Evolutionärer Funktionstest von eingebetteten Systemen für abstandsbasierte Fahrerassistenzfunktionen im Automobil

Dissertation

der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Ing. (FH) Oliver Bühler, MSc
aus Bissingen an der Teck

Tübingen
2007

Tag der mündlichen Qualifikation:	??.??.2007
Dekan:	Prof. Dr. Michael Diehl
1. Berichterstatter:	Prof. Dr. Wolfgang Rosenstiel
2. Berichterstatter:	Prof. Dr.-Ing. Ina Schieferdecker

Vorwort

Die vorliegende Arbeit entstand in der Abteilung „Entwicklung Fahrerassistenzsysteme“ und der Forschungsabteilung „Software-Methoden und -Tools“ der DaimlerChrysler AG in Sindelfingen und in Berlin sowie am Lehrstuhl für Technische Informatik des Wilhelm-Schickard-Instituts für Informatik der Universität Tübingen.

Mein besonderer Dank gilt Herrn Prof. Dr. rer. nat. W. Rosenstiel für die Betreuung der Arbeit und für die Übernahme des Hauptberichts. Ebenso bedanke ich mich ganz herzlich bei Frau Prof. Dr.-Ing. I. Schieferdecker für die Übernahme des Zweitberichts.

Bei allen Kollegen der Abteilung „Entwicklung Fahrerassistenzsysteme“ möchte ich mich für die kollegiale Atmosphäre und für die Hilfsbereitschaft bedanken. Insbesondere gilt mein Dank Herrn Dr. Joachim Wegener für die fachlichen Diskussionen, die mir immer wieder neue Anregungen für meine Arbeit gaben. Ein ganz besonderer Dank gilt Herrn Dr. Helmut Keller und Herrn Bernd Danner, die diese Arbeit in der Abteilung „Entwicklung Fahrerassistenzsysteme“ ermöglicht haben.

Bissingen an der Teck, im April 2007

Oliver Bühler

Inhaltsverzeichnis

1 Einführung **1**

1.1 Motivation 1

1.2 Aufgabenstellung 2

1.3 Aufbau der Arbeit 3

2 Grundlagen **5**

2.1 Statische Testmethoden 6

2.2 Dynamische Testmethoden 7

2.2.1 Strukturorientierte Testmethoden 8

2.2.2 Funktionsorientierte Testmethoden 10

2.2.3 Diversifizierende Testmethoden 14

2.2.4 Statistischer Test 15

3 Stand der Technik **17**

3.1 Testen von im Automobil eingebetteten Systemen 17

3.1.1 Testen eingebetteter Systeme im Fahrzeug 17

3.1.2 Testen eingebetteter Software in Simulationsumgebungen 18

3.2 Entwicklungsprozess in der Automobilindustrie 20

3.2.1 Entwurfsprozess 23

3.2.2 Integrations- und Testprozess 25

3.3 Evolutionäres Testen 26

3.3.1 Evolutionäre Algorithmen 27

3.3.2 Prinzip des evolutionären Testens 29
3.3.3 Evolutionärer Strukturtest 31
3.3.4 Evolutionärer Test nicht-funktionaler Eigenschaften 31
3.3.5 Evolutionärer Funktionstest 32
3.4 Bedarf für eine neue Methode 32

4 Evolutionärer Funktionstest für Realzeitsysteme 35
4.1 Architektur des Testsystems 36
4.2 Codierung der Testdaten 39
4.2.1 Ereignissteuerung 40
4.2.2 Klassifikation von Genen 41
4.2.3 Codierung von Kennlinienverläufen 42
4.3 Entwurf der Zielfunktion 45
4.3.1 Vorgehensweise 46
4.3.2 Ergebnisklassen 47
4.3.3 Optimierungsziel 48
4.3.4 Klassifikationsmerkmale 49
4.3.5 Merkmalsbasierte Zielfunktion 49
4.4 Einordnung in den Entwicklungsprozess 50

5 Evolutionärer Funktionstest für Fahrerassistenzsysteme 53
5.1 Das „Automatische Parksystem“ 54
5.1.1 Anwendung 54
5.1.2 Codierung 55
5.1.3 Zielfunktion 57
5.2 Generierung der Populationen für das Parksystem 63
5.2.1 Fitnesszuweisung 64
5.2.2 Elternselektion 64
5.2.3 Rekombination 66
5.2.4 Mutation 68

5.2.5 Bewertung 70
5.2.6 Wiedereinfügen 70
5.3 Der „Abstandsbasierte Bremsassistent“ 73
5.3.1 Anwendung 73
5.3.2 Codierung 75
5.3.3 Zielfunktion 81

6 Experimentelle Untersuchungen **87**
6.1 Das „Automatische Parksystem“ 88
6.1.1 Ergebnisse des evolutionären Funktionstests ohne Seeding 89
6.1.2 Ergebnisse des evolutionären Funktionstests mit Seeding 95
6.1.3 Ergebnisse des evolutionären Funktionstests mit Seeding manueller und zufälliger Werte 101
6.1.4 Ergebnisse der manuellen Tests 106
6.1.5 Ergebnisse der Zufallstests 110
6.2 Der „Abstandsbasierte Bremsassistent“ 114
6.2.1 Ergebnisse des evolutionären Funktionstests ohne Seeding 114
6.2.2 Ergebnisse des evolutionären Funktionstests mit Seeding 115
6.2.3 Ergebnisse der manuellen Tests 117
6.2.4 Ergebnisse der Zufallstests 118
6.3 Bewertung 119

7 Zusammenfassung und Ausblick **123**
7.1 Zusammenfassung 123
7.2 Ausblick 125

A Begriffsverzeichnis **127**

B Verlauf einer Optimierung **135**
B.1 Generation 1 bis 20 135
B.2 Generation 70 bis 80 155

Kapitel 1

Einführung

1.1 Motivation

Die Anzahl der in einem Fahrzeug verbauten Steuergeräte steigt und damit nimmt auch der Umfang der Software [18] ständig zu. Die Ursache dafür ist der Wunsch nach mehr Sicherheit und mehr Komfort. „In einem modernen Oberklassefahrzeug können sich durchaus 80 MByte Software auf über 70 Steuergeräte verteilen, die über Sensoren Daten erfassen, Regelalgorithmen ausführen, über Aktoren auf das Fahrzeugverhalten wirken und untereinander Daten austauschen.“ [20].

Mit der steigenden Komplexität der verteilten Rechensysteme im Auto steigt auch die Wahrscheinlichkeit für Fehler. Da ein wesentlicher Anteil der Funktionalität in Software realisiert wird, haben auftretende Fehler häufig ihre Ursache in der Software. Bei eingebetteten Systemen im Automobil kann ein Update der Software zur Fehlerbeseitigung heutzutage nur in einer Werkstatt erfolgen und kann nicht einfach online wie bei einem PC durchgeführt werden. Für einen Fehler, der nach der Auslieferung des Fahrzeugs gefunden wurde, können daher enorme Kosten entstehen. Im Falle eines sicherheitskritischen Fehlers können die Fehlerfolgen sogar katastrophal sein [69]. Zum einen können hohe Kosten für die Fehlerbehebung entstehen, wenn ganze Baureihen in die Werkstatt gebracht werden müssen. Zum anderen entsteht der Herstellerfirma durch sogenannte „Rückrufaktionen“ ein großer Image-Schaden, der langfristige Auswirkungen auf den Absatz ihrer Produkte haben kann.

Daher ist es für die Entwicklung einer Anwendungsfunktion im Automobil essentiell, möglichst alle Fehler, die bei der Entwicklung entstehen, zu finden und zu beheben, bevor das Fahrzeug produziert und ausgeliefert wird. Dabei ist es wichtig, dass ein Fehler bereits in einer frühen Entwicklungsphase behoben wird. Denn je früher ein Fehler gefunden wird, desto preiswerter ist die Fehlerbehebung ([35],[69]) und umso schneller kann die Entwicklung des Systems abgeschlossen werden. Erschwerend kommt hinzu, dass ein Fehler, der erst in einer sehr späten Entwicklungsphase gefunden wird, den Zeitdruck auf die Entwickler wesentlich erhöht und dadurch zu Folgefehlern führt.

Um qualitativ hochwertige Systeme mit wenigen Fehlern zu produzieren, sind also systematische Tests der Systeme und Subsysteme bereits in den frühen Phasen eines Projektes nötig. Diese Tests sollten außerdem automatisierbar sein, um Zeit und Kosten in einem vernünftigen Rahmen zu halten. Der manuelle Test ist aufgrund der Intuition und der Kreativität eines menschlichen Testers eine unverzichtbare Komponente beim Testen eines Systems. Jedoch sind manuelle Tests selbst zum einen aufwändig, zum anderen können sie auch unzureichend sein, da der Mensch stets etwas übersehen kann (human error). Automatisierte Tests sind hier eine sinnvolle Ergänzung des manuellen Tests.

Eine automatisierte Generierung von Testdaten ist für eine vollständige Automatisierung des Tests nicht ausreichend, da eine manuelle Bewertung der Testergebnisse dann das Nadelöhr darstellt. Wünschenswert wäre also eine automatische Generierung der Testdaten und zusätzlich eine automatische Bewertung der Testergebnisse.

Selbst mit automatisierten Tests kann die Dauer der Testläufe aufgrund der hohen Zahl der zu testenden Kombinationen „explodieren“. Grundsätzlich ist bei komplexen Systemen ein vollständiger Test nicht möglich. Daher wäre es zusätzlich wünschenswert, die Auswahl der Testdaten durch einen Algorithmus so zu optimieren, dass ein Fehler rasch aufgedeckt wird, wobei insbesondere auch solche Fehler gefunden werden sollen, die mit anderen Testmethoden schwer auffindbar sind. Ein Ansatz dazu ist, die Auswahl der Testdaten mit mathematischen Verfahren zu optimieren.

1.2 Aufgabenstellung

An dieser Stelle setzt die vorliegende Arbeit an. Ziel der Arbeit ist es, eine Testmethode zu entwickeln, mit deren Hilfe die Software von eingebetteten Systemen eines Automobils im Hinblick auf kundenerlebbare Fehler – also unter funktionalen Aspekten – automatisch und kostengünstig getestet werden kann.

Um dieses Ziel zu erreichen greift die Arbeit auf das bereits in Wegener [106] dargestellte Prinzip des evolutionären Testens[1] zurück, welches eine automatische Auswahl von Testdaten beinhaltet. Das Prinzip wird in dieser Arbeit auf das Testen von funktionalen Aspekten der Software eines eingebetteten Systems übertragen, wobei nicht nur die Auswahl von Testdaten sondern auch die Bewertung der Testergebnisse automatisiert werden soll. Ohne eine automatische Bewertung der Testergebnisse wären funktionale Tests nicht in ihrem Gesamtablauf automatisierbar. Für das Testen von funktionalen Aspekten mit evolutionären Algorithmen wird in der vorliegenden Arbeit der Begriff des „evolutionären Funktionstests“ verwendet (vgl. in [92]).

Bei dem Prinzip des evolutionären Testens wird das Problem der Auswahl von Testdaten transformiert in ein Optimierungsproblem, welches mit Hilfe eines evolutionären Algorithmus gelöst wird. Dazu müssen – wie in Abbildung 1.1 zu sehen – zwei wesentliche Aufgaben

[1] Der Begriff des evolutionären Testens wurde erstmals von Wegener und Grochtmann im Jahre 1998 eingeführt [56].

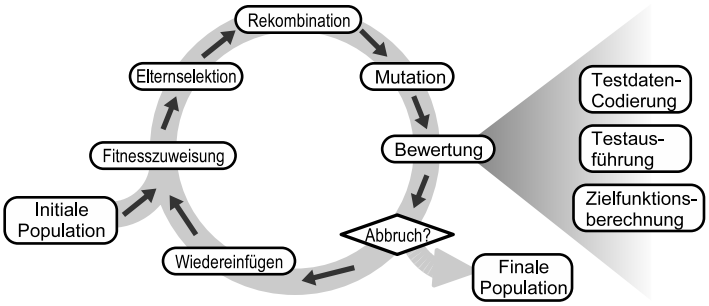


Abbildung 1.1: Ablauf eines evolutionären Tests

bewältigt werden: Es muss eine geeignete Codierung der Testdaten gefunden, sowie eine dazu passende Zielfunktion entwickelt werden.

Die Arbeit soll zusätzlich prüfen, ob die Testmethode des evolutionären Funktionstest in der Praxis von Nutzen ist. Dazu wird der evolutionäre Funktionstest auf zwei typische Beispiele aus dem Bereich der abstandsbasierten Fahrerassistenzsysteme angewendet, das „Automatische Parksystem“ und den „Abstandsbasierten Bremsassistenten“.

Bei beiden Systemen soll die vorgeschlagene Methode in Form eines prototypischen Aufbaus umgesetzt und experimentell untersucht werden. Der evolutionäre Funktionstest dient dazu, Funktionen zu testen und schwere Fehler im Entwurf und der Implementierung einer Anwendungsfunktion frühzeitig zu finden, die man sonst nicht oder sehr spät findet. Diese beiden Beispiele wurden nicht nur wegen ihrer aktuellen Bedeutung in der Automobilindustrie herangezogen, sondern auch weil sie verschiedene Aspekte erfassen. Das „Automatische Parksystem“ hat geometrische Aspekte, der „Abstandsbasierte Bremsassistent“ hat zudem auch zeitabhängige Aspekte.

Darüber hinaus soll der evolutionäre Funktionstest in den Entwicklungsprozess der Automobilindustrie eingeordnet werden. Zusätzlich soll die Leistungsfähigkeit der Methode quantitativ nachgewiesen werden, in dem sie mit den heute in der Industrie am häufigsten eingesetzten Verfahren verglichen wird – nämlich der manuellen Auswahl von Testdaten etwa durch die Klassifikationsbaummethode oder der rein zufälligen Auswahl bei den sogenannten Zufallstests.

1.3 Aufbau der Arbeit

Nach einem kurzen Überblick über die Grundlagen des Testens von Software-Programmen in Kapitel 2 enthält Kapitel 3 zum einen eine Übersicht über den Stand der Technik des Evolutionären Testens und zum anderen eine Einordnung des evolutionären Funktionstests in den Entwicklungsprozess. In Kapitel 4 wird dann die Methode des evolutionären Funktionstests vorgestellt. Für den evolutionären Funktionstest wird eine Architektur mit den folgenden Komponenten entwickelt (siehe Kapitel 4.1):

- einem evolutionären Algorithmus,
- einer Komponente „Codierung der Testdaten“,
- einer Komponente „Zielfunktion“ und
- einer Testumgebung für das Testobjekt.

Die Verknüpfung zwischen den evolutionären Algorithmen und der Begriffswelt des funktionalen Tests erfolgt zum einen durch die Codierung der Testdaten (siehe Kapitel 4.2) und zum anderen durch das Aufstellen einer Zielfunktion für die Bewertung der Testergebnisse. Für das Finden der Zielfunktion werden in dieser Arbeit die folgenden Schritte vorgeschlagen (siehe Kapitel 4.3):

- Festlegung des Optimierungsziels,
- Definition der Ergebnisklassen,
- Identifikation charakteristischer Merkmale und
- Formulieren der Zielfunktion, die ein Testergebnis auf der Basis von charakteristischen Merkmalen der Ergebnisklassen bewertet.

Kapitel 5 demonstriert den Einsatz der neuen Testmethode anhand der Beispiele „Automatisches Parksystem“ und „Abstandsbasierter Bremsassistent“. In Kapitel 6 werden die Ergebnisse der Experimente dargestellt und vor dem Hintergrund klassischer Testmethoden quantitativ bewertet. Kapitel 7 fasst die daraus gewonnenen Erkenntnisse zusammen und gibt einen Ausblick auf zukünftige Arbeiten.

Kapitel 2

Grundlagen

Testen befasst sich mit der Aufgabe, zu überprüfen, ob ein konstruiertes Produkt den Anforderungen entspricht und damit korrekt gebaut wurde oder ob es Fehler enthält. Testen hat stets zum Ziel, Fehler zu finden [49]. Man kann prinzipiell nicht die Abwesenheit von Fehlern, sondern nur ihre Anwesenheit nachweisen.

In Abbildung 2.1 ist ein Klassifikationsschema nach [78] dargestellt, in das die gängigen Software-Testmethoden eingeordnet sind.

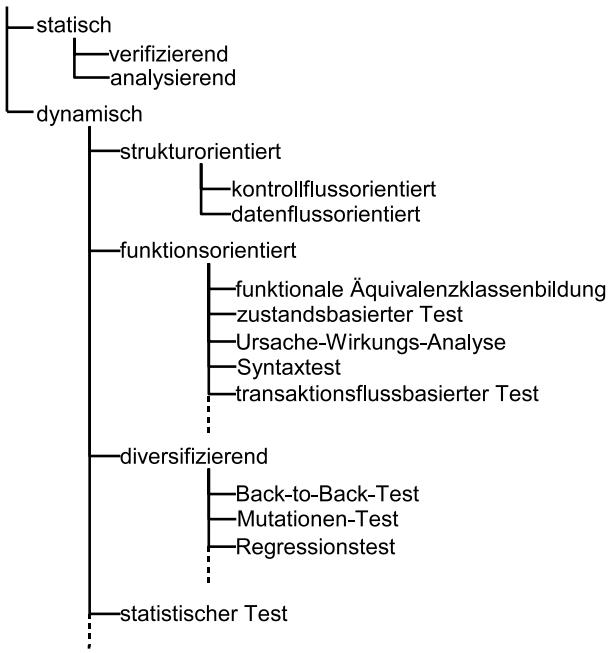


Abbildung 2.1: Klassifikation von Software-Testmethoden

Definition 2.1 (Testmethode bzw. Software-Testmethode).
Unter einer Testmethode versteht man ein planmäßiges, auf einem Regelwerk aufbauendes Vorgehen zum Finden von Testfällen. (Nach Spillner et al. in [69])

Dieses Klassifikationsschema unterscheidet auf oberster Ebene zwischen statischen und dynamischen Testmethoden und fächert sich daher in zwei Teilbäume auf. Statische Testmethoden (siehe Kapitel 2.1) befassen sich im Wesentlichen mit Review-Techniken und Code-Analysen. Bei dynamischen Testmethoden (siehe Kapitel 2.2) wählt man ein Testobjekt aus, das lediglich einen Teil oder das gesamte konstruierte Produkt umfassen kann. Anschließend führt man Testfälle auf dem gewählten Testobjekt aus, um zu untersuchen, ob es den Anforderungen genügt. Das Klassifikationsschema in [78] enthält weitere Testmethoden, wie etwa Zusicherungen (assertions), die jedoch für diese Arbeit keine Rolle spielen und daher nicht weiter betrachtet werden.

2.1 Statische Testmethoden

Charakteristisch für statische Testmethoden ist, dass hierbei keine Ausführung des zu testenden Programms erfolgt und der Quellcode somit nicht übersetzt werden muss.

Definition 2.2 (Statisches Testen).
Statisches Testen ist das Testen einer Komponente oder eines Systems auf Ebene der Spezifikation oder der Implementierung ohne Ausführung der Software, z.B. durch Reviews oder statische Codeanalyse. (Nach Veenendaal in [104])

Die statisch analysierenden Methoden werden in [23] beispielsweise unterschieden in Maße, Stilanalysen, Datenflussanomalieanalysen oder Inspektions- und Review-Techniken wie z.B. Fagan Inspection[1]. Bei Ehrenberger [13] werden diese als informelle Qualitätssicherungsverfahren bezeichnet und in Reviews, Inspektionen, Walkthroughs und Schreibtischprüfungen unterteilt. In Beizer [6] wird ein Software-Entwicklungsprozess ohne Inspektionen als „sehr fehlerhaft“ erachtet. Durch die manuelle Prüfung von Dokumenten, Modellen und Quellcode in Reviews werden oft Fehler gefunden, die durch andere Test- und Verifikationsmethoden nicht entdeckt werden. In [35] werden Untersuchungen vorgestellt, die zeigen, dass mit Methoden der Peer Reviews eine andere Art von Fehlern aufgedeckt wird als mit dynamischen Testmethoden. Myers bezeichnet in [26] Reviews als den Prozess des manuellen Testens und gibt die Empfehlung, Reviews in jedem Programmierprojekt einzusetzen. Thaller ordnet in [35] die Fagan Inspections und Code Walkthroughs den sogenannten Peer Reviews zu. Reviews von Autocode, der unter Verwendung von blockorientierten Modellierungs- und Simulationswerkzeugen [10] erzeugt wurde, kann im Vergleich zu Code-Reviews von manuell

[1] Nach [35] ist eine Fagan Inspection eine Review-Technik, die ohne den Kunden und ohne das Management durchgeführt wird. Für die Durchführung einer Fagan Inspection sind sechs Schritte definiert: Planungsphase, Einführungsveranstaltung, Vorbereitung, moderierte Inspektion, Überarbeitung und Verifikation.

geschriebenen Code mehr als eine Kombination von Modell- und Code-Review betrachtet werden [48].

Bei statisch verifizierenden Methoden unterscheidet man zwischen formal verifizierenden und symbolisch verifizierenden Techniken. Formale verifizierende Techniken setzen zwingend eine formale Spezifikation voraus. Mit ihrer Hilfe kann die Konsistenz zwischen Spezifikation und Quellcode formal bewiesen werden. Bei symbolisch verifizierenden Methoden wird der Quellcode in einer künstlichen Umgebung von einem Interpreter symbolisch ausgeführt – daher ist diese Methode streng genommen zwischen statischer Analyse und dynamischem Test einzuordnen.

Die Verwendung bestimmter statisch analysierender Methoden wird durch Standards in vielen Anwendungsbereichen gefordert. Im Anwendungsbereich der Automobilindustrie sind z.B. bei Verwendung der Programmiersprache C die MISRA Code-Konventionen [25] als Konstruktionsvorschrift für die Programme einzuhalten. Die Einhaltung dieser Konventionen wird wiederum durch statisch analysierende Prüfprogramme verifiziert. Die manuell ausgeführten, statisch analysierenden Methoden wie Inspektions- und Reviewtechniken sind eine leistungsfähige Ergänzung der anderen Methoden und haben in der Praxis eine hohe Relevanz, insbesondere zur Prüfung von Dokumenten und Anforderungen.

2.2 Dynamische Testmethoden

Die dynamischen Testmethoden umfassen strukturorientierte, funktionsorientierte, diversifizierende und statistische Testmethoden. Diese sind in den Kapiteln 2.2.1 bis 2.2.4 dargestellt.

Definition 2.3 (Dynamisches Testen).
Dynamisches Testen ist der Prozess, ein Programm (oder eine Funktionalität) mit der Absicht auszuführen, Fehler zu finden. (Nach Myers in [26])

Definition 2.4 (Testobjekt).
Ein Testobjekt ist die zu testende Komponente oder das zu testende System. (Nach Veenendaal in [104])

Definition 2.5 (Testfall).
Ein Testfall besteht aus einer Menge von Eingangsdaten, Vorbedingungen für die Ausführung, erwarteten Ergebnissen und Nachbedingungen der Ausführung, die für ein spezifisches Ziel oder eine spezifische Bedingung entworfen wurden, um einen spezifischen Programmpfad auszuführen oder die Übereinstimmung mit einer spezifischen Anforderung zu überprüfen. (Nach Veenendaal in [104])

Die Bedingungen für die Ausführung des Testfalls können – nach der Definition von Wegener in [106] – z.B. den Zeitpunkt für das Auftreten von Ereignissen spezifizieren oder auch den Zustand des Testobjekts vor der eigentlichen Testdurchführung definieren. Beim Test von eingebetteten Systemen ist insbesondere das Auslösen sogenannter Datenbedingungen im Sinne von Hatley/Pirbhai [72] wichtig.

Definition 2.6 (Vollständiger Test).
Ein vollständiger Test ist ein Testansatz, bei dem die Menge der Testfälle alle Kombinationen von Eingabewerten und Vorbedingungen umfasst. (Nach Veenendaal in [104])

Eingabewerte können beispielsweise Übergabeparameter und globale Daten im Falle von Routinen, Signale im Falle von Sensoren oder Nachrichten im Falle von verteilten Systemen sein. Ein vollständiger Test ist für die meisten Anwendungen aufgrund der hohen Zahl an Eingabewerten und der möglichen Zustände des Testobjekts in der Praxis nicht durchführbar. Testen ist also stets ein Stichprobenverfahren. Die Kunst des Testens besteht darin, alle relevanten Eingabewerte und Zustände zu finden und zu testen. Darüber hinaus ist wegen des diskreten Charakters der Software im Gegensatz zu stetigen Problemen eine Extrapolation und Interpolation von Testergebnissen nicht möglich.

Ein großes Problem ist die Frage, wie beim Testen vorgegangen werden soll, wenn man grundsätzlich bei nicht-trivialen Problemen nur punktuell testen kann. Der Entwurf der Testfälle bestimmt die Qualität des Tests, da die Auswahl der Testdaten, welche verwendet werden, um das Testobjekt zu testen, die Art, den Umfang und damit die Leistung des Tests festlegen. Falls Testfälle ausgelassen oder vergessen werden, welche für die praktische Anwendung des Systems von Bedeutung sind, sinkt die Wahrscheinlichkeit, einen Fehler zu entdecken, der sich im zu testenden System befindet. Wegen der besonderen Bedeutung des Testfallentwurfs beim Testen sind in den letzten Jahrzehnten eine ganze Reihe von Testmethoden entwickelt worden, um angemessene Testdaten auszuwählen.

2.2.1 Strukturorientierte Testmethoden

Strukturorientierte Testmethoden gehören zu den dynamischen Testmethoden, bei denen ein Programm ausgeführt wird. Bei den strukturorientierten Testmethoden erfolgt die Auswahl der Testfälle und die Bewertung der Testvollständigkeit anhand der Eigenschaften des vorliegenden Quellcodes. Bei Verwendung von blockorientierten Modellierungs- und Simulationswerkzeugen [10] können ebenso Überdeckungskriterien auf Modellebene eingesetzt werden [81]. Strukturorientierte Testmethoden gehören zu den Whitebox-Verfahren.

Definition 2.7 (Whitebox-Verfahren).
Whitebox-Verfahren sind Testmethoden, die zur Herleitung oder Auswahl der Testfälle Informationen über die innere Struktur des Testobjekts benötigen. (Nach Spillner et al. in [69])

Prinzipiell unterscheidet man zwischen kontrollflussorientierten und datenflussorientierten Methoden. Die datenflussorientierten Testmethoden nutzen den Datenfluss zur Beurteilung der Vollständigkeit der Testfälle. Die datenflussorientierten Testmethoden spielen gegenüber den kontrollflussorientierten Testmethoden in der Praxis eine untergeordnete Rolle. Nach [23] ist die praktische Nutzbarkeit datenflussorientierter Testmethoden aufgrund kaum vorhandener Werkzeugunterstützung stark eingeschränkt.

Die kontrollflussorientierten Testmethoden beurteilen die Testvollständigkeit anhand der erreichten Überdeckung der Kontrollstrukturelemente des Quellcodes bei der dynamischen Ausführung des Programms. Bei kontrollflussorientierten Testmethoden werden verschiedene Kriterien zur Messung der erreichten Überdeckung herangezogen, u.a. der Anweisungsüberdeckungsgrad, der Zweigüberdeckungsgrad oder der Bedingungsüberdeckungsgrad.

Definition 2.8 (Anweisungsüberdeckungsgrad).
Der Anweisungsüberdeckungsgrad C_0 ist definiert als das mathematische Verhältnis zwischen der Anzahl der Elemente in der Menge der ausgeführten Anweisungen und der Anzahl der Elemente in der Menge der Anweisungen des Programms.

$$C_0 = \frac{|M_{\text{Anweisungen, ausgeführt}}|}{|M_{\text{Anweisungen}}|}$$

Definition 2.9 (Zweigüberdeckungsgrad).
Der Zweigüberdeckungsgrad C_1 ist definiert als Verhältnis der Anzahl der Elemente in der Menge der durchlaufenen Zweige und der Anzahl der Elemente in der Menge der Zweige des Programms.

$$C_1 = \frac{|M_{\text{Zweige, durchlaufen}}|}{|M_{\text{Zweige}}|}$$

Der Bedingungsüberdeckungsgrad berücksichtigt die atomaren Teilbedingungen bei zusammengesetzten Entscheidungen im Kontrollfluss nach verschiedenen Kriterien wie die einfache Bedingungsüberdeckung, die Bedingungs- und Entscheidungsüberdeckung, die minimale Mehrfach-Bedingungsüberdeckung, die modifizierte Bedingungs- und Entscheidungsüberdeckung oder die Mehrfach-Bedingungsüberdeckung.

Definition 2.10 (Atomare Teilbedingung).
Eine atomare Teilbedingung ist eine Bedingung, die keine logischen Operatoren wie AND, OR oder NOT, sondern höchstens Relationssymbole wie '>' oder '=' enthält. (Nach Spillner et al. in [69])

Die einfache Bedingungsüberdeckung fordert den Test aller atomaren Teilbedingungen gegen wahr und falsch, d.h. jede atomare Teilbedingung muss durch die ausgeführten Tests zumindest einmal wahr und einmal falsch werden. Die Bedingungs- und Entscheidungsüberdeckung fordert zusätzlich zur einfachen Bedingungsüberdeckung die vollständige Zweigüberdeckung. Die minimale Mehrfach-Bedingungsüberdeckung fordert zusätzlich zur Bedingungs- und Entscheidungsüberdeckung, dass auch alle zusammengesetzten Teilbedingungen gegen wahr

und falsch getestet werden. Die modifizierte Bedingungs- und Entscheidungsüberdeckung verlangt, dass jede Teilbedingung den Wahrheitswert der Gesamtentscheidung unabhängig von den anderen Teilbedingungen beeinflussen kann. Die Mehrfach-Bedingungsüberdeckung fordert die Ausführung aller Wahrheitswertekombinationen der atomaren Teilbedingungen.

Bei den strukturorientierten Testmethoden spielen vor allem die kontrollflussorientierten Methoden für die Praxis eine wichtige Rolle. In [69] und [79] wird die Zweigüberdeckung u.a. als Kriterium für die Beurteilung der Testvollständigkeit empfohlen. Die Erfassung der während der Ausführung überdeckten Kontrollstrukturelemente beim strukturorientierten Test erfordert in der Regel eine Instrumentierung des Quelltextes.

Wenn es möglich ist, alle Teile eines Systems auf Prozedur-Ebene einem Whitebox-Test zu unterziehen, kann das ganze System damit getestet werden. Allerdings erhält man nur technische Aussagen, dass einzelne Routinen mit einem bestimmten Überdeckungsgrad getestet wurden. Ob diese Routinen im Verbund mit hoher Wahrscheinlichkeit korrekt zusammenarbeiten, kann man nicht mit Whitebox-Tests allein bewerten. Dazu ist ein funktionaler Test erforderlich, der Leistungen des Systems, die im operationellen Betrieb abgerufen werden, überprüft.

2.2.2 Funktionsorientierte Testmethoden

Funktionsorientierte Testmethoden gehören ebenfalls zu den dynamischen Testmethoden, bei denen das Testobjekt ausgeführt wird. Mit funktionsorientierten Testmethoden erfolgt der Test eines Systems oder von Systemkomponenten anhand der funktionalen Spezifikation. Diese umfasst Szenarien in der Form von Basisabläufen und Alternativabläufen. Funktionsorientierte Testmethoden gehören zu den Blackbox-Verfahren.

Definition 2.11 (Blackbox-Verfahren).
Blackbox-Verfahren sind Testmethoden zur Ableitung bzw. Auswahl von Testfällen basierend auf einer Analyse der funktionalen bzw. nicht-funktionalen Spezifikation einer Komponente oder eines Systems ohne die Einsichtnahme in die innere Struktur des Testobjekts. (Nach Veenendaal in [104])

Definition 2.12 (Funktionsorientierter Test bzw. funktionaler Test).
Ein funktionsorientierter Test basiert auf einer Analyse der funktionalen Spezifikation einer Komponente oder eines Systems. (Nach Veenendaal in [104])

Testparameter

Funktionale Tests werden durch konkrete Abrufe von Leistungen des realisierten Systems (Subsystems) im operationellen Betrieb realisiert. Wird eine Komponente im Auto getestet und ist diese nur über Nachrichten erreichbar, so stehen deren Aufrufparameter nicht in der Aufrufschnittstelle einer Funktion sondern in einer Nachricht, wie in Abbildung 2.2 gezeigt. Dies wird als Serialisierung bezeichnet und entspricht dem Aufruf einer Funktion über das Netz ohne Kenntnis der inneren Struktur eines Testobjektes.

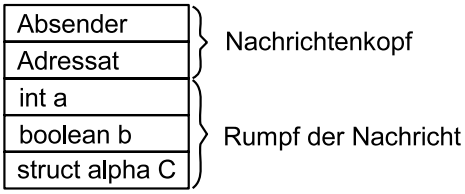


Abbildung 2.2: Schematische Darstellung einer Nachricht

Zustandsraum der Testparameter

Für die Testvollständigkeit ist es irrelevant, ob die Parameter über eine Aufrufschnittstelle oder über eine Nachricht übergeben werden. Ein vollständiger Test müsste in beiden Fällen alle Kombinationen der Datenwerte in der Schnittstelle abdecken. Für jeden Parametersatz ist ein Zustandsraum aufzubauen. Im Folgenden wird als Beispiel betrachtet, dass der Datentyp `struct alpha` eine `int-`, eine `float-` und eine `boolean`-Komponente enthält:

```
struct alpha {
  int x;
  float y;
  boolean z;
}
```

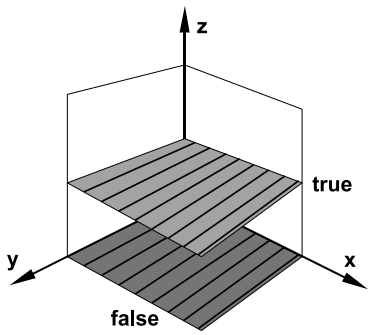


Abbildung 2.3: Beispiel für den Zustandsraum einer Struktur

Bei zusammengesetzten Datentypen wie einer Struktur ist der Zustandsraum der entsprechenden zusammengesetzten Variablen aufzuspannen. Die Abbildung 2.3 zeigt eine Darstellung des Zustandsraums der Struktur `struct alpha`. Gültige Werte sind Geraden in den Ebenen `z = true` und `z = false`, welche ganzzahlige x-Werte haben. Die y-Werte sind kontinuierlich und stellen die entsprechenden Geraden dar.

Der Zustandsraum bildet die Grundlage für das Prinzip der Äquivalenzklassen und der Klassifikationsbaum-Methode.

Äquivalenzklassen

Sind die Werte einer Variablen diskret und mit verschiedener Bedeutung wie bei Aufzählungstypen, so müssen beim Testen alle diskreten Werte abgearbeitet werden. Sind die Werte kontinuierlich wie bei `float`-Typen, so wird das Äquivalenzklassenprinzip angewandt, um den Testaufwand zu reduzieren. Dazu werden Bereiche mit derselben Semantik für eine Anwendung gesucht wie z.B. gültige Werte und ungültige Werte. Es ist wichtig bei einer Anwendung, die Semantik von Parametern zu erkennen, da beispielsweise verschiedene Bereiche von ganzen Zahlen für eine Anwendungsfunktion unterschiedliche Abläufe bedeuten können. Innerhalb des semantisch äquivalenten Bereichs wird dann ein exemplarischer Wert genommen als Stellvertreter für den gesamten Bereich. Das Äquivalenzklassenprinzip kann auch auf diskrete Werte angewandt werden, die semantisch äquivalent sind wie z.B. ganze Zahlen.

Definition 2.13 (Äquivalenzklasse).
Eine Äquivalenzklasse ist ein Teilbereich des Eingabe- oder Ausgaberaums, für welchen das Verhalten einer Komponente oder eines Systems gemäß der Spezifikation als gleichartig angenommen wird. (Nach Veenendaal in [104])

Die funktionale Äquivalenzklassenbildung unterteilt den Ein- und Ausgabedatenraum des Testobjekts in sogenannte Äquivalenzklassen. Unter einer Äquivalenzklasse versteht man einen Wertebereich, für den aufgrund der Spezifikation davon auszugehen ist, dass jeder Wert aus diesem Bereich semantisch äquivalent ist und daher vom Testobjekt funktional gleichartig bearbeitet wird. Testfälle werden anschließend so definiert, dass aus jeder Äquivalenzklasse mindestens ein Repräsentant vorkommt.

Da erfahrungsgemäß an den Grenzen zwischen zwei Bereichen oft Fehler vorkommen, wird das Prinzip der Äquivalenzklassen noch ergänzt durch die Grenzwertanalyse.

Liegt als Testparameter eine zusammengesetzte Variable vor, so kann das Prinzip der Äquivalenzklassen wieder auf diejenigen Komponenten angewandt werden, deren Werte semantisch äquivalente Bereiche haben wie `float-` oder `int`-Typen.

Klassifikationsbaum-Methode

Die Klassifikationsbaum-Methode befasst sich ebenfalls mit der Analyse des Zustandsraums der Testdaten. Testdaten können beispielsweise ein einzelnes Objekt der Objektorientierung oder aber auch komplexe Dinge wie z.B. ein Fahrszenario sein. Letztendlich geht es darum, alle Parameter und ihre Typen zu erkennen, die für den Test einer Anwendung relevant sind, damit man den Zustandsraum vollständig aufspannt. Jeder Punkt in diesem Zustandsraum stellt prinzipiell einen Satz an Testdaten dar. Die Klassifikationsbaum-Methode ist im Schema von Abbildung 2.1 nicht enthalten. Sie wird jedoch in der Praxis häufig angewendet. Nach Wegener [106] ist die Klassifikationsbaum-Methode eine Weiterentwicklung der von Ostrand und Balcer [37] vorgestellten *Category-Partition Method*. Sie wurde bereits von

Wegener und Grochtmann [55], Grochtmann und Grimm [54] und Grimm [15] ausführlich beschrieben. Eine Zusammenfassung der Methode ist in Simmes [33] zu finden.

Weitere funktionale Testmethoden

In [23] werden funktionsorientierte Testmethoden weiter unterteilt in den zustandsbasierten Test, den transaktionsflussbasierten Test, die Ursache-Wirkungsanalyse, den Syntaxtest und die Entscheidungstabellen.

Beim zustandsbasierten Test erfolgt die Testfallauswahl nach der Überdeckung in einem Zustandsautomaten. Dabei gibt es drei gängige Vollständigkeitskriterien, nämlich die Überdeckung aller Zustände, aller Zustandsübergänge oder aller Ereignisse, die zu Zustandsübergängen führen. Die Methode benötigt dazu die Spezifikation eines Zustandsautomaten. Dabei ist eine grafische Notation am übersichtlichsten.

Der transaktionsflussbasierte Test nutzt Sequenzdiagramme als Ausgangsbasis für die Erzeugung von Testfällen. Aus einem angegebenen Sequenzdiagramm werden direkt Testfälle formuliert, wobei beispielsweise durch zeitliche Verschiebung, Vertauschung oder Weglassen von Nachrichten oder Botschaften auch Fehlerfälle getestet werden.

Die Ursache-Wirkungs-Analyse wird in [26] als Cause-Effect Graphing beschrieben. Die Ursache-Wirkungs-Analyse berücksichtigt im Gegensatz zur Äquivalenzklassenbildung die Beziehungen, Wechselwirkungen und Abhängigkeiten zwischen einzelnen Äquivalenzklassen. Als Hilfsmittel definiert die Methode eine formale, grafische Sprache zur Erstellung und Darstellung eines kombinatorischen, logischen Netzwerks aus der funktionalen Spezifikation, den sogenannten Ursache-Wirkungs-Graph. Dieser wird in eine Entscheidungstabelle umgeformt, aus deren Spalten anschließend die Testfälle erzeugt werden.

Beim Syntaxtest erfolgt die Auswahl der Testfälle hinsichtlich Abdeckungskriterien in einem Syntaxdiagramm. Es wird daher eine Spezifikation in (erweiterter) Backus-Naur-Form oder in Form eines Syntaxdiagramms vorausgesetzt. Diese Methode eignet sich daher zum Test von Software zur Syntaxanalyse wie z.B. einem Parser.

Alle hier genannten Testmethoden sind dynamisch. Das Testobjekt wird mit den nach der entsprechenden Methode gefundenen Testfällen ausgeführt.

Zusammenfassung

Funktionsorientierte Testmethoden sind unverzichtbar, da sie den Gesamtablauf der Anwendungsfunktion testen. Die dynamische Ausführung von Testfällen hat den Vorteil, dass sie die Einflüsse der Betriebsumgebung mit berücksichtigt. Ein wesentlicher Vorteil der funktionsorientierten Tests ist, dass sie prinzipiell auf jeder Zerlegungsebene eingesetzt werden können. Der größte Nachteil dieser Methoden ist, dass man nur schwer automatisiert überprüfen kann, ob ein Fehler gerade auftritt oder nicht, wenn man das Sollergebnis nicht kennt.

Kennt man das Sollergebnis für die aktuellen Testdaten, kann man durch Vergleichen diese Frage beantworten. Ein Testorakel könnte als Informationsquelle dienen, um die jeweiligen Sollergebnisse zu einem Testfall zu erfahren. Nach Fewster et al. [52] ist ein Testorakel eine Quelle, welche das korrekte Testergebnis sagt. Nach Spillner et al. [69] ist ein Testorakel definiert als:

Definition 2.14 (Testorakel).
Ein Testorakel ist eine Informationsquelle zur Ermittlung der jeweiligen Sollergebnisse eines Testfalls. (Nach Spillner et al. [69])

Leider gibt es für praktische Anwendungen ein solches Testorakel meistens nicht. Um funktionale Tests trotzdem automatisieren zu können, müssen andere Ansätze verfolgt werden. Eine Möglichkeit stellen die evolutionären Funktionstests dar, die im Rahmen dieser Arbeit entwickelt werden und in Kapitel 4 vorgestellt werden.

2.2.3 Diversifizierende Testmethoden

Den diversifizierenden Testmethoden ist gemeinsam, dass konkrete Testergebnisse unterschiedlicher Testobjekte miteinander verglichen werden. Daher ist bei diesen Testmethoden eine automatisierte Bewertung der Testergebnisse relativ einfach möglich. Zu den wichtigsten diversifizierenden Testmethoden zählen der Back-to-Back-Test und der Regressionstest[2]. Der Mutationen-Test wird in dem Klassifikationsschema nach Liggesmeyer [23] ebenfalls den diversifizierenden Testmethoden zugeordnet.

Definition 2.15 (Back-to-Back-Test).
Beim Back-to-Back-Test werden zwei oder mehr Varianten einer Komponente oder eines Systems mit den gleichen Eingaben ausgeführt und deren Ergebnisse dann verglichen. Im Fall von Abweichungen wird die Ursache analysiert. (Nach IEEE 610 in [19])

Beim Back-to-Back-Test werden verschiedene Software-Varianten mit denselben Testdaten ausgeführt. Anschließend werden die Reaktionen bzw. Ausgaben der verschiedenen Testobjekte miteinander verglichen. Ein Nachteil des Back-to-Back-Tests ist die Blindheit gegenüber gemeinsamen Fehlern in den diversitären Software-Versionen. Ein Fehler wird durch einen Back-to-Back-Test nur dann erkannt, wenn er zu einem nicht identischen Verhalten zwischen den verschiedenen gewählten Varianten führt. Liegt ein Fehler in allen diversitären Software-Versionen identisch vor – z.B. aufgrund eines Fehlers in der Spezifikation – führt dies zu identischem, aber fehlerhaften Verhalten aller Varianten und wird daher mit dieser Testmethode nicht erkannt.

[2] In der Praxis ist der Regressionstest jedoch keine Testmethode. Er wird in der Praxis nicht durch Vergleich mit den Testergebnissen der letzten Version des Testobjekts durchgeführt, sondern stellt einfach einen erneuten Test nach erfolgten Code-Änderungen dar.

Definition 2.16 (Regressionstest).
Ein Regressionstest ist ein erneuter Test eines bereits getesteten Programms nach dessen Modifikation mit dem Ziel, festzustellen, dass durch die vorgenommene Änderung keine Fehler hinzugekommen sind oder (bisher maskierte) Fehler in unveränderten Teilen der Software freigelegt wurden. (Nach Veenendaal in [104])

Regressionstests sind in der Praxis unverzichtbar und von zentraler Bedeutung, um Programmierfehler nach Modifikationen weitgehend auszuschließen. Sie werden in der Regel mit denselben Testmethoden und Testwerkzeugen wie der vorangegangene Test durchgeführt.

Definition 2.17 (Mutationen-Test).
Der Mutationen-Test ist ein Back-to-Back-Test, bei dem die Varianten einer Komponente oder eines Systems durch künstliches Einfügen kleiner, definierter Modifikationen in die Originalversion der Software erzeugt werden. (In Anlehnung an Liggesmeyer in [23])

Der Mutationen-Test arbeitet wie der Back-to-Back-Test mit diversitären Software-Versionen, die sowohl manuell als auch mit Hilfe spezieller Werkzeuge erstellt werden können. Diese erzeugen veränderte Varianten der Originalversion der Software durch kleine Modifikationen, die gezielt manuell oder vom Werkzeug zufällig ausgewählt werden. Die so veränderten Software-Versionen nennt man „mutierte Version“ oder „Mutant“. Nun können die Originalversion und die mutierten Versionen der Software mit Hilfe anderer Testmethoden getestet werden. Dabei möchte man untersuchen, wie viele der Mutanten durch die jeweilige Testmethode erkannt werden. Daraus kann man wiederum die Art der durch die Testmethode gefundenen Fehler und deren Leistungsfähigkeit beurteilen. Demnach ist der Mutationen-Test ein Instrument für den Vergleich der Leistungsfähigkeit von Testmethoden und keine Testmethode im Sinne der Definition 2.1.

2.2.4 Statistischer Test

Beim Zufallstest bzw. statistischen Test wird aus der Menge der möglichen Eingabedaten per Zufallsgenerator ein gewünschter Satz von Testeingabedaten ausgewählt. Dabei ist es jedoch erforderlich, zu jeder der generierten Testeingaben die zugehörigen erwarteten Sollwerte manuell zu definieren.

Definition 2.18 (Zufallstest).
Beim Zufallstest werden die Testeingabedaten zufallsgesteuert generiert. (Nach Spillner et al. in [69])

Definition 2.19 (Statistischer Test).
Der statistische Test ist eine Testmethode, bei der ein Modell von der statistischen Verteilung der Eingabedaten verwendet wird, um repräsentative Testeingabedaten zu generieren. (Nach Veenendaal in [104])

Beim statistischen Test kann der Zufallsgenerator nach verschiedenen statistischen Wahrscheinlichkeitsverteilungen arbeiten. Eine besondere Rolle spielt hierbei das operationelle Profil (siehe Definition 2.20), mit dem man versucht, einen realen Einsatz des Testobjekts zu simulieren.

Definition 2.20 (Operationelles Profil).
Ein operationelles Profil gibt an, welche Eingabedaten im realen Betrieb wie häufig auftreten. (Nach Liggesmeyer in [23])

Ein statistischer Test entsprechend dem operationellen Profil ermöglicht es, durch statistische Analyse der Ergebnisse Aussagen über Qualitätseigenschaften – wie z.B. Zuverlässigkeit – zu erhalten [23].

Die Leistungsfähigkeit des Zufallstests bzw. des statistischen Tests gegenüber den deterministischen Testmethoden wird in der Literatur unterschiedlich bewertet. In Myers [26] wird der Zufallstest als wahrscheinlich schlechteste Methode für die Auswahl von Testdaten beschrieben. Andere Untersuchungen zeigen nach [23], dass der Zufallstest in Bezug auf seine Leistungsfähigkeit nicht deutlich schlechter ist als die deterministischen Testmethoden. Der Zufallstest wird dort ergänzend zu funktions- und strukturorientierten Testmethoden empfohlen. Nachteilig ist jedoch, dass die Sollwerte manuell definiert werden müssen.

Kapitel 3

Stand der Technik

Das Kapitel 3.1 befasst sich mit den Besonderheiten des Testens von Systemen, die in ein Automobil eingebettet sind. Dabei wird die Testumgebung erläutert, die später in Kapitel 4.1 zum Aufbau eines Testsystems für den evolutionären Funktionstest verwendet wird. In Kapitel 3.2 wird der Entwicklungsprozess in der Automobilindustrie analysiert und in das V-Modell eingeordnet. In den dort dargestellten Entwicklungsprozess wird später in Kapitel 4.4 der evolutionäre Funktionstest eingefügt. In Kapitel 3.3 wird die Methode des evolutionären Testens und der aktuelle Stand der Technik vorgestellt. Der Bedarf für eine neue Methode wird anschließend in Kapitel 3.4 dargelegt.

3.1 Testen von im Automobil eingebetteten Systemen

Als eingebettete Systeme (engl.: embedded systems) bezeichnet man im Allgemeinen Teilsysteme, die in größere Systeme oder Umgebungen integriert sind und in der Regel aus Hardware und Software bestehen. Eingebettete Systeme werden für spezielle Anwendungen entworfen und steuern oder regeln häufig ein mechanisches System. Das Programm läuft auf einem Mikrocontroller, welcher Signale von Sensoren verarbeitet und Aktoren ansteuert. Ein eingebettetes System soll nicht nur die gewünschten Funktionen zur Verfügung stellen sondern darüber hinaus auch Anforderungen vor allem bezüglich der gewünschten Leistung, der Kosten, der Zuverlässigkeit, der Sicherheit und des Energieverbrauchs usw. erfüllen. [30]

Ein eingebettetes System im Automobil ist nur ein Teil des Fahrzeugs und es muss daher im Kontext des umgebenden Systems getestet werden. Aus diesem Grund unterscheidet sich der funktionale Test eines eingebettetes Systems vom funktionalen Test beispielsweise einer Desktop-Anwendung.

3.1.1 Testen eingebetteter Systeme im Fahrzeug

Eingebettete Systeme im Fahrzeug bestehen aus Hardware und aus Software. Daher müssen

Hard- und Software getestet werden. Nach erfolgtem Hardware-Test kann die Software im Rahmen eines Funktionstests im Fahrzeug getestet werden. Bei dieser Art des Testens befindet sich das eingebettete System im Fahrzeug und interagiert mit dem Fahrzeug. Wie bei Yap [110] dargestellt, nimmt der Fahrer über Gaspedalstellung, Bremspedalstellung, Lenkradeinschlag, usw. Einfluss auf das Fahrzeug. Fahrzeug und Fahrer interagieren mit der Fahrzeugumgebung. So sieht z.B. ein Fahrer in seinem Sichtfeld vorausfahrende Fahrzeuge oder ein ACC-Sensor[1] kann vorausfahrende Fahrzeuge detektieren [109]. Abbildung 3.1 zeigt schematisch die Testsituation auf.

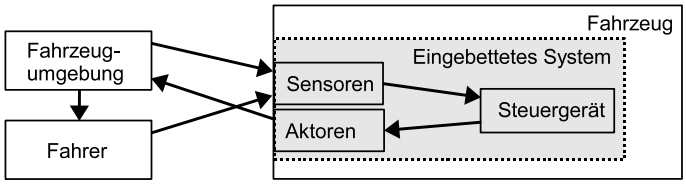


Abbildung 3.1: Interaktion zwischen Fahrzeugumgebung, Fahrer, Fahrzeug und eingebettetem System.

Um ein eingebettetes System im Automobil auf Fahrzeugebene zu testen, benötigt man zumindest ein Fahrzeug, einen Fahrer und eine geeignete Fahrzeugumgebung, d.h. in der Regel eine Versuchsstrecke. Daher ist ein Fahrzeugtest mit realem Fahrzeug (Versuchsträger) sehr teuer und die Testdurchführung ist zeitaufwändig. Im Allgemeinen werden für funktionale Tests die Fahrmanöver manuell erstellt nach Analyse und Interpretation der Spezifikation und unter Hinzunahme des Applikationswissens. Für die Testausführung der so erstellten Fahrmanöver wird eine gewünschte Situation in einer speziell dazu präparierten Fahrzeugumgebung von einem Fahrer gefahren. Das spezifizierte erwartete Verhalten wird mit dem tatsächlichen Verhalten des Fahrzeugs verglichen.

3.1.2 Testen eingebetteter Software in Simulationsumgebungen

Da Tests im Fahrzeug kostspielig und im fahrdynamischen Grenzbereich gefährlich sind, wird der Test der Software eines eingebetteten Systems im Fahrzeug oft in einer Simulationsumgebung ohne das reale Fahrzeug durchgeführt. Zur Durchführung solcher Tests muss der Kontext des umgebenden Systems aus Sicht der zu testenden eingebetteten Software simuliert werden. Dabei wird die eingebettete Software als das Testobjekt betrachtet, das in einer dazu geeigneten Testumgebung getestet wird. Dies bedeutet im Fall von abstandsbasierten Fahrerassistenzsystemen, dass der Fahrer, die Fahrzeugumgebung und das Rest-Fahrzeug simuliert werden müssen. Die Simulation muss dabei so ausgelegt sein, dass die eingebettete Software damit betrieben werden kann.

Eine Testumgebung besteht, wie in Abbildung 3.2 dargestellt, aus der eingebetteten Software und einer Simulationsumgebung, in Form von Fahrer- und Fahrzeugsimulation sowie

[1] ACC – Adaptive Cruise Control, Sensor zur Erkennung vorausfahrender Fahrzeuge

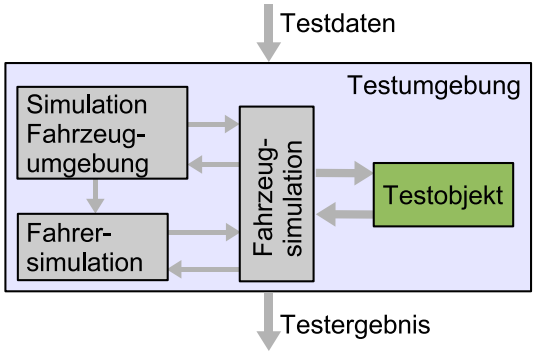


Abbildung 3.2: Testumgebung mit Simulationsumgebung und eingebetteter Software

einer Simulation der Fahrzeugumgebung. Die Ein- und Ausgänge der eingebetteten Software sind mit der Fahrzeugsimulation verbunden. Diese simuliert das Rest-Fahrzeug aus der Sicht der eingebetteten Software inklusive aller relevanten Sensoren und Aktoren mit ihren Wechselwirkungen zum Testobjekt. Die Fahrzeugsimulation interagiert mit der Fahrersimulation und der Simulation der Fahrzeugumgebung.

Die Testdaten werden nicht direkt für die Schnittstelle des Testobjekts formuliert, sondern als Eingabedaten für die Testumgebung. Diese Testdaten beschreiben für die Testumgebung alles, was von der eingebetteten Software nicht beeinflusst werden kann.[2] Die eingebettete Software kann z.B. das Verhalten der Fahrzeuge in der Fahrzeugumgebung nicht beeinflussen, daher muss dieses Verhalten durch die Testdaten definiert werden.

Das Testergebnis ist das simulierte Szenario, als Ergebnis der Interaktion zwischen Testobjekt und Simulationsumgebung. Als Testergebnis interpretiert man die Gesamtheit aller Vorgänge, die direkt oder indirekt durch die eingebettete Software ausgelöst oder beeinflusst wurde. So kann z.B. die eingebettete Software das simulierte Fahrzeug lenken oder abbremsen. Dies ist dann die Reaktion des Testobjekts auf eine vorgegebene Situation.

Ein Problem bei dieser Art des Testens ist, dass die Ursache für ein fehlerhaftes Verhalten auch an einem Fehler in der Simulationsumgebung liegen kann. Ist die Simulationsumgebung für das Testobjekt und die Testdaten nicht geeignet oder arbeitet sie fehlerhaft, kann dies zu einem fehlerhaften funktionalen Verhalten in der Simulation führen, obwohl das Testobjekt fehlerfrei arbeitet.

Ein wichtiger Faktor für die Durchführung von Tests in Simulationsumgebungen ist daher auch die Qualität der Simulation. Entscheidend ist dabei, dass eine Simulation stets ein Modell der Realität darstellt und somit nicht alle Details beliebig genau abbilden kann. Das Modell für die Simulation ist so zu wählen, dass wichtige Details genau genug abgebildet

[2] Dies sind Eigenschaften der Fahrzeugumgebung wie z.B. Fahrbahn oder umgebende Fahrzeuge, das Verhalten des Fahrers oder Eigenschaften des Fahrzeugs wie z.B. eine bestimmte Fahrzeugmasse durch Beladung.

werden und die unwichtigen Details vernachlässigt werden. In anderen Worten, ein Modell ist ein Abbild der Realität [80]. Die Herausforderung liegt also darin, die wichtigen Details von den unwichtigen zu trennen.

3.2 Entwicklungsprozess in der Automobilindustrie

Die Technologieinnovation in der Fahrzeugentwicklung betrifft inzwischen zu einem Großteil sogenannte E/E/PE-Systeme[3], wobei die kundenerlebbare Funktionalität[4] vollständig oder teilweise durch Software realisiert wird [40]. Die Realisierung einer Anwendungsfunktion für ein Automobil erfolgt heutzutage im Normalfall mit Hilfe von eingebetteten Systemen. D.h. die sogenannten E/E/PE-Systeme enthalten eingebettete Systeme aus Hard- und Software. Die kundenerlebbare Funktionalität wird zu einem großen Teil durch Software zur Verfügung gestellt, die auf einem Mikrocontroller in einer speziellen Hardware-Umgebung läuft.

An diese sogenannten E/E/PE-Systeme werden besondere Anforderungen bezüglich Hardware und Software gestellt, wie z.B. Robustheit, Verfügbarkeit, Zeitverhalten, Kosten, Gewicht, Bauraum, Energieverbrauch, usw. Diese besonderen Anforderungen bestimmen die Art des Entwicklungs- und des Testprozesses maßgeblich mit. Sie müssen umgesetzt werden bei begrenzten Kosten, verkürzter Entwicklungszeit und zunehmender Variantenvielfalt [103]. Die Entwicklung einer Anwendungsfunktion für ein Fahrzeug mit Hilfe eines eingebetteten Systems ist durch eine Reihe von Besonderheiten charakterisiert:

- Die Anforderungen an die Anwendungsfunktionen beziehen sich auf das Gesamtsystem, also an das Fahrzeug als Ganzes.
- Die Entwicklungsaktivitäten selbst beziehen sich auf eine Komponente[5] des Fahrzeugs, die in das Fahrzeug eingebaut und vernetzt wird und damit ein eingebettetes System darstellt. Das restliche, umgebende System ist in der Regel von außen vorgegeben.
- Die zu entwickelnde Komponente hat in der Regel einen hohen Vernetzungsgrad mit anderen Komponenten und damit eine hohe Zahl von Abhängigkeiten.
- In den frühen Phasen der Entwicklung können sich, z.B. durch Änderung der Anforderungen, die Kommunikationsschnittstellen von Komponenten ändern. Solche Änderungen haben Rückwirkungen auf andere Komponenten, welche diese Schnittstellen

[3] Nach DIN EN 61508-4 [12] ist ein E/E/PE-System ein System zur Steuerung, zum Schutz oder zur Überwachung, basierend auf einem oder mehreren elektrischen/elektronischen/programmierbaren elektronischen Geräten, einschließlich aller Elemente des Systems wie z.B. Energieversorgung, Sensoren und anderen Eingabegeräten, Datenverbindungen und anderen Kommunikationswegen sowie Aktoren und anderen Ausgabeeinrichtungen.

[4] In dieser Arbeit als „Anwendungsfunktion“ bezeichnet.

[5] Unter Komponente sollen hier die Teile eines Fahrzeugs verstanden werden, die beim Zusammenbau am Band als Einzelteile angeliefert werden. In diesem Fall ist mit Komponente das gesamte Steuergerät gemeint.

benutzen. Ziel ist es, die Kommunikationsschnittstellen der Komponenten möglichst rasch zu stabilisieren.

- Aufgrund der besonderen, nicht-funktionalen Anforderungen an Hardware und Software müssen beide maßgeschneidert entwickelt werden.

Die nicht-funktionalen Anforderungen haben zur Konsequenz, dass die endgültige Ausführungsplattform zu Beginn der Entwicklung der Software noch nicht zur Verfügung steht und deren Eigenschaften wie z.B. der verwendete Zielprozessor gar nicht bekannt sind. Umgekehrt sind die Anforderungen der Software an die Hardware wie z.B. der benötigte Speicherplatz in Form von RAM und ROM zu Beginn der Hardware-Entwicklung auch nicht bekannt. Dies führt dazu, dass die Software zu Beginn der Entwicklung nicht auf der eigentlichen Zielhardware getestet werden kann, sondern in einer dafür vorgesehenen Simulationsumgebung, z.B. auf dem PC (siehe Kapitel 3.1.2). Solche Simulationsumgebungen eröffnen die Möglichkeit, bereits in einer frühen Phase eines Projektes das funktionale Verhalten der Programme zu testen.

In der nachfolgenden Abbildung 3.3 ist der Entwicklungsprozess einer Anwendungsfunktion in der Automobilindustrie anhand des V-Modells in der Produktsicht dargestellt. Die nachfolgenden Kapitel 3.2.1 und 3.2.2 beschreiben anhand dieser Darstellung den Entwurfsprozess (siehe Kapitel 3.2.1) und den Integrations- und Testprozess (siehe Kapitel 3.2.2). Diese Darstellung dient später in Kapitel 4.4 als Grundlage für die Einordung des evolutionären Funktionstests in den Entwicklungsprozess.

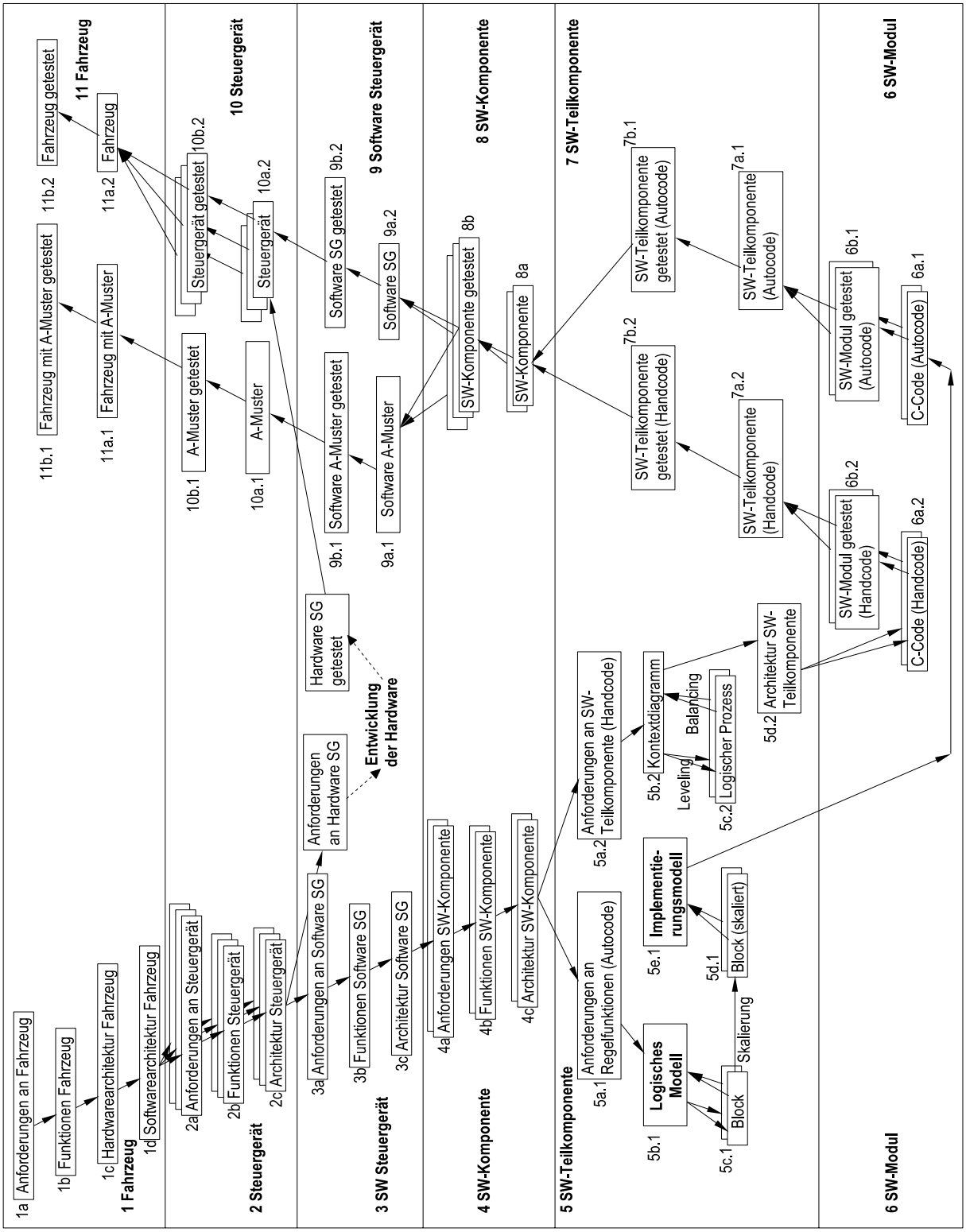


Abbildung 3.3: Produkte der Systementwicklung nach dem V-Modell.

3.2.1 Entwurfsprozess

Da die „wahren Anforderungen“ erst durch einen Prototyp erkannt werden müssen und die Grenzen zwischen Hard- und Software zu Beginn eines Projektes noch nicht festgelegt sind, ist eine iterative Entwicklung erforderlich. Nach Balzert [4] werden Prototypen verwendet, um relevante Anforderungen oder Entwicklungsprobleme zu klären, als Diskussionsbasis bei Entscheidungen und um praktische Erfahrungen zu sammeln. Eine lineare Vorgehensweise entsprechend dem Wasserfallmodell von Royce [31] und Boehm [7] wäre hier nicht praktikabel, da viele Einflussfaktoren und Randbedingungen erst während der Entwicklung bekannt werden.

Die iterative Vorgehensweise ist zu Projektbeginn gekennzeichnet durch Simulation und Rapid Prototyping [71],[42],[43]. Mit Rapid Prototyping werden neue Konzepte und Ideen für Anwendungsfunktionen mit regelungs- und steuerungstechnischen Anteilen in frühen Phasen der Entwicklung effizient getestet [50]. Diese Strategie ermöglicht es, dass die Software des Prototyps unabhängig von der Hardware entwickelt werden kann [76]. Erst dann, wenn die Hardware entwickelt wurde und existiert, kann die Software auf die Zielhardware portiert werden.

Das Ziel der prototypischen Realisierung ist eine möglichst frühzeitige Umsetzung und Darstellung einer geforderten Anwendungsfunktion im Gesamtsystem Fahrzeug zur Validierung der Anforderungen und als Nachweis der Machbarkeit. Aufgrund der Erkenntnisse und Erfahrungen mit diesen Prototypen ändern sich nachfolgend als Konsequenz in der Regel die Anforderungen an die Anwendungsfunktion.

Das in der Automobilindustrie in der Praxis weit verbreitete V-Modell stellt – wie bereits das Wasserfallmodell – auf den ersten Blick eine rein sequenzielle Vorgehensweise dar. Jedoch unterstützt das V-Modell das iterative Vorgehen ebenfalls, indem es erlaubt, von späteren Aktivitäten beliebig in frühere Aktivitäten zurückzuspringen.

Die Abbildung 3.3 zeigt links die Produkte der Zerlegungshierarchie für ein V-Modell, welches angepasst ist auf eine Entwicklung, die Autocode[6] und Handcode umfasst. In Kapitel 4.4 wird darin dargestellt, welche Produkte im Entwurfsprozess mit dem evolutionären Funktionstest getestet werden können.

Die Entwicklung beginnt bei den Anforderungen an Fahrzeug (1a) mit der Definition der Anforderungen an das Fahrzeug auf oberster Ebene. Diese Anforderungen spezifizieren das gewünschte Verhalten eines Fahrzeugs, welches die zu entwickelnden Anwendungsfunktionen enthält. Dabei steht das durch den Fahrer erlebbare Verhalten des Fahrzeugs im Vordergrund. Die Anforderungen sollen nach [77] so spezifiziert sein, dass sie nachprüfbar sind. D.h. bei der Anforderungsdefinition müssen sofort auch die Fahrmanöver auf Ebene des Fahrzeugs spezifiziert werden. Die Funktionen Fahrzeug (1b) werden im Rahmen der Systemanalyse des Fahrzeugs für jede Anwendungsfunktion, die für den Fahrer erlebbar ist, ermittelt. Dabei wird festgelegt, welche logischen Systemfunktionen erforderlich sind und

[6] Autocode wird automatisch aus blockorientierten Modellierungs- und Simulationswerkzeug generiert.

wie diese zusammenwirken müssen, damit der Fahrer die geforderte Anwendungsfunktion erhält. In der Hardwarearchitektur Fahrzeug (1c) ist die Topologie des Netzwerks aller Steuergeräte im Fahrzeug festgelegt. Dabei ist auch definiert, welche Steuergeräte es im Fahrzeug gibt. Bei der Softwarearchitektur Fahrzeug (1d) werden die logischen Systemfunktionen den Steuergeräten zugeordnet. Eine Anwendungsfunktion kann dabei durch ein Steuergerät allein oder durch ein Verbund von Steuergeräten zur Verfügung gestellt werden. Durch diese Softwarearchitektur wird auch festgelegt, welche Botschaften ein Steuergerät senden und empfangen muss. Müssen mehrere Steuergeräte eng zusammenarbeiten, so werden sie an denselben CAN-Bus gelegt. Dies erfolgt nach den Prinzipien des Software Engineerings „Strong Coherence“ und „Loose Coupling“ [32].

Welche Funktionalität ein einzelnes Steuergerät bereitzustellen hat, ist in den Anforderungen an Steuergerät (2a) festgehalten. Aus der nachfolgenden Systemanalyse auf der Ebene Steuergerät resultieren die Funktionen Steuergerät (2b). Beim Systementwurf des Steuergeräts wird die Architektur Steuergerät (2c) festgelegt. Dabei wird festgelegt, was in Hardware und was in Software realisiert wird. Dann teilt sich die Entwicklung in Hardware und Software auf.

Bei der Anforderungsdefinition an die Software des Steuergeräts werden die Anforderungen an Software SG (3a) festgelegt. Das Ergebnis der Systemanalyse der Software sind die Funktionen Software SG (3b), welche die logischen Systemfunktionen und ihre Kommunikation im Steuergerät festgelegen. Die Architektur Software SG (3c) definiert, in welche Betriebssystemprozesse und Interrupt-Service-Routinen die Software des Steuergeräts zerlegt wird und wie diese Software-Komponenten zusammenwirken. Außerdem wird das Betriebssystem festgelegt. Dieser Schritt entscheidet auch, ob es sich um das A-Muster oder das eigentliche eingebettete System handelt.

Anschließend müssen die Anforderungen SW-Komponente (4a) formuliert werden. Die Funktionen SW-Komponente (4b) sind das Ergebnis der Systemanalyse der SW-Komponente. Hier wird zunächst analysiert, welche Klassen von Funktionen das Steuergerät beinhalten soll. Solche Klassen sind typischerweise:

- Signalvorverarbeitung
- verschiedene Regleraufgaben
- Kommunikation
- Datenspeicherung (FRAM)
- Start-up und Shut-down
- Fehlerbehandlung
- Betriebssystem
- Systemtreiber

Die Architektur SW-Komponente (4c) legt fest, was klassisch und was modellbasiert, d.h. mit blockorientierten Modellierungs- und Simulationswerkzeugen [10], entwickelt wird. Beinhaltet das Steuergerät Regelfunktionen, so werden diese oftmals modellbasiert entwickelt. Im Kontext der modellbasierten Entwicklung werden diese logischen Systemfunktionen „Blöcke“ genannt, während sie im Rahmen einer konventionellen Entwicklung als „logische Prozesse“ bezeichnet werden.

Alle anderen Funktionen außer den Regelfunktionen werden üblicherweise konventionell entwickelt. Abbildung 3.3 zeigt den Fall, dass ein Betriebssystemprozess sowohl Regelfunktionen als auch andere Funktionen enthält. Daher werden durch die Anforderungen Regelfunktionen (Autocode) (5a.1) bzw. Anforderungen Regelfunktionen (Handcode) (5a.2) die Anforderungen an die Regelfunktionen und die Anforderungen an die restlichen Funktionen des Steuergerätes getrennt dargestellt. Enthält der Betriebssystemprozess keine Regelfunktion, so ist (5a.2) identisch zu (4a). Die Modellierung im Rahmen der Systemanalyse einer Software-Teilkomponente ist auf der Ebene (5b.1) bzw. (5b.2) dargestellt.

Bei Verwendung von blockorientierten Modellierungs- und Simulationswerkzeugen werden die Datenflüsse in und aus der Software-Teilkomponente analysiert und im Logischen Modell (5b.1) dargestellt. Anschließend erfolgt eine rekursive Zerlegung (Levelling) in einzelne, signalverarbeitende Blöcke (5c.1). Das Modellierungs- und Simulationswerkzeug unterstützt und überwacht das Balancing bei der Integration des logischen Modells. Bei der Skalierung der Blöcke (skaliert) (5d.1) werden den Signalflüssen des logischen Modells geeignete Datentypen, Wertebereiche und Auflösungen zugeordnet. Anschließend werden in das Implementierungsmodell (5e.1) die skalierten Blöcke integriert. Danach wird aus dem Implementierungsmodell mit Hilfe eines Code-Generators in der Regel C-Code generiert.

Im Kontextdiagramm (5b.2) werden die Datenflüsse in und aus der SW-Teilkomponente analysiert. Dann erfolgt eine rekursive Zerlegung (Levelling) in einzelne, Logische Prozesse (5c.2). Durch das Balancing dieser logischen Prozesse entsteht das logische Modell. Anschließend erfolgt der Entwurf der Architektur SW-Teilkomponente (5d.2). Im Rahmen der Codierung werden schließlich die SW-Module als C-Code (Handcode) (6a.2) implementiert.

3.2.2 Integrations- und Testprozess

In Abbildung 3.3 sind die Produkte der Systemintegration nach dem V-Modell rechts dargestellt. Auf jeder Integrationsebene wird das entsprechende Produkt gegen die Anforderungsdefinition geprüft.

Der automatisch generierte C-Code (Autocode) (6a.1) wird zunächst auf Software-Modulebene getestet. Ebenso werden die nicht modellbasiert entwickelten Teile (6a.2) getestet, so dass anschließend die getesteten SW-Module (6b.1) und (6b.2) vorliegen. Danach erfolgt die Integration der SW-Teilkomponenten (7a.1) und (7a.2), welche dann separat getestet werden (7b.1 und 7b.2). Die entstandenen SW-Teilkomponenten werden nun zu einer SW-Komponente (8a) integriert und anschließend getestet (8b).

Typisch für die Automobilindustrie ist, dass iterativ mit Prototypen gearbeitet wird. Bei DaimlerChrysler gibt es z.B. A-Muster, B-Muster, C-Muster usw. Die getesteten SW-Komponenten können zu Beginn der Fahrzeugentwicklung nicht auf dem zukünftigen Zielsteuergerät integriert werden, da dieses sich noch in der Hardware-Entwicklung befindet. Sie werden zunächst auf einem PC mit einem Realzeit-Betriebssystem integriert, dem sogenannten A-Muster-Rechner (siehe Abbildung 3.3).

Zunächst werden die SW-Komponenten (8b) zusammen mit dazu vorgesehenen SW-Komponenten für das Betriebssystem und die Hardware-Treiber des A-Muster-Rechners zu einer ablauffähigen Software A-Muster (9a.1) integriert und danach getestet (9b.1). Anschließend erfolgt die Integration der getesteten A-Muster-Software auf den A-Muster-Rechner zum lauffähigen A-Muster (10a.1), das nun getestet werden kann. Nach der Integration in ein Fahrzeug kann auf Testfahrten im Fahrzeug (10b.1, 11a.1 und 11b.1) getestet werden.

Basierend auf den Erkenntnissen der Testfahrten wird nachfolgend die Software auf das endgültige Zielsteuergerät portiert (siehe Abbildung 3.3). Die getesteten Software-Komponenten (8b) werden zu einer ablauffähigen Software SG (9a.2) integriert und getestet (9b.2). Die getestete Software SG wird danach in das Steuergerät (10a.2) integriert und getestet (10b.2). Das getestete Zielsteuergerät kann danach in das Fahrzeug (11a.2) integriert werden. Abschließend kann das gesamte Fahrzeug getestet werden (11b.2).

3.3 Evolutionäres Testen

Beim evolutionären Testen wird das Problem der Auswahl von Testdaten in ein Optimierungsproblem transformiert. Das entstandene Optimierungsproblem wird dann mit Hilfe eines evolutionären Algorithmus „gelöst“. Evolutionäres Testen wurde in der Praxis bereits erfolgreich auf mehreren Gebieten eingesetzt.

Evolutionäre Algorithmen werden in dieser Arbeit für die Generierung von Testdaten verwendet, da ihre Robustheit und ihre Tauglichkeit für die Lösung verschiedener Aufgaben beim Test bereits in mehreren Arbeiten gezeigt wurde. Wegen „ihrer Spezialisierung auf die Lösung von komplexen Optimierungsproblemen mit vielen Dimensionen und wenig verstandenen Suchräumen“ wurden von Wegener in [106] evolutionäre Algorithmen für den Test des Zeitverhaltens verwendet. Nach McMinn [24] kann evolutionäres Testen eingesetzt werden für systematische und automatisierte Tests von nicht-funktionalen Eigenschaften genauso wie für automatisch generierte Tests für konventionelle Testmethoden wie z.B. Strukturtests.

Evolutionäres Testen basiert auf der Verwendung von evolutionären Algorithmen als metaheuristisches Suchverfahren für die Generierung von Testdaten. Das Schlüsselelement, um evolutionäres Testen bei Software-basierten Systemen anzuwenden, ist die Definition einer passenden Zielfunktion. In Abhängigkeit vom Optimierungsziel muss eine geeignete Zielfunktion vorgegeben werden, um die Testergebnisse zu bewerten.

3.3.1 Evolutionäre Algorithmen

Nach Branke [8] sind evolutionäre Algorithmen iterative, stochastische Optimierungsheuristiken, deren Funktionsweise der natürlichen Evolution nachempfunden ist. Sie basieren auf dem Prinzip des „Survival of the Fittest“ [11].

Der evolutionäre Algorithmus wird in mehreren Iterationen ausgeführt. Dabei simuliert er in jeder Iteration einen evolutionären Zyklus und berechnet damit eine neue Generation der Population. In der nachfolgenden Abbildung 3.4 ist die Funktionsweise als Kreisprozess dargestellt. Der Ablauf startet mit einer initialen Population von Individuen[7]. Anschließend beginnt der iterative Zyklus mit den Schritten: Fitnesszuweisung, Elternselektion, Rekombination, Mutation, Bewertung und Wiedereinfügen. Nach Erfüllung des Abbruchkriteriums endet die Berechnung mit der finalen Population.

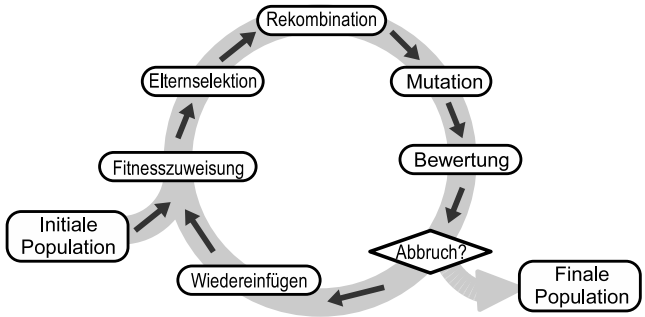


Abbildung 3.4: Ablauf eines evolutionären Algorithmus

Bei der Fitnesszuweisung wird zu jedem Individuum jeweils ein Fitnesswert berechnet, basierend auf den Zielfunktionswerten aller Individuen der Population. Die Ausführung dieses Schrittes ist abhängig von der verwendeten Art der Elternselektion und des Wiedereinfügens: Werden bei der Elternselektion und beim Wiedereinfügen keine Fitnesswerte benötigt, da z.B. die Auswahl rein stochastisch erfolgt bzw. direkt mit den Zielfunktionswerten gearbeitet wird, kann dieser Schritt entfallen.

Der Schritt Elternselektion wird in der Literatur auch häufig nur als „Selektion“ bezeichnet. Weicker weist in [108] allerdings darauf hin, dass nicht nur in diesem Schritt eine Selektion stattfindet und unterscheidet zwischen „Elternselektion“ und „Umweltselektion“[8]. Die Elternselektion wählt eine Anzahl von Individuen aus der Population in Generation n aus. Diese werden zu der Menge der selektierten Individuen übernommen und dann zur Bildung von Nachkommen herangezogen. Bei der Auswahl der Eltern kann basierend auf deren

[7] Bei der Darstellung in Abbildung 3.4 wird davon ausgegangen, dass für die nachfolgende Fitnesszuweisung zu jedem Individuum der initialen Population bereits eine Bewertung stattgefunden hat und somit der jeweilige Zielfunktionswert bereits bekannt ist.

[8] Die „Umweltselektion“ nach Weicker [108] findet im Schritt Wiedereinfügen statt, wenn zwischen Nachkommen und Eltern zur Übernahme in die nächste Generation ausgewählt wird.

Fitnesswerten, deren Zielfunktionswerten oder rein stochastisch vorgegangen werden. Werden zur Auswahl die Fitnesswerte oder die Zielfunktionswerte herangezogen, so entsteht an dieser Stelle ein Selektionsdruck (selection pressure).

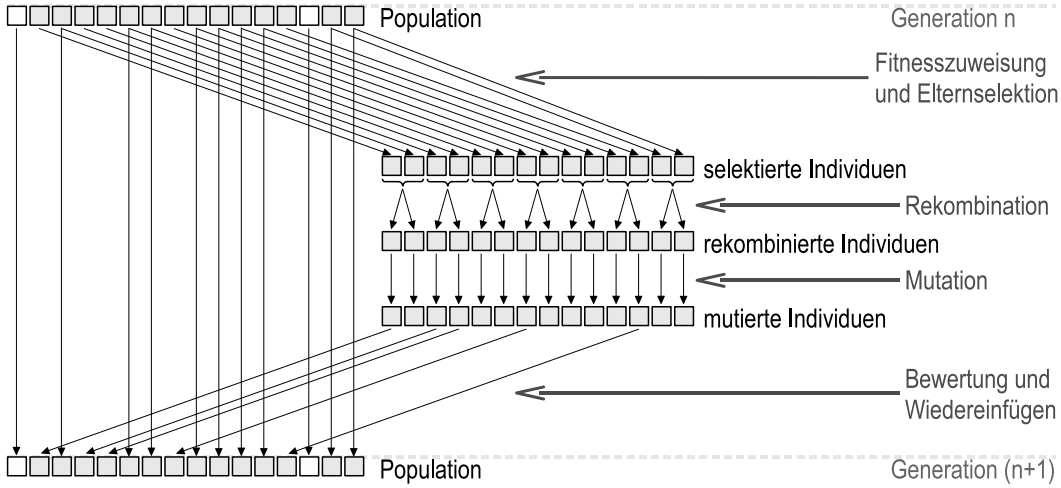


Abbildung 3.5: Schema zur Berechnung der Generation $(n + 1)$ aus Generation n

Bei der Rekombination werden aus den zuvor selektierten Individuen neue Nachkommen erzeugt. Diese werden in Abbildung 3.5 als rekombinierte Individuen bezeichnet. Es werden jeweils zwei Individuen zweimal rekombiniert, so dass aus zwei Eltern genau zwei Nachkommen hervorgehen. Dabei wird formal jedem Elternteil eines der Nachkommen zugeordnet.

Bei der Mutation wird in den neu gebildeten Nachkommen jeweils eine gezielte Veränderung vorgenommen. Die Häufigkeit und Art der Veränderung kann mit deterministischen oder stochastischen Operatoren erfolgen. Dabei ist nicht gesagt, dass jedes rekombinierte Individuum in diesem Schritt auch verändert wird. D.h. es gibt mutierte Individuen, die unverändert aus der Menge der rekombinierten Individuen übernommen werden.

Bei der Bewertung wird zu jedem Individuum aus der Menge der mutierten Individuen sein jeweiliger Zielfunktionswert berechnet. Hier erfolgt die Ausführung der Testdaten mit dem Testobjekt beim evolutionären Funktionstest. Als Ergebnis dieser Operation ist anschließend zu jedem mutierten Individuum sein zugehöriger Zielfunktionswert bekannt.

Das Wiedereinfügen sortiert nun einzelne Individuen aus der bestehenden Population aus und fügt ausgewählte Nachkommen in die Population ein. Dabei werden genau so viele „alte“ Individuen entfernt, wie „neue“ Individuen in die Population eingefügt werden, so dass die Anzahl von Individuen in der Population konstant bleibt. Dabei ist nicht gesagt, dass alle neu erzeugten Nachkommen auch in die Population der nächsten Generation eingefügt werden. Da sowohl die zu entfernenden Individuen als auch die neu einzufügenden Individuen jeweils ausgewählt werden, entsteht auch an dieser Stelle ein Selektionsdruck (selection pressure). Daher wird dieser Schritt in [108] als „Umweltselektion“ bezeichnet.

Die Abbildung 3.5 zeigt die Arbeitsweise der verschiedenen Operationen nacheinander schematisch. Dargestellt ist die Berechnung einer neuen Population für die Generation $(n + 1)$ aus der vorangegangenen Population aus der Generation n.

3.3.2 Prinzip des evolutionären Testens

Der Ablauf beim evolutionären Testen beginnt mit einer Vorgabe des evolutionären Algorithmus. Dieser generiert ein Individuum mit einer vorher festgelegten Struktur seiner Gene[9]. Das Individuum wird anschließend in Testdaten übersetzt (siehe Abbildung 3.6). Diese Testdaten werden dann mit dem Testobjekt ausgeführt. Das Testergebnis wird zusammen mit den Testdaten von der Zielfunktion bewertet. Der berechnete Zielfunktionswert wird an den evolutionären Algorithmus zurückgemeldet und dort dem Individuum zugeordnet, das diese Testdaten erzeugt hat. Der komplette Ablauf wiederholt sich für jedes Individuum einer Generation. Wurde allen Individuen einer Generation ein Zielfunktionswert zugeordnet, beginnt der evolutionäre Algorithmus, eine neue Generation von Individuen zu berechnen.

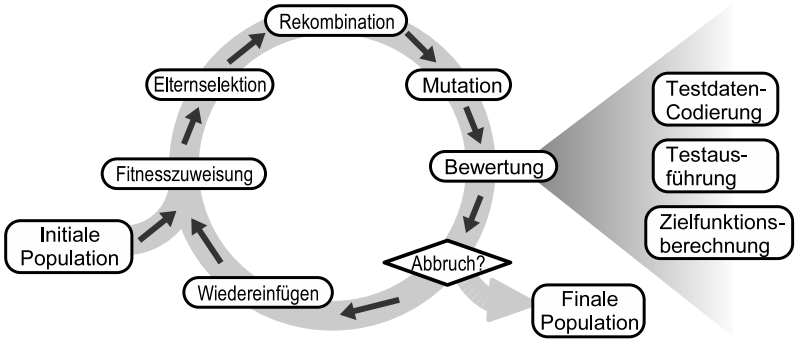


Abbildung 3.6: Ablauf des evolutionären Tests

Die neu erzeugten Individuen repräsentieren Punkte im Suchraum, deren Zielfunktionswerte jedoch noch nicht bekannt sind. Jeder Punkt im Suchraum kann einem Satz an Testdaten zugeordnet werden. Um den Zielfunktionswert für ein Individuum zu berechnen, muss daher der zugehörige Satz an Testdaten ausgeführt werden und das Testergebnis durch eine geeignete Zielfunktion bewertet werden. Der durch die Zielfunktion berechnete Wert wird wiederum dem Individuum als Zielfunktionswert zugeordnet.

Im folgenden Abschnitt wird eine kurze Übersicht gegeben, für welche Arten von Tests bisher erfolgreich Zielfunktionen definiert wurden und evolutionäres Testen angewandt werden konnte. Einen Überblick über die Anwendungsgebiete des evolutionären Testens bieten [101], [86] und [39].

Um z.B. Zeitverhaltenstests zu automatisieren, werden die Ausführungszeiten der generierten Testdaten gemessen, welche dann die Basis für die definierte Zielfunktion bilden [56].

[9] Ein sogenanntes Gen ist nichts anderes als eine Datenstruktur.

Das Optimierungsziel ist dabei, einen Satz an Testdaten mit der längsten Ausführungszeit eines Programms zu finden. Für Safety-Tests wird die Zielfunktion von Pre- und Post-Bedingungen einzelner Module abgeleitet [36] und für Robustheitstests oder Mechanismen zur Fehlertoleranz dient die Anzahl der überwachten Fehlerarten als Ausgangspunkt für die Zielbewertung [53]. Bei Tests zur Strukturüberdeckung eines Systems basiert die Zielfunktion auf den ausgeführten Pfad- und Zweigbedingungen eines Programms ([38], [34], [46]). Evolutionäres Testen wurde bereits ebenfalls für den Modultest von objekt-orientierter Software eingesetzt [99],[100],[67].

Aufgrund des diskreten Verhaltens von Software (if-Statements, Schleifen, usw.) resultiert die Abbildung des Findens der wichtigen Testdaten auf eine numerische Optimierungsaufgabe gewöhnlich in einem komplexen, unstetigen und nichtlinearen Suchraum. Die Dimensionen des Suchraums stehen in direkter Beziehung mit der Anzahl der Eingabedimensionen des zu testenden Systems. Die Ausführung von verschiedenen Programmpfaden und die geschachtelten Strukturen in Software-basierten Systemen führen zu multimodalen Suchräumen. Neben lokalen Minima weist der Suchraum auch Unstetigkeiten auf, sowie Plateaus mit identischen Zielfunktionswerten. In diesem Fall führen verschiedene Eingabewerte zu identischen Zielfunktionswerten. Sogenannte Neighbourhood-Search-Methoden wie Hill-Climbing sind nicht geeignet für einen Suchraum dieser Komplexität. Sie basieren auf einer sogenannten „Neighborhood-Function“, welche die Suche in Richtung guter Lösungen führen soll [63]. Eine solche „Neighborhood-Function“ ist für einen komplexen, unstetigen und nichtlinearen Suchraum nicht vorhanden. Daher werden meta-heuristische Suchverfahren wie z.B. evolutionäre Algorithmen, Simulated-Annealing [51] oder Taboo-Search [66] für den Test von Software-basierten Systemen eingesetzt.

Falls eine angemessene Repräsentation der Testdaten und eine passende Zielfunktion für das betrachtete Optimierungsziel definiert werden kann und ein evolutionärer Algorithmus als Suchtechnik angewendet wird, läuft der evolutionäre Test ab, wie in Abbildung 3.6 dargestellt. Die Anfangsmenge an Individuen wird gewöhnlich per Zufall generiert. Prinzipiell können Testdaten, die man durch einen vorangegangenen Test oder eine Analysetechnik erhalten hat, als Anfangspopulation verwendet werden. Somit kann der evolutionäre Test von bestehendem Wissen über das zu testende System profitieren. Jedes Individuum innerhalb der Population repräsentiert einen Satz an Testdaten, mit dem das zu testende System ausgeführt wird. Jede Testausführung wird überwacht und der Zielfunktionswert für das zugehörige Individuum wird entsprechend dem definierten Optimierungsziel berechnet. Als nächstes werden automatisch Individuen mit einem hohen Zielfunktionswert mit einer höheren Wahrscheinlichkeit ausgewählt. Die ausgewählten Individuen werden den Rekombinations- und Mutationsprozessen unterworfen, um aus ihnen neue Individuen als Nachkommen zu generieren. Die Szenarien, welche aus den neu erzeugten Individuen als Nachkommen resultieren, werden ebenso bewertet, indem sie auf dem zu testenden System ausgeführt werden und deren Testausführung überwacht wird. Schließlich wird eine neue Population gebildet, indem Nachkommen und Eltern-Individuen zusammengeführt werden. Dabei wird durch den evolutionären Algorithmus entschieden, welche Eltern-Individuen durch Nachkommen ersetzt werden. Von da an wiederholt sich der Prozess und beginnt erneut mit der Selektion, solange

bis das Optimierungsziel erreicht wurde.

3.3.3 Evolutionärer Strukturtest

Arbeiten über die automatische Generierung von Testdaten für Strukturtests sind weit verbreitet [85], [59], [34], [87], [46], [60], [41], [88], [38], [5], [58]. Das Ziel dieser Arbeiten ist es, zu bestimmen, welche Mengen an Testdaten für verschiedenartige Strukturtest-Kriterien wie z.B. Anweisungsüberdeckung, Zweigüberdeckung oder Bedingungsüberdeckung die höchste mögliche Überdeckung der betrachteten internen Programmstruktur erreichen. Dabei werden ebenso Überdeckungskriterien für blockorientierte Modellierungs- und Simulationswerkzeuge betrachtet [68],[83].

Strukturtests basieren auf der Annahme, dass ein System nur vollständig getestet werden kann, wenn alle seine Anteile während des Tests zumindest einmal ausgeführt werden. Der größte Schwachpunkt von Strukturtests ist, dass sie keine fehlenden Funktionen entdecken können, die für das System spezifiziert wurden, da sie sich ausschließlich am Programmcode orientieren.

Die Zielfunktion für Strukturtests basiert typischerweise auf der Berechnung eines Abstandes für jedes Individuum, der angibt, wie weit dieser Abstand von der Ausführung einer Anweisung entfernt ist, die gerade benötigt wird, um eine weitere Überdeckung des Programmcodes zu erreichen. Arbeiten über die Terminierungsbedingung für evolutionäre Strukturtests befinden sich in [97], [90] und [75].

3.3.4 Evolutionärer Test nicht-funktionaler Eigenschaften

Weitere Arbeiten verwenden evolutionäre Algorithmen zum Testen von nicht-funktionalen Eigenschaften wie Sicherheitsbedingungen und Zeitbedingungen. Das Ziel ist hierbei, die nicht-funktionalen Eigenschaften des Testobjekts zu testen unter Vewendung von evolutionären Algorithmen, indem man nach Testdaten sucht, für die das System die spezifizierten Sicherheits- oder Zeitbedingungen verletzt.

Beim Safety-Testing basiert die Zielfunktion auf der Berechnung einer Distanz bis zur Verletzung der Sicherheitsbedingung. Arbeiten zum Test von Sicherheitsbedingungen sind in [44], [45] und [47]. Beim Zeitverhaltenstest wird die Ausführungszeit gemessen, die das zu testende System für das generierte Testdatum benötigt. Die Arbeiten [56], [98], [70], [89], [62], [16], [102], [17], [91], [106] und [107] befassen sich mit dem evolutionären Test von Zeitbedingungen. Weitere Arbeiten beschreiben den Einsatz von evolutionären Algorithmen für Robustheitstests [53].

3.3.5 Evolutionärer Funktionstest

In den letzten Jahren wurde eine Reihe von Papieren veröffentlicht, die erfolgreich evolutionäre Algorithmen zur Erzeugung von Testdaten verwendet haben. Diese verfolgen verschiedene Optimierungsziele mit verschiedenen Testmethoden.

Die Anwendung des evolutionären Testens zur automatisierten Erzeugung von Testdaten für funktionale Tests eines Systems ist recht neu. Burton verwendet in [9] die Z-Notation und formale Methoden für die automatische Erzeugung von Testdaten für Modultests. Ein evolutionärer Funktionstest ohne Verwendung von formalen Methoden wurde von Baresel et al. in [74] dargestellt.

3.4 Bedarf für eine neue Methode

Im Automobil werden heutzutage immer mehr abstandsbasierte Fahrerassistenzsysteme integriert. Diese basieren auf der Erweiterung des Fahrzeugs um umgebungserfassende Sensoren. Die Anwendungsfunktionen abstandsbasierter Fahrerassistenzsysteme sind meist aus der Perspektive des Fahrers bzw. des Fahrzeugs definiert. Daher müssen die Anwendungsfunktionen aus Sicht des Fahrers bzw. des Fahrzeugs getestet werden.

Nach dem Stand der Technik in der Automobilindustrie erfolgt bei den funktionalen Tests sowohl die Auswahl der Testdaten als auch die Bewertung der Testergebnisse manuell. Die bisherigen Ansätze zur Automatisierbarkeit von funktionalen Tests bringen nur eine unwesentliche Verbesserung, da bei ihnen zwar die Auswahl aber nicht die Bewertung automatisiert werden kann. Sie erlauben daher immer noch nicht den gewünschten hohen Durchsatz an Tests. Als bekanntes und in der Praxis auch relevantes Beispiel sei der Zufallstest erwähnt, bei dem zur automatischen Ergebnisbewertung ein Testorakel benötigt würde, was für praktische Anwendungen meist nicht zur Verfügung steht.

Da der funktionale Test von Anwendungsfunktionen im Fahrzeug einen wesentlichen Zeit- und Kostenfaktor darstellt, wird eine Methode für den funktionalen Test benötigt, die einen möglichst hohen Grad an Automatisierung für funktionale Tests erlaubt. Um dies zu erreichen, wäre eine Methode wünschenswert, bei der sowohl die Auswahl der Testdaten als auch die Bewertung des Testergebnisses bei einer Anwendungsfunktion im Fahrzeug automatisch erfolgt. Diese Methode sollte sich darüber hinaus einfach in den jetzigen Entwicklungsprozess der Automobilindustrie integrieren lassen und für die heutigen Entwicklungsingenieure beherrschbar sein, damit durch die neue Methode keine zusätzlichen Kosten entstehen und weitere Ressourcen gebunden werden. Diese Methode sollte es überdies ermöglichen, dass Fehler möglichst früh gefunden werden.

Bei anderen Arten von Tests hat sich evolutionäres Testen als vielversprechender Ansatz zur Automatisierung herausgestellt. Der Einsatz evolutionärer Algorithmen auf dem Gebiet des funktionalen Testens ist jedoch bisher limitiert:

- Ein Ansatz (Baresel et al. [74]) erlaubt die Optimierung eines Signalverlaufs mit dem Ziel einer Grenzwertüberschreitung, d.h. es wird eine eindimensionale zeitabhängige Regelgröße bewertet. Nicht betrachtet werden mehrdimensionale Probleme mit Zielfunktionen, bei denen mehrere zeitabhängige Signale nötig sind, um einen Zielfunktionswert zu berechnen.
- Ein anderer Ansatz (Burton [9]) verwendet formale Methoden und ist so zur Zeit nicht in den Entwicklungsprozess der Automobilindustrie integrierbar, da die Entwicklungsingenieure darin nicht ausgebildet sind.

Die hier vorliegende Arbeit greift den Ansatz des evolutionären Testens auf und entwickelt Beschreibungsmittel, die es gestatten, die Sprache der Realzeitsysteme mit mehreren zeitlichen Signalverläufen, Ereignissen und Aktionen, die die heutigen Entwicklungsingenieure beherrschen, auf die Sprache der evolutionären Algorithmen mit einer Population von Individuen, die Gene tragen, abzubilden. Die aus dieser Idee entstandene Methode des evolutionären Funktionstests wird im Folgenden vorgestellt. Es wird gezeigt, wie sich die Auswahl der Testdaten und die Bewertung der Testergebnisse automatisieren lassen und wie sich die Methode in den bisherigen Entwicklungsprozess nahtlos integrieren lässt, indem sie von den bisherigen Aktivitäten profitiert und deren Ergebnisse nutzt.

Kapitel 4

Evolutionärer Funktionstest für Realzeitsysteme

In diesem Kapitel wird das Konzept des evolutionären Funktionstests für Realzeitsysteme entworfen. Ziel des evolutionären Funktionstests ist es stets, Fehler bei der Ausführung einer Anwendungsfunktion zu finden. Da der evolutionäre Algorithmus ein Optimierungsverfahren ist, dem ein Ziel vorgegeben werden kann, soll der evolutionäre Funktionstest dazu verwendet werden, das funktionale Verhalten einer Anwendungsfunktion in Richtung eines funktionalen Fehlverhaltens zu optimieren. Dazu braucht man letztendlich Kenntnis von der Fehlersituation und ein Maß für den Abstand von der Fehlersituation, um dem evolutionären Algorithmus das gewünschte Ziel vorzugeben.

Die Idee ist also, Testdaten in die Gene der Individuen des evolutionären Algorithmus zu packen und mit diesen Genen Tests durchzuführen. Hierbei ist anzumerken, dass der in Kapitel 2 in Definition 2.5 eingeführte Begriff des Testfalls in den folgenden Kapiteln, die sich auf den evolutionären Funktionstest beziehen, weiter verwendet wird. Dabei ist jedoch zu beachten, dass bei einem evolutionären Funktionstest das Sollergebnis nicht bekannt sein kann. Der Begriff „Testfall“ für einen evolutionären Funktionstest umfasst also nur die Testdaten, die für eine Testausführung erforderlich sind, nicht aber das Sollergebnis. Die Zielfunktion hat ein Maß für den Abstand von Fehlersituationen und bewertet Testdaten und Testergebnisse. Aufgrund dieser Bewertung generiert der evolutionäre Algorithmus daraus in den weiteren Zyklen der Simulation automatisch neue Gene, die sich automatisch an die Fehlersituation heranarbeiten. D.h., der Vorgang des Testens wird auf ein Optimierungsproblem abgebildet.

Das Neuartige ist, dass hier ein automatischer Test konzipiert wird, der nicht nur die Testfälle automatisch generiert, sondern auch die Bewertung der Testergebnisse automatisch durchführt. Bei anderen automatischen Testverfahren in der Literatur musste die Bewertung manuell erfolgen, was keinen hohen Durchsatz an Tests erlaubte.

Für dieses neuartige Konzept müssen zuallererst die Beschreibungsmittel gefunden werden, die es gestatten, die Sprache der evolutionären Algorithmen mit einer Population von Indi-

viduen, die Gene tragen, auf die Sprache der Realzeitsysteme mit Ereignissen und Aktionen oder Kennlinien abzubilden. Diese Abbildung wird in der vorliegenden Arbeit nach der Vorstellung der Architektur eines Testsystems in Kapitel 4.1 unter dem Begriff „Codierung der Testdaten“ in Kapitel 4.2 durchgeführt. Evolutionäre Algorithmen sind Optimierungsverfahren, wobei eine Zielfunktion vorhanden sein muss, um die Individuen einer Population zu bewerten, damit beim Erzeugen der nächsten Generation der Population Individuen mit noch besseren Genen erzeugt werden können. Um die Güte eines Individuums, welches Testdaten trägt, für den Test bewerten zu können, muss man zuallererst Ergebnisklassen eines Tests finden wie z.B. „Anwendungsfunktion hat ausgelöst“ oder „Anwendungsfunktion hat nicht ausgelöst“. Im nächsten Schritt muss man dann Merkmale finden, die beschreiben, zu welcher der Klassen ein Individuum zugeordnet werden kann. Diese Zuordnung kann deterministisch sein oder aber auch mit einer Unschärfe verbunden sein. Kapitel 4.3 behandelt die Methodik der Gewinnung einer Zielfunktion mit den Schritten:

- Festlegung des Optimierungsziels,
- Definition der Ergebnisklassen,
- Identifikation charakteristischer Merkmale und
- Formulieren der Zielfunktion.

Damit wird eine neue Methode vorgeschlagen, die grundsätzlich für beliebige Realzeitsysteme eingesetzt werden kann. Für verschiedene Anwendungsfunktionen mit ihren unterschiedlichen Anforderungen muss im Rahmen der hier vorgestellten Methode die Codierung der Testdaten und die Zielfunktion applikationsspezifisch vorgegeben werden.

Nach der Vorstellung einer Architektur des Testsystems in Kapitel 4.1 wird in Kapitel 4.2 eine Vorgehensweise zur Codierung von Testdaten, die eine Ereignissteuerung einschließt, vorgestellt. In Kapitel 4.3 folgt eine Darstellung einer Methode zum Entwurf einer applikationsspezifischen Zielfunktion. In Kapitel 4.4 wird dargestellt, wie der evolutionäre Funktionstest in den Entwicklungsprozess eingeordnet werden kann.

In Kapitel 5 werden dann für zwei Beispiele aus der Welt der Fahrerassistenzsysteme, das „Automatische Parksystem“ und der „Abstandsbasierte Bremsassistent“ die Codierung der Testdaten und die Gewinnung der Zielfunktion detailliert vorgestellt. Sie beinhalten also die Methodik des evolutionären Funktionstests für beide Anwendungen. Die für beide Systeme durchgeführten Experimente sind danach in Kapitel 6 beschrieben.

4.1 Architektur des Testsystems

In diesem Kapitel wird die Architektur eines Testsystems für einen evolutionären Funktionstest von Realzeitsystemen im Automobil entwickelt. Die Architektur umfasst die Zerlegung des Systems in Komponenten, die für einen evolutionären Funktionstest notwendig sind, die

Beschreibung des Zusammenwirkens der Komponenten sowie die Darstellung der Strategie für diese Architektur.

Ein Testsystem für einen evolutionären Funktionstest besteht aus vier Komponenten, nämlich:

- einem evolutionären Algorithmus,
- einer Komponente „Codierung Testdaten“,
- einer Komponente „Zielfunktion“ und
- einer Testumgebung für das Testobjekt.

Die Architektur des Testsystems ist Abbildung 4.1 dargestellt.

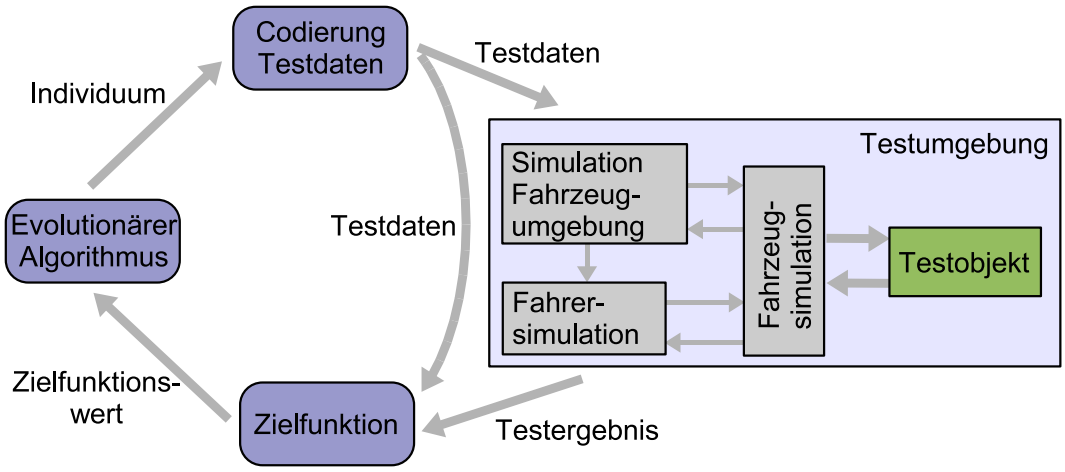


Abbildung 4.1: Architektur des Testsystems für einen evolutionären Funktionstest

Hinter dieser Architektur steht die folgende Strategie:

Der evolutionäre Algorithmus steuert die Optimierung. Die Optimierung verfolgt das Ziel, ein Individuum zu erzeugen, das bei der Bewertung durch die Zielfunktion einen möglichst kleinen Zielfunktionswert erhält. Hierzu wird durch den evolutionären Algorithmus ein Satz von Individuen erzeugt, eine sogenannte Population. Für jedes Individuum dieser Population erfolgt eine Codierung in die Form von Testdaten, die der Testumgebung zugeführt werden. Testdaten und Testergebnis eines jeden Individuums werden dann durch die Zielfunktion bewertet. Basierend auf den Informationen über Individuum und Zielfunktionswert aller Individuen der Population generiert nun der evolutionäre Algorithmus die nächste Generation von Individuen, also den nächsten Satz von Individuen, mit dem Ziel, dass diese Generation bessere Individuen mit kleineren Zielfunktionswerten enthält.

Nachfolgend eine kurze Beschreibung der Komponenten und ihres Zusammenwirkens:

Die Komponente „Codierung Testdaten“ erzeugt aus einem übergebenen Individuum die Testdaten für einen einzelnen Testfall. Das Modell für diese Codierung ist ein wesentlicher Aspekt beim Entwurf eines Testsystems. Die Art und Weise der Codierung beeinflusst maßgeblich die Eigenschaften des Suchraums und die Spannweite des kombinatorisch möglichen Eingabedatenraums.

Die „Testumgebung“ und das darin eingebettete Testobjekt stellt eine weitere Komponente der Architektur dar. Die erzeugten Testdaten bilden die Eingangsparameter für die Simulationskomponenten der Testumgebung. Mit diesem Satz an Testdaten wird anschließend eine Simulation eines Fahrmanövers durchgeführt. Die Testdaten definieren dabei die Situation, mit welcher das Testobjekt in der Testumgebung konfrontiert wird. Wie in Kapitel 3.1.2 dargestellt bildet das simulierte Szenario das Testergebnis.

Die Komponente „Zielfunktion“ berechnet zu jedem Testergebnis einen zugehörigen Zielfunktionswert. Für die Berechnung des Zielfunktionswerts sind die Testdaten und das Testergebnis erforderlich. Die Zielfunktion implementiert ein definiertes Optimierungsziel des Tests. Entsprechend diesem Optimierungsziel wird der Zielfunktionswert berechnet und an den evolutionären Algorithmus zurückgegeben. Die formulierte Zielfunktion ist kein Testorakel im Sinne der Definition 2.14, da die Zielfunktion im Unterschied zu einem Testorakel nicht das Sollergebnis vorhersagen kann. Die Zielfunktion kann jedoch eine Auskunft geben, zu welcher Ergebnisklasse das betrachtete Testergebnis gehört.

Für die Komponente des „Evolutionären Algorithmus“ kann eine verfügbare Implementierung verwendet werden. Die Testumgebung mit den Simulationsanteilen wird üblicherweise im normalen Entwicklungsprozess erstellt. Somit müssen im Rahmen des evolutionären Funktionstests lediglich die Komponenten „Codierung Testdaten“ und „Zielfunktion“ entwickelt werden. In den folgenden Kapiteln 4.2 und 4.3 wird dargestellt, wie diese Komponenten für eine Anwendungsfunktion erstellt werden können.

Zuvor soll jedoch noch der Aspekt diskutiert werden, wie ein evolutionärer Funktionstest gestartet wird, oder anders ausgedrückt, wie die initiale Population gewählt werden soll. Die Individuen können entweder zufällig bestimmt werden oder über ein sogenanntes „Seeding“. Als Seeding bezeichnet man die Vorgehensweise, dass als Startpunkt für die initiale Population des evolutionären Algorithmus manuell definierte Testfälle des Standard-Entwicklungsprozesses verwendet werden [73]. In anderen Worten, beim Seeding besteht die initiale Population für die evolutionäre Suche aus den Individuen zu den Testdaten der manuell definierten Testfälle.

Dadurch wird es möglich, das bei der manuellen Testfalldefinition gewonnene Wissen über den Problembereich für die evolutionäre Suche zu nutzen. Die Testdaten jedes manuell definierten Testfalls sollen mit dem gewählten Modell zur Codierung der Testdaten dargestellt werden können. Das den Testdaten entsprechende Individuum soll als Individuum der initialen Population verwendet werden. Damit wird einerseits implizit die minimale Größe der initialen Population festgelegt. Andererseits erzwingt diese Methode, dass ein angemessen flexibles Modell für die Codierung der Testdaten verwendet wird, um der Anwendung gerecht zu werden.

Wird innerhalb des evolutionären Algorithmus ein elitärer Selektionsmechanismus verwendet, bleibt stets die bisher beste Lösung in jeder weiteren Generation der Population bestehen [108]. Damit wird erreicht, dass zumindest das beste Testergebnis der manuell definierten Testfälle bestehen bleibt oder aber dass ein noch besseres Testergebnis durch den evolutionären Funktionstest gefunden wird. Die Ergebnisse der experimentellen Untersuchungen in Kapitel 6 bestätigen dies.

4.2 Codierung der Testdaten

Codierung der Testdaten bedeutet, die Gene eines Individuums in reale Testdaten für das Testobjekt zu übersetzen. Für die Codierung der Testdaten muss ein für die Anwendung und die Testumgebung passendes Modell entworfen werden. In Abbildung 4.2 ist dargestellt, dass ein Individuum in seinen Genen die Informationen für die Testdaten eines Testfalls tragen muss.

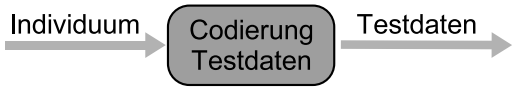


Abbildung 4.2: Abbildung eines Individuums auf Testdaten

Beim Ablauf der Simulation übergibt dann die Komponente „Codierung Testdaten“ die von ihr berechneten Testdaten als Eingangsparameter an die Testumgebung. Man hat zu reflektieren, wie die Testdaten für die Anwendungsfunktion auf einen Suchraum abgebildet werden könnten. Gefordert wird dabei, dass das Modell verschiedenste Testdaten erzeugen kann und dass die generierten Testdaten den Bereich aller sinnvollen Testdaten für eine Anwendung möglichst vollständig abdecken. Außerdem soll die Abbildung eines Individuums auf Testdaten möglichst bijektiv sein. D.h. jedem Satz an Testdaten sollte eindeutig ein Individuum zuzuordnen sein und umgekehrt.

Beim Entwurf des Modells zur Codierung der Testdaten muss auch die Struktur der Gene der Individuen festgelegt werden. Dabei wird die Anzahl der Gene festgelegt und ferner für jedes Gen der Datentyp und der Wertebereich. Die Anzahl der Gene entspricht der Anzahl der Dimensionen des Suchraums (d.h. der linear unabhängigen Variablen). Diese Festlegungen beeinflussen maßgeblich die Struktur des Suchraums.

Ein Ziel beim Entwurf der Codierung ist es, die Größe des Suchraums möglichst klein zu halten. Dazu sollte man mit möglichst wenigen Dimensionen auskommen. Parallel dazu versucht man, für jede Dimension den Wertebereich einzuschränken und die Granularität der Quantisierung zu erhöhen. Beide Maßnahmen führen dazu, dass die Anzahl der Punkte im Suchraum minimiert wird, was letztendlich der Anzahl an Kombinationen der Codierung und der Anzahl der in der Testumgebung durchführbaren Testfälle entspricht. Dies

unterscheidet sich von der Anzahl bzw. Menge der möglichen Testfälle für die Anwendungsfunktion, da in Abhängigkeit vom Modell zur Codierung der Testdaten und der Auslegung der Testumgebung möglicherweise nicht alle kombinatorisch denkbaren Testfälle für die Anwendungsfunktion erreicht werden können. Es muss ein Kompromiss gefunden werden und das Modell für eine bestimmte Anwendung so konstruiert werden, dass für diese Anwendung alle in der Praxis relevanten Testfälle erreicht werden können und gleichzeitig der Suchraum möglichst klein ist.

Um evolutionäre Algorithmen für einen evolutionären Funktionstest effizient nutzen zu können, muss der Suchraum so weit wie möglich eingeschränkt werden, ohne aber die Charakteristika der jeweiligen Anwendung zu verlieren. Hierzu ist die Codierung der Testdaten zu optimieren, damit der Ressourcenverbrauch eingeschränkt wird. Bei dieser Optimierung müssen die Eigenschaften der zu testenden Anwendungsfunktion und das Optimierungsziel im Auge behalten werden. Wegen der großen Bandbreite von Anwendungsfunktionen sind allgemeingültige Ansätze nicht zu erwarten. Daher wurde in beiden Anwendungen „Automatisches Parksystem“ und „Abstandsbasierter Bremsassistent“ das Anwendungswissen verwendet, um eine möglichst effiziente, problembezogene Codierung zu erhalten. Die entsprechenden Codierungen für beide Anwendungen sind in den Kapiteln 5.1.2 und 5.3.2 dargestellt.

Arbeiten die sich mit der Codierung von Kennlinienverläufen befassen, wurden bereits in [74] und [14] dargestellt. Neu ist die Codierung von Testdaten unter Verwendung einer Ereignissteuerung.

4.2.1 Ereignissteuerung

Im Hinblick auf den Test von Realzeitsystemen im Automobil sind die Testdaten durch folgende Eigenschaften gekennzeichnet:

- Schaltzustände (z.B. Blinker gesetzt, Anwendungsfunktion eingeschaltet, etc.)
- Signalverläufe (zeit- oder weggesteuerte Signalverläufe wie z.B. Lenkwinkelvorgabe des Fahrers)
- Ereignisse (z.B. Warnton ertönt, Relativ-Abstand wird unterschritten, etc.)
- Aktionen (Fahrer tritt auf die Bremse, Zielfahrzeug beschleunigt, etc.)

Diese Eigenschaften sind oft Inhalt der Testdaten für Realzeitsysteme. Eine Codierung der Testdaten muss daher diese Eigenschaften codieren können. Im Folgenden wird dargestellt, wie eine solche Codierung aussehen kann.

Ein Beispiel für eine Datenbedingung nach Hatley und Pirbhai ist im Falle eines startenden Flugzeugs auf der Startbahn „Ist die Rollgeschwindigkeit gleich der Abhebegeschwindigkeit, so muss ein Steuerfluss erzeugt werden, welcher das Höhenleitwerk stellt, so dass das Flugzeug abhebt.“

4.2.2 Klassifikation von Genen

Allgemein kann man zur Codierung eines Individuums für einen evolutionären Funktionstest abstandsbasierter Fahrerassistenzsysteme sagen, dass die Individuen verschiedene Klassen (bzw. Typen, Arten) von Genen enthalten. Diese Klassen sollen im Folgenden vorgestellt werden:

Gene vom Typ I

Gene vom Typ I codieren Konstanten bzw. Schaltzustände im Kontext der Anwendung. Diese Werte haben die Bedeutung einer initialen Belegung. Diese Werte sind entweder nur zu Beginn der Testausführung relevant oder bleiben über den gesamten Verlauf der Testausführung konstant. Die Belegung dieser Werte kann sich jedoch von Testfall zu Testfall ändern. Typische Beispiele sind eine Schalterstellung zu Beginn des Szenarios, die initiale Geschwindigkeit eines Fahrzeugs oder der Reibwert der Fahrbahn.

Gene vom Typ II

Gene vom Typ II codieren den Verlauf einer Größe in Form einer Kennlinie. D.h. die codierte Größe ist von einer anderen Größe abhängig. Beispiele für Größen sind die Zeit oder der Ort des Systemfahrzeugs in der Simulation. Charakteristisch ist dabei, dass eine solche Kennlinie für die Durchführung einer Simulation für den gesamten Ablauf eines Szenarios, d.h. vom Simulationsbeginn bis zum Simulationsende, erforderlich ist.

Die Codierung der Kennlinie kann in Form von einzelnen Zeitschritten erfolgen. Diese Art der Zeitschritt-basierten Codierung für ein Signal oder mehrere Signale ist im Kapitel 4.2.3 dargestellt. Eine andere Möglichkeit zur Codierung der Kennlinie ist die Definition eines anwendungsspezifischen Modells, was ebenfalls in Kapitel 4.2.3 gezeigt wird. Mit einem solchen Modell kann die Zahl der erforderlichen Gene zur Codierung der Kennlinie wesentlich reduziert werden und somit auch die Größe des Suchraums.

Gene vom Typ III

Gene vom Typ III codieren die Datenbedingungen von Ereignissen. Die Datenbedingungen hängen von der Art der Ereignisse ab. Beispiele für Datenbedingungen sind:

- Relativ-Abstand wird unter- bzw. überschritten
- Relativ-Geschwindigkeit wird unter- bzw. überschritten
- Systemfahrzeug erreicht eine vorgegebene Sollgeschwindigkeit

Die Datenbedingungen werden während der Simulation durch die Testumgebung überwacht. Sobald eine der Datenbedingungen erfüllt ist (d.h. die Datenbedingung „zündet“), wird das zugehörige Ereignis ausgelöst. Nachfolgend können dadurch Aktionen der Testumgebung ausgelöst werden.

Ein Gen vom Typ III codiert also eine Datenbedingung, die für die Testumgebung relevant ist. Es handelt sich dabei nicht um Datenbedingungen, die durch das Testobjekt verarbeitet werden. In anderen Worten, es werden Datenbedingungen codiert, die einen Test definieren und nicht den Systementwurf des Testobjekts modellieren.

Gene vom Typ IV und Typ V

Gene vom Typ IV und Typ V codieren Aktionen, die aufgrund von Ereignissen ablaufen. Dabei codieren Gene vom Typ IV die Aktionen entsprechend dem Typ I, z.B. einen Schalter in einen neuen Schaltzustand bringen. Gene vom Typ V codieren Aktionen entsprechend dem Typ II in Form eines Kennlinienverlaufs. Das Charakteristische an den Genen vom Typ IV und Typ V ist jedoch, dass sie erst ab dem Zeitpunkt gelten, wenn das der Aktion zugeordnete Ereignis eingetreten ist, d.h. wenn die Datenbedingung „gezündet“ hat.

4.2.3 Codierung von Kennlinienverläufen

Ein Kennlinienverlauf kann durch eine Codierung mit diskreten Stützpunkten, d.h. mit einer vorgegebenen Schrittweite, erfolgen. Arbeiten über die Codierung von Kennlinienverläufen befinden sich in [74] und [14].

Bei der Codierung ist die Schrittweite so zu wählen, dass sie der Taktrate entspricht, mit der die Kennlinie bei der Testausführung abgefragt wird. Definiert der Kennlinienverlauf einen zeitlichen Signalverlauf, so muss dabei für jeden Zeitschritt ein Wert codiert werden. Diese Art der Codierung ist nachfolgend dargestellt. Analog kann dieses Prinzip auf den Verlauf mehrerer Signale erweitert werden, was ebenfalls in Kapitel 4.2.3 gezeigt wird.

Codierung eines Signalverlaufs

Insbesondere für den Test von reaktiven Systemen werden Signalverläufe definiert. Die Codierung der Testdaten für den evolutionären Funktionstest muss daher die Möglichkeit bieten, Signalverläufe abbilden zu können. Bei dieser Codierung müssen Punkte im Zustandsraum der Individuen abgebildet werden auf einen Signalverlauf. Dieser Signalverlauf liegt anschließend vor in Form einer Kennlinie, d.h. er ist abhängig von einer anderen Größe (z.B. der Simulationszeit). Werden keine weiteren Einschränkungen getroffen, kann im Signalverlauf der Wert eines Signals zu jedem Zeitschritt einen beliebigen vorgegebenen Wert aus dem Wertebereich des Signals annehmen. In Abbildung 4.3 ist eine solche Codierung dargestellt.

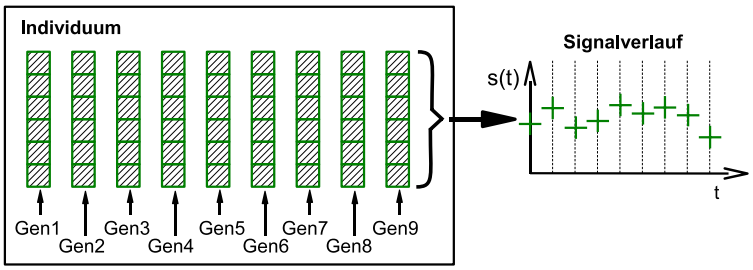


Abbildung 4.3: Codierung der Signalwerte zu einzelnen Zeitschritten

Bei dieser Methode wird ein Signalwert zu einem bestimmten Zeitschritt durch ein eigenes Gen des Individuums codiert. Aufeinanderfolgende Gene des Individuums codieren eine Abfolge von Signalwerten. Im dargestellten Fall wird der Wert von *Gen1* in einen Signalwert $s(t_0)$ zum Zeitpunkt t_0 übersetzt. Der Wert in *Gen2* codiert den Signalwert $s(t_1)$ zum Zeitpunkt t_1, usw. Diese Methode der Codierung berücksichtigt dabei keinerlei Abhängigkeiten zwischen den einzelnen Signalwerten. In anderen Worten, die maximale Frequenz des codierten Signals ist entsprechend dem Abtasttheorem lediglich durch die Größe der Zeitdifferenz Δt zwischen den Signalwerten definiert. Die Unabhängigkeit der einzelnen Signalwerte ist durch die Codierung in jeweils eigenen Genen bedingt. Durch die Auswahl der Anzahl an Genen, der Wortbreite eines Gens und der Zeitdifferenz Δt lassen sich die Eigenschaften des codierten Signalverlaufs bestimmen.

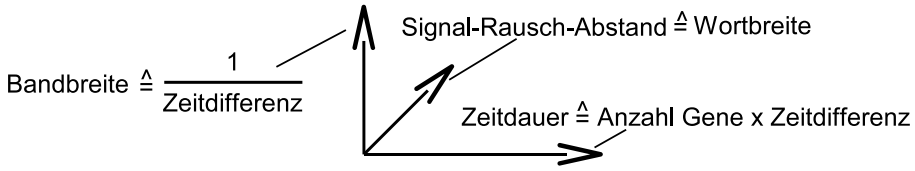


Abbildung 4.4: Nachrichtenquader für die Signal-Codierung in einem Individuum

Die im Individuum und im Signalverlauf enthaltene Information ist äquivalent, d.h. durch die Codierung wird keine Information hinzugefügt oder entfernt. Die Bandbreite des generierten Signals wird (reziprok) durch die codierte Zeitdifferenz Δt festgelegt. Die Zeitdauer des codierten Signals hängt ab vom Produkt aus der Anzahl an Genen und der codierten Zeitdifferenz Δt. Der Signal-Rausch-Abstand des Signals entspricht dem Quantisierungsfehler und wird bestimmt durch die Wortbreite der Gene.

Codierung mehrerer Signalverläufe

Um für mehrere Signale parallel einen zeitlichen Signalverlauf vorgeben zu können, kann man die Gene eines Individuums in unterschiedliche Abschnitte einteilen. Jeder Abschnitt codiert den Wertebereich für jeweils ein Signal, wie in Abbildung 4.5 dargestellt.

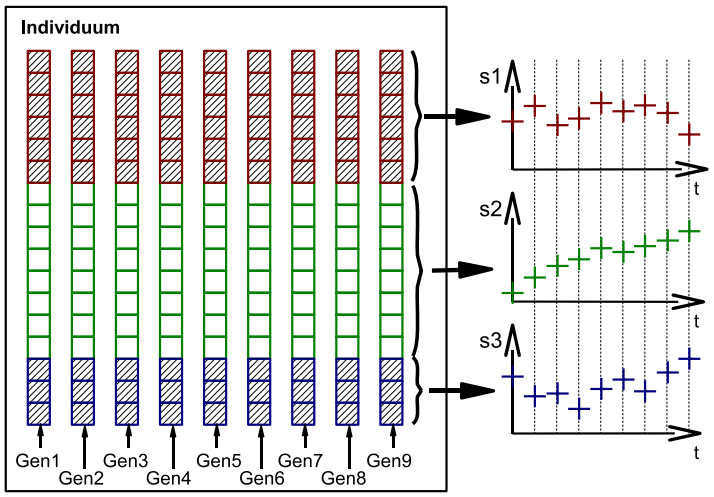


Abbildung 4.5: Codierung verschiedener Signalverläufe in einem Individuum

In Abbildung 4.5 ist ein Individuum dargestellt, das 9 Gene enthält. Jedes Gen ist aufgeteilt in drei Abschnitte. Der oberste Abschnitt in allen 9 Genen codiert zusammen genommen den Verlauf von Signal s1. Der mittlere Abschnitt codiert den Verlauf von Signal s2 und der untere Abschnitt codiert den Verlauf von Signal s3. In Abhängigkeit vom Wertebereich eines Signals erhält der zugehörige Genabschnitt dann die dafür erforderliche Breite. Betrachtet man nur einen Zeitschritt in den Signalverläufen von s1, s2 und s3, dann enthält das zu diesem Zeitschritt zugehörige Gen die Information für alle drei Signalwerte zu diesem Zeitpunkt, jeweils in einem eigenen Abschnitt.

Optimierte Codierung eines Signalverlaufs

Der Nachteil der Schritt-basierten Codierung ist der Speicherbedarf für die Codierung. Daher ist es im Kontext einer konkreten Anwendung oft sinnvoll, spezifisches Wissen über ein Signal bei der Codierung mit zu berücksichten. Das Prinzip ist, dass man mit dem Wissen aus der Anwendung überlegt, ob man einen Signalverlauf mit weniger Stützpunkten beschreiben kann als mit der Schritt-basierten Codierung. Dabei betrachtet man die relevanten Eigenschaften des Signals mit dem Wissen aus dem Kontext der Anwendung. Außerdem berücksichtigt man Einschränkungen, die man beim Signalverlauf machen kann.

Ein Beispiel ist die Codierung einer Fahrerbremsmomentvorgabe. Der Verlauf der Bremsmomentvorgabe soll in Form einer Kennlinie codiert werden, die als Aktion zeitgesteuert abgerufen werden kann.

In Abbildung 4.6 ist ein typischer Verlauf einer Fahrerbremsmomentvorgabe dargestellt. Links ist die Schritt-basierte Codierung dargestellt, wobei jeder der Stützpunkte durch ein Gen codiert werden muss. Rechts wurde die Fahrerbremsvorgabe approximiert und wird

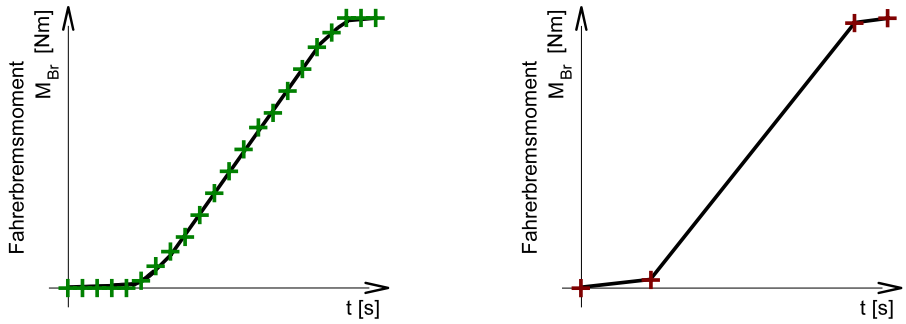


Abbildung 4.6: Beispiel für die Codierung einer Fahrerbremsmomentvorgabe

nun nur noch durch vier Stützpunkte vorgegeben. Dadurch kann die Anzahl der zur Codierung notwendigen Gene erheblich reduziert werden. Jedoch müssen dabei anwendungsspezifische Einschränkungen bzw. charakteristische Eigenschaften des Signalverlaufs berücksichtigt werden. In anderen Worten, es wird Problemwissen über die Anwendungsfunktion für die Codierung verwendet.

Durch eine solche Codierung reduziert man die Anzahl der zur Definition eines Signalverlaufs notwendigen Gene. Dies hat zur Konsequenz, dass der Suchraum kleiner wird bzw. weniger Gene in einem Individuum codiert werden müssen. Andererseits bedeutet eine solche Codierung ebenso eine Einschränkung der möglichen Signalverläufe. Würde das Modell der Codierung beispielsweise Sprünge im Signalverlauf ausschließen und wären diese Sprünge im Signal jedoch relevant für den Test der Anwendung, so würde man durch diesen Ausschluss unter Umständen einen ganz wesentlichen Aspekt beim Testen der Anwendung verlieren. Daher muss die Approximation einer Kennlinie unter Berücksichtigung der Anwendungsfunktion durchgeführt werden. Die Approximation eines Bremspedalverlaufs wird in Kapitel 5.3.2 dargestellt.

4.3 Entwurf der Zielfunktion

Der Entwurf der Zielfunktion ist ein entscheidender Schritt beim Aufbau einer Testumgebung für den evolutionären Funktionstest. Eine geeignete Zielfunktion ermöglicht es, in automatisierter Weise Testergebnisse zu bewerten. Dies ist die entscheidende Neuerung der Methode des evolutionären Funktionstests im Vergleich zu existierenden Verfahren zur Testautomatisierung. Die manuelle Bewertung der Testergebnisse ist bei den bisherigen Verfahren der limitierende Faktor, der einen hohen Durchsatz verhindert und damit den Vorteil der automatischen Generierung von Testdaten stark relativiert. Die maschinelle Bewertung von Testergebnissen zusätzlich zur automatischen Generierung der Testdaten bringt einen erheblichen Zeitgewinn und ermöglicht es nun, die Anzahl an durchführbaren Tests drastisch zu steigern.

Der Entwurf einer Zielfunktion für einen evolutionären Funktionstest erfordert eine neue Vorgehensweise gegenüber den anderen Arten von evolutionären Tests. Beim evolutionären Test des Zeitverhaltens ist das Optimierungziel generell *„Finden der Testdaten mit maximaler (minimaler) Ausführungszeit"*. Beim evolutionären Strukturüberdeckungstest gibt es mehrere, jedoch generalisierte Optimierungziele, wie z.B. *„Finden von Testdaten mit maximaler Anweisungsüberdeckung (Strukturüberdeckung)"*. Bei beiden Verfahren ist das Optimierungsziel anwendungsinvariant (Zeitdauer bzw. Überdeckung). In anderen Worten, diese Ziele gelten unabhängig von der Anwendung.

Auf Grund der großen Vielfalt und hohen Komplexität beliebiger Anwendungsfunktionen gibt es keine allgemeingültigen Größen zur Bewertung aller Anwendungsfunktionen als Gesamtheit. Für jede einzelne Anwendungsfunktion ist das Optimierungsziel jeweils anwendungsspezifisch herauszuarbeiten. Eine Generalisierung wie beim Zeitverhalten oder der Strukturüberdeckung ist hier nicht möglich.

Dieses Kapitel stellt eine Vorgehensweise zum systematischen Entwurf einer Zielfunktion für einen evolutionären Funktionstest vor. Die der Vorgehensweise zu Grunde liegende Überlegung besteht darin, die Zielfunktion als einen Klassifikator analog der Mustererkennung aufzufassen. Nach Theodoridis [65] ist Mustererkennung die wissenschaftliche Disziplin, deren Ziel die Klassifikation von Objekten in eine Anzahl von Kategorien oder Klassen ist. Dabei können diese Objekte Bilder, Signalformen oder jede Art von Messung sein, die klassifiziert werden muss. Im Kontext des evolutionären Funktionstests soll die Zielfunktion eine Klassifizierung von Testergebnissen ermöglichen.

Eine wesentliche Aufgabe der Mustererkennung besteht darin, Merkmale zu finden, die klassentrennend sind. Einen formalen Weg, wie z.B. bei der Signalerkennung im Rauschen, gibt es hier nicht. [22]. Die hauptsächliche Schwierigkeit dabei ist, wie bei allen Problemen der Mustererkennung auch, solche charakteristischen Merkmale zu identifizieren. Damit soll eine Zuordnung eines Testergebnisses zu einer Ergebnisklasse sinnvoll möglich werden. Dabei sind die identifizierten Merkmale spezifisch für die Anwendung und das gewählte Ziel der Optimierung.

4.3.1 Vorgehensweise

Im Folgenden wird gezeigt, wie man eine Zielfunktion für den evolutionären Funktionstest einer Anwendungsfunktion in systematischer Weise entwerfen kann. Die Vorgehensweise ist abgeleitet aus den Methoden der Mustererkennung [84]. Nach der Aufzählung der einzelnen Schritte zum Entwurf einer Zielfunktion erfolgt für jeden Schritt eine Erörterung der Umsetzung dieser Strategie in die Praxis. Hier zunächst die Folge der Schritte:

1. Ergebnisklassen
   In diesem Schritt muss zunächst die Anforderung ausgewählt werden, die betrachtet werden soll. Durch Analyse dieser Anforderung werden dann die relevanten Ergebnisklassen modelliert.

2. Optimierungsziel
   In diesem Schritt wird das Optimierungsziel ausgewählt, das erreicht werden soll und für das die Zielfunktion entworfen werden soll. Das Optimierungsziel wird dabei so ausgewählt, dass bei seinem Erreichen die ausgewählte Anforderung gebrochen wird.
3. Klassifikationsmerkmale
   In diesem Schritt müssen charakteristische Klassifikationsmerkmale identifiziert werden, anhand derer unterschieden werden kann, ob ein Szenario dem Optimierungsziel entspricht oder nicht. Werden alternative Merkmale identifiziert, muss eines für die nachfolgende Umsetzung ausgewählt werden.
4. Merkmalsbasierte Zielfunktion
   In diesem Schritt muss das identifizierte und ausgewählte Klassifikationsmerkmal mit Hilfe einer mathematischen Funktion ausgewertet werden. Der durch die formulierte Funktion berechnete Wert ist der Rückgabewert der Zielfunktion.

Diese vier Schritte zum systematischen Herleiten einer Zielfunktion werden im Folgenden nochmals ausführlicher diskutiert.

4.3.2 Ergebnisklassen

Zunächst muss bei diesem Schritt aus der Gesamtheit aller Anforderungen an die Anwendungsfunktion diejenige ausgewählt werden, die nachfolgend getestet werden soll. Dabei ist zu beachten, dass durch den evolutionären Funktionstest dann nur diese eine Anforderung getestet wird. Fehler, die andere Anforderungen betreffen, werden nicht erkannt und normalerweise nicht gefunden.

Dies bedeutet zum einen, dass ein einzelner evolutionärer Funktionstest niemals einen vollständigen funktionalen Test abdecken kann. Zum anderen folgt daraus, dass die ausgewählte Anforderung eine zentrale Bedeutung für die Anwendungsfunktion haben sollte. Es werden hier daher bevorzugt Anforderungen an das Fahrzeug auf oberster Ebene ausgewählt (vgl. Kapitel 3.2.1).

Aus der Analyse der gewählten Anforderungen werden anschließend Ergebnisklassen abgeleitet, die sich aus Sicht der gewählten Anforderung unterscheiden lassen und relevant sind. Eine Ergebnisklasse ist dabei eine Menge, deren Elemente einzelne Szenarien sind. Bei der Modellierung der Ergebnisklassen kann zunächst auf oberster Ebene unterschieden werden zwischen Szenarien, welche mit der Anforderung übereinstimmen oder Szenarien, welche mit der Anforderung nicht übereinstimmen.

In Abbildung 4.7 sind mögliche Ergebnisklassen einer Anforderung exemplarisch dargestellt. Dabei umfasst die Ergebnisklasse A Szenarien, die in Übereinstimmung mit der gewählten Anforderung sind. Diese Szenarien sind für den Test uninteressant, daher wird diese Ergebnisklasse nicht weiter modelliert. Szenarien die nicht mit der gewählten Anforderung übereinstimmen werden in der Ergebnisklasse $\bar{A}$ dargestellt. Diese Szenarien sind für den

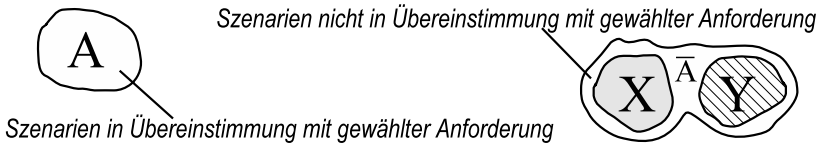


Abbildung 4.7: Exemplarische Darstellung betrachteter Ergebnisklassen einer Anforderung

Test interessant. Lassen sich verschiedene Typen dieser Szenarien unterscheiden, können diese durch getrennte Klassen innerhalb der Ergebnisklasse $\bar{A}$ modelliert werden. In der exemplarischen Darstellung in Abbildung 4.7 sind dies die Klassen X und Y.

Dieser Schritt bedeutet eine Spezialisierung, da man sich auf eine aus mehreren möglichen Anforderungen festlegt.

4.3.3 Optimierungsziel

In diesem Schritt wird das Optimierungsziel des evolutionären Funktionstests festgelegt, dieses Ziel wird durch die zu entwerfende Zielfunktion implementiert und während eines Testlaufs verfolgt. Als Grundlage für die Auswahl des Optimierungsziels dienen die modellierten Ergebnisklassen. Aus diesen wird eine Klasse ausgewählt, die für den Test interessant ist. D.h. die gewählte Klasse sollte Szenarien enthalten, die der Anforderung widersprechen. Gibt es dazu verschiedene Typen von Ergebnisklassen, muss eine davon ausgewählt werden. Durch die Auswahl der Klasse wird das Optimierungsziel festgelegt, d.h. das Optimierungsziel ist nun Elemente der ausgewählten Klasse zu finden.

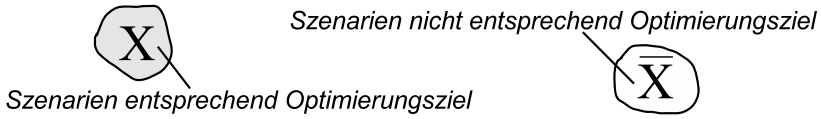


Abbildung 4.8: Auswahl der betrachteten Ergebnisklasse X

In Abbildung 4.8 ist die Auswahl der Ergebnisklasse X dargestellt. Das Optimierungsziel ist es nun Szenarien der Klasse X zu finden. Die Zielfunktion, welche das Optimierungsziel implementiert, ist dann nichts anderes als ein Klassifikator, der ein übergebenes Szenario der Klasse X oder der Klasse $\bar{X}$ zuordnet. Die Einordnung erfolgt nach dem Erfüllungsgrad des Optimierungsziels im Sinne einer Fuzzy-Definition, die den Grad der Zugehörigkeit eines Elements zu einer Klasse ausdrückt.

Dieser Schritt bedeutet eine Spezialisierung, denn er erfordert eine Entwurfsentscheidung. Man legt sich auf eine von eventuell mehreren möglichen Ergebnisklassen fest, hier im Beispiel ist es die Ergebnisklasse X. Eine solche Entscheidung für eine Klasse ist gleichzeitig eine Entscheidung gegen andere Ergebnisklassen.

Sollen verschiedene Anforderungen verletzt werden, so ist für jede Anforderung getrennt ein eigenes Optimierungsziel festzulegen. Daher ist eine Reihe unabhängiger Testläufe notwendig, um diese verschiedenen Optimierungsziele zu verfolgen. Die Testläufe sind voneinander nicht kausal abhängig. Selbst wenn durch verschiedene Zielfunktionen zu einem Individuum mehrere Zielfunktionswerte berechnet werden können, ist dies keine Problemstellung für eine multi-kriterielle Optimierung nach der Pareto-Optimalität[1], denn es reicht aus, einen der Zielfunktionswerte zu minimieren (maximieren) unabhängig von den anderen Zielfunktionswerten. In anderen Worten, die verschiedenen Fehler können in sequentieller Reihenfolge gefunden werden. Ein Finden zur selben Zeit ist nicht erforderlich, denn wie beim manuellen Testen ist es ja auch nicht erforderlich, einen Testfall zu finden, bei dem zwei Anforderungen gleichzeitig nicht eingehalten werden. Es ist ausreichend, einen Testfall zu finden, bei dem eine einzige Anforderung nicht eingehalten wird.

4.3.4 Klassifikationsmerkmale

In diesem Schritt müssen charakteristische Klassifikationsmerkmale identifiziert werden, die eine Zuordnung eines übergebenen Szenarios zur Ergebnisklasse des Optimierungsziels ermöglichen. Dabei ist zu untersuchen, wie sich Elemente aus der Ergebnisklasse des Optimierungsziels von den anderen Elementen unterscheiden. Es sind Merkmale zu suchen, anhand derer sich eine Zuordnung vornehmen lässt. Eine Möglichkeit wäre, dass es Signalverläufe bei Elementen entsprechend dem Optimierungsziel gibt, die eine Korrelation aufweisen, die bei anderen Elementen keine Korrelation aufweisen.

Ein Merkmal zu finden, ist eine Aufgabe, die kein eindeutiges Ergebnis haben muss und letztlich in heuristischer Weise durchgeführt wird. Die Tätigkeit der Identifikation von charakteristischen Merkmalen ist somit eine Ingenieursleistung. Eine Hilfestellung kann dabei geben, auf was ein manueller Testfallbewerter achtet, wenn er ein Testergebnis bezüglich der ausgewählten Anforderung beurteilt. Charakteristisch könnte sein, dass zwei Signalverläufe in einem bestimmten Verhältnis stehen müssen. Oder es könnte sein, dass ein bestimmter Wert zu einem Ereignis vorhanden sein muss.

Gibt es mehrere alternative Klassifikationsmerkmale anhand derer eine Klassifikation möglich ist, muss für die nachfolgende Umsetzung in eine mathematische Funktion eine Auswahl getroffen werden. Diese Auswahl bedeutet wieder eine Spezialisierung, denn es wird eine Art von Klassifikationsmerkmal von eventuell verschiedenen ausgewählt.

4.3.5 Merkmalsbasierte Zielfunktion

Das ausgewählte Klassifikationsmerkmal ist in diesem Schritt der Ausgangspunkt, um eine Zielfunktion zu formulieren. Die Zielfunktion muss in einen Algorithmus umgesetzt werden,

[1] Die Pareto-Optimalität ist in [108] und [27] dargestellt.

so dass sie ein Maß im Sinne des Optimierungsziels bildet. Auf Basis des charakteristischen Merkmals sollen zu einem beliebigen Testergebnis der Grad der Zugehörigkeit zu einer Ergebnisklasse berechnet werden. Dazu muss das Merkmal mit Hilfe eines mathematischen Algorithmus ausgewertet werden. Der durch den Algorithmus berechnete Wert für ein Szenario bildet den einzigen Rückgabewert der Zielfunktion. Dieser soll eine möglichst kontinuierliche Zuordnung des Szenarios zur Ergebnisklasse des Optimierungsziels darstellen. Durch den kontinuierlichen Wert soll die Optimierung in Richtung des Optimierungsziels geleitet werden.

Das Problem dabei sind Szenarien, bei denen durch die Zielfunktion kein kontinuierlicher Wert berechnet werden kann, da z.B. die zu testende Anwendungsfunktion gar nicht aktiv wurde. In diesem Fall benötigt man eine Zielfunktion mit Nebenbedingung, die zunächst prüft, ob eine Bewertung überhaupt möglich ist, d.h. ob die Anwendungsfunktion aktiv wurde. Eine solche zusätzliche Bedingung bedeutet, dass der Suchraum von „Sperrgebieten“ durchzogen ist, die nicht betreten werden sollen. Eine Möglichkeit mit diesem Problem umzugehen, ist die Zuweisung eines angemessenen „Strafwertes“[2]. Dieser Strafwert soll die Optimierung davon abhalten, Testdaten zu generieren, bei denen eine Bewertung des Szenarios nicht sinnvoll ist, da z.B. die Anwendungsfunktion nicht aktiv wurde. Ein Nachteil dieser Methode liegt darin, dass der Übergang zwischen gewünschten und unerwünschten Lösungen nicht fließend ist [64].

Bei diesem Schritt spezialisiert man sich auf einen ausgewählten Algorithmus, der ein gegebenes Klassifikationsmerkmal auswertet. Stehen mehrere alternative Algorithmen für eine Auswertung zur Verfügung, muss eine Variante ausgewählt werden. Der gewählte Algorithmus sollte dabei möglichst effizient zu implementieren sein, um die Anzahl der durchführbaren Tests in einer vorgegebenen Zeiteinheit möglichst hoch zu halten.

4.4 Einordnung in den Entwicklungsprozess

Dieses Kapitel ordnet den evolutionären Funktionstest in den in Kapitel 3.1.2 vorgestellten Entwicklungsprozess von eingebetteten Systemen im Automobil ein. Dabei wird diskutiert, auf welche Produkte des Entwicklungsprozesses der evolutionäre Funktionstest angewandt werden kann. Darüber hinaus werden die Abhängigkeiten des evolutionären Funktionstests von Produkten des Standard-Entwicklungsprozesses berücksichtigt.

Der evolutionäre Funktionstest kann einen konventionellen Standard-Testprozess nicht ersetzen. Dieser ist auf jeden Fall auszuführen. D.h. der evolutionäre Funktionstest ist eine wertvolle Ergänzung des manuellen Tests mit dem Ziel, eine Verletzung besonders wichtiger Anforderungen zu erkennen. Die manuelle Testfalldefinition läuft ganz normal wie im Falle ohne einen evolutionären Funktionstest.

[2] Ein „Strafwert“ (engl. Penalty) ist eine Bewertungszahl welche die Zielfunktion zurückliefert, wenn während der Suche ein Punkt im Suchraum erreicht wurde, der für das Optimierungsziel nicht relevant ist. In der vorliegenden Arbeit liefert eine Zielfunktion einen Strafwert zurück, wenn in einem Szenario die zu bewertende Anwendungsfunktion nicht aktiv wurde.

Wie in Abbildung 4.9 dargestellt, können grundsätzlich die folgenden Entwicklungsprodukte mit einem evolutionären Funktionstest getestet werden: Steuergerät (10a.2), A-Muster (10a.1), SW-Komponente (8a), SW-Teilkomponente Autocode (7a.1) und Handcode (7a.2), sowie bereits auf der Zerlegungsseite das Logische Modell (5b.1) und das Implementierungsmodell (5e.1). Voraussetzung dazu ist, dass für diese Entwicklungsprodukte jeweils eine geeignete Testumgebung vorliegt, wie sie in Kapitel 3.1.2 vorgestellt wurde und wie sie normalerweise in einem Standard-Testprozess erstellt wird.

Für den Aufbau eines Testsystems für einen evolutionären Funktionstest entsprechend Abbildung 4.1 entsteht dann lediglich ein Zusatzaufwand für die Codierung der Testdaten und den Entwurf der Zielfunktion. Die Codierung der Testdaten und die Zielfunktion müssen dabei im Projekt mit zeitlichem Vorlauf begonnen werden. Das Testsystem für den evolutionären Funktionstest kann dabei parallel zum Standard-Entwicklungsprozess aufgebaut werden. Zum Zeitpunkt der Testdurchführung kann dann parallel zum Standard-Entwicklungsprozess mit dem evolutionären Funktionstest getestet werden.

Ein Entwurf der Zielfunktion kann erst erfolgen, wenn die Anforderungen an das Fahrzeug formuliert sind. Parallel zur manuellen Definition von Testfällen für einen Standard-Testprozess können Optimierungsziele für den evolutionären Funktionstest definiert und mit dem Entwurf geeigneter Zielfunktionen begonnen werden. Ebenso kann ab hier das Modell zur Codierung der Testdaten entworfen werden, da sich die Testverantwortlichen im Standard-Entwicklungsprozess ebenfalls mit dem funktionalen Eingabedatenraum beschäftigen. Dabei ist das Ziel, die Codierung der Testdaten so zu entwerfen, dass alle manuell definierten Testfälle für das Fahrzeug damit abgebildet werden können. Dies ist eine Voraussetzung für das in Kapitel 4.1 beschriebene „Seeding“. Die Aktivitäten zum Entwurf der Zielfunktionen und zum Entwurf der Codierung der Testdaten profitieren also beide von den Aktivitäten des Standard-Entwicklungsprozesses.

Wie bereits in Kapitel 4.1 „Seeding“ dargestellt, können dabei die manuellen Testfälle als initiale Population für den evolutionären Funktionstest verwendet werden. Das Finden der manuellen Testdaten bedeutet hier also keinen Zusatzaufwand.

Wie aus Abbildung 4.9 ersichtlich kann der evolutionäre Funktionstest im Falle einer modellbasierten Entwicklung nicht erst für das Implementierungsmodell eingesetzt werden, sondern bereits für das logische Modell. Damit können funktionale Fehler bereits in der Phase der Systemanalyse gefunden werden.

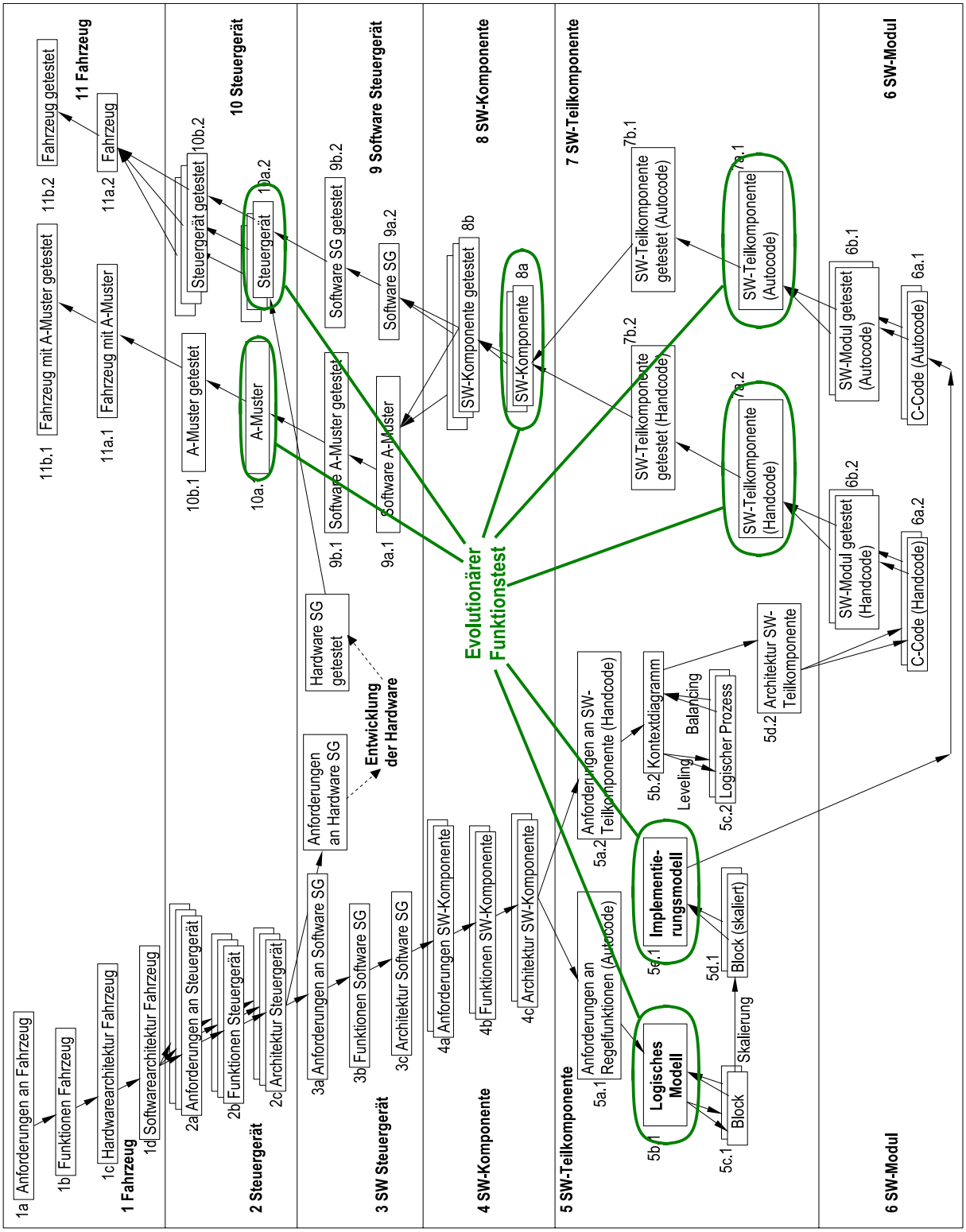


Abbildung 4.9: Mögliche Testobjekte für den evolutionären Funktionstest.

Kapitel 5

Evolutionärer Funktionstest für Fahrerassistenzsysteme

Abstandsbasierte Fahrerassistenzsysteme basieren auf der Erweiterung des Fahrzeugs um umgebungserfassende Sensoren. Dabei wird das Fahrzeug mit Sensoren und elektronischen Systemen ausgestattet, welche die Fahrzeugumgebung wahrnehmen und interpretieren [21]. Diese ermöglichen das blitzschnelle Erfassen einer Umfeldsituation in der Längs- und Querbewegung des Fahrzeugs [29]. Nach Ammon [2] wird durch Nutzung der umgebungserfassenden Sensorik die Systemgrenze deutlich über das Fahrzeug hinaus auf die Fahrzeugumgebung erweitert. Dabei beschreibt der Begriff *Systemfahrzeug* das Fahrzeug mit dem Assistenzsystem. Die Begriffe *Zielfahrzeug* bzw. *Zielobjekt* beschreiben ein Fahrzeug bzw. Objekt, das durch die Sensorik erfasst wird. Im Kontext des Anwendungsbereichs abstandsbasierter Fahrerassistenzsysteme sind die Anforderungen meist aus der Sicht des Fahrers bzw. des Fahrzeugs definiert. Aus dieser Perspektive werden Situationen beschrieben durch Fahrzeuge oder Objekte, die sich in der Fahrzeugumgebung befinden und von dessen Sensorik erfasst werden. Für die Reaktion eines Fahrers im Verkehrsgeschehen sind oft relative Größen von Bedeutung. Fährt beispielsweise ein Fahrer einem vorausfahrenden Fahrzeug mit etwa derselben Geschwindigkeit nach, so tritt er auf die Bremse, wenn der Abstand zum vorausfahrenden Fahrzeug kleiner wird als der gefühlte Sicherheitsabstand. Nähert sich ein Fahrer mit hoher Geschwindigkeit einem langsam vorausfahrenden Fahrzeug, so tritt er in Abhängigkeit von der Relativgeschwindigkeit und seiner Risikobereitschaft bei einem bestimmten Abstand auf die Bremse. Solche Situationen lassen sich beschreiben durch sogenannte Datenbedingungen, wie sie von Hatley und Pirbhai in [72] in der Strukturierten Analyse für Echtzeitsysteme beschrieben werden.

Die Codierung eines Individuums für den evolutionären Funktionstest eines abstandsbasierten Fahrerassistenzsystems erfordert daher oftmals die Abbildung von Testfällen, die eine solche Ereignissteuerung beinhalten. Die Begründung dafür ist, dass das funktionale Verhalten dieser Art von Anwendungen oft ereignisgesteuert definiert ist und dementsprechend beim Testen überprüft werden muss.

Eine mögliche Vorgehensweise beim Entwurf der Codierung ist, die manuell definierten Testfälle wie z.B. den Fahrmanöver-Katalog als Ausgangsbasis zu nehmen und diese bereits definierten Testfälle auf Ereignisse und Aktionen hin zu analysieren. Dabei durchsucht man den Text nach Ereignissen, die auftreten, und Aktionen, die erfolgen. Ebenso ist wichtig zu erkennen, wie die Ereignisse und Aktionen miteinander verknüpft sind. Anschließend erfolgt der Entwurf der Codierung. Codieren lässt sich dabei die Parametrisierung der Datenbedingungen für die Ereignisse. Ebenso können die Aktionen beispielsweise in Form von zeitlichen Signalverläufen codiert werden. Zusätzlich muss die Zuordnung von Aktionen zu Ereignissen abgebildet werden.

5.1 Das „Automatische Parksystem“

Das „Automatische Parksystem“ ist ein Fahrerassistenzsystem zur Unterstützung des Fahrers beim Einparken in Längsparklücken. Ein Prototyp eines solchen Systems wurde im Rahmen eines Vorentwicklungsprojekts entwickelt und untersucht. Die Entwicklung und Implementierung dieser Fahrerassistenzfunktion wurde unterstützt durch Modellbildung und Simulation. Damit war es möglich, die Anwendungsfunktion entwicklungsbegleitend in einer Simulationsumgebung zu testen. Diese Simulationsumgebung wurde zu einer Testumgebung für einen evolutionären Funktionstest erweitert und bildet die Basis für die in diesem Abschnitt dargestellten experimentellen Untersuchungen. Ein evolutionärer Funktionstest mit dieser Anwendungsfunktion wurde bereits von Bühler und Wegener in [92] vorgestellt.

5.1.1 Anwendung

Aufgrund der Größe und der aerodynamisch optimierten Form eines modernen Oberklassefahrzeugs ist es für einen ungeübten Fahrer oft schwierig, in Längsparklücken einzuparken. Das „Automatische Parksystem“ bietet hier dem Fahrer eine Funktionalität, mit der er das Fahrzeug automatisch in eine Längsparklücke einparken lassen kann. Das Fahrzeug ist dazu mit einer umgebungserfassenden Sensorik ausgerüstet, welche Objekte auf der rechten und linken Seite des Fahrzeugs erkennen kann. In Abbildung 5.1 ist dargestellt, wie das „Automatische Parksystem“ arbeitet.

Abbildung 5.1: Vorbeifahrt und Einparkvorgang des „Automatischen Parksystems“

Bei der Suche nach einer Längsparklücke entlang einer Straße kann die umgebungserfassende Sensorik während der Vorbeifahrt die Größe von potentiellen Parklücken erkennen

und vermessen. Wird dabei eine Längsparklücke mit einer ausreichenden Größe erkannt, wird dies dem Fahrer über eine Anzeige mitgeteilt. Der Fahrer hat nun die Möglichkeit, das Fahrzeug zu stoppen und dem „Automatischen Parksystem“ mitzuteilen, dass er in die gefundene Längsparklücke einparken möchte. Entscheidet sich der Fahrer dazu, in die Parklücke einzuparken, erfolgt der Einparkvorgang automatisch und das System parkt das Fahrzeug automatisch in die Längsparklücke ein.

Das „Automatische Parksystem“ besteht aus den Komponenten „Umgebungserfassende Sensorik“, „Fahrzeug-Sensorik“, „Fahrzeug-Aktorik“, „Parklückenerkennung“ und „Einparksteuerung“. Dies ist im Datenflussdiagramm in Abbildung 5.2 dargestellt. Die „Umgebungserfassende Sensorik“ erkennt Objekte auf der linken und rechten Seite des Fahrzeugs. Die „Fahrzeug-Sensorik“ liefert Informationen über die Bewegung des Fahrzeugs, aus denen die Position und Ausrichtung des Fahrzeugs berechnet werden kann. Die „Fahrzeug-Aktorik“ steuert die Geschwindigkeit und den Lenkwinkel des Fahrzeugs an. Die „Parklückenerkennung“ verarbeitet die Daten der „Umgebungserfassenden Sensorik“ und liefert die erkannte Geometrie einer Parklücke in Form von fünf Geometriepunkten. Die „Einparksteuerung“ nutzt diese Information der Geometriepunkte über die Parklücke zusammen mit den Daten der „Fahrzeug-Sensorik“. Mit diesen Daten steuert die „Einparksteuerung“ die „Fahrzeug-Aktorik“ an, um das Fahrzeug in die Parklücke zu bringen. Dabei werden Soll-Werte für die Geschwindigkeit und den Lenkwinkel an die „Fahrzeug-Aktorik“ übergeben.

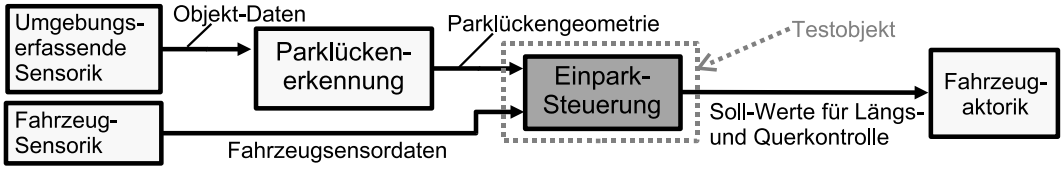


Abbildung 5.2: Datenflussdiagramm für die Komponenten des „Automatischen Parksystems“

5.1.2 Codierung

Die Codierung der Testdaten für das „Automatische Parksystem“ hat die Aufgabe, einen frei wählbaren Suchraum auf eine Menge von für die Anwendung sinnvollen Testdaten abzubilden. Sinnvolle Testdaten für das „Automatische Parksystem“ sind alle Parksituationen, die in der Realität vorkommen können.

Die für das „Automatische Parksystem“ gewählte Codierung der Testdaten basiert auf einem Modell, das rechtwinklige Parklücken erzeugen kann. In Abbildung 5.3 ist die Wirkungsweise des Modells dargestellt. Die Codierung des Modells benötigt fünf Gene in einem Individuum zur Definition eines Testfalls, d.h. einer vorgegebenen Parksituation. Diese Codierung spannt also einen 5-dimensionalen Suchraum auf. Die Struktur der Gene für die Codierung ist in der nachfolgenden Tabelle 5.1 dargestellt, die Bedeutung der einzelnen Gene ist in Abbildung 5.3 ersichtlich.

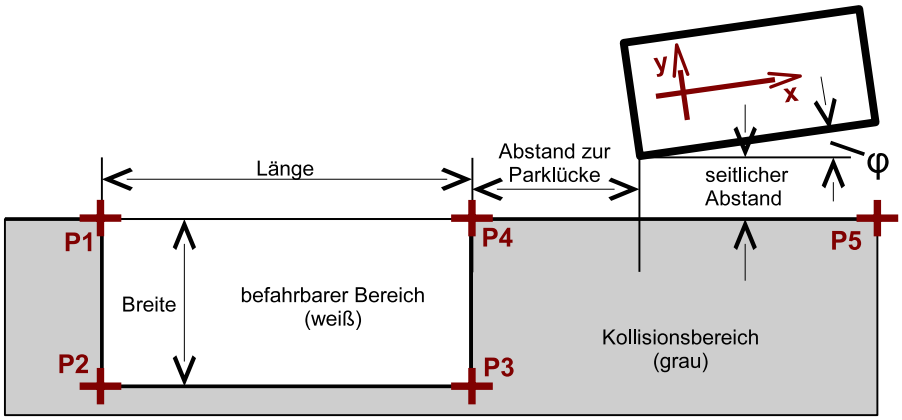


Abbildung 5.3: Modell zur Generierung der Testdaten

Nr.	Bedeutung	Typ	Wertebereich	Maßeinheit
Gen 1	Länge	I	0 bis 20	$[m]$
Gen 2	Breite	I	0 bis 10	$[m]$
Gen 3	seitlicher Abstand	I	0 bis 10	$[m]$
Gen 4	Abstand zur Parklücke	I	0 bis 20	$[m]$
Gen 5	Verdrehungswinkel φ	I	-10 bis +10	$[DEG]$

Tabelle 5.1: Gene zur Codierung eines Park-Szenarios

Die Codierung der Testdaten bildet die im Modell definierte Parksituation auf die Eingangswerte der Testumgebung ab. Innerhalb der Testumgebung wird die Situation durch die Geometriepunkte der Parklücke definiert. Diese Geometriepunkte bestehen aus den x-/y-Koordinaten von fünf Punkten $P1$ bis $P5$, wie in Abbildung 5.3 dargestellt. Sie bilden die Kontur der Parklücke und definieren die Parksituation aus Sicht des „Automatischen Parksystems“. Die Kontur der Parklücke legt die Grenze zwischen dem befahrbaren Bereich und dem Kollisionsbereich fest. Die Definition der Geometriepunkte erfolgt relativ zu einem Bezugssystem, das aus Sicht des Fahrzeugs festgelegt wurde.

Das dargestellte Modell bildet einen Punkt aus dem durch die fünf Eingangsparameter definierten 5-dimensionalen Suchraum ab auf die x-/y-Koordinaten von fünf Punkten, welche die Parksituation eindeutig definieren. Die berechneten x-/y-Koordinaten werden über die Testumgebung dem „Automatischen Parksystem“ vorgegeben.

Das Modell ist durch die so festgelegten Eingangsparameter nicht überbestimmt. D.h. keiner der Parameter lässt sich in einer Parksituation aus den anderen berechnen. Darüber hinaus kann durch eine geeignete Festlegung der Suchraumgrenzen eine Erzeugung von nicht sinnvollen Parksituationen ausgeschlossen werden. Unter „nicht sinnvoll“ werden hier Situationen verstanden, bei denen das Fahrzeug zu Beginn bereits im Kollisionsbereich steht.

5.1.3 Zielfunktion

Die Zielfunktion bewertet ein Park-Szenario, das von dem „Automatischen Parksystem“ in der Simulationsumgebung durchgeführt wurde. Nachfolgend wird eine Zielfunktion für das „Automatische Parksystem“ nach der in Kapitel 4.3 dargestellten Vorgehensweise entwickelt.

Ergebnisklassen

Entsprechend der Vorgehensweise müssen zuerst die Ergebnisklassen modelliert werden. Dazu muss eine Anforderung ausgewählt werden, die nachfolgend weiter analysiert wird. Eine elementare Anforderung für die Anwendungsfunktion des „Automatischen Parksystems“ ist:

Anforderung: Wenn das „Automatische Parksystem“ die gegebene Parksituation akzeptiert und das Fahrzeug einparkt, dann soll das Fahrzeug beim Einparken den Kollisionsbereich der Parklücken-Geometrie nicht betreten.

Das „Automatische Parksystem“ soll das Fahrzeug in eine Parklücke rangieren, ohne mit einem Hindernis (z.B. einem stehenden Fahrzeug) zu kollidieren. Dazu muss – neben einer zuverlässigen Erkennung und Vermessung der Parklücken-Geometrie durch die umgebungserfassende Sensorik und die Komponente „Parklückenerkennung“ – die Einpark-Trajektorie von der Komponente „Einpark-Steuerung“ so berechnet werden, dass das Fahrzeug den erkannten Kollisionsbereich nicht betritt. Alternativ kann die „Einpark-Steuerung“ den Einparkvorgang als nicht durchführbar ablehnen.

Ein Fehler in der Komponente „Einpark-Steuerung“ bezogen auf die ausgewählte Anforderung tritt auf, wenn ein Park-Szenario gefunden wird, bei dem das Fahrzeug den Kollisionsbereich betritt. Somit ist das Ziel der Optimierung, beim Einparken eine Kollision zu verursachen. Dieses generelle Ziel der Optimierung lässt sich weiter detaillieren, da eine Kollision während eines Manövers an verschiedenen Stellen passieren kann.

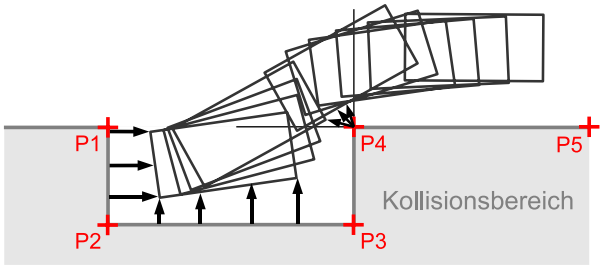


Abbildung 5.4: Mögliche Kollisionen eines Park-Szenarios

Wie in Abbildung 5.4 dargestellt, kann es bei der Vorbeifahrt an der Ecke P_4 zur Kollision kommen oder das Fahrzeug kann beim Einfahren in die Parklücke nach dem Umlenkpunkt zu weit auf die Seite kommen. Dann wird die Strecke $\overline{P_2P_3}$ überfahren. Oder das Fahrzeug wird zu spät angehalten und die Strecke $\overline{P_1P_2}$ überfahren.

Mit dieser Erkenntnis lassen sich nun mögliche Park-Szenarien in verschiedene Ergebnisklassen einordnen. In Abbildung 5.5 sind die Ergebnisklassen A, B, C und D dargestellt, wie man sie aufgrund der Analyse der ausgewählten Anforderung bilden kann. Zunächst kann man unterscheiden zwischen Szenarien, bei denen es eine Kollision gibt und Szenarien, die ohne Kollision ablaufen. Die Ergebnisklasse A enthält alle Szenarien ohne Kollision. Darunter fallen auch die Szenarien, bei denen die Parksituation abgelehnt wird. Die Vereinigungsmenge $B \cup C \cup D$ enthält die Szenarien mit Kollision.

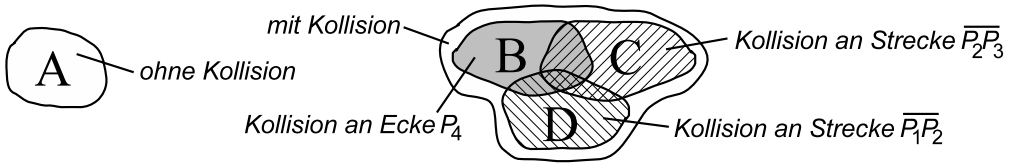


Abbildung 5.5: Betrachtete Ergebnisklassen der Anforderung *Einparken ohne Kollision*

Ergebnisklasse	Bedeutung
A	Szenarien ohne Kollision
B	Szenarien mit Kollision an der Ecke P_4
C	Szenarien mit Kollision an der Strecke $\overline{P_2P_3}$
D	Szenarien mit Kollision an der Strecke $\overline{P_1P_2}$

Tabelle 5.2: Ergebnisklassen der Anforderung *Einparken ohne Kollision*

Dabei sollen der Ergebnisklasse B alle Szenarien mit Kollision an der Ecke P_4 zugeordnet werden. Die Ergebnisklasse C soll die Szenarien mit Kollision an der Strecke $\overline{P_2P_3}$ enthalten und die Ergebnisklasse D soll die Szenarien mit Kollision an der Strecke $\overline{P_1P_2}$ umfassen. Da es Szenarien geben kann, bei denen das Fahrzeug an der Ecke P_4 und an der Strecke $\overline{P_2P_3}$ kollidiert, überlappen sich die Ergebnisklassen B und C. Die Szenarien mit Kollision an der Ecke und an der Seite befinden sich dann in der Schnittmenge $B \cap C$. Analog kann es Szenarien geben, bei denen das Fahrzeug an der Strecke $\overline{P_2P_3}$ und an der Strecke $\overline{P_1P_2}$ kollidiert. Diese befinden sich dann in der Schnittmenge $C \cap D$.

Es ist zu beachten, dass durch die Einordnung der Testergebnisse in die Ergebnisklassen A oder $B \cup C \cup D$ nicht zwischen *richtig* oder *falsch* im Sinne aller Anforderungen der Anwendungsfunktion unterschieden wird. Die Unterscheidung zur Zuordnung der Testergebnisse zu den Ergebnisklassen erfolgt ausschließlich bezüglich einer gewählten Anforderung und bezüglich eines gewählten Optimierungsziels. Dies bedeutet, die Ergebnisklasse A, welche alle Szenarien ohne Kollision beinhaltet, enthält ebenfalls fehlerhafte Szenarien, wie z.B. ein Szenario, bei dem ein durchführbarer Einparkvorgang abgelehnt wird oder den Fall, dass die Linie $\overline{P_1P_4}$ vom Fahrzeug noch überschritten wird.

Optimierungsziel

Aus den modellierten und in Abbildung 5.5 dargestellten Ergebnisklassen lassen sich also drei unterschiedliche Optimierungsziele für Szenarien mit Kollision formulieren:

1. „Kollision am Punkt P_4“
2. „Kollision an der Strecke $\overline{P_2P_3}$“
3. „Kollision an der Strecke $\overline{P_1P_2}$“

Für jedes der Optimierungsziele wird eine eigene Zielfunktion benötigt. Im Folgenden werden für die beiden Optimierungsziele „Kollision am Punkt P_4“ und „Kollision an der Strecke $\overline{P_2P_3}$“ entsprechende Zielfunktionen getrennt voneinander entwickelt. Das Optimierungsziel „Kollision an der Strecke $\overline{P_1P_2}$“ wird nachfolgend nicht weiter betrachtet.

„Kollision am Punkt P_4“ Die betrachteten Ergebnismengen für das Optimierungsziel „Kollision am Punkt P_4“ sind in Abbildung 5.6 dargestellt.

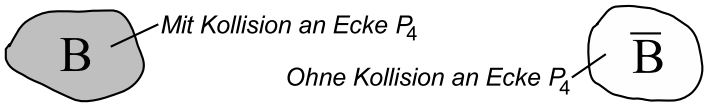


Abbildung 5.6: Ergebnismengen des Optimierungsziels „Kollision am Punkt P_4“

Das Ziel ist es, Szenarien zu finden, bei denen eine Kollision an der Ecke P_4 auftritt. Diese Szenarien sollen mit der Zielfunktion der Ergebnisklasse B zugeordnet werden. Die Ergebnisklasse $\overline{B}$ ist das „Sammelbecken“ für alle anderen Szenarien, die im Sinne des Optimierungsziels nicht interessieren. Darin werden in diesem Fall alle Szenarien eingeordnet, bei denen es keine Kollision an der Ecke P_4 gab. Dazu gehören Szenarien, die überhaupt ohne Kollision abgelaufen sind, z.B. weil die Parksituation abgelehnt wurde oder Szenarien, bei denen nur eine Kollision an einer anderen Stelle als der Ecke P_4 aufgetreten sind.

„Kollision an der Strecke $\overline{P_2P_3}$“ In Abbildung 5.7 sind die für das Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“ betrachteten Ergebnismengen dargestellt.

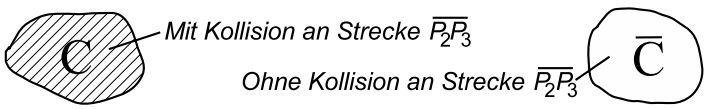


Abbildung 5.7: Ergebnismengen des Optimierungsziels „Kollision an der Strecke $\overline{P_2P_3}$“

Das Ziel ist es, Szenarien zu finden, bei denen eine Kollision an der Strecke $\overline{P_2P_3}$ stattgefunden hat. Die zu entwerfende Zielfunktion soll diese Szenarien der Ergebnisklasse C zuordnen. Die Ergebnisklasse $\overline{C}$ ist wiederum das „Sammelbecken“ für die uninteressanten Szenarien im Sinne des betrachteten Optimierungsziels. Hier werden die Szenarien zugeordnet, bei denen es keine Kollision an der Strecke $\overline{P_2P_3}$ gab. Dazu gehören wieder alle Szenarien, die überhaupt keine Kollision verursacht haben oder Szenarien, bei denen zwar eine Kollision stattgefunden hat, jedoch nicht an der Strecke $\overline{P_2P_3}$.

Klassifikationsmerkmale

Entsprechend der Vorgehensweise sollen nun charakteristische Merkmale gefunden werden, anhand derer ein vorliegendes Szenario klassifiziert und somit einer Ergebnisklasse zugeordnet werden kann. Diese Identifikation wird nachfolgend wieder getrennt für die beiden Optimierungsziele durchgeführt. Die gezeigten Kriterien zur Bewertung der beiden Optimierungsziele wurden bereits von Bühler und Wegener in [93] und [94] dargestellt.

„Kollision am Punkt P_4“ Als charakteristisches Merkmal zur Zuordnung eines beliebigen Park-Szenarios zu der Ergebnisklasse B kann man den Abstand zwischen dem Fahrzeug und der Geometrie des Kollisionsbereichs betrachten. Als Abstand wird dabei der geringste Abstand zwischen der Fahrzeughülle und dem Punkt P_4 betrachtet, der sich während eines Park-Szenarios ergibt. Dies ist in Abbildung 5.8 auf der linken Seite dargestellt.

Als Alternative könnte man als charakteristisches Merkmal auch die Fläche betrachten, die zwischen der gefahrenen Trajektorie, der Senkrechten durch den Punkt P_4 und der Waagerechten durch den Punkt P_4 eingeschlossen wird. Dies ist in Abbildung 5.8 auf der rechten Seite dargestellt.

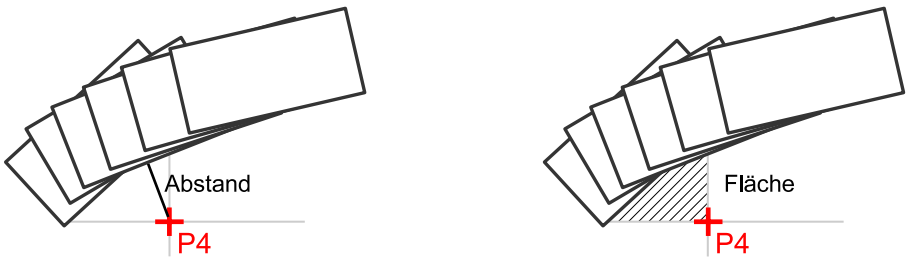


Abbildung 5.8: Charakteristische Merkmale für „Kollision am Punkt P_4“

Für die weitere Betrachtung wird hier der Abstand als Klassifikationsmerkmal ausgewählt. Interessant als Maßzahl für die *Schwere* der Kollision ist der kleinste Abstandswert (vom Betrag her der größte negative Wert), der in einem Szenario auftritt. Dieser Wert kann als *Grad* der Schwere der Kollision betrachtet werden und man kann ihn damit auch als Grad der Zugehörigkeit zur Ergebnisklasse B interpretieren.

„Kollision an der Strecke $\overline{P_2P_3}$“ Als charakteristisches Merkmal zur Zuordnung eines beliebigen Park-Szenarios zu der Ergebnisklasse C kann man den Abstand zwischen dem Fahrzeug und der Geometrie des Kollisionsbereichs betrachten. Als Abstand wird dabei der geringste Abstand zwischen der Fahrzeughülle und der Strecke $\overline{P_2P_3}$ betrachtet, der sich während eines Park-Szenarios ergibt. Dies ist in Abbildung 5.9 links dargestellt.

Alternativ könnte man hier auch die Fläche betrachten, die zwischen der gefahrenen Trajektorie des Punktes HR[1], der Strecke $\overline{P_2P_3}$, der Senkrechten in den Punkt P_3 und der Senkrechten in den Punkt HR in der Endposition der Trajektorie eingeschlossen wird. In Abbildung 5.8 ist diese Fläche im rechten Teilbild dargestellt.

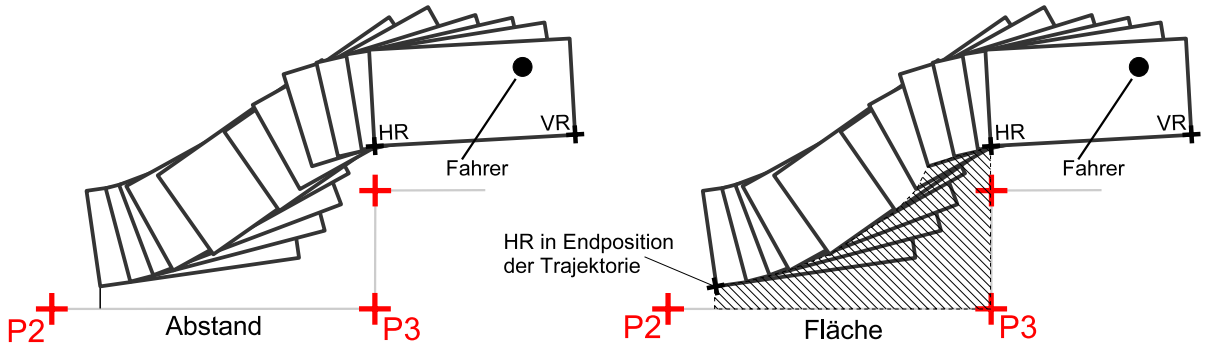


Abbildung 5.9: Charakteristische Merkmale für „Kollision an der Strecke $\overline{P_2P_3}$“

Für die weitere Betrachtung wird hier der Abstand als Klassifikationsmerkmal ausgewählt. Dieses Klassifikationsmerkmal soll durch die Zielfunktion ausgewertet werden.

Merkmalsbasierte Zielfunktion

Das festgelegte Klassifikationsmerkmal muss nun für jedes Optimierungsziel, also Zusammenstoß mit dem Punkt P_4 oder mit der Strecke $\overline{P_2P_3}$, getrennt durch die entsprechende Zielfunktion berechnet werden. Den beiden Zielfunktionen ist gemeinsam, dass bei einem abgelehnten Einparkvorgang durch die Zielfunktion ein Strafwert vergeben wird. Die Umsetzung der beiden Zielfunktionen wird nachfolgend wieder getrennt nach den Optimierungszielen vorgestellt.

„Kollision am Punkt P_4“ In Abbildung 5.10 ist die Arbeitsweise der Zielfunktion dargestellt, um den minimalen Abstand zwischen dem Fahrzeug und dem Punkt P_4 zu berechnen.

Zunächst muss für jede einzelne Fahrzeugposition der minimale Abstand des Fahrzeugs vom Punkt P_4 berechnet werden. Dazu wird durch Differenzbildung zwischen den Punkten HR

[1] Der Punkt HR bezeichnet die hintere rechte Ecke des Fahrzeugs

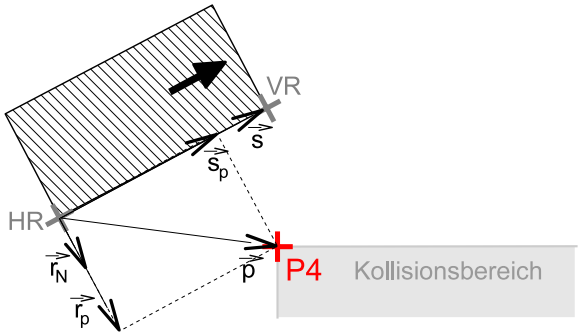


Abbildung 5.10: Berechnung des minimalen Abstands zum Punkt P_4

und P_4 der Vektor $\vec{p}$ berechnet. Ebenso wird der Vektor $\vec{s}$ durch die Differenz aus VR und HR gebildet.[2] Anschließend wird der Vektor $\vec{s_p}$ durch die senkrechte Projektion von $\vec{p}$ auf $\vec{s}$ berechnet. Mit Hilfe des Einheitsvektors $\vec{r_N}$, senkrecht zu $\vec{s}$, wird $\vec{r_p}$ durch die senkrechte Projektion von $\vec{p}$ auf den Einheitsvektors $\vec{r_N}$ gebildet.

Der Abstand zum Kollisionsbereich an der Ecke P_4 ist nun die Länge des Vektor $\vec{r_p}$, falls die Länge des Vektors $\vec{s_p}$ innerhalb von $\vec{s}$ liegt. Falls die Länge des Vektors $\vec{s_p}$ außerhalb von $\vec{s}$ liegt, wird der Abstand an dieser Fahrzeugposition auf NaN gesetzt.[3]

Der endgültige Wert der Zielfunktion für ein übergebenes Park-Szenario ergibt sich dann aus dem minimalen Wert der Abstände vom Punkt P_4 aller Fahrzeugpositionen.

„Kollision an der Strecke $\overline{P_2P_3}$“ In Abbildung 5.11 ist die Arbeitsweise der Zielfunktion dargestellt, um den minimalen Abstand zwischen dem Fahrzeug und der Strecke $\overline{P_2P_3}$ zu berechnen.

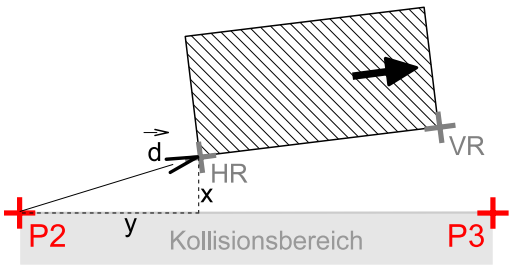


Abbildung 5.11: Berechnung des minimalen Abstands zur Strecke $\overline{P_2P_3}$

Zur Berechnung des minimalen Abstands an einer einzelnen Fahrzeugposition reicht es aus,

[2] Der Punkt VR bezeichnet die vordere rechte Ecke des Fahrzeugs, der Punkt HR bezeichnet die hintere rechte Ecke des Fahrzeugs.
[3] NaN = Not a Number – kennzeichnet einen ungültigen Wert

die Abstände von HR und VR zur Geraden durch P_2 und P_3 zu berechnen. Der kleinere von beiden Abständen gilt dann als der minimale Abstand für diese Fahrzeugposition.

Zur Berechnung des Abstands zwischen HR und der Geraden durch P_2 und P_3 wird zunächst der Vektor $\vec{d}$ berechnet. Anschließend lässt sich der Winkel α zwischen der Geraden P_2P_3 und dem Vektor $\vec{d}$ ermitteln. Über trigonometrische Funktionen lassen sich dann die Größen für x und y berechnen. Die Größe x wird dann als der Abstand zwischen HR und der Strecke $\overline{P_2P_3}$ genommen, falls die Größe von y innerhalb der Strecke $\overline{P_2P_3}$ liegt. Anderenfalls liegt der Punkt HR außerhalb des betrachteten Bereichs und der Abstandswert für diese Fahrzeugposition wird auf NaN gesetzt.

Der endgültige Wert der Zielfunktion für ein übergebenes Park-Szenario ergibt sich dann aus dem minimalen Wert der Abstände aller Fahrzeugpositionen.

5.2 Generierung der Populationen für das Parksystem

Im Folgenden wird für die Anwendung „Automatisches Parksystem“ gezeigt, in welchen Einzelschritten der evolutionäre Algorithmus die Populationen generiert. Den gezeigten Experimenten liegt die Toolbox von H. Pohlheim GEATbx [28] und ihre Beschreibung in [27] zugrunde.

Für die Optimierungen mit dem evolutionären Algorithmus im „Automatischen Parksystem“ wurde das sogannte Nachbarschaftsmodell, ein lokales Modell, zugrunde gelegt. In diesem Modell interagiert ein Individuum nur in seiner näheren Umgebung. Damit kann man eine Partitionierung erreichen und parallel in der Umgebung verschiedener Minima optimieren, wobei eine vorzeitige Konvergenz (premature convergence) in ein lokales Minimum vermieden wird. Dies wird auch als Isolation durch Distanz bezeichnet. Das gewählte lokale Modell hat vor allem Auswirkungen auf die Art der Elternselektion und des Wiedereinfügens.

Wie in [27] dargestellt ist, haben Collins und Jefferson in [61] verschiedene Selektionsmethoden untersucht und dabei das lokale Selektionsmodell mit einer einheitlichen Gesamtpopulation verglichen. Nach [27] ist „Besonders bei der Lösung schwieriger Probleme ... die Lokalität des lokalen Modells von Vorteil, da sich verschiedene Nischen mit Lösungen etablieren können, was zu einer robusteren Suche führt.“

Zur Demonstration der Wirkungsweise des evolutionären Algorithmus mit Hilfe eines Beispiels wurde der Verlauf des evolutionären Funktionstests am „Automatischen Parksystem“ zwischen Generation 10 und 11 ausgewählt. Bei diesem evolutionären Funktionstest beträgt die Populationsgröße 25 Individuen.

Wie in Kapitel 3.3.1 beschrieben, bestehen die einzelnen Schritte des evolutionären Algorithmus aus:

1. Fitnesszuweisung
2. Elternselektion
3. Rekombination
4. Mutation
5. Bewertung
6. Wiedereinfügen

Diese Schritte werden in den folgenden Kapiteln am Beispiel erklärt.

5.2.1 Fitnesszuweisung

Bei der Operation „Fitnesszuweisung“ (fitness assignment) wird für jedes Individuum der Population ausgehend vom Zielfunktionswert eines Individuums sein Fitnesswert berechnet. Da die verwendeten Operatoren der Elternselektion und des Wiedereinfügens keine Fitnesswerte verwenden, wird in diesem Schritt keine Berechnung vorgenommen.

5.2.2 Elternselektion

Im Schritt „Elternselektion“ erfolgt eine Auswahl von Individuen aus der Population zur Bildung von Nachkommen. Die Auswahl von Individuen zur Erzeugung von Nachkommen kann nach einer Reihe unterschiedlicher Verfahren erfolgen. Prinzipiell lassen sich rein stochastische Verfahren unterscheiden von Verfahren, die auf den Fitnesswerten bzw. den Zielfunktionswerten der Individuen basieren. Rein stochastisch arbeitende Verfahren erzeugen keinen Selektionsdruck.

Es wird eine Generationslücke (generation gap) von 0.9 verwendet[4]. Damit wird festgelegt, dass nur 0.9 mal so viele Nachkommen erzeugt werden, wie Individuen in der Population vorhanden sind. D.h. dass im Beispiel des „Automatischen Parksystems“ bei einer Populationsgröße von 25 Individuen $\lceil 25 \cdot 0.9 = 22.5 \rceil = 23$ Individuen als Nachkommen erzeugt werden.

Da jedes bei der Elternselektion ausgewählte Individuum in der nachfolgenden Rekombination ein Nachkomme hervorbringt, wird mit der Generationslücke bereits die Anzahl der durch die Elternselektion auszuwählenden Individuen festgelegt.

[4] Nach Pohlheim [27] soll bei einem lokalen Modell eine Generationslücke von knapp unter 1 verwendet werden.

Bei der Elternselektion nach dem Nachbarschaftsmodell werden die Individuen so ausgewählt, dass sie paarweise aus einer Nachbarschaft stammen. Dies lässt sich in zwei Schritten erreichen. Zunächst werden die Zentren der Nachbarschaften ausgewählt. Anschließend wird innerhalb jeder Nachbarschaft ein weiteres Elternteil gewählt. Da insgesamt 23 Individuen ausgewählt werden müssen, wird für das letzte Zentrum kein Partner aus seiner Nachbarschaft gewählt.

Für die Auswahl der Zentren werden im ersten Schritt gleichmäßig verteilt zufällig 12 Individuen aus der Population ausgewählt. Zu den ersten 11 dieser Individuen wird im zweiten Schritt jeweils ein Nachbar ausgewählt.

Für die Auswahl eines Nachbars werden die Individuen der Population zunächst in einer zweidimensionalen Topologie angeordnet. Die Anordnung der Individuen der gesamten Population erfolgt dabei in einer 5 x 5 Matrix, die auf einen Torus abgebildet wird. Dies ist in Abbildung 5.12 dargestellt. Mit dieser Anordnung gibt es keine Randknoten mehr, sondern nur noch innere Knoten.

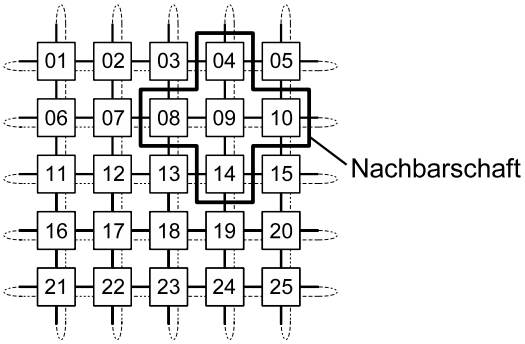


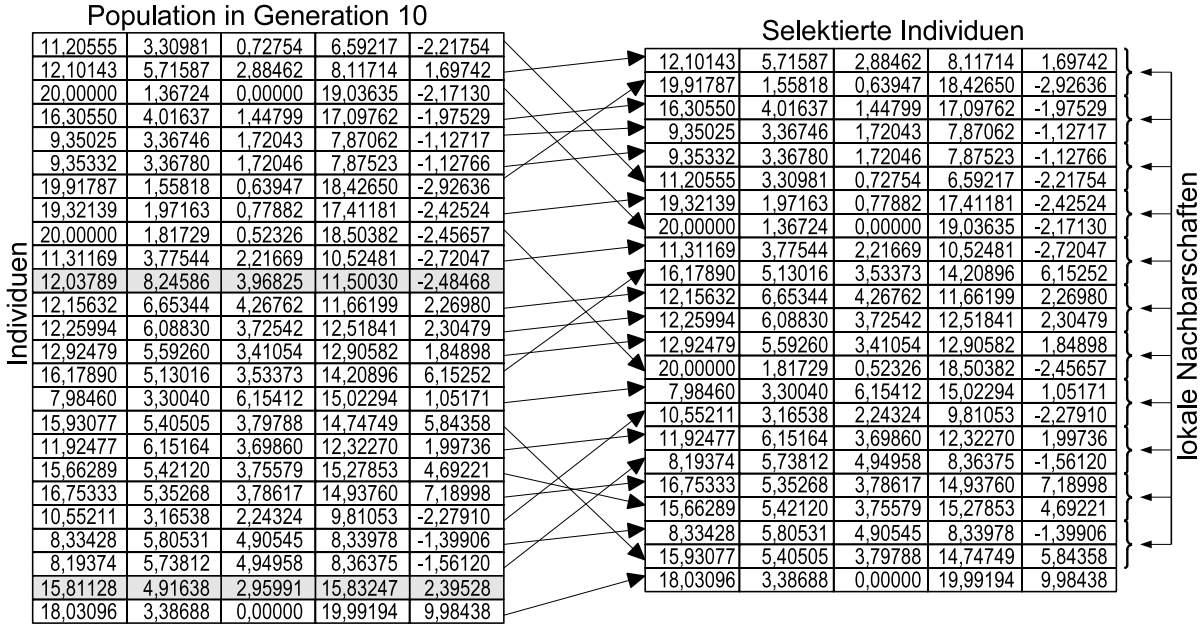
Abbildung 5.12: Zweidimensionale Topologie auf dem Torus.

Die Nachbarschaft wird als sogenanntes „ganzes Kreuz“ visualisiert. Wie in Abbildung 5.12 dargestellt entsteht dann die Form eines „ganzen Kreuzes“ mit einer Distanz von einem Element um ein Individuum, das als Zentrum ausgewählt wurde. Im Beispiel ist dies das Individuum 09, mit den Individuen 04, 08, 10 und 14 als dessen Nachbarn. Von diesen vier Nachbarn wird einer stochastisch ausgewählt.

Aufgrund einer gewählten Generationslücke von 0.9 und einer Populationsgröße von 25 Indivduen werden bei der Elternselektion 23 Individuen zur Bildung von Nachkommen ausgewählt. Dies bedeutet, dass zwei Individuen unverändert in die nächste Generation übernommen werden. Aufgrund der Anwendung eines lokalen Populationsmodells muss bei der Selektion bereits die gewählte Nachbarschaftstopologie in Form eines ganzen Kreuzes der Größe eins berücksichtigt werden.

In Abbildung 5.13 ist die Auswahl der Individuen aus der Population von Generation 10 dargestellt. Die Matrix auf der linken Seite zeigt die Individuen der Population in Generation

10. Jedes Individuum ist in einer Zeile mit seinen fünf Genen[5] dargestellt. Die Matrix auf der rechten Seite zeigt die selektierten Individuen, organisiert in ihren lokalen Nachbarschaften.



11,20555	3,30981	0,72754	6,59217	-2,21754
12,10143	5,71587	2,88462	8,11714	1,69742
20,00000	1,36724	0,00000	19,03635	-2,17130
16,30550	4,01637	1,44799	17,09762	-1,97529
9,35025	3,36746	1,72043	7,87062	-1,12717
9,35332	3,36780	1,72046	7,87523	-1,12766
19,91787	1,55818	0,63947	18,42650	-2,92636
19,32139	1,97163	0,77882	17,41181	-2,42524
20,00000	1,81729	0,52326	18,50382	-2,45657
11,31169	3,77544	2,21669	10,52481	-2,72047
12,03789	8,24586	3,96825	11,50030	-2,48468
12,15632	6,65344	4,26762	11,66199	2,26980
12,25994	6,08830	3,72542	12,51841	2,30479
12,92479	5,59260	3,41054	12,90582	1,84898
16,17890	5,13016	3,53373	14,20896	6,15252
7,98460	3,30040	6,15412	15,02294	1,05171
15,93077	5,40505	3,79788	14,74749	5,84358
11,92477	6,15164	3,69860	12,32270	1,99736
15,66289	5,42120	3,75579	15,27853	4,69221
16,75333	5,35268	3,78617	14,93760	7,18998
10,55211	3,16538	2,24324	9,81053	-2,27910
8,33428	5,80531	4,90545	8,33978	-1,39906
8,19374	5,73812	4,94958	8,36375	-1,56120
15,81128	4,91638	2,95991	15,83247	2,39528
18,03096	3,38688	0,00000	19,99194	9,98438

12,10143	5,71587	2,88462	8,11714	1,69742
19,91787	1,55818	0,63947	18,42650	-2,92636
16,30550	4,01637	1,44799	17,09762	-1,97529
9,35025	3,36746	1,72043	7,87062	-1,12717
9,35332	3,36780	1,72046	7,87523	-1,12766
11,20555	3,30981	0,72754	6,59217	-2,21754
19,32139	1,97163	0,77882	17,41181	-2,42524
20,00000	1,36724	0,00000	19,03635	-2,17130
11,31169	3,77544	2,21669	10,52481	-2,72047
16,17890	5,13016	3,53373	14,20896	6,15252
12,15632	6,65344	4,26762	11,66199	2,26980
12,25994	6,08830	3,72542	12,51841	2,30479
12,92479	5,59260	3,41054	12,90582	1,84898
20,00000	1,81729	0,52326	18,50382	-2,45657
7,98460	3,30040	6,15412	15,02294	1,05171
10,55211	3,16538	2,24324	9,81053	-2,27910
11,92477	6,15164	3,69860	12,32270	1,99736
8,19374	5,73812	4,94958	8,36375	-1,56120
16,75333	5,35268	3,78617	14,93760	7,18998
15,66289	5,42120	3,75579	15,27853	4,69221
8,33428	5,80531	4,90545	8,33978	-1,39906
15,93077	5,40505	3,79788	14,74749	5,84358
18,03096	3,38688	0,00000	19,99194	9,98438

Abbildung 5.13: Elternselektion aus Population in Generation 10

Es ist zu sehen, dass in diesem Beispiel die Individuen 11 und 24 nicht ausgewählt werden. Individuum 25 wird hier als Individuum ohne Nachbarschaft selektiert. Als Zentren für eine Nachbarschaft werden die Individuen 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 und 22 festgelegt. Als jeweilige lokale Nachbarn werden die Individuen 7, 5, 1, 3, 15, 13, 9, 21, 23, 19 und 17 ausgewählt. Damit werden insgesamt 23 Individuen selektiert. Die Auswahl der Zentren und der Nachbarn erfolgt in diesem Schritt rein stochastisch.

5.2.3 Rekombination

Die Rekombination sorgt für die Bildung von Nachkommen aus den ausgewählten Individuen. Als Rekombinationsfunktion wurde die Linien-Rekombination zwischen den benachbarten Paaren von Individuen gewählt. Dabei wird jedes Paar einer Nachbarschaft zweimal rekombiniert und erzeugt somit zwei Nachkommen. Jeder der Nachkommen wird formal einem Elternteil zugeordnet, um dieses bei der Operation „Wiedereinfügen“ ggf. zu ersetzen.

Die Linien-Rekombination ist ein Spezialfall der intermediären Rekombination, bei der für alle Variablenwerte der gleiche Skalierungsfaktor verwendet wird. Die Abbildung 5.14 zeigt

[5] Die fünf Gene sind die Länge und Breite der Parklücke, der Abstand von Fahrzeug zur Parklücke seitlich und längs, sowie der Verdrehungswinkel zur Parklücke, vgl. Kapitel 5.1.2.

die Darstellung der Linien-Rekombination nach [27]. Das Prinzip der Linien-Rekombination besteht darin, die Positionen der beiden Elternteile in einem n-dimensionalen Hyperraum durch eine Gerade zu verbinden. Der neu erzeugte Nachkomme wird als Punkt auf dieser gedachten Geraden gewählt. Wie in Abbildung 5.14 dargestellt, kann sich der Nachkomme sowohl auf der Geraden zwischen den beiden Eltern E1 und E2 befinden als auch in einem definierten Bereich außerhalb.

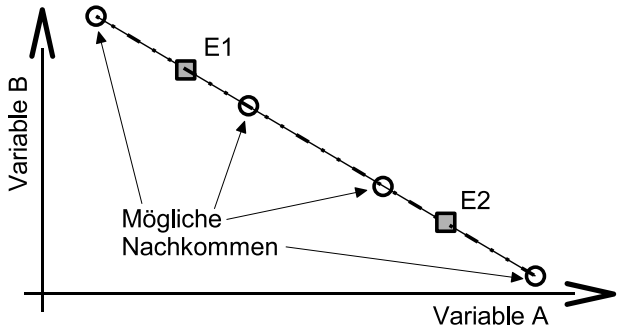


Abbildung 5.14: Darstellung der Linien-Rekombination

Das Gen k des Nachkommenvektors $\vec{N}$, d.h. die k-te Komponte des Nachkommenvektors, berechnet sich nach der Rekombinationsformel [27]:

$$N_k = a \cdot E1_k + (1 - a) \cdot E2_k \tag{5.1}$$

Der Skalierungsfaktor a bestimmt das Gewicht der Elternteile $E1$ und $E2$ bei der Vererbung. Hierbei ist $a \in \mathbb{R}$ eine gleichverteilte Zufallsvariable aus dem Bereich $[-d; 1 + d]$, wobei $d = 0.25$ gewählt wird. Der Wert von $d = 0.25$ sorgt nach [27] dafür, dass sich die gebildeten Nachkommen im statistischen Mittel im Rahmen des durch die Eltern vorgegebenen Fensters bewegen. D.h. in anderen Worten, dass über verschiedene Generationen hinweg dieses Fenster nicht immer enger wird.

Aus den 23 selektierten Individuen werden durch Rekombination wiederum 23 neue Individuen (rekombinierte Individuen, Nachkommen) erzeugt. Das lokale Populationsmodell bildet aus dem Selektionspool jeder lokalen Nachbarschaft seine Individuen durch intermediäre Rekombination.

Wie in Abbildung 5.15 zu sehen ist, werden die selektierten Individuen (links) paarweise rekombiniert und bringen jeweils zwei neue Individuen hervor. Da das unterste Individuum keiner lokalen Nachbarschaft angehört und somit keinen Rekombinationspartner besitzt, wird es unverändert übernommen.

Es ist zu beachten, dass durch die Rekombination die Bereichsgrenzen des Suchraums überschritten werden können. Dies ist im Beispiel in Abbildung 5.15 an den Individuen 13 und 14 in Gen 1 (Länge der Parklücke) zu erkennen. Diese Individuen tragen in ihrem Gen 1

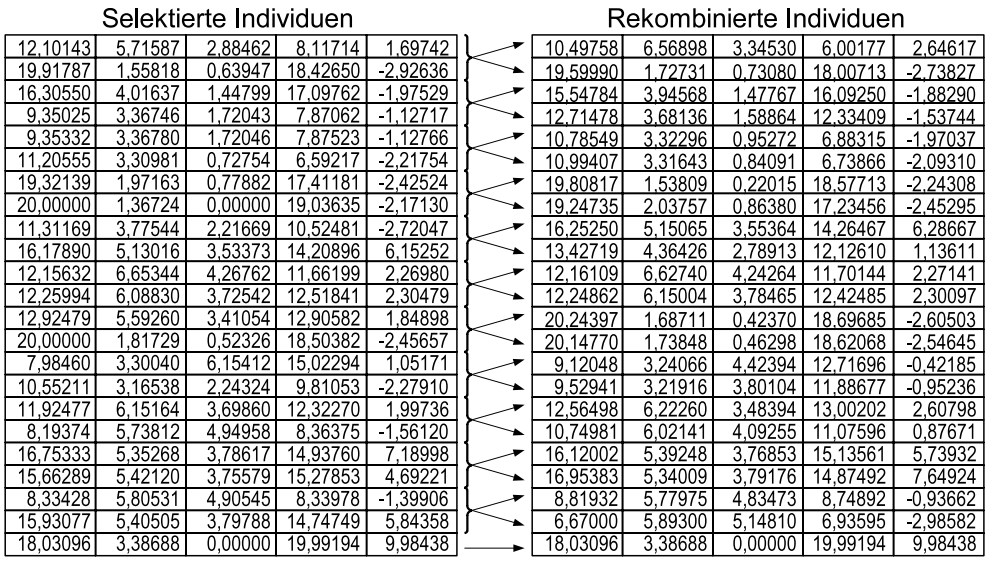


Abbildung 5.15: Bildung neuer, rekombinierter Individuen aus den selektierten Individuen

einen Wert > 20, obwohl der Suchraum bei diesem Gen auf eine obere Grenze von 20, d.h. eine maximale Länge der Parklücke von 20 m, begrenzt ist. Daher muss die nachfolgende Mutationsoperation die Begrenzung auf die vorgegebenen Suchraumgrenzen gewährleisten. Dies wird bei Überschreitung durch Auswahl der oberen bzw. unteren Grenze erreicht.

5.2.4 Mutation

Durch die Mutation werden die durch Rekombination neu erzeugten Individuen verändert. D.h. durch die Mutation wird dafür gesorgt, dass ein Nachkomme kein exaktes Abbild der Elterneigenschaften darstellt sondern einen zufälligen Anteil enthält. Durch diese fluktuierende Größe kann sichergestellt werden, dass jeder Punkt des Suchraums prinzipiell erreichbar ist. Die zur Optimierung verwendete Mutationsfunktion ist ein „Breeder Genetic Algorithm“ nach Mühlenbein [82]. Dabei wird ein Gen k eines Individuums durch die Mutation mit einer Wahrscheinlichkeit von 20% verändert. D.h. bei einer Anzahl von 5 Genen wird im statistischen Mittel jedes Individuum an einem Gen verändert.

Nach dem „Breeder Genetic Algorithm“ berechnet sich das mutierte Indiviuum $Indv^M$ aus dem rekombinierten Individuum $Indv^R$ durch:

$$Indv_k^M = Indv_k^R + m_k \cdot s_k \cdot \frac{1}{5} \cdot (G_k^{oben} - G_k^{unten}) \cdot d_k \text{ mit } k = 1..5 \tag{5.2}$$

Hierbei sind:

$$m_k = \begin{cases} 1 \text{ falls } x_1 < \frac{1}{5} \\ 0 \text{ falls } x_1 \geq \frac{1}{5} \end{cases} \quad \text{und} \tag{5.3}$$

$$s_k = \begin{cases} -1 \text{ falls } x_2 < \frac{1}{2} \\ +1 \text{ falls } x_2 \geq \frac{1}{2} \end{cases} \quad \text{und} \tag{5.4}$$

$$d_k = \sum_{i=0}^{15} \alpha_{k,i} \cdot 2^{(-i)} \text{ und} \tag{5.5}$$

$$\alpha_{k,i} = \begin{cases} 1 \text{ falls } x_{3+i} < \frac{1}{16} \\ 0 \text{ falls } x_{3+i} \geq \frac{1}{16} \end{cases} \quad \text{und} \tag{5.6}$$

$$\left.\begin{matrix} G_k^{oben} \text{ obere} \\ G_k^{unten} \text{ untere} \end{matrix}\right\} \quad \text{Grenzen des Suchraums der Komponente } k. \tag{5.7}$$

Dabei sei $\vec{x} = (x_1, x_2, ..., x_n)$ ein Vektor von n skalaren Zufallsvariablen, wobei x_n ein gleichverteilter Zufallswert im Bereich von 0 bis 1 ist. Die Mutationsfunktion des „Breeder Genetic Algorithm“ erfordert 18 Komponenten des Vektors $\vec{x}$ für jedes Gen des Individuums.

Rekombinierte Individuen						Mutierte Individuen				
10,49758	6,56898	3,34530	6,00177	2,64617	→	10,49758	6,56898	3,34530	6,00177	2,64617
19,59990	1,72731	0,73080	18,00713	-2,73827	→	19,59990	1,72731	0,00000	20,00000	-4,80077
15,54784	3,94568	1,47767	16,09250	-1,88290	→	15,54784	3,95362	1,47767	16,09250	-1,88290
12,71478	3,68136	1,58864	12,33409	-1,53744	→	12,71478	3,68136	1,58864	12,33409	-1,53744
10,78549	3,32296	0,95272	6,88315	-1,97037	→	10,78549	3,32296	0,95272	6,88315	-1,97037
10,99407	3,31643	0,84091	6,73866	-2,09310	→	10,99407	3,81643	1,34091	6,73866	-2,09310
19,80817	1,53809	0,22015	18,57713	-2,24308	→	19,80817	1,53809	0,22015	18,57713	-2,24308
19,24735	2,03757	0,86380	17,23456	-2,45295	→	19,24735	2,03757	0,86380	17,23456	-2,45295
16,25250	5,15065	3,55364	14,26467	6,28667	→	16,25237	5,15065	3,55364	14,26467	6,28667
13,42719	4,36426	2,78913	12,12610	1,13611	→	13,42719	4,36426	2,78913	12,12610	1,13611
12,16109	6,62740	4,24264	11,70144	2,27141	→	12,16109	6,67428	4,24264	11,70144	2,27141
12,24862	6,15004	3,78465	12,42485	2,30097	→	12,24862	6,15004	3,78465	12,42485	2,30097
20,24397	1,68711	0,42370	18,69685	-2,60503	→	20,00000	1,68711	0,42370	18,69685	-2,60503
20,14770	1,73848	0,46298	18,62068	-2,54645	→	20,00000	1,73848	0,46298	18,62068	-2,64020
9,12048	3,24066	4,42394	12,71696	-0,42185	→	9,12048	3,24066	4,42394	12,71696	-0,43748
9,52941	3,21916	3,80104	11,88677	-0,95236	→	9,52941	3,21916	3,80104	11,88677	-0,98751
12,56498	6,22260	3,48394	13,00202	2,60798	→	10,52982	6,22260	3,48394	13,00202	2,60798
10,74981	6,02141	4,09255	11,07596	0,87671	→	10,74981	6,02141	4,09255	11,07596	0,87671
16,12002	5,39248	3,76853	15,13561	5,73932	→	16,12002	7,39248	3,76853	11,13561	5,73932
16,95383	5,34009	3,79176	14,87492	7,64924	→	16,95383	5,34009	3,79176	14,87492	7,64924
8,81932	5,77975	4,83473	8,74892	-0,93662	→	8,81932	5,77975	4,83473	8,74892	-0,93662
6,67000	5,89300	5,14810	6,93595	-2,98582	→	6,67000	5,89288	5,14810	6,93595	-2,98582
18,03096	3,38688	0,00000	19,99194	9,98438	→	18,03096	3,64469	0,00000	20,00000	9,98438

Abbildung 5.16: Veränderung der rekombinierten Individuen zu mutierten Individuen

Im Beispiel werden durch die Mutation die 23 rekombinierten Individuen verändert. Die Änderungen an den Genen eines Individuums werden einzeln und statistisch vorgenommen. Nachfolgend an die Mutation wird der Wert jedes Gens auf den vorgegebenen Wertebereich beschränkt. In Abbildung 5.16 ist die Matrix mit den mutierten Individuen (rechts) dargestellt.

5.2.5 Bewertung

Bei der Bewertung der Individuen wird zu jedem neuen Nachkommen sein Zielfunktionswert berechnet. Beim evolutionären Testen bedeutet dies die Ausführung des Testfalls auf dem Testobjekt. Der berechnete Zielfunktionswert wird anschließend dem Individuum zugeordnet. Es werden also pro Generation genau so viele Testfälle auf dem Testobjekt ausgeführt, wie Nachkommen gebildet werden.

Im Beispiel des „Automatischen Parksystems“ mit einer Populationsgröße von 25 Individuen und einer Generationslücke von 0.9 sind dies 23 Nachkommen und somit 23 Testfälle pro Generation.

Jedes der Individuen stellt einen Testfall dar, der auf dem Testobjekt ausgeführt wird. Danach wird durch die Zielfunktion aus dem Testergebnis ein Zielfunktionswert berechnet, der dem jeweiligen Individuum zugeordnet ist. In Abbildung 5.17 sind die jeweils zugeordneten Zielfunktionswerte dargestellt. Die Individuen 1, 17, 21 und 23 wurden mit einem Strafwert von +100 bewertet, d.h. diese Individuen führen zu Testfällen, bei denen die Parksituation abgelehnt wurde.

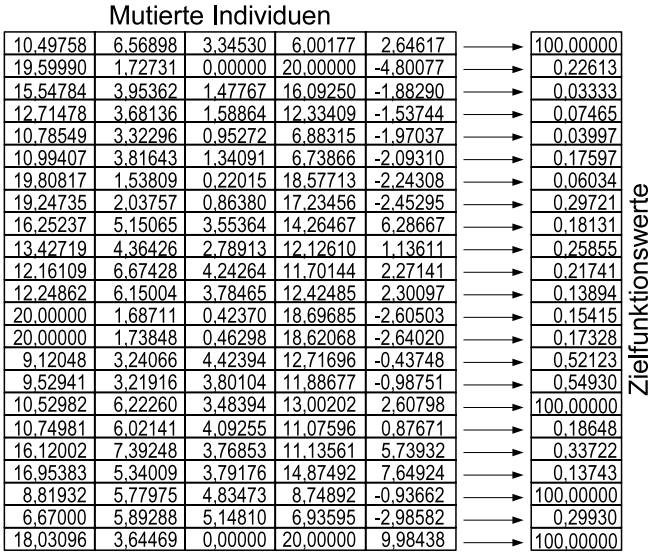


Abbildung 5.17: Berechnung der Zielfunktionswerte zu den neuen Individuen

5.2.6 Wiedereinfügen

Durch das Wiedereinfügen werden wahlweise die Eltern (selektierte Individuen) oder deren Nachkommen (mutierte Individuen) in die neue Population eingefügt. Es wird dabei mit

berücksichtigt, dass ein Nachkomme jeweils nur seinen zugeordneten Elternteil in der bestehenden Population ersetzen kann. Umgekehrt kann ein Individuum aus der bestehenden Population nur durch seinen ihm zugeordneten Nachkommen ersetzt werden.

In Abbildung 5.18 ist der Ablauf des Wiedereinfügens bei einem lokalen Modell schematisch dargestellt. Aufgrund einer gewählten Generationslücke (generation gap) kleiner als eins werden weniger Nachkommen erzeugt als sich Individuen in der Population befinden. Daher werden beim Schritt des Wiedereinfügens zunächst diejenigen Individuen aus der alten Population übernommen, die bei der Elternselektion nicht ausgewählt wurden und somit keine Nachkommen erzeugt haben.

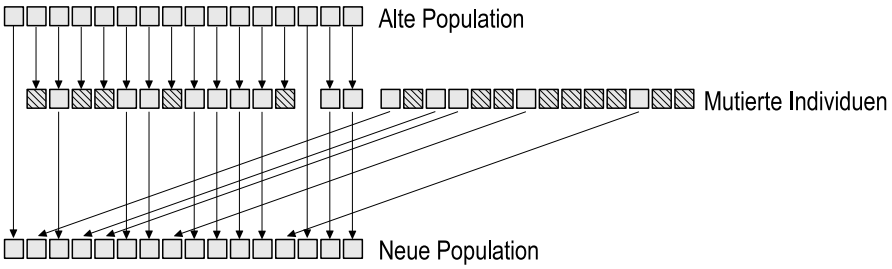


Abbildung 5.18: Wiedereinfügen der mutierten Nachkommen in die neue Population

Die bei der Elternselektion ausgewählten Individuen werden anschließend mit ihren zugeordneten Nachkommen anhand des jeweiligen Zielfunktionswerts verglichen. Dabei wird das Individuum, Elternteil oder Nachkomme, in die neue Population übernommen, das den besseren Zielfunktionswert besitzt (elitest reinsertion). Damit ist sichergestellt, dass das bisher beste Individuum in der Population erhalten bleibt. D.h. ein gefundener als sehr gut bewerteter Testfall kann nur durch einen noch besser bewerteten Testfall ersetzt werden. An dieser Stelle entsteht somit ein Selektionsdruck, der die Optimierung in Richtung besserer Zielfunktionswerte lenkt.

Dadurch dass die Nachkommen jeweils nur ihre Eltern ersetzen können, ist das Lokalitätsprinzip der Nachbarschaften und somit die Isolation durch Distanz gewahrt. Dadurch wird verhindert, dass die Optimierung zu schnell auf einen einzigen Punkt im Suchraum konvergiert und so die Gefahr besteht, in einem lokalen Minimum stecken zu bleiben.

Das Wiedereinfüge-Szenario aus dem Beispiel ist in Abbildung 5.19 dargestellt. Die blau hinterlegten Individuen wurden nicht zur Rekombination selektiert und werden daher direkt in die nächste Generation übernommen. Die gelb hinterlegten Individuen sind selektierte Individuen, die jeweils einen Nachkommen erzeugt haben, der jedoch einen schlechteren Zielfunktionswert hatte und somit nicht eingefügt wurde. Die grün hinterlegten Individuen sind durch Rekombination und Mutation neu entstanden und haben einen besseren Zielfunktionswert als ihr jeweiliges Elternteil. Die grau hinterlegten Individuen werden nicht in die nachfolgende Generation übernommen und scheiden damit aus.

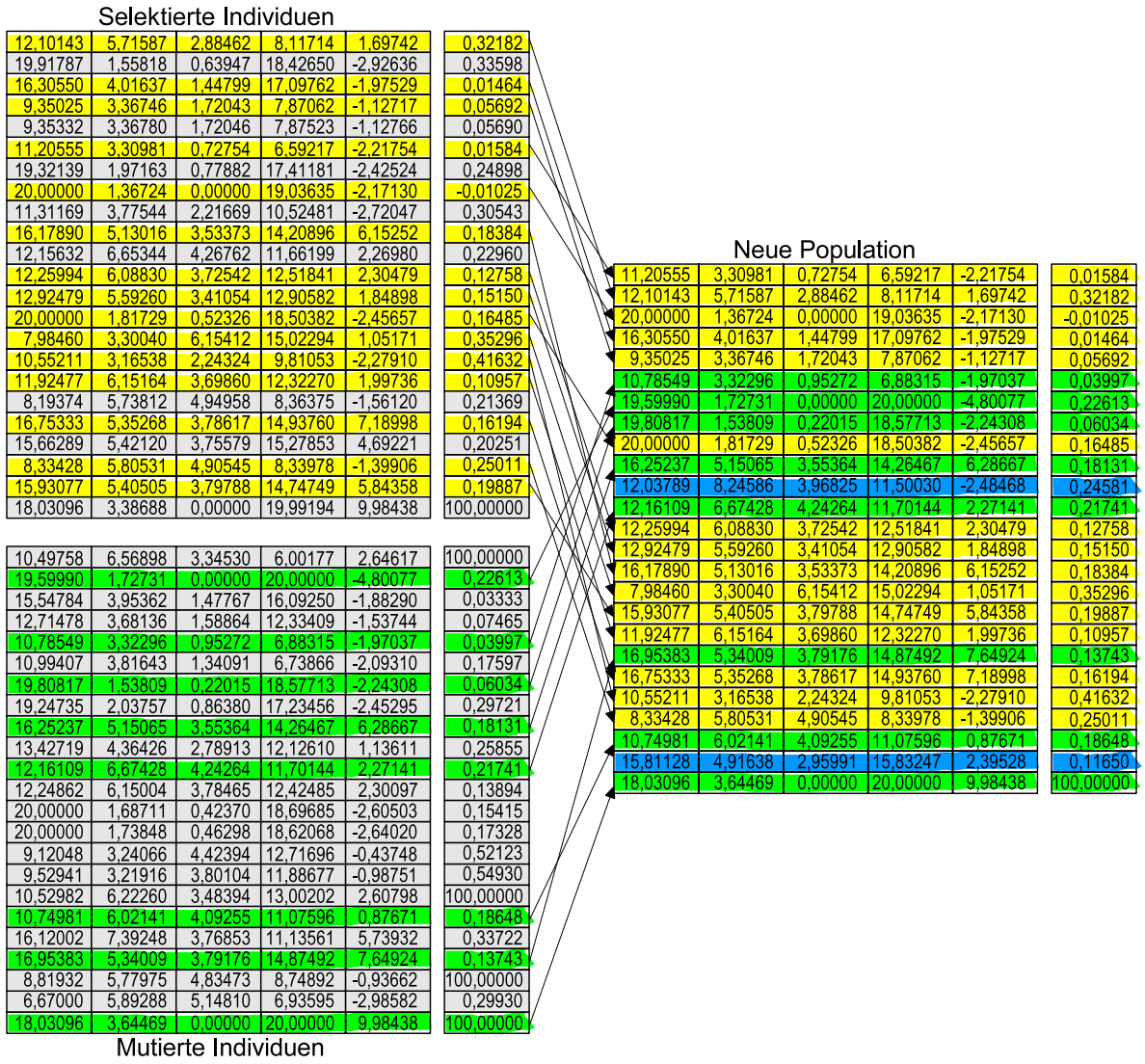


Abbildung 5.19: Einfügen ausgewählter Individuen in die neue Population

5.3 Der „Abstandsbasierte Bremsassistent“

Der „Abstandsbasierte Bremsassistent“ ist ein abstandsbasiertes Fahrerassistenzsystem, das die aktive Sicherheit eines Kraftfahrzeugs erhöhen und somit zur Reduzierung des Unfallrisikos beitragen soll. Die bereits seit Jahren bewährte Funktion des Bremsassistenten wurde durch den Einbau einer umgebungserfassenden Sensorik weiter verbessert.

Die Testumgebung für den evolutionären Funktionstest des „Abstandsbasierten Bremsassistenten“ basiert auf einer Software-in-Loop Testumgebung[6], die bereits für manuelle Tests aufgebaut wurde und mit der auch manuelle Tests für die serienreifen Software-Komponenten durchgeführt werden. Ein evolutionärer Funktionstest mit dieser Anwendungsfunktion wurde bereits von Bühler und Wegener in [95] vorgestellt.

5.3.1 Anwendung

Eine Fahrerassistenzfunktion, die den Fahrer in kritischen Situationen beim Bremsen unterstützt, nennt man Bremsassistent. Eine Verbesserung des Bremsverhaltens mit dem Ziel, den Bremsweg zu verkürzen, versuchte man lange Zeit durch die Entwicklung und Verbesserung der Bremsanlage zu erreichen. Dabei wurden beachtliche Fortschritte erzielt [111]. Die Intention beim Entwurf des Bremsassistenten geht in eine andere Richtung. Die Idee beim Bremsassistenten ist es, die Zeit vor der eigentlichen Bremsung zu minimieren. Diese Optimierungen zielen speziell auf die Zeitspanne von der Fahrerreaktion bis zum Erreichen der (maximalen) Fahrzeugverzögerung ab. Eine zeitliche Minimierung ist nicht in allen Phasen des Bremsvorgangs möglich. Die Zeit der Wahrnehmung und Entscheidung durch den Fahrer kann nicht verkürzt werden, wenn man keinen autonomen Bremseingriff ausführen möchte.

Bremsassistent der ersten Generation

Eine experimentelle Untersuchung mit Probanden, dargestellt in [1], hat gezeigt, dass der durchschnittliche Fahrer in vielen Fällen nicht schnell genug den notwendigen Bremsdruck aufbaut, wenn er plötzlich mit einer kritischen Situation konfrontiert wird. Dadurch gehen wertvolle Meter an Bremsweg verloren. Um den Fahrer in dieser Situation zu unterstützen, überwacht ein Bremsassistent der ersten Generation permanent die Geschwindigkeit des Bremspedals. Innerhalb eines Sekundenbruchteils wird entschieden, ob der Fahrer eine Gefahrenbremsung ausführen möchte. Wird die Absicht einer Gefahrenbremsung erkannt, initiiert der Bremsassistent sofort den vollen Bremsdruck. Der notwendige Bremsdruck wird damit schneller aufgebaut und dadurch kann der Bremsweg verkürzt werden.

[6] Bei einer Software-in-Loop (SiL) Testumgebung läuft die zu testende Software-Komponente zunächst auf einem PC statt im Steuergerät. Alle Teile, mit denen die Software-Komponente interagiert, einschließlich Fahrzeug, Fahrzeugumgebung und Fahrer werden hier simuliert.

Bremsassistent der zweiten Generation

Ein Bremsassistent der zweiten Generation erweitert dieses Konzept und verwendet zusätzlich zum Bremspedal die Information aus der umgebungserfassenden Sensorik. Bei einem Test mit 100 Probanden im Fahrsimulator verringerte sich die Unfallquote von 44 Prozent ohne BAS PLUS auf 11 Prozent mit BAS PLUS [105]. Wie in Abbildung 5.20 dargestellt, werden dabei die Informationen über den Netto-Relativ-Abstand und die Relativ-Geschwindigkeit zum vorausfahrenden Fahrzeug zur Bewertung der aktuellen Situation mit einbezogen.

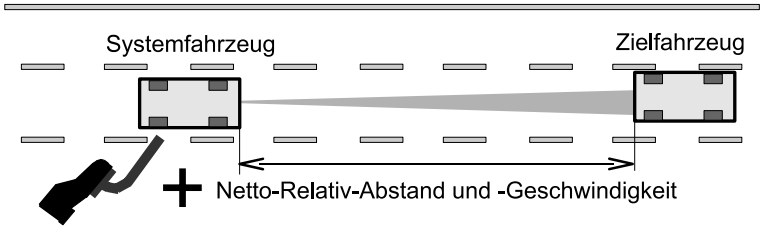


Abbildung 5.20: Prinzip Bremsassistent der zweiten Generation

Der „Abstandsbasierte Bremsassistent“ verwendet dabei die Information über das Bremspedal zusammen mit der Information über die Relativ-Geschwindigkeit und den Netto-Relativ-Abstand, um zu entscheiden, ob der Fahrer eine Gefahrenbremsung beabsichtigt. Falls der „Abstandsbasierte Bremsassistent“ eine Gefahrensituation erkennt, verstärkt er, wenn notwendig, die Fahrerbremsung. Dazu berechnet er die optimale Verzögerung des Fahrzeugs, um einen Unfall zu vermeiden. Der „Abstandsbasierte Bremsassistent“ stellt dann zusätzlich zum Fahrerbremsmoment ein additives Bremsmoment, damit die optimale Verzögerung des Fahrzeugs erreicht werden kann. Die Abhängigkeit dieser Anwendungsfunktion von den Aktionen des Fahrers und vom Verhalten der Fahrzeuge in der Fahrzeugumgebung macht einen „Abstandsbasierten Bremsassistenten“ komplexer als einen Bremsassistenten der ersten Generation.

Das durch den „Abstandsbasierten Bremsassistenten“ additiv angeforderte Bremsmoment stellt eine Gratwanderung dar. Es darf einerseits nicht zu hoch sein, da unnötig starkes Bremsen vom Fahrer nicht akzeptiert würde und außerdem die Gefahr einer Kollision mit dem Hintermann hervorrufen kann. Auf der anderen Seite sollte die Bremsunterstützung nicht zu gering ausfallen, da sonst keine effiziente Vermeidung des Auffahrens mehr möglich ist. Für den funktionalen Test der Anwendung interessieren insbesondere auch die Grenzwerte, also z.B. relativ unkritische Situationen, bei denen die Bremsunterstützung zu stark ist, bzw. relativ kritische Situationen, bei denen die Bremsunterstützung zu schwach oder gar nicht vorhanden ist.

5.3.2 Codierung

Durch die Codierung der Testdaten werden die Gene eines Individuums in Testdaten transformiert mit denen ein Szenario simuliert werden kann. Für den Test des „Abstandsbasierten Bremsassistenten“ werden nachfolgend verschiedene Modelle und deren Codierung vorgestellt. Die Modelle unterscheiden sich in der Vielfalt an möglichen Szenarien, die sie generieren können, und damit auch in der Anzahl von Genen, die zur Codierung benötigt werden. Für jedes der vorgestellten Modelle wird die zugehörige Codierung eines Individuums in Form einer Tabelle dargestellt. Jedem der verwendeten Gene wird eine der in Kapitel 4.2.2 eingeführten Gen-Klassen als Typ zugeordnet. Ebenso wird die Bedeutung des Gens bei der Codierung erläutert sowie sein Wertebereich und seine Maßeinheit.

Codierung einfacher Auffahr-Szenarien

In Abbildung 5.21 ist ein Auffahr-Szenario dargestellt. Das hier beschriebene (einfache) Modell benötigt fünf physikalische Größen zur Definition eines Fahrmanövers, nämlich Ort und Geschwindigkeit pro Fahrzeug und einen Parameter s_{System}, der einen Abstand bezeichnet und die Bedeutung hat, dass wenn der tatsächliche Netto-Relativ-Abstand zwischen beiden Fahrzeugen gleich dem Wert dieses Parameters ist, dass dann die Bremsung des Systemfahrzeugs ausgelöst wird. Der Ort $x(t)$ eines Fahrzeugs zum Zeitpunkt t ist bei konstanter Geschwindigkeit v des Fahrzeugs gegeben durch: $x(t) = x_0 + v \cdot (t - t_0)$. In anderen Worten, zum Zeitpunkt $t = t_0$ befindet sich das Fahrzeug an der Stelle x_0. In die Modellrechnung gehen die Orte der beiden Fahrzeuge zum Zeitpunkt $t = t_0$ als konstante Parameter ein. Die von außen vorgebbaren Modellparameter sind dann die Geschwindigkeiten der beiden Fahrzeuge v_{System} und v_{Ziel} zum Zeitpunkt $t = t_0$ und der Parameter s_{System}. Hierbei stellen v_{System} und v_{Ziel} absolute Geschwindigkeiten und keine Relativgeschwindigkeiten dar. Negative Werte von v_{Ziel} bedeuten, dass das Zielfahrzeug kein Führungsfahrzeug darstellt, sondern dass ein entgegenkommendes Fahrzeug auf der eigenen Spur entgegen kommt.

Die Geschwindigkeit des Systemfahrzeugs muss stets höher gewählt werden als die Geschwindigkeit des Zielfahrzeugs, so dass das Systemfahrzeug von hinten auf das Zielfahrzeug auffährt. Beide Fahrzeuge bewegen sich zu Beginn des Szenarios mit ihrer vorgegebenen, konstanten Geschwindigkeit so lange, bis der dritte Parameter s_{System} erreicht wird. In diesem Moment zündet die Datenbedingung und löst ein Ereignis aus. Als Folge dieses Ereignisses wird der Bremsvorgang des Systemfahrzeugs ausgelöst. Ab diesem Zeitpunkt bewegt sich dann das Fahrzeug mit einer Bremsverzögerung und damit nicht mehr mit konstanter Geschwindigkeit. Dieses Modell arbeitet mit einem fest hinterlegten Bremspedalverlauf des Fahrers.

Die Codierung des Modells für einfache Auffahr-Szenarien benötigt drei Gene in einem Individuum zur Definition eines Testfalls. Diese Codierung spannt also einen 3-dimensionalen Suchraum auf. Die Struktur der Gene für die Codierung ist in der nachfolgenden Tabelle 5.3 dargestellt.

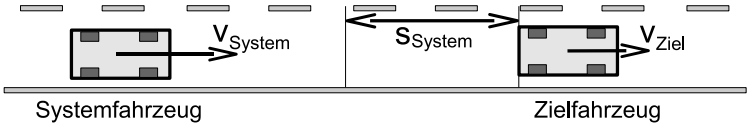


Abbildung 5.21: Modell eines einfachen Auffahr-Szenarios

Nr.	Bedeutung	Typ	Wertebereich	Maßeinheit
Gen 1	s_{System}	III	1 bis 100	[m]
Gen 2	$v_{relativ}$	I	-120 bis -1	$[\frac{m}{s}]$
Gen 3	v_{Ziel}	I	-50 bis 50	$[\frac{m}{s}]$

Tabelle 5.3: Gene zur Codierung eines einfachen Auffahr-Szenarios

Das Gen 1 hat dabei die Bedeutung des Parameters s_{System} für die Datenbedingung des Ereignisses „Netto-Relativ-Abstand unterschritten“. Es ist ein Gen vom Typ III, da es eine Datenbedingung für ein Ereignis codiert. Die Werte des Gens haben die Maßeinheit $[m]$, der Wertebereich ist zwischen 1 m und 100 m festgelegt. Damit können also Szenarien codiert werden, bei denen das Systemfahrzeug bei einem Netto-Relativ-Abstand zwischen 1 m und 100 m beginnt zu bremsen.

Das Gen 2 codiert die Relativ-Geschwindigkeit zwischen dem Zielfahrzeug und dem Systemfahrzeug zu Beginn der Simulation und legt damit indirekt die Größe des Parameters v_{System} fest. Es ist ein Gen vom Typ I, da es eine Konstante für den Initialzustand des Szenarios codiert.

Das Gen 3 codiert die Geschwindigkeit des Zielfahrzeugs während des Szenarios. Dieses Gen codiert ebenfalls eine Konstante für den Initialzustand des Szenarios und ist damit vom Typ I. Das Zielfahrzeug fährt mit dieser konstanten Geschwindigkeit während des gesamten Szenarios. Der Wertebereich von Gen 3 ist zwischen -50 $\frac{m}{s}$ und $+50$ $\frac{m}{s}$ festgelegt. Dies bedeutet, dass das Zielfahrzeug in die gleiche Richtung fahren kann wie das Systemfahrzeug und auch in die entgegengesetzte Richtung.

Durch diese Art der Codierung lässt sich erreichen, dass die Geschwindigkeit des Systemfahrzeugs stets höher gewählt wird als die Geschwindigkeit des Zielfahrzeugs, indem man den Wertebereich von Gen 2 auf negative Werte zwischen -120 $\frac{m}{s}$ bis -1 $\frac{m}{s}$ festlegt. Die für die Simulation des Szenarios benötigte initiale Geschwindigkeit des Systemfahrzeugs v_{System} berechnet sich aus den codierten Größen nach den Gleichungen 5.8 und 5.9.

$$v_x = v_{Ziel} - v_{relativ} \tag{5.8}$$

$$v_{System} = \begin{cases} 0\frac{m}{s} & \text{falls } v_x < 0\frac{m}{s} \\ v_x & \text{falls } 0 < v_x < 70\frac{m}{s} \\ 70\frac{m}{s} & \text{falls } v_x > 70\frac{m}{s} \end{cases} \tag{5.9}$$

Codierung zur Beschleunigung bzw. Verzögerung des Zielfahrzeugs

Das nachfolgend betrachtete Modell erweitert das einfache Modell, bei dem nur das Systemfahrzeug bremst. Die Erweiterung besteht darin, dass nun auch das Zielfahrzeug während des Szenarios eine beschleunigte oder verzögerte Bewegung ausführen kann. Ein Szenario beginnt wie beim einfachen Auffahr-Szenario mit konstanten initialen Geschwindigkeiten des System- und Zielfahrzeugs, wobei die initiale Geschwindigkeit des Zielfahrzeugs nun mit $v_{Ziel,1}$ bezeichnet wird. Die Aktionen beider Fahrzeuge werden ereignisgesteuert ausgelöst, d.h. das Bremsen oder Beschleunigen der beiden Fahrzeuge kann gleichzeitig beginnen oder aber auch zeitlich versetzt. Das Modell arbeitet mit einer festgelegten Kennlinie für den Bremspedalverlauf des Fahrers.

Während in realen Verkehrssituationen das Zielfahrzeug beliebig nach Wunsch seines Fahrers beschleunigt oder verzögert werden kann, wird hier in der Simulation der Testumgebung das Zielfahrzeug beschleunigt oder verzögert, wenn der Netto-Relativ-Abstand zwischen Ziel- und Systemfahrzeug den Parameter s_{Ziel} unterschreitet. Wird der Wert dieses Parameters erreicht, so zündet die Datenbedingung, ein Ereignis wird ausgelöst und das Zielfahrzeug beschleunigt oder verzögert mit einer Beschleunigung von a_{Ziel}. Dabei berechnet sich der Wert für diese Beschleunigung a_{Ziel} nach Gleichung 5.10 zu:

$$a_{Ziel} = \frac{-v_{Ziel,2} + v_{Ziel,1}}{t_{Ziel}} \tag{5.10}$$

Dabei ist t_{Ziel} die vorgegebene Zeit zur Änderung der Geschwindigkeit des Zielfahrzeugs von $v_{Ziel,1}$ auf $v_{Ziel,2}$. Beim Erreichen der Geschwindigkeit $v_{Ziel,2}$ sieht das Modell vor, dass das Zielfahrzeug wieder mit konstanter Geschwindigkeit weiter fährt, wie in Abbildung 5.22 dargestellt.

Die Codierung für dieses Modell benötigt 6 Gene in einem Individuum zur Definition eines Szenarios. Die Struktur der Gene für die Codierung ist in der nachfolgenden Tabelle 5.4 dargestellt. Die Gene 1 bis 3 entsprechen dabei den Genen der Codierung einfacher Auffahr-Szenarien.

Das Gen 4 codiert den Parameter s_{Ziel}, d.h. den Abstand zwischen den beiden Fahrzeugen, welcher die Datenbedingung für das Ereignis zur Auslösung der Verzögerung bzw. Beschleunigung des Zielfahrzeugs darstellt. Das Gen 4 ist vom Typ III, da es eine Datenbedingung für ein Ereignis codiert.

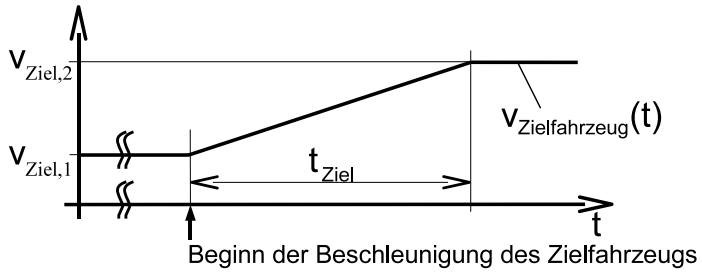


Abbildung 5.22: Beschleunigte Bewegung des Zielfahrzeugs

Nr.	Bedeutung	Typ	Wertebereich	Maßeinheit
Gen 1	s_{System}	III	1 bis 100	$[m]$
Gen 2	$v_{relativ}$	I	-120 bis -1	$[\frac{m}{s}]$
Gen 3	$v_{Ziel,1}$	I	-50 bis 50	$[\frac{m}{s}]$
Gen 4	s_{Ziel}	III	0 bis 200	$[m]$
Gen 5	t_{Ziel}	IV	0.02 bis 20	$[s]$
Gen 6	$v_{Ziel,2}$	IV	0 bis 100	$[\frac{m}{s}]$

Tabelle 5.4: Gene zur Codierung eines Auffahr-Szenarios mit Beschleunigung bzw. Verzögerung des Zielfahrzeugs

Das Gen 5 codiert die Zeitspanne t_{Ziel} für die Beschleunigung bzw. Verzögerung des Zielfahrzeugs. Dieses Gen ist vom Typ IV da es eine Konstante codiert, die das Verhalten einer Aktion definiert. Für Gen 5 ist ein Wertebereich von 0.02 s bis 20 s festgelegt.[7]

Das Gen 6 codiert die Endgeschwindigkeit $v_{Ziel,2}$ des Zielfahrzeugs. Da dieses Gen ebenfalls das Verhalten einer Aktion definiert, ist es ein Gen vom Typ IV. Das bedeutet, dass das Zielfahrzeug seine Geschwindigkeit von $v_{Ziel,1}$ auf $v_{Ziel,2}$ innerhalb der 0.02 s bis 20 s ändern kann, wenn das Ereignis für das Zielfahrzeug eintritt. Dabei wird auch der Fall modelliert, dass das Zielfahrzeug mit dieser Art der Codierung eine Verzögerung von mehr als -12 $\frac{m}{s}$ erreichen kann. Dies ist zwar mit einer konventionellen Bremse im Fahrzeug in der Regel nicht möglich, jedoch kann in der Realität bei Kollisionen sehr wohl ein hoher Wert für a_{Ziel} auftreten, z.B. wenn das vorausfahrende Zielfahrzeug mit seinem Vordermann kollidiert.

Codierung zur Variation des Bremspedalverlaufs

Das hier betrachtete Modell erweitert das einfache Modell zur Codierung von Auffahr-Szenarien, indem es keine feste Kennlinie für die Bremspedalkurve des Fahrers verwendet, sondern diese Kennlinie variiert. Die Kennlinie wird nach dem Unterschreiten des Bremsabstands abgefragt und sie definiert, wie der Fahrer auf das Bremspedal tritt. Die Codierung

[7] Der Wert 0.02 s entspricht der Zykluszeit für das eingebettete System.

der Kennlinie für die Bremspedalkurve wird in optimierter Form mit Hilfe von 5 Parametern definiert.

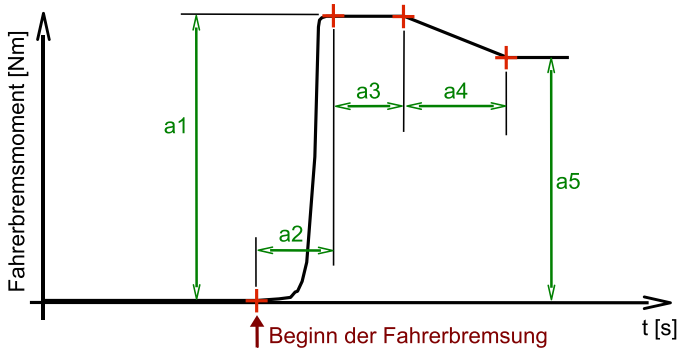


Abbildung 5.23: Bremspedalkurve des Fahrers mit den Modellparametern a_1 bis a_5

In Abbildung 5.23 ist die Bedeutung der Parameter dargestellt, mit denen die Stützpunkte für die Kurve eines Fahrerbremsmoments definiert werden. Die Aktion der Fahrerbremsung wird im Szenario durch ein Ereignis ausgelöst, wenn der Netto-Relativ-Abstand zwischen System- und Zielfahrzeug die Datenbedingung mit dem Parameter s_{System} auslöst.[8]

Die Codierung für dieses Modell benötigt insgesamt 8 Gene. Dabei ist zu beachten, dass die ersten drei Gene wieder den Genen der Codierung einfacher Auffahr-Szenarien entsprechen. Die Struktur der Gene für die Codierung ist in der nachfolgenden Tabelle 5.5 dargestellt. Die Gene 4 bis 8 sind alles Gene vom Typ V, da sie Kennlinienparameter für eine Aktion codieren.

Gen Nr.	Bedeutung	Gen-Typ	Wertebereich	
Gen 1	s_{System}	III	1 bis 100	[m]
Gen 2	$v_{relativ}$	I	-120 bis -1	$[\frac{m}{s}]$
Gen 3	v_{Ziel}	I	-50 bis 50	$[\frac{m}{s}]$
Gen 4	a_1	V	0 bis 12000	[Nm]
Gen 5	a_2	V	0 bis 20	[s]
Gen 6	a_3	V	0 bis 20	[s]
Gen 7	a_4	V	0 bis 20	[s]
Gen 8	a_5	V	0 bis 12000	[Nm]

Tabelle 5.5: Gene zur Codierung eines Auffahr-Szenarios mit Variation des Bremspedalverlaufs

Das Gen 4 codiert den Parameter a_1, also die Höhe des Fahrerbremsmoments, das während der ersten Bremspedalbetätigung erreicht wird. Der Wertebereich des Gens ist auf 0 Nm

[8] Der Parameter s_{System} wurde bereits im Kapitel 5.3.2 „Codierung einfacher Auffahr-Szenarien“ beschrieben.

bis 12000 Nm festgelegt. Dies bedeutet, bei der ersten Bremspedalbetätigung kann jeder Wert zwischen „keine Bremsung“ und „Vollbremsung“ auftreten.

Das Gen 5 codiert den Parameter a_2, der die Zeitspanne für das Erreichen der maximalen Bremspedalbetätigung definiert. Der Wertebereich hier ist auf 0.02 s bis 20 s festgelegt. D.h. der Fahrer kann sprunghaft in die Bremse treten oder sehr langsam den Maximalwert der ersten Betätigung erreichen.

Die Gene 6 und 7 codieren die Zeitspannen a_3 und a_4. Durch die Zeitspanne a_3 wird festgelegt, wie lange der durch a_1 definierte Bremspedalwert anliegen soll. Mit dem Parameter a_4 wird die nachfolgende Zeitspanne festgelegt, wie lange der Übergang zum letztendlichen Bremspedalwert dauern soll. Beide Gene besitzen einen Wertebereich von 0.02 s bis 20 s.

Das Gen 8 codiert schließlich den endgültigen Bremspedalwert, auf den über eine Rampe vom Wert a_1 gefahren wird. Der Wertebereich von Gen 8 ist wie der Wertebereich von Gen 4 auf 0 Nm bis 12000 Nm festgelegt.

Codierung des verwendeten Modells für die experimentellen Untersuchungen

Für die experimentellen Untersuchungen des „Abstandsbasierten Bremsassistenten“ wurde ein Modell verwendet, das die Aspekte der drei vorgestellten Modelle kombiniert. Damit ist es möglich, Auffahr-Szenarien zu definieren, mit denen eine abstandsbasierte Auslösung der Bremsung des Systemfahrzeugs, eine Variation des Bremspedalverlaufs des Fahrers und eine Verzögerung des Zielfahrzeugs möglich ist. Für die Codierung dieses Modells sind insgesamt 11 Gene notwendig. Die Struktur der Gene für die Codierung ist in der nachfolgenden Tabelle 5.6 dargestellt.

Nr.	Bedeutung	Typ	Wertebereich	Maßeinheit
Gen 1	s_{System}	III	1 bis 100	$[m]$
Gen 2	$v_{relativ}$	I	-120 bis -1	$[\frac{m}{s}]$
Gen 3	$v_{Ziel,1}$	I	-50 bis 50	$[\frac{m}{s}]$
Gen 4	a_1	V	0 bis 12000	$[Nm]$
Gen 5	a_2	V	0 bis 20	$[s]$
Gen 6	a_3	V	0 bis 20	$[s]$
Gen 7	a_4	V	0 bis 20	$[s]$
Gen 8	a_5	V	0 bis 12000	$[Nm]$
Gen 9	s_{Ziel}	III	0 bis 200	$[m]$
Gen 10	t_{Ziel}	IV	0.02 bis 20	$[s]$
Gen 11	$v_{Ziel,2}$	IV	0 bis 100	$[\frac{m}{s}]$

Tabelle 5.6: Gene der experimentellen Untersuchungen

Die erste Gensequenz mit den drei Genen Gen 1 bis Gen 3 codiert die Parameter v_{System}, $v_{Ziel,1}$ und s_{System}. Die zweite Gensequenz mit den fünf Genen Gen 4 bis Gen 8 codiert

die Parameter a_1 bis a_5 für die Kennlinie der Bremspedalvorgabe des Fahrers. Die dritte Gensequenz mit den drei Genen Gen 9 bis Gen 11 codiert das Verhalten des Zielfahrzeugs während des Szenarios. Die Bedeutungen und die Wertebereiche der Gene entsprechen der in den zuvor dargestellten Modellen verwendeten Codierung.

5.3.3 Zielfunktion

Die Zielfunktion bewertet ein Brems-Szenario, das vom „Abstandsbasierten Bremsassistenten“ in der Simulationsumgebung durchgeführt wurde. Nachfolgend wird eine Zielfunktion für die Anwendungsfunktion „Abstandsbasierter Bremsassistent“ nach der in Kapitel 4.3 dargestellten Vorgehensweise entworfen.

Ergebnisklassen

Entsprechend der Vorgehensweise wird zunächst eine grundlegende Anforderung an die Anwendungsfunktion eines „Abstandsbasierten Bremsassistenten“ ausgewählt. Für eine solche Anwendungsfunktion gilt u.a. die folgende Anforderung:

Anforderung: Wenn eine Situation kritisch ist, der „Abstandsbasierte Bremsassistent“ aktiv und der Fahrer in dieser Situation ungenügend bremst, dann und nur dann soll durch den „Abstandsbasierten Bremsassistenten“ eine hohe Bremsunterstützung erfolgen.

Die Anwendungsfunktion soll den Fahrer also in kritischen Situationen möglichst gut unterstützen, es darf jedoch nicht vorkommen, dass eine unnötige Bremsunterstützung[9] in unkritischen Situationen erfolgt.

Es lassen sich nun mögliche Brems-Szenarien in verschiedene Ergebnisklassen einordnen. In Abbildung 5.24 sind die betrachteten Ergebnisklassen A, B, C, D und E dargestellt, die man durch Analyse der gewählten Anforderung bilden kann. Zunächst kann man unterscheiden zwischen Szenarien, bei denen sich die Anwendungsfunktion aktiviert hat und Szenarien, die ohne Aktivierung der Anwendungsfunktion abgelaufen sind. Die Ergebnisklasse A enthält alle Szenarien ohne Aktivierung. Die Vereinigungsmenge aus $B \cup C \cup D \cup E$ enthält die Szenarien mit Aktivierung.

Die Szenarien mit Aktivierung können weiter unterteilt werden in Szenarien mit unkritischer Situation, dargestellt durch $B \cup C$, und Szenarien mit kritischer Situation, dargestellt durch $D \cup E$. Letztlich können in Ergebnisklasse B alle Szenarien mit hoher Bremsunterstützung in kritischer Situation eingeordnet werden. Der Ergebnisklasse C können die Szenarien mit geringer Bremsunterstützung in kritischer Situation zugeordnet werden. Die Ergebnisklasse D soll Szenarien mit hoher Bremsunterstützung in unkritischer Situation enthalten und Ergebnisklasse E soll Szenarien mit geringer Bremsunterstützung in unkritischer Situation umfassen.

[9] Eine hohe Bremsunterstützung bedeutet, dass der „Abstandsbasierte Bremsassistent“ ein hohes zusätzliches Bremsmoment stellt. Umgekehrt bedeutet ein geringes zusätzliches Bremsmoment eine geringe Bremsunterstützung.

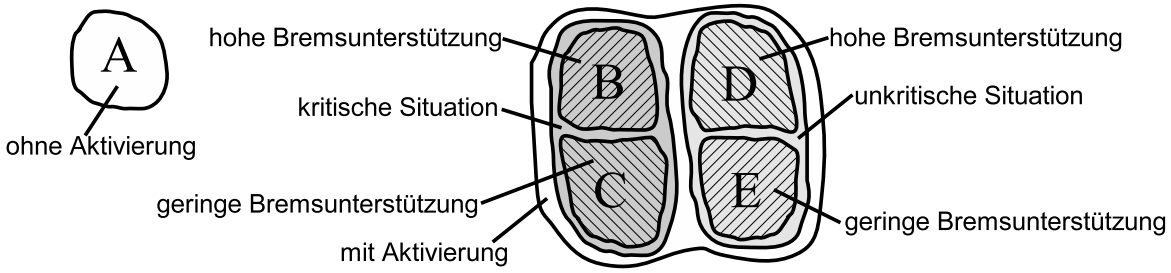


Abbildung 5.24: Betrachtete Ergebnisklassen der Anforderung „hohe Bremsunterstützung“

Optimierungsziel

Aus den modellierten und in Abbildung 5.24 dargestellten Ergebnisklassen lassen sich zwei unterschiedliche Optimierungsziele formulieren:

1. „Hohe Bremsunterstützung bei unkritischer Situation“
2. „Niedrige Bremsunterstützung bei kritischer Situation“

Für jedes der beiden Optimierungsziele wird eine eigene Zielfunktion benötigt. Nachfolgend wird ausschließlich das erste Optimierungsziel entwickelt. Das Optimierungsziel „Niedrige Bremsunterstützung bei kritischer Situation“ wird nicht weiter betrachtet.

Die Ergebnisklassen für das betrachtete Optimierungsziel „Hohe Bremsunterstützung bei unkritischer Situation“ sind in Abbildung 5.25 dargestellt. Das Ziel ist es Szenarien zu finden, bei denen in einer unkritischen Situation eine hohe Bremsunterstützung erfolgt. Diese Szenarien sollen durch die Zielfunktion der Ergebnisklasse D zugeordnet werden.

In die Ergebnisklasse $\overline{D}$ werden alle anderen Szenarien eingeordnet, die im Sinne des Optimierungsziels nicht interessieren. Dazu gehören alle Szenarien, bei denen die Anwendungsfunktion überhaupt nicht aktiv war. Es zählen ebenso alle Szenarien dazu, bei denen sich die Anwendungsfunktion zwar aktiviert hat, jedoch keine oder nur eine geringe Bremsunterstützung geleistet hat. Darüber hinaus werden hier Szenarien zugeordnet, bei denen in einer kritischen Situation eine hohe Bremsunterstützung geleistet wurde.

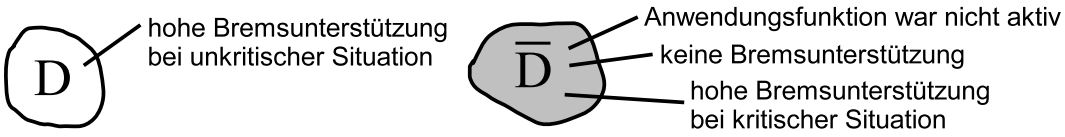


Abbildung 5.25: Ergebnisklassen des Optimierungsziels „Hohe Bremsunterstützung bei unkritischer Situation“

Klassifikationsmerkmale

Es sollen nun charakteristische Merkmale gefunden werden, mit denen ein vorliegendes Szenario klassifiziert und somit der Ergebnisklasse D oder $\overline{D}$ zugeordnet werden kann. Das interessante bei einem Szenario der Ergebnisklasse D sind die Zeitpunkte, bei denen die Bremsunterstützung hoch ist und *gleichzeitig* die Situation unkritisch ist.

Im ersten Schritt sollen nun zunächst Maße für die Bremsunterstützung und die Kritikalität einer Situation gefunden werden. Als ein Maß für die Bremsunterstützung kann das vom „Abstandsbasierten Bremsassistent“ zusätzlich angeforderte Bremsmoment $M_{Br,Add}$ betrachtet werden. Dies berechnet sich mit:

$$M_{Br,Add}(t) = M_{Br,Fahrzeug}(t) - M_{Br,Fahrer}(t) \tag{5.11}$$

wobei $M_{Br,Fahrzeug}$ das vom Fahrzeug gestellte Bremsmoment und $M_{Br,Fahrer}$ das vom Fahrer angeforderte Bremsmoment darstellt. Je höher der Wert von $M_{Br,Add}$ ist, desto höher ist die Unterstützung des Fahrers beim Bremsen. Je niedriger der Wert von $M_{Br,Add}$ ist, desto niedriger ist diese beim Bremsen.

Als ein Maß für die Kritikalität einer Situation kann man die Zeit bis zur Kollision T_{tc} betrachten[10]. Je höher der Wert der T_{tc} ist, desto unkritischer ist die Situation, da ja noch viel Zeit bis zur Kollision vergeht. Je niedriger der Wert der T_{tc} ist, desto kritischer ist die Situation, da die Kollision kurz bevor steht. Die Zeit bis zur Kollision kann für jeden Zeitpunkt t des Szenarios berechnet werden. Das Zeitintervall $T_{tc}(t)$ wird berechnet aus dem Quotient zwischen dem Netto-Abstand $x_{Ziel,System}$ zu dem vorausfahrenden Fahrzeug und der Relativgeschwindigkeit $v_{x,Ziel,System}$. Damit berechnet sich die $T_{tc}(t)$ mit:

$$T_{tc}(t) = \frac{x_{Ziel,System}(t)}{-min(v_{x,Ziel,System}(t), v_{rel,max})} \tag{5.12}$$

$$x_{Ziel,System}(t) = x_{Ziel}(t) - x_{System}(t) \tag{5.13}$$

$$v_{x,Ziel,System}(t) = v_{x,Ziel}(t) - v_{x,System}(t) \tag{5.14}$$

Dabei wird von einer maximalen Relativgeschwindigkeit von $v_{rel,max} = -1\frac{m}{s}$ ausgegangen, um negative Werte von T_{tc} oder eine Division durch Null zu vermeiden. Außerdem wird damit erreicht, dass bei einer Relativgeschwindigkeit nahe Null und gleichzeitig einem sehr geringen Netto-Abstand trotzdem ein kleiner Wert für T_{tc} berechnet wird. Dies liefert ein besseres Maß für die Kritikalität in dieser Situation.

Damit die beiden identifizierten Größen ein charakteristisches Merkmal bilden können, müssen sie nun noch in geeigneter Weise verknüpft werden. Um einen Zusammenhang zu erkennen, kann die Beziehung zwischen dem Signalverlauf der T_{tc} und dem Signalverlauf von

[10] aus dem Engl. Time-To-Collision

$M_{Br,Add}$ mit Hilfe eines Streudiagramms graphisch dargestellt werden. Dabei wird jeder Zeitpunkt des Szenarios als einzelner Punkt im Diagramm dargestellt. Die Position eines Punkts im Diagramm ergibt sich aus dem Wert der T_{tc} und dem Wert von $M_{Br,Add}$ zum selben Zeitpunkt im Szenario.

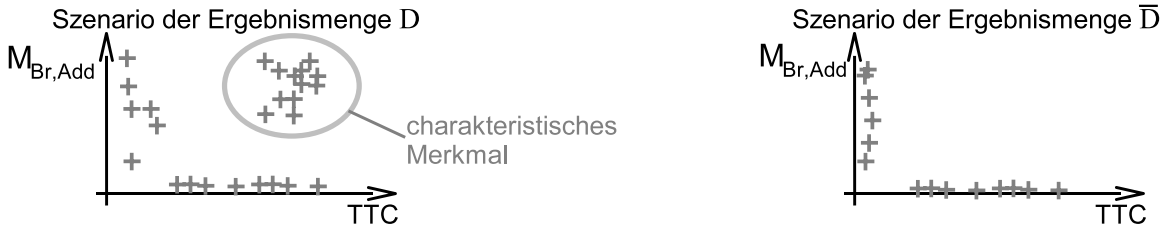


Abbildung 5.26: Streudiagramme von Szenarien der Ergebnisklassen D und $\overline{D}$.

In Abbildung 5.26 ist jeweils ein Brems-Szenario aus der Ergebnisklasse D und der Ergebnisklasse $\overline{D}$ als Streudiagramm dargestellt. Das linke Diagramm zeigt ein Szenario aus der Ergebnisklasse D. Im rechten Diagramm ist ein Szenario aus der Ergebnisklasse $\overline{D}$ dargestellt. Als markanter Unterschied zwischen den beiden Szenarien ist eine Menge an Punkten im linken Diagramm rechts oben zu erkennen. Die Position dieser Punkte bedeutet, dass es im jeweiligen Szenario Zeitpunkte gab, bei denen ein hoher Wert für die T_{tc} und gleichzeitig ein hoher Wert für $M_{Br,Add}$ vorlag. Dies bedeutet, dass zu diesem Zeitpunkt in einer unkritischen Situation ein hohes zusätzliches Bremsmoment angefordert wurde. Je mehr Punkte in diesem Bereich liegen, desto mehr interessante Zeitpunkte gab es während des Szenarios.

Die im linken Diagramm eingekreist dargestellten Punkte stellen also das identifizierte charakteristische Merkmal dar, mit dem sich ein Szenario der Ergebnisklasse D zuordnen lässt. In anderen Worten, das charakteristische Merkmal für ein Szenario der Ergebnisklasse D sind viele Zeitpunkte im Szenario, die im Streudiagramm weit rechts oben liegen.

Merkmalsbasierte Zielfunktion

Die Zielfunktion soll das identifizierte Klassifikationsmerkmal mathematisch in Form einer Maßzahl ausdrücken. Dabei sollen Szenarien als gut bewertet werden, bei denen es viele Zeitschritte mit hohem Wert für T_{tc} und gleichzeitig hohem Wert von $M_{Br,Add}$ gab. Szenarien, bei denen diese hohen Werte nicht gleichzeitig oder gar nicht aufgetreten sind, sollen als schlecht bewertet werden.

In Abbildung 5.27 ist die Arbeitsweise der Zielfunktion dargestellt. Die Idee ist hier, die von einem Punkt im Streudiagramm und dem Nullpunkt aufgespannte Fläche eines Rechtecks zu betrachten. Die Punkte, die sehr weit rechts oben im Diagramm liegen, spannen ein Rechteck mit einer großen Fläche auf. Die Punkte, die sehr nahe an der x- oder y-Achse im Diagramm liegen, spannen ein Rechteck mit einer kleinen Fläche auf.

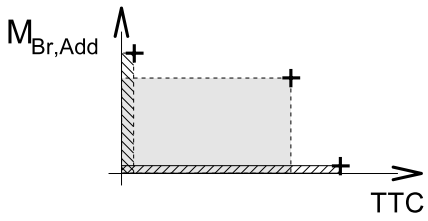


Abbildung 5.27: Betrachten der durch einen Punkt aufgespannten Fläche im Streudiagramm

$$v_{Ziel} = \begin{cases} -\sum T_{tc}(t) \cdot M_{Br,Add}(t) & \text{falls BAS aktiv war} \\ v_{Strafwert} & \text{falls BAS nicht aktiv war} \end{cases} \tag{5.15}$$

In Gleichung 5.15 ist die Zielfunktion des „Abstandsbasierten Bremsassistenten“ dargestellt. Die Zielfunktion enthält eine Fallunterscheidung und liefert den Zielfunktionswert in Abhängigkeit davon, ob die Anwendungsfunktion des „Abstandsbasierten Bremsassistenten“ aktiv war oder nicht. Falls in dem zu bewertenden Szenario die Anwendungsfunktion aktiv wurde, wird der Zielfunktionswert für ein Szenario berechnet, indem die durch alle Punkte aufgespannten Flächen aufsummiert werden. Dabei ist zu beachten, dass das Streudiagramm nur Punkte für diejenigen Zeitpunkte enthält, bei denen die Anwendungsfunktion des „Abstandsbasierten Bremsassistenten“ aktiv war. Falls im zu bewertenden Szenario die Anwendungsfunktion nicht aktiv wurde, wird ein Strafwert von $v_{Strafwert} = +100$ als Zielfunktionswert verwendet.

Kapitel 6

Experimentelle Untersuchungen

Die Methode des evolutionären Funktionstests wird in diesem Kapitel anhand von zwei verschiedenen Anwendungsbeispielen untersucht. Diese sind zum einen ein „Automatisches Parksystem“ und zum anderen ein Bremsassistent der zweiten Generation. Bei beiden Anwendungsbeispielen handelt es sich um abstandsbasierte Fahrerassistenzsysteme, die im Umfeld der Automobilindustrie eine zunehmende Bedeutung erlangen. Fahrerassistenzsysteme sind Anwendungsfunktionen in einem Automobil, die den Fahrer unterstützen. Die Assistenzfunktionen der abstandsbasierten Systeme basieren auf Informationen, die durch umgebungserfassende Sensoren am Fahrzeug bereitgestellt werden. Diese ermöglichen eine Detektion von Objekten in der Fahrzeugumgebung. D.h. beim Testen von abstandsbasierten Fahrerassistenzsystemen muss die Fahrzeugumgebung mit in den Test einbezogen werden.

Das Ziel der experimentellen Untersuchungen ist es, den evolutionären Funktionstest mit dem Stand der Technik, das heißt der manuellen Auswahl von Testfällen und mit einem Zufallstest quantitativ zu vergleichen. Dabei werden für die beiden Anwendungsfunktionen, das „Automatischen Parksystem“ und den „Abstandsbasierten Bremsassistenten“, jeweils evolutionäre, manuelle und zufällige Testläufe durchgeführt.

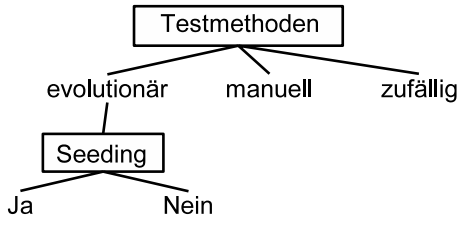


Abbildung 6.1: Testmethoden für die experimentellen Untersuchungen

In Abbildung 6.1 sind die Testmethoden für die experimentellen Untersuchungen dargestellt. Dabei kann beim evolutionären Funktionstest noch unterschieden werden, ob die Testläufe mit einer manuellen Vorgabe der initialen Population (mit Seeding) oder mit einer zufälligen

Auswahl der initialen Population (ohne Seeding) vorgenommen wurden. Manuelle Tests und Zufallstests haben keine Zielfunktion bei der Durchführung der Testfälle. D.h. die manuellen Tests und die Zufallstests können unabhängig von einer Zielfunktion durchgeführt werden.

Bei Testläufen des evolutionären Funktionstests hängen bereits der Verlauf der Tests und damit auch die Testergebnisse von der verwendeten Zielfunktion ab. Daher müssen bei evolutionären Funktionstests grundsätzlich für jede betrachtete Zielfunktion separate Testläufe durchgeführt werden.

Um die Qualität der Ergebnisse des evolutionären Funktionstests mit den Ergebnissen des manuellen und des zufälligen Tests zu vergleichen, werden die vorhandenen Zielfunktionen des evolutionären Funktionstests als Bibliotheksfunktionen für alle Testmethoden eingesetzt. D.h. für alle Testergebnisse in Form von simulierten Szenarien werden mit Hilfe der vorhandenen Zielfunktionen jeweils Zielfunktionswerte berechnet.

6.1 Das „Automatische Parksystem“

In diesem Kapitel werden die Ergebnisse der Experimente zum „Automatischen Parksystem“ dargestellt. Eine vergleichende Analyse und Interpretation der Ergebnisse erfolgt in Kapitel 6.3.

Zunächst werden die Ergebnisse des evolutionären Funktionstests dargestellt. Dabei werden die Experimente von evolutionären Funktionstests ohne Seeding und mit Seeding vorgestellt. Das Prinzip des Seeding wurde bereits in Kapitel 4.1 diskutiert. In Kapitel 6.1.1 sind die Testergebnisse ohne Seeding und in Kapitel 6.1.2 sind die Testergebnisse mit Seeding dargestellt. Die Unterscheidung bei der Testdurchführung soll einen quantitativen Vergleich zwischen den Testläufen mit und ohne Seeding ermöglichen.

Um einen quantitativen Vergleich zu anderen Methoden zu ermöglichen, werden anschließend die Ergebnisse vorgestellt, bei denen die Auswahl der Testdaten manuell bzw. zufällig erfolgte. Dabei enthält Kapitel 6.1.4 die Ergebnisse der Testläufe mit den manuellen Testfalldefinitionen. In Kapitel 6.1.5 sind die Resultate für eine zufällige Auswahl von Testdaten dargestellt.

Alle durchgeführten Experimente werden mit den beiden in Kapitel 5.1.3 vorgestellten Zielfunktionen separat bewertet. Dabei ist beim evolutionären Funktionstest pro betrachteter Zielfunktion ein eigenständiger Testlauf erforderlich, da der Verlauf der Optimierung von der verwendeten Zielfunktion abhängt. Für den manuellen Test und den Zufallstest ist jeweils nur ein Testlauf erforderlich, da hier die Testdurchführung nicht von einer Zielfunktion abhängt. Die Ergebnisse können hier jedoch nach der eigentlichen Testdurchführung mit den Zielfunktionen bewertet werden.

6.1.1 Ergebnisse des evolutionären Funktionstests ohne Seeding

Die nachfolgend vorgestellten Ergebnisse stammen von Testläufen des evolutionären Funktionstests ohne Seeding. Bei diesen Testläufen wurde eine initiale Population mit 25 zufällig ausgewählten Individuen verwendet. Die Testläufe des evolutionären Funktionstests wurden nach 80 Generationen abgebrochen. Dies bedeutet, dass das Abbruchkriterium für die Optimierung sich nicht auf einen erreichten Zielfunktionswert bezog, sondern auf letztlich auf die Anzahl durchgeführter Testfälle.

Bei einer initialen Population mit 25 Individuen werden in der ersten Generation 25 Testfälle durchgeführt, da zunächst für jedes Individuum ein Zielfunktionswert benötigt wird. In den nachfolgenden Generationen werden pro Generation nur noch 23 Testfälle durchgeführt, da für eine neuen Generation nur 23 neue Individuen erzeugt werden. Zwei Individuen aus der vorangegangenen Generation bleiben in der Population erhalten. Bei einem Testlauf mit einer Populationsgröße von 25 Individuen werden über 80 Generationen also insgesamt $(80 \cdot 23) + 2 = 1842$ Testfälle durchgeführt.

Testlauf mit dem Optimierungsziel „Kollision am Punkt P_4“

Das Optimierungsziel bei diesem evolutionären Funktionstest ist es, eine Kollision des Fahrzeugs mit dem Kollisionsbereich am Punkt P_4 zu erreichen. Die Lage von Punkt P_4 ist aus Abbildung 5.4 ersichtlich. Die Zielfunktionswerte aller Individuen der ersten Generation sind in Abbildung 6.2 dargestellt.

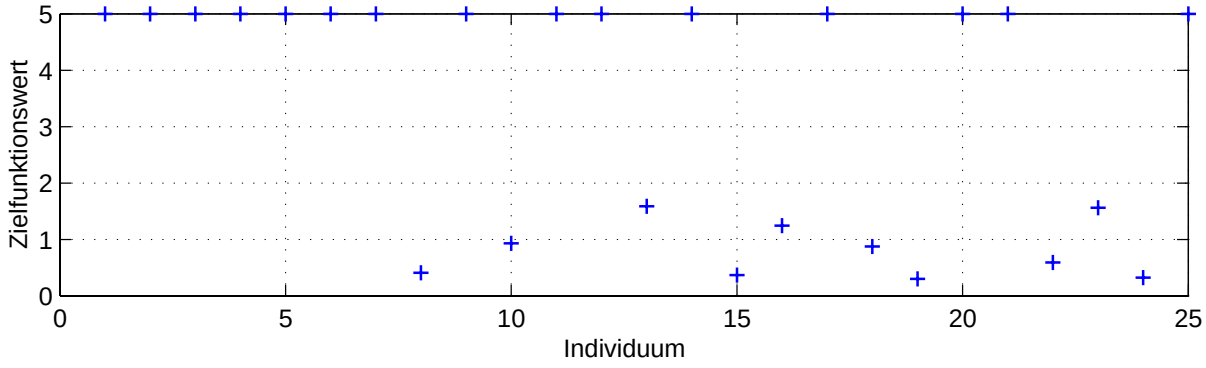


Abbildung 6.2: Zielfunktionswerte aller Individuen der ersten Generation

Von den 25 Individuen führten nur 10 zu einem Testfall, bei dem das „Automatische Parksystem“ eingeparkt hat. Diese sind im Diagramm die Individuen 8, 10, 13, 15, 16, 18, 19, 22, 23 und 24.

Bei 15 Individuen wurde die Parksituation abgelehnt. Die Zielfunktion vergibt bei einem abgelehnten Park-Manöver einen Strafwert von +100. Symbolisch sind diese Individuen

in Abbildung 6.2 mit dem Wert +5 eingetragen, um die Zielfunktionswerte der anderen Individuen besser darzustellen.

Der beste erreichte Zielfunktionswert in der ersten Generation liegt bei +0.30173. Dies bedeutet, dass es in der ersten Generation von Testfällen keine Kollision des Fahrzeugs mit der Ecke am Punkt P_4 gibt. Der minimale Abstand zwischen Fahrzeug und Ecke P_4 beträgt bei diesen Testfällen 30.2 cm.

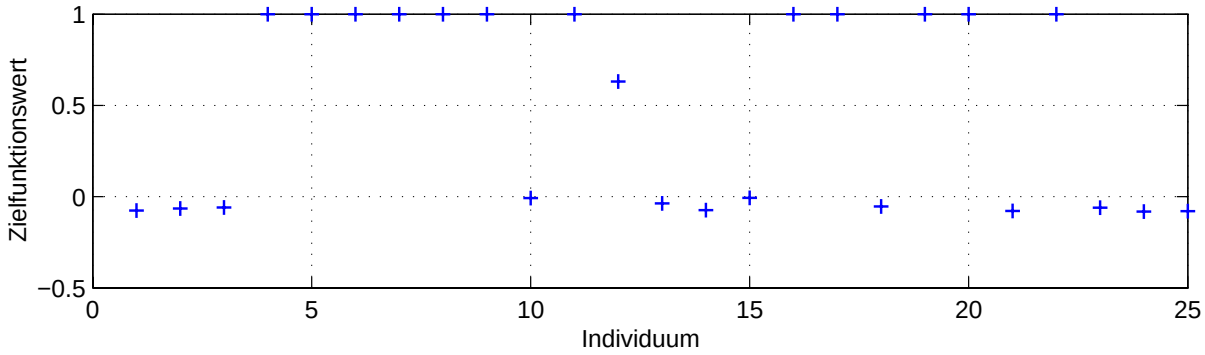


Abbildung 6.3: Zielfunktionswerte aller Individuen der Generation 80

In Abbildung 6.3 sind die Zielfunktionswerte aller Individuen der Generation 80 dargestellt. Dies ist die letzte Generation des Testlaufs. Hier gibt es 12 Individuen, bei denen das Einparken vom „Automatischen Parksystem“ abgelehnt wurde. Dabei handelt es sich um die Individuen 4, 5, 6, 7, 8, 9, 11, 16, 17, 19, 20 und 22. Diese sind im Diagramm mit dem Wert +1 gezeichnet, um eine sinnvolle grafische Darstellung der anderen Individuen im Diagramm zu ermöglichen. Der tatsächliche Zielfunktionswert dieser Individuen ist der Strafwert von +100.

Bei den anderen 13 Individuen hat das „Automatische Parksystem“ die Parksituation akzeptiert und versucht einzuparken. Der beste dabei erreichte Zielfunktionswert liegt bei −0.08109. Dieser Zielfunktionswert wurde bereits in Generation 77 durch Individuum 16 erreicht. Bei diesem Testfall erfolgt eine Kollision des Fahrzeugs mit der Ecke P_4, bei der das Fahrzeug um 8.1 cm in den Kollisionsbereich einfährt.

Die Abbildung 6.4 zeigt den gesamten Verlauf des Tests. Es ist jeweils nur der beste Zielfunktionswert aus einer Generation über alle 80 Generationen hinweg dargestellt. Im Diagramm ist zu sehen, dass der evolutionäre Funktionstest in der ersten Generation mit einem besten Zielfunktionswert von +0.30173 startet. Die evolutionäre Optimierung kann während des Testlaufs den jeweils besten Zielfunktionswert erheblich verbessern. Ab Generation 10 wird dabei ein Testfall gefunden, bei dem eine Kollision am Punkt P_4 auftritt. Im weiteren Verlauf des Testdurchgangs kann der Zielfunktionswert auf −0.08109 weiter verbessert werden.

Die Optimierung kann bis zum Ende des Testlaufs bei 80 Generationen noch kleine Optimierungen erreichen. In Abbildung 6.5 ist ein Ausschnitt der Abbildung 6.4 zwischen Generation

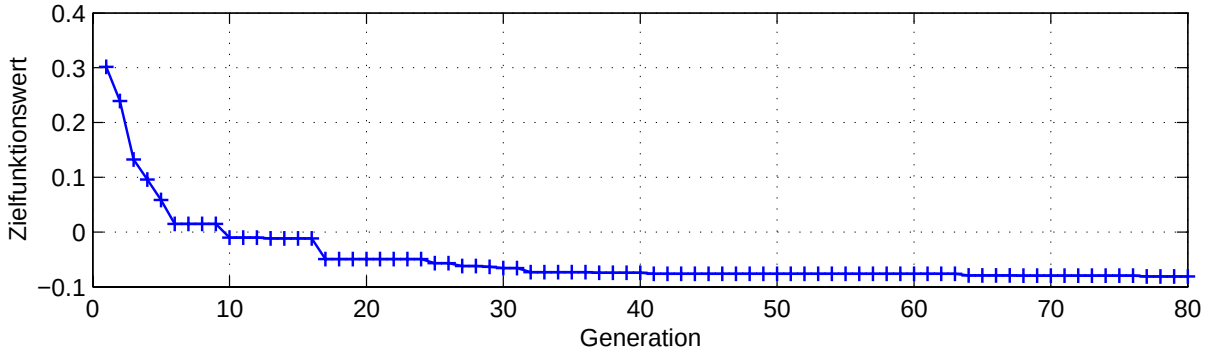


Abbildung 6.4: Jeweils bester Zielfunktionswert einer Generation über 80 Generationen

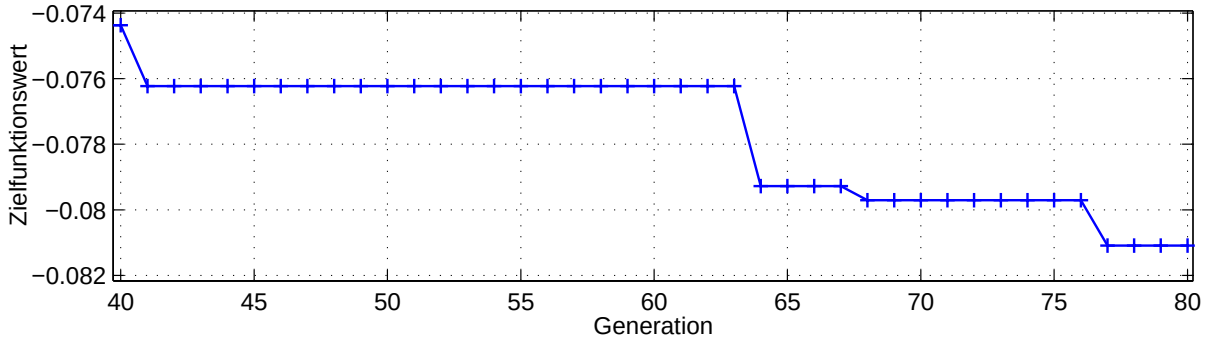


Abbildung 6.5: Vergrößerte Darstellung der besten Zielfunktionswerte zwischen Generation 40 und Generation 80

40 und Generation 80 vergrößert dargestellt. Hier ist zu erkennen, dass der beste gefundene Zielfunktionswert erst ab Generation 77 auftritt und sich bis zum Abbruch nach Generation 80 nicht mehr verbessern läßt. Die Optimierung konnte jedoch zwischen Generation 40 und Generation 77 den Zielfunktionswert weiter verbessern.

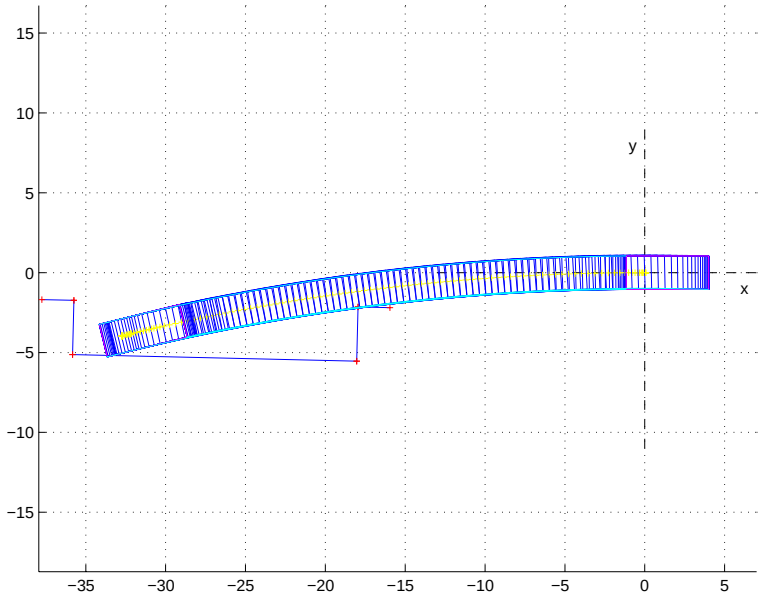


Abbildung 6.6: Testfall Individuum 16 in Generation 77

In Abbildung 6.6 ist das Park-Szenario dargestellt, das als bester Testfall im Rahmen des evolutionären Funktionstests gefunden wurde. Es handelt sich dabei um das Individuum 16 aus der Generation 77. Wie in Abbildung 6.6 zu erkennen ist, tritt die Fehlersituation auf, wenn die Parklücke sehr lang ist, das Fahrzeug sich sehr weit von der Parklücke entfernt befindet und die Ausrichtung der Parklücke leicht zum Fahrzeug gedreht ist.

Testlauf mit dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

Bei diesem evolutionären Funktionstest ist das Optimierungsziel, eine Kollision des Fahrzeugs mit dem Kollisionsbereich an der Strecke $\overline{P_2P_3}$ zu erreichen. Die Lage der Strecke $\overline{P_2P_3}$ ist aus Abbildung 5.4 ersichtlich. In Abbildung 6.7 sind die Zielfunktionswerte aller Individuen der ersten Generation des Testlaufs dargestellt.

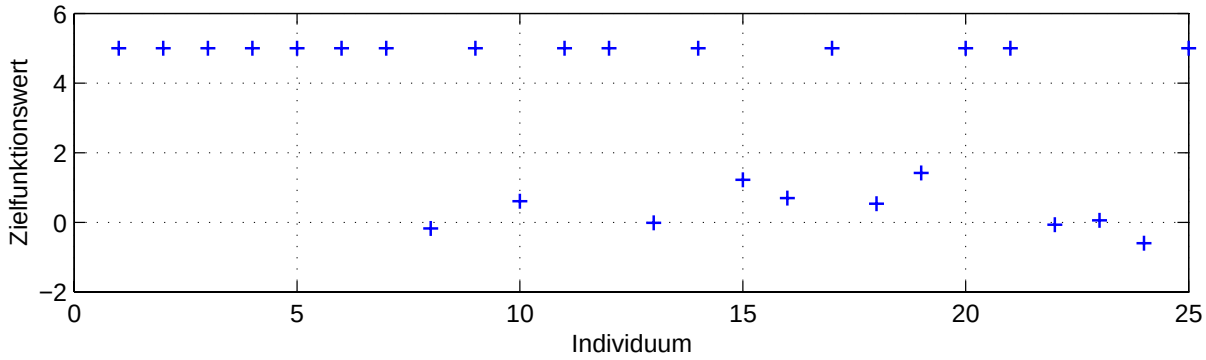


Abbildung 6.7: Zielfunktionswerte aller Individuen der ersten Generation

Da dieser Testlauf ohne Seeding durchgeführt wurde, sind dies die Zielfunktionswerte von 25 zufällig ausgewählten Individuen. Dabei handelt es sich um die gleiche Start-Population, wie sie für den Testlauf mit dem Optimierungsziel „Kollision am Punkt P_4“ in diesem Kapitel verwendet wurde. D.h. die simulierten Szenarien der ersten Generation hier entsprechen exakt den Szenarien der ersten Generation aus diesem Testlauf. Allerdings werden diese Szenarien nun mit einer anderen Zielfunktion bewertet.

Die 10 Individuen, bei denen das „Automatische Parksystem“ eingeparkt hat, sind auch hier die Individuen 8, 10, 13, 15, 16, 18, 19, 22, 23 und 24. Der beste erreichte Zielfunktionswert liegt dabei bereits bei -0.59840. Dies bedeutet, in der ersten Generation von Testfällen tritt bereits eine Kollision des Fahrzeugs mit dem Kollisionsbereich an der Strecke $\overline{P_2P_3}$ auf. Das Fahrzeug tritt dabei mit dem Heck bereits 59.8 cm weit in den Kollisionsbereich ein. Dieser Zielfunktionswert wird bei Individuum 24 erreicht.

Die Bewertung der 15 Individuen, bei denen die Parksituation abgelehnt wurde, ist identisch und liegt beim Strafwert von $+100$. Diese Individuen sind im Diagramm erneut mit dem Wert $+5$ gezeichnet, um die Zielfunktionswerte der anderen Individuen besser darzustellen.

Nachfolgend ist in Abbildung 6.8 das Ergebnis des evolutionären Funktionstest in Form der Zielfunktionswerte aller Individuen in der Generation 80 dargestellt. Die Generation 80 ist die letzte Generation des Testlaufs.

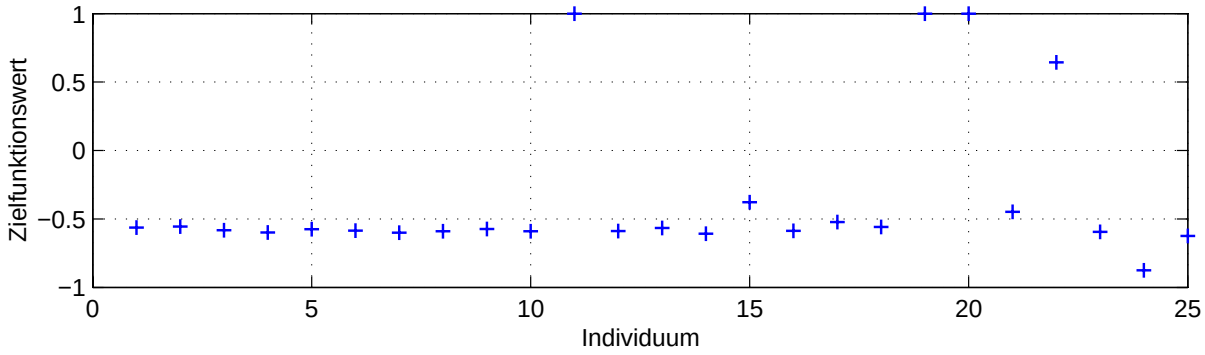


Abbildung 6.8: Zielfunktionswerte aller Individuen in Generation 80

Nur drei der Individuen in dieser Generation haben einen Strafwert von +100 erhalten. Zur besseren Darstellung ist der Zielfunktionswert dieser Individuen hier mit dem Wert +1.0 eingezeichnet.

Der evolutionäre Funktionstest konnte bei diesem Testlauf als besten Zielfunktionswert einen Wert von −0.87584 erreichen. Damit wurde ein Testfall gefunden, der eine noch schwerwiegendere Kollision des Fahrzeugs an der Strecke $\overline{P_2P_3}$ verursacht als der beste Testfall in der ersten Generation. Das Fahrzeug tritt dabei 87.6 cm weit in den Kollisionsbereich ein. Dieser Zielfunktionswert wurde bereits in Generation 38 durch Individuum 15 erreicht. Der Wert von 87.6 cm konnte in den weiteren Generationen dann nicht mehr verbessert werden.

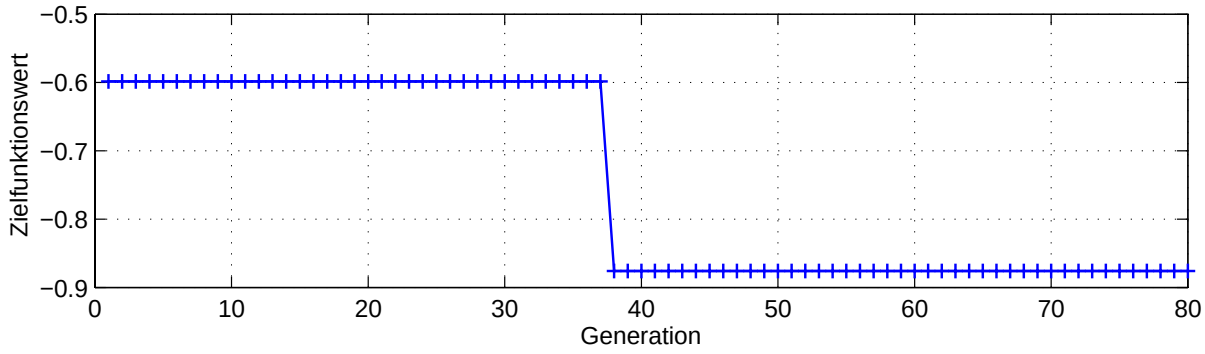


Abbildung 6.9: Jeweils bester Zielfunktionswert einer Generation über 80 Generationen

In Abbildung 6.9 ist der Verlauf des jeweils besten Zielfunktionswerts einer Generation über den gesamten evolutionären Funktionstest dargestellt. Wie bereits in Abbildung 6.7 zu sehen, beginnt die evolutionäre Suche mit einem besten Zielfunktionswert von −0.59840. Bei

diesem Verlauf ist auffällig, dass sich der beste Zielfunktionswert ab der Generation 38 plötzlich schlagartig verbessert und nachfolgend keine weitere Optimierung des Zielfunktionswerts mehr möglich ist. Dies kann so interpretiert werden, dass mit der Generation 38 zum ersten Mal ein neues lokales Minimum („Tal“) betreten wurde. Wie zu sehen ist, wird der Testlauf bei Generation 80 mit einem besten Zielfunktionswert von -0.87584 abgebrochen.

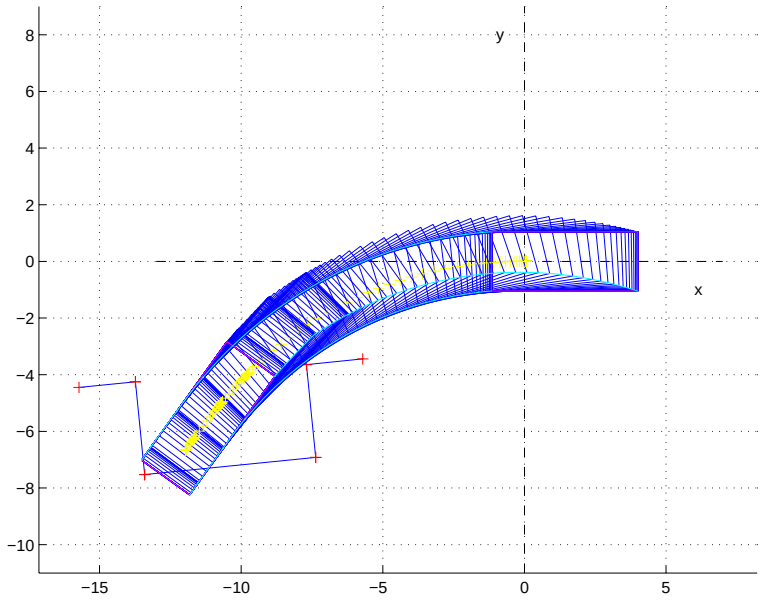


Abbildung 6.10: Park-Szenario des besten gefundenen Testfalls in Generation 38 mit Individuum 15

Das in Generation 38 gefundene Individuum 15 wurde mit dem besten Zielfunktionswert des gesamten Testlaufs bewertet. Das zugehörige Park-Szenario ist in Abbildung 6.10 dargestellt. Bei diesem Park-Szenario ist die Ausrichtung der Parklücke relativ stark vom Fahrzeug weggedreht, dies ist für eine Park-Situation eher untypisch. Die Parklücke ist sehr kurz und relativ breit. Der Abstand des Fahrzeugs zur Parklücke und zur Seite sind normal.

Beim Einpark-Vorgang stößt das Fahrzeug mit dem Heck sehr weit in den Kollisionsbereich an der Strecke $\overline{P_2P_3}$ ein. Das gestellte Optimierungsziel wird durch diesen Testfall sehr gut erfüllt.

6.1.2 Ergebnisse des evolutionären Funktionstests mit Seeding

Nachfolgend sind die Ergebnisse des evolutionären Funktionstests mit Seeding dargestellt. Die initiale Population mit 25 Individuen entspricht dabei den 25 manuell ausgewählten Testfällen aus Kapitel 6.1.4.

Testlauf mit dem Optimierungsziel „Kollision am Punkt P_4“

Das Optimierungsziel bei diesem Testlauf ist es, eine Kollision des Fahrzeugs am Punkt P_4 zu erreichen. Der evolutionäre Funktionstest wurde über 80 Generationen ausgeführt und mit einer initialen Population von 25 Individuen gestartet.

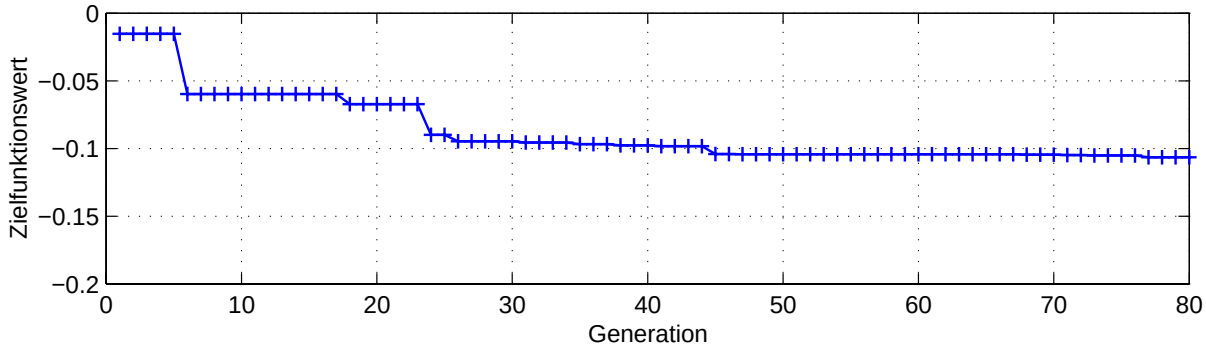


Abbildung 6.11: Jeweils bester Zielfunktionswert einer Generation über 80 Generationen

In Abbildung 6.11 ist der Verlauf des jeweils besten Zielfunktionswerts über alle Generationen des Testlaufs dargestellt. Wie zu sehen ist, beginnt die Optimierung in Generation 1 mit einem besten Zielfunktionswert von -0.01529. Dies entspricht dem besten Testfall der manuellen Tests, die in Kapitel 6.1.4 dargestellt sind.

Im Verlauf der Optimierung kann dieser Wert schnell weiter verbessert werden. Bereits in Generation 10 liegt der minimale Zielfunktionswert der Population nahe dem Wert von -0.06. In Generation 20 kann dieser Wert auf -0.06713 weiter verbessert werden. Der minimal gefundene Zielfunktionswert durch den evolutionären Funktionstest nach 80 Generationen ist -0.10636. Dieser Wert wurde in Generation 77 mit Individuum 2 gefunden und konnte bis Generation 80 nicht mehr weiter verbessert werden.

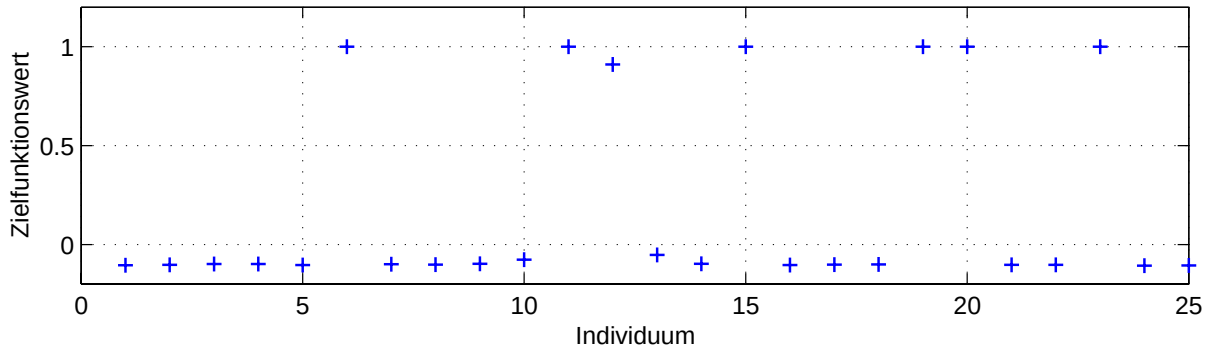


Abbildung 6.12: Zielfunktionswerte aller Individuen in Generation 80

In Abbildung 6.12 sind die Zielfunktionswerte aller Individuen in der Generation 80 dargestellt. Bei nur 6 Individuen wurde die Parksituation abgelehnt, nämlich bei den Individuen 6, 11, 15, 19, 20 und 23. Zur verbesserten Darstellung sind deren Zielfunktionswerte mit dem Wert +1.0 gezeichnet.

Wie im Diagramm zu sehen, ist das Individuum 12 das einzige, das außer den abgelehnten Individuen einen positiven Zielfunktionswert besitzt. Die Zielfunktionswerte der verbleibenden Individuen liegen im Bereich zwischen −0.10636 und −0.05, dabei liegen die besten 16 Individuen im Bereich zwischen −0.10636 und −0.095.

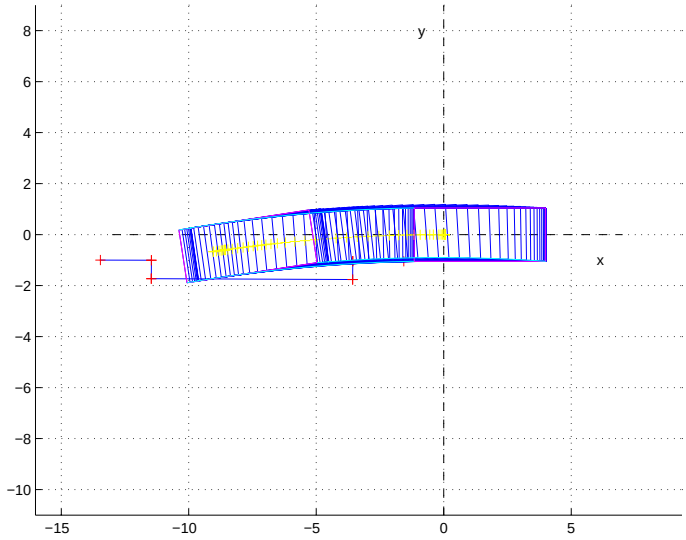


Abbildung 6.13: Park-Szenario des besten gefundenen Testfalls in Generation 77 mit Individuum 2

Das beste Park-Szenario, das durch den evolutionären Funktionstest mit Seeding gefunden wurde, ist in Abbildung 6.13 dargestellt. Bei dieser Park-Situation ist die Ausrichtung der Parklücke leicht zum Fahrzeug gedreht. Die Parklücke ist relativ lang und sehr schmal. Gleichzeitig steht das Fahrzeug sehr nahe an der seitlichen Begrenzung und ist nicht weit von der Parklücke entfernt. Beim Einparkvorgang überfährt das Fahrzeug den Punkt P_4. Ebenso wird der Kollisionsbereich durch das Heck des Fahrzeugs betreten.

Testlauf mit dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

Das Optimierungsziel in diesem Kapitel ist es, eine Kollision des Fahrzeugs an der Strecke $\overline{P_2P_3}$ zu erreichen.

In Abbildung 6.14 ist der Verlauf des jeweils besten Zielfunktionswerts einer Generation über alle 80 betrachteten Generationen dargestellt. Wie zu sehen ist, beginnt die Optimierung in Generation 1 mit einem besten Zielfunktionswert von −0.27557. Dies entspricht dem besten

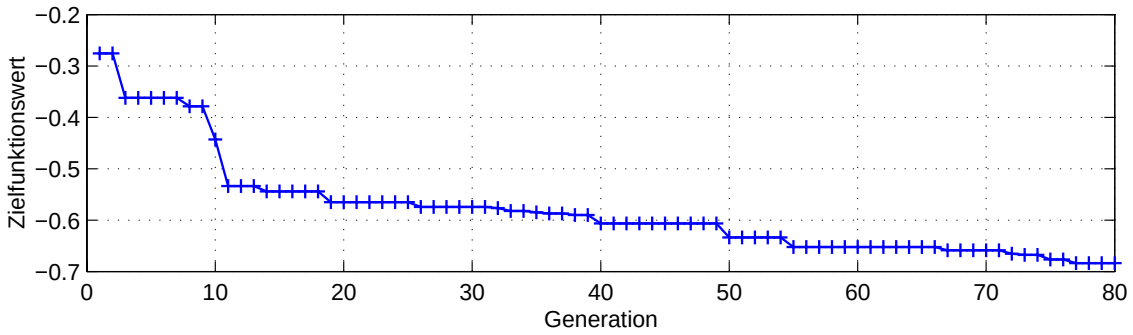


Abbildung 6.14: Jeweils bester Zielfunktionswert einer Generation über 80 Generationen

Testfall aus den manuellen Tests, die in Kapitel 6.1.4 dargestellt sind. Die Optimierung kann in den ersten Generationen des Testlaufs den Zielfunktionswert rasch weiter verbessern. In Generation 11 konnte ein Zielfunktionswert von -0.53352 erreicht werden und in Generation 20 liegt der Zielfunktionswert bei -0.56505. Bis Generation 80 kann der Zielfunktionswert auf -0.68356 verbessert werden. Dieser Wert wurde bereits in Generation 77 mit Individuum 15 gefunden.

In Abbildung 6.15 sind die Zielfunktionswerte aller Individuen in Generation 80 dargestellt. Bei den beiden Individuen 19 und 20 wurde der Parkvorgang abgelehnt. Ihre Zielfunktionswerte sind mit dem Wert $+1$ gezeichnet, um die Darstellung zu verbessern.

Das Individuum 15 stellt einen Einparkvorgang dar, bei dem das Fahrzeug nicht den Kollisionsbereich an der Strecke $\overline{P_2P_3}$ betreten hat. Daher ist hier der Zielfunktionswert positiv. Bei allen verbleibenden Individuen betritt das Fahrzeug beim Einparkvorgang den Kollisionsbereich an der Strecke $\overline{P_2P_3}$.

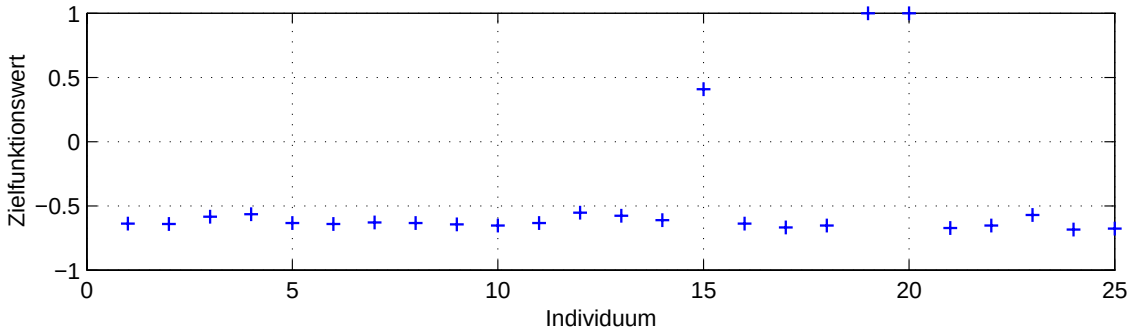


Abbildung 6.15: Zielfunktionswerte aller Individuen in Generation 80

Das beste Park-Szenario, das durch den evolutionären Funktionstest mit Seeding gefunden wurde, ist in Abbildung 6.16 dargestellt. Dies ist das Park-Szenario von Individuum 15 aus Generation 77.

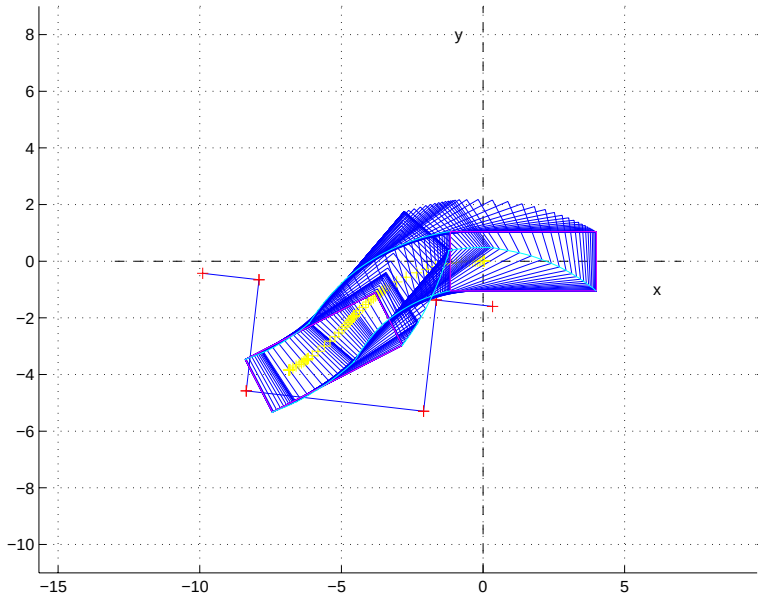


Abbildung 6.16: Park-Szenario des besten gefundenen Testfalls

Bei dieser Park-Situation ist die Ausrichtung der Parklücke relativ stark zum Fahrzeug verdreht. Die Parklücke ist sehr kurz und relativ breit. Das Fahrzeug steht sehr nahe an der Parklücke mit relativ wenig seitlichem Abstand. Beim Einparken fährt das Heck des Fahrzeugs sehr weit in den Kollisionsbereich an der Strecke $\overline{P_2P_3}$ ein.

Die Abbildung 6.17 zeigt alle Zielfunktionswerte des Testlaufs mit Seeding und dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“. Um den Verlauf der Optimierung sichtbar zu machen wurden die positiven Zielfunktionswerte beim Wert $+1$ abgeschnitten. Von vorne nach hinten sind die Zielfunktionswerte der einzelnen Individuen einer Generation sortiert in aufsteigender Reihenfolge aufgezeichnet. Von rechts nach links sind die 80 Generationen des Testlaufs dargestellt und zeigen somit den Verlauf der Optimierung. Die vordere Schnittkante der Fläche entspricht dem Verlauf der Kurve in Abbildung 6.14, er stellt den jeweils besten Zielfunktionswert einer Generation über 80 Generationen dar.

Wie in der Abbildung 6.17 rechts zu sehen ist startet die Optimierung in der initialen Population mit mehreren Individuen, die einen Zielfunktionswert im Bereich -0.3 bis -0.2 haben. Durch die Optimierung gelingt es rasch, das Niveau der Zielfunktionswerte zu minimieren. Ab Generation 40 hat sich dieses Niveau bei einem Großteil der Individuen angeglichen. Bis zum Ende der Optimierung in Generation 80 erhalten einzelne Individuen immer wieder einen Strafwert.

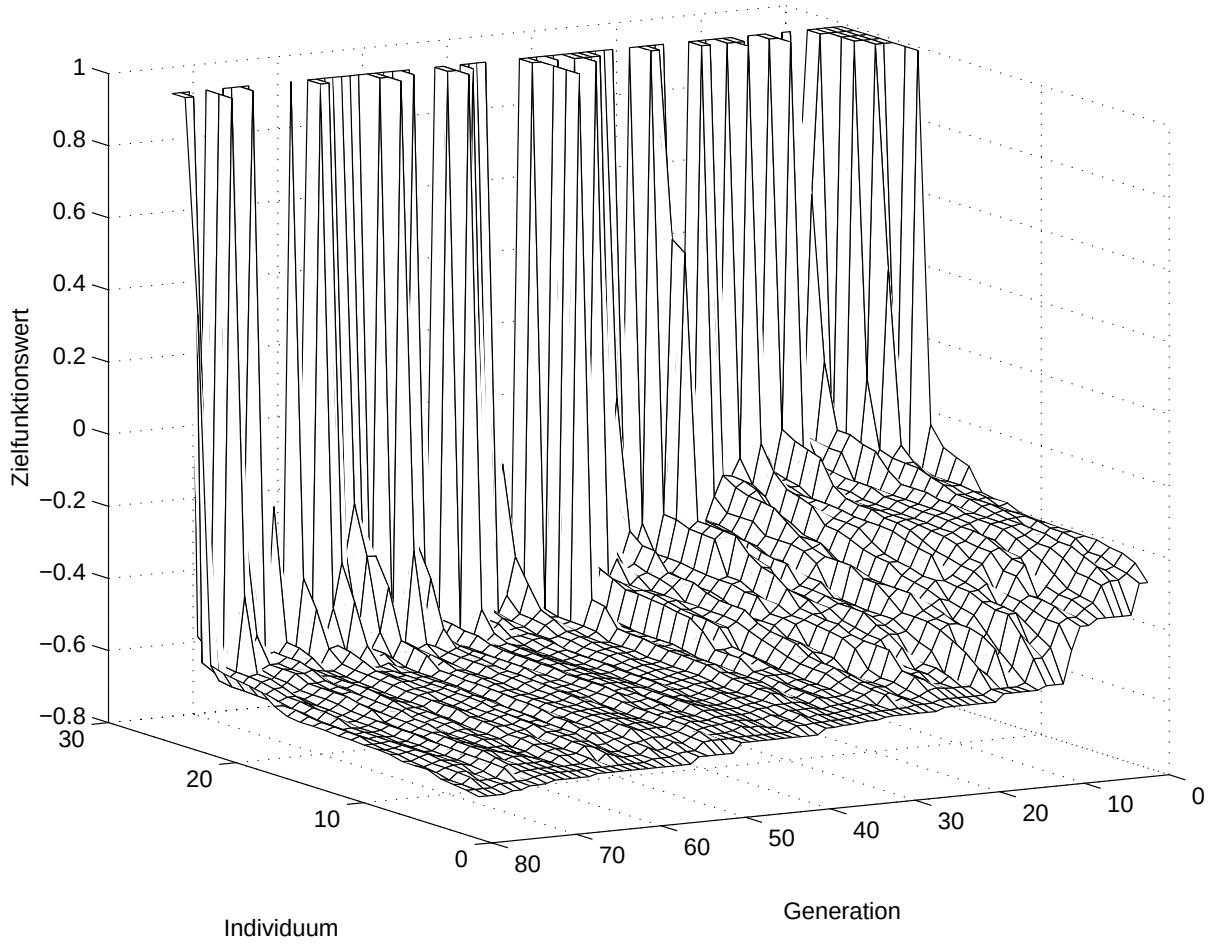


Abbildung 6.17: Zielfunktionswerte über alle Generationen

6.1.3 Ergebnisse des evolutionären Funktionstests mit Seeding manueller und zufälliger Werte

Bei der Durchführung der Testläufe der evolutionären Funktionstests mit Seeding manueller und zufälliger Werte wurde die Größe der Population auf 50 Individuen erhöht. Dabei wurde die initiale Population zusammengesetzt aus der initialen Population der Testläufe mit Seeding und aus der initialen Population der Testläufe ohne Seeding.

Um die Anzahl der insgesamt durchgeführten Testfälle vergleichbar zu halten mit der Anzahl der durchgeführten Testfälle bei den anderen Testläufen, wird die Zahl der Generationen bei diesen Tests auf 41 Generationen begrenzt. D.h. es werden bei den Testläufen des evolutionären Funktionstests mit Seeding manueller und zufälliger Werte insgesamt $(41 \cdot 45) + 5 = 1850$ Testfälle ausgeführt.

Testlauf mit dem Optimierungsziel „Kollision am Punkt P_4“

Das Optimierungsziel bei diesem evolutionären Funktionstest ist es, eine Kollision des Fahrzeugs mit dem Kollisionsbereich am Punkt P_4 zu erreichen.

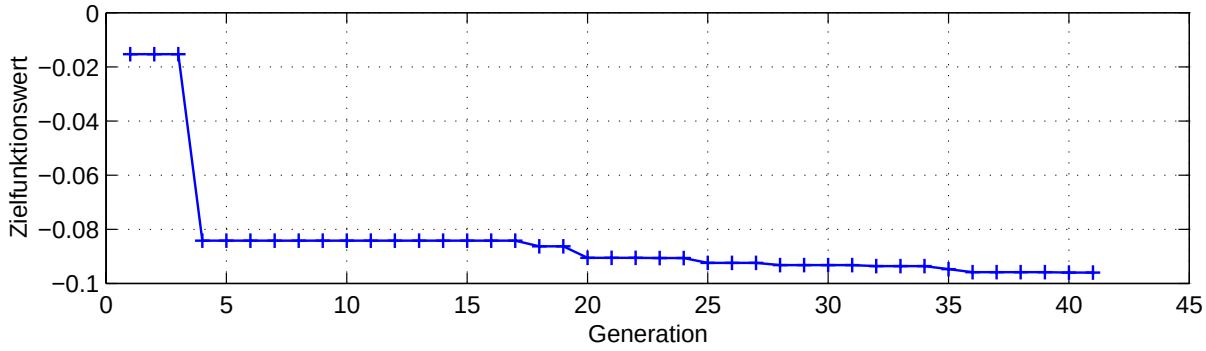


Abbildung 6.18: Jeweils bester Zielfunktionswert einer Generation über 41 Generationen

In Abbildung 6.18 ist der Verlauf des jeweils besten Zielfunktionswerts über alle 41 Generationen des Testlaufs dargestellt. Wie zu sehen ist, beginnt die Optimierung in Generation 1 mit einem besten Zielfunktionswert von -0.01529. Dies entspricht dem besten Testfall aus der ersten Generation der Testläufe mit Seeding und damit auch dem besten Testfall der manuellen Tests. D.h. der evolutionäre Funktionstest beginnt bereits mit einem Testfall, bei dem eine Kollision am Punkt P_4 auftritt.

Wie in Abbildung 6.18 zu sehen ist, kann die evolutionäre Optimierung in Generation 4 den Zielfunktionswert sprunghaft um -0.06887 weiter verbessern auf einen Wert von -0.08416. Im weiteren Verlauf kann der Zielfunktionswert verbessert werden, in Generation 20 auf -0.09047 und in Generation 30 auf -0.09317. In Generation 41 wird ein Zielfunktionswert von -0.09596 erreicht.

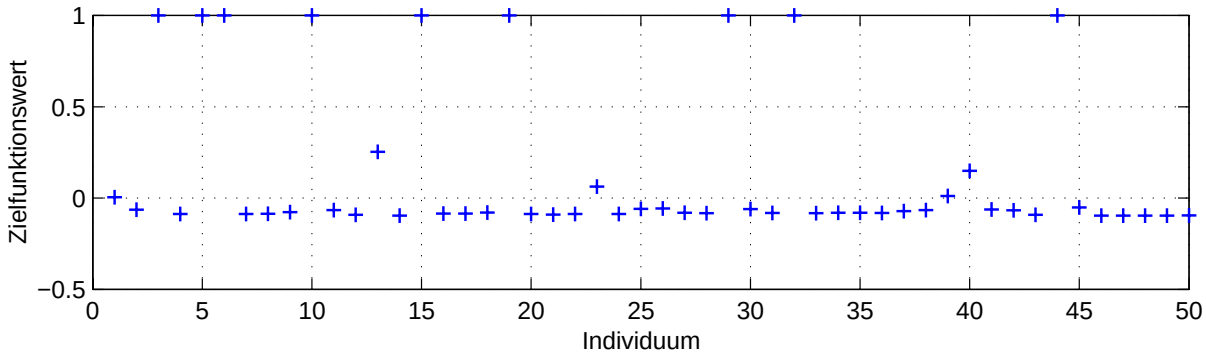


Abbildung 6.19: Zielfunktionswerte aller Individuen in Generation 41

In Abbildung 6.19 sind die Zielfunktionswerte der Population in Generation 41 dargestellt. Darin ist erkennbar, dass bei den Individuen 3, 5, 6, 10, 15, 19, 29, 32 und 44 der Einparkvorgang abgelehnt wurde. Die Zielfunktionswerte dieser Individuen wurden zur besseren Darstellung mit dem Wert +1 gezeichnet. Die Individuen 1, 13, 23, 39 und 40 haben positive Zielfunktionswerte und stellen somit Testfälle dar, bei denen keine Kollision am Punkt P_4 erfolgt.

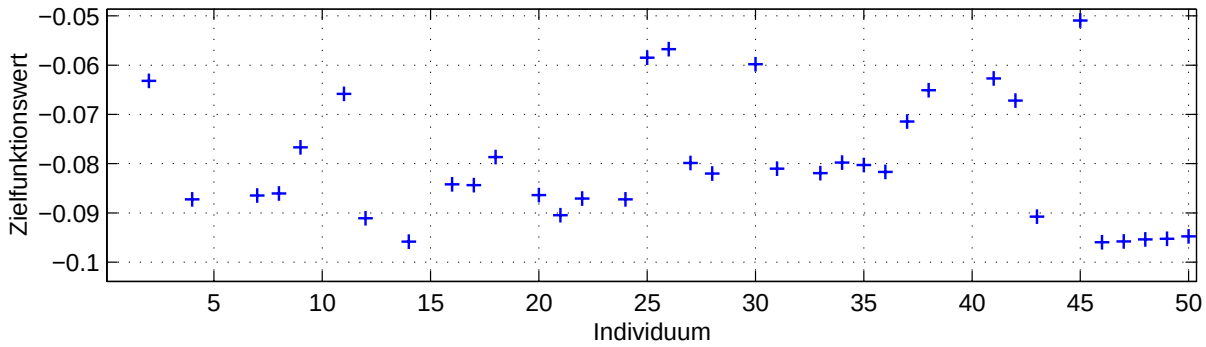


Abbildung 6.20: Vergrößerter Ausschnitt der Zielfunktionswerte aller Individuen in Generation 41

In Abbildung 6.20 ist ein Ausschnitt aus Abbildung 6.19 vergrößert dargestellt. Es ist zu sehen, dass sich von den 50 Individuen der Population die Zielfunktionswerte von 36 Individuen im Bereich zwischen -0.05 und -0.1 aufhalten. Das Individuum 46 erreicht dabei mit einem Wert von -0.09596 den besten Zielfunktionswert, der durch diesen evolutionären Funktionstest gefunden wurde. Dieses Individuum wurde bereits in Generation 40 mit Individuum 14 gefunden. Dessen Zielfunktionswert konnte in Generation 41 von keinem neuen Individuum verbessert werden.

In Abbildung 6.21 ist das Park-Szenario von Individuum 14 aus der Generation 40 dargestellt. Bei der gestellten Park-Situation steht das Fahrzeug parallel zur Parklücke. Die

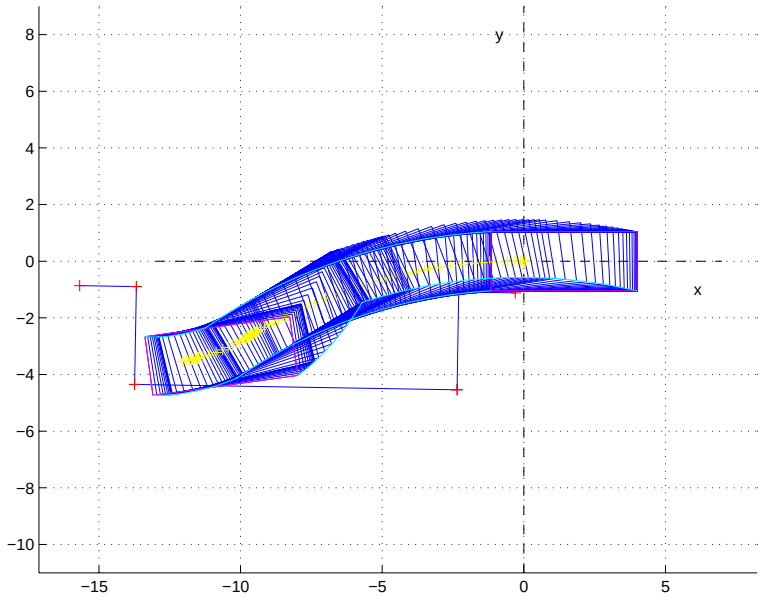


Abbildung 6.21: Park-Szenario des besten gefundenen Testfalls in Generation 40 mit Individuum 14

Parklücke ist ausreichend groß. Es fällt auf, dass zu Beginn des Einparkens überhaupt kein seitlicher Abstand zwischen Fahrzeug und Parklücke besteht. Das Fahrzeug steht außerdem sehr nahe an der Parklücke. Beim Einparken überfährt das Fahrzeug den Kollisionsbereich am Punkt P_4.

Testlauf mit dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

Das Optimierungsziel bei diesem evolutionären Funktionstest ist es, eine Kollision des Fahrzeugs mit dem Kollisionsbereich an der Strecke $\overline{P_2P_3}$ zu verursachen.

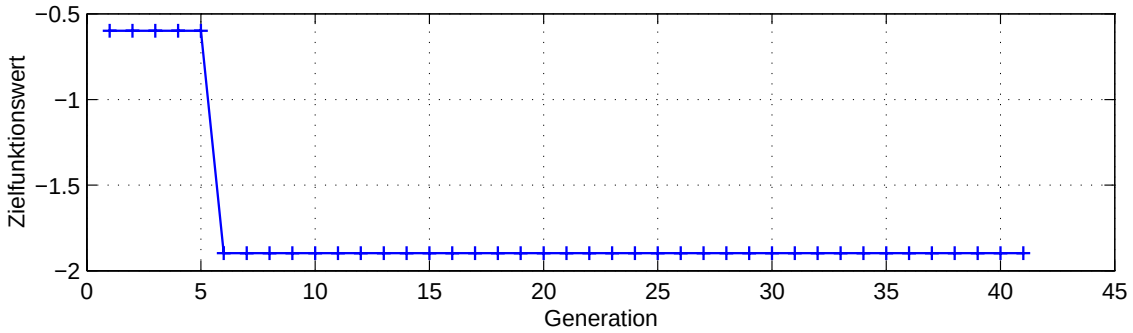


Abbildung 6.22: Jeweils bester Zielfunktionswert einer Generation über 41 Generationen

In Abbildung 6.22 ist der Verlauf des jeweils besten Zielfunktionswerts über alle 41 Generationen des Testlaufs dargestellt. Die Optimierung beginnt in Generation 1 mit einem Zielfunktionswert von −0.59840. Dieser Wert entspricht dem besten Zielfunktionswert, der in Generation 1 des evolutionären Funktionstests ohne Seeding – dargestellt in Kapitel 6.1.1 – erreicht wurde. Dies bedeutet, dass der evolutionäre Funktionstest die Optimierung mit einem Testfall beginnt, der bereits zu einer Kollision des Fahrzeugs mit dem Kollisionsbereich an der Strecke $\overline{P_2P_3}$ führt. Dabei taucht das Fahrzeug mit 59.8 cm in den Kollisionsbereich ein. Beim Verlauf fällt auf, dass sich von Generation 5 auf 6 der Zielfunktionswert plötzlich um −1.29873 auf einen Wert von −1.89713 verbessert. Dieser Wert bleibt dann der beste gefundene Zielfunktionswert über 41 Generationen.

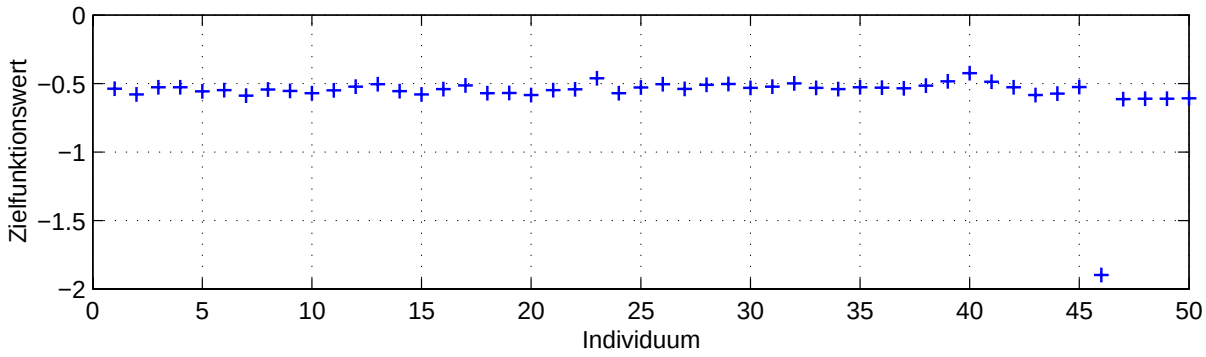


Abbildung 6.23: Zielfunktionswerte aller Individuen in Generation 41

In Abbildung 6.23 sind die Zielfunktionswerte der Population in Generation 41 dargestellt. Bei keinem Individuum in dieser Generation wurde der Einparkvorgang abgelehnt. Bei 49 Individuen der Population liegt der Zielfunktionswert im Bereich zwischen −0.39030 und −0.60948. Nur der Zielfunktionswert von Individuum 46 liegt bei −1.89713. Dieses Individuum stammt aus Generation 6 und tauchte dort als Individuum 39 erstmals auf.

In Abbildung 6.24 ist das Park-Szenario von Individuum 39 aus der Generation 6 dargestellt. Die Parklücke in dieser Park-Situation ist sehr breit und relativ kurz. Die Ausrichtung der Parklücke ist vom Fahrzeug weg gedreht.

Das Fahrzeug fährt bei diesem Manöver mit der ganzen Breite des Hecks in den Kollisionsbereich ein. Dieses Manöver entspricht voll dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“.

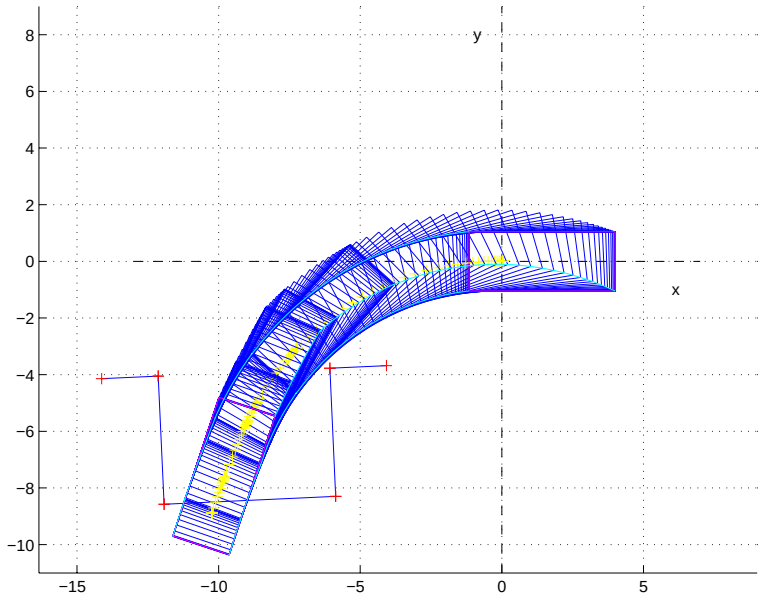


Abbildung 6.24: Park-Szenario des besten gefundenen Testfalls in Generation 06 mit Individuum 39

6.1.4 Ergebnisse der manuellen Tests

In diesem Kapitel werden die Ergebnisse der Testläufe mit manuell ausgewählten Testfällen dargestellt. Zunächst wird die Auswahl der manuellen Testfälle für das „Automatische Parksystem“ anhand von Abbildung 6.25 erläutert.

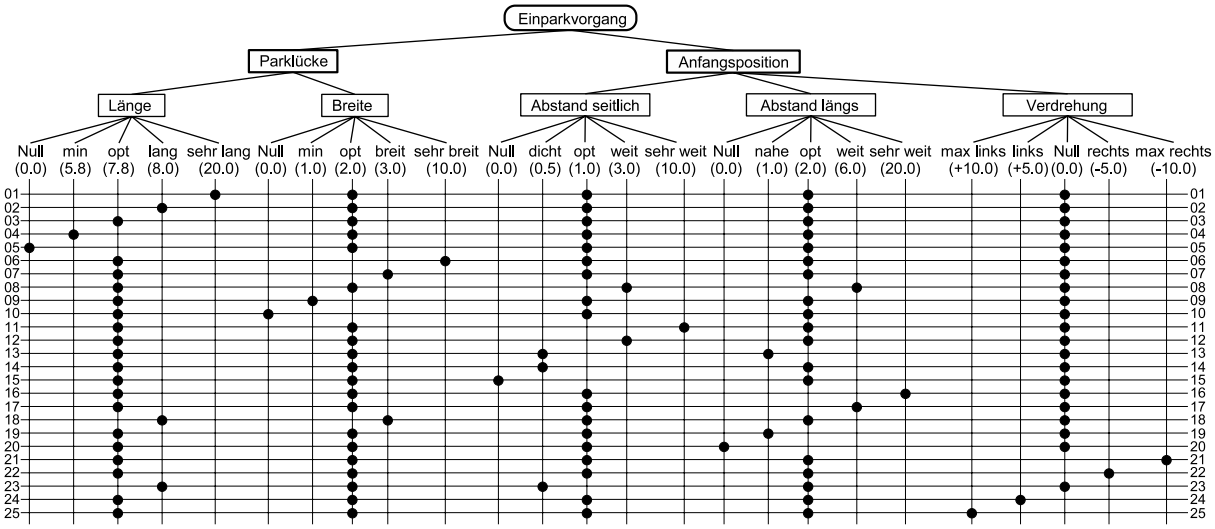


Abbildung 6.25: Klassifikationsbaum der manuellen Testfälle

Abbildung 6.25 zeigt den Klassifikationsbaum für die manuelle Auswahl der Testfälle für das „Automatische Parksystem“. Klassifikationen sind die Länge und die Breite der Parklücke, der seitliche Abstand, der Längsabstand sowie die Verdrehung des Fahrzeugs in der Anfangsposition.

Es sind jeder der Klassifikationen jeweils fünf Klassen zugeordnet. Die untere und obere Klasse repräsentieren dabei die Grenzen des gewählten Wertebereichs für die jeweilige Klassifikation. Die Festlegung des Wertebereichs erfolgte für die Parklücke und für die Ausgangsposition nach unterschiedlichen Gesichtspunkten. Für die Parklücke lassen sich harte Grenzen angeben, in denen ein Einparken noch möglich ist.

Für die Ausgangsposition des Fahrzeugs lassen sich solche Grenzen nicht mehr so angeben, da es von allen Parametern gleichzeitig abhängt, ob ein Einparken aus dieser Ausgangsposition und mit dieser Parklückengröße noch möglich ist. Es mag z.B. mit einem Verdrehungswinkel von -5 Grad für manche Ausgangspositionen noch möglich sein einzuparken und für andere Ausgangspositionen nicht mehr. Daher wurden die Grenzen der Wertebereiche für die Klassifikationen der Ausgangsposition nach dem Gesichtspunkt ausgewählt, was in der Praxis bei normalem Gebrauch auftritt und daher aus funktionaler Sicht sinnvoll ist. Die manuell definierten Testfälle aus dem Klassifikationsbaum in Abbildung 6.25 wurden mit der Testumgebung ausgeführt. Bei sieben Testfällen hat das „Automatische Parksystem“ die Parksituation abgelehnt, bei den verbleibenden 18 Testfällen hat das „Automatische Parksystem“ die Parksituation akzeptiert und ein Einparkvorgang wurde durchgeführt.

Bewertung nach dem Optimierungsziel „Kollision am Punkt P_4“

Die Testergebnisse der manuell definierten Testfälle aus dem Klassifikationsbaum in Abbildung 6.25 wurden mit der Zielfunktion bewertet, welche als Optimierungsziel eine „Kollision am Punkt P_4“ hat. Die berechneten Zielfunktionswerte sind in Abbildung 6.26 dargestellt. Nach rechts sind die Rückgabewerte der Zielfunktion für die 25 durchgeführten Testfälle dargestellt. Nach oben ist das Minimum aus dem berechneten Zielfunktionswert oder aus dem Wert +1 dargestellt. Dadurch wird die obere Grenze des Wertebereichs für positive Zielfunktionswerte im Diagramm auf +1 begrenzt, um eine genauere Darstellung der Zielfunktionswerte im negativen Bereich zu erhalten und um keine Strafwerte darzustellen.

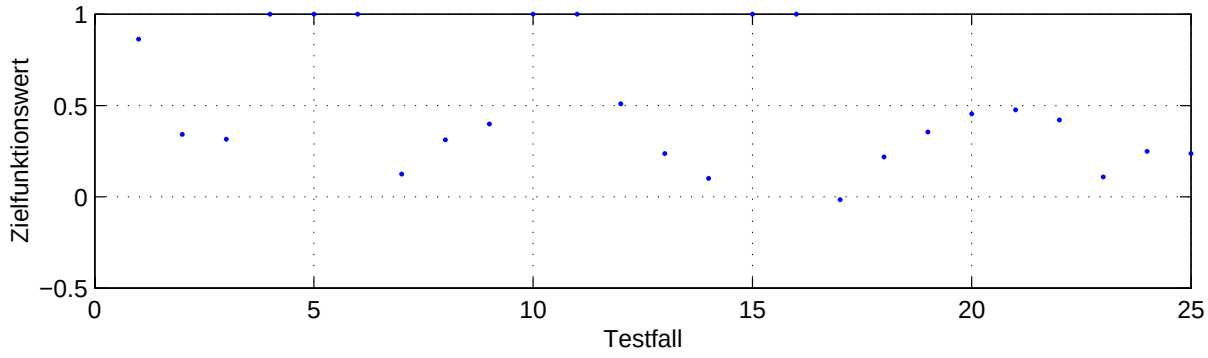


Abbildung 6.26: Manuelle Testfälle bewertet nach Optimierungsziel „Kollision am Punkt P_4“

Wie in Abbildung 6.26 zu sehen ist, erreicht Testfall 17 bereits einen negativen Zielfunktionswert von −0.01529. In anderen Worten, bei diesem Testfall überfährt das einparkende Fahrzeug beim Einparken den Punkt P_4 mit einer Eindringtiefe von 1.5 cm. Die oberen Punkte zeigen die sieben Testfälle 4, 5, 6, 10, 11, 15 und 16, bei denen das „Automatische Parksystem“ die Parksituation abgelehnt hat. Zur besseren Darstellung wurde der Strafwert von +100 bei diesen Testfällen auf +1 geändert.

In Abbildung 6.27 ist das beste Park-Szenario der manuellen Testfälle bei der Bewertung nach dem Optimierungsziel „Kollision am Punkt P_4“dargestellt. Es ist zu sehen, dass das Fahrzeug am Punkt P_4 durch den Kollisionsbereich fährt.

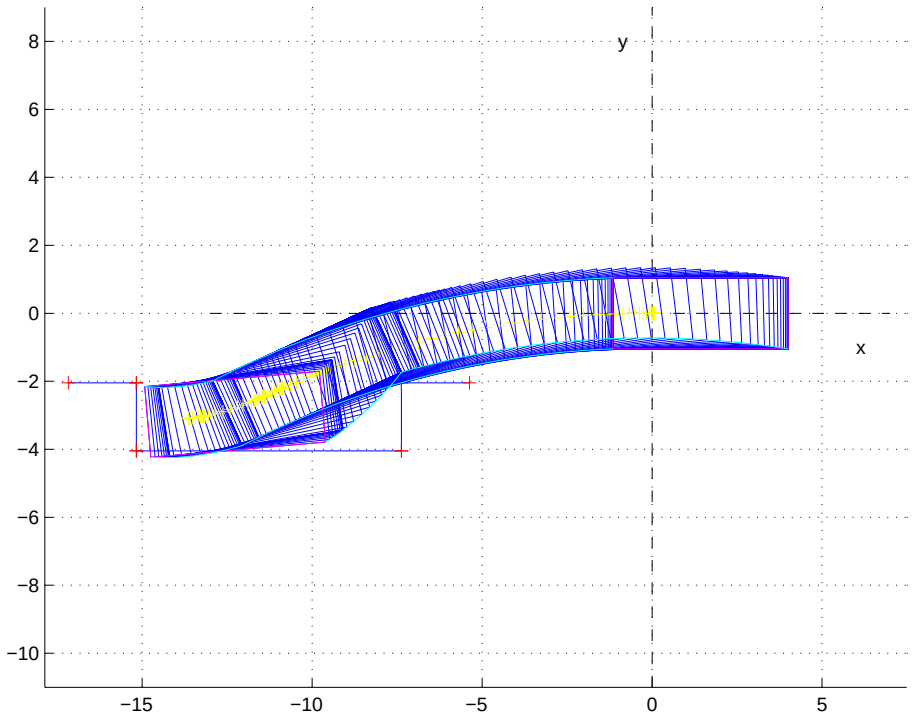


Abbildung 6.27: Bestes Szenario der manuellen Testfälle bei der Bewertung nach dem Optimierungsziel „Kollision am Punkt P_4“

Bewertung nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

In der nachfolgenden Abbildung 6.28 sind die Zielfunktionswerte dargestellt, die nach dem Abstandskriterium an der Strecke $\overline{P_2P_3}$ für die manuellen Testergebnisse berechnet wurden.

Nach rechts sind die 25 durchgeführten Testfälle dargestellt. Nach oben ist das Minimum aus dem berechneten Zielfunktionswert oder aus dem Wert 1 dargestellt. Dadurch wird der Wertebereich für positive Zielfunktionswerte im Diagramm auf +1 begrenzt, um eine genauere Darstellung der Zielfunktionswerte im negativen Bereich zu erhalten.

Der beste Zielfunktionswert der manuellen Testfälle nach dem Abstandskriterium an der Strecke $\overline{P_2P_3}$ ist -0.27557. Dieser Wert wird von Testfall 25 erreicht. Das zugehörige Szenario ist in Abbildung 6.29 dargestellt.

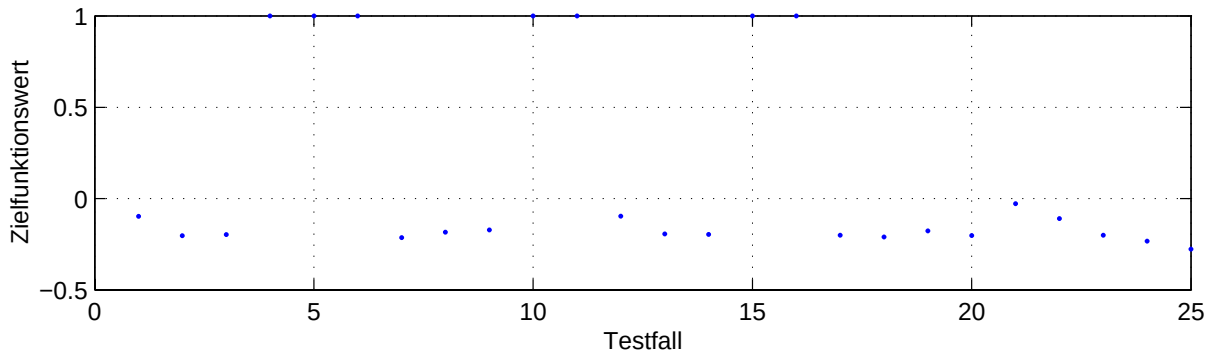


Abbildung 6.28: Manuelle Testfälle bewertet nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

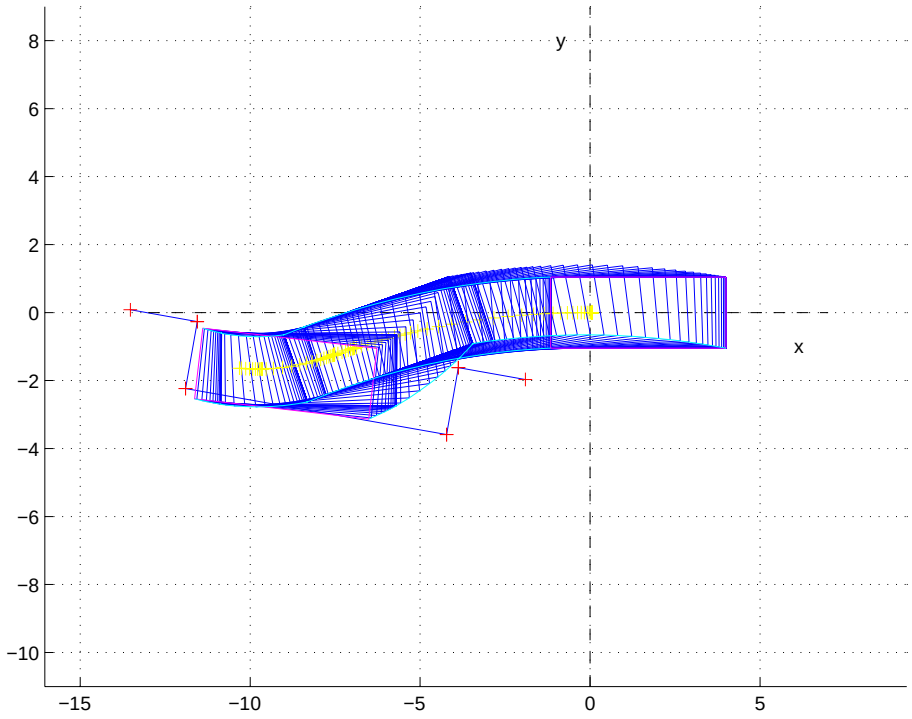


Abbildung 6.29: Bestes Szenario der manuellen Testfälle bei der Bewertung nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

6.1.5 Ergebnisse der Zufallstests

In diesem Kapitel werden die Ergebnisse der Zufallstests dargestellt. Dabei wurden die Testdaten für 1842 Testfälle zufällig ausgewählt und als Simulation ausgeführt. Von den 1842 ausgeführten Testfällen hat bei 1130 Testfällen das „Automatische Parksystem“ den Einparkvorgang abgelehnt. D.h. nur in 712 Fällen hat das „Automatische Parksystem“ das Fahrzeug eingeparkt.

Nachfolgend werden die ausgeführten Testfälle mit Hilfe der beiden Zielfunktionen des evolutionären Funktionstest bewertet. D.h. für jeden einzelnen Testfall werden zwei verschiedene Zielfunktionswerte berechnet, zum einen ein Zielfunktionswert nach dem Optimierungsziel „Kollision am Punkt P_4“ und zum anderen ein Zielfunktionswert nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“. Damit sollen in Kapitel 6.3 die durch den evolutionären Funktionstest gefundenen Testfälle den Testfällen der Zufallstests gegenüber gestellt werden, um die Leistungsfähigkeit des evolutionären Funktionstests mit dem Zufallstest zu vergleichen.

Bewertung nach dem Optimierungsziel „Kollision am Punkt P_4“

Der beste Zielfunktionswert aus dem Zufallstest, bewertet nach dem Optimierungsziel „Kollision am Punkt P_4“, ist -0.04131. D.h. es wurde ein Testfall gefunden, bei dem das Fahrzeug mit 4.1 cm in den Kollisionsbereich am Punkt P_4 eindringt. In Abbildung 6.30 sind die Zielfunktionswerte aller 1842 Testfälle mit der Bewertung nach diesem Optimierungsziel dargestellt.

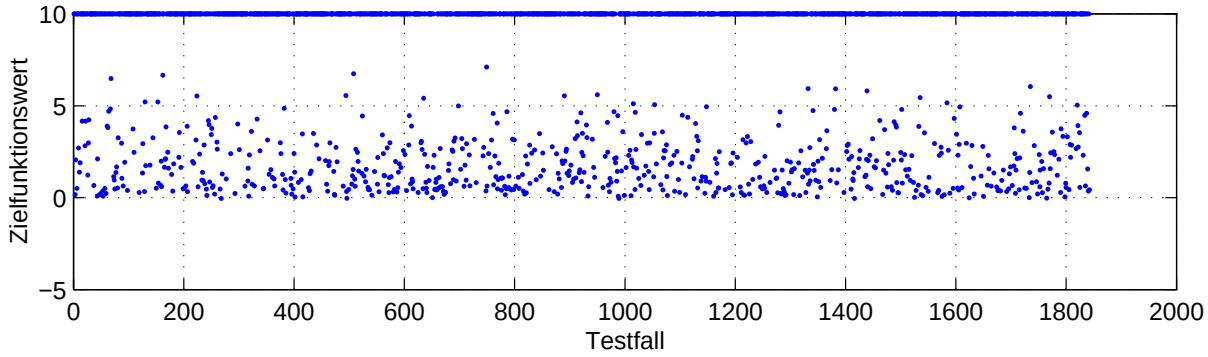


Abbildung 6.30: Zielfunktionswerte nach dem Optimierungsziel „Kollision am Punkt P_4“

Die in Abbildung 6.30 mit dem Zielfunktionswert 10 dargestellten Testfälle entsprechen den vom „Automatischen Parksystem“ abgelehnten Parksituationen. Zur besseren Darstellung in der Abbildung wurde der Strafwert von $+100$ auf $+10$ reduziert.

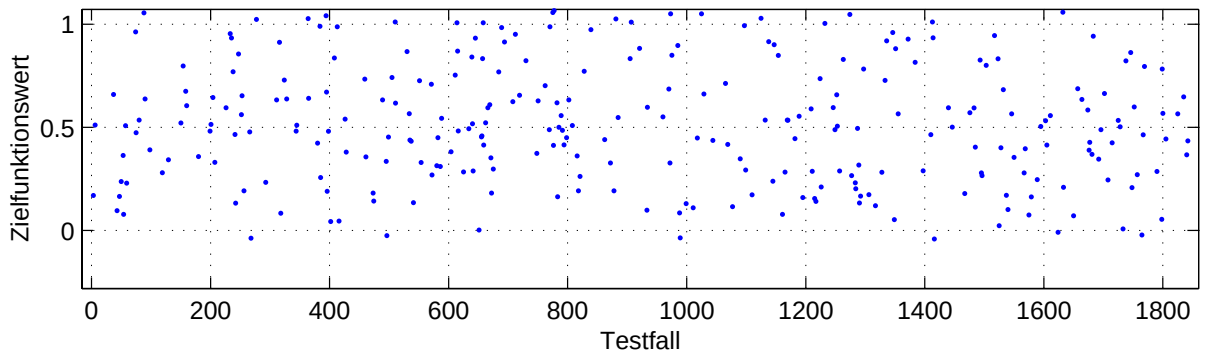


Abbildung 6.31: Ausschnittvergrößerung der Zielfunktionswerte nach dem Optimierungsziel „Kollision am Punkt P_4“

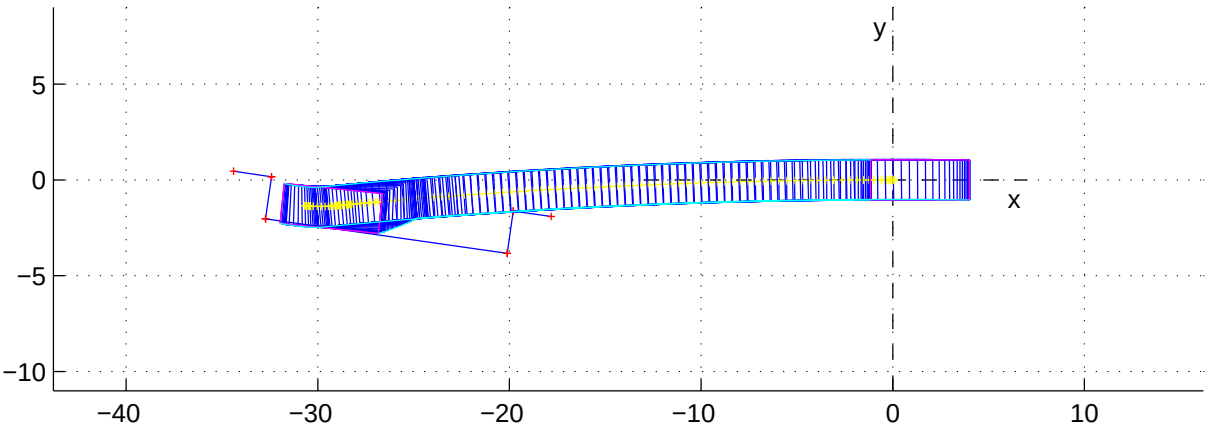


Abbildung 6.32: Bester gefundener Testfall des Zufallstests bewertet nach dem Optimierungsziel „Kollision am Punkt P_4“

Bewertung nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

Der beste Zielfunktionswert aus dem Zufallstest, bewertet nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“, ist -0.60649. In Abbildung 6.33 sind die Zielfunktionswerte aller 1842 Testfälle mit der Bewertung nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“ dargestellt. Abbildung 6.34 zeigt eine Ausschnittvergrößerung im Wertebereich zwischen -0.5 und $+0.5$. Die Abbildung 6.35 stellt den besten gefundenen Testfall der Zufallstests dar, wenn deren Ergebnisse nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“ bewertet werden.

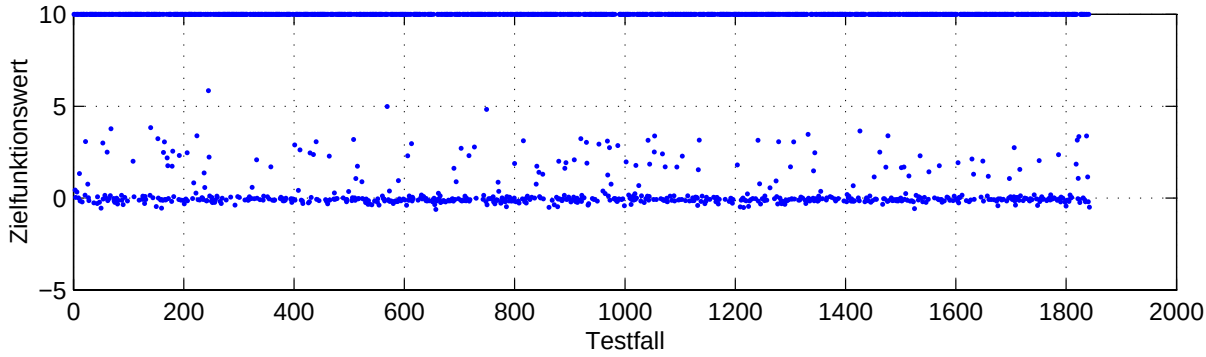


Abbildung 6.33: Zielfunktionswerte nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

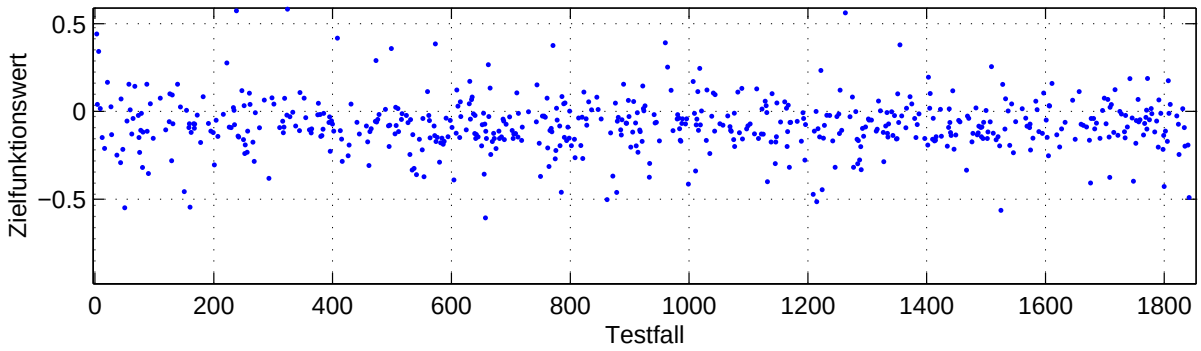


Abbildung 6.34: Ausschnittvergrößerung der Zielfunktionswerte nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

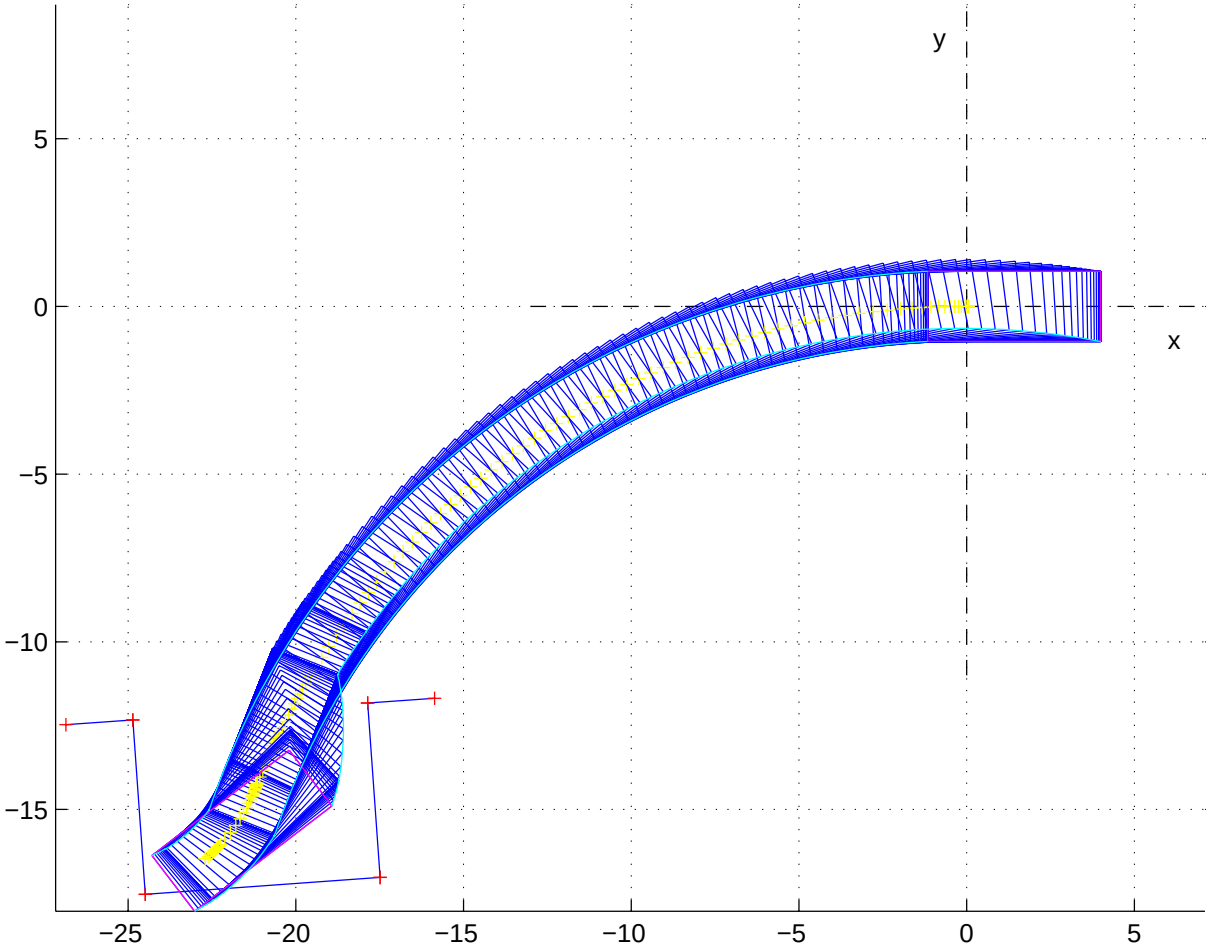


Abbildung 6.35: Bester gefundener Testfall der Zufallstests bewertet nach dem Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“

6.2 Der „Abstandsbasierte Bremsassistent“

In diesem Kapitel werden für den „Abstandsbasierten Bremsassistenten“ die experimentellen Ergebnisse der Tests vorgestellt. Kapitel 6.2.1 und Kapitel 6.2.2 enthalten die Ergebnisse des evolutionären Funktionstests mit und ohne Seeding. Anschließend werden in Kapitel 6.2.3 die Ergebnisse von manuellen Tests vorgestellt. Kapitel 6.2.4 betrachtet die Ergebnisse der Zufallstests. Der Vergleich der gewonnenen Ergebnisse wird in Kapitel 6.3 dargestellt.

Bei den im Folgenden vorgestellten Experimenten wurde die Anzahl der Testfälle auf ungefähr 600 Auffahrszenarien limitiert, um die Simulation über Nacht durchführen zu können.

6.2.1 Ergebnisse des evolutionären Funktionstests ohne Seeding

In Abbildung 6.36 ist der Verlauf des Zielfunktionswerts des besten Individuums einer Generation bei einem evolutionären Funktionstest ohne Seeding dargestellt. Bei diesem Testlauf besteht die initiale Population aus zufällig ausgewählten Individuen. Insgesamt wurden bei diesem Testlauf 623 Auffahrszenarien simuliert, was bei einer Generationslücke von 0.9 der Erzeugung von 20 Generationen entspricht. Der beste Zielfunktionswert in der ersten Generation liegt bei $-3.8 \cdot 10^3$. Man sieht, dass sich der Zielfunktionswert in Generation 14 deutlich verbessert.

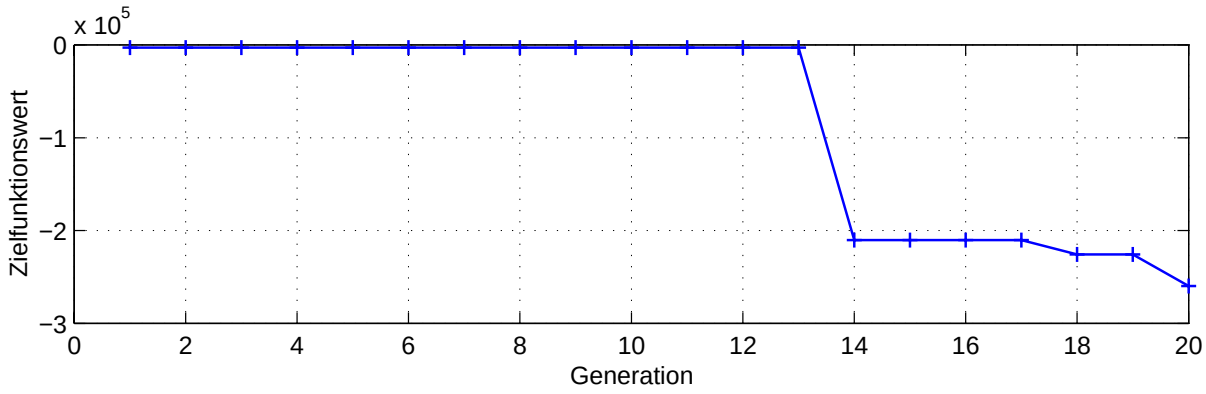


Abbildung 6.36: Jeweils bester Zielfunktionswert während des Testlaufs ohne Seeding

In Abbildung 6.37 ist die Verteilung der Zielfunktionswerte aller Individuen in der letzten Generation dargestellt. Es ist zu sehen, dass nur wenige Individuen den Bereich von -10^5 und dass die meisten Individuen den Strafwert von +100 erreicht haben. Ein Strafwert von +100 bedeutet hier, dass der „Abstandsbasierte Bremsassistent“ gar nicht aktiv wurde. Der Testlauf ohne Seeding erreicht nach 20 Generationen als Endergebnis einen besten Zielfunktionswert von $-2.59602 \cdot 10^5$.

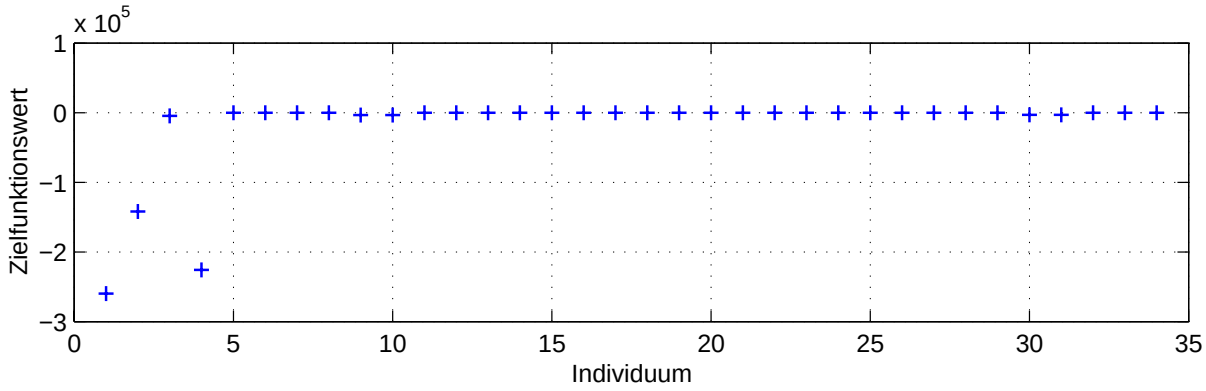


Abbildung 6.37: Zielfunktionswerte in Generation 20 beim Testlauf ohne Seeding

6.2.2 Ergebnisse des evolutionären Funktionstests mit Seeding

In Abbildung 6.38 ist der Verlauf bei einem evolutionären Funktionstest mit Seeding dargestellt. Die Abbildung zeigt wieder den Verlauf des Zielfunktionswerts des besten Individuums einer Generation über alle 20 Generationen. Insgesamt wurden bei diesem Testlauf 623 Auffahrszenarien simuliert.

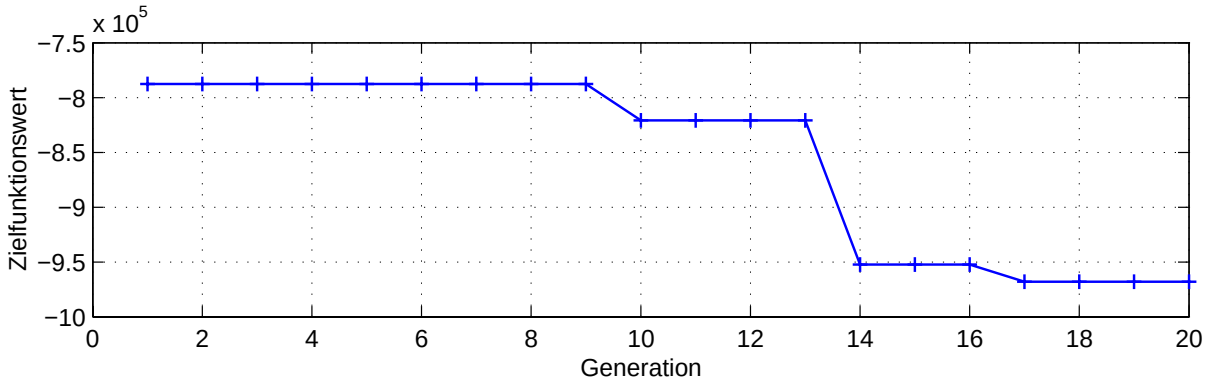


Abbildung 6.38: Jeweils bester Zielfunktionswert während des Testlaufs mit Seeding

Es ist zu sehen, dass die durch die manuell definierten Testfälle ausgewählten Individuen der initialen Population mit einem Zielfunktionswert von $-7.87445 \cdot 10^5$ starten. Dieser Startwert ist bereits besser als der beste Zielfunktionswert, der in 20 Generationen beim Testlauf ohne Seeding erreicht wurde. Bei diesem Testlauf mit Seeding wird in den 20 Generationen der Suche ein bester Zielfunktionswert von $-9.67808 \cdot 10^5$ erreicht. D.h. der evolutionäre Funktionstest konnte den besten Zielfunktionswert der manuell definierten Testfälle aus Generation 1 weiter verbessern.

In Abbildung 6.39 ist die Verteilung der Zielfunktionswerte aller Individuen beim Testlauf

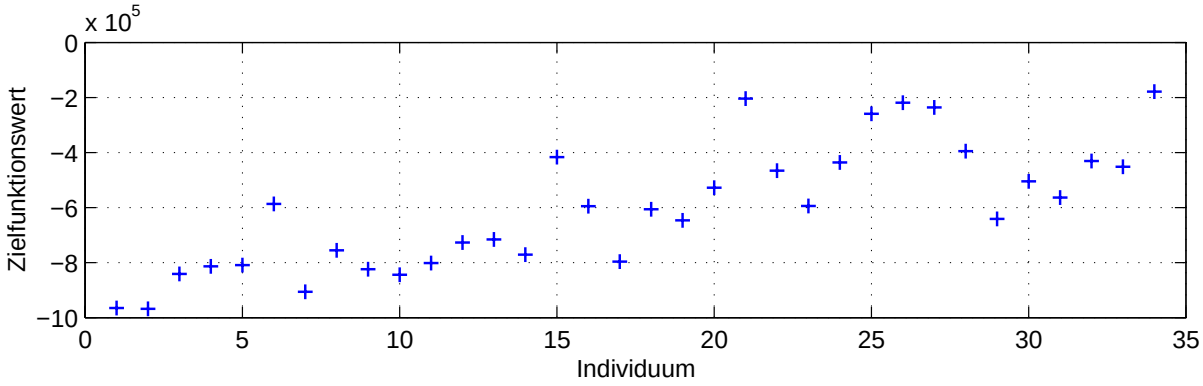


Abbildung 6.39: Zielfunktionswerte in Generation 20 beim Testlauf mit Seeding

mit Seeding in der letzten Generation dargestellt. Es ist zu sehen, dass sich die Zielfunktionswerte nahezu aller Individuen im Bereich zwischen $-2 \cdot 10^5$ und $-10 \cdot 10^5$ befinden.

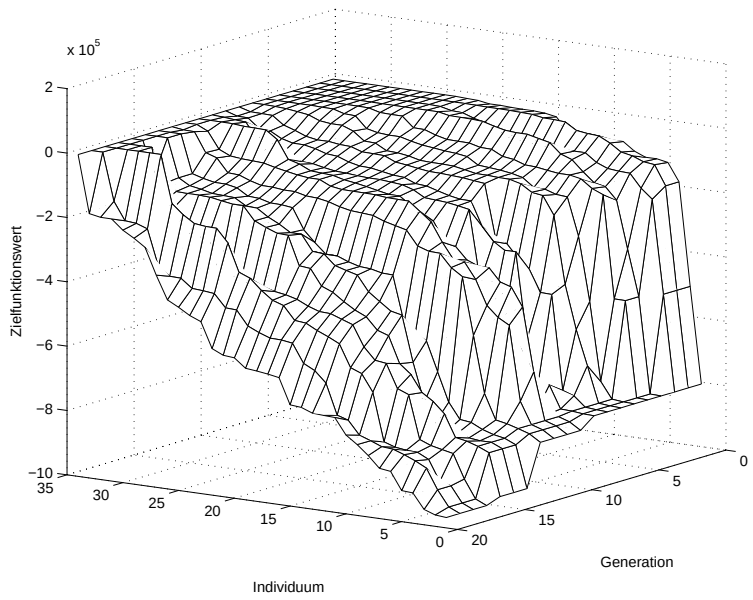


Abbildung 6.40: Zielfunktionswerte des Testlaufs mit Seeding

Die Abbildung 6.40 zeigt alle Zielfunktionswerte des Testlaufs mit Seeding über alle Generationen. Man sieht, dass bereits in Generation 1 mehrere Individuen einen Zielfunktionswert in der Größenordnung von $-1 \cdot 10^5$ oder besser haben. Im Verlauf der evolutionären Suche gelingt es, bis Generation 10 mehrere Individuen in die Größenordnung $-8 \cdot 10^5$ zu optimieren. In Generation 20 ist eine Verteilung der Zielfunktionswerte über den gesamten Wertebereich zu beobachten. Dies deutet darauf hin, dass auch bei diesem Testlauf mit dem

vorgegebenen Simulationsumfang keine Konvergenz der Population erreicht wurde.

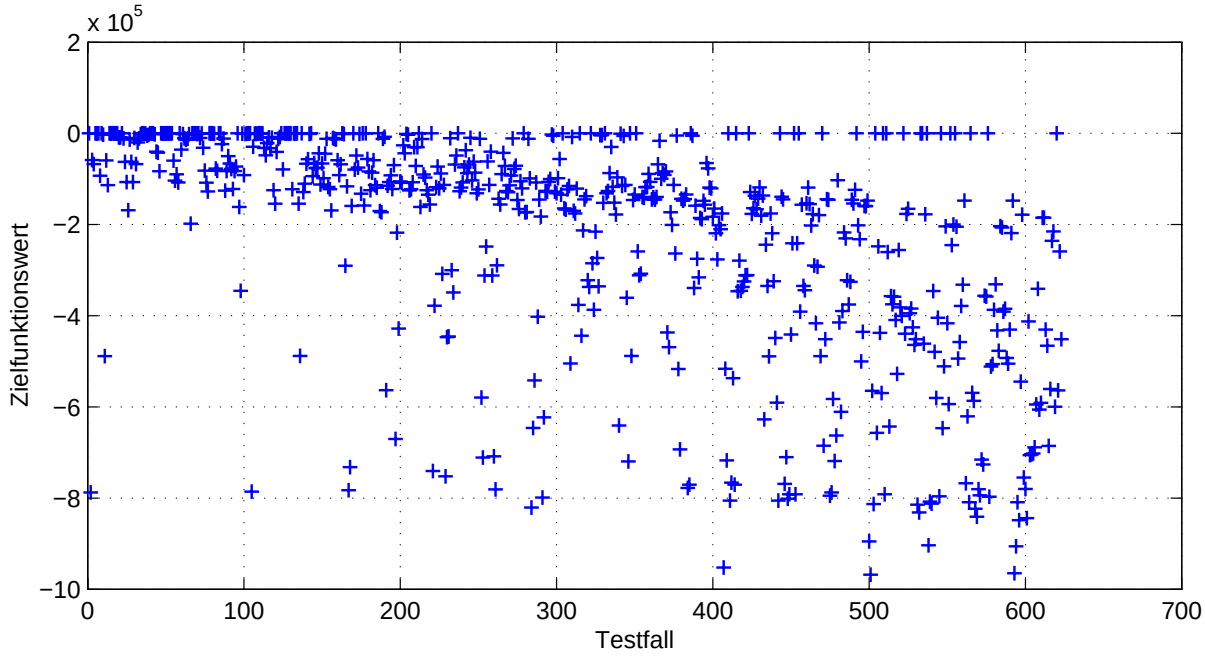


Abbildung 6.41: Zielfunktionswerte aller Testfälle

Beim Testlauf des evolutionären Funktionstests mit Seeding wurden 623 Testfälle durchgeführt. Die Zielfunktionswerte dieser Testfälle sind in Abbildung 6.41 dargestellt. Es ist zu sehen, dass die meisten der ausgeführten Testfälle des evolutionären Funktionstest einen Zielfunktionswert erreichen, der besser ist als -10^4. Der beste überhaupt erreichte Wert ist $-9.67808 \cdot 10^5$. Von den 623 ausgeführten Testfällen gab es 84, bei denen der „Abstandsbasierte Bremsassistent“ nicht ausgelöst hatte.

6.2.3 Ergebnisse der manuellen Tests

In diesem Kapitel werden die Ergebnisse des manuellen Tests dargestellt. Die Auswahl der Testfälle basiert auf dem im Projekt verwendeten Fahrmanöverkatalog. Dieser wurde mit der verwendeten Testumgebung abgebildet und simuliert.

In Abbildung 6.42 sind alle Zielfunktionswerte der manuell definierten Testfälle dargestellt. Bei den manuell definierten Testfällen wurde durch Individuum 2 der beste Zielfunktionswert von $-7.87445 \cdot 10^5$ erreicht.

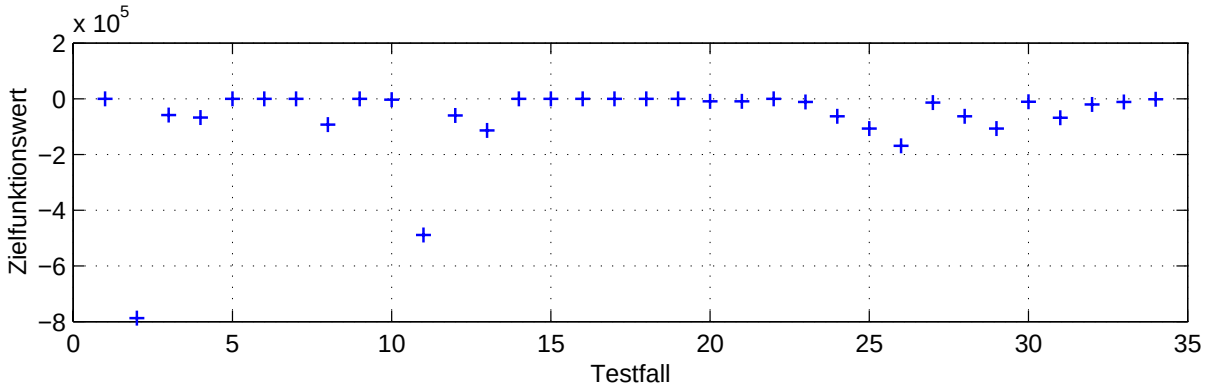


Abbildung 6.42: Zielfunktionswerte der manuell definierten Testfälle

6.2.4 Ergebnisse der Zufallstests

Wie bei den bisherigen Testläufen des evolutionären Funktionstests wurden auch beim Zufallstest 623 Testfälle durchgeführt, um eine Vergleichbarkeit zu ermöglichen.

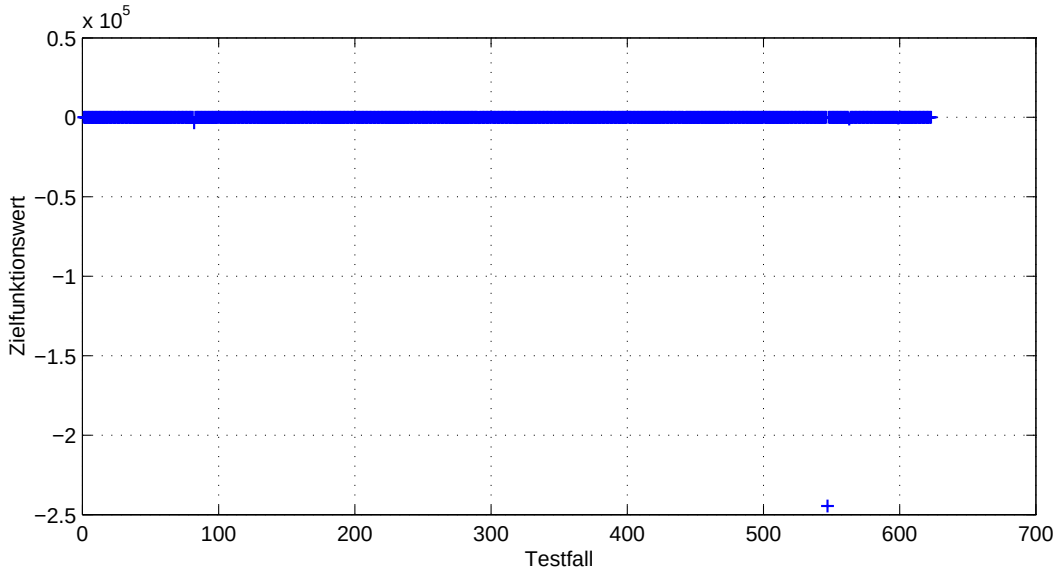


Abbildung 6.43: Zielfunktionswerte aller Testfälle der zufälligen Testfallauswahl

In Abbildung 6.43 sind die Zielfunktionswerte der 623 zufällig ausgewählten Testfälle dargestellt. Von den 623 ausgeführten Testfällen gibt es 616, bei denen der „Abstandsbasierte Bremsassistent“ nicht ausgelöst hatte. Nur sieben Szenarien führten zu einer Aktivierung des „Abstandsbasierten Bremsassistenten“, davon wurde nur ein Szenario mit einem besseren Zielfunktionswert als -10^5 bewertet. Da der Zufallstest stochastisch ist, ist es nicht

verwunderlich, dass nur der beste zufällig ausgewählte Testfall zu einem Zielfunktionswert von $-2.44604 \cdot 10^5$ führt.

6.3 Bewertung

In diesem Kapitel werden die Ergebnisse der Experimente für die beiden Anwendungen „Automatisches Parksystem“ und „Abstandsbasierter Bremsassistent“ analysiert und interpretiert. Hierbei werden die Ergebnisse des evolutionären Funktionstests mit den Ergebnissen einer manuellen Auswahl von Testfällen und einer zufälligen Auswahl von Testdaten verglichen. Dabei ist zu beachten, dass die Bewertung der Ergebnisse des Zufallstests ohne die Verwendung einer automatischen Bewertungsfunktion wie z.B. der Zielfunktion des evolutionären Funktionstests in der Praxis nicht durchführbar ist, da sehr viele Testergebnisse manuell ausgewertet werden müssen.

Im Falle des evolutionären Funktionstests wurden Testläufe mit und ohne Seeding durchgeführt. Dabei wurde nur die initiale Population geändert. Das Testsystem selbst war vollkommen identisch. Im Falle der Zufallstests und der manuellen Tests wurde im Testsystem lediglich die Komponente „Evolutionärer Algorithmus“ ersetzt durch eine manuelle Vorgabe der Individuen durch einen Testingenieur bzw. durch einen Zufallsgenerator.

Wie bereits in Kapitel 6.1 dargestellt, wurden für die Anwendung „Automatisches Parksystem“ zwei Fehlersituationen „Kollision am Punkt P_4“ und „Kollision an der Strecke $\overline{P_2P_3}$“ identifiziert und dafür Zielfunktionen implementiert. Anschließend wurde je ein Testlauf mit manuellen Tests und mit Zufallstests durchgeführt und die Testergebnisse mit diesen beiden Zielfunktionen bewertet. Ferner wurden evolutionäre Funktionstests mit beiden Zielfunktionen durchgeführt. Dabei wurden Testläufe ohne Seeding und mit zwei verschiedenen Seeding-Varianten[1] getestet. In Tabelle 6.1 sind die jeweils besten Zielfunktionswerte dieser Testläufe dargestellt.

Optimierungsziel	Evolutionärer Funktionstest ohne Seeding	Evolutionärer Funktionstest mit Seeding	Evolutionärer Funktionstest mit Seeding 2	Manueller Test	Zufallstest
"Kollision am Punkt P_4"	-0.08109	-0.10636	-0.09596	-0.01529	-0.04131
"Kollision an der Strecke $\overline{P_2P_3}$"	-0.87584	-0.68356	-1.89713	-0.27557	-0.60649

Tabelle 6.1: Ergebnisse des „Automatischen Parksystems“

Bei beiden Optimierungszielen wurden die besten Zielfunktionswerte durch den evolutionären Funktionstest erreicht. Beim Optimierungsziel „Kollision am Punkt P_4“ wurde als bester Zielfunktionswert ein Wert von -0.10636 beim Testlauf „Evolutionärer Funktionstest

[1] Die beiden Seeding-Varianten wurden in Kapitel 6.1 vorgestellt.

mit Seeding“ erreicht. Der zweitbeste Wert von -0.09596 ergab sich beim Testlauf „Evolutionärer Funktionstest mit Seeding 2“.

Beim anderen Optimierungsziel „Kollision an der Strecke $\overline{P_2P_3}$“ wurde der beste Zielfunktionswert durch den Testlauf „Evolutionärer Funktionstest mit Seeding 2“ und der zweitbeste Zielfunktionswert beim Testlauf „Evolutionärer Funktionstest ohne Seeding“ gefunden. Bemerkenswert ist hierbei, dass nach der gleichen Anzahl von Testfällen die Ergebnisse aus dem Testlauf ohne Seeding besser sind als die Ergebnisse aus dem Testlauf mit Seeding. Dies ist darauf zurückzuführen, dass zufällig eine günstige Anfangspopulation ausgewählt wurde.

Bei den Testläufen des „Abstandsbasierten Bremsassistenten“ wurde die Simulationszeit auf 12 Stunden begrenzt. Damit war es möglich, 623 Auffahr-Szenarien zu simulieren. Dies entspricht 20 Generationen beim evolutionären Funktionstest.

In Tabelle 6.2 sind die jeweils besten Zielfunktionswerte des „Abstandsbasierten Bremsassistenten“ dargestellt. Der beste Zielfunktionswert hier ist $-9.67808 \cdot 10^5$. Dieser wurde bei einem evolutionären Funktionstest mit Seeding gefunden. Bemerkenswert bei diesem Testlauf ist, dass sich der zweitbeste Zielfunktionswert von $-7.87445 \cdot 10^5$ beim manuellen Test ergab. Der evolutionäre Funktionstest ohne Seeding konnte hier nur einen Wert von $-2.59602 \cdot 10^5$ erreichen. Dies ist darauf zurückzuführen, dass die manuelle Auswahl der Testfälle hier eine sehr gute Ausgangsbasis für die Suche bietet. Der evolutionäre Funktionstest ohne Seeding konnte in den vorgegebenen 20 Generationen der Suche nicht das Niveau der manuellen Testfallauswahl erreichen.

Optimierungsziel	Evolutionärer Funktionstest ohne Seeding	Evolutionärer Funktionstest mit Seeding	Manueller Test	Zufallstest
"Hohe Bremsunterstützung bei unkritischer Situation"	-2.59602 · 10^5	-9.67808 · 10^5	-7.87445 · 10^5	-2.44604 · 10^5

Tabelle 6.2: Ergebnisse des „Abstandsbasierten Bremsassistenten“

Prinzipiell kann ein evolutionärer Funktionstest ohne Seeding eine ungünstige Ausgangsbasis durch eine längere Laufzeit der Suche ausgleichen. Dies bedeutet, ohne Seeding lassen sich prinzipiell gleich gute Ergebnisse erreichen wie mit Seeding, wenn man die Randbedingung „Laufzeit“ nicht betrachtet. Dies wurde anhand eines weiteren Experiments in Bühler et al. [96] dargestellt.

Beim manuellen Test wird bei der Wahl der Testfälle Wissen über die Anwendung benutzt. Lässt man einen evolutionären Algorithmus ohne Seeding laufen, so startet dieser von einer rein zufällig generierten Anfangspopulation, in welcher kein Anwendungswissen enthalten ist. Wie man den Tabellen 6.1 und 6.2 entnehmen kann, ist es für die Laufzeit von Vorteil, wenn man die Anfangspopulation mit Seeding festlegt. Damit enthält die Anfangspopulation schon Wissen über die Anwendung, da sie manuell mit dieser Kenntnis gewählt wurde. Es

werden daher weniger Iterationen benötigt, um einen vergleichbaren Zielfunktionswert zu erreichen.

Bei allen durchgeführten Experimenten erreichte der evolutionäre Funktionstest bei vergleichbarer Anzahl an durchgeführten Tests einen besseren Zielfunktionswert als der Zufallstest. Dies wird durch den evolutionären Algorithmus bewirkt, welcher die Individuen im Suchraum so generiert, dass sie einen guten Zielfunktionswert haben. Beim Zufallstest und beim manuellen Test gibt es eine solche Führung nicht. Der evolutionäre Funktionstest mit Seeding kombiniert das Wissen über die Anwendung mit einer geführten Suche und kann damit die besten Ergebnisse erzielen.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

In der vorliegenden Arbeit wurde gezeigt, dass das Verfahren des evolutionären Testens [56] in allgemeiner Weise auf einen funktionalen Test erweitert werden kann, so dass es möglich ist, einen funktionalen Test voll automatisiert durchzuführen. Für den Funktionstest mit evolutionären Algorithmen – in der vorliegenden Arbeit als „evolutionärer Funktionstest“ bezeichnet – wurde eine Architektur (siehe Kapitel 4.1) bestehend aus den Komponenten „evolutionärer Algorithmus“, „Codierung der Testdaten“, „Zielfunktion“ und „Testumgebung“ entwickelt.

Die Komponente „evolutionärer Algorithmus“ ist ein auf dem Markt verfügbares System. Als Testumgebung kann die im Rahmen des Standard-Entwicklungsprozesses hergestellte Testumgebung für das Testobjekt verwendet werden. Die Bereitstellung dieser beiden Komponenten verursacht damit keinen zusätzlichen Entwicklungsaufwand. Die entwickelten Komponenten „Codierung der Testdaten“ und „Zielfunktion“ stellen eine Verknüpfung zwischen den evolutionären Algorithmen und der Begriffswelt des funktionalen Tests her. Für beide Komponenten wurden in der vorliegenden Arbeit systematische Herleitungsstrategien entwickelt. So wurde in Kapitel 4.2 gezeigt, wie aus Ereignissen aus dem Bereich der Realzeitsysteme die Gene von Individuen, auf denen ein evolutionärer Algorithmus arbeiten kann, gewonnen werden können.

Das Finden der Zielfunktion, welche die Bewertung der Testergebnisse vornimmt (siehe Kapitel 4.3), erfolgt in den Schritten „Festlegung des Optimierungsziels“, „Definition der Ergebnisklassen“, „Identifikation charakteristischer Merkmale“ und „Formulieren der Zielfunktion“.

Die vorgeschlagene Methodik zur Codierung der Testdaten und zum Entwurf einer Zielfunktion wurde in zwei verschiedenen Anwendungen aus der Welt der abstandsbasierten Fahrerassistenzsysteme überprüft. Dabei handelt es sich um das „Automatische Parksystem“ (siehe Kapitel 5.1) und um den „Abstandsbasierten Bremsassistenten“ (siehe Kapitel 5.3).

Bei beiden Anwendungen konvergierte der evolutionäre Algorithmus rasch und konnte ein fehlerhaftes Verhalten einer Anwendungsfunktion finden. Die Verwendung eines evolutionären Algorithmus erwies sich als zielführend, da er in einem unstetigen Suchraum ein robustes Optimierungsverfahren darstellt.

Die Arbeit hat gezeigt, dass es möglich ist, einen evolutionären Funktionstest in der industriellen Praxis umzusetzen. Bei zwei unterschiedlichen Anwendungsfunktionen wurde für jeweils eine ausgewählte Anforderung ein Optimierungsziel definiert und entsprechend diesem Optimierungsziel eine geeignete Zielfunktion konstruiert. Für beide Anwendungen wurden Testsysteme aufgebaut und die entwickelten Zielfunktionen implementiert und integriert. Darüber hinaus wurden geeignete Modelle für beide Anwendungen entwickelt, mit denen ein Suchraum auf funktional sinnvolle Testeingaben für die Testumgebung abgebildet werden kann. Die aufgebauten Testsysteme fanden die Fehler in den gewählten Anwendungsfunktionen entsprechend dem gewählten Optimierungsziel automatisch.

Die Zielfunktion bewertet ein Testergebnis nur entsprechend dem gewählten speziellen Optimierungsziel. Für gewöhnlich wird das Optimierungsziel aus der Verletzung einer Anforderung abgeleitet. Diese Verletzung einer einzigen Anforderung stellt den gesuchten Fehler dar. Es ist legitim, für den Test ein einzelnes Optimierungsziel zu betrachten, denn aus Sicht des Testers ist es hinreichend zu zeigen, dass eine einzelne Anforderung nicht eingehalten wird. Es ist nicht notwendig, die Verletzung aller Anforderungen der Anwendungsfunktion gleichzeitig zu betrachten. Der Fokus auf ein einzelnes Optimierungsziel reduziert jedoch die Komplexität der Zielfunktion.

Da durch den evolutionären Funktionstest ein hoher Grad an Automatisierung erreicht wird, kann das zu testende System mit einer sehr großen Anzahl an Tests getestet werden. Die Testdaten werden automatisch generiert und verfolgen das ausgewählte Optimierungsziel. Bei den dargestellten Anwendungen war es möglich, autonom mehrere tausend Tests zu generieren und auszuführen, vorwiegend in Testläufen, die über Nacht oder am Wochenende durchliefen. Aufgrund seiner Effizienz kann der evolutionäre Funktionstest einen wichtigen Beitrag zur Verbesserung der Software-Qualität leisten und gleichzeitig das Aufwand-Nutzen-Verhältnis beim Testen erheblich verbessern.

Bei manuellen Testmethoden muss für jeden Testfall manuell das zu erwartende Ergebnis bereitgestellt werden. Beim evolutionären Funktionstest ist der Aufwand für den Entwurf eines Modells zur Codierung der Testdaten und einer geeigneten Zielfunktion ein fixer Kostenanteil. Stehen diese Komponenten erst einmal zur Verfügung, ist die Anzahl der möglichen Tests in erster Linie abhängig von der verfügbaren Rechnerkapazität. Mit dem evolutionären Funktionstest gelingt es somit, die Anzahl der Testfälle mit Hilfe von Computer-Performance erheblich zu steigern.

7.2 Ausblick

Weitere wissenschaftliche Arbeiten könnten sich mit dem Zusammenhang zwischen funktionalen und strukturorientierten Tests beschäftigen. Diese Arbeiten könnten beispielsweise Fragen beantworten wie: „Welche Überdeckung erreicht ein evolutionärer Funktionstest?“ oder „Ist es sinnvoll, für die Anfangsverteilung eines evolutionären Funktionstests die durch einen evolutionären Strukturtest ermittelten Testfälle zu verwenden?“ Ein weiteres Forschungsziel könnte zum Inhalt haben, die in dieser Arbeit vorgestellte Methodik zur Herleitung einer Codierung von Testdaten für Systemtests von eingebetteten Systemen noch weiter zu generalisieren. Im Bereich des Optimierers wäre es sinnvoll, zu untersuchen, ob evolutionäre Algorithmen mit sehr großen Populationsgrößen und einer kleinen Generationslücke eine bessere Performance bieten, da sich dann mehr Individuen im Suchraum aufhalten und somit das Problemwissen der Suche besser erhalten bleibt.

Das hier für die beiden Anwendungen „Automatisches Parksystem“ und „Abstandsbasierter Bremsassistent“ vorgestellte Verfahren des evolutionären Funktionstests könnte auch auf weitere Anwendungsfunktionen speziell im Umfeld der Fahrerassistenzsyteme angewandt werden. So ist es vorstellbar, einen Abstandsregeltempomaten ACC oder automatische Notbremssysteme nach diesem Verfahren zu testen. Damit können die Vorteile einer automatischen Generierung der Testdaten und einer automatischen Bewertung der Testergebnisse auch bei diesen Systemen zu einem effizienteren funktionalen Test führen.

Der Schwerpunkt der vorliegenden Arbeit lag im Bereich des Testens von abstandsbasierten Fahrerassistenzsystemen. Die vorgeschlagene Methode ist jedoch so allgemein, dass eine Übertragung auf alle möglichen Anwendungen von eingebetteten Systemen untersucht werden kann.

Anhang A

Begriffsverzeichnis

ACC-Sensor

Das Akronym ACC steht für Adaptive Cruise Control. Ein ACC-Sensor ist für die Detektion vorausfahrender Fahrzeuge zuständig. Dieser Sensor liefert den Abstand und die Relativgeschwindigkeit zu vorausfahrenden Fahrzeugen.

Äquivalenzklasse

Eine Äquivalenzklasse ist ein Anteil des Eingabe- oder Ausgabebereichs, für welches das Verhalten einer Komponente oder eines Systems, basierend auf der Spezifikation, als dasselbe angenommen wird. (Nach Veenendaal in [104], siehe Definition 2.13)

Anweisungsüberdeckungsgrad

Der Anweisungsüberdeckungsgrad C_0 ist definiert als das mathematische Verhältnis zwischen der Anzahl der Elemente in der Menge der ausgeführten Anweisungen und der Anzahl der Elemente in der Menge der Anweisungen des Programms. (Siehe Definition 2.8)

Anwendungsfunktion

Im allgemeinen ist eine Anwendungsfunktion eine Leistung eines Systems und wird auch als „Use Case“ bezeichnet. In dieser Arbeit ist eine Anwendungsfunktion eine vom Kunden erlebbare Funktionalität in einem Fahrzeug. Innovative Anwendungsfunktionen werden vom Hersteller oft als Sonderausstattung für ein Fahrzeug angeboten.

Atomare Teilbedingung

Eine atomare Teilbedingung ist eine Bedingung, die keine logischen Operatoren wie AND, OR oder NOT, sondern höchstens Relationssymbole wie '>' oder '=' enthält. (Nach Spillner et al. in [69], siehe Definition 2.10)

Back-to-Back-Test

Beim Back-to-Back-Test werden zwei oder mehr Varianten einer Komponente oder eines Systems mit den gleichen Eingaben ausgeführt und deren Ergebnisse dann verglichen. Im Fall von Abweichungen wird die Ursache analysiert. (Nach IEEE 610 in [19], siehe Definition 2.15)

Blackbox-Verfahren

Blackbox-Verfahren sind Testmethoden zur Ableitung und/oder Auswahl von Testfällen basierend auf einer Analyse der Spezifikation, entweder funktional oder nicht-funktional, von einer Komponente oder eines Systems ohne die Einsichtnahme der inneren Struktur (des Testobjekts). (Nach Veenendaal in [104], siehe Definition 2.11)

Closed-Loop-Test

Bei einem Closed-Loop-Test sind die Ausgänge des Testobjekts über eine Wirkungskette (Regelstrecke) zu den Eingängen zurückverbunden und beeinflussen diese während der Testdurchführung.

Codierung der Testdaten

Die „Codierung der Testdaten“ ist eine Komponente des Testsystems für einen evolutionären Funktionstest, welche die Gene eines Individuums in reale Testdaten übersetzt.

Dynamisches Testen

Dynamisches Testen ist der Prozess, ein Programm (oder eine Funktionalität) mit der Absicht auszuführen, Fehler zu finden. (Nach Myers in [26], siehe Definition 2.3)

E/E/PE-Systeme

Nach DIN EN 61508-4 [12] ist ein E/E/PE-System ein System zur Steuerung, zum Schutz oder zur Überwachung, basierend auf einem oder mehreren elektrischen/elektronischen/programmierbaren elektronischen Geräten, einschließlich aller Elemente des Systems wie z.B. Energieversorgung, Sensoren und anderen Eingabegeräten, Datenverbindungen und anderen Kommunikationswegen sowie Aktoren und anderen Ausgabeeinrichtungen.

Eingebettete Software

Als eingebettete Software bezeichnet man nach Conrad [10] die in ein eingebettetes System integrierte Software.

Eingebettetes System

Als eingebettete Systeme (engl.: embedded systems) bezeichnet man im Allgemeinen Teilsysteme, die in größere Systeme oder Umgebungen integriert sind und in der Regel aus Hardware und Software bestehen. [30]

Fahrzeugumgebung

Als Fahrzeugumgebung wird die Umgebung des Systemfahrzeugs bezeichnet. Die Fahrzeugumgebung kann z.B. umgebende Fahrzeuge oder andere Objekte enthalten.

Funktionsorientierter Test bzw. funktionaler Test

Ein funktionsorientierter Test basiert auf einer Analyse der funktionalen Spezifikation einer Komponente oder eines Systems. (Nach Veenendaal in [104], siehe Definition 2.12)

Hardware-in-Loop (HiL)

Bei einem Hardware-in-Loop Prüfstand wird ein reales Steuergerät in einem dafür vorgesehenen Prüfstand getestet, wobei der nicht real vorhandene Teil des Fahrzeugs, die Fahrzeugumgebung, wie z.B. die umgebenden Fahrzeuge, und der Fahrer jeweils durch eine eigene Simulation ersetzt werden. Die Simulation ermöglicht es, die Interaktion der Einzelsysteme systematisch zu untersuchen, abzustimmen und die entstehende Variantenvielfalt zu beherrschen [57]. Nach Athanasas [3] ist eine geeignete Simulation dabei eine der schwierigsten Aufgaben beim Aufbau eines solchen Prüfstands.

Model-in-Loop (MiL)

Eine Software-Komponente für ein Steuergerät kann zunächst als Modell realisiert werden. Ein solches Modell kann in einer Simulation auf dem PC in einer Testumgebung ausgeführt werden. Dabei werden alle Teile, mit denen dieses Modell interagiert, einschließlich Fahrzeug, Fahrzeugumgebung und Fahrer in der Model-in-Loop Umgebung simuliert. Eine solche Testumgebung nennt man Model-in-Loop Umgebung.

Mutationen-Test

Der Mutationen-Test ist ein Back-to-Back-Test, bei dem die Varianten einer Komponente oder eines Systems durch künstliches Einfügen kleiner, definierter Modifikationen in die Originalversion der Software erzeugt werden. (In Anlehnung an Liggesmeyer in [23], siehe Definition 2.17)

Open-Loop-Test

Bei einem Open-Loop-Test werden die Eingänge des Testobjekts bei der Testdurchführung nicht über eine Wirkungskette von den Ausgängen des Testobjekts beeinflusst.

Operationelles Profil

Ein operationelles Profil gibt an, welche Eingabedaten im realen Betrieb wie häufig auftreten. (Nach Liggesmeyer in [23], siehe Definition 2.20)

Regressionstest

Ein Regressionstest ist ein erneuter Test eines bereits getesteten Programms nach dessen Modifikation mit dem Ziel, festzustellen, dass durch die vorgenommene Änderung keine Fehler hinzugekommen sind oder (bisher maskierte) Fehler in unveränderten Teilen der Software freigelegt wurden. (Nach Veenendaal in [104], siehe Definition 2.16)

Szenario

Ein Szenario ist das Ergebnis einer Simulation. Ein Szenario stellt damit Menge aller in einer Simulation vorkommenden Größen dar.

Software-in-Loop (SiL)

Bei einer Software-in-Loop Testumgebung läuft die zu testende Software-Komponente zunächst auf einem PC statt im Steuergerät. Alle Teile, mit denen die Software-Komponente interagiert, einschließlich Fahrzeug, Fahrzeugumgebung und Fahrer werden bei einer Software-in-Loop Testumgebung simuliert.

Statisches Testen

Statisches Testen ist das Testen einer Komponente oder eines Systems auf Ebene der Spezifikation oder der Implementierung ohne Ausführung der Software, z.B. durch Reviews oder statische Codeanalyse. (Nach Veenendaal in [104], siehe Definition 2.2)

Statistischer Test

Der statistische Test ist eine Testmethode, bei der ein Modell von der statistischen Verteilung der Eingabedaten verwendet wird, um repräsentative Testeingabedaten zu generieren. (Nach Veenendaal in [104], siehe Definition 2.19)

Strafwert

Ein „Strafwert“ (engl. Penalty) ist eine Bewertungszahl, welche die Zielfunktion zurückliefert, wenn während der Suche ein Punkt im Suchraum erreicht wurde, der für das Optimierungsziel nicht relevant ist. In der vorliegenden Arbeit liefert eine Zielfunktion einen Strafwert zurück, wenn in einem Szenario die zu bewertende Anwendungsfunktion nicht aktiv wurde.

Systemfahrzeug

Als Systemfahrzeug wird das Fahrzeug mit dem Assistenzsystem bezeichnet.

Testfall

Nach Veenendaal in [104] besteht ein Testfall aus einer Menge von Eingangsdaten, Vorbedingungen für die Ausführung, erwarteten Ergebnissen und Nachbedingungen der Ausführung, die für ein spezifisches Ziel oder eine spezifische Bedingung entworfen wurden, um einen spezifischen Programmpfad auszuführen oder die Übereinstimmung mit einer spezifischen Anforderung zu überprüfen. (Nach Veenendaal in [104], siehe Definition 2.5)

Bei einem evolutionären Funktionstest ist das Sollergebnis nicht bekannt. Daher besteht ein Testfall hier nur aus Eingangsdaten und Vorbedingungen. Die Eingangsdaten und Vorbedingungen werden in Form von Testdaten an die Testumgebung übergeben. Ein Satz von Testdaten entspricht der Information eines Individuums.

Testmethode

Unter einer Testmethode versteht man ein planmäßiges, auf einem Regelwerk aufbauendes Vorgehen zum Finden von Testfällen. (Nach Spillner et al. in [69], siehe Definition 2.1)

Testobjekt

Ein Testobjekt ist die zu testende Komponente oder das zu testende System. (Nach Veenendaal in [104], siehe Definition 2.4)

Testorakel

Ein Testorakel ist eine Informationsquelle zur Ermittlung der jeweiligen Sollergebnisse eines Testfalls. (Nach Spillner et al. [69], siehe Definition 2.14)

Testsystem

Ein Testsystem für einen evolutionären Funktionstest besteht aus den Komponenten „evolutionärer Algorithmus“, „Codierung der Testdaten“, „Zielfunktion“ und „Testumgebung“.

Testumgebung

Als Testumgebung wird in der vorliegenden Arbeit eine Komponente des Testsystems für einen evolutionären Funktionstest bezeichnet, die aus der eingebetteten Software und einer Simulationsumgebung, in Form von Fahrer- und Fahrzeugsimulation sowie einer Simulation der Fahrzeugumgebung, besteht.

Vollständiger Test

Ein vollständiger Test ist ein Testansatz, bei dem die Menge der Testfälle alle Kombinationen von Eingabewerten und Vorbedingungen umfasst. (Nach Veenendaal in [104], siehe Definition 2.6)

Whitebox-Verfahren

Whitebox-Verfahren sind Testmethoden, die zur Herleitung oder Auswahl der Testfälle Informationen über die innere Struktur des Testobjekts benötigen. (Nach Spillner et al. in [69], siehe Definition 2.7)

Zielfahrzeug

Als Zielfahrzeug wird ein Fahrzeug bezeichnet, das sich in der Fahrzeugumgebung befindet und durch die umgebungserfassende Sensorik erfasst wird.

Zielfunktion

Der Begriff der Zielfunktion wird in der vorliegenden Arbeit im Sinne einer Bewertungsfunktion verwendet. Die Zielfunktion generiert ein Maß für den Grad der Verletzung einer Anforderung.

Zufallstest

Beim Zufallstest werden die Testeingabedaten zufallsgesteuert generiert. (Nach Spillner et al. in [69], siehe Definition 2.18)

Zweigüberdeckungsgrad

Der Zweigüberdeckungsgrad C_1 ist definiert als Verhältnis der Anzahl der Elemente in der Menge der durchlaufenen Zweige und der Anzahl der Elemente in der Menge der Zweige des Programms. (Siehe Definition 2.9)

Anhang B

Verlauf einer Optimierung

Dieses Kapitel stellt den Verlauf der Optimierung eines Testlaufs beim „Automatischen Parksystem“ dar. Dabei handelt es sich um den evolutionären Funktionstest ohne Seeding mit dem Optimierungsziel „Kollision am Punkt P_4“. Die Diskussion dieses Testlaufs befindet sich in Kapitel 6.1.1. In Kapitel 5.2 wird die Berechnung der Generation 11 ausführlich vorgestellt.

B.1 Generation 1 bis 20

Initiale Population in Generation 1

$$\underset{\text{Population}}{\begin{pmatrix}
1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\
1.68160 & 4.54360 & 4.41830 & 7.06500 & -6.92790 \\
13.51290 & 6.99210 & 7.27510 & 9.56770 & 1.09680 \\
2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\
5.09540 & 8.65600 & 2.32350 & 16.09740 & 8.16800 \\
4.63790 & 2.39310 & 0.49750 & 1.56770 & 2.81630 \\
3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\
8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\
11.83050 & 1.19750 & 0.38130 & 9.17200 & 7.39730 \\
18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\
12.91660 & 9.66890 & 6.64930 & 17.40760 & -9.80150 \\
2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\
13.74650 & 3.46110 & 1.66030 & 3.11230 & -6.17770 \\
8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\
9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\
5.94810 & 0.49160 & 6.93180 & 13.00210 & 9.65980 \\
11.05350 & 4.00070 & 1.98790 & 12.50400 & 4.66730 \\
7.51770 & 0.09880 & 4.19860 & 15.07340 & 5.87740 \\
18.39910 & 8.44720 & 3.67750 & 12.41600 & 4.62550 \\
3.87790 & 9.04810 & 5.69210 & 12.63580 & -5.31170 \\
10.97560 & 9.31580 & 3.35200 & 13.11060 & -2.16190 \\
12.54630 & 6.99080 & 3.97180 & 8.27260 & 3.10430 \\
16.75170 & 3.71610 & 4.25250 & 11.89330 & 1.31480 \\
6.13320 & 3.29410 & 4.50720 & 5.33060 & 7.17670 \\
18.02290 & 3.41410 & 0.00000 & 20.00000 & 10.00000
\end{pmatrix}}
\begin{pmatrix}
\vec{z}_{\text{Pop}} \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
0.41194 \\
100.00000 \\
0.93412 \\
100.00000 \\
100.00000 \\
1.58963 \\
100.00000 \\
0.37197 \\
1.24840 \\
100.00000 \\
0.87875 \\
0.30173 \\
100.00000 \\
100.00000 \\
0.59218 \\
1.56502 \\
0.32640 \\
100.00000
\end{pmatrix}$$

Generation 2

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 1.68160 & 4.54360 & 4.41830 & 7.06500 & -6.92790 \\ 13.51290 & 6.99210 & 7.27510 & 9.56770 & 1.09680 \\ 5.09540 & 8.65600 & 2.32350 & 16.09740 & 8.16800 \\ 18.02290 & 3.41410 & 0.00000 & 20.00000 & 10.00000 \\ 3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\ 4.63790 & 2.39310 & 0.49750 & 1.56770 & 2.81630 \\ 11.83050 & 1.19750 & 0.38130 & 9.17200 & 7.39730 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 12.91660 & 9.66890 & 6.64930 & 17.40760 & -9.80150 \\ 5.94810 & 0.49160 & 6.93180 & 13.00210 & 9.65980 \\ 13.74650 & 3.46110 & 1.66030 & 3.11230 & -6.17770 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 3.87790 & 9.04810 & 5.69210 & 12.63580 & -5.31170 \\ 11.05350 & 4.00070 & 1.98790 & 12.50400 & 4.66730 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 18.39910 & 8.44720 & 3.67750 & 12.41600 & 4.62550 \\ 7.51770 & 0.09880 & 4.19860 & 15.07340 & 5.87740 \\ 10.97560 & 9.31580 & 3.35200 & 13.11060 & -2.16190 \\ 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 16.75170 & 3.71610 & 4.25250 & 11.89330 & 1.31480 \\ 7.51770 & 0.09880 & 4.19860 & 15.07340 & 5.87740 \\ 18.02290 & 3.41410 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 1.18866 & 4.44159 & 4.29928 & 6.96073 & -7.26224 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 5.79540 & 8.37216 & 2.19769 & 16.30872 & 8.26720 \\ 18.27282 & 3.31276 & -0.04492 & 20.07545 & 10.03542 \\ 3.85714 & 8.14801 & 1.67931 & 3.32703 & 9.54598 \\ 4.25875 & 5.18781 & 1.07141 & 2.42207 & 6.08437 \\ 9.61329 & 6.02488 & 3.34580 & 13.85801 & 2.03252 \\ 11.64800 & 1.59485 & 0.62531 & 9.55771 & 6.95571 \\ 11.81361 & 8.21630 & 6.69401 & 16.71029 & -6.72113 \\ 7.63304 & 2.71061 & 6.86349 & 14.06732 & 4.95418 \\ 13.21576 & 3.68902 & 1.78767 & 3.82087 & -5.65325 \\ 2.66120 & 8.22153 & 4.32066 & 17.91187 & 4.77626 \\ 10.34368 & 3.47564 & 3.76569 & 19.25766 & -10.92760 \\ 10.30917 & 3.50538 & 3.77597 & 19.22233 & -10.89763 \\ 3.81944 & 7.64405 & 4.00132 & 17.11803 & 4.69419 \\ 4.63633 & 7.23263 & 3.77396 & 16.59701 & 4.69115 \\ 13.85454 & 4.96054 & 3.89513 & 13.52585 & 5.14835 \\ 14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\ -0.29962 & 2.84253 & 6.75213 & 14.53619 & 4.74199 \\ 1.17425 & 3.68871 & 6.30767 & 14.34984 & 3.83953 \\ 12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\ 12.09574 & 1.89219 & 4.22532 & 13.49677 & 3.61535 \\ 18.02290 & 3.41410 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 1.18866 & 4.44159 & 4.29928 & 6.96073 & -7.26224 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 5.79540 & 8.37216 & 2.19769 & 16.30872 & 8.26720 \\ 18.27282 & 3.31276 & 0.00000 & 20.00000 & 10.00000 \\ 3.85518 & 8.14801 & 1.67931 & 3.32508 & 9.54598 \\ 4.25875 & 5.18781 & 1.07141 & 2.42207 & 6.08437 \\ 9.61329 & 6.02488 & 3.34586 & 13.85801 & 2.03264 \\ 11.64800 & 1.59485 & 0.62531 & 9.55771 & 6.95571 \\ 11.81361 & 8.21630 & 6.69401 & 16.71029 & -6.72113 \\ 7.63304 & 2.71061 & 6.86349 & 14.06732 & 4.95418 \\ 13.21576 & 3.68902 & 1.78767 & 3.82087 & -5.65325 \\ 2.66120 & 8.22153 & 4.32066 & 17.91187 & 4.77626 \\ 10.34368 & 3.47564 & 3.76569 & 19.25766 & -10.00000 \\ 10.30917 & 3.50538 & 3.77597 & 19.22233 & -10.00000 \\ 3.81944 & 7.64405 & 4.00132 & 17.11803 & 4.19321 \\ 4.63633 & 7.23288 & 3.77396 & 16.59701 & 4.69115 \\ 13.85454 & 4.96054 & 3.89513 & 13.52585 & 5.14835 \\ 14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\ 0.00000 & 2.84253 & 6.75213 & 14.53619 & 4.74199 \\ 1.17425 & 3.68871 & 6.30767 & 14.34984 & 3.82000 \\ 12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\ 12.09574 & 1.89609 & 4.22532 & 13.49677 & 3.61535 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \begin{pmatrix} \vec{z}_{\text{mut}} \\ 100.00000 \\ 0.23932 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.43676 \\ 1.47003 \\ 100.00000 \\ 0.82723 \\ 0.81593 \\ 100.00000 \\ 100.00000 \\ 0.25217 \\ 0.24613 \\ 100.00000 \\ 100.00000 \\ 1.12326 \\ 1.07863 \\ 100.00000 \end{pmatrix}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 1.68160 & 4.54360 & 4.41830 & 7.06500 & -6.92790 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\ 5.09540 & 8.65600 & 2.32350 & 16.09740 & 8.16800 \\ 4.63790 & 2.39310 & 0.49750 & 1.56770 & 2.81630 \\ 3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 11.83050 & 1.19750 & 0.38130 & 9.17200 & 7.39730 \\ 18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\ 12.91660 & 9.66890 & 6.64930 & 17.40760 & -9.80150 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 13.21576 & 3.68902 & 1.78767 & 3.82087 & -5.65325 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 7.63304 & 2.71061 & 6.86349 & 14.06732 & 4.95418 \\ 11.05350 & 4.00070 & 1.98790 & 12.50400 & 4.66730 \\ 14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\ 13.85454 & 4.96054 & 3.89513 & 13.52585 & 5.14835 \\ 10.30917 & 3.50538 & 3.77597 & 19.22233 & -10.00000 \\ 10.97560 & 9.31580 & 3.35200 & 13.11060 & -2.16190 \\ 12.54630 & 6.99080 & 3.97180 & 8.27260 & 3.10430 \\ 12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\ 6.13320 & 3.29410 & 4.50720 & 5.33060 & 7.17670 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}} \begin{pmatrix} \vec{z}_{\text{Pop}} \\ 100.00000 \\ 100.00000 \\ 0.23932 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.41194 \\ 100.00000 \\ 0.93412 \\ 100.00000 \\ 100.00000 \\ 1.47003 \\ 100.00000 \\ 0.37197 \\ 0.43676 \\ 100.00000 \\ 0.24613 \\ 0.25217 \\ 0.81593 \\ 100.00000 \\ 0.59218 \\ 1.12326 \\ 0.32640 \\ 100.00000 \end{pmatrix}$$

Generation 3

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 1.68160 & 4.54360 & 4.41830 & 7.06500 & -6.92790 \\ 12.54630 & 6.99080 & 3.97180 & 8.27260 & 3.10430 \\ 2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\ 5.09540 & 8.65600 & 2.32350 & 16.09740 & 8.16800 \\ 4.63790 & 2.39310 & 0.49750 & 1.56770 & 2.81630 \\ 5.09540 & 8.65600 & 2.32350 & 16.09740 & 8.16800 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 13.21576 & 3.68902 & 1.78767 & 3.82087 & -5.65325 \\ 18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\ 11.83050 & 1.19750 & 0.38130 & 9.17200 & 7.39730 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 13.21576 & 3.68902 & 1.78767 & 3.82087 & -5.65325 \\ 11.05350 & 4.00070 & 1.98790 & 12.50400 & 4.66730 \\ 12.54630 & 6.99080 & 3.97180 & 8.27260 & 3.10430 \\ 13.85454 & 4.96054 & 3.89513 & 13.52585 & 5.14835 \\ 6.13320 & 3.29410 & 4.50720 & 5.33060 & 7.17670 \\ 10.97560 & 9.31580 & 3.35200 & 13.11060 & -2.16190 \\ 12.54630 & 6.99080 & 3.97180 & 8.27260 & 3.10430 \\ 12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\ 6.13320 & 3.29410 & 4.50720 & 5.33060 & 7.17670 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 11.55782 & 6.76815 & 4.01242 & 8.16273 & 2.19156 \\ 11.37894 & 6.72786 & 4.01977 & 8.14285 & 2.02639 \\ 2.17823 & 4.13108 & 7.59754 & 18.01644 & -5.69090 \\ 5.39107 & 9.11463 & 1.78895 & 15.90289 & 9.57268 \\ 4.52426 & 0.83746 & 0.04394 & -2.04133 & 1.48699 \\ 5.04409 & 7.95363 & 2.11872 & 14.46794 & 7.56782 \\ 11.28267 & 3.56283 & 2.38008 & 5.34318 & -4.11462 \\ 10.21503 & 3.49314 & 2.70727 & 6.18396 & -3.26483 \\ 20.32774 & 2.99137 & 1.89586 & 19.44315 & -8.27230 \\ 16.99400 & 2.28757 & 1.30165 & 15.41345 & -2.12461 \\ 3.20030 & 8.29482 & 3.20741 & 11.66156 & 6.91338 \\ 3.28554 & 8.31469 & 3.00467 & 10.52311 & 7.32379 \\ 8.65632 & 8.34795 & 4.76703 & 15.77513 & -0.99635 \\ 11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.69047 \\ 12.44287 & 6.78363 & 3.83434 & 8.56577 & 3.21259 \\ 12.39821 & 6.69417 & 3.77499 & 8.69237 & 3.25936 \\ 8.62068 & 3.83095 & 4.31002 & 7.97075 & 6.52325 \\ 6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01403 \\ 10.61949 & 9.84292 & 3.21148 & 14.20746 & -3.35585 \\ 11.72348 & 8.20876 & 3.64712 & 10.80700 & 0.34558 \\ 5.57475 & 3.39782 & 4.53126 & 4.64117 & 7.50572 \\ 6.43534 & 3.23798 & 4.49418 & 5.70360 & 6.99869 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 11.55782 & 6.76815 & 4.01242 & 8.16273 & 2.19156 \\ 11.37894 & 6.72786 & 3.51196 & 9.15847 & 3.04201 \\ 2.17823 & 4.00510 & 7.59754 & 18.01644 & -5.69090 \\ 5.39107 & 9.11463 & 1.78895 & 15.90253 & 9.57268 \\ 4.52426 & 0.83746 & 0.00000 & 0.00000 & 1.48699 \\ 5.04409 & 7.95363 & 2.11872 & 14.46794 & 7.56782 \\ 11.28267 & 3.56283 & 2.38008 & 5.34318 & -4.11462 \\ 10.21503 & 3.49314 & 2.70727 & 6.18396 & -3.26483 \\ 20.00000 & 2.99137 & 1.89586 & 19.44315 & -8.27230 \\ 16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\ 3.20030 & 8.29482 & 3.20741 & 11.34906 & 6.91338 \\ 3.28554 & 8.31469 & 3.00467 & 10.52311 & 7.32379 \\ 8.65632 & 8.34795 & 4.76703 & 15.77513 & -4.99635 \\ 11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\ 12.44287 & 6.78363 & 3.83434 & 8.56577 & 3.21259 \\ 12.39821 & 6.69417 & 3.77499 & 8.69237 & 3.25936 \\ 6.62068 & 3.83095 & 4.31002 & 9.97075 & 6.52325 \\ 6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \\ 10.62340 & 9.84292 & 3.21148 & 14.20746 & -3.35585 \\ 11.72348 & 8.20876 & 3.64712 & 10.80700 & 0.34558 \\ 5.57475 & 3.39782 & 4.53126 & 4.64117 & 7.50572 \\ 6.43534 & 3.23798 & 4.49418 & 5.70360 & 6.99869 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix} 0.51557 \\ 0.13231 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 1.24962 \\ 0.97857 \\ 1.45003 \\ 0.39728 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.55143 \\ 0.48824 \\ 0.44508 \\ 100.00000 \\ 0.24359 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.32800 \\ 100.00000 \end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 11.55782 & 6.76815 & 4.01242 & 8.16273 & 2.19156 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\ 5.09540 & 8.65600 & 2.32350 & 16.09740 & 8.16800 \\ 4.63790 & 2.39310 & 0.49750 & 1.56770 & 2.81630 \\ 3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\ 18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\ 12.91660 & 9.66890 & 6.64930 & 17.40760 & -9.80150 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 7.63304 & 2.71061 & 6.86349 & 14.06732 & 4.95418 \\ 12.44287 & 6.78363 & 3.83434 & 8.56577 & 3.21259 \\ 14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\ 13.85454 & 4.96054 & 3.89513 & 13.52585 & 5.14835 \\ 10.30917 & 3.50538 & 3.77597 & 19.22233 & -10.00000 \\ 10.97560 & 9.31580 & 3.35200 & 13.11060 & -2.16190 \\ 12.39821 & 6.69417 & 3.77499 & 8.69237 & 3.25936 \\ 12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\ 6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix} 100.00000 \\ 0.51557 \\ 0.23932 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.41194 \\ 0.39728 \\ 0.93412 \\ 100.00000 \\ 100.00000 \\ 0.55143 \\ 100.00000 \\ 0.37197 \\ 0.43676 \\ 0.48824 \\ 0.24613 \\ 0.25217 \\ 0.81593 \\ 100.00000 \\ 0.44508 \\ 1.12326 \\ 0.24359 \\ 100.00000 \end{pmatrix}}$$

Generation 4

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 10.97560 & 9.31580 & 3.35200 & 13.11060 & -2.16190 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\ 5.09540 & 8.65600 & 2.32350 & 16.09740 & 8.16800 \\ 18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\ 3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\ 11.55782 & 6.76815 & 4.01242 & 8.16273 & 2.19156 \\ 16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 12.91660 & 9.66890 & 6.64930 & 17.40760 & -9.80150 \\ 18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\ 11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 7.63304 & 2.71061 & 6.86349 & 14.06732 & 4.95418 \\ 14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\ 13.85454 & 4.96054 & 3.89513 & 13.52585 & 5.14835 \\ 10.30917 & 3.50538 & 3.77597 & 19.22233 & -10.00000 \\ 10.97560 & 9.31580 & 3.35200 & 13.11060 & -2.16190 \\ 12.39821 & 6.69417 & 3.77499 & 8.69237 & 3.25936 \\ 11.55782 & 6.76815 & 4.01242 & 8.16273 & 2.19156 \\ 6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 3.98909 & 5.30475 & 5.45883 & 13.99394 & 2.11598 \\ 7.74524 & 7.46121 & 4.32614 & 13.51903 & -0.18393 \\ 7.21160 & 4.91527 & 5.28987 & 8.88176 & -2.70453 \\ 13.67741 & 1.53103 & 4.01739 & 14.20315 & 4.56894 \\ 15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\ 20.28991 & 1.93441 & 1.51789 & 17.61771 & -6.82647 \\ 7.78950 & 7.58147 & 2.90559 & 5.85171 & 5.93760 \\ 11.34470 & 6.81415 & 3.94982 & 8.03203 & 2.40343 \\ 15.63892 & 2.47153 & 1.60567 & 14.07223 & -2.12636 \\ 10.56288 & 3.16061 & 2.74543 & 9.04996 & -2.13292 \\ 16.48279 & 5.32599 & 3.52937 & 17.43820 & -6.98279 \\ 14.14510 & 8.17284 & 5.57453 & 17.41814 & -8.83050 \\ 2.87490 & 8.14768 & 4.28189 & 17.66551 & 4.56617 \\ 6.27042 & 7.14142 & 3.78240 & 14.11390 & 1.23777 \\ 7.38749 & 2.41929 & 7.30808 & 13.42429 & 7.36136 \\ 8.07948 & 3.24027 & 6.05519 & 15.23642 & 0.57767 \\ 14.47948 & 5.44000 & 3.86521 & 13.37323 & 5.07645 \\ 13.95776 & 5.03973 & 3.89019 & 13.50064 & 5.13647 \\ 10.56564 & 5.74148 & 3.61281 & 16.87027 & -6.98356 \\ 10.19658 & 2.52369 & 3.84760 & 20.25492 & -11.32427 \\ 11.35252 & 6.78623 & 4.07043 & 8.03334 & 1.93071 \\ 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.34238 \\ 6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 3.98909 & 5.30475 & 5.45883 & 13.99394 & 2.11598 \\ 7.75306 & 7.46121 & 4.32614 & 13.51903 & -0.18393 \\ 7.33660 & 4.91527 & 5.35237 & 8.88176 & -2.70453 \\ 13.67741 & 1.53103 & 4.01739 & 13.20315 & 4.56894 \\ 15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\ 20.00000 & 1.91872 & 1.51789 & 17.61771 & -6.79510 \\ 7.78950 & 7.58147 & 2.90559 & 6.10171 & 5.93760 \\ 11.34470 & 6.81415 & 3.94982 & 8.03203 & 2.40343 \\ 15.63892 & 2.47153 & 1.60567 & 14.07223 & -2.12636 \\ 10.56288 & 3.16061 & 2.74543 & 9.04996 & -2.13292 \\ 16.48279 & 4.82599 & 3.52937 & 17.43820 & -6.98279 \\ 14.14510 & 8.17332 & 5.57453 & 17.41814 & -8.83050 \\ 2.87490 & 8.14768 & 4.28189 & 17.66551 & 4.56617 \\ 6.27042 & 7.14142 & 3.78240 & 14.11390 & 1.23777 \\ 7.38749 & 2.41929 & 7.30796 & 13.42429 & 7.36111 \\ 8.07948 & 3.24027 & 6.05519 & 15.23642 & 0.57767 \\ 14.47948 & 5.44000 & 3.86521 & 13.37323 & 5.07645 \\ 13.95776 & 5.03973 & 3.89019 & 13.50064 & 5.13647 \\ 10.56564 & 5.74148 & 3.61379 & 16.87222 & -6.98356 \\ 10.19658 & 1.89869 & 3.84760 & 20.00000 & -10.00000 \\ 11.35252 & 6.78623 & 4.07043 & 7.53334 & 1.93071 \\ 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\ 6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix} 100.00000 \\ 100.00000 \\ 0.35507 \\ 1.23238 \\ 0.09578 \\ 1.32876 \\ 100.00000 \\ 0.45667 \\ 0.48969 \\ 0.73891 \\ 1.31338 \\ 1.09603 \\ 100.00000 \\ 100.00000 \\ 0.43706 \\ 0.38060 \\ 0.24731 \\ 0.25153 \\ 0.23725 \\ 1.05383 \\ 0.65594 \\ 0.44171 \\ 0.24359 \end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\ 15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\ 4.63790 & 2.39310 & 0.49750 & 1.56770 & 2.81630 \\ 3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\ 18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\ 16.48279 & 4.82599 & 3.52937 & 17.43820 & -6.98279 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 8.07948 & 3.24027 & 6.05519 & 15.23642 & 0.57767 \\ 12.44287 & 6.78363 & 3.83434 & 8.56577 & 3.21259 \\ 14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\ 13.95776 & 5.03973 & 3.89019 & 13.50064 & 5.13647 \\ 10.56564 & 5.74148 & 3.61379 & 16.87222 & -6.98356 \\ 10.19658 & 1.89869 & 3.84760 & 20.00000 & -10.00000 \\ 12.39821 & 6.69417 & 3.77499 & 8.69237 & 3.25936 \\ 12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\ 6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000 \end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix} 100.00000 \\ 0.44171 \\ 0.23932 \\ 100.00000 \\ 0.09578 \\ 100.00000 \\ 100.00000 \\ 0.41194 \\ 0.39728 \\ 0.93412 \\ 1.31338 \\ 100.00000 \\ 0.55143 \\ 100.00000 \\ 0.37197 \\ 0.38060 \\ 0.48824 \\ 0.24613 \\ 0.25153 \\ 0.23725 \\ 1.05383 \\ 0.44508 \\ 1.12326 \\ 0.24359 \\ 100.00000 \end{pmatrix}}$$

Generation 5

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\
2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\
6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \\
4.63790 & 2.39310 & 0.49750 & 1.56770 & 2.81630 \\
16.48279 & 4.82599 & 3.52937 & 17.43820 & -6.98279 \\
8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\
11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\
18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\
16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\
2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\
3.81770 & 8.43870 & 1.73900 & 3.41590 & 9.88590 \\
8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\
11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\
12.44287 & 6.78363 & 3.83434 & 8.56577 & 3.21259 \\
14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\
13.95776 & 5.03973 & 3.89019 & 13.50064 & 5.13647 \\
6.75244 & 3.42775 & 4.45811 & 5.98785 & 7.01379 \\
10.19658 & 1.89869 & 3.84760 & 20.00000 & -10.00000 \\
10.56564 & 5.74148 & 3.61379 & 16.87222 & -6.98356 \\
12.60571 & 2.09196 & 4.22830 & 13.32114 & 3.36337 \\
12.39821 & 6.69417 & 3.77499 & 8.69237 & 3.25936 \\
18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
13.34606 & 6.50976 & 3.96246 & 9.27634 & 2.60558 \\
14.61562 & 6.25275 & 4.25872 & 10.05719 & 1.63555 \\
4.94037 & 3.87945 & 5.58793 & 10.95315 & 2.18117 \\
3.45270 & 4.25029 & 6.51548 & 15.02953 & -1.78628 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
8.85050 & 3.44982 & 3.14004 & 7.33571 & -2.16917 \\
9.31983 & 3.87291 & 3.12148 & 7.62835 & -2.46061 \\
18.71730 & 2.65138 & 1.60882 & 17.49652 & -5.30252 \\
17.40136 & 2.37357 & 1.37407 & 15.90548 & -2.87583 \\
3.97997 & 8.47651 & 1.35307 & 1.24878 & 10.66714 \\
3.02958 & 8.25504 & 3.61344 & 13.94154 & 6.09146 \\
8.31926 & 8.69237 & 4.98728 & 16.65886 & -0.65227 \\
11.92253 & 5.01048 & 2.63273 & 7.21157 & -4.32760 \\
15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\
13.14038 & 6.50228 & 3.83701 & 9.88584 & 3.72183 \\
8.29488 & 3.77282 & 4.33654 & 7.59610 & 6.61191 \\
15.54817 & 5.39553 & 3.76484 & 15.15892 & 4.72210 \\
10.63634 & 6.47762 & 3.56900 & 16.27305 & -6.40572 \\
10.55425 & 5.62285 & 3.62100 & 16.96878 & -7.07668 \\
12.41621 & 6.29494 & 3.81431 & 9.09391 & 3.26838 \\
12.35168 & 7.72608 & 3.67335 & 7.65450 & 3.23603 \\
18.02290 & 3.38285 & 0.00000 & 20.00000 & 10.00000
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
13.34606 & 6.44335 & 3.89606 & 9.27634 & 2.60558 \\
14.61562 & 6.25275 & 4.25872 & 10.05719 & 1.63555 \\
4.94037 & 3.87945 & 5.58793 & 10.95315 & 2.18117 \\
3.45270 & 4.25029 & 6.51548 & 15.02953 & -1.78628 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
8.85050 & 3.44982 & 3.14004 & 7.33571 & -2.16917 \\
9.30421 & 3.87291 & 3.12929 & 7.62835 & -2.46061 \\
18.74855 & 2.65138 & 1.60882 & 17.49652 & -5.30252 \\
17.40136 & 2.37357 & 1.37407 & 15.90548 & -2.87583 \\
3.97997 & 8.47651 & 1.35307 & 1.24878 & 10.00000 \\
3.02958 & 8.75894 & 3.61344 & 13.94154 & 6.09146 \\
8.31926 & 8.69237 & 4.98728 & 16.65886 & -0.65227 \\
11.92253 & 5.01048 & 2.63273 & 7.21157 & -4.32760 \\
15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\
13.14038 & 6.50228 & 3.83701 & 9.88584 & 3.72183 \\
9.29683 & 3.77282 & 4.83752 & 6.59415 & 6.61191 \\
15.54817 & 5.39553 & 3.76484 & 15.15892 & 4.72210 \\
10.63634 & 6.47762 & 3.56900 & 16.27305 & -6.40572 \\
10.55425 & 5.62285 & 3.62100 & 16.96878 & -7.07692 \\
12.41621 & 6.29494 & 3.81437 & 9.09391 & 3.26838 \\
12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\
18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \begin{pmatrix}
\vec{z}_{\text{mut}} \\
0.62942 \\
1.01394 \\
100.00000 \\
100.00000 \\
0.05839 \\
0.30947 \\
0.46214 \\
0.58484 \\
0.94731 \\
0.52330 \\
100.00000 \\
100.00000 \\
100.00000 \\
0.94938 \\
0.20805 \\
0.40848 \\
0.68049 \\
0.20261 \\
100.00000 \\
0.24942 \\
0.45554 \\
0.43734 \\
100.00000
\end{pmatrix}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\
2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\
15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
14.61562 & 6.25275 & 4.25872 & 10.05719 & 1.63555 \\
8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\
16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\
18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\
11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\
8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\
9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\
8.07948 & 3.24027 & 6.05519 & 15.23642 & 0.57767 \\
15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\
14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\
13.95776 & 5.03973 & 3.89019 & 13.50064 & 5.13647 \\
10.56564 & 5.74148 & 3.61379 & 16.87222 & -6.98356 \\
10.19658 & 1.89869 & 3.84760 & 20.00000 & -10.00000 \\
12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\
12.41621 & 6.29494 & 3.81437 & 9.09391 & 3.26838 \\
15.54817 & 5.39553 & 3.76484 & 15.15892 & 4.72210 \\
18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}} \begin{pmatrix}
\vec{z}_{\text{Pop}} \\
100.00000 \\
0.44171 \\
0.23932 \\
100.00000 \\
0.09578 \\
0.05839 \\
1.01394 \\
0.41194 \\
0.39728 \\
0.93412 \\
0.30947 \\
100.00000 \\
0.55143 \\
100.00000 \\
0.37197 \\
0.38060 \\
0.20805 \\
0.24613 \\
0.25153 \\
0.23725 \\
1.05383 \\
0.43734 \\
0.45554 \\
0.20261 \\
100.00000
\end{pmatrix}$$

Generation 6

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 2.42090 & 4.50750 & 7.15880 & 17.85680 & -4.53800 \\ 15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\ 9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\ 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 11.29405 & 5.65266 & 3.04341 & 8.85933 & -3.68656 \\ 18.68470 & 2.64450 & 1.60300 & 17.45710 & -5.24240 \\ 11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\ 2.74020 & 8.18760 & 4.30170 & 17.80640 & 4.69820 \\ 14.61562 & 6.25275 & 4.25872 & 10.05719 & 1.63555 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\ 8.07948 & 3.24027 & 6.05519 & 15.23642 & 0.57767 \\ 13.95776 & 5.03973 & 3.89019 & 13.50064 & 5.13647 \\ 15.54817 & 5.39553 & 3.76484 & 15.15892 & 4.72210 \\ 10.19658 & 1.89869 & 3.84760 & 20.00000 & -10.00000 \\ 8.07948 & 3.24027 & 6.05519 & 15.23642 & 0.57767 \\ 12.41621 & 6.29494 & 3.81437 & 9.09391 & 3.26838 \\ 14.92552 & 5.78221 & 3.84385 & 13.26430 & 5.02513 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 9.42266 & 6.10080 & 4.59582 & 8.38532 & -0.17769 \\ 12.03081 & 6.60480 & 3.87596 & 8.68400 & 2.78785 \\ 16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\ 1.62117 & 4.53579 & 7.48773 & 17.90053 & -4.68561 \\ 4.30422 & 3.56097 & 4.56259 & 11.91923 & 1.94973 \\ 4.83714 & 3.54157 & 4.26645 & 11.50787 & 1.62793 \\ 9.74997 & 4.26068 & 3.10447 & 7.89655 & -2.72772 \\ 11.31850 & 5.67470 & 3.04244 & 8.87458 & -3.70174 \\ 11.56487 & 3.73660 & 2.19562 & 10.76065 & -2.80707 \\ 17.87824 & 2.76820 & 1.67013 & 16.69860 & -4.96655 \\ 12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\ 0.16914 & 8.60650 & 4.31101 & 19.48412 & 5.36127 \\ 8.73658 & 6.89022 & 4.58106 & 17.02457 & -4.53485 \\ 8.42980 & 8.67124 & 4.92396 & 16.27158 & -0.53428 \\ 12.13813 & 4.48799 & 4.85358 & 14.79704 & 3.24125 \\ 12.08219 & 4.47080 & 4.87015 & 14.80310 & 3.20454 \\ 15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\ 15.65479 & 5.41939 & 3.75643 & 15.27008 & 4.69432 \\ 8.67718 & 2.86152 & 5.43194 & 16.58127 & -2.40861 \\ 7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\ 12.33929 & 6.31066 & 3.81347 & 8.96607 & 3.21453 \\ 15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 9.42266 & 6.10080 & 4.59582 & 8.38532 & -0.17769 \\ 12.03081 & 6.60480 & 3.87596 & 8.68400 & 2.78785 \\ 16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\ 1.62117 & 4.53579 & 7.48773 & 17.90053 & -4.68561 \\ 4.30422 & 3.56097 & 4.56259 & 11.91923 & 1.94973 \\ 4.83714 & 3.54157 & 4.26645 & 11.50787 & 1.62793 \\ 9.74997 & 4.26068 & 3.10447 & 7.89655 & -2.72772 \\ 11.31850 & 5.67470 & 3.04244 & 10.99958 & -3.70174 \\ 11.56487 & 3.73660 & 2.19562 & 10.74503 & -2.80707 \\ 17.87824 & 2.76820 & 1.67013 & 16.69835 & -4.96655 \\ 12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\ 0.16914 & 8.60650 & 4.12326 & 19.48412 & 5.36127 \\ 8.73658 & 6.89022 & 4.58106 & 17.02457 & -4.53485 \\ 8.42980 & 8.67124 & 4.92396 & 16.27158 & -0.53428 \\ 12.13813 & 4.48799 & 4.85358 & 14.79704 & 3.24125 \\ 12.08219 & 4.47080 & 4.87015 & 14.80310 & 3.20454 \\ 15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\ 15.65479 & 5.41939 & 3.75643 & 15.27008 & 4.69432 \\ 9.17742 & 2.86152 & 5.43194 & 16.58127 & -2.40861 \\ 7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\ 12.33929 & 6.31066 & 3.81347 & 8.96607 & 3.21453 \\ 15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix} 100.00000 \\ 0.45912 \\ 0.01464 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.65560 \\ 0.28691 \\ 0.32898 \\ 0.86961 \\ 0.22960 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.43189 \\ 0.43127 \\ 0.20251 \\ 0.20285 \\ 0.65737 \\ 0.35296 \\ 0.46342 \\ 0.20462 \\ 100.00000 \end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\ 15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\ 9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\ 14.61562 & 6.25275 & 4.25872 & 10.05719 & 1.63555 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\ 11.56487 & 3.73660 & 2.19562 & 10.74503 & -2.80707 \\ 11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\ 12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\ 11.31850 & 5.67470 & 3.04244 & 10.99958 & -3.70174 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\ 15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\ 15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\ 15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\ 10.56564 & 5.74148 & 3.61379 & 16.87222 & -6.98356 \\ 9.17742 & 2.86152 & 5.43194 & 16.58127 & -2.40861 \\ 12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\ 12.41621 & 6.29494 & 3.81437 & 9.09391 & 3.26838 \\ 15.54817 & 5.39553 & 3.76484 & 15.15892 & 4.72210 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix} 100.00000 \\ 0.44171 \\ 0.23932 \\ 0.01464 \\ 0.09578 \\ 0.05839 \\ 1.01394 \\ 0.41194 \\ 0.39728 \\ 0.32898 \\ 0.30947 \\ 0.22960 \\ 0.28691 \\ 100.00000 \\ 0.37197 \\ 0.35296 \\ 0.20805 \\ 0.20462 \\ 0.20251 \\ 0.23725 \\ 0.65737 \\ 0.43734 \\ 0.45554 \\ 0.20261 \\ 100.00000 \end{pmatrix}}$$

Generation 7

$$\xrightarrow{\text{Selektion}} \begin{pmatrix} 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\ 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\ 15.54817 & 5.39553 & 3.76484 & 15.15892 & 4.72210 \\ 14.61562 & 6.25275 & 4.25872 & 10.05719 & 1.63555 \\ 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\ 16.99400 & 2.28757 & 1.30140 & 15.41296 & -2.12461 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\ 11.56487 & 3.73660 & 2.19562 & 10.74503 & -2.80707 \\ 11.31850 & 5.67470 & 3.04244 & 10.99958 & -3.70174 \\ 15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\ 9.14710 & 4.50690 & 4.12220 & 18.03220 & -9.88830 \\ 11.56487 & 3.73660 & 2.19562 & 10.74503 & -2.80707 \\ 15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\ 12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\ 15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\ 15.54817 & 5.39553 & 3.76484 & 15.15892 & 4.72210 \\ 9.17742 & 2.86152 & 5.43194 & 16.58127 & -2.40861 \\ 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 12.41621 & 6.29494 & 3.81437 & 9.09391 & 3.26838 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}$$

Nachkommen$_{\text{sel}}$

$$\xrightarrow{\text{Rekombination}} \begin{pmatrix} 14.55701 & 7.24597 & 3.28314 & 7.69364 & 3.17374 \\ 8.45354 & 5.62043 & 4.66331 & 10.72550 & 3.48319 \\ 16.16303 & 4.27581 & 1.88383 & 16.73292 & -0.71540 \\ 15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\ 12.82199 & 6.61586 & 3.84017 & 8.95401 & 3.00601 \\ 14.40654 & 6.29508 & 4.20993 & 9.92859 & 1.79530 \\ 17.75965 & 2.18364 & 1.12949 & 16.17050 & -2.12362 \\ 8.83735 & 3.39486 & 3.13287 & 7.34271 & -2.13515 \\ 11.59869 & 3.73142 & 2.19281 & 10.77445 & -2.81864 \\ 11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\ 13.16736 & 5.66695 & 3.39672 & 12.43804 & 0.30982 \\ 11.73527 & 5.67295 & 3.12230 & 11.32383 & -2.79747 \\ 12.12267 & 3.55889 & 1.75114 & 9.06381 & -1.17336 \\ 10.30844 & 4.13690 & 3.19679 & 14.53190 & -6.48693 \\ 14.35778 & 6.34718 & 3.77047 & 11.91580 & 4.64402 \\ 11.79552 & 8.10837 & 3.60643 & 6.47311 & 2.84569 \\ 15.55216 & 5.39643 & 3.76452 & 15.16308 & 4.72106 \\ 15.67988 & 5.42500 & 3.75445 & 15.29625 & 4.68778 \\ 10.58427 & 2.71880 & 5.27724 & 16.97192 & -3.50628 \\ 5.83726 & 3.20036 & 5.79924 & 15.65379 & 0.19750 \\ 8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\ 12.02315 & 6.24311 & 3.92005 & 9.02594 & 2.81880 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}$$

Nachkommen$_{\text{rek}}$

$$\xrightarrow{\text{Mutation}} \begin{pmatrix} 14.55701 & 7.24597 & 3.28314 & 7.69364 & 3.17374 \\ 8.45354 & 5.62043 & 4.66307 & 10.72550 & 3.48319 \\ 16.16303 & 4.27581 & 1.88383 & 16.73292 & -0.71540 \\ 15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\ 12.82199 & 6.61586 & 3.84017 & 8.95401 & 3.00601 \\ 14.40459 & 6.29508 & 4.20993 & 9.92859 & 1.79335 \\ 17.75965 & 2.18364 & 1.12949 & 15.91171 & -2.38241 \\ 8.83735 & 3.39486 & 3.13287 & 7.34271 & -2.13515 \\ 11.59869 & 3.73142 & 2.19281 & 10.77445 & -2.81864 \\ 11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\ 13.16736 & 5.66695 & 3.39672 & 12.43804 & 0.30982 \\ 11.73527 & 5.67295 & 3.12230 & 11.32383 & -2.79747 \\ 12.12267 & 3.55889 & 1.75114 & 9.06381 & -1.17336 \\ 10.30844 & 4.13690 & 3.19679 & 16.53190 & -8.48693 \\ 14.35778 & 6.34718 & 3.77047 & 11.91580 & 4.64402 \\ 11.79552 & 8.10837 & 3.60643 & 6.47311 & 2.84569 \\ 15.55216 & 5.39643 & 3.76452 & 15.16308 & 4.72106 \\ 15.67976 & 5.42500 & 3.75445 & 15.29625 & 4.68766 \\ 10.58427 & 2.71880 & 5.27821 & 16.97192 & -3.50628 \\ 5.83726 & 3.20525 & 5.79924 & 15.65379 & 0.20726 \\ 8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\ 12.02315 & 6.24311 & 3.92005 & 9.02594 & 2.81892 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}$$

Nachkommen$_{\text{mut}}$

$$\xrightarrow{\text{Bewertung}} \begin{pmatrix} 0.64875 \\ 100.00000 \\ 0.02800 \\ 0.11650 \\ 0.52474 \\ 0.95383 \\ 0.39052 \\ 0.42635 \\ 0.33383 \\ 0.30543 \\ 0.25113 \\ 0.28758 \\ 0.24640 \\ 0.41104 \\ 0.24552 \\ 100.00000 \\ 0.20372 \\ 0.20288 \\ 1.07437 \\ 100.00000 \\ 0.21369 \\ 0.47152 \\ 100.00000 \end{pmatrix}$$

$\vec{z}_{\text{mut}}$

$$\xrightarrow{\text{Wiedereinfügen}} \begin{pmatrix} 1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\ 12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\ 6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\ 16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\ 15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\ 9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\ 12.82199 & 6.61586 & 3.84017 & 8.95401 & 3.00601 \\ 8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\ 17.75965 & 2.18364 & 1.12949 & 15.91171 & -2.38241 \\ 11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\ 11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\ 12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\ 13.16736 & 5.66695 & 3.39672 & 12.43804 & 0.30982 \\ 8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\ 12.12267 & 3.55889 & 1.75114 & 9.06381 & -1.17336 \\ 7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\ 15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\ 15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\ 15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\ 10.56564 & 5.74148 & 3.61379 & 16.87222 & -6.98356 \\ 9.17742 & 2.86152 & 5.43194 & 16.58127 & -2.40861 \\ 12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\ 8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\ 15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\ 18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}$$

Population

$$\vec{z}_{\text{Pop}} \begin{pmatrix} 100.00000 \\ 0.44171 \\ 0.23932 \\ 0.01464 \\ 0.09578 \\ 0.05839 \\ 0.52474 \\ 0.41194 \\ 0.39052 \\ 0.30543 \\ 0.30947 \\ 0.22960 \\ 0.25113 \\ 100.00000 \\ 0.24640 \\ 0.35296 \\ 0.20805 \\ 0.20462 \\ 0.20251 \\ 0.23725 \\ 0.65737 \\ 0.43734 \\ 0.21369 \\ 0.11650 \\ 100.00000 \end{pmatrix}$$

Generation 8

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
12.82199 & 6.61586 & 3.84017 & 8.95401 & 3.00601 \\
16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\
15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
1.15160 & 3.67570 & 6.31450 & 14.35270 & 3.85340 \\
8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\
17.75965 & 2.18364 & 1.12949 & 15.91171 & -2.38241 \\
11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\
15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\
8.44900 & 8.55980 & 4.90250 & 16.31870 & -0.78460 \\
17.75965 & 2.18364 & 1.12949 & 15.91171 & -2.38241 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
15.54077 & 5.53404 & 3.84620 & 14.42868 & 5.47431 \\
15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\
13.16736 & 5.66695 & 3.39672 & 12.43804 & 0.30982 \\
10.56564 & 5.74148 & 3.61379 & 16.87222 & -6.98356 \\
15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\
8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\
15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\
18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
12.78104 & 6.62415 & 3.83061 & 8.92882 & 3.03730 \\
12.77765 & 6.62483 & 3.82982 & 8.92674 & 3.03989 \\
16.36350 & 4.01432 & 1.42413 & 17.09445 & -1.96458 \\
15.33554 & 4.05068 & 1.84694 & 17.15066 & -2.15431 \\
6.55570 & 3.47903 & 3.31145 & 10.18134 & 0.59021 \\
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.43306 & 2.90673 & 2.32550 & 10.79532 & -2.23551 \\
9.21516 & 3.34357 & 3.04804 & 7.70441 & -2.14677 \\
11.35142 & 3.76935 & 2.21339 & 10.55937 & -2.73406 \\
11.34946 & 3.76965 & 2.21355 & 10.55766 & -2.73339 \\
13.85984 & 6.09000 & 4.05550 & 13.05457 & 3.88275 \\
15.93077 & 5.40505 & 3.79764 & 14.74749 & 5.84358 \\
19.77231 & 0.80532 & 0.31389 & 15.82373 & -2.72781 \\
8.39382 & 8.59759 & 4.92486 & 16.32111 & -0.77513 \\
10.08575 & 3.92151 & 5.51236 & 14.85769 & 2.28150 \\
11.21613 & 4.25565 & 5.16710 & 14.76879 & 2.94311 \\
15.34944 & 5.65780 & 3.81486 & 14.13576 & 5.04438 \\
14.09895 & 5.66304 & 3.57523 & 13.16284 & 2.33112 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
12.94350 & 5.59207 & 3.68003 & 16.12877 & -1.53682 \\
14.22044 & 5.67156 & 4.04816 & 13.22064 & 4.19535 \\
7.55373 & 5.74518 & 5.04531 & 7.84797 & -2.17253 \\
18.02290 & 3.38285 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
12.78104 & 6.62415 & 3.83061 & 8.92882 & 3.03730 \\
12.77765 & 6.62483 & 3.82982 & 8.92674 & 3.03989 \\
16.36350 & 4.01481 & 1.42413 & 17.09445 & -1.96458 \\
15.33554 & 4.05068 & 1.84694 & 17.15066 & -2.15431 \\
6.30570 & 3.47903 & 3.31145 & 10.18134 & 0.59021 \\
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.43306 & 2.90673 & 2.32550 & 10.79532 & -2.23551 \\
9.21516 & 3.35920 & 3.03241 & 7.70441 & -2.11552 \\
11.35142 & 3.76935 & 2.21339 & 10.55937 & -2.73406 \\
11.34946 & 3.76965 & 2.21355 & 10.55766 & -2.73339 \\
13.85984 & 6.09000 & 4.05550 & 13.05457 & 3.88275 \\
15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\
19.77231 & 0.80532 & 0.31389 & 15.82373 & -2.72781 \\
8.39382 & 8.59759 & 4.92486 & 16.32111 & -0.77513 \\
10.58672 & 3.92151 & 5.51236 & 15.35867 & 2.78248 \\
11.21613 & 4.25565 & 5.16710 & 14.76879 & 2.94311 \\
15.34944 & 5.65780 & 3.81486 & 14.13576 & 5.04438 \\
14.09895 & 5.66304 & 3.31742 & 13.16284 & 2.33112 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
12.94350 & 5.59207 & 3.68003 & 16.12877 & -1.53682 \\
14.22044 & 5.67058 & 4.04816 & 13.22064 & 4.19535 \\
7.55373 & 5.74518 & 5.04531 & 7.84797 & -2.17253 \\
18.03096 & 3.38688 & 0.00000 & 19.99194 & 9.98438
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
0.51426 \\
0.51349 \\
0.02079 \\
0.10464 \\
100.00000 \\
0.01584 \\
0.69515 \\
0.57698 \\
0.31014 \\
0.31001 \\
0.24337 \\
0.19887 \\
0.34428 \\
100.00000 \\
0.51610 \\
0.46760 \\
0.21020 \\
0.17221 \\
0.16194 \\
0.23794 \\
0.32177 \\
0.28446 \\
100.00000
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\
16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\
15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
12.77765 & 6.62483 & 3.82982 & 8.92674 & 3.03989 \\
8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\
17.75965 & 2.18364 & 1.12949 & 15.91171 & -2.38241 \\
11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\
14.09895 & 5.66304 & 3.31742 & 13.16284 & 2.33112 \\
19.77231 & 0.80532 & 0.31389 & 15.82373 & -2.72781 \\
12.12267 & 3.55889 & 1.75114 & 9.06381 & -1.17336 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\
15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\
15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
9.17742 & 2.86152 & 5.43194 & 16.58127 & -2.40861 \\
12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\
8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\
18.03096 & 3.38688 & 0.00000 & 19.99194 & 9.98438
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
0.01584 \\
0.44171 \\
0.23932 \\
0.01464 \\
0.09578 \\
0.05839 \\
0.51349 \\
0.41194 \\
0.39052 \\
0.30543 \\
0.30947 \\
0.22960 \\
0.17221 \\
0.34428 \\
0.24640 \\
0.35296 \\
0.19887 \\
0.20462 \\
0.20251 \\
0.16194 \\
0.65737 \\
0.43734 \\
0.21369 \\
0.11650 \\
100.00000
\end{pmatrix}}$$

Generation 9

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
9.17742 & 2.86152 & 5.43194 & 16.58127 & -2.40861 \\
6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\
8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\
15.52077 & 4.04413 & 1.77075 & 17.14053 & -2.12013 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
12.77765 & 6.62483 & 3.82982 & 8.92674 & 3.03989 \\
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
17.75965 & 2.18364 & 1.12949 & 15.91171 & -2.38241 \\
8.79580 & 3.40050 & 3.14220 & 7.30160 & -2.13520 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
14.09895 & 5.66304 & 3.31742 & 13.16284 & 2.33112 \\
12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\
12.12267 & 3.55889 & 1.75114 & 9.06381 & -1.17336 \\
19.77231 & 0.80532 & 0.31389 & 15.82373 & -2.72781 \\
15.53828 & 5.65701 & 3.85104 & 14.28269 & 5.45412 \\
15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
12.12267 & 3.55889 & 1.75114 & 9.06381 & -1.17336 \\
12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
11.00516 & 3.26551 & 1.19237 & 7.57917 & -2.23642 \\
10.55211 & 3.16538 & 2.24324 & 9.81053 & -2.27910 \\
6.35684 & 5.30990 & 5.31682 & 7.95637 & -3.75938 \\
7.17372 & 4.67039 & 4.58848 & 7.73707 & -3.21539 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
12.03070 & 3.66140 & 1.74229 & 11.89743 & -1.55850 \\
12.51365 & 6.67828 & 3.76822 & 8.76437 & 3.24160 \\
12.49548 & 6.68195 & 3.76398 & 8.75319 & 3.25548 \\
8.00188 & 3.50828 & 3.32046 & 6.53901 & -2.11330 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42549 \\
9.61427 & 3.52845 & 4.24271 & 12.85445 & -0.78358 \\
10.41064 & 3.63989 & 3.30866 & 11.79476 & -1.68044 \\
13.26483 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
12.88185 & 6.28355 & 3.91274 & 12.22253 & 2.29270 \\
20.21213 & 0.64700 & 0.23125 & 16.21240 & -2.81718 \\
21.13956 & 0.31316 & 0.05700 & 17.03195 & -3.00564 \\
15.55853 & 5.61869 & 3.83557 & 14.44448 & 5.33034 \\
15.57238 & 5.59248 & 3.82498 & 14.55521 & 5.24562 \\
11.68423 & 3.38905 & 1.55846 & 8.50767 & -1.96523 \\
13.20283 & 3.97731 & 2.22584 & 10.43395 & 0.77749 \\
12.34680 & 7.77140 & 3.63704 & 7.60738 & 3.23412 \\
12.32759 & 7.94957 & 3.61738 & 7.42210 & 3.22659 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
11.00516 & 3.26551 & 1.19237 & 7.57917 & -2.23545 \\
10.55211 & 3.16538 & 2.24324 & 9.81053 & -2.27910 \\
6.35684 & 5.30990 & 5.31682 & 7.95637 & -3.75938 \\
7.17372 & 4.67039 & 4.58848 & 7.73707 & -3.21539 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
12.03070 & 3.66140 & 1.74229 & 11.89743 & -1.55850 \\
12.57615 & 6.64703 & 3.76822 & 8.76437 & 3.24160 \\
12.99548 & 6.93195 & 3.76398 & 8.75319 & 3.75548 \\
8.00188 & 3.50828 & 3.32339 & 6.53901 & -2.11330 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42524 \\
9.61427 & 3.52845 & 4.24271 & 12.85445 & -0.78358 \\
10.41064 & 3.63989 & 3.30866 & 11.79476 & -1.68044 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
12.88185 & 6.28355 & 3.91274 & 11.97253 & 2.29270 \\
20.00000 & 0.64700 & 0.23125 & 16.21240 & -2.81718 \\
20.00000 & 0.31316 & 0.05700 & 17.03195 & -3.00564 \\
15.55853 & 5.61869 & 3.83557 & 14.44448 & 5.33034 \\
15.57238 & 5.59248 & 3.82498 & 14.55521 & 5.24562 \\
11.68423 & 3.38905 & 1.55846 & 8.50767 & -1.96523 \\
13.20283 & 3.97731 & 2.22584 & 10.43395 & 0.77749 \\
12.34680 & 7.77384 & 3.63704 & 7.60738 & 3.23412 \\
12.32759 & 7.94957 & 3.61738 & 7.42210 & 3.22659 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
0.13994 \\
0.41632 \\
0.29708 \\
0.35707 \\
0.05692 \\
0.09452 \\
0.46686 \\
0.46922 \\
0.27843 \\
0.24898 \\
0.50397 \\
0.45319 \\
0.12758 \\
0.23999 \\
0.33275 \\
0.31440 \\
0.20421 \\
0.20344 \\
0.25553 \\
0.23509 \\
0.43754 \\
0.44239 \\
0.11650
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\
16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
12.57615 & 6.64703 & 3.76822 & 8.76437 & 3.24160 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42524 \\
8.00188 & 3.50828 & 3.32339 & 6.53901 & -2.11330 \\
11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
20.00000 & 0.31316 & 0.05700 & 17.03195 & -3.00564 \\
13.20283 & 3.97731 & 2.22584 & 10.43395 & 0.77749 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\
15.55853 & 5.61869 & 3.83557 & 14.44448 & 5.33034 \\
15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
10.55211 & 3.16538 & 2.24324 & 9.81053 & -2.27910 \\
12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\
8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\
18.03096 & 3.38688 & 0.00000 & 19.99194 & 9.98438
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
0.01584 \\
0.44171 \\
0.23932 \\
0.01464 \\
0.05692 \\
0.05839 \\
0.46686 \\
0.24898 \\
0.27843 \\
0.30543 \\
0.30947 \\
0.22960 \\
0.12758 \\
0.31440 \\
0.23509 \\
0.35296 \\
0.19887 \\
0.20421 \\
0.20251 \\
0.16194 \\
0.41632 \\
0.43734 \\
0.21369 \\
0.11650 \\
100.00000
\end{pmatrix}}$$

Generation 10

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.46355 & 6.68842 & 3.75653 & 8.73355 & 3.27988 \\
6.15353 & 5.46907 & 5.49810 & 8.01096 & -3.89477 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42524 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
9.41781 & 3.37487 & 1.72098 & 7.97211 & -1.13804 \\
12.57615 & 6.64703 & 3.76822 & 8.76437 & 3.24160 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42524 \\
8.00188 & 3.50828 & 3.32339 & 6.53901 & -2.11330 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42524 \\
11.34395 & 3.77049 & 2.21401 & 10.55287 & -2.73151 \\
12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
20.00000 & 0.31316 & 0.05700 & 17.03195 & -3.00564 \\
13.20283 & 3.97731 & 2.22584 & 10.43395 & 0.77749 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
15.55853 & 5.61869 & 3.83557 & 14.44448 & 5.33034 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
13.20283 & 3.97731 & 2.22584 & 10.43395 & 0.77749 \\
12.35168 & 7.72608 & 3.64204 & 7.65450 & 3.23603 \\
8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
12.33656 & 6.34735 & 3.45076 & 8.51738 & 2.72493 \\
12.10143 & 5.71587 & 2.88462 & 8.11714 & 1.69742 \\
21.59688 & 1.36724 & -0.03670 & 19.03635 & -2.17130 \\
16.28129 & 2.77909 & 1.86837 & 15.24142 & -2.76451 \\
9.34014 & 3.36635 & 1.72035 & 7.85543 & -1.12554 \\
9.35332 & 3.36780 & 1.72046 & 7.87523 & -1.12766 \\
19.91787 & 1.55818 & 0.51447 & 18.17650 & -2.92636 \\
11.56626 & 7.34702 & 4.21579 & 7.46969 & 4.09003 \\
20.45826 & 1.81729 & 0.52326 & 18.50382 & -2.45657 \\
12.48459 & 2.89974 & 2.31570 & 10.84482 & -2.23684 \\
12.03789 & 6.23317 & 3.96825 & 11.50030 & 1.54071 \\
11.50400 & 4.33849 & 2.61861 & 10.77139 & -1.74616 \\
15.90320 & 3.36993 & 1.99869 & 14.64294 & -0.19484 \\
12.92430 & 5.59260 & 3.41054 & 12.90582 & 1.84898 \\
12.90970 & 3.86376 & 2.09702 & 10.06212 & 0.24807 \\
15.36466 & 4.81475 & 3.17589 & 13.17612 & 4.68192 \\
11.92477 & 6.13601 & 3.71422 & 12.32270 & 1.99736 \\
13.26422 & 5.94532 & 3.75895 & 13.10481 & 3.22594 \\
12.69514 & 3.78065 & 2.00272 & 9.78996 & -0.13944 \\
16.17890 & 5.13016 & 3.53373 & 14.20896 & 6.15252 \\
8.33428 & 5.80531 & 4.90539 & 8.33978 & -1.39906 \\
8.74716 & 6.00271 & 4.77555 & 8.26935 & -0.92270 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
8.33265 & 8.34931 & 3.45076 & 4.51348 & 2.72493 \\
12.10143 & 5.71587 & 2.88462 & 8.11714 & 1.69742 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
16.28129 & 2.77909 & 1.86837 & 15.24142 & -2.76451 \\
9.34014 & 4.39773 & 1.72035 & 7.85543 & -1.12554 \\
9.35332 & 3.36780 & 1.72046 & 7.87523 & -1.12766 \\
19.91787 & 1.55818 & 0.63947 & 18.42650 & -2.92636 \\
11.56626 & 7.34702 & 4.21579 & 7.46969 & 4.09003 \\
20.00000 & 1.81729 & 0.52326 & 18.50382 & -2.45657 \\
12.48459 & 0.00000 & 2.31570 & 10.84482 & -2.23684 \\
12.03789 & 8.24586 & 3.96825 & 11.50030 & -2.48468 \\
11.50400 & 4.33849 & 2.61861 & 10.77139 & -1.74616 \\
15.90320 & 3.35430 & 1.99869 & 14.64294 & -0.19484 \\
12.92479 & 5.59260 & 3.41054 & 12.90582 & 1.84898 \\
12.90970 & 3.86376 & 2.09702 & 10.06212 & 0.24807 \\
15.36466 & 4.81475 & 3.17589 & 13.17612 & 4.68192 \\
11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\
13.26422 & 5.94532 & 3.75895 & 13.10481 & 3.22594 \\
12.69514 & 3.78065 & 2.00272 & 9.78996 & -0.13944 \\
16.17890 & 5.13016 & 3.53373 & 14.20896 & 6.15252 \\
8.33428 & 5.80531 & 4.90545 & 8.33978 & -1.39906 \\
8.74716 & 6.00271 & 4.77555 & 8.26935 & -0.92270 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
100.00000 \\
0.32182 \\
-0.01025 \\
0.61829 \\
100.00000 \\
0.05690 \\
0.33598 \\
0.56273 \\
0.16485 \\
100.00000 \\
0.24581 \\
0.27886 \\
0.24313 \\
0.15150 \\
0.23018 \\
0.21148 \\
0.10957 \\
0.15962 \\
0.22958 \\
0.18384 \\
0.25011 \\
100.00000 \\
0.11650
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.10143 & 5.71587 & 2.88462 & 8.11714 & 1.69742 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
9.35332 & 3.36780 & 1.72046 & 7.87523 & -1.12766 \\
19.91787 & 1.55818 & 0.63947 & 18.42650 & -2.92636 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42524 \\
20.00000 & 1.81729 & 0.52326 & 18.50382 & -2.45657 \\
11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\
12.03789 & 8.24586 & 3.96825 & 11.50030 & -2.48468 \\
12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
12.92479 & 5.59260 & 3.41054 & 12.90582 & 1.84898 \\
16.17890 & 5.13016 & 3.53373 & 14.20896 & 6.15252 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\
11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\
15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
10.55211 & 3.16538 & 2.24324 & 9.81053 & -2.27910 \\
8.33428 & 5.80531 & 4.90545 & 8.33978 & -1.39906 \\
8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\
18.03096 & 3.38688 & 0.00000 & 19.99194 & 9.98438
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
0.01584 \\
0.32182 \\
-0.01025 \\
0.01464 \\
0.05692 \\
0.05690 \\
0.33598 \\
0.24898 \\
0.16485 \\
0.30543 \\
0.24581 \\
0.22960 \\
0.12758 \\
0.15150 \\
0.18384 \\
0.35296 \\
0.19887 \\
0.10957 \\
0.20251 \\
0.16194 \\
0.41632 \\
0.25011 \\
0.21369 \\
0.11650 \\
100.00000
\end{pmatrix}}$$

Generation 11

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
12.10143 & 5.71587 & 2.88462 & 8.11714 & 1.69742 \\
19.91787 & 1.55818 & 0.63947 & 18.42650 & -2.92636 \\
16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
9.35332 & 3.36780 & 1.72046 & 7.87523 & -1.12766 \\
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
19.32139 & 1.97163 & 0.77882 & 17.41181 & -2.42524 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
11.31169 & 3.77544 & 2.21669 & 10.52481 & -2.72047 \\
16.17890 & 5.13016 & 3.53373 & 14.20896 & 6.15252 \\
12.15632 & 6.65344 & 4.26762 & 11.66199 & 2.26980 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
12.92479 & 5.59260 & 3.41054 & 12.90582 & 1.84898 \\
20.00000 & 1.81729 & 0.52326 & 18.50382 & -2.45657 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
10.55211 & 3.16538 & 2.24324 & 9.81053 & -2.27910 \\
11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\
8.19374 & 5.73812 & 4.94958 & 8.36375 & -1.56120 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
15.66289 & 5.42120 & 3.75579 & 15.27853 & 4.69221 \\
8.33428 & 5.80531 & 4.90545 & 8.33978 & -1.39906 \\
15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\
18.03096 & 3.38688 & 0.00000 & 19.99194 & 9.98438
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
10.49758 & 6.56898 & 3.34530 & 6.00177 & 2.64617 \\
19.59990 & 1.72731 & 0.73080 & 18.00713 & -2.73827 \\
15.54784 & 3.94568 & 1.47767 & 16.09250 & -1.88290 \\
12.71478 & 3.68136 & 1.58864 & 12.33409 & -1.53744 \\
10.78549 & 3.32296 & 0.95272 & 6.88315 & -1.97037 \\
10.99407 & 3.31643 & 0.84091 & 6.73866 & -2.09310 \\
19.80817 & 1.53809 & 0.22015 & 18.57713 & -2.24308 \\
19.24735 & 2.03757 & 0.86380 & 17.23456 & -2.45295 \\
16.25250 & 5.15065 & 3.55364 & 14.26467 & 6.28667 \\
13.42719 & 4.36426 & 2.78913 & 12.12610 & 1.13611 \\
12.16109 & 6.62740 & 4.24264 & 11.70144 & 2.27141 \\
12.24862 & 6.15004 & 3.78465 & 12.42485 & 2.30097 \\
20.24397 & 1.68711 & 0.42370 & 18.69685 & -2.60503 \\
20.14770 & 1.73848 & 0.46298 & 18.62068 & -2.54645 \\
9.12048 & 3.24066 & 4.42394 & 12.71696 & -0.42185 \\
9.52941 & 3.21916 & 3.80104 & 11.88677 & -0.95236 \\
12.56498 & 6.22260 & 3.48394 & 13.00202 & 2.60798 \\
10.74981 & 6.02141 & 4.09255 & 11.07596 & 0.87671 \\
16.12002 & 5.39248 & 3.76853 & 15.13561 & 5.73932 \\
16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\
8.81932 & 5.77975 & 4.83473 & 8.74892 & -0.93662 \\
6.67000 & 5.89300 & 5.14810 & 6.93595 & -2.98582 \\
18.03096 & 3.38688 & 0.00000 & 19.99194 & 9.98438
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
10.49758 & 6.56898 & 3.34530 & 6.00177 & 2.64617 \\
19.59990 & 1.72731 & 0.00000 & 20.00000 & -4.80077 \\
15.54784 & 3.95362 & 1.47767 & 16.09250 & -1.88290 \\
12.71478 & 3.68136 & 1.58864 & 12.33409 & -1.53744 \\
10.78549 & 3.32296 & 0.95272 & 6.88315 & -1.97037 \\
10.99407 & 3.81643 & 1.34091 & 6.73866 & -2.09310 \\
19.80817 & 1.53809 & 0.22015 & 18.57713 & -2.24308 \\
19.24735 & 2.03757 & 0.86380 & 17.23456 & -2.45295 \\
16.25237 & 5.15065 & 3.55364 & 14.26467 & 6.28667 \\
13.42719 & 4.36426 & 2.78913 & 12.12610 & 1.13611 \\
12.16109 & 6.67428 & 4.24264 & 11.70144 & 2.27141 \\
12.24862 & 6.15004 & 3.78465 & 12.42485 & 2.30097 \\
20.00000 & 1.68711 & 0.42370 & 18.69685 & -2.60503 \\
20.00000 & 1.73848 & 0.46298 & 18.62068 & -2.64020 \\
9.12048 & 3.24066 & 4.42394 & 12.71696 & -0.43748 \\
9.52941 & 3.21916 & 3.80104 & 11.88677 & -0.98751 \\
10.52982 & 6.22260 & 3.48394 & 13.00202 & 2.60798 \\
10.74981 & 6.02141 & 4.09255 & 11.07596 & 0.87671 \\
16.12002 & 7.39248 & 3.76853 & 11.13561 & 5.73932 \\
16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\
8.81932 & 5.77975 & 4.83473 & 8.74892 & -0.93662 \\
6.67000 & 5.89288 & 5.14810 & 6.93595 & -2.98582 \\
18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
100.00000 \\
0.22613 \\
0.03333 \\
0.07465 \\
0.03997 \\
0.17597 \\
0.06034 \\
0.29721 \\
0.18131 \\
0.25855 \\
0.21741 \\
0.13894 \\
0.15415 \\
0.17328 \\
0.52123 \\
0.54930 \\
100.00000 \\
0.18648 \\
0.33722 \\
0.13743 \\
100.00000 \\
0.29930 \\
100.00000
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.10143 & 5.71587 & 2.88462 & 8.11714 & 1.69742 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
10.78549 & 3.32296 & 0.95272 & 6.88315 & -1.97037 \\
19.59990 & 1.72731 & 0.00000 & 20.00000 & -4.80077 \\
19.80817 & 1.53809 & 0.22015 & 18.57713 & -2.24308 \\
20.00000 & 1.81729 & 0.52326 & 18.50382 & -2.45657 \\
16.25237 & 5.15065 & 3.55364 & 14.26467 & 6.28667 \\
12.03789 & 8.24586 & 3.96825 & 11.50030 & -2.48468 \\
12.16109 & 6.67428 & 4.24264 & 11.70144 & 2.27141 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
12.92479 & 5.59260 & 3.41054 & 12.90582 & 1.84898 \\
16.17890 & 5.13016 & 3.53373 & 14.20896 & 6.15252 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\
11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\
16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
10.55211 & 3.16538 & 2.24324 & 9.81053 & -2.27910 \\
8.33428 & 5.80531 & 4.90545 & 8.33978 & -1.39906 \\
10.74981 & 6.02141 & 4.09255 & 11.07596 & 0.87671 \\
15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\
18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
0.01584 \\
0.32182 \\
-0.01025 \\
0.01464 \\
0.05692 \\
0.03997 \\
0.22613 \\
0.06034 \\
0.16485 \\
0.18131 \\
0.24581 \\
0.21741 \\
0.12758 \\
0.15150 \\
0.18384 \\
0.35296 \\
0.19887 \\
0.10957 \\
0.13743 \\
0.16194 \\
0.41632 \\
0.25011 \\
0.18648 \\
0.11650 \\
100.00000
\end{pmatrix}}$$

Generation 12

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 12.10143 & 5.71587 & 2.88462 & 8.11714 & 1.69742 \\ 19.59990 & 1.72731 & 0.00000 & 20.00000 & -4.80077 \\ 16.30550 & 4.01637 & 1.44799 & 17.09762 & -1.97529 \\ 20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\ 10.78549 & 3.32296 & 0.95272 & 6.88315 & -1.97037 \\ 19.59990 & 1.72731 & 0.00000 & 20.00000 & -4.80077 \\ 19.80817 & 1.53809 & 0.22015 & 18.57713 & -2.24308 \\ 12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\ 16.25237 & 5.15065 & 3.55364 & 14.26467 & 6.28667 \\ 16.17890 & 5.13016 & 3.53373 & 14.20896 & 6.15252 \\ 12.16109 & 6.67428 & 4.24264 & 11.70144 & 2.27141 \\ 19.59990 & 1.72731 & 0.00000 & 20.00000 & -4.80077 \\ 16.17890 & 5.13016 & 3.53373 & 14.20896 & 6.15252 \\ 16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\ 15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\ 8.33428 & 5.80531 & 4.90545 & 8.33978 & -1.39906 \\ 16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\ 11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\ 10.55211 & 3.16538 & 2.24324 & 9.81053 & -2.27910 \\ 7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\ 10.74981 & 6.02141 & 4.09255 & 11.07596 & 0.87671 \\ 15.81128 & 4.91638 & 2.95991 & 15.83247 & 2.39528 \\ 18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 19.32904 & 1.87139 & 0.10420 & 19.57077 & -4.56604 \\ 16.23229 & 3.51860 & 1.29550 & 14.66334 & -1.88239 \\ 19.80756 & 1.50524 & 0.07542 & 18.93536 & -2.16109 \\ 19.54844 & 1.69103 & 0.17698 & 18.79939 & -2.14734 \\ 11.84634 & 3.13092 & 0.83806 & 8.46182 & -2.31102 \\ 21.36581 & 1.40764 & -0.19087 & 22.62787 & -5.36782 \\ 16.68183 & 3.42271 & 1.67197 & 16.06771 & -0.35943 \\ 13.91692 & 5.08944 & 2.95595 & 13.84841 & 1.30645 \\ 16.24222 & 5.14782 & 3.55089 & 14.25696 & 6.26812 \\ 16.25759 & 5.15210 & 3.55505 & 14.26862 & 6.29619 \\ 17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\ 15.44646 & 4.48944 & 2.36887 & 15.36651 & -0.85203 \\ 16.87989 & 5.40171 & 3.84179 & 15.09813 & 7.41854 \\ 16.31533 & 5.18301 & 3.59368 & 14.38202 & 6.39892 \\ 14.81212 & 5.46399 & 3.96098 & 13.80390 & 4.77703 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 11.30468 & 6.25170 & 3.68711 & 12.00801 & 1.30048 \\ 12.89705 & 5.99474 & 3.71661 & 12.81613 & 3.09006 \\ 10.87972 & 3.14815 & 1.74422 & 9.14544 & -2.70411 \\ 7.53870 & 3.32385 & 6.83333 & 15.92820 & 1.63018 \\ 10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\ 16.33303 & 4.80247 & 2.84316 & 16.32277 & 2.55182 \\ 18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 19.32904 & 1.73711 & 0.10420 & 19.57077 & -4.29748 \\ 16.10681 & 3.51860 & 1.29550 & 14.66334 & -2.00788 \\ 19.80756 & 1.50524 & 0.07542 & 18.93536 & -2.16109 \\ 19.54844 & 1.72228 & 0.20823 & 18.86189 & -2.20984 \\ 11.84634 & 3.13092 & 0.83806 & 8.46182 & -2.31102 \\ 20.00000 & 1.40764 & 0.00000 & 20.00000 & -5.36782 \\ 16.68183 & 3.42271 & 1.67197 & 16.06771 & -0.42193 \\ 13.91692 & 5.08944 & 2.95595 & 13.84841 & 1.30645 \\ 16.24222 & 5.14782 & 3.55089 & 14.25696 & 6.26812 \\ 16.25759 & 5.15210 & 3.55505 & 14.26862 & 6.29619 \\ 17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\ 15.43864 & 4.48944 & 2.36887 & 15.36651 & -0.85984 \\ 17.37989 & 5.40171 & 3.84179 & 15.09813 & 7.41854 \\ 16.31533 & 5.18301 & 3.59368 & 14.39764 & 6.41454 \\ 14.87462 & 5.46399 & 3.96098 & 13.86640 & 4.77703 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 11.30468 & 6.25170 & 3.68711 & 12.00801 & 1.30048 \\ 12.89705 & 5.99474 & 3.71661 & 12.81613 & 3.09006 \\ 10.86117 & 3.14815 & 1.74422 & 9.14544 & -2.72266 \\ 7.53870 & 3.32385 & 6.83333 & 15.86570 & 1.63018 \\ 10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\ 16.20803 & 4.80247 & 2.84316 & 16.19777 & 2.67682 \\ 18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix} 0.19718 \\ 0.13127 \\ -0.00766 \\ 0.01386 \\ 0.03655 \\ 0.38878 \\ 0.09575 \\ 0.14437 \\ 0.18191 \\ 0.18112 \\ 0.19809 \\ 0.20639 \\ 0.18411 \\ 0.18050 \\ 0.29297 \\ 0.00140 \\ 100.00000 \\ 0.13986 \\ 0.32248 \\ 0.38990 \\ 0.18168 \\ 0.03409 \\ 100.00000 \end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\ 19.32904 & 1.73711 & 0.10420 & 19.57077 & -4.29748 \\ 20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\ 19.80756 & 1.50524 & 0.07542 & 18.93536 & -2.16109 \\ 9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\ 11.84634 & 3.13092 & 0.83806 & 8.46182 & -2.31102 \\ 15.43864 & 4.48944 & 2.36887 & 15.36651 & -0.85984 \\ 19.80817 & 1.53809 & 0.22015 & 18.57713 & -2.24308 \\ 20.00000 & 1.81729 & 0.52326 & 18.50382 & -2.45657 \\ 16.25237 & 5.15065 & 3.55364 & 14.26467 & 6.28667 \\ 12.03789 & 8.24586 & 3.96825 & 11.50030 & -2.48468 \\ 17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\ 12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\ 12.92479 & 5.59260 & 3.41054 & 12.90582 & 1.84898 \\ 16.25759 & 5.15210 & 3.55505 & 14.26862 & 6.29619 \\ 7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\ 15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\ 11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\ 16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\ 16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\ 10.86117 & 3.14815 & 1.74422 & 9.14544 & -2.72266 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\ 16.20803 & 4.80247 & 2.84316 & 16.19777 & 2.67682 \\ 18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438 \end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix} 0.01584 \\ 0.19718 \\ -0.01025 \\ -0.00766 \\ 0.05692 \\ 0.03655 \\ 0.20639 \\ 0.06034 \\ 0.16485 \\ 0.18131 \\ 0.24581 \\ 0.19809 \\ 0.12758 \\ 0.15150 \\ 0.18112 \\ 0.35296 \\ 0.19887 \\ 0.10957 \\ 0.13743 \\ 0.16194 \\ 0.32248 \\ 0.00140 \\ 0.18168 \\ 0.03409 \\ 100.00000 \end{pmatrix}}$$

Generation 13

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
19.32904 & 1.73711 & 0.10420 & 19.57077 & -4.29748 \\
15.43864 & 4.48944 & 2.36887 & 15.36651 & -0.85984 \\
19.80756 & 1.50524 & 0.07542 & 18.93536 & -2.16109 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
15.43864 & 4.48944 & 2.36887 & 15.36651 & -0.85984 \\
11.84634 & 3.13092 & 0.83806 & 8.46182 & -2.31102 \\
20.00000 & 1.81729 & 0.52326 & 18.50382 & -2.45657 \\
19.80756 & 1.50524 & 0.07542 & 18.93536 & -2.16109 \\
12.03789 & 8.24586 & 3.96825 & 11.50030 & -2.48468 \\
11.84634 & 3.13092 & 0.83806 & 8.46182 & -2.31102 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
12.92479 & 5.59260 & 3.41054 & 12.90582 & 1.84898 \\
16.25759 & 5.15210 & 3.55505 & 14.26862 & 6.29619 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
15.93077 & 5.40505 & 3.79788 & 14.74749 & 5.84358 \\
7.98460 & 3.30040 & 6.15412 & 15.02294 & 1.05171 \\
16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\
16.20803 & 4.80247 & 2.84316 & 16.19777 & 2.67682 \\
10.86117 & 3.14815 & 1.74422 & 9.14544 & -2.72266 \\
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
16.92919 & 3.43493 & 1.50120 & 16.97731 & -2.17692 \\
16.84367 & 3.49543 & 1.55098 & 16.88489 & -2.10135 \\
20.01281 & 1.35806 & -0.00502 & 19.04307 & -2.17198 \\
19.92060 & 1.42418 & 0.03112 & 18.99468 & -2.16708 \\
14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\
11.84213 & 3.12933 & 0.83626 & 8.45373 & -2.31272 \\
19.94726 & 1.73177 & 0.40052 & 18.62209 & -2.37559 \\
19.97628 & 1.77883 & 0.46806 & 18.55701 & -2.42015 \\
12.01636 & 7.67075 & 3.61629 & 11.15866 & -2.46515 \\
12.05211 & 8.62541 & 4.20052 & 11.72577 & -2.49756 \\
13.08846 & 5.47056 & 3.33303 & 13.00120 & 1.73677 \\
12.56423 & 5.86143 & 3.58131 & 12.69572 & 2.09618 \\
16.02930 & 5.10100 & 3.62677 & 14.28943 & 6.15147 \\
14.73480 & 4.81126 & 4.03346 & 14.40746 & 5.33085 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
10.29770 & 3.91305 & 5.46823 & 14.94276 & 2.44660 \\
17.13002 & 5.46709 & 4.01585 & 14.56241 & 8.82391 \\
16.47647 & 4.99598 & 3.18460 & 15.72163 & 4.46658 \\
11.10387 & 3.26208 & 1.02773 & 7.34605 & -2.36669 \\
11.06497 & 3.24382 & 1.14254 & 7.63440 & -2.42373 \\
13.81120 & 5.72488 & 3.87287 & 13.31786 & 3.90666 \\
8.88403 & 6.25268 & 4.28970 & 9.62331 & -0.68290 \\
18.03096 & 3.64469 & 0.00000 & 20.00000 & 9.98438
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
16.92919 & 3.43493 & 1.50120 & 16.97731 & -2.17692 \\
16.84367 & 3.49543 & 1.55098 & 16.88489 & -2.10135 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
19.92060 & 1.42418 & 0.03112 & 18.99468 & -2.16708 \\
14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
19.94726 & 1.73177 & 0.40052 & 18.62209 & -2.37559 \\
19.97628 & 1.77883 & 0.46806 & 18.55701 & -2.42015 \\
12.01636 & 7.67075 & 3.61629 & 11.15866 & -2.46515 \\
12.05211 & 8.62541 & 4.20052 & 12.72577 & -1.49756 \\
14.08846 & 5.47056 & 3.33303 & 13.00120 & 1.73677 \\
12.56423 & 5.86143 & 3.58131 & 12.69572 & 2.09618 \\
16.02930 & 5.10100 & 3.62677 & 14.28943 & 6.15147 \\
14.73480 & 4.81126 & 4.03346 & 14.40746 & 5.33085 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
10.29770 & 3.91305 & 5.46798 & 14.94276 & 2.44660 \\
17.12611 & 5.46709 & 4.01585 & 14.56241 & 8.82391 \\
16.47647 & 4.99598 & 3.18460 & 15.72163 & 4.46658 \\
11.10387 & 3.26208 & 1.02773 & 7.34605 & -2.36669 \\
11.06497 & 3.24382 & 1.14254 & 7.63440 & -2.42373 \\
13.81120 & 5.72488 & 3.87287 & 13.31786 & 3.90666 \\
8.88403 & 6.25268 & 4.28970 & 9.62331 & -0.68277 \\
18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \begin{pmatrix}
\vec{z}_{\text{mut}} \\
0.19203 \\
0.19414 \\
-0.00925 \\
-0.00835 \\
0.17736 \\
0.03620 \\
0.11781 \\
0.14294 \\
0.22725 \\
0.15570 \\
0.23919 \\
0.13798 \\
0.19662 \\
0.33204 \\
-0.01187 \\
0.51012 \\
0.17377 \\
0.09577 \\
0.10520 \\
0.13826 \\
0.20558 \\
100.00000 \\
100.00000
\end{pmatrix}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
16.92919 & 3.43493 & 1.50120 & 16.97731 & -2.17692 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\
19.80817 & 1.53809 & 0.22015 & 18.57713 & -2.24308 \\
19.94726 & 1.73177 & 0.40052 & 18.62209 & -2.37559 \\
16.25237 & 5.15065 & 3.55364 & 14.26467 & 6.28667 \\
12.01636 & 7.67075 & 3.61629 & 11.15866 & -2.46515 \\
17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\
12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\
12.56423 & 5.86143 & 3.58131 & 12.69572 & 2.09618 \\
16.25759 & 5.15210 & 3.55505 & 14.26862 & 6.29619 \\
14.73480 & 4.81126 & 4.03346 & 14.40746 & 5.33085 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\
16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\
16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\
11.10387 & 3.26208 & 1.02773 & 7.34605 & -2.36669 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\
16.20803 & 4.80247 & 2.84316 & 16.19777 & 2.67682 \\
18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438
\end{pmatrix}} \begin{pmatrix}
\vec{z}_{\text{Pop}} \\
0.01584 \\
0.19203 \\
-0.01025 \\
-0.00925 \\
0.05692 \\
0.03620 \\
0.17736 \\
0.06034 \\
0.11781 \\
0.18131 \\
0.22725 \\
0.19809 \\
0.12758 \\
0.13798 \\
0.18112 \\
0.33204 \\
-0.01187 \\
0.10957 \\
0.13743 \\
0.16194 \\
0.10520 \\
0.00140 \\
0.18168 \\
0.03409 \\
100.00000
\end{pmatrix}$$

Generation 14

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\ 18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438 \\ 20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\ 19.80817 & 1.53809 & 0.22015 & 18.57713 & -2.24308 \\ 9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\ 16.25237 & 5.15065 & 3.55364 & 14.26467 & 6.28667 \\ 14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\ 16.92919 & 3.43493 & 1.50120 & 16.97731 & -2.17692 \\ 19.94726 & 1.73177 & 0.40052 & 18.62209 & -2.37559 \\ 12.56423 & 5.86143 & 3.58131 & 12.69572 & 2.09618 \\ 12.01636 & 7.67075 & 3.61629 & 11.15866 & -2.46515 \\ 17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\ 12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\ 12.56423 & 5.86143 & 3.58131 & 12.69572 & 2.09618 \\ 16.25759 & 5.15210 & 3.55505 & 14.26862 & 6.29619 \\ 16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\ 11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\ 12.25994 & 6.08830 & 3.72542 & 12.51841 & 2.30479 \\ 16.75333 & 5.35268 & 3.78617 & 14.93760 & 7.18998 \\ 18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\ 16.20803 & 4.80247 & 2.84316 & 16.19777 & 2.67682 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 17.82753 & 3.63471 & 0.02168 & 19.59518 & 9.62071 \\ 19.32955 & 3.70841 & -0.13842 & 22.54457 & 12.30591 \\ 19.88951 & 1.46565 & 0.12680 & 18.77184 & -2.21264 \\ 19.89477 & 1.46096 & 0.12077 & 18.78444 & -2.21067 \\ 9.82111 & 3.48911 & 1.84549 & 8.30682 & -0.62140 \\ 13.10549 & 4.33764 & 2.71783 & 11.34943 & 2.90648 \\ 15.06985 & 4.11941 & 1.99759 & 14.47897 & -1.28227 \\ 17.37609 & 3.27041 & 1.38189 & 17.57779 & -2.39195 \\ 16.21477 & 3.81951 & 2.00857 & 15.62602 & -0.11489 \\ 19.10309 & 2.20395 & 0.76421 & 17.94448 & -1.86429 \\ 17.30708 & 3.30823 & 1.26347 & 17.41840 & -2.92494 \\ 12.77469 & 7.04546 & 3.27906 & 12.05588 & -2.53105 \\ 12.30642 & 6.05365 & 3.70341 & 12.54549 & 2.27293 \\ 12.53616 & 5.88235 & 3.59460 & 12.67936 & 2.11542 \\ 16.56623 & 5.27698 & 3.69894 & 14.68511 & 6.85264 \\ 16.82856 & 5.38312 & 3.82125 & 15.03912 & 7.32562 \\ 12.00590 & 6.13631 & 3.70509 & 12.37007 & 2.07178 \\ 12.12263 & 6.11425 & 3.71443 & 12.43823 & 2.17885 \\ 17.24595 & 4.69412 & 2.32632 & 16.88746 & 8.26743 \\ 17.22639 & 4.72027 & 2.38429 & 16.81004 & 8.22465 \\ 15.32714 & 5.56249 & 3.74462 & 14.45456 & 5.31872 \\ 9.01397 & 6.23876 & 4.27870 & 9.72075 & -0.56186 \\ 16.20803 & 4.80247 & 2.84316 & 16.19777 & 2.67682 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 17.82753 & 3.63471 & 0.05391 & 19.65964 & 9.62071 \\ 19.32955 & 3.70841 & 0.11158 & 20.00000 & 10.00000 \\ 19.88951 & 1.46565 & 0.12680 & 19.77184 & -2.21264 \\ 19.89477 & 1.46096 & 0.12077 & 18.78444 & -2.21067 \\ 14.82124 & 3.48911 & 4.34556 & 8.30682 & -0.62140 \\ 13.10549 & 4.33764 & 2.71783 & 11.34943 & 2.90648 \\ 15.06985 & 4.11941 & 1.99759 & 14.47897 & -1.28227 \\ 17.37609 & 3.27041 & 1.38189 & 17.57779 & -2.39195 \\ 16.21477 & 3.81951 & 2.00857 & 15.62602 & -0.11489 \\ 19.10309 & 2.20395 & 0.76421 & 17.95241 & -1.86429 \\ 17.30708 & 3.30823 & 1.26347 & 17.41840 & -2.92494 \\ 13.14969 & 7.23296 & 3.27906 & 12.05588 & -2.90605 \\ 12.05251 & 6.18060 & 3.57645 & 12.54549 & 2.52684 \\ 12.53616 & 5.88235 & 3.59460 & 12.67936 & 1.86542 \\ 16.57013 & 5.27698 & 3.69894 & 14.68511 & 6.85264 \\ 16.82856 & 5.38312 & 3.82125 & 15.03912 & 7.32562 \\ 12.00590 & 6.13631 & 3.70460 & 12.37007 & 2.07178 \\ 12.12263 & 6.11425 & 3.71443 & 12.43823 & 2.17885 \\ 17.24595 & 4.66281 & 2.32632 & 16.88746 & 8.26743 \\ 17.22639 & 4.84552 & 2.38429 & 16.55955 & 8.22465 \\ 15.32714 & 5.56249 & 3.74462 & 14.45456 & 5.31872 \\ 9.01397 & 6.23876 & 4.27870 & 9.72075 & -0.56186 \\ 16.20803 & 4.79845 & 2.84316 & 16.19777 & 2.67682 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix} 100.00000 \\ 100.00000 \\ 0.02210 \\ 0.02312 \\ 2.01811 \\ 0.16921 \\ 0.18285 \\ 0.19840 \\ 0.13578 \\ 0.11913 \\ 0.20848 \\ 0.27450 \\ 100.00000 \\ 0.14430 \\ 0.17127 \\ 0.15731 \\ 0.11584 \\ 0.12087 \\ 100.00000 \\ 100.00000 \\ 0.17673 \\ 100.00000 \\ 0.07409 \end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\ 16.92919 & 3.43493 & 1.50120 & 16.97731 & -2.17692 \\ 20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\ 20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\ 9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\ 11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\ 14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\ 19.89477 & 1.46096 & 0.12077 & 18.78444 & -2.21067 \\ 19.94726 & 1.73177 & 0.40052 & 18.62209 & -2.37559 \\ 13.10549 & 4.33764 & 2.71783 & 11.34943 & 2.90648 \\ 17.30708 & 3.30823 & 1.26347 & 17.41840 & -2.92494 \\ 17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\ 12.12263 & 6.11425 & 3.71443 & 12.43823 & 2.17885 \\ 19.10309 & 2.20395 & 0.76421 & 17.95241 & -1.86429 \\ 16.57013 & 5.27698 & 3.69894 & 14.68511 & 6.85264 \\ 14.73480 & 4.81126 & 4.03346 & 14.40746 & 5.33085 \\ 17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\ 11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\ 16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\ 16.82856 & 5.38312 & 3.82125 & 15.03912 & 7.32562 \\ 11.10387 & 3.26208 & 1.02773 & 7.34605 & -2.36669 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\ 16.20803 & 4.79845 & 2.84316 & 16.19777 & 2.67682 \\ 18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438 \end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix} 0.01584 \\ 0.19203 \\ -0.01025 \\ -0.00925 \\ 0.05692 \\ 0.03620 \\ 0.17736 \\ 0.02312 \\ 0.11781 \\ 0.16921 \\ 0.20848 \\ 0.19809 \\ 0.12087 \\ 0.11913 \\ 0.17127 \\ 0.33204 \\ -0.01187 \\ 0.10957 \\ 0.13743 \\ 0.15731 \\ 0.10520 \\ 0.00140 \\ 0.18168 \\ 0.07409 \\ 100.00000 \end{pmatrix}}$$

Generation 15

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 16.92919 & 3.43493 & 1.50120 & 16.97731 & -2.17692 \\ 11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\ 9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\ 18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438 \\ 14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\ 16.92919 & 3.43493 & 1.50120 & 16.97731 & -2.17692 \\ 19.94726 & 1.73177 & 0.40052 & 18.62209 & -2.37559 \\ 19.10309 & 2.20395 & 0.76421 & 17.95241 & -1.86429 \\ 17.30708 & 3.30823 & 1.26347 & 17.41840 & -2.92494 \\ 11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\ 12.12263 & 6.11425 & 3.71443 & 12.43823 & 2.17885 \\ 19.10309 & 2.20395 & 0.76421 & 17.95241 & -1.86429 \\ 16.57013 & 5.27698 & 3.69894 & 14.68511 & 6.85264 \\ 16.82856 & 5.38312 & 3.82125 & 15.03912 & 7.32562 \\ 17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\ 17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\ 16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\ 11.92477 & 6.15164 & 3.69860 & 12.32270 & 1.99736 \\ 11.10387 & 3.26208 & 1.02773 & 7.34605 & -2.36669 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 14.71754 & 3.38658 & 1.20225 & 12.96443 & -2.19262 \\ 15.22387 & 3.39765 & 1.27069 & 13.88312 & -2.18903 \\ 8.13611 & 3.32869 & 1.96107 & 6.17488 & -2.68130 \\ 19.48374 & 3.69109 & -0.28793 & 22.02367 & 11.84397 \\ 16.94619 & 3.42867 & 1.49666 & 17.00015 & -2.18510 \\ 15.33197 & 4.02292 & 1.92761 & 14.83117 & -1.40839 \\ 19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\ 19.29415 & 2.09708 & 0.68190 & 18.10398 & -1.98001 \\ 15.36480 & 3.24465 & 1.11164 & 14.23234 & -2.70735 \\ 18.39069 & 3.34371 & 1.34818 & 19.19593 & -3.04633 \\ 20.84183 & 1.22995 & 0.02935 & 19.32592 & -2.87139 \\ 17.72917 & 2.97359 & 1.34488 & 16.86709 & -1.06851 \\ 16.77286 & 5.36024 & 3.79488 & 14.96281 & 7.22366 \\ 16.85320 & 5.39324 & 3.83290 & 15.07286 & 7.37069 \\ 17.64401 & 2.49794 & 0.71616 & 18.40631 & -4.75015 \\ 17.65881 & 3.01174 & 1.10223 & 17.84171 & -2.97731 \\ 13.33817 & 5.92355 & 3.72478 & 13.03999 & 3.58580 \\ 13.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\ 17.76738 & 5.35673 & 3.60032 & 16.36400 & 7.66757 \\ 14.41313 & 4.30233 & 2.30534 & 11.82458 & 2.61657 \\ 12.17477 & 5.90018 & 4.01131 & 12.09081 & 2.38235 \\ 8.76504 & 6.26543 & 4.29976 & 9.53409 & -0.79373 \\ 18.03096 & 3.64469 & 0.00000 & 19.99463 & 9.98438 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 14.71754 & 3.38658 & 1.20225 & 12.96443 & -2.19262 \\ 17.22387 & 3.39765 & 1.27069 & 13.88312 & -2.18903 \\ 6.07361 & 3.32869 & 1.96107 & 6.17488 & -4.74380 \\ 19.48374 & 3.69109 & 0.00000 & 20.00000 & 10.00000 \\ 16.94619 & 3.42867 & 1.49666 & 17.00015 & -2.18510 \\ 15.33197 & 4.02487 & 1.92956 & 14.82726 & -1.40839 \\ 19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\ 19.29415 & 2.09708 & 0.68190 & 18.10398 & -1.98001 \\ 14.80230 & 3.24465 & 1.11164 & 14.79484 & -2.70735 \\ 18.39069 & 3.34371 & 1.34818 & 19.19593 & -3.04633 \\ 20.00000 & 1.22995 & 0.02935 & 19.32592 & -2.87139 \\ 17.72893 & 2.97359 & 1.34488 & 16.86709 & -1.06851 \\ 16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\ 16.85320 & 5.39324 & 3.83290 & 15.07286 & 7.37069 \\ 17.64401 & 2.49794 & 0.71616 & 18.40631 & -4.75015 \\ 17.65979 & 3.01174 & 1.10223 & 17.84269 & -2.97731 \\ 13.33817 & 5.92355 & 3.72478 & 13.03999 & 3.58580 \\ 12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\ 17.76738 & 5.35673 & 3.60032 & 16.36400 & 7.66757 \\ 18.41313 & 4.30233 & 2.30534 & 11.82458 & 2.61657 \\ 12.17868 & 5.90018 & 4.01131 & 12.09081 & 2.38235 \\ 8.76504 & 6.26543 & 4.29976 & 9.53409 & -0.79398 \\ 18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix} 0.10794 \\ 0.25574 \\ 0.17694 \\ 100.00000 \\ 0.19458 \\ 0.18861 \\ 0.11534 \\ 0.11980 \\ 0.09163 \\ 0.24982 \\ 0.09727 \\ 0.12540 \\ 0.05034 \\ 0.15456 \\ 0.33083 \\ 0.20200 \\ 0.14995 \\ 0.09352 \\ -0.00841 \\ 0.39382 \\ 0.20932 \\ 100.00000 \\ 100.00000 \end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\ 15.33197 & 4.02487 & 1.92956 & 14.82726 & -1.40839 \\ 20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\ 20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\ 9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\ 11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\ 14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\ 19.89477 & 1.46096 & 0.12077 & 18.78444 & -2.21067 \\ 19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\ 13.10549 & 4.33764 & 2.71783 & 11.34943 & 2.90648 \\ 14.80230 & 3.24465 & 1.11164 & 14.79484 & -2.70735 \\ 17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\ 20.00000 & 1.22995 & 0.02935 & 19.32592 & -2.87139 \\ 19.10309 & 2.20395 & 0.76421 & 17.95241 & -1.86429 \\ 16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\ 14.73480 & 4.81126 & 4.03346 & 14.40746 & 5.33085 \\ 17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\ 12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\ 16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\ 16.85320 & 5.39324 & 3.83290 & 15.07286 & 7.37069 \\ 17.76738 & 5.35673 & 3.60032 & 16.36400 & 7.66757 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\ 16.20803 & 4.79845 & 2.84316 & 16.19777 & 2.67682 \\ 18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474 \end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix} 0.01584 \\ 0.18861 \\ -0.01025 \\ -0.00925 \\ 0.05692 \\ 0.03620 \\ 0.17736 \\ 0.02312 \\ 0.11534 \\ 0.16921 \\ 0.09163 \\ 0.19809 \\ 0.09727 \\ 0.11913 \\ 0.05034 \\ 0.33204 \\ -0.01187 \\ 0.09352 \\ 0.13743 \\ 0.15456 \\ -0.00841 \\ 0.00140 \\ 0.18168 \\ 0.07409 \\ 100.00000 \end{pmatrix}}$$

Generation 16

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
15.33197 & 4.02487 & 1.92956 & 14.82726 & -1.40839 \\
14.76092 & 4.23314 & 2.08007 & 14.06387 & -1.13362 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
16.20803 & 4.79845 & 2.84316 & 16.19777 & 2.67682 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
14.80230 & 3.24465 & 1.11164 & 14.79484 & -2.70735 \\
19.89477 & 1.46096 & 0.12077 & 18.78444 & -2.21067 \\
20.00000 & 1.22995 & 0.02935 & 19.32592 & -2.87139 \\
13.10549 & 4.33764 & 2.71783 & 11.34943 & 2.90648 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
17.65899 & 3.01806 & 1.10697 & 17.83477 & -2.95552 \\
14.80230 & 3.24465 & 1.11164 & 14.79484 & -2.70735 \\
19.10309 & 2.20395 & 0.76421 & 17.95241 & -1.86429 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
14.73480 & 4.81126 & 4.03346 & 14.40746 & 5.33085 \\
17.76738 & 5.35673 & 3.60032 & 16.36400 & 7.66757 \\
12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
17.76738 & 5.35673 & 3.60032 & 16.36400 & 7.66757 \\
16.85320 & 5.39324 & 3.83290 & 15.07286 & 7.37069 \\
10.46741 & 6.08307 & 4.15575 & 10.81058 & 0.79199 \\
12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\
18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
15.42587 & 3.99063 & 1.90482 & 14.95279 & -1.45357 \\
14.73108 & 4.24402 & 2.08793 & 14.02398 & -1.11927 \\
18.11567 & 3.06767 & 1.41284 & 17.62917 & 0.23751 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.76234 \\
12.40620 & 3.15130 & 0.88874 & 9.66214 & -2.38792 \\
15.27960 & 3.26324 & 1.15604 & 15.81727 & -2.77098 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.96318 & 1.31079 & 0.06134 & 19.13644 & -2.64018 \\
13.34537 & 4.40453 & 2.78828 & 11.65987 & 3.18886 \\
12.49406 & 4.16715 & 2.53826 & 10.55814 & 2.18672 \\
17.72291 & 3.01299 & 1.10687 & 17.90279 & -2.96107 \\
14.89601 & 3.23722 & 1.11149 & 14.89456 & -2.71549 \\
19.25246 & 2.12040 & 0.69986 & 18.07090 & -1.95476 \\
19.24568 & 2.12419 & 0.70278 & 18.06553 & -1.95066 \\
15.96553 & 5.03263 & 3.85768 & 15.20149 & 6.27918 \\
18.35098 & 5.46171 & 3.51697 & 16.74052 & 8.11726 \\
13.81929 & 5.88286 & 3.61797 & 13.53040 & 4.60218 \\
18.91871 & 5.88520 & 3.15379 & 15.03116 & 7.63586 \\
17.20206 & 5.37931 & 3.74415 & 15.56557 & 7.48399 \\
17.28498 & 5.37600 & 3.72305 & 15.68269 & 7.51091 \\
12.04786 & 5.93384 & 3.83896 & 12.56399 & 3.08202 \\
11.84517 & 5.95298 & 3.87959 & 12.33912 & 2.78832 \\
18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
15.42587 & 4.24111 & 1.90482 & 14.95279 & -0.95260 \\
14.73108 & 4.24402 & 2.08793 & 14.02398 & -1.11927 \\
18.11567 & 3.06767 & 1.41284 & 17.62917 & 0.23751 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.76039 \\
12.40620 & 3.15130 & 0.88874 & 9.66214 & -2.45237 \\
15.27960 & 3.26324 & 1.15604 & 15.81727 & -2.77098 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.96318 & 1.31030 & 0.06134 & 19.13644 & -2.64018 \\
13.34537 & 4.40453 & 2.78828 & 11.65987 & 3.15761 \\
12.49406 & 4.16715 & 2.53826 & 10.55814 & 2.18672 \\
17.72291 & 3.01299 & 1.10687 & 17.90169 & -2.96107 \\
14.89210 & 3.23526 & 1.11149 & 14.89456 & -2.71549 \\
19.25246 & 2.12040 & 0.69986 & 18.07090 & -1.95476 \\
19.24568 & 2.12419 & 0.70278 & 18.06553 & -1.95066 \\
15.96553 & 5.03263 & 3.85768 & 15.20931 & 6.27918 \\
18.35098 & 5.46171 & 3.51697 & 16.74052 & 8.11726 \\
17.81929 & 5.88286 & 3.61797 & 13.53040 & 4.60218 \\
18.91871 & 5.88520 & 3.15623 & 15.03116 & 7.63586 \\
17.20206 & 5.37931 & 3.72755 & 15.56557 & 7.48399 \\
17.28498 & 5.37600 & 3.72305 & 15.68269 & 7.51091 \\
12.04786 & 5.93384 & 3.83896 & 12.56399 & 3.08202 \\
11.84517 & 5.95298 & 3.75459 & 12.33912 & 2.78832 \\
18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
0.10558 \\
0.17703 \\
-0.00613 \\
0.03764 \\
0.05066 \\
0.10294 \\
0.01271 \\
0.07272 \\
0.16821 \\
0.17444 \\
0.18908 \\
0.09641 \\
0.11336 \\
0.12056 \\
0.21179 \\
100.00000 \\
0.43098 \\
100.00000 \\
0.08219 \\
0.07527 \\
0.12012 \\
0.10701 \\
100.00000
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
15.42587 & 4.24111 & 1.90482 & 14.95279 & -0.95260 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
14.73108 & 4.24402 & 2.08793 & 14.02398 & -1.11927 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
13.34537 & 4.40453 & 2.78828 & 11.65987 & 3.15761 \\
14.80230 & 3.24465 & 1.11164 & 14.79484 & -2.70735 \\
17.72291 & 3.01299 & 1.10687 & 17.90169 & -2.96107 \\
19.96318 & 1.31030 & 0.06134 & 19.13644 & -2.64018 \\
19.25246 & 2.12040 & 0.69986 & 18.07090 & -1.95476 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
15.96553 & 5.03263 & 3.85768 & 15.20931 & 6.27918 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\
16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\
17.28498 & 5.37600 & 3.72305 & 15.68269 & 7.51091 \\
17.76738 & 5.35673 & 3.60032 & 16.36400 & 7.66757 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
12.04786 & 5.93384 & 3.83896 & 12.56399 & 3.08202 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.76039 \\
18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
0.01584 \\
0.10558 \\
-0.01025 \\
-0.00925 \\
0.05692 \\
0.03620 \\
0.17703 \\
0.01271 \\
0.11534 \\
0.16821 \\
0.09163 \\
0.18908 \\
0.07272 \\
0.11336 \\
0.05034 \\
0.21179 \\
-0.01187 \\
0.09352 \\
0.13743 \\
0.07527 \\
-0.00841 \\
0.00140 \\
0.12012 \\
0.03764 \\
100.00000
\end{pmatrix}}$$

Generation 17

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
15.42587 & 4.24111 & 1.90482 & 14.95279 & -0.95260 \\
14.73108 & 4.24402 & 2.08793 & 14.02398 & -1.11927 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474 \\
14.73108 & 4.24402 & 2.08793 & 14.02398 & -1.11927 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
19.25246 & 2.12040 & 0.69986 & 18.07090 & -1.95476 \\
14.80230 & 3.24465 & 1.11164 & 14.79484 & -2.70735 \\
15.96553 & 5.03263 & 3.85768 & 15.20931 & 6.27918 \\
19.96318 & 1.31030 & 0.06134 & 19.13644 & -2.64018 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
19.25246 & 2.12040 & 0.69986 & 18.07090 & -1.95476 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
17.72291 & 3.01299 & 1.10687 & 17.90169 & -2.96107 \\
16.95383 & 5.34009 & 3.79176 & 14.87492 & 7.64924 \\
12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\
17.76738 & 5.35673 & 3.60032 & 16.36400 & 7.66757 \\
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
12.04786 & 5.93384 & 3.83896 & 12.56399 & 3.08202 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
15.19470 & 4.24208 & 1.96574 & 14.64376 & -1.00805 \\
14.66295 & 4.24431 & 2.10589 & 13.93290 & -1.13561 \\
14.74460 & 3.53974 & 0.65132 & 15.40493 & 5.77798 \\
19.79784 & 3.70112 & -0.35018 & 22.46280 & 12.24647 \\
16.39060 & 3.35848 & 1.45833 & 15.53125 & -1.43748 \\
16.23920 & 3.43927 & 1.51577 & 15.39374 & -1.40845 \\
19.24817 & 2.12280 & 0.70171 & 18.06750 & -1.95216 \\
19.24734 & 2.12326 & 0.70206 & 18.06685 & -1.95166 \\
15.18067 & 3.82623 & 2.00485 & 14.92965 & 0.21571 \\
15.62374 & 4.50728 & 4.14457 & 15.08752 & 3.63869 \\
19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\
19.90285 & 1.44308 & 0.11375 & 18.82601 & -2.26140 \\
16.75927 & 5.37800 & 3.81185 & 16.08480 & 7.27397 \\
16.48172 & 5.74065 & 4.15828 & 15.86370 & 8.30135 \\
17.72604 & 3.49495 & 1.46839 & 17.36178 & -1.30009 \\
17.72904 & 3.95665 & 1.81472 & 16.84455 & 0.29108 \\
14.17740 & 5.68536 & 3.75212 & 13.78907 & 5.24463 \\
17.54805 & 5.26619 & 3.80024 & 15.10731 & 8.16388 \\
18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\
13.73398 & 4.09854 & 1.83449 & 10.35749 & 1.59143 \\
12.15520 & 5.92196 & 3.83340 & 12.63356 & 3.16629 \\
18.83141 & 5.18282 & 3.48736 & 16.96051 & 8.40779 \\
18.03096 & 3.64469 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
15.19470 & 4.16383 & 1.96574 & 14.64376 & -1.00805 \\
14.66295 & 4.24431 & 2.10589 & 13.93290 & -1.13561 \\
14.74460 & 2.53766 & 0.65132 & 15.40493 & 5.77798 \\
19.79784 & 3.70112 & 0.00000 & 20.00000 & 10.00000 \\
16.39060 & 3.35848 & 1.45833 & 15.53125 & -1.43748 \\
16.23920 & 3.43927 & 1.51577 & 15.39374 & -1.40845 \\
19.24817 & 2.12280 & 0.70171 & 18.06750 & -1.95216 \\
19.24734 & 2.12326 & 0.70206 & 18.06685 & -1.95166 \\
15.18067 & 3.82623 & 2.00485 & 14.92965 & 0.21571 \\
15.62374 & 4.50728 & 4.14457 & 15.08752 & 3.63869 \\
19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\
19.90285 & 1.44308 & 0.11375 & 18.82601 & -2.26140 \\
16.75927 & 5.37800 & 3.81185 & 16.08480 & 7.27397 \\
16.46512 & 5.73235 & 4.15828 & 15.86370 & 8.30135 \\
17.72604 & 3.49495 & 1.46839 & 17.36568 & -1.30009 \\
17.72904 & 3.95665 & 1.81472 & 17.84455 & -0.70892 \\
14.17740 & 5.68536 & 3.75212 & 13.78907 & 5.24463 \\
17.54805 & 5.26619 & 3.80024 & 15.10731 & 8.16388 \\
18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\
13.73398 & 4.09854 & 1.83449 & 10.35749 & 1.59143 \\
12.15520 & 5.92196 & 3.80215 & 12.63356 & 3.16629 \\
18.83141 & 5.18282 & 3.48736 & 16.96051 & 8.40779 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \begin{pmatrix}
\vec{z}_{\text{mut}} \\
0.14121 \\
0.18022 \\
100.00000 \\
100.00000 \\
0.13949 \\
0.14602 \\
0.12061 \\
0.12060 \\
0.08182 \\
0.59518 \\
0.05746 \\
0.03124 \\
0.05561 \\
0.06079 \\
0.08821 \\
0.07945 \\
0.13706 \\
0.13891 \\
-0.04958 \\
0.06742 \\
0.11578 \\
100.00000 \\
100.00000
\end{pmatrix}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
15.42587 & 4.24111 & 1.90482 & 14.95279 & -0.95260 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
16.39060 & 3.35848 & 1.45833 & 15.53125 & -1.43748 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
13.34537 & 4.40453 & 2.78828 & 11.65987 & 3.15761 \\
15.18067 & 3.82623 & 2.00485 & 14.92965 & 0.21571 \\
17.72904 & 3.95665 & 1.81472 & 17.84455 & -0.70892 \\
19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\
16.46512 & 5.73235 & 4.15828 & 15.86370 & 8.30135 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
15.96553 & 5.03263 & 3.85768 & 15.20931 & 6.27918 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\
14.17740 & 5.68536 & 3.75212 & 13.78907 & 5.24463 \\
17.28498 & 5.37600 & 3.72305 & 15.68269 & 7.51091 \\
18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
12.15520 & 5.92196 & 3.80215 & 12.63356 & 3.16629 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.76039 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \begin{pmatrix}
\vec{z}_{\text{Pop}} \\
0.01584 \\
0.10558 \\
-0.01025 \\
-0.00925 \\
0.05692 \\
0.03620 \\
0.13949 \\
0.01271 \\
0.11534 \\
0.16821 \\
0.08182 \\
0.07945 \\
0.05746 \\
0.06079 \\
0.05034 \\
0.21179 \\
-0.01187 \\
0.09352 \\
0.13706 \\
0.07527 \\
-0.04958 \\
0.00140 \\
0.11578 \\
0.03764 \\
100.00000
\end{pmatrix}$$

Generation 18

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
11.20555 & 3.30981 & 0.72754 & 6.59217 & -2.21754 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
15.42587 & 4.24111 & 1.90482 & 14.95279 & -0.95260 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
16.39060 & 3.35848 & 1.45833 & 15.53125 & -1.43748 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
13.34537 & 4.40453 & 2.78828 & 11.65987 & 3.15761 \\
15.18067 & 3.82623 & 2.00485 & 14.92965 & 0.21571 \\
11.84213 & 3.12933 & 0.83626 & 8.45385 & -2.31272 \\
19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\
12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
17.28498 & 5.37600 & 3.72305 & 15.68269 & 7.51091 \\
17.74156 & 5.88466 & 3.26094 & 14.68472 & 6.93557 \\
12.59387 & 5.88229 & 3.72952 & 13.16976 & 3.87317 \\
17.28498 & 5.37600 & 3.72305 & 15.68269 & 7.51091 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
15.42587 & 4.24111 & 1.90482 & 14.95279 & -0.95260 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.76039
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
11.26326 & 3.29345 & 0.73740 & 6.76095 & -2.22617 \\
11.61535 & 3.19363 & 0.79753 & 7.79061 & -2.27881 \\
14.29819 & 4.94962 & 2.37442 & 13.94606 & -0.65215 \\
20.28810 & 1.18623 & -0.11997 & 19.29355 & -2.24806 \\
10.82960 & 3.22609 & 1.19553 & 8.21687 & -1.83099 \\
10.28139 & 3.27848 & 1.39005 & 8.08856 & -1.57017 \\
11.11626 & 3.09276 & 0.73699 & 7.32439 & -2.45240 \\
14.19548 & 3.24789 & 1.15812 & 12.11566 & -1.85988 \\
17.39169 & 2.84189 & 1.35861 & 16.05473 & -0.34730 \\
18.61666 & 2.36882 & 0.92579 & 17.38522 & -1.40838 \\
14.12602 & 3.60608 & 1.63569 & 12.88395 & -0.58302 \\
14.31704 & 3.64595 & 1.70255 & 13.25446 & -0.43835 \\
14.75553 & 4.55151 & 2.65594 & 14.89164 & 1.99785 \\
19.35473 & 1.72010 & 0.37179 & 18.55516 & -1.99212 \\
16.74464 & 5.35937 & 3.79884 & 16.11838 & 7.20784 \\
17.13251 & 5.37131 & 3.74444 & 15.80562 & 7.42540 \\
17.76084 & 5.88467 & 3.25918 & 14.69040 & 6.94703 \\
13.37104 & 5.88265 & 3.65877 & 13.39848 & 4.33551 \\
17.90909 & 4.34625 & 0.60820 & 19.29054 & 9.58062 \\
18.03099 & 4.14512 & -0.00017 & 19.99519 & 9.98485 \\
16.47736 & 4.75229 & 2.68745 & 15.53936 & 3.05512 \\
15.69985 & 4.37431 & 2.10874 & 15.10563 & 0.09165 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.76039
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
11.26326 & 3.31298 & 0.73740 & 6.76095 & -2.22617 \\
11.61339 & 3.19363 & 0.79655 & 7.79061 & -2.27881 \\
14.29819 & 4.94962 & 2.37442 & 13.94410 & -0.65215 \\
20.00000 & 1.18623 & 0.00000 & 19.29355 & -2.24806 \\
10.82960 & 3.22218 & 1.19553 & 8.21687 & -1.83099 \\
10.28139 & 3.27848 & 1.39005 & 8.08758 & -1.57017 \\
11.11626 & 3.09276 & 0.73699 & 7.32439 & -2.45240 \\
14.19548 & 3.24789 & 1.15812 & 12.11566 & -1.85988 \\
18.39169 & 2.84189 & 1.35861 & 15.05473 & -0.34730 \\
18.61666 & 2.36882 & 0.92579 & 17.38522 & -1.40838 \\
14.16509 & 3.58655 & 1.63569 & 12.88395 & -0.58302 \\
13.31704 & 3.64595 & 1.70255 & 13.25446 & -0.43835 \\
13.75553 & 5.05151 & 2.65594 & 15.89164 & 1.99785 \\
19.35473 & 1.72010 & 0.34054 & 18.55516 & -1.99212 \\
16.74464 & 5.35950 & 3.79884 & 16.11838 & 7.20784 \\
17.13251 & 5.37131 & 3.74444 & 16.80758 & 7.42540 \\
17.76084 & 5.88369 & 3.25918 & 14.69040 & 6.94703 \\
13.37104 & 5.88265 & 3.90877 & 12.89848 & 4.33551 \\
17.90909 & 4.34698 & 0.60820 & 19.28907 & 9.58062 \\
18.03099 & 4.14512 & 0.00000 & 19.99519 & 9.98485 \\
16.47736 & 4.75229 & 2.68745 & 15.53936 & 3.21137 \\
15.69985 & 4.37431 & 2.10874 & 15.10563 & 0.09165 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.77211
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
0.01493 \\
0.02683 \\
0.13213 \\
0.01930 \\
0.05862 \\
0.06139 \\
0.01969 \\
0.10191 \\
0.14936 \\
0.10524 \\
0.09063 \\
0.02019 \\
100.00000 \\
0.03892 \\
0.05782 \\
-0.01454 \\
-0.01959 \\
0.17494 \\
100.00000 \\
100.00000 \\
0.04627 \\
0.07406 \\
0.04836
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.26326 & 3.31298 & 0.73740 & 6.76095 & -2.22617 \\
15.69985 & 4.37431 & 2.10874 & 15.10563 & 0.09165 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
13.31704 & 3.64595 & 1.70255 & 13.25446 & -0.43835 \\
11.11626 & 3.09276 & 0.73699 & 7.32439 & -2.45240 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
18.61666 & 2.36882 & 0.92579 & 17.38522 & -1.40838 \\
15.18067 & 3.82623 & 2.00485 & 14.92965 & 0.21571 \\
17.72904 & 3.95665 & 1.81472 & 17.84455 & -0.70892 \\
19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\
16.46512 & 5.73235 & 4.15828 & 15.86370 & 8.30135 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
15.96553 & 5.03263 & 3.85768 & 15.20931 & 6.27918 \\
17.76084 & 5.88369 & 3.25918 & 14.69040 & 6.94703 \\
19.35473 & 1.72010 & 0.34054 & 18.55516 & -1.99212 \\
14.17740 & 5.68536 & 3.75212 & 13.78907 & 5.24463 \\
17.13251 & 5.37131 & 3.74444 & 16.80758 & 7.42540 \\
18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
12.15520 & 5.92196 & 3.80215 & 12.63356 & 3.16629 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.77211 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
0.01493 \\
0.07406 \\
-0.01025 \\
-0.00925 \\
0.05692 \\
0.02019 \\
0.01969 \\
0.01271 \\
0.11534 \\
0.10524 \\
0.08182 \\
0.07945 \\
0.05746 \\
0.06079 \\
0.05034 \\
0.21179 \\
-0.01959 \\
0.03892 \\
0.13706 \\
-0.01454 \\
-0.04958 \\
0.00140 \\
0.11578 \\
0.04836 \\
100.00000
\end{pmatrix}}$$

Generation 19

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
15.69985 & 4.37431 & 2.10874 & 15.10563 & 0.09165 \\
11.26326 & 3.31298 & 0.73740 & 6.76095 & -2.22617 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
13.31704 & 3.64595 & 1.70255 & 13.25446 & -0.43835 \\
9.35025 & 3.36746 & 1.72043 & 7.87062 & -1.12717 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
18.61666 & 2.36882 & 0.92579 & 17.38522 & -1.40838 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\
16.46512 & 5.73235 & 4.15828 & 15.86370 & 8.30135 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
18.61666 & 2.36882 & 0.92579 & 17.38522 & -1.40838 \\
17.76084 & 5.88369 & 3.25918 & 14.69040 & 6.94703 \\
17.72904 & 3.95665 & 1.81472 & 17.84455 & -0.70892 \\
14.17740 & 5.68536 & 3.75212 & 13.78907 & 5.24463 \\
17.13251 & 5.37131 & 3.74444 & 16.80758 & 7.42540 \\
18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\
17.13251 & 5.37131 & 3.74444 & 16.80758 & 7.42540 \\
12.15520 & 5.92196 & 3.80215 & 12.63356 & 3.16629 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
16.78077 & 4.63288 & 2.44285 & 17.13871 & 0.65637 \\
16.59571 & 4.58862 & 2.38565 & 16.79065 & 0.55969 \\
17.89773 & 1.75472 & 0.33961 & 16.83762 & -1.96573 \\
7.52197 & 3.71242 & 2.01579 & 5.95261 & -0.94780 \\
10.08096 & 3.41876 & 1.71714 & 8.86236 & -1.00028 \\
8.58583 & 3.31380 & 1.72388 & 6.83312 & -1.25991 \\
19.56535 & 1.80535 & 0.41367 & 18.38663 & -2.02756 \\
19.24057 & 2.12502 & 0.70279 & 18.06245 & -1.94587 \\
17.85937 & 3.59748 & 2.10420 & 16.85555 & 2.13702 \\
17.74646 & 3.78066 & 2.27990 & 16.77658 & 2.66562 \\
20.23947 & 0.98362 & -0.26956 & 19.29338 & -3.42839 \\
20.51228 & 0.64039 & -0.58960 & 19.54127 & -4.27621 \\
18.61837 & 2.36606 & 0.92315 & 17.38641 & -1.41635 \\
16.87430 & 5.19566 & 3.63703 & 16.16658 & 6.74875 \\
17.76534 & 6.15657 & 3.46373 & 14.24375 & 8.03116 \\
17.75981 & 5.82128 & 3.21240 & 14.79256 & 6.69906 \\
15.87021 & 5.50546 & 3.74772 & 15.51819 & 6.49386 \\
15.56983 & 5.53738 & 3.74850 & 15.21137 & 6.27219 \\
17.27458 & 5.39004 & 3.75586 & 16.85203 & 7.55348 \\
16.90063 & 5.34072 & 3.72579 & 16.73501 & 7.21632 \\
15.47721 & 5.55417 & 3.64891 & 14.78660 & 5.77440 \\
13.05150 & 5.82273 & 3.76080 & 13.21447 & 3.86998 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
16.78077 & 4.63288 & 2.44285 & 17.13871 & 0.65637 \\
16.61134 & 4.59643 & 2.38565 & 16.79065 & 0.55969 \\
17.89773 & 1.75472 & 0.33961 & 16.83762 & -1.96573 \\
7.52197 & 3.71242 & 2.01579 & 5.95261 & -0.94780 \\
6.04971 & 3.41876 & 1.71714 & 4.83111 & -1.00028 \\
8.58583 & 3.31380 & 1.72388 & 6.83312 & -1.25991 \\
19.56535 & 1.80535 & 0.41367 & 18.38663 & -2.02756 \\
19.23666 & 2.12502 & 0.70279 & 18.06245 & -1.94587 \\
17.85937 & 3.59748 & 2.10420 & 16.85555 & 2.13702 \\
17.74646 & 3.78066 & 2.27990 & 16.77658 & 2.68125 \\
20.00000 & 0.84300 & 0.00000 & 19.57463 & -3.42839 \\
20.00000 & 0.64039 & 0.00000 & 19.53932 & -4.27816 \\
18.61837 & 2.36606 & 0.92315 & 17.38641 & -1.41635 \\
16.87430 & 5.19566 & 3.63703 & 16.22908 & 6.81125 \\
17.76534 & 6.15657 & 3.46373 & 14.24375 & 8.03116 \\
17.75981 & 5.82128 & 3.21240 & 14.79256 & 6.69906 \\
15.87021 & 5.50546 & 3.74772 & 15.51819 & 6.49386 \\
15.56983 & 5.53738 & 3.74850 & 15.21137 & 6.27219 \\
17.27458 & 5.39004 & 3.59949 & 16.53929 & 7.55348 \\
16.90063 & 5.34072 & 3.72579 & 16.73501 & 7.21632 \\
15.47721 & 5.55417 & 1.64891 & 14.78660 & 5.77440 \\
13.05150 & 5.82273 & 4.76282 & 13.21447 & 3.86998 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
0.07514 \\
0.04267 \\
0.01942 \\
0.11272 \\
0.09196 \\
0.05442 \\
0.06554 \\
0.11604 \\
0.04650 \\
0.05845 \\
0.22548 \\
0.38447 \\
0.10157 \\
0.02260 \\
0.00690 \\
-0.00649 \\
0.06159 \\
0.08154 \\
100.00000 \\
-0.04231 \\
100.00000 \\
0.41155 \\
100.00000
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
11.26326 & 3.31298 & 0.73740 & 6.76095 & -2.22617 \\
15.69985 & 4.37431 & 2.10874 & 15.10563 & 0.09165 \\
20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\
20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\
8.58583 & 3.31380 & 1.72388 & 6.83312 & -1.25991 \\
13.31704 & 3.64595 & 1.70255 & 13.25446 & -0.43835 \\
11.11626 & 3.09276 & 0.73699 & 7.32439 & -2.45240 \\
19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\
19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\
16.87430 & 5.19566 & 3.63703 & 16.22908 & 6.81125 \\
15.18067 & 3.82623 & 2.00485 & 14.92965 & 0.21571 \\
17.75981 & 5.82128 & 3.21240 & 14.79256 & 6.69906 \\
19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\
16.46512 & 5.73235 & 4.15828 & 15.86370 & 8.30135 \\
16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\
15.96553 & 5.03263 & 3.85768 & 15.20931 & 6.27918 \\
17.76084 & 5.88369 & 3.25918 & 14.69040 & 6.94703 \\
19.35473 & 1.72010 & 0.34054 & 18.55516 & -1.99212 \\
15.87021 & 5.50546 & 3.74772 & 15.51819 & 6.49386 \\
16.90063 & 5.34072 & 3.72579 & 16.73501 & 7.21632 \\
18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\
17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\
12.15520 & 5.92196 & 3.80215 & 12.63356 & 3.16629 \\
16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.77211 \\
18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
0.01493 \\
0.07406 \\
-0.01025 \\
-0.00925 \\
0.05442 \\
0.02019 \\
0.01969 \\
0.01271 \\
0.11534 \\
0.02260 \\
0.08182 \\
-0.00649 \\
0.05746 \\
0.06079 \\
0.05034 \\
0.21179 \\
-0.01959 \\
0.03892 \\
0.06159 \\
-0.04231 \\
-0.04958 \\
0.00140 \\
0.11578 \\
0.04836 \\
100.00000
\end{pmatrix}}$$

Generation 20

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix} 15.69985 & 4.37431 & 2.10874 & 15.10563 & 0.09165 \\ 11.26326 & 3.31298 & 0.73740 & 6.76095 & -2.22617 \\ 20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\ 16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.77211 \\ 13.31704 & 3.64595 & 1.70255 & 13.25446 & -0.43835 \\ 11.11626 & 3.09276 & 0.73699 & 7.32439 & -2.45240 \\ 19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\ 19.23579 & 2.12973 & 0.70704 & 18.05768 & -1.94467 \\ 16.87430 & 5.19566 & 3.63703 & 16.22908 & 6.81125 \\ 16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\ 17.75981 & 5.82128 & 3.21240 & 14.79256 & 6.69906 \\ 17.76084 & 5.88369 & 3.25918 & 14.69040 & 6.94703 \\ 16.46512 & 5.73235 & 4.15828 & 15.86370 & 8.30135 \\ 16.77286 & 5.36024 & 3.79488 & 16.09562 & 7.22366 \\ 17.76084 & 5.88369 & 3.25918 & 14.69040 & 6.94703 \\ 17.75981 & 5.82128 & 3.21240 & 14.79256 & 6.69906 \\ 15.87021 & 5.50546 & 3.74772 & 15.51819 & 6.49386 \\ 16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.77211 \\ 18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\ 11.26326 & 3.31298 & 0.73740 & 6.76095 & -2.22617 \\ 12.15520 & 5.92196 & 3.80215 & 12.63356 & 3.16629 \\ 19.35473 & 1.72010 & 0.34054 & 18.55516 & -1.99212 \\ 18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474 \end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix} 16.79069 & 4.63526 & 2.44592 & 17.15737 & 0.66155 \\ 16.63274 & 4.59747 & 2.39710 & 16.86028 & 0.57903 \\ 15.35812 & 5.56955 & 3.48040 & 15.56005 & 3.77533 \\ 15.97187 & 5.01270 & 3.02022 & 16.02058 & 2.98898 \\ 12.19662 & 3.36432 & 1.21098 & 10.23546 & -1.46370 \\ 11.42132 & 3.16944 & 0.87083 & 8.14639 & -2.17322 \\ 19.30699 & 2.05965 & 0.64366 & 18.12875 & -1.96257 \\ 19.49050 & 1.87903 & 0.48030 & 18.31192 & -2.00873 \\ 16.83973 & 5.25176 & 3.69083 & 16.18359 & 6.95181 \\ 16.85487 & 5.22719 & 3.66727 & 16.20351 & 6.89025 \\ 17.76054 & 5.86570 & 3.24570 & 14.71985 & 6.87555 \\ 17.75960 & 5.80830 & 3.20267 & 14.81380 & 6.64750 \\ 16.39843 & 5.81298 & 4.23703 & 15.81344 & 8.53486 \\ 16.72945 & 5.41273 & 3.84615 & 16.06291 & 7.37569 \\ 17.76074 & 5.87771 & 3.25470 & 14.70019 & 6.92327 \\ 17.76052 & 5.86414 & 3.24453 & 14.72241 & 6.86934 \\ 15.87890 & 5.48473 & 3.72032 & 15.53838 & 6.37450 \\ 15.91421 & 5.40050 & 3.60897 & 15.62040 & 5.88948 \\ 17.54158 & 5.28467 & 3.49978 & 16.03681 & 7.31695 \\ 17.09317 & 5.14385 & 3.30248 & 15.37430 & 6.63536 \\ 16.73743 & 3.24763 & 1.59896 & 16.40244 & -0.11684 \\ 19.08057 & 1.88011 & 0.47236 & 18.32966 & -1.79568 \\ 18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474 \end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix} 16.79069 & 4.63526 & 2.44592 & 17.15737 & 0.66155 \\ 16.63274 & 4.59747 & 2.39710 & 16.86028 & 0.57903 \\ 15.35812 & 5.56955 & 3.48040 & 15.56396 & 3.77533 \\ 15.97773 & 5.01270 & 3.02022 & 16.01472 & 2.98898 \\ 12.19662 & 3.36432 & 1.21098 & 10.23546 & -1.46370 \\ 11.42132 & 3.16944 & 0.87071 & 8.14639 & -2.17322 \\ 19.30699 & 2.05965 & 0.64366 & 18.12875 & -1.96257 \\ 19.49050 & 1.87903 & 0.23030 & 17.81192 & -2.00873 \\ 16.83973 & 5.25176 & 3.72208 & 16.18359 & 6.95181 \\ 17.41932 & 5.22719 & 3.66727 & 16.76796 & 6.89025 \\ 17.76054 & 5.86570 & 3.24570 & 14.71985 & 6.87555 \\ 17.75960 & 5.80830 & 3.20267 & 14.81380 & 6.64750 \\ 16.39843 & 5.81298 & 4.23703 & 15.81344 & 8.53486 \\ 16.72945 & 5.41273 & 3.84615 & 16.06291 & 7.38350 \\ 17.76074 & 5.87771 & 3.25470 & 14.70019 & 6.92327 \\ 17.76052 & 5.86414 & 3.24453 & 14.72241 & 6.86934 \\ 15.87890 & 5.48473 & 3.72032 & 15.53838 & 6.37450 \\ 15.91421 & 5.40050 & 3.60897 & 15.62040 & 5.88948 \\ 17.54158 & 5.28467 & 3.49978 & 16.03681 & 7.31695 \\ 17.10879 & 5.15166 & 3.30248 & 15.37430 & 6.63536 \\ 16.73743 & 3.24763 & 1.59896 & 16.40244 & -0.11684 \\ 19.06494 & 1.88011 & 0.47236 & 18.32966 & -1.78006 \\ 18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474 \end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix} 0.06329 \\ 0.07610 \\ 0.10287 \\ 0.08064 \\ 0.04183 \\ 0.02644 \\ 0.10885 \\ -0.01036 \\ 0.04589 \\ 0.01592 \\ -0.00549 \\ -0.03174 \\ 0.06121 \\ 0.05499 \\ -0.00663 \\ -0.00556 \\ 0.06239 \\ 0.06443 \\ 0.00404 \\ -0.00002 \\ 0.05611 \\ 0.03960 \\ 100.00000 \end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix} 17.10879 & 5.15166 & 3.30248 & 15.37430 & 6.63536 \\ 16.79069 & 4.63526 & 2.44592 & 17.15737 & 0.66155 \\ 20.00000 & 1.36724 & 0.00000 & 19.03635 & -2.17130 \\ 20.00000 & 1.35806 & 0.00000 & 19.04307 & -2.17198 \\ 8.58583 & 3.31380 & 1.72388 & 6.83312 & -1.25991 \\ 13.31704 & 3.64595 & 1.70255 & 13.25446 & -0.43835 \\ 11.11626 & 3.09276 & 0.73699 & 7.32439 & -2.45240 \\ 19.87814 & 1.49748 & 0.13522 & 18.69885 & -2.10623 \\ 19.49050 & 1.87903 & 0.23030 & 17.81192 & -2.00873 \\ 16.87430 & 5.19566 & 3.63703 & 16.22908 & 6.81125 \\ 15.18067 & 3.82623 & 2.00485 & 14.92965 & 0.21571 \\ 17.75981 & 5.82128 & 3.21240 & 14.79256 & 6.69906 \\ 19.94096 & 1.35920 & 0.08064 & 19.02213 & -2.50070 \\ 16.46512 & 5.73235 & 4.15828 & 15.86370 & 8.30135 \\ 17.41932 & 5.22719 & 3.66727 & 16.76796 & 6.89025 \\ 15.96553 & 5.03263 & 3.85768 & 15.20931 & 6.27918 \\ 17.75960 & 5.80830 & 3.20267 & 14.81380 & 6.64750 \\ 19.35473 & 1.72010 & 0.34054 & 18.55516 & -1.99212 \\ 15.87021 & 5.50546 & 3.74772 & 15.51819 & 6.49386 \\ 16.90063 & 5.34072 & 3.72579 & 16.73501 & 7.21632 \\ 18.31340 & 5.52706 & 3.83937 & 17.17712 & 8.49012 \\ 17.63536 & 5.31523 & 3.54935 & 16.18533 & 7.46877 \\ 16.73743 & 3.24763 & 1.59896 & 16.40244 & -0.11684 \\ 16.14114 & 4.85913 & 2.89331 & 16.14759 & 2.77211 \\ 18.03096 & 4.14518 & 0.00000 & 19.99500 & 9.98474 \end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix} -0.00002 \\ 0.06329 \\ -0.01025 \\ -0.00925 \\ 0.05442 \\ 0.02019 \\ 0.01969 \\ 0.01271 \\ -0.01036 \\ 0.02260 \\ 0.08182 \\ -0.00649 \\ 0.05746 \\ 0.06079 \\ 0.01592 \\ 0.21179 \\ -0.03174 \\ 0.03892 \\ 0.06159 \\ -0.04231 \\ -0.04958 \\ 0.00140 \\ 0.05611 \\ 0.04836 \\ 100.00000 \end{pmatrix}}$$

B.2 Generation 70 bis 80

Generation 70

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.63518 & 2.96854 & 1.07296 & 16.95056 & 0.00594 \\
18.78309 & 3.09599 & 1.55786 & 17.73260 & 1.90492 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.67716 & 3.13225 & 1.18436 & 16.88291 & 0.33957 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
18.63555 & 2.96836 & 1.07281 & 16.95092 & 0.00595 \\
17.80364 & 3.40086 & 1.47909 & 16.45600 & 1.29882 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
16.87779 & 3.59353 & 1.63164 & 15.32110 & 1.81300 \\
16.93576 & 3.61010 & 1.62927 & 15.35434 & 1.80883 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.78309 & 3.09599 & 1.55786 & 17.73260 & 1.90492
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.35237 & 3.56983 & 1.72345 & 16.82382 & 2.14737 \\
18.39882 & 3.91614 & 1.98322 & 16.66303 & 2.98160 \\
18.66897 & 2.99766 & 1.18373 & 17.12921 & 0.43974 \\
18.66310 & 2.99260 & 1.16449 & 17.09817 & 0.36438 \\
18.76533 & 3.20772 & 1.64303 & 17.52042 & 2.25887 \\
18.71074 & 3.22602 & 1.64824 & 17.45560 & 2.25341 \\
18.12257 & 3.18879 & 1.26567 & 16.49697 & 0.49636 \\
18.37454 & 3.16310 & 1.22873 & 16.67232 & 0.42512 \\
17.74500 & 3.38799 & 1.49847 & 16.46115 & 1.39338 \\
17.85812 & 3.39471 & 1.48565 & 16.49250 & 1.33744 \\
18.50220 & 3.68690 & 1.57624 & 16.46018 & 1.47268 \\
18.51462 & 3.64752 & 1.54842 & 16.49019 & 1.39225 \\
18.62690 & 2.97221 & 1.07609 & 16.94246 & 0.01639 \\
18.63382 & 2.96913 & 1.07347 & 16.94922 & 0.00804 \\
17.89679 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.89161 & 3.39607 & 1.48271 & 16.50320 & 1.32525 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.90754 & 3.60203 & 1.63043 & 15.33816 & 1.81086 \\
18.98085 & 3.12441 & 1.61753 & 17.96701 & 2.08643 \\
18.26183 & 3.26875 & 1.60047 & 17.05767 & 1.93021 \\
17.31145 & 5.12590 & 2.52021 & 14.46671 & 4.44254 \\
17.79983 & 4.40080 & 2.14992 & 15.56944 & 3.37691 \\
18.78309 & 3.09599 & 1.55786 & 17.73260 & 1.90492
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
18.35139 & 3.56983 & 1.72394 & 16.82382 & 2.14737 \\
18.39882 & 3.91614 & 1.98322 & 16.66303 & 2.98160 \\
18.66897 & 2.99766 & 1.18373 & 17.12921 & 0.43974 \\
18.66310 & 2.99260 & 1.16449 & 17.09817 & 0.36438 \\
18.76533 & 3.20772 & 1.64303 & 17.52042 & 2.25887 \\
18.71074 & 3.22602 & 1.64824 & 17.45560 & 2.26904 \\
18.12257 & 3.18879 & 1.26567 & 14.46474 & 0.49636 \\
18.37454 & 3.16310 & 1.22873 & 16.67232 & 0.42512 \\
17.74500 & 3.38799 & 1.62371 & 16.46115 & 1.39338 \\
17.85812 & 3.39471 & 1.48565 & 16.49250 & 1.33744 \\
18.50220 & 3.68690 & 1.57624 & 16.46018 & 1.47268 \\
18.51462 & 3.64752 & 1.54842 & 16.49019 & 1.39225 \\
18.61908 & 2.96830 & 1.07609 & 16.94246 & 0.01639 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.95606 & 3.39607 & 1.48271 & 16.50320 & 1.32525 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.90754 & 3.60203 & 1.63043 & 15.33816 & 1.81086 \\
18.98085 & 3.12441 & 1.61753 & 17.96701 & 2.08643 \\
18.26183 & 3.26778 & 1.60047 & 17.05767 & 1.93021 \\
17.31145 & 5.12590 & 2.52070 & 14.46574 & 4.44254 \\
17.79983 & 4.40178 & 2.14992 & 15.56944 & 3.37496 \\
18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
-0.06030 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.06691 \\
100.00000 \\
0.01850 \\
100.00000 \\
-0.02603 \\
-0.07198 \\
100.00000 \\
100.00000 \\
-0.06241 \\
-0.06782 \\
-0.07608 \\
-0.07834 \\
-0.06472 \\
-0.06436 \\
100.00000 \\
100.00000 \\
-0.04643 \\
-0.06760 \\
0.12581
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63518 & 2.96854 & 1.07296 & 16.95056 & 0.00594 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67716 & 3.13225 & 1.18436 & 16.88291 & 0.33957 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.93576 & 3.61010 & 1.62927 & 15.35434 & 1.80883 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.97735 & 3.64347 & 1.68678 & 15.35071 & 1.91449 \\
18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06270 \\
-0.06776 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.06952 \\
-0.06782 \\
-0.06472 \\
-0.07970 \\
-0.07368 \\
-0.07331 \\
-0.06425 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06664 \\
-0.07391 \\
-0.06741 \\
-0.06190 \\
-0.05275 \\
0.12581
\end{pmatrix}}$$

Generation 71

$$\xrightarrow{\text{Selektion}} \underbrace{\begin{pmatrix} 18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\ 17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\ 18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\ 16.97735 & 3.64347 & 1.68678 & 15.35071 & 1.91449 \\ 18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\ 18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\ 18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\ 18.67716 & 3.13225 & 1.18436 & 16.88291 & 0.33957 \\ 18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\ 18.67716 & 3.13225 & 1.18436 & 16.88291 & 0.33957 \\ 17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\ 18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\ 18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\ 18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\ 17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\ 16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\ 16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\ 17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\ 16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\ 16.93576 & 3.61010 & 1.62927 & 15.35434 & 1.80883 \\ 17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\ 17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\ 18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492 \end{pmatrix}}_{\text{Nachkommen}_{\text{sel}}} \xrightarrow{\text{Rekombination}} \underbrace{\begin{pmatrix} 18.21444 & 3.90906 & 1.94427 & 16.47736 & 2.83344 \\ 18.26336 & 3.97132 & 1.99841 & 16.49025 & 3.01240 \\ 16.77558 & 3.71953 & 1.75794 & 15.15294 & 2.15124 \\ 18.24752 & 3.16464 & 1.23880 & 16.59571 & 0.42409 \\ 18.70494 & 3.37338 & 1.47513 & 16.96518 & 1.35854 \\ 18.70530 & 3.26120 & 1.60892 & 17.33822 & 2.04771 \\ 18.67634 & 3.13478 & 1.18616 & 16.88097 & 0.34475 \\ 18.67720 & 3.13212 & 1.18428 & 16.88300 & 0.33932 \\ 18.62915 & 2.95105 & 1.06111 & 16.95657 & -0.02871 \\ 18.67733 & 3.13288 & 1.18479 & 16.88265 & 0.34085 \\ 18.81284 & 3.45139 & 1.37751 & 16.75711 & 0.86528 \\ 18.47984 & 3.43162 & 1.41523 & 16.66481 & 1.02997 \\ 18.52495 & 3.62564 & 1.53255 & 16.50718 & 1.35025 \\ 18.51939 & 3.60646 & 1.52039 & 16.52074 & 1.30186 \\ 17.54228 & 3.43564 & 1.50435 & 16.19521 & 1.40205 \\ 16.81917 & 3.56175 & 1.57214 & 15.21186 & 1.64013 \\ 17.04249 & 3.81223 & 1.76648 & 15.30444 & 2.21921 \\ 17.81243 & 4.87617 & 2.43107 & 15.19963 & 4.21940 \\ 16.94359 & 3.61578 & 1.63527 & 15.36172 & 1.82663 \\ 16.83952 & 3.54031 & 1.55554 & 15.26376 & 1.59020 \\ 17.68278 & 4.60579 & 2.26961 & 15.29608 & 3.73885 \\ 17.50379 & 4.84593 & 2.37993 & 14.89937 & 4.04197 \\ 18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492 \end{pmatrix}}_{\text{Nachkommen}_{\text{rek}}}$$

$$\xrightarrow{\text{Mutation}} \underbrace{\begin{pmatrix} 18.21444 & 3.90906 & 1.94427 & 16.47736 & 2.83344 \\ 18.26336 & 3.97132 & 1.99834 & 16.49025 & 3.01240 \\ 16.77558 & 3.71953 & 1.75794 & 14.90099 & 2.40320 \\ 17.24752 & 3.16464 & 1.23880 & 16.59571 & 0.42409 \\ 18.70494 & 3.37338 & 1.47123 & 16.96518 & 1.35854 \\ 18.70530 & 3.26120 & 1.60892 & 17.33822 & 2.04771 \\ 18.67634 & 3.13478 & 1.18616 & 16.88097 & 0.34475 \\ 18.67720 & 3.13212 & 1.18428 & 16.88300 & 0.33932 \\ 18.62915 & 2.95105 & 1.06111 & 16.95657 & -0.02871 \\ 18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\ 18.81284 & 3.45139 & 1.37751 & 16.75711 & 0.86528 \\ 18.47984 & 3.43162 & 1.41523 & 16.66481 & 1.02997 \\ 19.08745 & 3.62564 & 1.81380 & 16.50718 & 1.91275 \\ 18.51939 & 3.60646 & 1.52039 & 16.45434 & 1.30186 \\ 17.47978 & 3.46689 & 1.50435 & 16.19521 & 1.40205 \\ 16.81917 & 5.71800 & 1.57214 & 15.21186 & 1.64013 \\ 17.04249 & 3.81223 & 1.76648 & 15.30444 & 2.21921 \\ 17.81243 & 4.87617 & 2.43107 & 15.19963 & 4.21940 \\ 16.94359 & 3.61578 & 1.63527 & 15.36184 & 1.82663 \\ 16.79265 & 3.54031 & 1.55554 & 15.31064 & 1.59020 \\ 17.68278 & 4.85579 & 2.26961 & 15.29608 & 3.73885 \\ 17.50379 & 4.84593 & 2.37993 & 15.47163 & 4.04197 \\ 18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492 \end{pmatrix}}_{\text{Nachkommen}_{\text{mut}}} \xrightarrow{\text{Bewertung}} \underbrace{\begin{pmatrix} -0.06153 \\ -0.06161 \\ 100.00000 \\ 100.00000 \\ -0.06327 \\ 100.00000 \\ -0.05932 \\ -0.06577 \\ -0.05977 \\ -0.07553 \\ -0.06303 \\ -0.06408 \\ 0.02231 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ -0.03392 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 100.00000 \\ 0.12581 \end{pmatrix}}_{\vec{z}_{\text{mut}}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underbrace{\begin{pmatrix} 18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\ 18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\ 18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\ 18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\ 18.63518 & 2.96854 & 1.07296 & 16.95056 & 0.00594 \\ 18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\ 18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\ 18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\ 18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\ 18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\ 18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\ 17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\ 18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\ 18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\ 18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\ 16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\ 17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\ 17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\ 16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\ 16.93576 & 3.61010 & 1.62927 & 15.35434 & 1.80883 \\ 16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\ 17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\ 17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\ 16.97735 & 3.64347 & 1.68678 & 15.35071 & 1.91449 \\ 18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492 \end{pmatrix}}_{\text{Population}} \underbrace{\begin{pmatrix} -0.07928 \\ -0.07593 \\ -0.07213 \\ -0.06270 \\ -0.06776 \\ -0.07437 \\ -0.07378 \\ -0.07434 \\ -0.07553 \\ -0.06782 \\ -0.06472 \\ -0.07970 \\ -0.07368 \\ -0.07331 \\ -0.06425 \\ -0.07720 \\ -0.07608 \\ -0.06485 \\ -0.06472 \\ -0.06664 \\ -0.07391 \\ -0.06741 \\ -0.06190 \\ -0.05275 \\ 0.12581 \end{pmatrix}}_{\vec{z}_{\text{Pop}}}$$

Generation 72

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.63518 & 2.96854 & 1.07296 & 16.95056 & 0.00594 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.93576 & 3.61010 & 1.62927 & 15.35434 & 1.80883 \\
18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492 \\
17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
16.97735 & 3.64347 & 1.68678 & 15.35071 & 1.91449
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
17.47079 & 3.42755 & 1.58171 & 16.05725 & 1.75835 \\
18.75751 & 3.16925 & 1.61223 & 17.68456 & 2.03791 \\
18.41315 & 4.02305 & 2.06341 & 16.61339 & 3.23913 \\
18.36412 & 3.65747 & 1.78919 & 16.78313 & 2.35849 \\
18.30238 & 3.13275 & 1.21292 & 16.64384 & 0.36796 \\
18.12147 & 3.22201 & 1.28900 & 16.47711 & 0.56476 \\
17.76760 & 3.42321 & 1.46411 & 16.36243 & 1.20885 \\
17.71609 & 3.43488 & 1.45613 & 16.30253 & 1.15899 \\
18.65657 & 3.12128 & 1.17456 & 16.87821 & 0.29924 \\
18.71882 & 3.53991 & 1.45287 & 16.68282 & 1.10046 \\
17.87345 & 3.39630 & 1.48253 & 16.50088 & 1.32395 \\
17.87467 & 3.39631 & 1.48253 & 16.50082 & 1.32391 \\
18.65122 & 2.84255 & 0.98489 & 17.02808 & -0.24322 \\
18.62831 & 2.96284 & 1.06952 & 16.94839 & -0.00225 \\
17.13528 & 3.50662 & 1.54251 & 15.64173 & 1.53605 \\
16.92830 & 3.54272 & 1.56191 & 15.36027 & 1.60420 \\
17.63487 & 4.67007 & 2.29914 & 15.18988 & 3.81999 \\
17.59858 & 4.71875 & 2.32150 & 15.10946 & 3.88144 \\
17.57240 & 3.43292 & 1.77718 & 16.17395 & 1.84195 \\
18.62958 & 3.13871 & 2.02280 & 17.53498 & 1.89694 \\
17.74494 & 3.31165 & 1.42474 & 16.35365 & 1.11607 \\
18.32051 & 4.04404 & 2.06165 & 16.50531 & 3.22145 \\
16.97735 & 3.64347 & 1.68678 & 15.35071 & 1.91449
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
17.47079 & 3.42755 & 1.58171 & 16.05725 & 1.75835 \\
18.75751 & 3.16925 & 1.61223 & 17.68456 & 2.03791 \\
18.41315 & 4.02305 & 2.06341 & 16.61339 & 3.23913 \\
18.36412 & 4.15943 & 1.78919 & 16.78313 & 3.36240 \\
18.17689 & 3.13275 & 1.21292 & 16.51835 & 0.36796 \\
18.12147 & 3.22201 & 1.28900 & 16.47711 & 0.56476 \\
17.76760 & 3.42321 & 1.46411 & 16.36243 & 1.20885 \\
17.71609 & 3.43488 & 1.45613 & 16.30253 & 1.15899 \\
19.15657 & 2.87128 & 1.17456 & 16.37821 & 0.29924 \\
18.71882 & 3.53991 & 1.45287 & 16.68282 & 1.10046 \\
17.87345 & 3.39630 & 1.48253 & 12.50088 & 1.32395 \\
17.87467 & 3.39631 & 1.48253 & 16.50082 & 0.32196 \\
18.65122 & 2.84255 & 0.98489 & 17.02808 & -0.24322 \\
18.63026 & 2.96284 & 1.06952 & 16.94839 & -0.00225 \\
13.11966 & 1.49881 & 1.54251 & 11.62611 & 1.53605 \\
16.92830 & 3.54272 & 1.56191 & 15.36027 & 1.60420 \\
17.63487 & 4.67007 & 2.29914 & 15.18988 & 3.81999 \\
17.59858 & 4.71875 & 2.32150 & 15.10946 & 3.88144 \\
17.57215 & 3.43292 & 1.77718 & 16.17395 & 1.84170 \\
18.62958 & 3.13871 & 4.33530 & 17.53498 & 1.89694 \\
17.74494 & 3.31165 & 1.42474 & 16.35365 & 1.11607 \\
18.32051 & 4.04404 & 2.06165 & 16.50531 & 3.22145 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 1.91449
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.05802 \\
-0.05733 \\
100.00000 \\
100.00000 \\
-0.00232 \\
-0.06410 \\
0.10808 \\
-0.00681 \\
-0.06166 \\
-0.06075 \\
0.32016 \\
-0.06949 \\
100.00000 \\
-0.04686 \\
-0.01056 \\
1.28613 \\
-0.06569 \\
-0.06371 \\
100.00000
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63518 & 2.96854 & 1.07296 & 16.95056 & 0.00594 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.93576 & 3.61010 & 1.62927 & 15.35434 & 1.80883 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 1.91449 \\
18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06270 \\
-0.06776 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.06472 \\
-0.07970 \\
-0.07368 \\
-0.07331 \\
-0.06425 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06664 \\
-0.07391 \\
-0.06741 \\
-0.06190 \\
100.00000 \\
0.12581
\end{pmatrix}}$$

Generation 73

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.78309 & 3.09599 & 2.05847 & 17.73260 & 1.90492 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.63518 & 2.96854 & 1.07296 & 16.95056 & 0.00594 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.63518 & 2.96854 & 1.07296 & 16.95056 & 0.00594 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
16.93576 & 3.61010 & 1.62927 & 15.35434 & 1.80883 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 1.91449
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.72843 & 3.16053 & 1.70676 & 17.65113 & 2.00199 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316 \\
18.36577 & 3.66972 & 1.79837 & 16.77745 & 2.38798 \\
18.42783 & 4.13249 & 2.14550 & 16.56258 & 3.50274 \\
18.67423 & 3.11269 & 1.39310 & 17.22784 & 1.25523 \\
18.63016 & 2.95000 & 1.03177 & 16.91488 & -0.15478 \\
18.69210 & 3.08480 & 1.15084 & 16.91906 & 0.24263 \\
18.55084 & 3.53267 & 1.46728 & 16.57772 & 1.15761 \\
18.63509 & 2.96859 & 1.07299 & 16.95046 & 0.00610 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
17.87104 & 3.39629 & 1.48254 & 16.50100 & 1.32401 \\
17.87293 & 3.39630 & 1.48253 & 16.50091 & 1.32396 \\
18.64830 & 2.85789 & 0.99568 & 17.01792 & -0.21249 \\
18.48584 & 3.71075 & 1.59575 & 16.45291 & 1.49600 \\
16.87138 & 3.54897 & 1.56578 & 15.29809 & 1.61955 \\
16.91924 & 3.54247 & 1.56203 & 15.35553 & 1.60550 \\
17.78075 & 3.29317 & 1.41691 & 16.61100 & 1.12492 \\
17.63248 & 4.82429 & 2.40191 & 15.04423 & 4.13702 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.86857 & 3.56137 & 1.57779 & 15.29110 & 1.65619 \\
18.09411 & 3.42967 & 1.54306 & 16.64561 & 1.49820 \\
17.94426 & 3.44436 & 1.54597 & 16.48026 & 1.51346 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 1.91449
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
20.00000 & 3.16053 & 1.70676 & 17.65113 & 2.00199 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316 \\
18.36577 & 3.66972 & 1.79837 & 16.77745 & 2.38798 \\
18.42393 & 4.13249 & 2.14745 & 16.55867 & 3.50274 \\
19.17423 & 3.11269 & 1.39310 & 17.22784 & 1.25523 \\
18.63016 & 2.95000 & 1.03177 & 16.91488 & -0.15478 \\
18.69210 & 3.08480 & 1.15084 & 16.91906 & 0.24263 \\
18.55084 & 3.53267 & 2.46923 & 16.57772 & 1.15761 \\
18.63509 & 2.96859 & 1.07299 & 16.95046 & 0.00610 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
17.87104 & 3.39629 & 1.35748 & 16.50100 & 1.32401 \\
18.00183 & 3.39630 & 1.48253 & 16.50091 & 1.32396 \\
18.64830 & 2.85789 & 0.99568 & 17.01792 & -0.21249 \\
18.48584 & 3.71075 & 1.59575 & 16.45291 & 1.49600 \\
16.87138 & 3.54897 & 1.56578 & 15.29809 & 1.61955 \\
17.91924 & 3.54247 & 1.56203 & 14.35553 & 1.60550 \\
17.78075 & 3.29317 & 1.41691 & 16.61100 & 1.12492 \\
17.63248 & 4.79304 & 2.40191 & 15.04423 & 4.19952 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.83610 & 3.56137 & 1.57779 & 15.32357 & 1.65619 \\
18.09411 & 3.42967 & 0.00000 & 16.64561 & 1.49820 \\
17.94426 & 3.43655 & 1.54597 & 16.48026 & 1.51346 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 0.91449
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \begin{pmatrix}
\vec{z}_{\text{mut}} \\
0.02123 \\
0.02255 \\
100.00000 \\
-0.06048 \\
-0.04944 \\
100.00000 \\
-0.06154 \\
0.37325 \\
-0.06771 \\
-0.06780 \\
100.00000 \\
-0.06775 \\
-0.05435 \\
100.00000 \\
-0.06924 \\
0.00973 \\
100.00000 \\
-0.05192 \\
-0.06978 \\
100.00000 \\
100.00000 \\
-0.06354 \\
100.00000
\end{pmatrix}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 0.91449 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \begin{pmatrix}
\vec{z}_{\text{Pop}} \\
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06270 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.06472 \\
-0.07970 \\
-0.07368 \\
-0.07331 \\
-0.06425 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06978 \\
-0.07391 \\
-0.06741 \\
-0.06190 \\
100.00000 \\
0.02255
\end{pmatrix}$$

Generation 74

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
17.85604 & 3.45302 & 1.54768 & 16.38293 & 1.52245 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 0.91449
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.41884 & 4.25159 & 2.22388 & 16.46967 & 3.73989 \\
18.58195 & 3.65749 & 1.88481 & 17.11125 & 2.79270 \\
18.64852 & 3.15045 & 1.18899 & 16.86263 & 0.37825 \\
18.09202 & 3.18797 & 1.27004 & 16.47580 & 0.49711 \\
18.67525 & 3.11664 & 1.40185 & 17.23536 & 1.28938 \\
18.67185 & 3.10412 & 1.37403 & 17.21126 & 1.18081 \\
18.67690 & 3.14448 & 1.18548 & 16.88184 & 0.36410 \\
18.67698 & 3.14529 & 1.18534 & 16.88200 & 0.36571 \\
18.65024 & 3.03653 & 1.11526 & 16.92421 & 0.14484 \\
18.67835 & 3.15273 & 1.18742 & 16.88109 & 0.38068 \\
18.85724 & 3.17640 & 1.63302 & 17.62968 & 2.26384 \\
18.75617 & 3.19929 & 1.61735 & 17.51214 & 2.16598 \\
18.51393 & 3.58763 & 1.50845 & 16.53405 & 1.25437 \\
18.52388 & 3.62194 & 1.53020 & 16.50980 & 1.34091 \\
18.83529 & 3.43813 & 1.37709 & 16.82591 & 0.86820 \\
16.60237 & 3.55509 & 1.59364 & 15.13697 & 1.73260 \\
17.73478 & 3.76789 & 1.72230 & 16.12522 & 2.05882 \\
17.65862 & 4.55431 & 2.22823 & 15.32049 & 3.60591 \\
16.88810 & 3.59850 & 1.63495 & 15.32872 & 1.82308 \\
16.85968 & 3.56414 & 1.58791 & 15.29045 & 1.68516 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
18.17541 & 3.85940 & 1.90108 & 16.46708 & 2.69068 \\
16.97735 & 3.89444 & 1.43580 & 15.35071 & 0.91449
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
19.54482 & 4.25159 & 2.22388 & 16.46967 & 3.73989 \\
18.58195 & 3.65553 & 1.88481 & 17.11125 & 2.79661 \\
18.64852 & 3.15045 & 1.18899 & 16.86263 & 0.37825 \\
18.09202 & 3.11961 & 1.27004 & 16.47580 & 0.49711 \\
18.67525 & 3.11664 & 1.40185 & 17.23536 & 1.28938 \\
18.67185 & 3.10412 & 0.37305 & 17.21126 & 3.18276 \\
18.67690 & 3.14448 & 1.18548 & 16.88184 & 0.36410 \\
18.67698 & 3.14529 & 1.18486 & 16.88200 & 0.36571 \\
18.65024 & 3.03653 & 1.11923 & 16.92421 & 0.14484 \\
18.67835 & 3.15273 & 1.18742 & 16.88109 & 0.38068 \\
18.85724 & 3.17640 & 1.63302 & 17.62968 & 2.26384 \\
18.75617 & 3.19929 & 1.61735 & 17.51214 & 2.16598 \\
18.51393 & 3.58763 & 1.50845 & 16.00280 & 1.78562 \\
18.27388 & 3.74694 & 1.53020 & 16.50980 & 1.09091 \\
18.83529 & 3.43813 & 1.37709 & 16.82591 & 0.86820 \\
16.60237 & 3.55509 & 1.59364 & 15.13697 & 1.73260 \\
17.73478 & 4.76789 & 1.72230 & 16.12522 & 2.05882 \\
17.65862 & 4.55431 & 2.22823 & 15.32049 & 3.60591 \\
16.88810 & 3.66198 & 1.63495 & 15.32872 & 1.82308 \\
16.85968 & 3.56414 & 1.58791 & 15.29045 & 1.55918 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
18.11291 & 3.85940 & 1.90108 & 16.40458 & 2.69068 \\
16.94610 & 3.89444 & 1.42018 & 15.35071 & 0.91449
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \begin{pmatrix}
\vec{z}_{\text{mut}} \\
-0.01662 \\
100.00000 \\
100.00000 \\
-0.04648 \\
-0.06759 \\
100.00000 \\
-0.06431 \\
-0.06871 \\
-0.06126 \\
-0.06555 \\
-0.07014 \\
-0.06920 \\
100.00000 \\
-0.06781 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.05219 \\
100.00000 \\
100.00000 \\
-0.07182 \\
-0.06055 \\
100.00000
\end{pmatrix}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.94610 & 3.89444 & 1.42018 & 15.35071 & 0.91449 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \begin{pmatrix}
\vec{z}_{\text{Pop}} \\
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06270 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.06472 \\
-0.07970 \\
-0.07368 \\
-0.07331 \\
-0.06425 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06978 \\
-0.07391 \\
-0.07182 \\
-0.06190 \\
100.00000 \\
0.02255
\end{pmatrix}$$

Generation 75

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.70471 & 3.44497 & 1.38976 & 16.72713 & 0.91876 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.62649 & 2.97239 & 1.07624 & 16.94207 & 0.01688 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.94610 & 3.89444 & 1.42018 & 15.35071 & 0.91449 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.37544 & 4.40966 & 2.31410 & 16.29896 & 3.99191 \\
18.72884 & 3.12245 & 1.57944 & 17.68905 & 1.93968 \\
17.53306 & 4.79688 & 2.35219 & 14.96708 & 3.95900 \\
17.51246 & 4.82747 & 2.36781 & 14.92057 & 4.00395 \\
18.65526 & 3.04297 & 1.23820 & 17.09358 & 0.65077 \\
18.64056 & 2.98878 & 1.11784 & 16.98930 & 0.18109 \\
18.76780 & 3.20690 & 1.64279 & 17.52336 & 2.25911 \\
18.73331 & 3.21845 & 1.64609 & 17.48240 & 2.25567 \\
18.67663 & 3.14198 & 1.18591 & 16.88133 & 0.35912 \\
18.67580 & 3.13427 & 1.18724 & 16.87977 & 0.34379 \\
17.94525 & 3.39988 & 1.47579 & 16.51665 & 1.29435 \\
18.83306 & 3.45259 & 1.37522 & 16.76271 & 0.85528 \\
18.70110 & 3.05626 & 1.13068 & 16.94081 & 0.18433 \\
18.64592 & 3.23119 & 1.25428 & 16.80748 & 0.54171 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
18.63163 & 2.97075 & 1.07486 & 16.94690 & 0.01234 \\
17.81591 & 3.39601 & 1.48275 & 16.50375 & 1.32555 \\
17.75092 & 3.39569 & 1.48300 & 16.50698 & 1.32736 \\
16.89530 & 3.64243 & 1.59837 & 15.32965 & 1.67233 \\
16.87106 & 3.52220 & 1.68339 & 15.31961 & 2.03388 \\
17.17653 & 3.67262 & 1.68651 & 15.53944 & 2.01126 \\
18.53109 & 4.18331 & 2.18895 & 16.63966 & 3.63873 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
18.37544 & 4.40966 & 2.31410 & 16.29896 & 3.99191 \\
18.72884 & 3.12245 & 1.57944 & 17.68905 & 1.93968 \\
17.53306 & 4.79688 & 2.35219 & 14.96708 & 3.95900 \\
17.51246 & 4.82747 & 2.36781 & 14.92057 & 4.00395 \\
18.65526 & 3.04297 & 1.23820 & 17.09358 & 0.65077 \\
18.64056 & 2.98878 & 1.11759 & 16.98930 & 0.18060 \\
18.76780 & 3.20690 & 3.64279 & 17.52336 & 2.25911 \\
18.73331 & 3.21845 & 1.64609 & 17.48240 & 2.25567 \\
18.67663 & 3.14198 & 1.18591 & 16.88133 & 0.35912 \\
18.67580 & 3.13427 & 1.18724 & 16.87977 & 0.34354 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
18.83306 & 3.45259 & 1.37522 & 16.76271 & 0.85528 \\
18.70110 & 3.05626 & 1.13068 & 16.94081 & 0.18433 \\
18.64592 & 3.23119 & 1.25428 & 16.80748 & 0.54171 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
18.63163 & 2.97075 & 1.07486 & 16.94690 & 0.01234 \\
17.81591 & 3.39601 & 1.48275 & 16.50375 & 1.04405 \\
17.75092 & 3.39569 & 1.48300 & 16.50698 & 1.32736 \\
16.89530 & 3.64243 & 1.59837 & 15.32965 & 1.67233 \\
16.87106 & 3.52220 & 1.68339 & 15.31961 & 2.03388 \\
17.17653 & 3.67189 & 1.68651 & 15.53944 & 2.01126 \\
18.53109 & 4.18331 & 2.18895 & 16.63966 & 3.63873 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
100.00000 \\
-0.06200 \\
-0.04435 \\
-0.04134 \\
100.00000 \\
-0.06130 \\
0.90161 \\
-0.06230 \\
100.00000 \\
-0.07133 \\
-0.06736 \\
-0.05971 \\
-0.06141 \\
100.00000 \\
-0.06675 \\
-0.06783 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.05242 \\
100.00000 \\
100.00000 \\
-0.06190
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.87106 & 3.52220 & 1.68339 & 15.31961 & 2.03388 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06270 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.06736 \\
-0.07970 \\
-0.07368 \\
-0.07331 \\
-0.06675 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06978 \\
-0.07391 \\
-0.07182 \\
-0.06190 \\
-0.05242 \\
0.02255
\end{pmatrix}}$$

Generation 76

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
16.87106 & 3.52220 & 1.68339 & 15.31961 & 2.03388 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.52331 & 3.61998 & 1.52896 & 16.51119 & 1.33597 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.36844 & 4.43515 & 2.32865 & 16.27144 & 4.03255 \\
18.67359 & 3.32369 & 1.69430 & 17.47173 & 2.26052 \\
16.78013 & 3.54549 & 1.71267 & 15.23546 & 2.14020 \\
17.57633 & 3.34156 & 1.45624 & 15.97229 & 1.20914 \\
18.65706 & 3.23875 & 1.63775 & 17.38597 & 2.19190 \\
18.68579 & 3.23225 & 1.64429 & 17.42358 & 2.22813 \\
18.58170 & 3.43483 & 1.39815 & 16.65229 & 0.95772 \\
18.51171 & 3.65677 & 1.55496 & 16.48314 & 1.41114 \\
17.94590 & 3.39992 & 1.47571 & 16.45364 & 1.29403 \\
17.91560 & 3.39812 & 1.47914 & 16.47727 & 1.30901 \\
17.80227 & 3.36695 & 1.47637 & 16.49882 & 1.32197 \\
18.20130 & 3.50697 & 1.50547 & 16.50566 & 1.32972 \\
18.13072 & 3.18857 & 1.25002 & 16.44643 & 0.56824 \\
18.15684 & 3.21531 & 1.26735 & 16.45291 & 0.61368 \\
17.86602 & 3.39627 & 1.48256 & 16.50125 & 1.32415 \\
17.89967 & 3.39644 & 1.48243 & 16.49958 & 1.32322 \\
17.17977 & 4.00193 & 1.88497 & 15.28575 & 2.57584 \\
17.17471 & 3.99493 & 1.88061 & 15.28644 & 2.56269 \\
16.98888 & 3.53300 & 1.55657 & 15.43913 & 1.58504 \\
16.88623 & 3.54695 & 1.56462 & 15.31591 & 1.61519 \\
17.73907 & 4.49101 & 2.19599 & 15.43224 & 3.50949 \\
17.81187 & 4.38293 & 2.14079 & 15.59662 & 3.35064 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
17.36844 & 4.43515 & 2.32865 & 16.27144 & 4.03255 \\
18.67359 & 3.32369 & 1.69430 & 17.47173 & 2.26052 \\
18.78404 & 3.54549 & 1.71267 & 15.23546 & 2.14020 \\
13.57633 & 3.34156 & 1.45624 & 19.97229 & 1.20914 \\
18.65706 & 3.23875 & 1.63775 & 17.38597 & 2.19190 \\
18.68579 & 3.23225 & 1.64429 & 17.42358 & 2.48619 \\
18.58195 & 3.43470 & 1.39815 & 16.65229 & 0.95747 \\
18.51171 & 3.65677 & 1.55496 & 16.48314 & 1.47364 \\
17.94590 & 3.39992 & 1.35071 & 16.45364 & 1.29403 \\
17.91560 & 3.39812 & 1.47914 & 16.47727 & 1.30901 \\
17.80227 & 3.35913 & 1.47637 & 16.49882 & 1.32197 \\
18.20130 & 3.50697 & 1.50547 & 16.50566 & 1.32972 \\
18.13047 & 3.18845 & 1.24990 & 16.44643 & 0.56824 \\
18.15684 & 3.21531 & 1.26735 & 16.45291 & 1.12931 \\
17.86602 & 3.45883 & 1.48256 & 16.50125 & 1.32415 \\
17.89967 & 3.39644 & 1.50611 & 16.49958 & 1.32322 \\
17.17977 & 4.00193 & 1.88497 & 15.28575 & 2.57584 \\
16.79971 & 3.99493 & 1.88061 & 15.28644 & 2.56269 \\
16.98888 & 3.53300 & 1.55657 & 15.43913 & 1.58504 \\
16.88623 & 3.54695 & 1.56462 & 15.31591 & 1.61519 \\
17.73907 & 4.49101 & 2.19599 & 15.43224 & 3.50949 \\
17.81187 & 4.38293 & 2.14079 & 15.59662 & 3.35064 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
100.00000 \\
100.00000 \\
0.02724 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.06624 \\
-0.06525 \\
100.00000 \\
-0.06995 \\
-0.07402 \\
-0.06640 \\
-0.06615 \\
100.00000 \\
100.00000 \\
-0.06099 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.06944 \\
-0.05348 \\
-0.05702 \\
0.02255
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
17.80227 & 3.35913 & 1.47637 & 16.49882 & 1.32197 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.87106 & 3.52220 & 1.68339 & 15.31961 & 2.03388 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06270 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.06736 \\
-0.07970 \\
-0.07402 \\
-0.07331 \\
-0.06675 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06978 \\
-0.07391 \\
-0.07182 \\
-0.06190 \\
-0.05242 \\
0.02255
\end{pmatrix}}$$

Generation 77

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
17.80227 & 3.35913 & 1.47637 & 16.49882 & 1.32197 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.88611 & 3.39637 & 1.48248 & 16.50026 & 1.32359 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.87106 & 3.52220 & 1.68339 & 15.31961 & 2.03388 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.38147 & 4.12161 & 2.12911 & 16.52138 & 3.44446 \\
18.42398 & 4.17569 & 2.17614 & 16.53257 & 3.59993 \\
18.42516 & 3.07217 & 1.16128 & 16.75699 & 0.23443 \\
18.49307 & 3.03866 & 1.13272 & 16.81958 & 0.16057 \\
18.70608 & 3.22377 & 1.64571 & 17.46386 & 2.23466 \\
18.70388 & 3.23713 & 1.65580 & 17.41558 & 2.29497 \\
18.67723 & 3.14759 & 1.18495 & 16.88247 & 0.37029 \\
18.67767 & 3.15165 & 1.18425 & 16.88329 & 0.37836 \\
18.53363 & 3.00124 & 1.10053 & 16.85498 & 0.09697 \\
18.75380 & 2.93077 & 1.04100 & 17.06182 & -0.09822 \\
18.40553 & 3.27871 & 1.56300 & 17.10434 & 1.82649 \\
18.01502 & 3.36717 & 1.50247 & 16.65018 & 1.44840 \\
18.03495 & 3.43097 & 1.48489 & 16.51262 & 1.29178 \\
18.55366 & 3.59110 & 1.50388 & 16.54340 & 1.22449 \\
17.88915 & 3.39638 & 1.48247 & 16.50010 & 1.32351 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
16.87455 & 3.53951 & 1.67115 & 15.32105 & 1.98183 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993 \\
16.84593 & 3.54495 & 1.56045 & 15.26979 & 1.60476 \\
16.84638 & 3.54528 & 1.56080 & 15.27022 & 1.60579 \\
17.87723 & 4.55445 & 2.28403 & 15.57736 & 3.82110 \\
18.47718 & 4.05426 & 2.10952 & 16.68939 & 3.40357 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
18.38147 & 4.12161 & 2.12911 & 16.52138 & 3.44446 \\
18.42398 & 4.17569 & 2.17614 & 16.53257 & 3.84993 \\
18.42516 & 3.07217 & 1.16128 & 16.75699 & 0.23443 \\
18.49307 & 3.03866 & 3.75772 & 16.81958 & 0.16057 \\
18.70608 & 3.22425 & 1.64571 & 17.46386 & 2.23563 \\
18.70364 & 3.23713 & 1.65580 & 17.41558 & 2.29522 \\
18.67723 & 3.14759 & 1.18495 & 16.88247 & 0.37029 \\
18.67767 & 3.15165 & 1.18425 & 16.88329 & 0.37836 \\
18.53363 & 3.00124 & 1.10053 & 16.85498 & 0.09697 \\
18.75380 & 2.93077 & 1.04100 & 17.06182 & -0.09822 \\
18.40553 & 3.27871 & 1.56300 & 17.10141 & 1.82356 \\
18.01502 & 3.36717 & 1.50247 & 16.65018 & 1.44840 \\
18.03495 & 3.43097 & 1.48489 & 16.51262 & 1.29178 \\
18.55366 & 3.59110 & 3.56638 & 16.54340 & 1.22449 \\
17.88915 & 3.39638 & 1.48247 & 16.50010 & 1.32351 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
16.87455 & 3.53951 & 1.67005 & 15.32105 & 1.98183 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993 \\
16.84593 & 4.80277 & 1.56045 & 15.26979 & 1.60476 \\
16.84650 & 3.54528 & 1.56073 & 15.27022 & 1.60579 \\
17.87723 & 4.55445 & 2.28403 & 15.57736 & 3.82110 \\
18.47718 & 4.05426 & 2.10952 & 16.68939 & 3.40357 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
-0.06183 \\
100.00000 \\
100.00000 \\
1.22827 \\
-0.06152 \\
-0.06360 \\
-0.06547 \\
-0.06662 \\
-0.06331 \\
-0.06034 \\
-0.07587 \\
100.00000 \\
-0.07091 \\
0.93493 \\
-0.07149 \\
-0.08109 \\
-0.05375 \\
-0.05583 \\
100.00000 \\
100.00000 \\
-0.05203 \\
100.00000 \\
0.02255
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
17.80227 & 3.35913 & 1.47637 & 16.49882 & 1.32197 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06270 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.06736 \\
-0.08109 \\
-0.07402 \\
-0.07331 \\
-0.06675 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06978 \\
-0.07391 \\
-0.07182 \\
-0.06190 \\
-0.05583 \\
0.02255
\end{pmatrix}}$$

Generation 78

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.20604 & 3.18028 & 1.25343 & 16.55505 & 0.47276 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
17.94525 & 3.39988 & 1.47579 & 16.45415 & 1.29435 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
18.38738 & 4.12913 & 2.13564 & 16.52293 & 3.46606 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.34901 & 3.54476 & 1.70464 & 16.83547 & 2.08697 \\
18.40267 & 3.94490 & 2.00479 & 16.64968 & 3.05087 \\
18.68620 & 3.14791 & 1.18350 & 16.88882 & 0.37021 \\
18.23608 & 3.17825 & 1.24906 & 16.57593 & 0.46634 \\
18.01518 & 3.38405 & 1.49170 & 16.54569 & 1.38252 \\
18.55055 & 3.26286 & 1.61352 & 17.24655 & 2.05760 \\
18.67567 & 3.13306 & 1.18744 & 16.87953 & 0.34137 \\
18.67602 & 3.13630 & 1.18689 & 16.88018 & 0.34783 \\
18.63434 & 2.96894 & 1.07326 & 16.94966 & 0.00736 \\
18.63470 & 2.96877 & 1.07313 & 16.95004 & 0.00676 \\
17.89856 & 3.39864 & 1.47799 & 16.47001 & 1.30430 \\
17.84760 & 3.39728 & 1.48041 & 16.48733 & 1.31517 \\
17.21569 & 3.44895 & 1.48342 & 15.63986 & 1.34744 \\
16.85701 & 3.57440 & 1.59133 & 15.31237 & 1.69920 \\
18.01113 & 3.68162 & 1.73734 & 16.51259 & 2.16062 \\
17.63531 & 3.23462 & 1.33947 & 16.50226 & 0.85664 \\
17.72210 & 4.75135 & 2.35311 & 15.21192 & 3.98474 \\
17.76129 & 4.80550 & 2.38693 & 15.20659 & 4.08655 \\
17.52185 & 3.80281 & 1.81460 & 15.81992 & 2.42615 \\
17.27334 & 3.70911 & 1.72242 & 15.61807 & 2.12757 \\
17.96718 & 4.15234 & 2.02304 & 15.94730 & 3.01176 \\
18.52665 & 3.32169 & 1.59885 & 17.21055 & 1.79100 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
18.28651 & 3.54476 & 1.70464 & 16.77297 & 2.08697 \\
18.40267 & 3.94490 & 2.00479 & 16.64968 & 3.05087 \\
18.68620 & 3.14791 & 1.18350 & 16.88882 & 0.37021 \\
18.23608 & 3.17825 & 1.24906 & 16.57593 & 0.46634 \\
18.01518 & 3.38405 & 1.49170 & 16.54569 & 1.38252 \\
18.55055 & 3.26286 & 1.61352 & 17.24655 & 2.05760 \\
18.67567 & 3.13306 & 1.18744 & 16.87953 & 0.34137 \\
18.67602 & 3.13630 & 1.18689 & 16.88043 & 0.34783 \\
18.69733 & 2.96894 & 1.04177 & 16.94966 & 0.00736 \\
18.63470 & 2.96682 & 1.07313 & 16.95004 & 0.00676 \\
17.89856 & 3.39839 & 1.47799 & 16.47001 & 1.30430 \\
17.81244 & 3.39728 & 1.48041 & 16.48733 & 1.31517 \\
17.21569 & 3.44895 & 1.48342 & 15.63986 & 1.34744 \\
16.85701 & 3.57440 & 1.59133 & 15.31237 & 1.69920 \\
18.01113 & 3.68162 & 1.73734 & 16.51259 & 2.16062 \\
17.64507 & 3.23462 & 1.34436 & 16.50226 & 0.85664 \\
17.69085 & 4.75135 & 2.35311 & 15.18067 & 3.98474 \\
17.75348 & 4.80550 & 2.38693 & 15.21440 & 4.08655 \\
17.52185 & 3.80281 & 1.81460 & 15.81980 & 2.42615 \\
17.27334 & 3.70911 & 1.72242 & 15.61807 & 2.12757 \\
16.96718 & 4.65234 & 2.02304 & 16.94730 & 3.01176 \\
18.52665 & 2.81754 & 1.59885 & 17.21055 & 1.79100 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
-0.05780 \\
100.00000 \\
-0.06566 \\
-0.05742 \\
100.00000 \\
-0.06509 \\
-0.06993 \\
100.00000 \\
100.00000 \\
-0.06051 \\
-0.07045 \\
-0.07503 \\
100.00000 \\
100.00000 \\
-0.06920 \\
-0.07573 \\
100.00000 \\
100.00000 \\
-0.06285 \\
100.00000 \\
100.00000 \\
0.01154 \\
0.02255
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.68620 & 3.14791 & 1.18350 & 16.88882 & 0.37021 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
17.81244 & 3.39728 & 1.48041 & 16.48733 & 1.31517 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
17.80227 & 3.35913 & 1.47637 & 16.49882 & 1.32197 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.64507 & 3.23462 & 1.34436 & 16.50226 & 0.85664 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07593 \\
-0.07213 \\
-0.06566 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.07503 \\
-0.08109 \\
-0.07402 \\
-0.07331 \\
-0.06675 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.06978 \\
-0.07391 \\
-0.07573 \\
-0.06190 \\
-0.05583 \\
0.02255
\end{pmatrix}}$$

Generation 79

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.43561 & 4.19050 & 2.18902 & 16.53564 & 3.64250 \\
17.64507 & 3.23462 & 1.34436 & 16.50226 & 0.85664 \\
18.68620 & 3.14791 & 1.18350 & 16.88882 & 0.37021 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
16.84062 & 3.54110 & 1.55638 & 15.26479 & 1.59269 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
17.66769 & 3.26196 & 1.36852 & 16.50322 & 0.93634 \\
18.26605 & 3.98548 & 2.00785 & 16.52848 & 3.04498 \\
17.00282 & 3.52531 & 1.62816 & 15.43088 & 1.83105 \\
18.40075 & 3.21190 & 1.25890 & 16.64160 & 0.61792 \\
18.68823 & 3.16446 & 1.50805 & 17.32737 & 1.70382 \\
18.64087 & 2.98993 & 1.12039 & 16.99151 & 0.19105 \\
18.61074 & 3.22549 & 1.29684 & 16.86789 & 0.71561 \\
18.44278 & 3.45615 & 1.57978 & 16.83643 & 1.66457 \\
18.09945 & 3.14021 & 1.21792 & 16.44708 & 0.48188 \\
18.38535 & 3.04870 & 1.14062 & 16.71568 & 0.22842 \\
17.79826 & 3.39592 & 1.48282 & 16.50462 & 1.32604 \\
17.77410 & 3.39580 & 1.48291 & 16.50582 & 1.32671 \\
18.23472 & 3.58101 & 1.52426 & 16.33421 & 1.32906 \\
17.22711 & 3.59209 & 1.60423 & 15.58086 & 1.69014 \\
17.77180 & 3.39579 & 1.48292 & 16.50594 & 1.32678 \\
17.76703 & 3.39577 & 1.48294 & 16.50618 & 1.32691 \\
16.77068 & 3.43663 & 1.53185 & 15.34145 & 1.51308 \\
16.93565 & 3.66458 & 1.67425 & 15.31899 & 1.94164 \\
16.85062 & 3.54836 & 1.56404 & 15.27421 & 1.61542 \\
16.85458 & 3.55123 & 1.56708 & 15.27793 & 1.62441 \\
17.97974 & 4.13369 & 2.01351 & 15.97567 & 2.98435 \\
18.32035 & 3.62799 & 1.75527 & 16.74473 & 2.24115 \\
18.74466 & 3.14136 & 1.81121 & 17.67532 & 1.97316
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
17.66769 & 3.26196 & 1.36852 & 16.50322 & 0.93634 \\
18.26605 & 3.98548 & 2.00785 & 16.52848 & 3.04498 \\
16.94008 & 3.49394 & 1.62816 & 15.43088 & 1.83105 \\
18.40075 & 3.18065 & 1.25890 & 16.57910 & 0.61792 \\
18.68823 & 3.16446 & 1.50805 & 17.32347 & 1.69991 \\
18.64087 & 2.98993 & 0.12039 & 16.99151 & 0.19105 \\
18.61074 & 3.22549 & 1.29684 & 16.86789 & 0.71561 \\
18.44278 & 3.45615 & 1.57978 & 16.83643 & 1.66457 \\
18.09945 & 3.14021 & 1.21792 & 16.43145 & 0.49750 \\
18.38535 & 3.04870 & 1.14062 & 16.71568 & 0.22842 \\
17.79826 & 3.39592 & 1.48282 & 16.50462 & 1.32604 \\
17.78191 & 3.39580 & 1.48291 & 16.50582 & 1.32671 \\
18.35972 & 3.58101 & 1.52426 & 16.33421 & 1.32906 \\
17.22711 & 3.59209 & 1.60423 & 15.58037 & 1.69014 \\
17.77180 & 3.39579 & 1.48292 & 16.50594 & 2.32678 \\
17.76703 & 3.39577 & 1.48294 & 16.50618 & 1.32691 \\
16.77068 & 3.31163 & 1.53185 & 15.34145 & 1.51308 \\
16.93565 & 3.66458 & 1.67425 & 15.31899 & 1.94164 \\
20.00000 & 3.54836 & 1.56404 & 15.27421 & 1.61542 \\
16.85458 & 3.55123 & 1.56708 & 15.27793 & 1.62441 \\
17.97974 & 4.13369 & 2.01351 & 15.97567 & 3.48435 \\
18.32035 & 3.62799 & 1.75527 & 16.74473 & 2.24213 \\
16.73685 & 2.13746 & 0.80731 & 17.67532 & 1.97316
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
-0.07754 \\
-0.06529 \\
-0.05691 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.07497 \\
-0.07559 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.07614 \\
-0.05004 \\
100.00000 \\
0.03854 \\
-0.07517 \\
100.00000 \\
100.00000 \\
100.00000
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
17.66769 & 3.26196 & 1.36852 & 16.50322 & 0.93634 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.68620 & 3.14791 & 1.18350 & 16.88882 & 0.37021 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
17.81244 & 3.39728 & 1.48041 & 16.48733 & 1.31517 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
17.80227 & 3.35913 & 1.47637 & 16.49882 & 1.32197 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.85458 & 3.55123 & 1.56708 & 15.27793 & 1.62441 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.64507 & 3.23462 & 1.34436 & 16.50226 & 0.85664 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993 \\
16.73685 & 2.13746 & 0.80731 & 17.67532 & 1.97316
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07754 \\
-0.07213 \\
-0.06566 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.07503 \\
-0.08109 \\
-0.07402 \\
-0.07331 \\
-0.06675 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.07517 \\
-0.07391 \\
-0.07573 \\
-0.06190 \\
-0.05583 \\
100.00000
\end{pmatrix}}$$

Generation 80

$$\xrightarrow{\text{Selektion}} \underset{\text{Nachkommen}_{\text{sel}}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
17.66769 & 3.26196 & 1.36852 & 16.50322 & 0.93634 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.68620 & 3.14791 & 1.18350 & 16.88882 & 0.37021 \\
17.81244 & 3.39728 & 1.48041 & 16.48733 & 1.31517 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
16.85458 & 3.55123 & 1.56708 & 15.27793 & 1.62441 \\
16.73685 & 2.13746 & 0.80731 & 17.67532 & 1.97316 \\
17.64507 & 3.23462 & 1.34436 & 16.50226 & 0.85664 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993
\end{pmatrix}} \xrightarrow{\text{Rekombination}} \underset{\text{Nachkommen}_{\text{rek}}}{\begin{pmatrix}
18.70560 & 3.22666 & 1.64789 & 17.45342 & 2.24770 \\
18.70376 & 3.23788 & 1.65637 & 17.41286 & 2.29838 \\
18.43433 & 3.45876 & 1.66885 & 17.00209 & 1.99245 \\
17.91382 & 4.23156 & 2.06349 & 15.82682 & 3.12819 \\
18.64823 & 3.01706 & 1.18066 & 17.04372 & 0.42623 \\
18.62408 & 2.92803 & 0.98290 & 16.87239 & -0.34546 \\
18.47873 & 3.20311 & 1.54863 & 17.27196 & 1.84094 \\
17.91933 & 3.24370 & 1.42440 & 16.74174 & 1.21702 \\
18.68145 & 3.14823 & 1.18419 & 16.88552 & 0.37122 \\
18.67912 & 3.14838 & 1.18453 & 16.88390 & 0.37172 \\
18.47403 & 3.27172 & 1.60513 & 17.20002 & 2.00990 \\
18.64677 & 3.23893 & 1.63770 & 17.38610 & 2.19130 \\
18.55804 & 3.45677 & 1.41271 & 16.63733 & 0.98788 \\
18.55599 & 3.46207 & 1.41663 & 16.63311 & 0.99848 \\
16.99318 & 3.52678 & 1.55036 & 15.43670 & 1.56566 \\
17.52787 & 3.33976 & 1.38951 & 15.92489 & 1.04128 \\
17.75732 & 3.53515 & 1.57258 & 16.36338 & 1.60097 \\
17.66009 & 4.53915 & 2.21847 & 15.33601 & 3.57608 \\
16.76202 & 2.43970 & 0.96974 & 17.16279 & 1.89860 \\
16.75354 & 2.33784 & 0.91500 & 17.33552 & 1.92373 \\
17.76025 & 3.38225 & 1.47129 & 16.50567 & 1.28731 \\
17.77044 & 3.39531 & 1.48251 & 16.50598 & 1.32541 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993
\end{pmatrix}}$$

$$\xrightarrow{\text{Mutation}} \underset{\text{Nachkommen}_{\text{mut}}}{\begin{pmatrix}
18.70560 & 3.22666 & 1.64789 & 17.45342 & 2.24770 \\
18.71157 & 3.23398 & 1.65246 & 17.41286 & 2.29838 \\
18.43433 & 3.45876 & 1.66885 & 17.00209 & 1.99245 \\
17.91382 & 4.23156 & 2.06349 & 15.82682 & 3.12819 \\
18.64823 & 3.01706 & 1.18066 & 17.04372 & 0.42623 \\
18.62408 & 2.92803 & 0.98290 & 16.93489 & -0.34546 \\
18.47873 & 3.20311 & 1.54863 & 17.27196 & 1.84094 \\
17.91933 & 3.24370 & 1.42440 & 16.74174 & 1.21702 \\
18.68145 & 3.14823 & 1.18419 & 16.88552 & 0.37122 \\
18.67912 & 2.64741 & 1.18453 & 16.88390 & 0.37172 \\
18.22403 & 3.27172 & 1.60513 & 17.20002 & 2.25990 \\
18.64677 & 3.23893 & 3.13770 & 17.38610 & 2.19130 \\
18.55804 & 3.45677 & 1.49279 & 16.79749 & 0.98788 \\
18.55599 & 3.46207 & 1.41663 & 16.63311 & 0.99848 \\
17.99318 & 3.02678 & 1.55036 & 16.43670 & 1.56566 \\
17.52787 & 3.33976 & 1.38951 & 15.92489 & 1.04128 \\
17.75732 & 3.53515 & 1.57258 & 16.36338 & 1.60097 \\
17.66009 & 4.53915 & 2.21847 & 15.33601 & 3.57608 \\
16.76202 & 2.43970 & 0.96974 & 17.16279 & 1.89860 \\
16.75354 & 2.33784 & 0.91500 & 17.33552 & 1.92373 \\
17.76025 & 3.38225 & 1.47129 & 16.50567 & 1.28731 \\
17.73919 & 3.39531 & 1.48251 & 16.50598 & 1.32541 \\
16.87736 & 3.55344 & 1.66130 & 15.44819 & 1.93993
\end{pmatrix}} \xrightarrow{\text{Bewertung}} \underset{\vec{z}_{\text{mut}}}{\begin{pmatrix}
-0.07587 \\
-0.06452 \\
-0.05873 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
100.00000 \\
-0.00812 \\
100.00000 \\
0.63144 \\
-0.03638 \\
-0.07396 \\
-0.00680 \\
100.00000 \\
100.00000 \\
-0.05412 \\
100.00000 \\
100.00000 \\
-0.07805 \\
100.00000 \\
-0.06044
\end{pmatrix}}$$

$$\xrightarrow{\text{Wiedereinfügen}} \underset{\text{Population}}{\begin{pmatrix}
18.71357 & 3.17807 & 1.61119 & 17.62899 & 2.02836 \\
17.66769 & 3.26196 & 1.36852 & 16.50322 & 0.93634 \\
18.35356 & 3.57868 & 1.73009 & 16.81971 & 2.16869 \\
18.68620 & 3.14791 & 1.18350 & 16.88882 & 0.37021 \\
18.63508 & 2.96859 & 1.07300 & 16.95045 & 0.00611 \\
18.70541 & 3.22780 & 1.64875 & 17.44928 & 2.25288 \\
18.86210 & 3.17530 & 1.63377 & 17.63533 & 2.26854 \\
18.67598 & 3.13591 & 1.18695 & 16.88010 & 0.34705 \\
18.67733 & 3.14850 & 1.18479 & 16.88265 & 0.37210 \\
18.63394 & 2.96913 & 1.07341 & 16.94922 & 0.00804 \\
17.81244 & 3.39728 & 1.48041 & 16.48733 & 1.31517 \\
17.79642 & 3.39592 & 1.48283 & 16.50471 & 1.32609 \\
17.80227 & 3.35913 & 1.47637 & 16.49882 & 1.32197 \\
18.51113 & 3.57797 & 1.50233 & 16.54087 & 1.23001 \\
18.08727 & 3.14410 & 1.22121 & 16.43564 & 0.49267 \\
16.96679 & 3.53600 & 1.55830 & 15.41261 & 1.59153 \\
17.77081 & 3.39579 & 1.48292 & 16.50599 & 1.32680 \\
17.64928 & 4.65073 & 2.29025 & 15.22184 & 3.79557 \\
16.88591 & 3.59585 & 1.63131 & 15.32576 & 1.81241 \\
16.85458 & 3.55123 & 1.56708 & 15.27793 & 1.62441 \\
16.85461 & 3.55125 & 1.56710 & 15.27796 & 1.62447 \\
17.76025 & 3.38225 & 1.47129 & 16.50567 & 1.28731 \\
17.46467 & 4.89841 & 2.40404 & 14.81268 & 4.10821 \\
16.87736 & 3.55344 & 1.66130 & 15.32222 & 1.93993 \\
16.73685 & 2.13746 & 0.80731 & 17.67532 & 1.97316
\end{pmatrix}} \underset{\vec{z}_{\text{Pop}}}{\begin{pmatrix}
-0.07928 \\
-0.07754 \\
-0.07213 \\
-0.06566 \\
-0.06780 \\
-0.07437 \\
-0.07378 \\
-0.07434 \\
-0.07553 \\
-0.06782 \\
-0.07503 \\
-0.08109 \\
-0.07402 \\
-0.07331 \\
-0.06675 \\
-0.07720 \\
-0.07608 \\
-0.06485 \\
-0.06472 \\
-0.07517 \\
-0.07391 \\
-0.07805 \\
-0.06190 \\
-0.05583 \\
100.00000
\end{pmatrix}}$$

Literaturverzeichnis

[1] ADAC. So bremst der Assistent - Automatische Helfer sind ein Sicherheitsplus. *Hyperlink: http://www.adac.de/*, 2001.

[2] Ammon, D. Künftige Fahrdynamik- und Assistenzsysteme. *atp 46(2004) Heft 6*, 2004.

[3] Athanasas, K. *Fast Prototyping Methodology for the Verification of Complex Vehicle Systems.* PhD thesis, Brunel University, London, UK, March 2005.

[4] Balzert, H. *Lehrbuch der Software-Technik.* Spektrum Akademischer Verlag, 1998.

[5] Baresel, A. Automatisierung von Strukturtests mit evolutionären Algorithmen. *Diplomarbeit, Humboldt-University zu Berlin*, Juli 2000.

[6] Beizer, B. *Black Box Testing: Techniques for Functional Testing of Software and Systems.* John Wiley & Sons Inc., New York, April 1995.

[7] Boehm, B.W. Software Engineering Economics. *Prentice-Hall, 1. Auflage*, 1981.

[8] Branke, J. Evolutionäre Optimierung dynamischer Probleme. *it - Information Technology, 3/2003*, 2003.

[9] Burton, S. Towards Automated Unit Testing of Statechart Implementations (YCS 319). Technical report, Department of Computer Science, University of York, August 1999.

[10] Conrad, M. *Modell-basierter Test eingebetteter Software im Automobil.* PhD thesis, Technische Universität Berlin, 2004.

[11] Darwin, C. *The Origin of Species.* John Murray, London, 1859.

[12] DIN EN 61508-4. Funktionale Sicherheit sicherheitsbezogener elektrischer/ elektronischer/ programmierbar elektronischer Systeme - Teil 4: Begriffe und Abkürzungen, Deutsche Fassung EN 61508-4:2001. *VDE Verlag*, 2001.

[13] Ehrenberger, W. *Software Verifikation.* Hanser, 2002.

[14] Griep, A. Kompakte Beschreibung von Testsequenzen für die automatisierte Testfallerstellung unter Beachtung regelungstechnischer Aspekte. *Diplomarbeit, Technische Universität Ilmenau, Fakultät für Informatik und Automatisierung, Institut für Automatisierungs- und Systemtechnik*, 2002.

[15] Grimm, K. Systematisches Testen von Software - Eine neue Methode und eine effektive Teststrategie. *GMD-Berichte*, Nr. 251, 1995.

[16] Groß, H.-G. *Measuring Evolutionary Testability of Real-Time Software*. PhD thesis, University of Glamorgan / Prifysgol Morgannwg, 2000.

[17] Groß, H.-G. A prediction system for dynamic optimisation-based execution time analysis. *ICSE 2001, Proceedings of the 1st International Workshop on Software Engineering using Metaheuristic Innovative Algorithms (SEMINAL 2001), Toronto, Canada*, May 2001.

[18] Heinisch, C. Konfigurationsmodell und Architektur für eine automatisierte Software-Aktualisierung von Steuergeräten im Automobil. *Dissertation, Universität Tübingen*, 2004.

[19] IEEE Standards Board. IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990. *IEEE Publications*, Januar 1991.

[20] Jochim, M. Zeitig steuern - Sichere Datenübertragung im Automobil. *c't magazin, 2/2007, Heise Verlag*, Januar 2007.

[21] Kallenbach, R. Fahrerassistenz-Systeme: Das feinfühlige Auto. *Automobil Elektronik, Ausgabe 3*, Juni 2006.

[22] Kroschel, K. *Statistische Informationstechnik*. Springer Verlag, 2003.

[23] Liggesmeyer, P. *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag, Heidelberg, Berlin, 2002.

[24] McMinn, P. Search-based Software Test Data Generation: A Survey. *Journal on Software Testing, Verification, and Reliability, Vol. 14, No. 2, pp. 105–156, June*, 2004.

[25] MISRA Consortium. *Guidelines For The Use Of The C Language In Vehicle Based Software*. The Motor Industry Research Association, Nuneaton Warwickshire UK, 1998.

[26] Myers, G.J. *Methodisches Testen von Programmen*. R. Oldenbourg Verlag, Wien, 4. edition, 1991.

[27] Pohlheim, H. *Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise für die Praxis*. Springer, 1999.

[28] Pohlheim, H. *Genetic and Evolutionary Algorithm Toolbox for use with Matlab*. Springer, 1999.

[29] Richter, R. Fahrerassistenzsysteme: Radar – Mehr Sicherheit im Lkw. *Automobiltechnische Zeitschrift ATZ 09|2006*, September 2006.

[30] Rosenstiel, W. Entwurf und Entwurfsmethodik eingebetteter Systeme. *Abschlussbericht DFG-Schwerpunktprogramm 1040, Universität Tübingen*, 1997 - 2003.

[31] Royce, W.W. Managing the Development of Large Software Systems. *Proceedings, IEEE Wescon, Seiten 1-9*, August 1970.

[32] Schach, S. R. *Classical and Object-Oriented Software Engineering with UML and C++*. McGraw Hill Higher Education, fourth edition, 1999.

[33] Simmes, D. *Entwicklungsbegleitender Systemtest für elektronische Fahrzeugsteuergeräte*. PhD thesis, Technische Universität München, 1997.

[34] Sthamer, H. *The automatic generation of software test data using genetic algorithms*. PhD thesis, University of Glamorgan, Pontyprid, Wales, Great Britain, 1996.

[35] Thaller, G.E. *Software-Qualität*. VDE Verlag, Berlin, 1. edition, 2000.

[36] Tracey, N. *A Search-Based Automated Test-Data Generation Framework for Safety-Critical Software*. PhD thesis, University of York, Department of Computer Science, 2001.

[37] Ostrand, T. und Balcer, M. The Category-Partition Method for Specifying and Generating Functional Tests. *Communications of the ACM*, 31. Jahrgang, Heft 6, S. 676 - 686, 1988.

[38] Wegener, J. und Baresel, A. und Sthamer, H. Evolutionary test environment for automatic structural testing. *Special Issue of Information and Software Technology devoted to the Application of Meta-heuristic Algorithms to Problems in Software Engineering*, 2001.

[39] Sthamer, H. und Baresel, A. und Wegener, J. Evolutionary Testing of Embedded Systems. *Proceedings of the 14th International Software Quality Week (QW '01), San Francisco, USA*, May 2001.

[40] Hauser, J. und Borberg, B. und Richter, A. und Grams, G. and Altmannsberger, B. und Sudbrack, S. Einführung eines Entwicklungsprozesses in der BMW Elektrik- / Elektronik-Entwicklung. *OBJEKTspektrum, 2/2005*, März/April 2005.

[41] Wegener, J. und Buhr, K. und Pohlheim, H. Automatic Test Data Generation for Structural Testing of Embedded Software Systems by Evolutionary Testing. *Proceedings of the Genetic and Evolutionary Computation Conference(GECCO 2002), New York(NY), USA*, July 2002.

[42] Müller-Glasser, K.D. und Burst, A. und Spitzer, B. und Kühl, M. Rapid Prototyping von eingebetteten elektronischen Systemen. *it + ti - Informationstechnik und Technische Informatik 42 (2000) 2*, 2/2000.

[43] Müller-Glasser, K.D. und Burst, A. und Spitzer, B. und Schmerler, S. Rapid Prototyping von Informationssystemen für Kraftfahrzeuge. *it + ti - Informationstechnik und Technische Informatik 41 (1999) 5*, 5/1999.

[44] Tracey, N. und Clark, J. und Mander, K. The way forward for unifying dynamic test-case generation: The optimisation-based approach. *International Workshop on Dependable Computing and Its Applications*, 1998.

[45] Tracey, N. und Clark, J. und Mander, K. Automated Program Flaw Finding using Simulated Annealing. *Software Engineering Notes, Proceedings of the International Symposium on Software Testing and Analysis, ACM SIGSOFT, Vol. 23, Nr. 2, pp. 73–81*, March 1998.

[46] Tracey, N. und Clark, J. und Mander, K. und McDermid, J. An Automated Framework for Structural Test-Data Generation. In *Proceedings of the 13th IEEE Conference on Automated Software Engineering*, Hawaii, USA, 1998.

[47] Tracey, N. und Clark, J. und Mander, K. und McDermid, J. Automated test-data generation for exception conditions. *Software Practice and Experience, Vol. 30, Nr. 1, pp. 61–79*, 2000.

[48] Stürmer, I. und Conrad, M. und Fey, I. und Dörr, H. Experiences with Model and Autocode Reviews in Model-based Software Development. In *Proceedings of the 2006 international workshop on Software engineering for automotive systems, SEAS'06, Shanghai, China*. ACM Press, 2006.

[49] Dijkstra, E. W. und Dahl, O. J. und Hoare, C. A. R. *Structured programming*. Academic Press, 1972.

[50] Otterbach, R. und Eckmann, M. und Mertens, F. Rapid Control Prototyping - neue Möglichkeiten und Werkzeuge. *atp 46(2004) Heft 6*, 2004.

[51] Kirkpatrick, S. und Gelatt Jr, C.D. und Vecchi, M.P. Optimization by simulated annealing. *Science 220, pp. 671-680*, 1983.

[52] Fewster, M. und Graham, D. *Software Test Automation*. Addison-Wesley, 1999.

[53] Schultz, A. C. und Grefenstette, J. J. und De Jong, K. A. Test and Evaluation by Genetic Algorithms. *IEEE Expert*, 8(5), 1993.

[54] Grochtmann, M. und Grimm, K. Classification Trees for Partition Testing. *Software Testing, Verification and Reliability*, Bd. 3, Nr. 2, S. 63-82, 1993.

[55] Wegener, J. und Grochtmann, M. Werkzeugunterstützte Testfallermittlung für den funktionalen Test mit dem Klassifikationsbaum-Editor CTE. *Proceedings der GI-Fachtagung Software-Technik ´93, Dortmund*, 1993.

[56] Wegener, J. und Grochtmann, M. Verifying timing constraints of real-time systems by means of Evolutionary Testing. *Real-Time Systems*, 15(3):275 – 298, 1998.

[57] Holzmann, H. und Hahn, K.M. Einsatz von HiL-Simulation im Entwicklungsprozess von modernen Fahrwerkregelsystemen am Beispiel Integrated Chassis Control (ICC). *atp 46(2004) Heft 6*, 2004.

[58] Pargas, R. und Harrold, M. und Peck, R. Test-Data Generation Using Genetic Algorithms. *Software Testing, Verification and Reliability*, 9(4), 1999.

[59] McMinn, P. und Holcombe, M. The State Problem for Evolutionary Testing. *Proceedings of the Genetic and Evolutionary Computation Conference(GECCO 2003), pp. 2488 - 2497, Chicago(IL), USA*, July 2003.

[60] Harman, M. und Hu, L. und Hierons, R. und Baresel, A. und Sthamer, H. Improving Evolutionary Testing by Flag Removal. *Proceedings of the Genetic and Evolutionary Computation Conference(GECCO 2002), New York(NY), USA*, July 2002.

[61] Collins, R. J. und Jeffeson, D. R. Selection in Massively Parallel Genetic Algorithmst. *Proceedings of the Fourth International Converence on Genetic Algorithms, pp. 249 – 256, Morgan Kaufmann Publishers, San Mateo, California, USA*, 1991.

[62] Groß, H.-G. und Jones, B. und Eyres, D. Evolutionary Algorithms for the verification of execution time bounds for Real-Time Software. *IEE Workshop on Applicable Modeling, Verification and Analysis Techniques, London, GB*, January 1999.

[63] Burke, E.K. und Kendall, G. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer-Verlag, Berlin, Heidelberg, New York, 2005.

[64] Gerdes, I. und Klawonn, F. und Kruse, R. *Evolutionäre Algorithmen*. Vieweg, Wiesbaden, 2004.

[65] Theodoridis, S. und Koutroumbas, K. *Pattern Recognition*. Elsevier Academic Press, 2006.

[66] Glover, F. und Laguna, M. *Tabu Search*. Kluwer, Boston, MA, USA, 1997.

[67] Wappler, S. und Lammermann, F. Using Evolutionary Algorithms for the Unit Testing of Object-Oriented Software. *In Proceedings of the Genetic and Evolutionary Compuation Conference (GECCO 2005), pages 1053–1060, Washington D.C., USA*, June 2005.

[68] Sadeghipour, S. und Lim, M. Einsatz automatischer Testvektorgenerierung im modellbasierten Test. In *Informatik 2005, Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Infomatik e.V. (GI), 19.-22. September 2005*, Bonn, 2005.

[69] Spillner, A. und Linz, T. *Basiswissen Softwaretest.* dpunkt.verlag, Heidelberg, 2003.

[70] Puschner, P. und Nossal, R. Testing the results of static worst-case execution-time analysis. *In Proceedings of the 19th IEEE Real-Time Systems Symposium*, 1998.

[71] Broekman, B. und Notenboom, E. *Testing Embedded Software.* Addison-Wesley, London, 2003.

[72] Hatley, Derek J. und Pirbhai, Imtiaz A. *Strategies for Real-Time System Specification.* Dorset House Publishing, 353 West 12th Street, New York, USA, 1987.

[73] Wegener, J. und Pitschinetz, R. und Sthamer, H. Automated Testing of Real-Time Tasks. *ICSE 2000, Proceedings of the 22nd International Conference on on Software Engineering, June 4-11, 2000, Limerick Ireland. ACM*, 2000.

[74] Baresel, A. und Pohlheim, H. und Sadeghipour, S. Structural and Functional Sequence Test of Dynamic and State-Based Software with Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003), pp. 2428-2441*, Chicago(IL), USA, July 2003.

[75] Jain, B. J. und Pohlheim, H. und Wegener, J. On Termination Criteria of Evolutionary Algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference(GECCO 2001), San Francisco(CA), USA*, July 2001.

[76] Kubin, H. und Preiß, H. und Vetter, C. und Walther, D. Rapid-Prototyping mit VME-Bus-Echtzeitrechner unter Nutzung von MATLAB/Simulink. *atp 44 (2002) Heft 9*, September 2002.

[77] Robertson, S. und Robertson, J. *Mastering the Requirements Process.* Addison-Wesley, 1. edition, 1999.

[78] Liggesmeyer, P. und Rombach, D. *Software Engineering eingebetteter Systeme.* Spektrum Akademischer Verlag, Heidelberg, Berlin, 2005.

[79] Spillner, A. und Roßner, T. und Winter, M. und Linz, T. *Praxiswissen Softwaretest Testmanagement.* dpunkt.verlag, Heidelberg, 2006.

[80] Booch, G. und Rumbaugh, J. und Jacobson, I. *The Unified Modeling Language User Guide.* Addison-Wesley, 2005.

[81] Conrad, M. und Sadeghipour, S. Einsatz von Überdeckungskriterien auf Modellebene - Erfahrungsbericht und experimentelle Ergebnisse. *Softwaretechnik-Trends 22 (2002) 2*, Mai 2002.

[82] Mühlenbein, H. und Schlierkamp-Voosen, D. Predictive Models for the Breeder Genetic Algorithm, I. Continuous Parameter Optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

[83] Hermes, T. und Schultze, A. und Predelli, O. Automatische Testvektorgenerierung in der modellbasierten Softwareentwicklung. In *Informatik 2005, Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Infomatik e.V. (GI), 19.-22. September 2005*, Bonn, 2005.

[84] Nadler, M. und Smith, E. P. *Pattern Recognition Engineering.* Wiley Interscience, 1993.

[85] Baresel, A. und Sthamer, H. Evolutionary Testing of Flag Conditions. *Proceedings of the Genetic and Evolutionary Computation Conference(GECCO 2003), pp. 2442 – 2454, Chicago(IL), USA*, July 2003.

[86] Wegener, J. und Sthamer, H. und Baresel, A. Application Fields for Evolutionary Testing. *Proceedings of the 9th European International Conference on Software Testing Analysis and Review (Eurostar 2001), Stockholm, Sweden*, November 2001.

[87] Jones, B.-F. und Sthamer, H. und Eyres, D. Automatic structural testing using genetic algorithms. *Software Engineering Journal*, 11(5), 1996.

[88] Baresel, A. und Sthamer, H. und Schmidt, M. Fitness Function Design to improve Evolutionary Structural Testing. *Proceedings of the Genetic and Evolutionary Computation Conference(GECCO 2002), New York(NY), USA*, July 2002.

[89] Sullivan, M. O' und Vössner, S. und Wegener, J. Testing Temporal Correctness of Real-Time Systems - A New Approach Using Genetic Algorithms and Cluster Analysis. *Proceedings of EuroSTAR'98, pp. 397–418*, 1998.

[90] Lammermann, F. und Wappler, S. Benefits of Software Measures for Evolutionary White-Box Testing. *In Proceedings of the Genetic and Evolutionary Compuation Conference (GECCO 2005), pages 1083–1084, Washington D.C., USA*, June 2005.

[91] Tlili, M. und Wappler, S. und Sthamer, H. Improving Evolutionary Real-Time Testing. *Proceedings of the 2006 Conference on Genetic and evolutionary compuation (GECCO 2006), New York, USA*, July 2006.

[92] Buehler, O. und Wegener, J. Evolutionary Functional Testing of an Automated Parking System. In *Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT '03) and the 9th. International Conference on Information Systems Analysis and Synthesis (ISAS '03)*, Orlando, Florida, USA, 2003. CCCT '03 and ISAS '03.

[93] Buehler, O. und Wegener, J. Automatic Testing of an Autonomous Parking System Using Evolutionary Computation. In *SAE Technical Paper Series 2004-01-0459*, Detroit, Michigan, USA, March 8-11 2004. SAE World Congress.

[94] Buehler, O. und Wegener, J. Evaluation of Different Fitness Functions for the Evolutionary Testing of an Autonomous Parking System. In *Proceedings of the Genetic and Evolutionary Computation Conference, Part 2 - GECCO 2004*, Seattle, Washington, USA, June 2004.

[95] Buehler, O. und Wegener, J. Evolutionary Functional Testing of a Vehicle Brake Assistant System. In *Proceedings of the 6th Metaheuristics International Conference (MIC2005)*, Vienna, Austria, August 2005.

[96] Buehler, O. und Wegener, J. Evolutionary functional testing. *Computers and Operations Research, Elsevier Ltd., doi:10.1016/j.cor.2007.01.015*, 2007.

[97] Lammermann, F. und Wegener, J. Test-Goal-Specific Termination Criteria for Evolutionary White-Box Testing by Means of Software Measures. In *Proceedings of the 6th Metaheuristics International Conference (MIC2005)*, Vienna, Austria, August 2005.

[98] Mueller, F. und Wegener, J. A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints. *Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium, Denver, USA*, June 1998.

[99] Wappler, S. und Wegener, J. Evolutionary Unit Testing of Object-Oriented Software using a Hybrid Evolutionary Algorithm. *IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16–21*, 2006.

[100] Wappler, S. und Wegener, J. Evolutionary unit testing of object-oriented software using strongly-typed genetic programming. *Proceedings of the 8th annual conference on Genetic and evolutionary compuation (GECCO 2006), pages 1925–1932, New York, USA*, July 2006.

[101] Sthamer, H. und Wegener, J. und Baresel, A. Using Evolutionary Testing to improve Efficiency and Quality in Software Testing. *Proceedings of the 2nd Asia-Pacific Conference on Software Testing Analysis and Review(AsiaSTAR2002), Melbourne, Australia*, July 2002.

[102] Pohlheim, H. und Wegener, J. und Sthamer, H. Testing the Temporal Behavior of Real-Time Engine Control Software Modules using Extended Evolutionary Algorithms. *VDI-Berichte 1526, VDI-Verlag*, 2000.

[103] Schäuffele, J. und Zurawka, T. *Automotive Software Engineering - Grundlagen, Prozesse, Methoden und Werkzeuge.* Vieweg Verlag, 2003.

[104] Veenendaal, E. v. Standard glossary of terms used in Software Testing. *Version 1.2, Produced by the 'Glossary Working Party', International Software Testing Qualification Board*, Juni 2006.

[105] Vieweg, C. *S-Klasse - Meisterstück auf Rädern. Geschichte - Design - Technik.* Delius Klasing Verlag, Juni 2006.

[106] Wegener, J. *Evolutionärer Test des Zeitverhaltens von Realzeit-Systemen.* PhD thesis, Humboldt-Universität zu Berlin, 2001.

[107] Wegener, J. Evolutionärer Test des Zeitverhaltens von Realzeit-Systemen. *it - Information Technology 46 (2004) 2, Oldenbourg Verlag*, 2004.

[108] Weicker, K. *Evolutionäre Algorithmen.* Teubner, März 2002.

[109] Wintermantel, M. *ARS300 Radarsensor dritter Generation für ACC und mehr.* Continental Automotive Systems, Technologie im Dialog, Juni 2005.

[110] Yap, A. *Beitrag zur mathematischen Modellierung der Fahrspurfindung im Fahrzeugführungsproblem.* VDI Verlag, April 1999.

[111] Zomotor, A. *Fahrwerktechnik: Fahrverhalten.* Vogel Buchverlag, Würzburg, 1987.