely, these heuristics are very good at optimally filling groups with items (the essence of the BPP), however they loose their effectiveness when dealing with Multi-Objective Combinatorial Optimization Problems (MOCOPs).

At the point of the Falkenauer crossover however, only a small number of orphaned items need to be reinserted into bins, dramatically reducing the the search space compared to the original problem. This opens up the possibility that heuristics can be used for this task. Heuristics such as the FFD that are actually aimed at single-objective problems. Algorithm 1 shows the the complete Falkenauer crossover operation with FFD in pseudo-code.

Algorithm 1 Falkenauer crossover with FFD.

```
Require: $p_1 = Parent1$
Require: $p_2 = Parent2$
 1: Select two crossover points $c_1$, $c_2$ randomly in $p_1$
 2: Select two crossover points $c_3$, $c_4$ randomly in $p_2$
 3: Insert groups between $c_1$ & $c_2$ at $c_3$ in $p_2$
 4: Insert groups between $c_3$ & $c_4$ at $c_1$ in $p_1$
 5: Remove original groups that contain duplicate signals
 6: $s$ = Unassigned signals sorted according to decreasing size
 7: while $s$ do
 8:   for $i := 0$ to s do
 9:     for each group in p1 do
10:       if size($s_i$) < space(group) then
11:         Insert $s_i$ in group
12:         Remove $s_i$ from $s$
13:       end if
14:     end for
15:   end for
16:   if $s$ then
17:     Add new empty group
18:   end if
19: end while
```

4-1-2 Mutation

The mutation operator must also be adapted to work with groups rather than items. For Falkenauer's genotype encoding, it inserts new characteristics into the population to enhance the search space by diversification. The mutation operator works on a single genotype that is mutated with very small chance P_{Mut}. In case of mutation two groups that are randomly selected from the genotype (bins A and D are selected in the example given in Figure 4-6). The items of the selected groups are removed from the genotype and new groups are built by reinserting these items similar to the crossover operation. In this example, items 1, 4, and 5 are reinserted to a new group, improving the five-group genotype to a four-group genotype. The details of the operation can be seen in Figure 4-6.

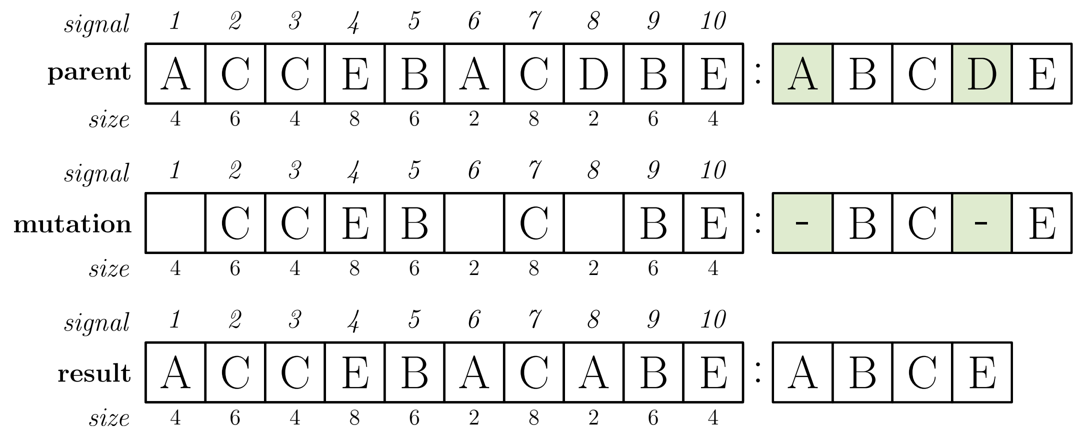

signal
1 2 3 4 5 6 7 8 9 10
parent
A C C E B A C D B E : A B C D E
size
4 6 4 8 6 2 8 2 6 4
mutation
C C E B C B E : - B C - E
result
A C C E B A C A B E : A B C E

Figure 4-6: Mutation process for Falkenauer genotype.

4-2 Selection

After generating the initial population, and after each reproduction phase, genotypes have to be selected that are suitable to act as parents for future generations. The goal of the BPP is to pack all groups as full as possible. This is one criteria to judge the genotypes on. For the specific problem of signal to PDU assignment, we have discussed several specific criteria besides the fill-rate (see subsection 3-1-1). Most importantly the bitrate (and thus bus load) the configuration requires.

Important for the selection metrics applied to the GA is their ability to guide the algorithm towards the optimal solution. For finding the minimum number of groups one could imagine simply using the number of groups used to pack all the items. This is correct from a mathematical point of view, but is unusable in practice as such a cost function leads to an extremely 'unfriendly' landscape of the search space. A very small number of optimal points in the space are lost in an exponential number of points where this apparent cost function is just one unit above the optimum. Additionally, all those sub-optimal points result in the same yield the same cost value. Therefore, such a cost function lacks the capacity of guiding an algorithm in the search, transforming the problem to that of finding a 'Needle-in-a-haystack' [30].

Equation 4-1 shows a more suitable fitness function proposed by Falkenauer [55]

$$f(s) = \frac{\sum_{i=1}^{N} (F_i/C)^k}{N} \tag{4-1}$$

With N being the number of groups used, F_i the sum of sizes of the objects in the bin i, C the bin capacity and k is a heuristic exponential constant, $k \geq 1$. The larger k is, the more are well-filled 'elite' groups preferred as opposed to a collection of about equally filled groups. Falkenauer uses $k = 2$ [30], but more recently Stawowy [56] reported that $k = 4$ gives slightly better results.

The second metric needed for the problem of signal to PDU assignment is that of the generated bus load. This can be computed by Equation 2-2, as described in section 2-2.

After evaluating all candidates of a given population based on these metrics, non-dominated sorting genetic algorithm II (NSGA-II) is used to select genotypes with optimal characteristics to act as parent genotypes (see subsection 3-4-2).

4-3 Implementation

With these fundamental design choices of the GA (encoding, genetic operations and selection criteria), an automated signal to PDU assignment-algorithm can be created. An ECUs communication matrix, listing the various signals, their size and timing requirements, is used as input. Based on these signals, the first generation is created by randomly assigning all signals to PDUs. From there on, the algorithm runs according to several additional parameters:

- population size: 150
- crossover rate: 0.96
- mutation rate: 1/number of signals.
- number of generations: up to 60.000

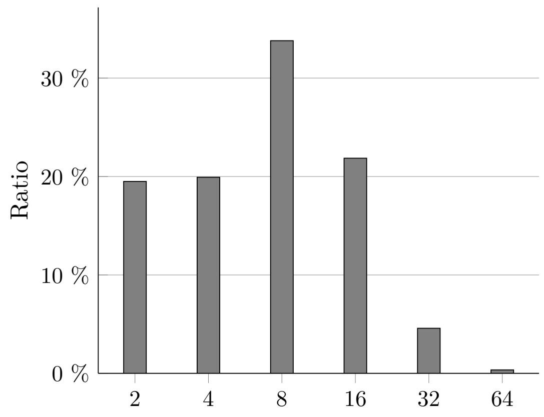
4-4 Experimental results

4-4-1 Set-up

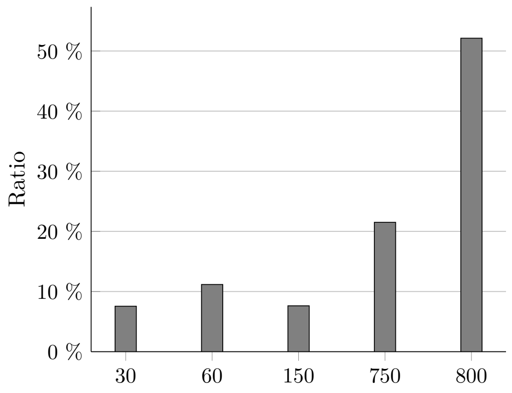
To evaluate the result of the algorithm, we compare the GA with different implementations and heuristics. To judge the overall performance of the result we take the assignments of ECUs currently in use that were created manually as a reference [1].

A realistic E/E-A set-up consisting of a CAN bus with 16 ECUs and an overall number of 1500 signals is carried out as a case study to show the applicability of the proposed methods. The distribution of the sizes and timing-requirements of the signals is illustrated in Figure 4-7. The signals are highly heterogeneous in terms of their timing-requirements and size. The experiments were carried out on an Intel Core i7 2.6 GHz machine with 16 GB RAM. The GA was implemented using the modular meta-heuristic optimization framework Opt4J in the version 3.1.2 [57].

[1]Due to the commercial sensitivity of this data, the names of the presented ECUs are anonymized.



(a) Distribution of signal sizes in bits.



(b) Distribution of signal timing requirements in milliseconds.

Figure 4-7: Distributions for signal sizes and timing requirements.

4-4-2 Numerical results

As the PDUs can only be filled by one transmitting ECU, the optimization has to be run for each ECU individually. The GA-based optimizations are repeated 1000 times to avoid local optima and assess the quality/reproducibility of the result. Table 4-1 shows the results for ECU_A, which transmits 260 signals and requires a bus load of 8.03% in the current configuration with an average PDU fill-ratio of 0.3197.

	Original	FFD	Holland	Falkenauer
# Runs			1000	1000
Generations			60k	20k
Run time/gen.		≤ 1 s	2.34 ms	10.99 ms
Busload:	**8.03%**	**9.32%**		
Min			**4.62%**	**5.50%**
Max			5.76%	7.16%
Avg (μ)			4.82%	6.36%
Std.dev (σ)			0.17%	0.30%
Fill-rate:	**0.3197**	**0.7294**		
Max			**0.9688**	**0.9706**
Min			0.7992	0.9642
Avg (μ)			0.9207	0.9693
Std.dev (σ)			0.0293	0.0013
Used PDUs	61	35	29	29

Table 4-1: Results for ECU_A.

Based on these results, a number of interesting observations can be made. The heuristic approach using the FFD-algorithm does not lead to a result that optimizes both metrics and even leads to an overall worse bus load, the most important metric for the E/E-A. This behavior is as we expected, since the FFD only optimizes one metric, namely the fill ratio of the PDUs. However, with such a large number of signals, the FFD struggles to find the optimal solution of using the minimum number of PDUs ($fill\ rate \rightarrow 1$). Looking at the GA, with the Holland-style implementation we notice a considerable improvement both in bus load and in fill rate. The bus load is almost halved, as is the number of PDUs required to fit all signals. We do notice that, despite the long run time of the algorithm, there is a large spreading of the outcomes., ranging from 4.62% to 5.76%. This indicates that the algorithm runs into a local optimum that it can not escape by mutation, even after 60.000 generations. The Falkenauer-style algorithm should in theory be better suitable to solve the group-based optimization problem of assigning signals to PDUs. The results of Table 4-1 however show a different image, although the fill ratio climes a small amount, the bus load decreases compared to the Holland-style results. We can see a very tight spread of fill ratio's, with a standard deviation of 0.0013, the bus load has a much larger spread.

4-4-3 Genetic Algorithm results

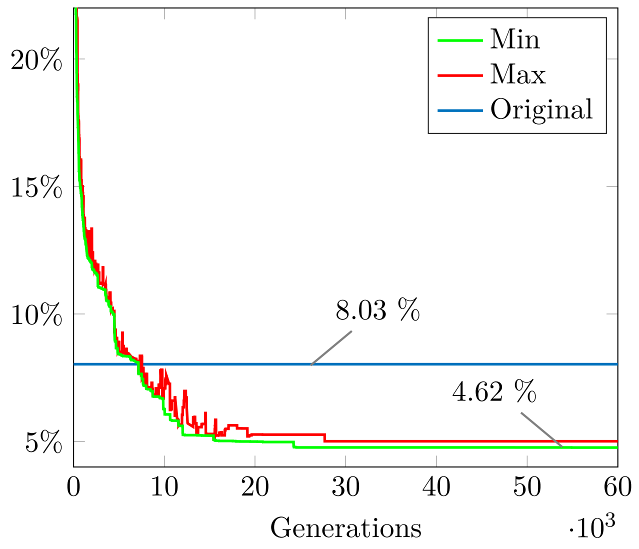
Figure 4-8 shows the evolution of the GA through the different generations. 4-8a and 4-8b show the process for the Holland-style algorithm, whilst 4-8c and 4-8d show the progression of the Falkenauer-style GA. The limited efficiency of the Holland-style algorithm discussed in section 4-1, displays itself by the slow convergences of the algorithm toward the final result, both in bus load and fill ratio. Despite this, the algorithm outperforms the original configuration after 10.000 generations, slowly converging to a minimum of 4.62%, which it reaches after around 30.000 generations.

The Falkenauer-style algorithm (4-8c) shows a much steeper convergence rate. This confirms the expectations of this approach being much better suited to solving group-based problems. The added complexity of the crossover process in this algorithm causes the higher average run time per generation. Despite this, the fill rate reaches its top after only 50 generations, whilst the bus load converges to a minimum of 5.50% after 500 generations.

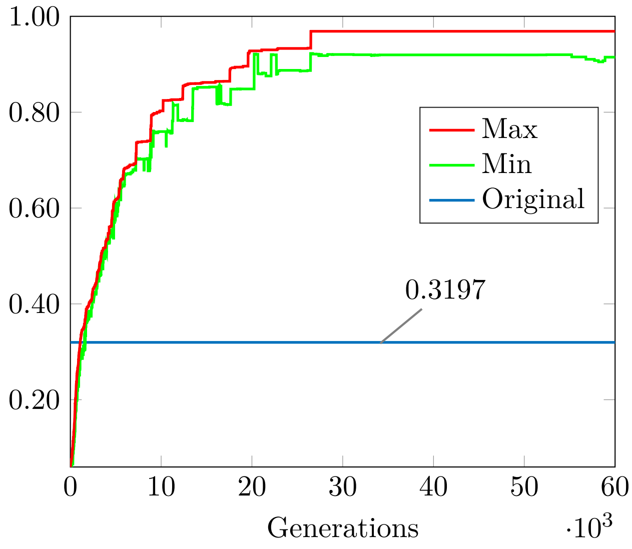
4-4-4 First conclusions

These preliminary tests already show strong results for the optimization of ECU_A. Both algorithms are able to decrease the bus load considerably below that of the manually assigned signals. This indicates a clear potential of optimizing the E/E-As using this approach. Figure 4-8 clearly illustrates the behavior we predicted. The straightforward Holland-style algorithm is able to reach a good solution, but takes a very long time reaching this result and has limited reliability. As to optimize the E/E-A in practice, the algorithm needs to be run for all ECUs individually, it is unfeasible to run the algorithm multiple hundred times per ECU. With similar ECUs as ECU_A this would require a processing time of multiple hours per ECU. The Falkenauer-algorithm provides a much faster convergence, but suffers from low reliability and non-optimal results for the bus load.

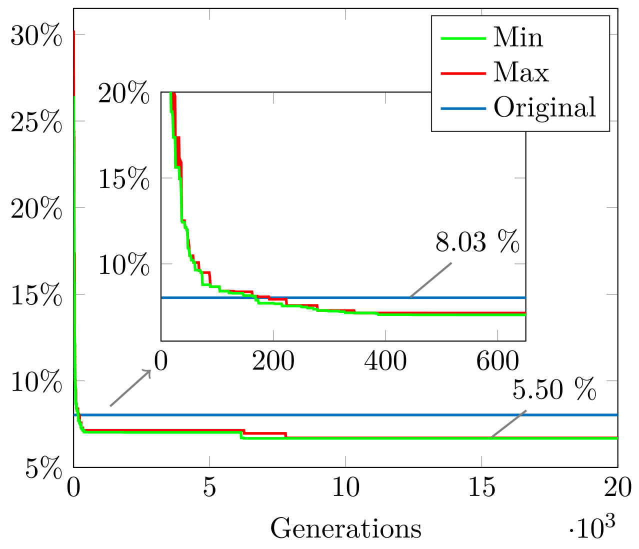
Although our initial optimization concepts have shown to be effective, similar to the process of Genetic Algorithms, this optimization concept also leaves room for evolution.



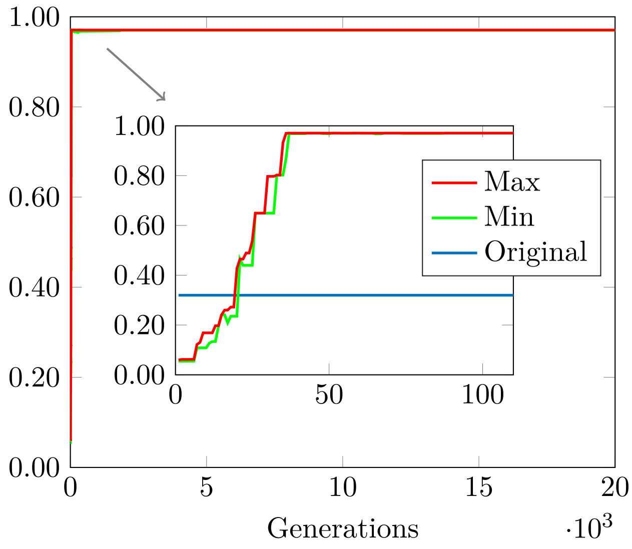
(a) Bus load using *Holland* encoding.



(b) Fill rate using *Holland* encoding.



(c) Bus load using *Falkenauer* encoding.



(d) Fill rate using *Falkenauer* encoding.

Figure 4-8: Results after GA optimization of ECU_A.

4-5 Revisiting genetic operators

Ideally we would like to combine the favorable properties of both algorithms; Namely the low bus rate and fast convergence. Explanation for the sub-optimal bus load of the Falkenauer-style algorithm can be found by taking a closer look at the genetic operators employed. Table 4-1 shows the Falkenauer-style GA reaches a higher fill rate than the Holland-style GA. This can be explained by the Falkenauer crossover operation. After recombining bins from different parents, the orphaned items are distributed over the remaining bins by means of the FFD. This algorithm is excellent in optimally filling PDUs, however, as we already discussed it only optimizes for one metric, that of the fill-ratio. Next we will propose a modification of the standard FFD algorithm to forgo this issue.

Although the crossover operator is very effective at carrying over good PDU configurations to future generations, the process of assigning orphaned signals is not suited for our optimization criteria. When these signals are assigned to new PDUs solely based on capacity, ‘good’ PDUs may be ‘contaminated’ with signals of much different timing-requirements. The chance that each PDU is randomly filled with signals of similar timing-requirements is very slim. As the signal with the most stringent requirement, determines the timing of he whole PDU, this can have disastrous effects on the performance of the whole system.

By a small amendment to Algorithm 1, we can change this behavior to include the timing-properties of a PDU and signal during the matching process. Algorithm 2 shows the improved FFD algorithm. Operation 6 is the main differentiation. This rule allows only signals to be assigned to PDU whose average timing requirements differ by no more then 10%.

Algorithm 2 Falkenauer crossover with modified FFD.

```
1:  while $s$ do
2:    for $i := 0$ to s do
3:      for each group in p1 do
4:        Calculate average $t_{cycl}$ of group
5:        if size($s_i$) < space(group) then
6:          if $t_{cycl}(s_i) \leq \pm\, 0.10 \cdot \overline{t_{cycl}}$ then
7:            Insert $s_i$ in group
8:            Remove $s_i$ from $s$
9:          end if
10:       end if
11:     end for
12:   end for
13:   if $s$ then
14:     Add new empty group
15:   end if
16: end while
```

4-5-1 Experimental results

Using the modified genetic operators we re-run the GA for ECU_A and compare it to the original implementation. The modified FFD is also applied to the example as a stand-alone heuristic, to compare the result to that of the GA. Table 4-2 shows the numerical results of the Falkenauer-algorithm with the modified FFD. Figure 4-9 shows the progress of the different GAs.

	Original	Holland	Falkenauer	Mod. Falk.	FFD	Mod. FFD
# Runs		1000	1000	1000		
Generations		60k	20k	500		
Run time/gen.		2.34 ms	10.99 ms	38.06 ms		
Busload:	**8.03%**				**9.32 %**	**10.82 %**
Min		**4.62%**	**5.50%**	**4.61%**		
Max		5.76%	7.16%	4.61%		
Avg (μ)		4.82%	6.36%	4.61%		
Std.dev (σ)		0.17%	0.30%	0.00		
Fill-rate:	**0.3197**				**0.7294**	**0.4084**
Max		**0.9688**	**0.9706**	**0.9132**		
Min		0.7992	0.9642	0.9095		
Avg (μ)		0.9207	0.9693	0.9131		
Std.dev (σ)		0.0293	0.0013	0.0004		
Used PDUs	61	29	29	29	35	52

Table 4-2: Results for ECU_A with modified FFD.

Very notable in the results from the modified Falkenauer algorithm is that bus load reaches the lowest value so far of 4.61%. It does so for all 1000 runs of the algorithm. We observe the algorithm produces a lower fill rate than the other implementations, the number of used PDUs is however equal to 29 for all cases. This indicates that the modified algorithm is capable of finding a more ideal balance between the two metrics, without deteriorating the end result. A higher run time per generation is noted due to the added complication in assigning signals to suitable PDUs. Due to the effectiveness of the new crossover operation, the need for the mutation operation diminishes. Taking away this operation, normally a vital part of the GA, has no negative effect on the quality of the solution, but takes away about 4 ms of processing time per generation.

Applying the heuristic directly, without the process of the GA deteriorates the result compared to the standard FFD. Reaffirming our previous statements that these heuristics are not suitable for MOCOP with large item sizes.

Looking at Figure 4-9a and Figure 4-9b, we can see that, compared to the original Falkenauer process below, the modified FFD converges to the value of 4.61% after only 50 generations (< 2 $seconds$).

The rapid convergence and reliability of the modified Falkenauer-algorithm make it highly effective for optimizing the complete E/E-A. Running the optimizing for all ECUs would take mere seconds whilst guaranteeing the highest quality result. This allows for the optimization to be easily integrated in the current design process.

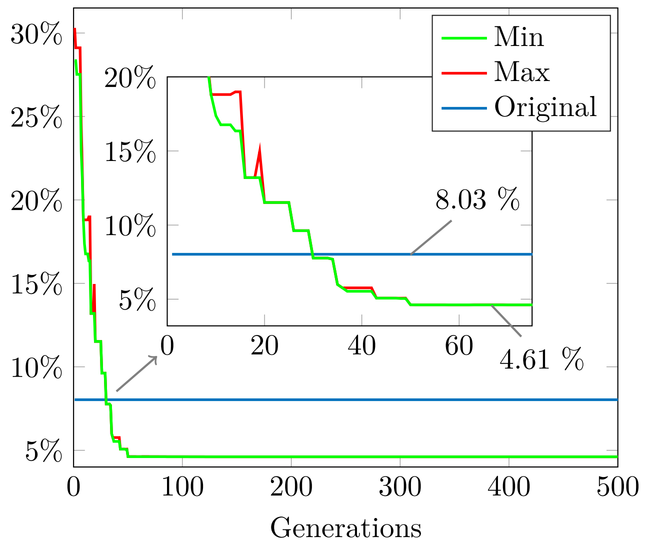
4-6 Optimizing the Electric/Electronic-Architecture

The previous tests have shown the feasibility of the presented optimization process for E/E-As. To further corroborate the effectiveness of the proposed algorithms, all three implementations are run in the same manner as before for additional ECUs. Detailed results are presented for ECU_B in Table 4-3, results for all ECUs connected to a single Controller Area Network (CAN) bus are summarized in Table 4-4.

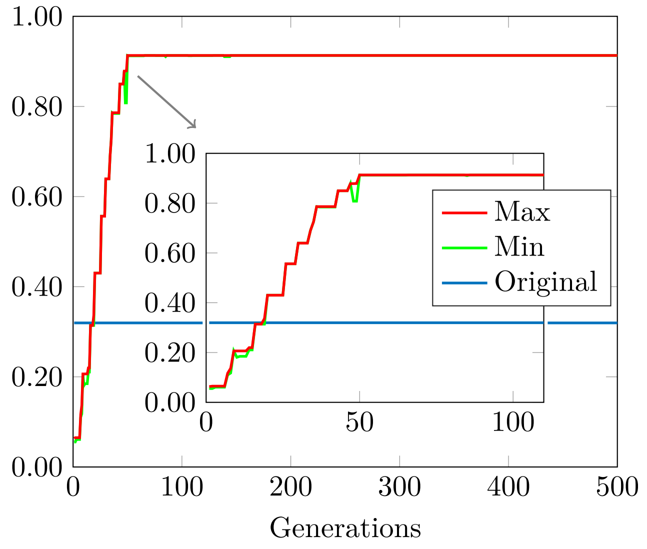
	Original	Holland	Falkenauer	Mod. Falk.
# Runs		1000	1000	1000
Generations		60k	20k	500
Run time/gen.		2.34 ms	12.45 ms	50.05 ms
Busload:	**9.10%**			
Min		**6.73%**	**7.62%**	**6.72%**
Max		8.21%	9.96%	6.87%
Avg (μ)		7.15%	8.62%	6.84%
Std.dev (σ)		0.26%	0.37%	0.06
Fill-rate:	**0.5192**			
Max		**0.9794**	**0.9804**	**0.9771**
Min		0.8643	0.9456	0.9373
Avg (μ)		0.9498	0.9797	0.9465
Std.dev (σ)		0.0223	0.0021	0.0145
Used PDUs	51	35	34	34

Table 4-3: Results for ECU_B with modified FFD.

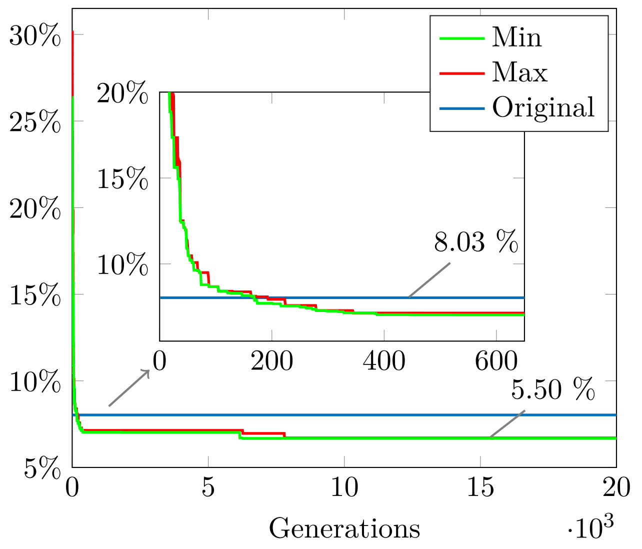
The results show the effectiveness of the proposed algorithm. The bus load of all ECUs is reduced considerably. The total bus load for this example CAN bus is reduced by more than 10%. The number of used PDUs is reduced by more then 40%.



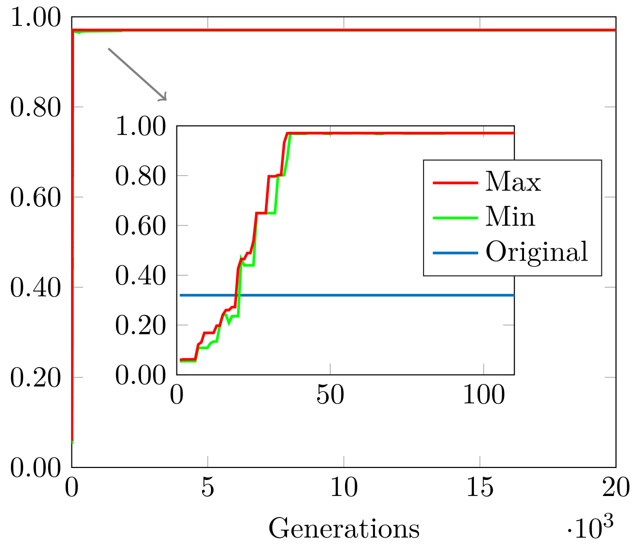
(a) Bus load using *Modified Falkenauer* encoding.



(b) Fill rate using *Modified Falkenauer* encoding.



(c) Bus load using *original Falkenauer* encoding.



(d) Fill rate using *original Falkenauer* encoding.

Figure 4-9: Results after modified GA optimization of ECU_A.

ECU	Original		Modified Falkenauer	
(# signals)	Fill ratio	Bus load	Fill ratio	Bus load
ECU_A (260)	0.3197 (61)	8.03%	0.9095 (29)	4.61%
ECU_B (287)	0.5192 (51)	9.10%	0.9373 (34)	6.72%
ECU_C (56)	0.3551 (16)	0.40%	0.9707 (8)	0.26%
ECU_D (119)	0.3745 (30)	2.16%	0.9200 (18)	1.72%
ECU_E (241)	0.3922 (46)	4.36%	0.9342 (26)	2.59%
ECU_F (301)	0.5201 (64)	3.09%	0.9698 (44)	2.48%
ECU_G (50)	0.3803 (10)	1.15%	0.7556 (7)	0.90%
ECU_H (88)	0.2385 (23)	1.77%	0.8705 (10)	0.82%
ECU_I (34)	0.4101 (13)	3.52%	0.7615 (9)	3.19%
Sum (1436)	(314)	33.58%	(185)	23.29%

Table 4-4: Results for all ECUs on a single CAN bus using the modified Falkenauer-algorithm.

Chapter 5

Conclusions and Recommendations

5-1 Conclusions

Optimizing the Electric/Electronic-Architecture (E/E-A) is a complex task due to the large number of Electronic Control Units (ECUs) and the many different communication technologies employed. The addition of evermore sensors, actuators, ECUs and Advanced Driver Assistance System (ADAS) functions have increased the complexity of this system considerably. This causes currently employed communication technologies to reach their technical limits, causing undesired levels of bit error rate (BER) leading to unexpected and potentially dangerous behavior for customers. Implementing new communication technologies is an unviable option due to the tremendous redevelopment costs associated. This research proposes a much less invasive way of optimizing the E/E-A. By introducing sophisticated new design methods, technologies already in place have been greatly optimized. The proposed solution is applicable throughout the various communication systems by focusing on a commonality between them, increasing the efficiency of all of them.

ECUs are interconnected with sensors, actuators and each other, using multiple physical communication technologies such as Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay (FR) and Ethernet. On a logical level, sensor data, diagnostic information and other signals are grouped in Protocol Data Units (PDUs) that are consequently transmitted over CAN, LIN, FR and Ethernet. Up to 70 ECUs need to exchange more than 10.000 different signals that have a wide range of timing-requirements and other properties. Determining which signals are grouped to which PDUs in so called communication matrices is a process that depends on many variables which make it a very complex task to solve efficiently. Due to lack of a proper method to deal with this efficiently. this process is currently performed manually.

In this research the problem is defined as a Bin Packing Problem (BPP). Several methods of solving BPPs are considered. Many of the currently known exact methods and heuristics prove to be unsuitable for the Multi-Objective Combinatorial Optimization Problem (MOCOP) of optimizing the signal to PDU assignments, due to the complexity created by the multitude of variables. Instead, a concept for a Genetic Algorithm (GA) approach is created. This concept implements widely established solving methods proposed by Coffman et al, and Falkenauer et al. which are combined and improved upon to achieve the best possible adaptation to the specific needs of the problem at hand.

Tests performed during this research show the potential of determining the communication matrices using the proposed GA concept. The algorithm is able to reduce the bus load induced by ECUs by up to 54% whilst also reducing the total number of needed PDUs. The unique solving method created has made a highly efficient and reliable algorithm that is able to reach a result within milliseconds. Meanwhile, the optimization process is easily implementable in the current design of the E/E-A.

When applied to the whole E/E-A currently in use in vehicles, these optimizations generate much welcomed capacity on the vehicles excising communication channels and allow the E/E-A to process up to 5.000 additional signals. This facilitates for the implementation of many more functions such as ADAS, paving the way forward to the self-driving cars of the future.

5-2 Scientific contributions

The claims made by Falkenauer and Khuri et al., that the standard ordering representation and genetic operators are inadequate whilst dealing with grouping problems [30], do not hold up when dealing with multi-objective grouping enetic algorithm (GGA). This research proves the strength of GAs; Even for highly specific MOCOPs, a slight modification of the standard Holland-scheme [44] and well chosen selection methods, gives very good results for various optimization problems. Experimental results show the extreme flexibility of evolutionary algorithms, which can be applied to generate very good solutions without going into the very structure of the problem. By modifying the crossover-operation proposed by Falkenauer, and by combining the First Fit Decreasing (FFD) heuristic of Coffman [31], group based encoding schemes can be effectively applied to bi-objective grouping problem. Only with these modifications, the representation proposed by Falkenauer clearly outperforms the standard representation scheme of Holland.

5-3 Future Work

The signal assignment strategy as proposed already provides an implementable optimization process. To further improve the design process, several aspects can be investigated further:

- The new signal assignment is optimized towards several metrics as described in subsection 3-1-1. The signals are therefore no longer assigned by function, although some ECUs need a number of signals to process specific function. For these cases, signals need to be buffered locally on the ECU, awaiting the arrival of all necessary signals.
- Some signals such as diagnostics have predefined identifiers to allow generic diagnostic readers and official authorities to communicate with the vehicle and extract fault codes. Therefore, the assignment of signals to ‘random’ ID’s has to be avoided. Several options

can be though of to achieve this. The signals and ID's in question could be excluded from the optimization process altogether. Or the diagnostic interface could be flashed with a lookup table, translating the correct ID's to the external devices.

- With each signal added to the system, the optimization process can be re-run to achieve the most optimal system performance. Due to the nature of the GA each iteration of this process will result in very different signal assignments as the PDU-ID a specific signal is assigned to is not maintained. As some ECUs are sourced from external suppliers, ECUs might not offer the ability to flash a new lookup table herself. To avoid high costs of re-engineering these devices, the optimization process could incorporate a way of excluding certain ECUs from the optimization. Another approach would be to preemptively add overhead bits to existing signals.

Chapter 6

Glossary

ECU	Electronic Control Unit
ABS	anti-lock brake system
ESC	electronic stability control
E/E-A	Electric/Electronic-Architecture
LIN	Local Interconnect Network
FR	FlexRay
MOST	Media Oriented Systems Transport
OEM	Original Equipment Manufacturer
CAN	Controller Area Network
ADAS	Advanced Driver Assistance System
BPP	Bin Packing Problem
MTP	
RTS	Real Time System
BFD	Best Fit Decreasing
FFD	First Fit Decreasing
MBS	Minimum Bin Slack
MTP	Martello-Toth Procedure
ACO	Ant Colony Optimization

GA	Genetic Algorithm
MOCOP	Multi-Objective Combinatorial Optimization Problem
PDU	Protocol Data Unit
EA	Evolutionary algorithms
NSGA-II	non-dominated sorting genetic algorithm II
BER	bit error rate
GGA	grouping enetic algorithm
OSI	Open Systems Interconnection
TPMS	Tire Pressure Monitoring System

Bibliography

[1] P. Gao, H.-W. Kaas, D. Mohr, and D. Wee - McKinsey&Company, "Automotive revolution - perspective towards 2030," 2016. [Online]. Available: https://www.mckinsey.de/sites/mck_files/files/automotive_revolution_perspective_towards_2030.pdf

[2] A. Gallasch and J. Grafe - IBM Business Consulting Services, "Challenges for the automotive industry in an on demand environment," 2004. [Online]. Available: ftp://ftp.software.ibm.com/software/plm/de/challenges_automotive.pdf

[3] R. Isermann, R. Schwarz, and S. Stolzl, "Fault-tolerant drive-by-wire systems," *Control Systems, IEEE*, vol. 22, no. 5, pp. 64–81, Oct 2002. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1035218

[4] ISO 42010, "ISO/IEC standard for systems and software engineering - recommended practice for architectural description of software-intensive systems," *ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15*, pp. c1–24, July 2007.

[5] T. Nolte, "Share-driven scheduling of embedded networks," Ph.D. dissertation, Mälardalen University, May 2006. [Online]. Available: http://www.es.mdh.se/publications/944-

[6] ISO 11898, "Road vehicles — interchange of digital information — controller area network (CAN) for high-speed communication." *International Standards Organisation (ISO)*, ISO Standard-11898, 1991.

[7] M. Ruff, "Evolution of local interconnect network (LIN) solutions," in *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, vol. 5, Oct 2003, pp. 3382–3389 Vol.5.

[8] R. Makowitz and C. Temple, "Flexray - a communication network for automotive control systems," in *2006 IEEE International Workshop on Factory Communication Systems*, 2006, pp. 207–212.

[9] MOST Cooperation, "MOST Specification Revision 2.3." Karlsruhe, 2008. [Online]. Available: http://www.mostcooperation.com/publications/specifications-organizational-procedures/

[10] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Commun. ACM*, vol. 19, no. 7, pp. 395–404, Jul. 1976. [Online]. Available: http://doi.acm.org/10.1145/360248.360253

[11] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 534–545, April 2015.

[12] A. Kern, "Ethernet and ip for automotive e/e-architectures," Ph.D. dissertation, 2013. [Online]. Available: https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/3747

[13] Controller Area Network, "Controller area network — Wikipedia, the free encyclopedia," 2016, [Online; accessed 29-January-2016]. [Online]. Available: https://de.wikipedia.org/wiki/Controller_Area_Network

[14] Texas Instruments, "Introduction to the Controller Area Network (CAN)," *Application Report SLOA101*, 2002.

[15] S. Ding, H. Tomiyama, and H. Takada, "An effective ga-based scheduling algorithm for flexray systems," *IEICE - Trans. Inf. Syst.*, vol. E91-D, no. 8, pp. 2115–2123, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1093/ietisy/e91-d.8.2115

[16] T. Thomsen and G. Drenkhan, "Ethernet for AUTOSAR." EBAutomotive Gmbh, Erlangen, 2008.

[17] S. Tuohy, D. O?Cualain, E. Jones, and M. Glavin, "Distance determination for an automobile environment using inverse perspective mapping in opencv," in *Proc. Irish Signals and Systems Conference*, vol. 2010, 2010.

[18] H. J. Song, H. P. Hsu, R. Wiese, and T. Talty, "Modeling signal strength range of tpms in automobiles," in *Antennas and Propagation Society International Symposium, 2004. IEEE*, vol. 3, June 2004, pp. 3167–3170 Vol.3.

[19] W. Niu, J. Li, and T. Talty, "Intra-vehicle UWB channel measurements and statistical analysis," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, Nov 2008, pp. 1–5.

[20] T. ElBatt, C. Saraydar, M. Ames, and T. Talty, "Potential for intra-vehicle wireless automotive sensor networks," in *Sarnoff Symposium, 2006 IEEE*, March 2006, pp. 1–4.

[21] O. Kiyotsugu, "Wiring harnesses for next generation automobiles," in *Fujikura Technical Review*, vol. 42, 2013, pp. 77–80.

[22] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *Proceedings of the 19th USENIX Conference*

on Security, ser. USENIX Security'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 21–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=1929820.1929848

[23] M. D. Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer Publishing Company, Incorporated, 2012. [Online]. Available: http://inst.cs.berkeley.edu/~ee249/fa08/Lectures/handout_canbus2.pdf

[24] E. Mayer - Vector Informatik GmbH, "Serial bus systems in the automobile," 2006. [Online]. Available: http://vector.com/portal/medien/cmc/press/PTR/SerialBusSystems_Part2_ElektronikAutomotive_200612_PressArticle_EN.pdf

[25] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990. [Online]. Available: http://www.or.deis.unibo.it/knapsack.html

[26] 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., 1979.

[27] S. S. Skiena, "Who is interested in algorithms and why?: Lessons from the stony brook algorithms repository," *SIGACT News*, vol. 30, no. 3, pp. 65–74, Sep. 1999. [Online]. Available: http://doi.acm.org/10.1145/333623.333627

[28] J. N. D. Gupta and J. C. Ho, "A new heuristic algorithm for the one-dimensional bin-packing problem," *Production Planning & Control*, vol. 10, no. 6, pp. 598–603, 1999. [Online]. Available: http://dx.doi.org/10.1080/095372899232894

[29] T. Dokeroglu and A. Cosar, "Optimization of one-dimensional bin packing problem with island parallel grouping genetic algorithms," *Computers & Industrial Engineering*, vol. 75, pp. 176 – 186, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S036083521400182X

[30] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, no. 1, pp. 5–30, 1996. [Online]. Available: http://dx.doi.org/10.1007/BF00226291

[31] J. Coffman, E.G., M. Garey, and D. Johnson, "Approximation algorithms for bin-packing - an updated survey," in *Algorithm Design for Computer System Design*, ser. International Centre for Mechanical Sciences, G. Ausiello, M. Lucertini, and P. Serafini, Eds. Springer Vienna, 1984, vol. 284, pp. 49–106. [Online]. Available: http://dx.doi.org/10.1007/978-3-7091-4338-4_3

[32] R. Harren, "Two-dimensional packing problems," Ph.D. dissertation, UniversitÃďt des Saarlandes, Postfach 151141, 66041 SaarbrÃijcken, 2010. [Online]. Available: http://scidok.sulb.uni-saarland.de/volltexte/2010/3470

[33] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003. [Online]. Available: http://doi.acm.org/10.1145/937503.937505

[34] E. Hopper, “Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods,” Ph.D. dissertation, University of Wales. Cardiff, 2000. [Online]. Available: http://vmk.ugatu.ac.ru/c%26p/article/hopper/PhDisser/part1.pdf

[35] K. Sim, “Novel hyper-heuristics applied to the domain of bin packing,” Ph.D. dissertation, Edinburgh Napier University, 2014. [Online]. Available: http://researchrepository.napier.ac.uk/7563/1/Sim_07005443_PhD.pdf

[36] M. A. Kaaouache and S. Bouamama, “Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud,” *Procedia Computer Science*, vol. 60, pp. 1061 – 1069, 2015, knowledge-Based and Intelligent Information &amp; Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915022784

[37] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, “Worst-case performance bounds for simple one-dimensional packing algorithms,” *SIAM Journal on Computing*, vol. 3, no. 4, pp. 299–325, 1974. [Online]. Available: http://dx.doi.org/10.1137/0203025

[38] Ö. Ülker, E. Korkmaz, and E. Özcan, “A grouping genetic algorithm using linear linkage encoding for bin packing,” in *Parallel Problem Solving from Nature Ű PPSN X*, ser. Lecture Notes in Computer Science, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, Eds. Springer Berlin Heidelberg, 2008, vol. 5199, pp. 1140–1149. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87700-4_113

[39] C. Coello Coello, C. Dhaenens, and L. Jourdan, “Multi-objective combinatorial optimization: Problematic and context,” in *Advances in Multi-Objective Nature Inspired Computing*, ser. Studies in Computational Intelligence, C. Coello Coello, C. Dhaenens, and L. Jourdan, Eds. Springer Berlin Heidelberg, 2010, vol. 272, pp. 1–21. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11218-8_1

[40] L. T. Bui, Y. S. Ong, N. X. Hoai, H. Ishibuchi, and P. N. Suganthan, *Simulated Evolution and Learning: 9th International Conference, SEAL 2012, Hanoi, Vietnam, December 16-19, 2012, Proceedings.* Springer, 2012, vol. 7673. [Online]. Available: https://books.google.com/books?id=eFG5BQAAQBAJ&printsec=frontcover

[41] A. Kafafy, A. Bounekkar, and S. Bonnevay, “A hybrid evolutionary metaheuristics (hemh) applied on 0/1 multiobjective knapsack problems,” in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’11. New York, NY, USA: ACM, 2011, pp. 497–504. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001645

[42] J. Brownlee, *Clever algorithms: nature-inspired programming recipes.* Jason Brownlee, 2011. [Online]. Available: http://www.cleveralgorithms.com/nature-inspired/evolution/nsga.html

[43] M. Ehrgott and X. Gandibleux, “Approximative solution methods for multiobjective combinatorial optimization,” *Top*, vol. 12, no. 1, pp. 1–63, 2004. [Online]. Available: http://dx.doi.org/10.1007/BF02578918

[44] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[45] C. L. Bridges and D. E. Goldberg, "An analysis of reproduction and crossover in a binary-coded genetic algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application.* Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987, pp. 9–13. [Online]. Available: http://dl.acm.org/citation.cfm?id=42512.42514

[46] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. [Online]. Available: http://dl.acm.org/citation.cfm?id=534133

[47] A. Kumar, "Encoding schemes in genetic algorithm," *International Journal of Advanced Research in IT and Engineering*, vol. 2, no. 3, pp. 1–7, 2013. [Online]. Available: http://garph.co.uk/IJARIE/Mar2013/1.pdf

[48] B. Fox and M. McMahon, "Genetic operators for sequencing problems," *Foundations of genetic algorithms*, vol. 1, pp. 284–009, 1990.

[49] P. Chu and J. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, 1998. [Online]. Available: http://dx.doi.org/10.1023/A%3A1009642405419

[50] E.-G. Talbi, *Metaheuristics: from design to implementation.* Wiley Publishing, 2009, vol. 74.

[51] L. S. Randaccio, "Resources optimization in multimedia communications," Ph.D. dissertation, 2007. [Online]. Available: http://www.diee.unica.it/DRIEI/tesi/18_sanna_randaccio.pdf

[52] R. Van Driessche and R. Piessens, "Load balancing with genetic algorithms," in *Parallel Problem Solving from Nature, 2*, R. Männer and B. Manderick, Eds., 1992, pp. 341–350. [Online]. Available: https://lirias.kuleuven.be/handle/123456789/133983

[53] H. Ding, A. El-Keib, and R. Smith, "Optimal clustering of power networks using genetic algorithms," *Electric Power Systems Research*, vol. 30, no. 3, pp. 209 – 214, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0378779694008574

[54] N. J. Radcliffe, "Forma analysis and random respectful recombination." in *ICGA*, vol. 91, 1991, pp. 222–229.

[55] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, vol. 2, 1992, pp. 1186–1192.

[56] A. Stawowy, "Evolutionary based heuristic for bin packing problem," *Computers & Industrial Engineering*, vol. 55, no. 2, pp. 465 – 474, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835208000090

[57] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich, “Opt4j: A modular framework for meta-heuristic optimization,” in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’11. New York, NY, USA: ACM, 2011, pp. 1723–1730. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001808

Delft University of Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science
Mekelweg 4
2628 CD Delft

T +31 (0)15 27 84568
F +31 (0)15 27 87022
E bureau@ewi.tudelft.nl

www.ewi.tudelft.nl