Universität
Augsburg
University

Fakultät für Informatik
Organic Computing

Vergleich verschiedener Rekombinationsalgorithmen für Cartesian Genetic Programming

Masterarbeit

Cindy Ebertz

Abgabedatum 26. März 2025
Matrikelnummer 1542003
Studiengang Master Ingenieurinformatik

Gutachter Prof. Dr. Jörg Hähner
Prof. Dr.-Ing. Lars Mikelsons

Zusammenfassung

Cartesian Genetic Programming (CGP) kann verwendet werden, um verschiedene Eingangs-Ausgangs-Beziehungen automatisiert durch Training eines CGP-Modells herzustellen. Teil davon sind die genetischen Operatoren der Mutation und Rekombination. Aussagen darüber, dass Rekombination in Standard-CGP die Effektivität des Trainings nicht sinnvoll steigere, basieren auf Papern aus den Jahren 1999 und 2011, werden jedoch in modernen Arbeiten weiterhin aufgegriffen. Jene Aussagen werden in dieser Arbeit kritisch hinterfragt und experimentell erforscht. Außerdem werden unterschiedliche Rekombinationsstrategien miteinander verglichen und bewertet, wobei der Schwerpunkt auf der Berechnung der Rekombinationsrate liegt. Die Ergebnisse in dieser Arbeit zeigen, dass Rekombination in Standard-CGPs durchaus effektiv eingebaut werden kann, wenn diese sinnvoll konfiguriert wird. Empfohlen wird eine hohe Rekombinationsrate zu Beginn des Trainings, wobei in späteren Iterationen die Rekombination stark reduziert oder vollkommen abgeschafft werden sollte. Außerdem wird aufgezeigt, dass Rekombination ein Potential besitzt, das durch gezielte Änderung des standardisierten CGP-Verfahrens effektiver genutzt werden könnte. Basierend auf den Erkenntnissen dieser Arbeit werden Empfehlungen für zukünftige Forschungsansätze gegeben, die das untersuchte Themengebiet weiter vertiefen und voranbringen können.

Inhaltsverzeichnis

Abbildungsverzeichnis **vii**

1 Motivation und Aufbau **1**

2 Grundlagen **3**
2.1 Aufbau Standard-CGP und initiale Population 4
2.2 Evaluation und Selektion 8
2.3 Evolutionärer Operator: Rekombination 9
2.3.1 One-Point Rekombination 9
2.3.2 Two-Point Rekombination 10
2.3.3 Uniform Rekombination 11
2.3.4 Rekombinationsraten 13
2.4 Evolutionärer Operator: Mutation 16
2.5 Evaluation Fitness, Stopp-Kriterium und Ergebnis 18
2.6 Bayes'sche Analyse 18

3 Experimente und Evaluation **21**
3.1 Aufbau der Experimente 21
3.1.1 Testszenarien 21
3.1.2 CGP-Konfigurationen 23
3.1.3 Hyperparameteroptimierung 25
3.1.4 Teststruktur 26
3.2 Evaluation 28

4 Ergebnisse **31**
4.1 Ergebnisse Rohdatenanalyse 31
4.1.1 Rohdatenanalyse: Parity 32
4.1.2 Rohdatenanalyse: Keijzer 36
4.1.3 Rohdatenanalyse: Encode 39
4.1.4 Rohdatenanalyse: Koza 43
4.1.5 Rohdatenanalyse: Zusammenfassung 46

4.2 Ergebnisse Bayes'sche Analyse 50
4.2.1 Bayes'sche Analyse: Parity 51
4.2.2 Bayes'sche Analyse: Keijzer 53
4.2.3 Bayes'sche Analyse: Encode 55
4.2.4 Bayes'sche Analyse: Koza 57
4.2.5 Bayes'sche Analyse: Zusammenfassung 59

5 Fazit aus Ausblick **63**

A Anhang **I**
A.1 Tabellen Rohdatenanalyse II
A.1.1 Parity: Tabellen Rohdatenanalyse II
A.1.2 Keijzer: Tabellen Rohdatenanalyse IV
A.1.3 Encode: Tabellen Rohdatenanalyse VII
A.1.4 Koza: Tabellen Rohdatenanalyse XI
A.2 Tabellen Bayes'sche Analyse XIV
A.2.1 Parity: Tabelle Bayes'sche Analyse XIV
A.2.2 Keijzer: Tabelle Bayes'sche Analyse XV
A.2.3 Encode: Tabellen Bayes'sche Analyse XVI
A.2.4 Koza: Tabellen Bayes'sche Analyse XIX

Abbildungsverzeichnis

2.1 Aufbau Standard-CGP, angelehnt an [TST22] 3
2.2 Darstellungsmöglichkeiten eines Chromosoms, angelehnt an [TST22] . . . 4
2.3 One-Point Rekombination, angelehnt an [TST22] 10
2.4 Two-Point Rekombination, angelehnt an [TST22] 11
2.5 Uniform Rekombination, angelehnt an [TST22] 12
2.6 Single Active Mutation, angelehnt an [TST22] 17
2.7 Bayes'sche Analyse, angelehnt an [Nen80] 19

4.1 Encode: Anzahl positive Rekombinationen für die verschiedenen Arten der Rekombinationsrate 42
4.2 Koza: Anzahl positive Rekombinationen für die verschiedenen Arten der Rekombinationsrate 45

1 Motivation und Aufbau

Das klassische Genetic Programming (GP) wird heutzutage für die Problemlösung in den unterschiedlichsten Domänen erforscht. Beispiele hierfür sind die Erstellung einer mathematischen Gleichung für einen industriellen Prozess [SB01], die Strukturanalyse von FGMs (funktionell gradierten Materialien) [Dem+22] und der Verarbeitung von natürlicher Sprache [Ara20].

Cartesian Genetic Programming (CGP) ist eine Methode des GPs, in der Lösungen für Probleme als Graphen dargestellt werden [Mil20]. Im Standard-CGP werde laut Miller der Rekombinationsschritt normalerweise nicht ausgeführt und in den meisten Arbeiten werde dieser Schritt gänzlich ignoriert. Er bezieht dieses Verhalten auf Forschungsergebnisse aus dem Jahr 1999, die nach Miller aufzeigen, dass der Rekombinationsschritt kaum einen Effekt auf die Effizienz von CGP habe. [Mil20] In mehreren weiteren Papern werden andere Rekombinationsalgorithmen mit dem Ziel vorgestellt, dass der Rekombinationsschritt neben der Mutation sinnvoll in CGP eingebaut werden kann. Dieser weitere Operator könnte die Effizienz von CGP-Modellen im Training steigern und somit komplexere Ausgangsprobleme lösbar machen. Innerhalb dieser Paper wird als Prämisse angenommen, dass wie von Miller geschildert, in Standard-CGP keine Rekombination verwendet wird. [CWM07; Kal20; TST22]
Da die Aussage, dass in Sandard-CGP der Rekombinationsschritt nicht zielführend sei, auf den Papern von Miller aus dem Jahr 1999 und 2011 basiert, kommt die Frage auf, ob dies immer noch zutrifft. Die erste Forschungsfrage, die in dieser Arbeit beantwortet werden soll, ist demnach: „Kann nach heutigem Forschungsstand Rekombination die Effizienz von Standard-CGP steigern?“

Da der Erfolg der Rekombination ebenfalls von der Rekombinationsrate abhängt, ist es sinnvoll diese näher zu betrachten. Clegg et al. und Torabi et al. beschreiben in ihren Papern unterschiedliche Herangehensweisen an die Rekombinationsrate [CWM07; TST22]: Clegg et al. führen eine variable Rekombinationsrate in ihrem Paper ein. Dabei wird eine hohe Rekombinationsrate linear verringert, sodass in den letzten Lernschritten keine Rekombination mehr ausgeführt wird. [CWM07]

Torabi et al. fügen für ihre Rekombination einen Offset zu Beginn des CGP ein. Dieser Offset wird durch einen Hyperparameter bestimmt und gibt an, bis zur wievielten Iteration keine Rekombination ausgeführt wird. [TST22]
Diese beiden Ansätze sollen mit einem selbst entwickelten Ansatz verglichen werden. Die neu vorgestellte Rekombinationsrate bezieht sich auf die One-Fifth Regel zur Berechnung der Mutationsrate. Auf Basis dessen ergibt sich die zweite Forschungsfrage: „Hat die Art und Weise der Rekombinationsratenberechnung eine Auswirkung auf die Effektivität des CGPs?“

Die beiden Forschungsfragen sollen anhand unterschiedlicher Experimente beantwortet werden. In Abschnitt 2 werden die theoretischen Grundlagen erläutert, die für das Verständnis des Experimentaufbaus und der Ergebnisse benötigt werden. Im Anschluss werden in Abschnitt 3 die jeweiligen Experimente, sowie die Evaluationsstrategien beschrieben. Darauf folgend werden in Abschnitt 4 die Ergebnisse der Experimente vorgestellt und ausgewertet. Abschließend wird in Abschnitt 5 ein Fazit aus den Ergebnissen zusammengefasst, sowie ein Ausblick auf weitere mögliche Forschungsfragen gegeben.

2 Grundlagen

Für den praktischen Teil dieser Arbeit in Abschnitt 3 werden mit Hilfe von CGP unterschiedliche Testprobleme gelöst. Für das Verständnis dieses Teils werden einige Grundkenntnisse vorausgesetzt, welche in diesem Abschnitt näher betrachtet werden.
CGP ist eine Art von GP, die Lösungen von Rechenproblemen als Graphen darstellt. [Mil20] Dabei wird der Maschine allerdings nicht beigebracht, wie diese Probleme zu lösen sind. Stattdessen lernt sie eigenständig über mehrere Iterationen hinweg das Ausgangsproblem zu lösen. Dies geschieht angelehnt an die Darwin'sche Theorie der biologischen Evolution. [Ahv+19]
Um den grundlegenden Aufbau von CGP zu erläutern, wird die folgende Abbildung 2.1 verwendet:

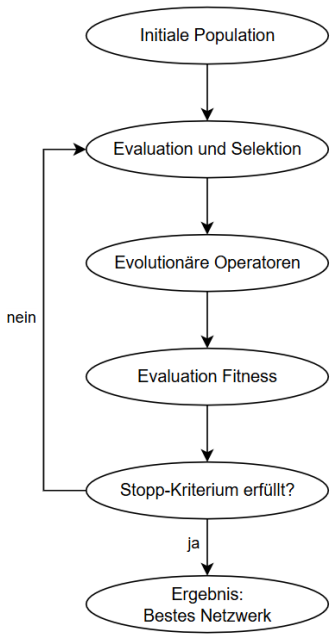
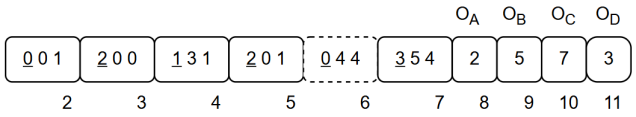


Abbildung 2.1: Aufbau Standard-CGP, angelehnt an [TST22]

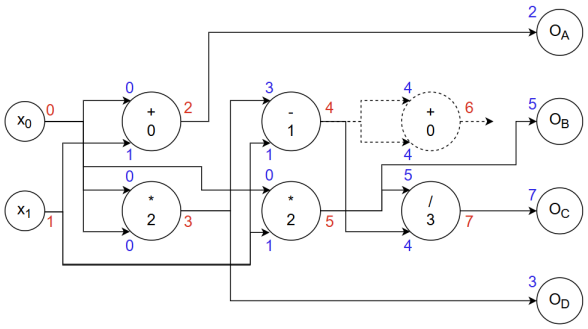
Die Abbildung 2.1 beschreibt den zyklischen Aufbau von Standard-CGP, durch den das System die beste Lösung eines Problems findet. Die folgenden Unterkapitel dieses Abschnitts erläutern diesen Ablauf näher und beziehen sich dabei weitestgehend auf die Knoten des Graphen aus der Abbildung.

2.1 Aufbau Standard-CGP und initiale Population

Dieser Abschnitt bezieht sich auf die Paper [Mil20], [TST22] und [Ahv+19]. Außerdem werden Erkenntnisse aus dem Quellcode von Cui verwendet [Cui24b].
Die *Population* von CGP ist eine Menge von *Chromosomen*, auch Individuen genannt. Chromosomen sind individuelle Möglichkeiten ein komplexes Ausgangsproblem zu lösen. Jedes Chromosom ist in CGP ein gerichteter, azyklischer Graph, bestehend aus *Eingangsknoten*, *Rechenknoten* und *Ausgangsknoten*. Dabei gibt es zwei Darstellungsmöglichkeiten für ein Chromosom. Anhand der folgenden Abbildung 2.2 kann beispielhaft dargestellt werden, wie die Chromosomen in CGP aufgebaut sind und funktionieren.



(a) Beispiel Genotyp



(b) Beispiel Phänotyp

Abbildung 2.2: Darstellungsmöglichkeiten eines Chromosoms, angelehnt an [TST22]

Anhand Abbildung 2.2b lässt sich die klassische Struktur von CGP erkennen: Es handelt sich um einen gerichteten, azyklischen Graphen mit Ein- und Ausgängen. In diesem Beispiel werden die beiden Systemeingänge x_0 und x_1 vorausgesetzt, welche durch Eingangsknoten zur Verfügung gestellt werden. Aus diesen Eingaben sollen anschließend durch verschiedene Berechnungen die Ausgänge resultieren. In diesem Fall werden vier Systemausgänge berechnet, die durch die Ausgangsknoten O_A – O_D dargestellt werden. Zwischen den Ein- und Ausgangsknoten liegen die Rechenknoten. Diese werden verwendet, um verschiedene Rechenoperationen an den Eingangsknoten auszuführen, bis schließlich die Inhalte der Ausgangsknoten als Ergebnis resultieren. Die Rechenoperationen, die von den Rechenknoten ausgeführt werden, werden je nach Anwendungsfall definiert und codiert. In diesem Beispiel ergibt sich folgende Kodierung:

Rechenoperation	**Kodierung**
+	0
-	1
*	2
/	3

Tabelle 2.1: Kodierung der Rechenoperationen

Diese Kodierungen werden in Abbildung 2.2b innerhalb der Knoten angegeben und stellen jeweils die erste (unterstrichene) Zahl innerhalb des Arrays in Abbildung 2.2a dar. Die umrandeten Teile des gesamten Arrays stellen jeweils einen Knoten des Graphen dar und werden folgend Array-Blöcke genannt. Unterhalb dieser Array-Blöcke wird jeweils der Index des zugehörigen Knotens angegeben. Im CGP-Graph wird dieser Index am Ausgang des jeweiligen Knotens (rot) angezeigt.
Des Weiteren werden in Abbildung 2.2a die Eingangskanten für jeden Knoten angegeben. In diesem Beispiel hat jeder Rechenknoten zwei Eingänge und jeder Ausgangsknoten hat jeweils nur einen Eingang, in dem das Ergebnis eines Rechenknoten weitergeleitet und ausgegeben wird. Die Knoteneingänge sind in Abbildung 2.2b blau markiert. Hier werden die Indices derjenigen Knoten angegeben, deren Ausgangswerte verwendet werden. Diese spiegeln sich ebenfalls in Abbildung 2.2a wider: hier werden die Knoteneingänge innerhalb der Array-Blöcke als nicht-unterstrichene Indices angegeben. Damit ergibt sich eine vollständige Beschreibung eines Knotens, indem einem Index Eingänge und gegebenenfalls eine Rechenoperation zugeschrieben werden.
Durch dieses Beispiel wird ebenfalls ersichtlich, was die Eingangsknoten eines Chromosoms ausmacht: Sie geben die Systemeingänge wieder, ohne diese auf irgendeine Weise

zu verarbeiten. Aus diesem Grund müssen die Eingangsknoten nicht im Genotyp aufgezeigt werden, um sie vollständig zu beschreiben, denn sie weisen weder Eingänge noch Rechenoperationen auf, die beschrieben werden müssten.

Die beiden Darstellungsmöglichkeiten eines Chromosoms werden wie in Abbildung 2.2 *Genotyp* und *Phänotyp* genannt. Der Phänotyp resultiert dabei aus dem Genotyp. Der Genotyp ist das volle Individuum, während der Phänotyp die Dekodierung dessen ist. Die Lösung des CGPs ist dementsprechend ein Phänotyp, welcher die Dekodierung des besten Individuums des CGPs darstellt. Die Genotyp- und Phänotyp-Räume können sich dabei stark voneinander unterscheiden. [ES15] Dies lässt sich dadurch erklären, dass nicht alle Teile des Genotyps für die Berechnung des Endergebnisses verwendet werden. Diejenigen Anteile, die für die Berechnung der Lösung nicht gebraucht werden, werden *inaktive Knoten* genannt. So ein inaktiver Knoten wird in Abbildung 2.2a durch gestrichelte Linien dargestellt. Da der Phänotyp die Lösung des CGPs ist, müssen inaktive Knoten auch nicht im Phänotyp dargestellt werden, denn diese finden für die Berechnung keine Verwendung. Der Vollständigkeit halber wird der inaktive Knoten aus Abbildung 2.2a auch in Abbildung 2.2b aufgenommen. So können also verschiedene Genotypen zum gleichen Phänotyp führen, indem sich verschiedene Genotypen nur anhand ihrer inaktiven Knoten unterscheiden. Da der Phänotyp auch den auszuführenden Programmcode darstellt, können auch verschiedene Genotypen zum gleichen Ergebnis führen. Dies kann die Weiterentwicklung der Chromosomen stören, indem beispielsweise nur inaktive Knoten eines Chromosoms angepasst werden. Das kann sich wiederum negativ auf die Trainingsdauer auswirken. Eine nähere Erklärung wird in Abschnitt 2.4 gegeben.
Ob ein Knoten für die Berechnung der Lösung verwendet wird oder nicht, hängt davon ab, ob ein nachfolgender Knoten über eine Kante auf diesen zugreift. Um aktive Knoten von inaktiven Knoten zu unterscheiden, werden demnach zuerst die Ausgangsknoten betrachtet, die offensichtlich für die Auswertung der Ausgabe verwendet werden. Anschließend werden iterativ die Eingangskanten der Rechenknoten zurückverfolgt, bis schließlich die Eingangsknoten des Graphen erreicht werden.

Folgend soll anhand der eingeführten Abbildungen 2.2a und 2.2b ein Beispiel berechnet werden. Angenommen werden die Systemeingänge x_0 = 10 und x_1 = 20. Demnach sind die Ausgänge der beiden Eingangsknoten mit den Indices 0 und 1 gleich den Werten 10 und 20. Der erste Rechenknoten (Index = 2) verwendet die beiden Eingangsknoten (Indices = 0 und 1) als Eingänge und besitzt die Rechenfunktion +. Demnach ist das Ergebnis des Rechenknotens 10 + 20 = 30. Wird dieses Vorgehen für die restlichen Rechenknoten ausgeführt, ergeben sich folgende Ergebnisse:

Knoten	Eingänge	Werte Eingänge	Rechenoperation	Ausgangswert
2	0; 1	10; 20	10 + 20	30
3	0; 0	10; 10	10 * 10	100
4	3; 1	100; 20	100 - 20	80
5	0; 1	10; 20	10 * 20	200
7	5; 4	200; 80	200 / 80	2,5

Tabelle 2.2: Ergebnisse Rechenknoten

Die Ausgangsknoten geben die jeweiligen Ergebnisse der Eingangs- oder Rechenknoten zurück. In diesem Beispiel werden durch das Chromosom folgende Ausgänge aus den beiden Eingängen (10; 20) berechnet:

Ausgangsknoten	Index Eingang	Wert des Ausgangsknotens
O_A	2	30
O_B	5	200
O_C	7	2,5
O_D	3	100

Tabelle 2.3: Ergebnisse Ausgangsknoten

Zusammengefasst hat das Beispielchromosom aus dem Eingangstupel (10; 20) das Ausgangstupel (30; 200; 2,5; 100) berechnet.

Wie bereits erläutert, ist die Population in CGP eine Menge an Chromosomen, also eine Menge an gerichteten, azyklischen Graphen, die aus definierten Systemeingaben Ausgaben berechnen können. Diese Population wird zu Beginn zufällig initialisiert. Dabei werden folgende Angaben vorausgesetzt:

- Größe der Population
- Anzahl der Systemeingänge
- Anzahl der Systemausgänge
- Anzahl der Rechenknoten pro Chromosom

Im Initialisierungsprozess werden für jedes Chromosom pro Knoten zufällige Eingangskanten und gegebenenfalls Rechenoperationen bestimmt. Dabei muss beachtet werden, dass es sich anschließend um einen azyklischen Graphen handeln muss.

Nachdem die initiale Population erstellt wurde, erfolgt der erste *Evaluations-* und *Selektionsschritt*. Der folgende Abschnitt 2.2 erläutert, wie diese Schritte ausgeführt werden.

2.2 Evaluation und Selektion

Der erste Selektionsschritt in CGP startet mit einer zufällig initialisierten Population, wie sie in Abschnitt 2.1 beschrieben wird. In dieser initialen Population ist die Güte mit der die einzelnen Individuen eine Problemlösung darstellen demnach rein zufällig. Das Ziel von CGP ist es, die Güte über Generationen hinweg zu verbessern, bis schließlich eine hinreichend genaue Lösung eines Problems gefunden wird. GP im Allgemeinen richtet sich nach dem darwin'schen Prinzip des Überlebens der Stärkeren. Demnach werden die performantesten Chromosomen verwendet, um die nächste Generation der Population zu erzeugen. [Koz95]

Um zu bestimmen, welche Individuen die Problemlösung am besten beschreiben, muss eine numerische Bewertung erfolgen. Diese wird durch die *Fitness* der einzelnen Chromosomen bestimmt. [Koz95] Wie der Fitnesswert berechnet wird, hängt von den zu lösenden Problemen ab. Beispielsweise verwenden Cui et al. für symbolische Regressionsprobleme den mittleren absoluten Fehler zwischen korrekter und tatsächlicher Lösung für einen Evaluationsdatensatz. [CHH24]

Koza beschreibt in seinem Paper, dass für die Selektion der Eltern der nächsten Generation, jedem Chromosom ein Wahrscheinlichkeitswert zugewiesen wird. Dieser hängt von dessen Fitness ab. Anschließend wird eine definierte Anzahl an Eltern selektiert, wobei die Wahrscheinlichkeitswerte dafür sorgen, dass Individuen, die eine bessere Fitness aufweisen, eine größere Chance haben, selektiert zu werden. Selektion heißt dabei, dass das Chromosom unverändert in die nächste Generation kopiert wird. [Koz95]
Dies ist ein Weg dafür zu sorgen, dass das Prinzip nach Darwin eingehalten wird und somit die fitteren Chromosomen „überleben". Eine andere Möglichkeit, dies zu erreichen, ist die Verwendung von sogenannten *Elitisten*. Die Individuen mit dem besten Fitnesswert einer Generation werden dabei als Elitisten erwählt und werden für die darauf folgende Generation selektiert. [Kra+13] Für die Auswahl der Elitisten wird in dieser Arbeit der *neutral search Algorithmus* verwendet. Dieser wird relevant, falls innerhalb einer Generation ein Elter-Chromosom und ein Kind-Chromosom die gleiche Fitness aufweisen. In diesem Fall

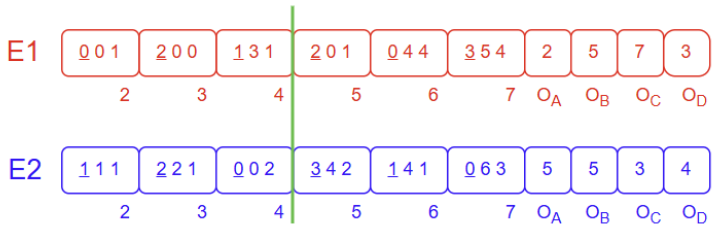
wird stets das Kind-Chromosom als Elitist ausgewählt. Dies führt dazu, dass anschließend bessere Nachkommen erzeugt werden können. [CMH22]
Mit dem in dieser Arbeit verwendeten Selektionsverfahren namens *(μ + λ)-Evolution Strategy (ES)* wird dieser Ansatz verfolgt. Dabei werden jeweils μ-viele Eltern für die folgende Generation selektiert. Im Standard-CGP wird μ mit 1 belegt. Die restlichen Individuen der Population (λ-viele) werden anschließend durch *Mutation* aus dem Elter-Chromosom gebildet. [SB18] Da für den *Rekombinationsschritt* jeweils zwei Eltern-Chromosomen gekreuzt werden, muss für μ ein Wert größer als 1 gewählt werden, falls Rekombination ausgeführt wird. Um eine bessere Vergleichbarkeit zu gewährleisten, werden im praktischen Teil auch unterschiedliche μ-Werte gewählt, selbst wenn keine Rekombination ausgeführt wird.
Die folgenden Abschnitte 2.3 und 2.4 erläutern die Funktionsweise von Rekombination und Mutation.

2.3 Evolutionärer Operator: Rekombination

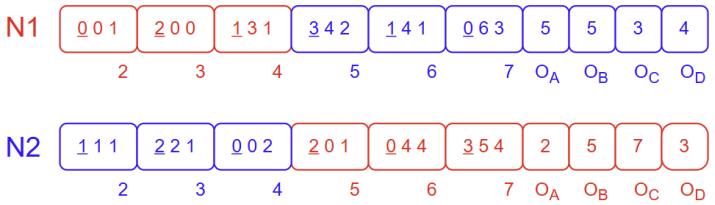
Nachdem die Selektion der Eltern-Chromosomen erfolgt ist, wie in Abschnitt 2.2 beschrieben, können im nächsten Schritt die evolutionären Operationen ausgeführt werden, die die Nachkommen aus den Eltern erzeugen.
Der erste evolutionäre Operator, der verwendet wird, ist die Rekombination. Dabei werden jeweils zwei zufällig ausgewählte Eltern-Chromosomen kombiniert und somit zwei neue (Nachkommen-)Chromosomen erzeugt [Kal20]. Der Rekombinationsschritt wird so oft ausgeführt, bis die Population dieser Generation gefüllt ist.
Innerhalb dieser Arbeit werden unterschiedliche Rekombinationsalgorithmen für die Nachwuchserzeugung miteinander verglichen. Diese werden in den folgenden Absätzen 2.3.1 bis 2.3.4 näher erläutert.

2.3.1 One-Point Rekombination

Das erste Standard-Rekombinationsverfahren, das in dieser Arbeit verwendet wird, ist die *One-Point Rekombination*. Für die Grundlagen aus diesem Abschnitt wurde [PG17] als Quelle verwendet.
Der One-Point Rekombinationsalgorithmus soll anhand folgender Beispielabbildung 2.3 erläutert werden:



(a) Eltern-Chromosomen One-Point Rekombination



(b) Nachwuchs-Chromosomen One-Point Rekombination

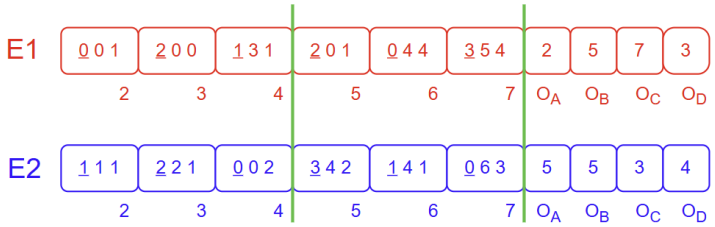
Abbildung 2.3: One-Point Rekombination, angelehnt an [TST22]

Das erste Eltern-Chromosom (E1, rot) in diesem Beispiel wurde aus Abbildung 2.2a entnommen. Das zweite Eltern-Chromosom (E2, blau) ist ein zufällig gewähltes Beispielchromosom.
Bei beiden Chromosomen wird vorerst nicht weiter betrachtet, ob die Knoten aktiv oder inaktiv sind, da sich dieses Merkmal mit der Rekombination und Mutation ändern kann. Angenommen wird, dass E1 und E2 aus der letzten Generation übernommen wurden, da sie die beste Fitness aufgewiesen haben.
Für die One-Point Rekombination wird zuerst eine zufällige Stelle innerhalb der Eltern-Chromosomen bestimmt. Diese ist in Abbildung 2.3a grün markiert. Die Eltern-Chromosomen werden anschließend an dieser Stelle geteilt und überkreuzt zusammengesetzt. Für dieses Beispiel ergeben sich die beiden Nachwuchs-Chromosomen aus Abbildung 2.3b.

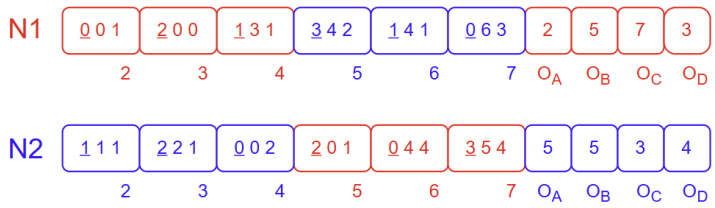
2.3.2 Two-Point Rekombination

Für die Grundlagen dieses Abschnitts wurde [PG17] als Quelle verwendet.
Das Verfahren der *Two-Point Rekombination* funktioniert beinahe identisch zu der in Ab-

schnitt 2.3.1 erläuterten One-Point Rekombination. Der Unterschied besteht darin, dass zwei zufällige Stellen ausgewählt werden, an denen die Chromosomen geteilt werden, anstatt einer.
Die folgende Abbildung 2.4 zeigt dieses Vorgehen anhand eines Beispiels.



(a) Eltern-Chromosomen Two-Point Rekombination



(b) Nachwuchs-Chromosomen Two-Point Rekombination

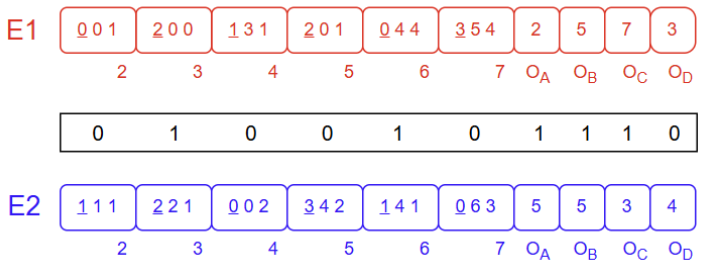
Abbildung 2.4: Two-Point Rekombination, angelehnt an [TST22]

Zu beobachten ist, dass die Eltern-Chromosomen an jeweils zwei Stellen aufgeteilt werden (grün). Die Nachwuchs-Chromosomen bilden sich anschließend abwechselnd aus den Teilstücken der beiden Eltern-Chromosomen. In der Abbildung 2.4 wird dieser Prozess deutlich, indem die Chromosomenteile des ersten Elternteils rot und des zweiten blau markiert sind.

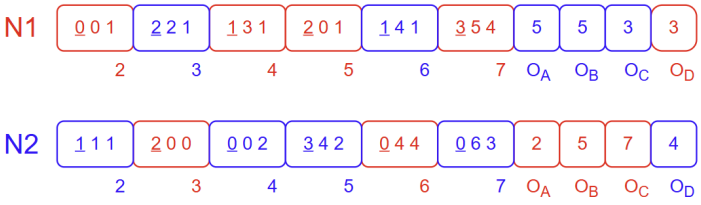
2.3.3 Uniform Rekombination

Die Erläuterung der *Uniform Rekombination* basiert auf [Sys89].
Der größte Unterschied zwischen Uniform Rekombination und One-Point / Two-Point

Rekombination ist die Verwendung einer Maske. Die Maske ist genauso lang wie die Eltern-Chromosomen selbst und beinhaltet für jede Stelle binäre Werte. Diese Werte geben jeweils an, von welchem Elternteil die jeweilige Stelle im Chromosom des Nachwuchses stammen soll. Der zweite gebildete Nachwuchs bekommt in diesem Prozess das Gen des jeweils anderen Elternteils.
Anhand der Abbildung 2.5 lässt sich die Uniform Rekombination beispielhaft erklären:



(a) Eltern-Chromosomen Uniform Rekombination



(b) Nachwuchs-Chromosomen Uniform Rekombination

Abbildung 2.5: Uniform Rekombination, angelehnt an [TST22]

Abbildung 2.5a zeigt die Ausgangssituation mit den beiden Eltern-Chromosomen (E1 und E2) in rot und blau. Zwischen den beiden Eltern-Chromosomen wird schwarz die Maske angezeigt. Diese besteht aus einem zufälligen Binärvektor mit der Länge der beiden Eltern-Chromosomen.
Im folgenden Schritt werden die Nachwuchs-Chromosomen unter Anbetracht der Maske erzeugt. Der entsprechende Index des Knotens kann jeweils unterhalb der Eltern-Chromosomen abgelesen werden. Für den ersten Wert der Maske ergibt das den Knotenindex 2. In diesem Beispiel enthält der erste Wert der Maske eine 0. Dementsprechend wird für den Knoten mit dem Index 2 des ersten Nachwuchs-Chromosoms (N1) der jeweilige Knoten

des ersten Eltern-Chromosoms (E1, rot) verwendet. Das zweite Nachwuchs-Chromosom (N2) bekommt demnach den entsprechenden Knoten aus dem zweiten Eltern-Chromosom (E2, blau).
Für den darauffolgenden Index hat die Maske den Wert 1. Dies bedeutet, dass die Vererbungen umgekehrt erfolgt: N1 bekommt den Knoten von E2 und N2 bekommt den Knoten von E1. Dieser Prozess wird für alle Indices ausgeführt. Die Abbildung 2.5b zeigt die resultierenden Ergebnisse für dieses Beispiel der Uniform Rekombination.

2.3.4 Rekombinationsraten

In den letzten Abschnitten wurden unterschiedliche Rekombinationsalgorithmen erläutert. Die Rekombination wird allerdings nicht für die Erzeugung jedes Nachwuchs-Chromosoms verwendet. Sie wird nur zu einer bestimmten Wahrscheinlichkeit ausgeführt, welche mit der *Rekombinationsrate* beschrieben wird. Die Effektivität des (C)GP-Systems hängt unter anderem von der richtigen Wahl der Rekombinationsrate ab. [Has+19]
Um das bestmögliche (C)GP-System zu erhalten, sollte das richtige Verhältnis aus *Exploration* (dt. Erforschung) und *Exploitation* (dt. Ausbeutung) des Lösungsraums erzielt werden. Exploration ist dabei der Prozess, neue Bereiche des Lösungsraums zu erkunden, während bei der Exploitation bereits erkundete Regionen des Lösungsraums näher betrachtet werden. Eine Stellschraube, um den Prozess dahingehend zu steuern, ist die Rekombinationsrate. [ČLM13] Wird die Rekombinationsrate sehr hoch eingestellt, wird zu einem hohen Maß Exploration betrieben. Dies führt allerdings dazu, dass die Exploitation gering ausfällt und somit die optimalen Lösungen verfehlt werden können. [PG17]
In unterschiedlichen Papern werden verschiedene Herangehensweisen vorgeschlagen, diesen Parameter zu wählen. Teilweise widersprechen sich diese Aussagen, wie in den folgenden Abschnitten näher dargestellt wird. Außerdem wird ein Einblick über die unterschiedlichen Möglichkeiten gegeben, die Rekombinationsrate zu setzen.

Konstante Rekombinationsrate

Wie von Hassanat et al. beschrieben, ist die konstante (statische) Rekombinationsrate die gängigste Variante. Als geläufiges Beispiel wird in ihrem Paper der Wert 0,9 für die Rekombinationsrate vergeben. [Has+19] Dies bedeutet, dass zur Initialisierung des CGP ein fester Wert für die Rekombinationsrate gewählt wird. Dieser gilt für alle Generationen gleichermaßen und wird nicht verändert.
Diese „klassische“ Form der Rekombination kann in der Evaluation der praktischen Tests

dazu verwendet werden, um die erste Forschungsfrage zu beantworten. Durch diese einfachste Form der Rekombinationsrate kann überprüft werden, ob CGPs ohne oder mit (klassischer) Rekombination effizienter sind.
Mit Hilfe der in den nächsten Abschnitten beschriebenen Anpassungen der Rekombinationsrate kann anschließend überprüft werden, ob die Effizienz von CGP mit Rekombination weiter verbessert werden kann.

Linear fallende Rekombinationsrate

Clegg et al. präsentieren in ihrem Paper aus 2007 eine neue Form der Rekombination. Dabei treffen sie auch einige Aussagen über die Rekombinationsrate, die in diesem Abschnitt näher betrachtet werden sollen. [CWM07]
Sie beobachten, dass sich für höhere Rekombinationsraten eine schnellere Konvergenz der Fitness innerhalb der ersten Generationen einstellt. Gleichzeitig stellen sie fest, dass in ihrem Beispiel Rekombination ab der 200. Generation keinen signifikanten Vorteil in der Performance liefert. Insgesamt betrachten sie 500 Iterationen.
Aus diesen beiden Beobachtungen konstruieren sie eine neue, dynamische Rekombinationsrate. Diese beginnt bei einem relativ hohen Startwert von 0,9 und sinkt linear, bis eine Rekombinationsrate von 0,0 erreicht wird. Anschließend wird keine Rekombination mehr ausgeführt. In ihrem Beispiel legen die Autoren über händische Analysen eine Generation fest, bis zu welcher die Rekombinationsrate auf 0,0 fallen soll.
Der Variable Parameter, der in der Hyperparameteranalyse optimiert werden soll, wird in dieser Arbeit „Delta-Rekombinationsrate“ genannt. Er gibt an mit welcher Schrittgröße die Rekombinationsrate pro Iteration reduziert wird. Innerhalb des praktischen Teils dieser Arbeit wird 0,9 für den Startwert der Rekombinationsrate übernommen, um die Anzahl der Parameter in der Hyperparameteranalyse zu reduzieren. Ein weiterer Vorteil davon nur einen variablen Parameter pro Rekombinationsraten-Art zu verwenden, ist es, dass die Änderungen innerhalb der Ergebnisse für die verschiedenen Parameter besser miteinander verglichen werden können.

One-Fifth-Regel angewandt auf die Rekombinationsrate

Die *One-Fifth-Regel* gibt es bereits für andere Parameter in GP, wie beispielsweise für die *Mutationsrate*, die in Abschnitt 2.4 näher erläutert wird. Diese Regel wird im Paper von

Milano und Nolfi verwendet. Dabei handelt es sich um einen Weg, die Mutationsrate automatisch und dynamisch an die Problemcharakteristiken und die evolutionäre Phase anzupassen. [MN18]
Betrachtet wird bei dieser Regel das Fitness-Verhältnis der Elitisten und der neuen Chromosomen. In anderen Worten werden die Kinder mit ihren Eltern verglichen. Erzielt werden soll ein Verhältnis von 20%. Das heißt, dass 20% der Nachwuchs-Chromosomen eine bessere Fitness aufweisen sollen als ihre Eltern. Wird dieses Verhältnis unter- oder übertroffen, wird der jeweilige Parameter verkleinert oder vergrößert. [DDL19]

In dieser Arbeit soll die One-Fifth-Regel für die dynamische Anpassung der Rekombinationsrate herangezogen werden. Um nur den Rekombinationsschritt ohne Einfluss der anschließenden Mutation zu bewerten, muss die Bewertung der Fitness vor der Mutation geschehen. Dementsprechend wird für die One-Fifth-Regel in dieser Arbeit direkt nach dem Rekombinationsschritt die Fitness der Eltern- mit der Fitness der Nachwuchs-Chromosomen verglichen. Anschließend wird betrachtet, ob 20% der Kinder eine bessere Fitness aufweisen als ihre Eltern. Wird dieser Wert übertroffen, wird die Rekombinationsrate mit 1,1 multipliziert, um den Erfolg des Rekombinationsschritts auszuschöpfen. Andernfalls wird die Rekombinationsrate mit 0,9 multipliziert und somit verringert.

Rekombinationsrate mit Offset

Torabi et al. beschreiben in ihrem Paper eine neue Rekombinationsstrategie [TST22]. Dabei verwenden sie einen Hyperparameter, der den *Offset der Rekombination* definieren soll. Das heißt, dass die Rekombination in ihrer Strategie für die ersten Generationen ausgesetzt wird und erst zu einer vorgegeben Iteration wieder einsetzt. Wie dieser Hyperparameter bestimmt wurde und welche Größenordnung dieser einhalten sollte, wird in dem Paper allerdings nicht erwähnt.

Torabi et al. behaupten außerdem, dass ihre Strategie „den richtigen Kompromiss aus Exploration und Exploitation“ erreiche. [TST22] Ein Vergleich dieser Behauptung mit der Aussage von Clegg et al. offenbart einen Widerspruch zwischen diesen Thesen. Während Clegg et al. vor allem in den ersten Generationen auf Rekombination setzen, da der Rekombinationsschritt hier zu einer schnelleren Konvergenz führen soll, meiden Torabi et al. in ihrer Strategie Rekombinationen in den ersten Generationen völlig. [CWM07; TST22]
Ein Ziel dieser Arbeit ist es herauszufinden, ob sich ein Offset für die Rekombination als sinnvoll erweist oder ob sich dieser nur für die Rekombinationsstrategie eignet, die Torabi et al. in ihrem Paper eingeführt haben.

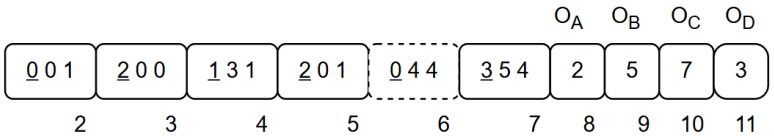
2.4 Evolutionärer Operator: Mutation

Der zweite evolutionäre Operator ist die Mutation. Diese wird direkt nach der Rekombination ausgeführt. Anders als bei der Rekombination werden bei der Mutation nicht zwei, sondern nur ein Chromosom einbezogen und daraus ein neues Chromosom erstellt. In diesem Prozess werden Teile des Genotyps des Chromosoms zufällig verändert, um daraus einen neuen Genotyp zu erzeugen. [Ahv+19]
Für *probabilistische Mutation* wird eine Mutationsrate verwendet, um die Wahrscheinlichkeit anzugeben, mit der ein Gen mutiert wird. Dadurch kann es allerdings dazu kommen, dass das Chromosom vor und nach der Mutation zwar unterschiedliche Genotypen aufweist, sich die aktiven Knoten allerdings nicht voneinander unterscheiden. Dies hat zur Folge, dass sich der Lösungsansatz des CGPs bezüglich des Ausgangsproblems nicht ändert, obwohl bereits eine Mutation ausgeführt wurde. Die Fitness des CGPs kann sich in so einem Fall nicht verbessern. Um dieses Problem in den Griff zu bekommen, wird in dieser Arbeit die *Single (Active) Mutation* herangezogen. In diesem Algorithmus werden zufällige Gene eines Chromosoms verändert, bis ein aktiver Knoten mutiert wurde. Anschließend bricht der Mutationsalgorithmus ab. [Mil20] Dies hat zusätzlich den Vorteil, dass keine Mutationsrate angepasst werden muss, was die Hyperparameteranalyse weniger rechenintensiv macht.

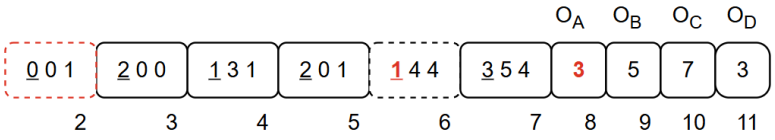
Die Veränderung eines Gens innerhalb der Mutation wird vorgenommen, indem das ausgewählte Zeichen des Genotyps zufällig geändert wird [Koz95]. Dies hat für unterschiedliche Knotenarten des entsprechenden Phänotyps verschiedene Effekte.
Ein Ausgangsknoten hat als relevanten Parameter nur seinen Vorgängerknoten. Wird also ein Ausgangsknoten mutiert, wird die eingehende Kante zufällig neu belegt. Dabei muss wie bei der Initialisierung beachtet werden, dass die azyklische Struktur von CGP erhalten bleibt. In dieser Arbeit werden dementsprechend nur Vorgängerknoten gewählt, deren Index kleiner ist als der mutierte Knoten.
Anders als die Ausgangsknoten wird ein Rechenknoten durch mindestens zwei Parameter definiert: ein Rechenoperator und mindestens eine Eingangskante. Welcher dieser Parameter mutiert werden soll, wird ebenfalls zufällig bestimmt. Wird eine Kante verändert, gelten die gleichen Regeln wie beim Mutieren eines Ausgangsknotens. Wird der Rechenoperator mutiert, wird dieser zufällig neu aus den kodierten Rechenfunktionen gewählt.
Da ein Eingangsknoten nur aus den Dateneingängen besteht und keine Parameter enthält, können diese nicht mutiert werden und werden im Mutationsschritt nicht betrachtet.

Um die Single Active Mutation anhand eines Beispiels näher zu erläutern wird folgender Genotyp eines Chromosoms aus Abbildung 2.2a erneut eingeführt. Es wird angenom-

men, dass dieses Chromosom durch vorherige Rekombination entstanden ist und somit im nächsten Schritt mutiert werden soll. Die folgende Abbildung 2.6 zeigt dieses Mutationsbeispiel:



(a) Genotyp vor Mutation



(b) Genotyp nach Mutation

Abbildung 2.6: Single Active Mutation, angelehnt an [TST22]

Zu Beginn des Algorithmus wird ein zufälliger Index für die Mutation ausgewählt. Angenommen dieser Index hat den Wert 6. Dabei handelt es sich um einen Rechenknoten. Da in diesem Beispiel die Rechenknoten jeweils 3 Parameter aufweisen, wird zufällig einer der drei Parameter für die Mutation bestimmt. In diesem Beispiel wird der erste Parameter verändert, also der Rechenoperator. Wie bereits in Tabelle 2.1 aufgezeigt, werden die Rechenoperationen durch 0 bis 3 kodiert. Sei beispielsweise die zufällig gewählte Zahl für den Rechenoperator gleich 1. Dieser neue Wert wird in Abbildung 2.6b rot markiert. Da der in diesem Schritt mutierte Knoten inaktiv ist, muss der Single Active Mutation zufolge erneut mutiert werden.
Angenommen der zufällig gewählte Index ist 8, was dem Index von O_A entspricht. Da es sich, um einen Ausgangsknoten handelt, muss ein neuer Vorgängerknoten zufällig gewählt werden. Beispielsweise wird nun dieser Parameter mit dem Index 3 belegt. Er wird in Abbildung 2.6b ebenfalls rot dargestellt. Da dieser Knoten Teil der aktiven Knoten ist, terminiert hier der Mutationsalgorithmus. Die Abbildung 2.6b zeigt das Ergebnis des Mutationsbeispiels (mit Kennzeichnung der neuen aktiven / inaktiven Knoten). Der neu entstandene inaktive Knoten wird in dieser Abbildung rot hervorgehoben.

2.5 Evaluation Fitness, Stopp-Kriterium und Ergebnis

Nachdem der Mutationsschritt für alle Chromosomen der Population ausgeführt wurde, findet erneut ein Evaluationsschritt statt. Dabei wird für jedes Chromosom ein Fitness-Wert bestimmt, der bewertet wie exakt die Trainingsdaten durch das Chromosom beschrieben werden. [Ahv+19] Die Berechnung der Fitness hängt wie in Abschnitt 2.2 erläutert von dem zu lösenden Ausgangsproblem ab.
Sobald für alle Chromosomen einer Population ein Fitness-Wert bestimmt wurde, werden diese auf das *Stopp-Kriterium* geprüft. Dieses gibt an, ab welcher Bedingung ein Algorithmus als konvergiert gilt. Im Beispiel von Cui et al. ist diese Bedingung bei symbolischen Regressionsbenchmarks für $Fitness < 0,01$ erfüllt. [CHH24]
Wird das Stopp-Kriterium erfüllt, wird der CGP-Algorithmus abgebrochen. Das Ergebnis ist das beste Chromosom der letzten Generation. Dieses löst das Ausgangsproblem hinreichend gut. Andernfalls wird eine weitere Iteration des CGP-Algorithmus gestartet. Dabei werden erneut Selektion, Rekombination und Mutation ausgeführt, um eine bessere Lösung des Ausgangsproblems zu finden.

2.6 Bayes'sche Analyse

Für die Evaluation der Ergebnisse werden in diesem Abschnitt statistische Grundlagen zur *Bayes'schen Analyse* erläutert.
Der Satz von Bayes (Bayes'sches Therorem) bietet Grundlagen zur Berechnung der bedingten Wahrscheinlichkeit. Zu Beginn wird eine Anfangshypothese über das Ergebnis eines Wahrscheinlichkeitsproblems angenommen. Mit zusätzlichen Informationen wird die Anfangshypothese schließlich korrigiert. Das Ergebnis ist eine neue Wahrscheinlichkeit, die alle vorhandenen Informationen einschließt. [Pey20]

Vereinfacht dargestellt funktioniert die Anwendung des Satzes von Bayes in der Bayes'schen Analyse wie in folgender Abbildung 2.7.

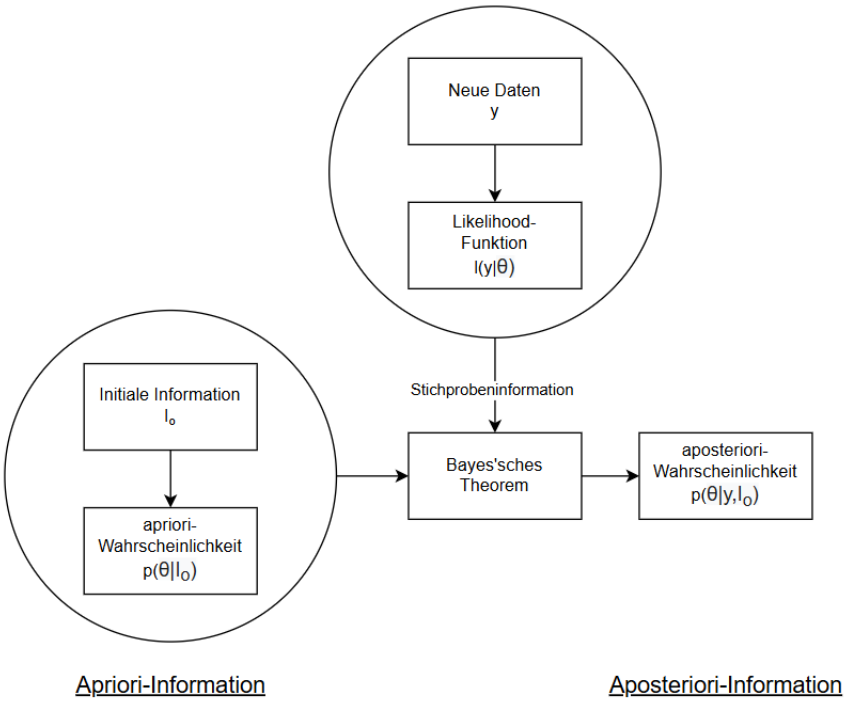


Abbildung 2.7: Bayes'sche Analyse, angelehnt an [Nen80]

Zu Beginn wird die Realität als Modell mit bestimmten Annahmen (θ) vereinfacht. Diese ursprünglichen Annahmen werden als Wahrscheinlichkeitsverteilung (*apriori-Wahrscheinlichkeit*) dargestellt. Aus den neuen Daten y kann die sogenannte *Likelihood-Funktion* berechnet werden, die eine Dichtefunktion für die Beobachtung y darstellt unter der Bedingung, dass θ zutrifft. Mit Hilfe des Bayes'schen Theorems können die apriori-Dichtefunktion und die Stichprobeninformationen miteinander verknüpft werden. Das Ergebnis ist eine neue Wahrscheinlichkeitsverteilung für θ (*aposteriori-Wahrscheinlichkeit*). [Nen80] Mit diesem Vorgehen kann demnach mit Hilfe von empirischen Daten eine Wahrscheinlichkeitsverteilung für ein Modell gefittet werden.

In dieser Arbeit werden basierend auf [CMH23] zwei Modelle verwendet: das *Plackett-Luce-Modell* und das *Gammaverteilung-basierte Modell.*
Mit Hilfe des Plackett-Luce-Modells können die Wahrscheinlichkeiten berechnet werden, mit denen die jeweiligen CGP-Konfigurationen besser als alle anderen Konfigurationen sind. Demnach kann eine Reihenfolge bestimmt werden, die die Effizienz der Konfigura-

tionen sortiert angibt. Das verwendete Modell basiert auf dem Plackett-Luce-Modell, das von Calvo et al. eingeführt wird. [CMH23; CCL18]
Für die Auswertung mit Hilfe des Gammaverteilung-basierten Modells wird der Code von Pätzel verwendet [Pät24]. Dieser ermöglicht es nicht-negative Daten zu vergleichen und daraus die Wahrscheinlichkeitsverteilung von μ_{config} zu schätzen. μ_{config} ist dabei eine zufällige Variable, die beispielsweise dem jeweiligen Mittelwert der Iterationenzahl entspricht, die gebraucht werden, um eine CGP-Lösung konvergieren zu lassen. Für jede CPG-Konfiguration kann so ein 95% HPDI (*highest posterior density interval*) von μ_{config} bestimmt werden. Dies bedeutet, dass innerhalb dieses Intervalls 95% der Ergebnisse liegen. [CMH23] Dadurch können die Streuungen der Iterationszahlen für jede CGP-Konfiguration miteinander verglichen werden, ohne von einer gleichverteilten Dichtefunktion ausgehen zu müssen.

Pätzel verwendet in seinem Code das *Markov Chain Monte-Carlo (MCMC) Sampling* zur Berechnung der Verteilungen [Pät24; CMH23]. Bei MCMC werden mehrere tausend Kombinationen von Parameterwerten ($\mu_1, \sigma_1, \mu_2, \sigma_2$, v) generiert, die jeweils zwei verschiedene Gamma-Wahrscheinlichkeitsverteilungen darstellen. Jede Wertekombination ist repräsentativ für glaubwürdige Parameterwerte, die gleichzeitig die eingehenden Daten und die vorherigen Verteilungen berücksichtigen. Diese Parameterwerte werden anschließend zusammengefasst. [Kru12]

3 Experimente und Evaluation

3.1 Aufbau der Experimente

Für die Beantwortung der Forschungsfragen werden unterschiedliche Experimente ausgeführt, deren Ergebnisse in dieser Arbeit ausgewertet und interpretiert werden sollen. Der Programmcode für die Experimente wurde in der Sprache Julia verfasst. Dabei fand eine Orientierung an folgendem Code von Cui statt: [Cui24b]
Für eine umfangreichere Bewertung wurden mehrere Benchmark-Testszenarien aus unterschiedlichen Domänen überprüft. Diese werden im folgenden Abschnitt 3.1.1 näher betrachtet.

3.1.1 Testszenarien

Boolesche Probleme

Nach der Aussage von Kalkreuth et al. spielen *Boolesche Probleme* eine wichtige Rolle in der Forschung zu GP. Grundsätzlich ist bei Booleschen Problemen das Ziel einen sinnvollen Zusammenhang zwischen Ein- und Ausgaben zu generieren, welcher von Booleschen Funktionen bestimmt wird. Diese können wiederum durch Boolesche Ausdrücke mathematisch beschrieben werden. Die verschiedenen Booleschen Funktionen können durch Wahrheitstabellen dargestellt werden, in denen die jeweiligen Ein- und Ausgaben miteinander verknüpft werden. [Kal+23] Das Ziel von CGP-Algorithmen ist es, aus Eingängen die richtigen Ausgänge zu generieren, welche dem Mapping der Booleschen Funktionen entsprechen.

Insgesamt werden in dieser Arbeit zwei Boolesche Benchmarkprobleme für die Evaluation der CGP-Algorithmen betrachtet: 3-bit Parity und 16-4-bit Encode (vereinfacht bezeichnet als Parity und Encode).
Obwohl Parity als zu leichtes Ausgangsproblem für GP bezeichnet wird [Whi+13], wird es häufig als Benchmarkproblem genutzt [YM01; KK17; KK20]. Um Benchmarkprobleme

mit unterschiedlichen Ein- und Ausgangsgrößen miteinzubeziehen, wird Encode verwendet.

Das verwendete Standardfunktionsset aller Testszenarien beinhaltet die Booleschen Rechenoperatoren AND, OR, NAND und NOR. Außerdem wird die Standardfitnessfunktion für Boolesche Benchmarkprobleme verwendet. Diese wird durch den Anteil an korrekt zugeordneten Bits definiert. [CMH23] Das Stopp-Kriterium ist erfüllt, wenn die Fitness den Wert 0 erreicht. Folgend werden die beiden verwendeten Booleschen Benchmarkprobleme näher erläutert:

Parity: N-bit Parity ist eine Mapping-Funktion, die angibt, ob die Summe der Komponenten eines Binär-Vektors gerade oder ungerade ist. Bei 3-bit Parity handelt es sich dabei um Binär-Vektoren der Länge 3. Das Evaluationsset besteht aus $2^N = 2^3$ Testvektoren. [HLS99] Demnach gibt es für das CGP drei Eingaben (die Komponenten eines Binär-Vektors) und eine Ausgabe (Binärwert für „gerade“ / „ungerade“).

Encode: Beim 16-4-bit Encoding wird aus einer 16-stelligen One-Hot-Kodierung ein 4-bit Integer erstellt. Die One-Hot-Kodierung besteht dabei aus einem 16-stelligen Binär-Vektor, wobei nur eine Stelle mit einer 1 belegt ist, die restlichen Stellen sind 0. Ziel ist es diejenige Stelle zu finden, die die 1 hält und diese als üblichen 4-bit Integer zu kodieren. [Cui+23; GP15] Daraus ergibt sich eine Eingabegröße von 16 und eine Ausgabegröße von 4. Der Testdatensatz enthält 16 verschiedene One-Hot-Kodierungen, die umgewandelt werden sollen.

Symbolische Regression

Symbolische Regression (SR) zählt seit Beginn von GP als Grundlage für methodologische Forschung und als primäres Anwendungsgebiet [OLM18]. Das Ziel von SR ist es eine Beziehung zwischen Ein- und Ausgängen aus einem gegebenen Datensatz zu erlernen. Diese Beziehung beruht auf interpretierbaren mathematischen Ausdrücken. Der Fehler von errechnetem und vorgegebenem Ausgang pro Eingang soll dabei minimiert werden. [MC24]

Die beiden in dieser Arbeit verwendeten SR Probleme werden aus dem Paper von Cui et al. übernommen: Keijzer-6 und Koza-3. Im folgenden werden diese durch Keijzer und Koza abgekürzt. Sie sind von der GP-Community empfohlen und wurden bereits in früheren Arbeiten verwendet [Whi+13; Kal20].
Der verwendete Funktionssatz besteht aus den folgenden acht mathematischen Funktionen: Addition, Subtraktion, Multiplikation, Division, Sinus, Cosinus, natürlicher Logarithmus und Exponentialfunktion. Bei der Division wird sichergestellt, dass eine Division

durch null abgefangen wird. [CHH24] Dabei wird der Wert 1,0 ausgegeben, statt eine Division durch null auszuführen. Zur Absicherung des natürlichen Logarithmus werden für alle Eingaben nur die absoluten Werte in die Berechnung einbezogen. Für den Fall, dass die Eingabe gleich 0 ist, wird vergleichbar zur Division, der Wert 1,0 zurückgegeben. Da es bei der Programmierung mit Julia zu Fehlern kommen kann, wenn die Eingaben von Sinus oder Cosinus zu groß sind, werden diese Fälle ebenfalls abgefangen. Dabei werden die Eingaben nicht weiter verrechnet sondern durchgereicht.
Für die Berechnung der Fitness wird der mittlere absolute Fehler zwischen vorhergesagter und tatsächlicher Ausgabe pro Eingabe berechnet. Das Stopp-Kriterium ist erfüllt, sobald die Fitness den Wert 0,01 unterschreitet. [CHH24]
Die folgende Tabelle 3.1 beschreibt die verwendeten SR Probleme näher.

Name	**Variablen**	**Gleichung**	**Trainingsdaten**	**Testdaten**
Keijzer	1	$\sum_{i}^{x} \frac{1}{i}$	E[1, 50, 1]	E[1, 120, 1]
Koza	1	$x^6 - 2 \cdot x^4 + x^2$	U[−1, 1, 20]	-

Tabelle 3.1: Beschreibung SR Benchmarkprobleme nach [CHH24]

Zu beobachten ist, dass in unterschiedlichen Papern verschiedene Keijzer-6 Funktionen beschrieben werden [Oli+18; Li+24; Kom18]. Wie bereits erwähnt werden in dieser Arbeit alle SR Benchmarkprobleme auf das Paper von Cui et al. bezogen [CHH24], weswegen die Keijzer-Funktion aus Tabelle 3.1 verwendet wird.

3.1.2 CGP-Konfigurationen

In dieser Arbeit werden unterschiedliche *CGP-Konfigurationen* miteinander verglichen und evaluiert. Diese enthalten verschiedene Parametrierungen innerhalb eines CGPs und werden in diesem Abschnitt näher erläutert.
Ein Ziel der Arbeit ist es die Effizienz von Rekombination in CGP zu bewerten. Ebenfalls sollen unterschiedliche Rekombinationsalgorithmen und -konfigurationen miteinander verglichen werden, um eine Aussage über deren Effektivität zu treffen. Aus diesem Grund müssen für die unterschiedlichen Testszenarien mehrere Rekombinationskonfigurationen getestet werden. Um eine Vergleichbarkeit der Ergebnisse zu gewährleisten müssen Selektion und Mutation in allen Experimenten den gleichen Algorithmen entsprechen. Diese werden im Folgenden beschrieben.

Selektion: Für die Selektion wird in allen Experimenten die (μ+λ)-ES verwendet. Bei μ und λ handelt es sich dabei um jeweils eigenständige Freiheitsgrade, wobei $\mu <= \lambda$ gelten muss. Für die CGP-Konfiguration ohne Rekombinationsschritt wurde außerdem nicht $\mu = 1$ festgelegt, obwohl dies den Standard-Einstellungen eines CGP ohne Rekombination entspricht. Dieser zusätzliche Freiheitsgrad soll einen ausgewogeneren Vergleich zwischen den CGP-Konfigurationen mit und ohne Rekombinationsschritt ermöglichen. Für die Auswahl der Elitisten wird das neutral search Verfahren herangezogen.

Mutation: In allen Experimenten dieser Arbeit wird die Single Active Mutation angewendet.

Die nachfolgende Tabelle 3.2 gibt alle Konfigurationen für die **Rekombination** an, die für jedes Testszenario aus Abschnitt 3.1.1 getestet werden.

Rekombinationsverfahren	Art der Rekombinationsrate	Offset
One-Point Rekombination	konstante Rekombinationsrate	aktiv
Two-Point Rekombination	linear fallende Rekombinationsrate	inaktiv
Uniform Rekombination	Rekombinationsrate mit One-Fifth Regel	-
keine Rekombination	-	-

Tabelle 3.2: Konfigurationen Rekombination

Zu beachten ist, dass diese einzelnen Einstellungen miteinander kombiniert werden, solange es die Verfahren zulassen. Falls keine Rekombination ausgeführt wird, wird selbstverständlich auch die Rekombinationsrate nicht angepasst und es kann kein Offset eingeführt werden. Grundsätzlich werden die Konfigurationskombinationen nach folgendem Muster erstellt:

- Auswahl Rekombinationsverfahren
- Auswahl Art der Rekombinationsrate
- Auswahl Offset aktiv / inaktiv

Kombinatorisch betrachtet ergeben sich 19 verschiedene Konfigurationen für den Rekombinationsschritt, die miteinander verglichen werden sollen.

3.1.3 Hyperparameteroptimierung

Da CGP mehrere Hyperparameter ausweist, die die Effektivität eines CGP-Modells beeinflussen, muss eine *Hyperparameteroptimierung* ausgeführt werden, die einen optimierten Parametersatz ausgibt. Für jede in Abschnitt 3.1.2 eingeführte CGP-Konfiguration wird eine eigene Hyperparameteroptimierung ausgeführt, da nicht von einem optimierten Datensatz auf den nächsten geschlussfolgert werden kann. Außerdem müssen die Hyperparameter auf das jeweilige Testszenario angepasst werden.
Für die Hyperparameteroptimierungen in dieser Arbeit wird das Julia-Paket HyperOpt.jl verwendet [Car25]. Mit Hilfe des Pakets können die Hyperparameter, sowie deren Wertebereiche angegeben werden, sodass eine automatisierte Optimierung stattfinden kann. Diese werden in folgender Tabelle 3.3 aufgelistet.

Parameter	**min**	**max**	**Schrittweite**
Anzahl Rechenknoten	50	2000	50
μ (Anzahl Elitisten)	2	20	2
λ (Anzahl Nachkommen)	10	60	2
Offset Rekombination (in Iterationen)	0	P:55 / K:300	P:5 / K:30
Konstante Rekombinationsrate	0,1	1,0	0,1
Fallende Rekombinationsrate (Abzug)	0,005	0,05	0,005
One-Fifth Regel Rekombinationsrate (Startwert)	0,3	0,75	0,05

Tabelle 3.3: Optimierte Hyperparameter und deren Wertebereiche

Zu beachten ist, dass $\mu <= \lambda$ gilt und die Hyperparameteranalysen nur für Parity und Keijzer (in Tabelle 3.3 durch P und K markiert) ausgeführt werden. Eine ausführliche Begründung liefert Abschnitt 3.1.4. Außerdem werden die Einträge der Tabelle 3.3 je nach CGP-Konfiguration verwendet. Zum Beispiel wird der Wert „Offset Rekombination“ nur verwendet, wenn der Rekombinationsoffset aktiv ist. Dieser Wert wird in Iterationen angegeben, in denen die Rekombination ausgesetzt wird. Um die obere Grenze des Offsets sinnvoll abzuschätzen wurden zuerst die Hyperparameteranalysen derjeniger CGP-Konfigurationen vorgenommen, die keinen Offset verwenden. Für jeweils ein Testszenario wurde anschließend der Mittelwert der benötigten Iterationen berechnet. Auf Basis dieser Mittelwerte konnten die Offset Wertebereiche sinnvoller gewählt werden, sodass sich der Wert der wahrscheinlich benötigten Iterationen mit der oberen Grenze deckt. Die drei letzten Zeilen der Tabelle 3.3 geben die verschiedenen Arten an Rekombinationsraten an, die in dieser Arbeit verglichen werden. Für jede dieser Ratenarten wird gezielt nur ein Hyperparame-

ter verwendet, um die Rechenzeit der Hyperparameteroptimierung zu reduzieren. Für die linear fallende Rekombinationsrate gibt dieser Parameter an, mit welcher Schrittweite die Rekombinationsrate pro Iteration sinkt. In der One-Fifth Regel wird der Startwert der Rekombinationsrate angegeben, also diejenige Rekombinationsrate, mit der das CGP-Modell initialisiert wird.

In einer Optimierungsschleife werden pro Parametersatz 10 Testdurchläufe durchgeführt, in denen das CGP trainiert wird. Für Boolesche Probleme wird die Effizienz die Parametersätze anhand der Iterationen gemessen, die der CGP-Algorithmus braucht, bis er konvergiert. Bei symbolischer Regression bezieht sich der Vergleich auf die berechnete Fitness. Es wurde eine Iterationsgrenze eingeführt, ab der der CGP-Algorithmus in der Hyperparameteroptimierung abbricht, um Rechenzeit zu sparen. Diese Grenze wurde durch vorhergehende Tests so gesetzt, dass nur wenige Ausreißer diese überschreiten. Trifft dies zu, wird bei Booleschen Problemen die Anzahl der benötigten Iterationen verdoppelt, um so das gescheiterte Training stärker zu bestrafen.
Nach 150 getesteten Parametersätzen bricht die Hyperparameteroptimierung ab und gibt den besten Parametersatz aus.

Das verwendete Julia-Paket bietet verschiedene Sampler an. Der als Standardeinstellung zur Verfügung gestellte Sampler ist ein Random-Sampler. Da Tests des verwendeten BHOB Sampler mit und ohne Hyperband keine Einsparungen in der Rechenzeit ergeben haben, wurde weiterhin der einfach zu konfigurierende Random-Sampler verwendet.

3.1.4 Teststruktur

Ursprünglich sollte für jedes Testproblem/Testszenario und für jede CGP-Konfiguration eine eigene Hyperparameteroptimierung durchgeführt werden. Mit den daraus resultierenden Parametersätzen hätten für jede Kombination aus Testszenario und CGP-Konfiguration eine annähernd optimale Lösung gefunden werden können. Diese hätten anschließend systematisch evaluiert und verglichen werden können. Angesichts der begrenzten Rechenkapazitäten konnten nur für die beiden leichtesten Testszenarien Parity und Keijzer sinnvolle Hyperparameteranalysen ausgeführt werden. Für die jeweiligen anderen Szenarien wurde deswegen eine stark vereinfachte Hyperparameterstudie ausgeführt. Hyperparameter, welche nicht die Rekombination betreffen (beispielsweise Anzahl der Rechenknoten) wurden aus der Arbeit von Cui et al. [Cui24a] herangezogen. So sollten nur die für die Rekombination relevanten Parameter optimiert werden, also die Rekombinationsrate und gegebenenfalls der Offset. Bei den Versuchen dieses Testverfahren für die umfangreicheren

Testszenarien auszuführen, wurde ebenfalls festgestellt, dass die verfügbare Rechenkapazität nicht ausreicht. Diese Beobachtung wurde auch gemacht als der Offset-Parameter bei der Optimierung vollständig ausgeschlossen wurde.
Aus diesen Gründen wurde eine neue Methodik entwickelt, die in diesem Abschnitt näher erläutert werden soll. Die Teststruktur wird in zwei voneinander unabhängige Testblöcke geteilt.

1. Testblock: einfache Testszenarien

Für die einfachen Testszenarien Parity und Keijzer kann eine Hyperparameteranalyse trotz eingeschränkter Rechenzeit ausgeführt werden, indem die Iterationenzahl bis zum Abbruch des CGP-Algorithmus heruntergesetzt werden. Dieser wird schrittweise reduziert, bis die Hyperparameteranalyse ausgeführt werden kann.
Um zu überprüfen, ob die angepassten Iterationsgrenzen für Parity sinnvoll sind, werden die Ergebnisse von Cui et al. herangezogen. Dabei werden nur die Ergebnisse mit $(\mu + \lambda)$-Selektion verwendet, die den CGP-Konfigurationen dieser Arbeit entsprechen [Cui24a]. Es ergibt sich ein HPDI-Maximalwert von ca. 495 Iterationen. Mit einer Iterationsgrenze von 500 ist es wahrscheinlich, dass vor allem schlechte Parametersätze zu einer Überschreitung dieses Werts führen. Durch die erhöhte Bestrafung dieser Überschreitung wird sichergestellt, dass die sichere Konvergenz des CGP-Algorithmus bis zur Iterationsgrenze priorisiert wird.
Da für Keijzer die Hyperparameteranalyse auf Basis der Fitness ausgewertet wird, können Güte-Aussagen ebenfalls getroffen werden, wenn der CGP-Algorithmus nicht vollständig konvergiert ist, da bereits vor Erreichen des Stopp-Kriteriums eine Aussage über das Konvergenzverhalten getroffen werden kann. Demnach wird für Keijzer die unter den Rechenzeit-Umständen höchste Iterationsgrenze von 500 Iterationen als genügend für die Hyperparameteroptimierung angesehen.

Mit Hilfe der optimierten Parametersätze können anschließend jeweils 50 CGP-Modelle pro Testszenario und CGP-Konfiguration ausgeführt werden. Die Ergebnisse dieser Modelle können verwendet werden, um die Güte der CGP-Konfigurationen auf einfache Testszenarien zu evaluieren. Das Evaluierungsverfahren wird in Abschnitt 3.2 näher erläutert.

2. Testblock: komplexe Testszenarien

Da sich durch die erhöhte Rechenzeit der komplexeren Testszenarien keine sinnvolle Hyperparameteroptimierung ausführen lässt, wird für diese Fälle eine andere Teststrategie

entwickelt.
Diejenigen Hyperparameter eines CGP-Modells, die nicht mit Rekombination in Verbindung stehen, werden aus den Ergebnissen von Cui et al. herangezogen [Cui24a]. Die Rekombinationsrate wird variiert und mit diesen Parametern in CGP-Modelle eingepflegt. Für jede CGP-Konfiguration werden 50 Testdurchläufe ausgeführt, um eine statistische Auswertung ausführen zu können. Dabei werden die CGP-Modelle gegebenenfalls nicht bis zur vollständigen Konvergenz trainiert, um die Rechenzeit zu reduzieren. Für Boolesche Probleme werden den CGP-Modellen dabei weniger Iterationen Trainingszeit zur Verfügung gestellt als für SR Probleme. Dies ergibt sich daraus, dass Boolesche Probleme in der Hyperparameteranalyse von Cui et al. mehr Rechenknoten brauchen und somit mehr Rechenzeit für jeweils eine Iteration benötigen als es bei SR der Fall ist. [Cui24a]
Da sich die Wirkungen der Rekombinationsrate und des Offsets gegenseitig beeinflussen können, ist es sinnvoller die Bewertung der beiden Parameter einzeln zu betrachten. Aufgrund der hohen Anzahl der zu bewertenden Testergebnisse, die sich bereits für einen variierenden Parameter ergeben, soll nur einer dieser Parameter in dieser Arbeit näher betrachtet werden. Dieser Parameter ist wie bereits beschrieben die Rekombinationsrate. Dies ergibt sich daraus, dass die Ergebnisse der Hyperparameter- und Rohdatenanalyse der einfachen Tests bereits vermuten lassen, dass der Offset keinen deutlichen Mehrwert für das CGP-Training bietet. Details zu den jeweiligen Ergebnissen können im Abschnitt 4.1 nachgelesen werden. Außerdem können mit Hilfe der Rekombinationsrate nähere Erkenntnisse zu den unterschiedlichen Rekombinationsarten gewonnen werden, die in dieser Arbeit verglichen werden.
Für die Evaluation stehen nach dem Testdurchlauf die Ergebnisse mehrerer CGP-Konfigurationen zur Verfügung, die allerdings nicht unbedingt die Ergebnisse der CGP-Modelle mit den jeweils besten Parametersätzen darstellen, da keine Hyperparameteroptimierung ausgeführt wurde. Trotzdem können die Ergebnisse verwendet werden, um einen näheren Einblick zum Verhalten des CGP-Modells zu erhalten, wenn die Rekombinationsrate variiert wird.

3.2 Evaluation

Für die Auswertung der Ergebnisse werden für jedes CGP-Training verschiedene Metriken aufgezeichnet, die einen Einblick in die Effizienz der Modelle geben sollen. Diese werden für jede Trainingsiteration gespeichert, um den Verlauf beobachten und bewerten zu können. In dieser Arbeit handelt es sich dabei um die Fitness nach der Rekombination und die

Fitness nach der Mutation. Die Evaluation umfasst zwei unterschiedliche Techniken zur Bewertung der ausgezeichneten Daten. Diese werden im Folgenden erläutert.

Analyse der Rohdaten: Der erste Evaluationsschritt umfasst eine händische Analyse der Rohdaten. Dabei werden die Ergebnisse der Hyperparameteranalyse näher betrachtet und bewertet. Es werden erste Erkenntnisse über die Effizienz der verschiedenen Rekombinationstypen gesammelt, indem beispielsweise die Anzahlen der Rechenknoten miteinander verglichen werden. Außerdem wird beobachtet wie die unterschiedlichen Rekombinationsparameter gesetzt werden.
Des Weiteren werden die Ergebnisse des CGP-Trainings näher analysiert. Der Trainingserfolg von Rekombination und Mutation kann anhand der jeweiligen Fitness-Werte gegenübergestellt werden.

Bayes'sche Analyse: Die Bayes'sche Analyse wird einerseits für die Sortierung der Effizienz der CGP-Konfigurationen verwendet. Dafür wird wie in Abschnitt 2.6 beschrieben das Plackett-Luce-Modell verwendet. Für die einfacheren Testszenarien Parity und Keijzer können so die Ergebnisse, der aus der Hyperparameteranalyse erhaltenen besten CGP-Konfigurationen miteinander verglichen werden. Für die komplexeren Testszenarien können vor allem unterschiedliche Rekombinationsarten und Rekombinationsraten-Arten miteinander verglichen werden.
Für weitere Bewertungen wird das Gammaverteilung-basierte Modell genutzt. Dabei muss für die komplexeren Testszenarien entschieden werden, ob die Iterationen bis zur Konvergenz oder die Fitness zu einer bestimmten Iteration bewertet werden muss, da zum Testzeitpunkt nicht klar ist, ob genug Durchläufe der CGP-Modelle konvergieren. Wenn genug Konvergenzen erreicht werden, können die Iterationen bis zu dieser, sinnvoll bewertet werden. Ist dies nicht der Fall, macht es mehr Sinn die Fitness zur letzten aufgezeichneten Iteration zu bewerten. Durch die Berechnung des Mittelwerts und HPDI der Iterationen/Fitness für jede CPG-Konfiguration können sowohl die einfacheren als auch die komplexeren Testszenarien sinnvoll ausgewertet werden. Für Parity und Keijzer kann somit eingeschätzt werden, durch welche Konfiguration ein CGP-Modell die Ausgangsprobleme schneller lösen kann als andere. Außerdem kann bewertet werden wie hoch die Streuung dieser Ergebnisse ist. Für die komplexeren Ausgangsprobleme kann beobachtet werden, welche Auswirkungen die Wahl der Rekombinationsparameter auf die Effizienz der CGP-Modelle hat.
Die *Prior-Sensitivitätsanalyse*, die Cui et al. in ihrem Paper untersuchen, wird aus Gründen des Umfangs in dieser Arbeit nicht näher betrachtet [CMH23].

4 Ergebnisse

In den folgenden Abschnitten werden die Ergebnisse dieser Arbeit aufgezeigt und analysiert. Sie sind unterteilt in die beiden Evaluations-Strategien, die in Abschnitt 3.2 erläutert wurden: Analyse der Rohdaten und bayes'sche Analyse. Zur Vereinfachung wird in den Ergebnistabellen die konstante Rekombinationsrate kurz „Konstant" genannt, die linear fallende Rekombinationsrate „Clegg " und die Rekombinationsrate mit der One-Fifth Regel „One-Fifth".

4.1 Ergebnisse Rohdatenanalyse

Die Analyse der Rohdaten besteht aus zwei Teilen. Die Ergebnisse der Hyperparameteranalyse werden für Parity und Keijzer näher betrachtet. Es werden dabei folgende Werte für jede CGP-Konfiguration aufgezeigt:

- **Anzahl der Rechenknoten:** Anzahl der Rechenknoten in jedem Chromosom des CGP-Modells
- λ **(Nachwuchs):** Anzahl der Nachwuchs-Chromosomen in jeder Generation; λ-Wert in der (μ+λ)-ES
- **Start-Rekombinationsrate:** Rekombinationsrate bei der Initialisierung; nur relevant für die konstante Rekombinationsrate und die One-Fifth Regel (linear fallende Rekombinationsrate wird mit 0,9 initialisiert)
- **Delta Rekombinationsrate:** Dieser Wert wird bei linear fallender Rekombinationsrate in jeder Generation von der bisherigen Rekombinationsrate abgezogen.
- μ **(Elitisten):** Anzahl der Elitisten in jeder Generation; μ-Wert in der (μ+λ)-ES
- **Offset:** Gibt an in wie vielen Trainings-Iterationen nach Initialisierung des CGP-Modells keine Rekombination ausgeführt wird.

Des Weiteren werden für alle Testszenarien die Ergebnisse des CGP-Trainings näher betrachtet. Dabei werden folgende Metriken für jede CGP-Konfiguration aufgezeigt:

- **Anzahl positive Mutationen:** Gibt an wie häufig Mutation zu einer Verbesserung der Fitness geführt hat (summiert über alle 50 Durchläufe).
- **Anzahl positive Rekombination:** Anzahl der Rekombinationsschritte, die die Fitness verbessert haben (summiert über alle 50 Durchläufe)
- **Anzahl negative Mutationen:** Anzahl der Mutationsschritte, bei denen der vorherige Rekombinationsschritt eine Verbesserung der Fitness erzielt hat, die durch Mutation reduziert oder vollständig aufgehoben wurde (summiert über alle 50 Durchläufe)
- **Median Iterationen positive Rekombination:** Median der Iterationen, bei denen der Rekombinationsschritt zur Verbesserung der Fitness beigetragen hat (über 50 Durchgänge hinweg)
- **Median Iterationen bis Konvergenz:** Median der Iterationen bis das Stopp-Kriterium erfüllt wurde (über 50 Durchläufe hinweg)
- **Stopp-Kriterium erfüllt:** Gibt an wie häufig das Stopp-Kriterium bei 50 Durchläufen erfüllt wurde. Dieser Wert wird verwendet, um Entscheidungen darüber zu treffen, welche Metrik in der bayes'schen Analyse betrachtet werden soll (für die komplexeren Testszenarien).

4.1.1 Rohdatenanalyse: Parity

Die Tabelle A.1 zeigt die Hyperparameter, die für die Ausführung des CGP-Trainings für Parity verwendet werden. Dabei werden die Ergebnisse der Hyperparameteranalyse verwendet, um die effizientesten CGP-Konfigurationen miteinander zu vergleichen. Zu beachten ist, dass bei der Optimierung des Rekombinations-Offsets dem Optimierer ebenfalls die Möglichkeit gegeben wurde den Offset auf 0 zu stellen und somit auszuschalten. Diese Möglichkeit wurde bei zwei aus neun CGP-Konfigurationen genutzt. Für diese Fälle aus der Hyperparameteroptimierung wurden die nächstbesten Hyperparameter gewählt, die einen Offset enthalten. Die jeweiligen Offset-Werte sind in der Tabelle rot markiert. Alle Beobachtungen in diesem Abschnitt beziehen sich ausschließlich auf das Parity-Testszenario. Basierend auf diesen Daten lassen sich demnach keine allgemeingültigen Schlüsse ziehen und alle Aussagen müssen kritisch begutachtet werden.

Betrachtet wird zuerst die Anzahl der Rechenknoten aus Tabelle A.1. Zu beobachten ist hier, dass die CGP-Konfiguration ohne Rekombination weniger Rechenknoten benötigt als der Mittelwert aller Verfahren von ca. 1305 Rechenknoten pro Chromosom. Die Modelle, die die One-Point Rekombination verwenden, weisen mit ca 1442 den höchsten Mittelwert an Rechenknoten auf. Mit durchschnittlich 1350 Rechenknoten haben die Modelle mit Two-Point Rekombination ein wenig mehr Rechenknoten benötigt als der Durchschnitt. Mit 1200 Rechenknoten im Mittel haben Modelle, welche die Uniform Rekombination verwenden weniger Rechenknoten als der Durchschnitt benötigt. Da eine höhere Anzahl der Rechenknoten eine höhere Rechenzeit impliziert, ist es von Vorteil diese reduzieren zu können. Unter dem Aspekt lässt sich in der Analyse der Hyperparameteroptimierung von Parity ein Trend schätzen: wird keine Rekombination verwendet, könnten Rechenzeit und Speicherressourcen gespart werden, indem weniger Rechenknoten benötigt werden. Falls Rekombination verwendet werden soll, könnte es sich lohnen die aufwendigeren Rekombinationsalgorithmen einzusetzen, da es den Anschein hat, dass diese weniger Rechenknoten benötigen als die einfacheren Rekombinationsverfahren. Diese Aussagen können allerdings nicht allein auf der Auswertung eines einzelnen Testszenarios basieren, liefern aber Denkanstöße für weitere Betrachtungen.
Des Weiteren fällt auf, dass die Streuung der Anzahl der Rechenknoten immer größer zu werden scheint, umso komplexer das Rekombinationsverfahren gewählt wird. Während bei One-Point Rekombination alle Werte relativ nah am Mittelwert liegen, entsteht bei der Uniform Rekombination ein großer Unterschied zwischen extrem großen und sehr kleinen Werten. So könnte es sich im Hinblick auf die verwendeten Ressourcen lohnen Rekombination einzusetzen, wenn durch eine gute Hyperparameteranalyse festgestellt werden kann, mit welcher CPG-Konfiguration das Einsparpotential am besten ausgeschöpft werden könnte. Zu beachten ist hierbei allerdings auch, dass komplexere Rekombinationsalgorithmen (und Rekombination im allgemeinen) grundsätzlich mehr Rechenzeit benötigen, als wenn dieser Rekombinationsschritt ausgelassen wird. Diese Metriken müssten sinnvoll miteinander abgeglichen werden.
Wird die Populationsgröße ($\mu + \lambda$) näher betrachtet, fällt auf, dass auch hier das Verfahren ohne Rekombination eine deutlich kleinere Population benötigt als die meisten anderen Verfahren. Auch dieser Faktor spart Ressourcen ein und kann für die Entscheidung für oder gegen ein Verfahren eine Rolle spielen. Die CGP-Konfiguration „Two-Point Clegg mit Offset“ hat ebenfalls eine sehr niedrige Populationsgröße erreicht, allerdings handelt es sich bei den Werten dieses Verfahrens nicht um die beste Parametrierung der Konfiguration. Hier wäre der Offset wie vorher beschrieben auf 0 gesetzt worden, weshalb nur der zweitbeste Parametersatz gewählt wurde. Der beste Parametersatz hat die Werte $\lambda = 44$ und $\mu = 12$, was wiederum eine größere Populationsgröße ergibt.
Bei der Begutachtung der verschiedenen Rekombinationsraten (Start-Rekombinationsrate

und Delta Rekombinationsrate) können keine genauen Zusammenhänge zwischen den CGP-Konfigurationen und der Höhe der Rekombinationsraten erkannt werden. Für die Start-Rekombinationsrate kann allerdings bei allen Verfahren mit konstanter Rekombinationsrate beobachtet werden, dass extrem hohe oder sehr niedrige Raten vermieden werden. Bei der One-Fifth Regel wird der gesamte freigegebene Wertebereich für die initiale Rekombinationsrate ausgenutzt. In weiteren Tests könnte überprüft werden, ob die vorgegebenen Wertebereiche für die One-Fifth Regel sinnvoll gewählt wurden oder ob die Hyperparameteranalyse für diese Fälle Werte außerhalb des vorgegebenen Intervalls bevorzugen würde. Auch bei der linear fallenden Rekombinationsrate wurde nahezu der gesamte Definitionsbereich der Hyperparameteranalyse ausgenutzt.
Zuletzt wird die Offset-Spalte betrachtet. Der Mittelwert der gewählten Offsets liegt bei ca. 27 Iterationen. Am geringsten fällt der Mittelwert für die Two-Point Rekombination aus mit einem Durchschnittswert von ca. 17 Iterationen. Werden nicht die nachbearbeiteten Zeilen der Hyperparameteranalyse betrachtet, sondern die originalen Ergebnisse, ist der Mittelwert sogar noch kleiner, mit 10 Iterationen ohne Rekombination. Im Vergleich dazu liegt die Uniform Rekombination mit durchschnittlich ca. 37 Iterationen ohne Rekombination weit darüber. Auf Basis dessen Stellt sich die Frage, ob die Uniform Rekombination weniger effizient ist als beispielsweise die Two-Point Rekombination, da mehr Rekombinationsschritte ausgesetzt werden. Die zwei neu eingepflegten Offset-Werte (rot) entsprechen ungefähr dem allgemeinen Mittelwert.

Die folgend bewertete Tabelle A.2 gibt einen Einblick in die Rohdaten der CGP-Trainings. Alle CGP-Konfigurationen wurden dabei 50 mal am Parity-Datensatz getestet. Die Tabelle zeigt einige wichtige Merkmale des CGP-Trainings auf.
Bei der Betrachtung der Tabelle wird deutlich, dass die Mutationsschritte deutlich häufiger zu einer Verbesserung der Fitness führen als die Rekombinationsschritte. Für die CGP-Konfigurationen mit Offset kann beobachtet werden, dass keine einzige Rekombination die Fitness verbessern konnte. Dies deutet darauf hin, dass Rekombination in den ersten Iterationen des Trainings effizienter ist, da sie eigenständig zu einer besseren Fitness führt und nicht nur die Kombination aus Rekombination und anschließender Mutation die Fitness verbessert. Um diesen Zusammenhang zu validieren, wird der Median der Iterationen betrachtet, bei denen die positiven Rekombinationen auftreten. Wird dieser Wert mit dem Median der Iterationen bis zur Konvergenz verglichen, kann festgestellt werden, dass die Rekombination besonders in frühen Trainingsphasen einen höheren Mehrwert bringt. Für CGP-Konfigurationen mit linear fallender Rekombinationsrate werden relativ hohe Werte an positiver Rekombination erreicht. Dies könnte damit zusammenhängen, dass bei der Initialisierung eine hohe Rekombinationsrate (0,9) gewählt wird und somit die Rekombination in den anfänglichen Iterationen wahrscheinlicher ausgeführt wird. Für

die konstante Rekombinationsrate ist dies nicht der Fall, da die Rekombinationsrate laut Tabelle A.1 recht niedrig gewählt wurde (0,3). Werden die Mediane der Iterationen bis zur Konvergenz betrachtet, sollte dementsprechend festgestellt werden können, dass die CGP-Modelle mit linear fallender Rekombinationsrate deutlich geringere Werte aufweisen als die restlichen Modelle. Dies ist allerdings nicht Fall. Alle CGP-Modelle mit Rekombination brauchen ähnlich viele Iterationen bis zur Konvergenz unabhängig von der Rekombinationsraten-Art. Dies kann an mehreren Gründen liegen. Einer davon kann sein, dass der Median der Iterationen bis zur Konvergenz nur diejenigen Trainingsdurchläufe einbezieht, die das Stopp-Kriterium erfüllen. Es kann also sein, dass linear fallende Rekombinationsraten dazu führen, dass mehr Testdurchläufe konvergieren, ohne dass dies in der Tabelle deutlich wird. Ein anderer Grund könnte sein, dass nicht alle positiven Rekombinationen aus Tabelle A.2 auch in das Training des CGP-Modells einfließen. Die Spalte „Anzahl negative Mutationen“ gibt an in wie vielen Fällen eine positive Rekombination durch Mutation zerstört wurde. Das heißt in solchen Trainingsschritten wurde durch Rekombination eine Verbesserung der Fitness bewirkt. Anschließend wurde in der gleichen Iteration ein Mutationsschritt ausgeführt, der die Fitness wieder verschlechtert. Da die Selektion der neuen Elitisten nur nach dem Mutationsschritt stattfindet, können Fortschritte aus dem Rekombinationsschritt verloren gehen. Zu beobachten ist, dass die Anzahlen der verlorenen positiven Rekombinationen höhere Werte aufweisen, wenn bereits mehr Rekombinationsschritte zu einer Verbesserung der Fitness beigetragen haben. Dies könnte ein Grund dafür sein, dass sich die Anzahl der positiven Rekombinationen nicht auf die Iterationen bis zur Konvergenz auswirkt.
Bei Verwendung der One-Fifth Regel erreichen die CGP-Modelle einen geringeren Wert an positiven Rekombinationen, was ein Zeichen dafür sein kann, dass diese Rekombinationsrate die Effizienz der Rekombination nicht sinnvoll steigert. Außerdem kann auch festgestellt werden, dass komplexere Rekombinationsalgorithmen mehr positive Rekombinationen aufweisen leichtere. Dies kann darauf hindeuten, dass komplexere Rekombinationsarten, wie die Uniform Rekombination, die Fitness mit einer höheren Wahrscheinlichkeit verbessern. Allerdings wurden für Uniform Rekombinationen ohne Offset laut Tabelle A.1 relativ hohe Rekombinationsraten gewählt, die auch einen Grund für die hohe Anzahl an positiven Rekombinationen darstellen könnten.
Wird der „Median Iterationen bis Konvergenz“ von CGPs ohne Rekombination mit den Durchschnittswerten pro Rekombinationsart (One-Point, Two-Point und Uniform) verglichen, dann fällt auf, dass die CGPs ohne Rekombination mehr Iterationen brauchen bis sie konvergieren. Das deutet darauf hin, dass es sich lohnen könnte Rekombination anzuwenden, da diese im Durchschnitt weniger Iterationen bis zur Konvergenz benötigen. Allerdings werden hier wieder nur die Durchläufe gewertet, die das Stopp-Kriterium erreicht haben. Diese Aussage muss also mit der bayes'schen Analyse und der graphischen

Analyse erneut bewertet werden.
Wenn die Auswirkungen des Offsets mit den Ergebnissen der Hyperparameteroptimierung verglichen werden, wirken die Ergebnisse vorerst widersprüchlich. Da die Offsets den Mehrwert der Rekombination blockieren besteht die Frage, wieso 7 aus 9 CGP-Konfigurationen trotzdem einen Offset verwenden sollen. Dies kann einerseits daran liegen, dass die Hyperparameteranalyse mehr Parametersätze hätte testen müssen, um auf effizientere Ergebnisse zu stoßen. Ein anderer Grund könnte sein, dass der Offset im Zusammenhang mit den negativen Mutationen steht. Es könnte sein, dass die positiven Ergebnisse der Rekombination mögliche positive Ergebnisse der Mutation beeinflussen und anschließend (bei einer Kombination aus beiden genetischen Operatoren) eine schlechtere Fitness erzielt wird. In diesen Fällen würde die Rekombination Fortschritte bringen, wenn anschließend keine Mutation ausgeführt werden würde. Wenn aber eine Mutation ausgeführt wird, kann nicht nur der Fortschritt der Rekombination aufgehoben werden, sondern auch der potentielle Fortschritt der Mutation entfallen. Die Uniform Rekombination weist hohe Werte auf, wenn es um die Anzahl der negativen Mutationen geht. Werden die Ergebnisse auf der Hyperparameteranalyse in Tabelle A.1 hinzugezogen, kann beobachtet werden, dass Uniform die höchsten Werte an Offsets aufweist. Außerdem musste keiner der Uniform-Offset-Werte korrigiert werden, da keine CGP-Konfiguration mit Uniform Rekombination den Offset anfänglich ausgesetzt hat. Der Zusammenhang zwischen negativen Mutationen und Offset lässt sich allerdings nicht für alle Werte beobachten und muss demnach weiter erforscht werden.

4.1.2 Rohdatenanalyse: Keijzer

Die Umstände der Hyperparameteroptimierung für den Keijzer-Testfall sind identisch zu der des Parity-Testszenarios und können in Abschnitt 4.1.1 nachgelesen werden. Wie bei den Ergebnissen bei Parity gibt es auch bei der Offset-Optimierung im Keijzer-Testszenario zwei aus neun Fällen, bei denen der Offset ursprünglich auf 0 gesetzt wurde. Diese Parametersätze wurden durch den jeweils zweitbesten Parametersatz ersetzt und sind in der Tabelle rot markiert. Die Beobachtungen in diesem Abschnitt beziehen sich nur auf den Keijzer-Benchmark. In Abschnitt 4.1.5 werden abschließende Erkenntnisse aus den Hyperparameteranalysen von Parity und Keijzer zusammengetragen.
Die Tabelle A.3 zeigt die Ergebnisse der Hyperparameteranalyse des Keijzer-Testszenarios. Wie bereits in Abschnitt 4.1.1 erklärt, kann eine Reduktion der Rechenknoten in einem CGP-Modell dazu führen, dass Systemressourcen und Rechenzeit eingespart werden können. Dies gilt natürlich nur unter Anbetracht, dass die Lösungsentwicklung des CGPs nicht darunter leidet. Wird der allgemeine Mittelwert aller verwendeten Rechenknoten gebildet,

liegt der Wert bei ungefähr 1305. Zu beobachten ist, dass die Anzahl der Rechenknoten beim CGP-Verfahren ohne Rekombination deutlich höher als dieser Durchschnittswert ausfällt. Alle Mittelwerte der Anzahl der Rechenknoten für die einzelnen Rekombinationsalgorithmen liegen unterhalb des allgemeinen Mittelwerts. Das in diesem Sinne beste Verfahren ist hier die Two-Point Rekombination. Dieser Rekombinationsalgorithmus braucht durchschnittlich ca. 983 Rechenknoten und liegt damit weit unter dem allgemeinen Mittelwert. Mit ca. 1117 Rechenknoten im Mittel folgt die One-Point Rekombination, während mit durchschnittlich 1300 Rechenknoten die Uniform Rekombination ungefähr dem allgemeinen Mittelwert entspricht. Bei Betrachtung der Werte fällt außerdem auf, dass die One-Point Rekombination einen großen Unterschied zwischen höchster und niedrigster Anzahl der Rechenknoten aufweist: Einerseits wird der allgemein niedrigste Wert für die verwendeten Rechenknoten aufgezeigt (350), andererseits wird die obere Grenze ausgereizt (2000). Bei der Two-Point Rekombination kann dieser Effekt nicht beobachtet werden. Außer eines Ausreißers (1700) sind alle Werte relativ ähnlich. Allerdings kann auf Basis der Werte keine Aussage darüber getroffen werden, welche Parameter mit diesen Beobachtungen zusammenhängen. Weder die Rekombinationsrate noch andere Parameter weisen offensichtliche Korrelationen auf. Dieser Aspekt könnte in weiteren Untersuchungen näher betrachtet werden.
Ein Faktor, der ebenfalls Rechenzeit einsparen kann ist die Populationsgröße (μ+λ). Diese weisen im Keijzer-Testszenario keine deutlich ersichtlichen Zusammenhänge zu den CGP-Konfigurationen auf. Die μ- und λ-Werte weichen nicht stark voneinander ab, mit einigen unregelmäßigen Schwankungen, bei denen die Werte deutlich kleiner werden. Allerdings können diese Schwankungen nicht gleichzeitig bei μ und λ festgestellt werden. Für Two-Point und Uniform Rekombination verringert sich der λ-Wert bei der linear fallenden Rekombinationsrate ohne Offset. Allerdings kann durch diese Ergebnisse nicht näher ergründet werden, ob darin ein näherer Zusammenhang liegt.
Bei den Rekombinationsraten fällt auf, dass für die Start-Rekombinationsrate der One-Fifth Regel nur sehr hohe oder sehr niedrige Raten gewählt wurden. Dies könnte darauf hindeuten, dass noch extremere Werte gewählt worden wären, wenn die Grenzen der Start-Rekombinationsrate einen größeren Spielraum zugelassen hätten. Für die One-Point und Uniform Rekombination wurden für die konstanten Start-Rekombinationsraten ebenfalls Werte nahe des Definitionsbereichs gewählt. Für die Two-Point Rekombination trifft dies nicht zu. Ebenso werden bei der linear fallenden Rekombinationsrate solche extremen Werte beobachtet. Zusammenfassend lässt sich feststellen, dass für das Keijzer-Testszenario überwiegend die Grenzen des Definitionsbereichs als Rekombinationsraten gewählt wurden.
Die Offsets wurden durchschnittlich bei 153 Iterationen gewählt, wobei zu beachten ist, dass zwei Parametersätze ursprünglich bessere Ergebnisse erbracht hätten, ohne die Re-

kombination zu Beginn auszusetzen. Die in der Tabelle A.3 rot markierten Offsets wären dementsprechend ursprünglich gleich 0 gewesen. Mit den originalen Parametersätzen wäre der Mittelwerts des Offsets bei 100 gelegen. Werden die Werte der Tabelle A.3 mit den angepassten (roten) Einträgen betrachtet, fällt auf, dass der Offset-Mittelwert der CGP-Konfigurationen mit Uniform Rekombination kleiner ist als derjenige von One-Point und Two-Point Rekombination. Werden diese verbesserten Parametersätze wieder auf ihren Originalwert (0) zurückgesetzt, ändert sich dies allerdings: Dadurch wird der durchschnittliche Offset-Wert der Uniform Rekombination höher als die jeweils anderen. In diesem Fall ist der Durchschnitts-Offset der Two-Point Rekombination der geringste. Es fällt außerdem auf, dass die zweitbesten Parametersätze Offset-Werte besitzen, die deutlich höher sind als der Mittelwert aller Offsets. Der Offset der One-Point Rekombination mit linear fallender Rekombinationsrate ist dabei sogar der maximal mögliche Offset-Wert (300). Diese extreme Schwankung der Offsets zwischen erstbestem und zweitbestem Parametersatz spricht dafür, dass die Güte eines CGPs nicht kausal mit der Höhe des Offsets zusammenhängt.

Folgend werden die Rohdaten der Keijzer-Testdurchläufe anhand von Tabelle A.4 bewertet. Wie für Parity wurden dabei 50 Durchläufe für jede CGP-Konfiguration ausgeführt, um statistische Abweichungen einzufangen. Im allgemeinen lässt sich durch die Betrachtung der Tabelle feststellen, dass die Mutation deutlich öfter eine bessere Fitness hervorbringt als die Rekombination. Außerdem tritt keine Verbesserung der Fitness durch Rekombination ein, wenn ein Offset eingesetzt wurde. Dieser Fakt und derjenige, dass der Median der Iterationen für positive Rekombinationen deutlich geringere Werte hervorbringt als der Median der Iterationen bis zur Konvergenz, lässt darauf schließen, dass Rekombinationsschritte zu Beginn des Trainings eine höhere Wahrscheinlichkeit aufweisen die Fitness zu verbessern als zu späteren Iterationen.
Ein Trend, der sich durch die Ergebnisse aufzeigt, ist, dass mehr positive Rekombinationen auftreten, umso komplexer die Rekombinationsarten gewählt werden. So weisen CGP-Konfigurationen mit Uniform Rekombination deutlich mehr positive Rekombinationen auf als solche mit One-Point Rekombination. Außerdem treten vermehrt negative Mutationen auf, wenn mehr positive Rekombinationen beobachtet werden. Für die One-Point und Two-Point Rekombination führt die linear fallende Rekombinationsrate ohne Offset zu den meisten positiven Mutationen. Dies lässt sich dadurch erklären, dass die linear fallende Rekombinationsrate einen hohen Initialisierungswert (0,9) aufweist und die anderen Rekombinationsraten-Arten in diesen Fällen einen niedrigeren Wert durch die Hyperparameteranalyse erhalten haben. Zu beobachten ist, dass die konstante Rekombinationsrate für die Uniform Rekombination eine höhere Start-Rekombinationsrate verwendet (0,8) und deutlich mehr positive Rekombinationen erzielt als die linear fallende Rekombinationsrate. Die Rekombination mit der One-Fifth Regel erzielt auch mit hohen Start-

Rekombinationsraten relativ wenige Rekombinationserfolge. Es könnte demnach ratsam sein eine konstant hohe Rekombinationsrate zu Beginn des Trainings einzusetzen und diese anschließend vollkommen auszusetzen.
Wird Median der Iterationen bis zur Konvergenz betrachtet, weißt das CGP-Modell ohne Rekombination einen deutlich höheren Wert auf, als der Durchschnitt der Mediane pro Rekombinationsart. So benötigt die Uniform Rekombination durchschnittlich ca. 389 Iterationen im Median bis zur Konvergenz, die One-Point Rekombination ca. 461 und die Two-Point Rekombination 533. Dies kann darauf hindeuten, dass es sich im Durchschnitt lohnt Rekombination auszuführen, allerdings muss beachtet werden, dass hierbei nur die konvergierten Trainings betrachtet werden. Diese Aussage muss also in weiteren Analysen validiert werden.
Wird der Einfluss bewertet, der der Offset auf die Mediane der Iterationen bis zur Konvergenz hat, kann zusammengefasst werden, dass die meisten CGP-Konfigurationen mit Offset weniger Iterationen bis zur Konvergenz brauchen als die jeweils gleichen CGP-Konfigurationen ohne Offset. Es könnte sein, dass positive Rekombinationen die Mutation negativ beeinflussen, sodass eine Rekombination zwar die Fitness verbessert aber sich negativ auf die nachfolgende Mutation auswirkt. Demnach könnte es sein, dass die alleinige Mutation eine größere Verbesserung der Fitness hervorrufen könnten als es eine positive Rekombination und anschließende Mutation schafft. Da in den Ergebnissen nur quantitative Bewertungen stattfinden und keine qualitativen, lässt sich dieser Fall nicht ausschließen oder bestätigen. Zu beachten ist außerdem, dass hierbei nur die Durchläufe betrachtet werden, die das Stopp-Kriterium bereits erreicht haben.
Neben dieser Beobachtung zum Offset-Verhalten, lässt dich erkennen, dass die Offsets in den meisten Fällen größer ausfallen, wenn der Anteil der verlorenen Rekombinationserfolge ($= \frac{\text{Anzahl negative Mutationen}}{\text{Anzahl positive Rekombinationen}}$) höher ist und gleichzeitig Rekombinationsschritte zum Erfolg geführt hätten. Dies kann ein Indiz dafür sein, dass der Offset in der Hyperparameteranalyse nur gewählt wurde, weil Rekombination und Mutation in einigen Iterationen Gegenspieler sind, statt einen gemeinsamen Erfolg zu erzielen. In diesen Fällen könnte es besser sein die Rekombination komplett auszusetzen.

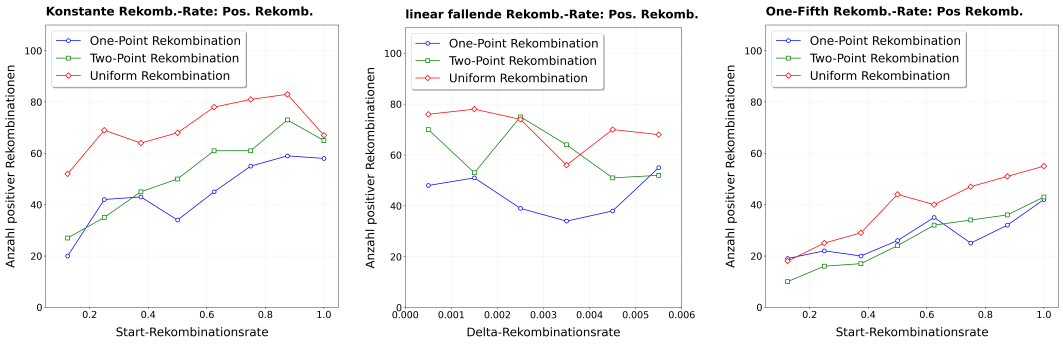
4.1.3 Rohdatenanalyse: Encode

In diesem Abschnitt werden die Rohdaten der Encode Benchmark Tests ausgewertet. Zu beachten ist, dass sich dieser Abschnitt nur auf die Ergebnisse von Encode bezieht und alle Schlussfolgerungen und Thesen in Abschnitt 4.1.5 erneut evaluiert werden. Denkanstöße und Beobachtungen jeder Art sind in diesem Abschnitt nur auf die jeweiligen Ergebnisse von Encode bezogen und stellen keine allgemeinen Aussagen dar.

Die Tabellen A.5, A.6 und A.7 zeigen die Ergebnisse von jeweils 50 Testdurchläufen mit 10000 Iterationen für die unterschiedlichen CGP-Konfigurationen. Dabei ist jede Tabelle auf eine Rekombinationsart eingeschränkt. Ziel ist es mehr über das Zusammenspiel von Rekombinationsart und Einstellungen der Rekombinationsrate zu erfahren. Der Offset wurde hier nicht weiter betrachtet, da die Ergebnisse von Parity und Keijzer zeigen, dass der Offset dazu führt, dass die Rekombination die Fitness des CGPs nicht verbessern kann. Außerdem schließt der Offset die Rekombination für einige Iterationen aus, was den Einblick auf die Auswirkungen der Rekombinationsraten auf den Erfolg des CGPs einschränkt. Aus diesen Gründen wird der Offset für diesen Abschnitt ausgeschlossen.

Werden die drei Tabellen A.5, A.6 und A.7 betrachtet, lässt sich feststellen, dass innerhalb der 10000 Iterationen nur die wenigsten CGP-Modelle konvergiert sind. Aus diesem Grund wird für den Encode Benchmark bei der bayes'schen Analyse die Wahrscheinlichkeitsfunktion der Fitness nach der 10000. Iteration gefittet und analysiert.
Um Zusammenhänge zwischen den CGP-Konfigurationen und der Güte der CGP-Modelle zu erforschen wurden Muster in verschiedenen Spalten der Tabellen miteinander abgeglichen. Dabei wurden die Spalten „Anzahl positive Rekombinationen", „Anzahl negativer Mutationen" und „Median Iterationen positive Rekombination" mit den zwei Spalten „Median Iterationen bis Konvergenz" und „Stopp-Kriterium erfüllt" verglichen. Bei diesem Musterabgleich wurde kein Hinweis darauf gefunden, dass die Spalten sich gegenseitig beeinflussen.
Bei der One-Point Rekombination haben durchschnittlich ca. 8 Durchläufe das Stopp-Kriterium erfüllt, bei der Two-Point Rekombination 3,7 und bei der Uniform Rekombination 8,5. Die CGP-Modelle mit Two-Point Rekombination erfüllen damit deutlich weniger Stopp-Kriterien als die CGP-Modelle ohne Rekombination. Dies gilt nicht nur für den Durchschnitt, sondern auch jeweils für alle CGP-Konfigurationen mit Two-Point Rekombination. Wird der Durchschnitt der konvergierten CGP-Modelle für jede Rekombinationsraten-Art (konstant, linear fallend und mit One-Fifth Regel) für jede Rekombinationsart verglichen, kann festgestellt werden, dass sich die Mittelwerte kaum voneinander unterscheiden. Beispielsweise konvergieren für die One-Point Rekombination mit konstanter Rekombinationsrate durchschnittlich 8,1 Durchläufe, bei der linear fallenden Rekombinationsrate 7,7 Durchläufe und bei der Rekombinationsrate mit One-Fifth Regel 8,4 Durchläufe. Dies gilt für alle Rekombinationsarten gleichermaßen. Auf den ersten Blick scheint es, als ob die Rekombinationsraten-Arten keinen großen Unterschied zueinander aufweisen, wenn es darum geht wie schnell die CGP-Modelle konvergieren. Außerdem scheint die Anzahl der positiven Rekombinationen keinen Einfluss auf die Anzahl der erfüllten Stopp-Kriterien zu haben, da bei der One-Point Rekombination mit One-Fifth Regel vergleichsweise wenig positive Rekombinationen zu einer ähnlichen Anzahl an konvergierten

CGP-Modellen führt. Über diese beiden Punkte lässt sich allerdings vorerst keine Aussage treffen, da nur ersichtlich ist, wie viele Durchläufe bis zur 10000. Iteration konvergieren. Es wird nicht aufgezeigt wie schnell die CGP-Modelle danach konvergieren. Ein Vergleich der Rekombinationsraten-Arten für das Encode-Testszenario muss demnach später bei weiteren Analysen stattfinden. Für weitere Testfälle könnten mehr Erkenntnisse gewonnen werden, wenn mehr Durchgänge das Stopp-Kriterium erfüllen.
Wird versucht die Anzahl der konvergierten CGP-Modelle mit der Höhe der jeweiligen Rekombinationsrate in Verbindung zu bringen, ergeben sich folgende Erkenntnisse: Für konstante Rekombinationsraten werden die meisten Stopp-Kriterien bei besonders hohen Rekombinationsraten erfüllt. Außerdem werden hierfür in der ersten Hälfte des Definitionsbereichs (Rekombinationsrate gleich 0,125-0,5) sporadisch auch Erfolge in der Konvergenz festgestellt. Für die linear fallende Rekombinationsrate werden bei besonders kleinen und großen Schrittweiten, mit denen die Rekombinationsrate verkleinert wird, eine größere Anzahl an erfüllten Stopp-Kriterien beobachtet. Für die One-Fifth Regel konnten keine Muster gefunden werden. Allerdings unterscheiden sich die Anzahlen der konvergenten Durchgänge nicht stark voneinander, was auch bedeuten kann, dass diese Schwankungen rein zufälliger Natur sind.
Die unterschiedlichen Mediane der Iterationen, bei denen die Rekombinationserfolge entstehen, unterscheiden sich nicht stark voneinander. Diese Beobachtung kann für alle CGP-Konfigurationen getroffen werden, unabhängig davon welche Rekombinationsart oder Rekombinationsraten-Art verwendet wird. Dies weist darauf hin, dass das Fenster an Iterationen, in denen die Rekombination die Fitness verbessern kann, nicht mit Hilfe der Rekombinationsrate beeinflusst werden kann.
Die folgende Abbildung 4.1 beschreibt die Zusammenhänge aus Rekombinationsraten-Arten und der Anzahl positiver Rekombinationen.



(a) Konstant (b) linear fallend (c) One-Fifth Regel

Abbildung 4.1: Encode: Anzahl positive Rekombinationen für die verschiedenen Arten der Rekombinationsrate

In Abbildung 4.1 kann beobachtet werden, dass für jede Rekombinationsraten-Art die Uniform Rekombination die höchsten Anzahlen an positiver Rekombination produziert. Für die konstante und linear fallende Rekombinationsrate ist die Two-Point Rekombination in dieser Hinsicht besser als die One-Point Rekombination. Außerdem wird ersichtlich, dass die Rekombinationsrate mit der One-Fifth Regel zu niedrigeren Rekombinationserfolgen führt als die anderen Rekombinationsraten-Arten.
Eine höhere Rekombinationsrate bedeutet zudem eine höhere Chance, dass die Rekombination ausgeführt wird und somit die Fitness verbessert. Die positive Steigung in den Abbildungen 4.1a und 4.1c kann dadurch erklärt werden. Allerdings bleibt unklar, warum die konstanten Rekombinationsraten von 1,0 weniger Rekombinationserfolge hervorbringen als die konstanten Rekombinationsraten von 0,875. Dies gilt für alle Rekombinationsarten mit der konstanten Rekombinationsrate.
Ebenfalls überraschend sind die kurzfristigen lokalen Maxima, die die Abbildungen 4.1a und 4.1c bei geringen Rekombinationsraten aufzeigen. Kleine Rekombinationsraten führen dazu, dass die Rekombination unwahrscheinlicher ausgeführt wird. Dass für eine kleinere Rekombinationsrate mehr Rekombinationserfolge vermerkt werden als für eine höhere Rekombinationsrate ist demnach ungewöhnlich. Dies könnte darauf hindeuten, dass die selteneren Rekombinationsschritte jeweils effektiver waren als diejenigen Rekombinationsschritte, die häufiger ausgeführt wurden.

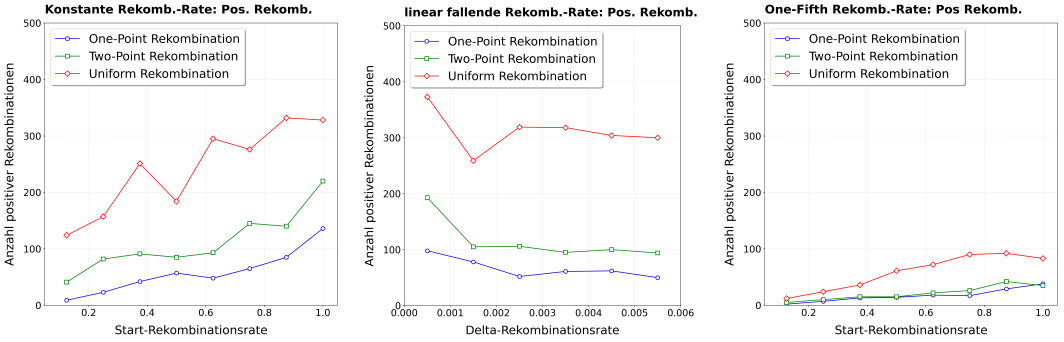
4.1.4 Rohdatenanalyse: Koza

Die Tabellen A.8, A.9 und A.10 zeigen die Rohdaten-Auswertungen des Koza-Testszenarios. Für jede CGP-Konfiguration wurden 50 Modelle innerhalb von maximal 50000 Iterationen trainiert. Jede der Tabellen stellt eine Rekombinationsart mit verschiedenen Parametrierungen der Rekombinationsraten dar. Wie in Abschnitt 4.1.3 wurde für das Koza-Testszenario der Offset als Parameter nicht weiter betrachtet. Dies geht aus den Ergebnissen von Parity und Keijzer hervor, bei denen keine erfolgreichen Rekombinationsschritte ausgeführt wurden, wenn ein Offset aktiv war. Da mit dieser Arbeit die Effizienz der Rekombination ausgewertet werden soll, ist es sinnvoll den Offset nicht näher zu betrachten, um den vollen Umfang der Fitnessverbesserungen durch Rekombination zu bewerten. Ziel des Abschnittes ist es die Zusammenhänge zwischen verschiedenen Rekombinationsraten-Arten und dem Erfolg des CGP-Trainings zu finden. Außerdem soll bestimmt werden, welche Metrik für die Auswertung der bayes'schen Analyse verwendet werden soll. In diesem Abschnitt werden nur die Ergebnisse von Koza ausgewertet und erlauben noch keine allgemeinen Aussagen. Diese werden in Abschnitt 4.1.5 ausgewertet, indem die Ergebnisse aus allen Testszenarien zusammengetragen und verglichen werden.

Aus den Spalten „Stopp-Kriterium erfüllt" der drei Tabellen A.8, A.9 und A.10 geht hervor, dass relativ viele Durchgänge des CGP-Trainings konvergiert sind. Deswegen wird für die bayes'sche Analyse in Abschnitt 4.2.4 die Anzahl an Iterationen bis zur Konvergenz als Metrik verwendet. Außerdem lässt sich erkennen, dass mehr Mutationen zu einer Verbesserung der Fitness geführt haben als Rekombinationsschritte. Gleichzeitig gehen einige Erfolge der Rekombinationsschritte verloren, indem anschließend noch eine Mutation ausgeführt wurde, wie in den Spalten „Anzahl negative Mutationen" zu sehen ist. Dies kann dadurch passieren, dass die Auswahl der Elitisten erst nach der Mutation getroffen wird und so keine Evaluation der Fitness zwischen Rekombination und Mutation stattfindet. Eine Möglichkeit dies zu beheben wäre es, die Elitisten nach der Rekombination mit den Elitisten nach der Mutation zu vergleichen und die besten Chromosomen zu verwenden, die aus beiden genetischen Operationen resultieren.
Die Rekombination ist dabei vor allem in den frühen Iterationen erfolgreich. Dabei unterscheiden sich die unterschiedlichen Rekombinationsraten-Arten. Die Rekombination mit der One-Fifth Regel weist vor allem in den ersten Iterationen Erfolge auf, mit der linear fallenden Rekombinationsrate ist der Median etwas höher und mit der konstanten Rekombinationsrate sind teilweise relativ hohe Iterationen noch erfolgreich. Für die konstante Rekombinationsrate werden diese hohen Iterationsmediane allerdings nur für die One-Point und Two-Point Rekombination gesichtet. Dabei fällt auf, dass diese späten Rekombinationserfolge im Training nicht im Zusammenhang mit der Höhe der Rekombinationsrate

stehen, da einige erhöhte Werte in den Spalten „Median Iterationen positive Rekombination“ auch für kleinere Rekombinationsraten aufgeführt werden. Außerdem kann kein Zusammenhang zwischen den Spalten „Median Iterationen positive Rekombination“ und „Median Iterationen bis Konvergenz“ gefunden werden. Das bedeutet, dass die Rekombinationserfolge nicht durch längeres Training automatisch in späteren Iterationen stattfinden, sondern dass dieses Verhalten unabhängig von der Trainingsdauer auftritt. Dieses Wissen könnte genutzt werden, um so Rekombination gezielter einzusetzen. Beispielsweise könnten die frühen Rekombinationserfolge der linear fallenden Rekombinationsrate genutzt werden, wobei diese ähnlich viele positive Rekombinationen aufweist wie die konstante Rekombinationsrate. So könnten die Erfolge der Rekombination in den ersten Iterationen genutzt und anschließend die Rekombinationsrate schneller reduziert werden.
Werden die Iterationen betrachtet, die die Modelle bis zur Konvergenz gebraucht haben, lässt sich feststellen, dass die CGP-Modelle mit Two-Point Rekombination durchschnittlich weniger Iterationen (278) gebraucht hat als die anderen. Darauf folgt die Uniform Rekombination (349 Iterationen) und anschließend die One-Point Rekombination (835 Iterationen). Demnach sind alle Mittelwerte der Iterationsmediane bis zur Konvergenz größer als der Median der Anzahl an Iterationen, die das CGP-Modell ohne Rekombination gebraucht hat (208). Dabei muss beachtet werden, dass bei der CGP-Konfiguration ohne Rekombination alle Hyperparameter optimiert wurden. Demnach liefern die CGP-Modelle ohne Rekombination optimierte Ergebnisse während alle anderen CGP-Konfigurationen dieses Potential noch ausschöpfen können. Trotzdem können für einige vereinzelte CGP-Konfigurationen bereits frühere Konvergenzen bei gleicher Anzahl an erfüllten Stopp-Kriterien beobachtet werden. Ein Beispiel bietet die Konfiguration „Uniform Clegg: 0,0015“ aus Tabelle A.10. Dies ist ein Anzeichen dafür, dass CGP-Modelle durch Rekombination effizienter werden können, wenn die Rekombination sinnvoll eingesetzt wird.
Des weiteren wurde beim Vergleich der Spalten „Anzahl positive Rekombinationen“, „Anzahl negative Mutationen“ und „Median Iterationen positive Rekombinationen“mit den Spalten „Median Iterationen bis Konvergenz“ und „Stopp-Kriterium erfüllt“ keine zusammenhängenden Muster erkannt. Eine Möglichkeit dafür, dass die Anzahl der positiven Rekombinationen keine deutliche Veränderungen in der Effizienz des CGPs hervorruft, könnten die negativen Mutationen sein. In weiteren Tests könnte dieser Zusammenhang untersucht werden, indem keine Mutation ausgeführt wird, wenn bereits eine Rekombination die Fitness des Modells verbessert hat. So könnte überprüft werden, ob die CGP-Modelle mit mehr Rekombinationserfolgen auch bessere Ergebnisse liefern oder ob Mutationserfolge relevanter für ein erfolgreiches CGP-Training sind.
Ein weiterer Punkt der auffällt sind die relativ hohen Zahlen der negativen Mutationen im Koza-Testszenario. Dies könnte daran liegen, dass die Fitness-Werte in SR Benchmarks kontinuierlich sind, während sie in booleschen Problemen durch diskrete Werte abgebil-

det werden. Dadurch können auch kleinere Trainingserfolge durch Rekombination in den Tabellen abgebildet werden. Dies könnte bedeuten, dass vor allem bei SR die Evaluierung nach der Rekombination als Zwischenschritt sinnvoll sein könnte.
Um die positiven Rekombinationen zu bewerten wird folgende Abbildung 4.2 eingeführt:



(a) Konstant (b) linear fallend (c) One-Fifth Regel

Abbildung 4.2: Koza: Anzahl positive Rekombinationen für die verschiedenen Arten der Rekombinationsrate

Abbildung 4.2 beschreibt die positiven Rekombinationsraten pro CGP-Konfiguration, aufgeteilt in die verschiedenen Rekombinationsraten-Arten. Dabei sind die verschiedenen Rekombinationsarten mit unterschiedlichen Farben dargestellt. Zu erkennen ist, dass die Uniform Rekombination für jede Rekombinationsraten-Art mehr positive Rekombinationen erzeugt als es die anderen Rekombinationsarten schaffen. Die Rekombination mit der One-Fifth Regel schneidet in dieser Betrachtung schlechter ab als die konstante und linear fallende Rekombinationsrate. Zu beachten ist, dass es sich dabei um eine quantitative Bewertung handelt. Es wird also nicht ersichtlich wie hoch die jeweiligen Trainingserfolge der Rekombinationen ausfallen. Es macht Sinn dies besonders für SR Benchmarks in weiteren Tests näher zu betrachten, da hier die Fitness einen kontinuierlichen Wert einnimmt und sich die Trainingserfolge stark voneinander unterscheiden können.
Für die konstante Rekombinationsrate und diejenige mit der One-Fifth Regel, werden mit höheren Rekombinationsraten auch höhere Chancen gegeben, dass eine Rekombination die Fitness verbessert. Demnach macht es Sinn, dass für höhere Rekombinationsraten die Anzahl an positiven Rekombinationen steigt, wie es in den Abbildungen 4.2a und 4.2c der Fall ist. Interessant werden diejenigen lokalen Maxima, die bereits entstehen, bevor jeweils die maximale Rekombinationsrate erreicht wurde. In diesen Punkten könnte die Rekombinationsrate für diese Konfigurationen der Hyperparameter besonders gut ausfallen, da

hier prozentual mehr ausgeführte Rekombinationen zu einer Verbesserung der Fitness führen.

4.1.5 Rohdatenanalyse: Zusammenfassung

Zuerst sollen die Ergebnisse der Hyperparameteranalyse von Parity und Keijzer miteinander verglichen werden, um Zusammenhänge zu erschließen und einzelne Beobachtungen validieren zu können. Die folgenden Abschnitte beziehen sich dabei auf die jeweiligen Spalten der Tabellen A.1 und A.3.
Rechenknoten: Bei Parity kann beobachtet werden, dass die CGP-Konfiguration ohne Rekombination am wenigsten Rechenknoten aufweist. Außerdem kann der Trend beobachtet werden, dass komplexere Rekombinationsalgorithmen durchschnittlich weniger Rechenknoten zugeschrieben bekommen als einfachere Algorithmen. Dabei ist die Streuung um den Mittelwert höher, umso komplexer die Rekombinationsalgorithmen sind. Dies könnte darauf hindeuten, dass eine größere Anzahl an Rechenknoten zu einer geringeren Streuung der Rechenknotenanzahl pro Rekombinationsart führt. Im Vergleich dazu ergeben die Ergebnisse von Keijzer ein völlig anderes Bild: gegensätzlich zu Parity hat die CGP-Konfiguration ohne Rekombination die höchste Anzahl an Rechenknoten im Vergleich zu den Mittelwerten über die unterschiedlichen Rekombinationsalgorithmen. Den bei Parity beobachteten Trend, dass komplexere Rekombinationen eine niedrigere Anzahl an Rechenknoten erfordern könnten, kann bei Keijzer nicht bestätigt werden. Hier erfordert die Uniform-Rekombination die höchste durchschnittliche Anzahl an Rechenknoten. Ebenso ergibt sich für Keijzer ein anderes Bild als bei Parity, wenn die Streuung der Rechenknotenanzahl betrachtet wird: Die Two-Point Rekombination weist zwar die geringste Anzahl an Rechenknoten auf, zeichnet sich aber gleichzeitig durch die niedrigste Streuung aus. Zusammenfassend lässt sich für die Anzahl der Rechenknoten sagen, dass Parity und Keijzer vollkommen unterschiedliche Ergebnisse liefern. Demnach können keine Aussagen getroffen werden, wie die Anzahl der Rechenknoten mit den unterschiedlichen CGP-Konfigurationen zusammenhängen. Die Möglichkeit besteht ebenfalls, dass überhaupt kein Zusammenhang zwischen diesen beiden Komponenten besteht. Durch weitere Tests könnte dies näher betrachtet werden.
Populationsgröße: Bei Betrachtung der Populationsgröße kann für Parity zusammengefasst werden, dass die CGP-Konfiguration ohne Rekombination die deutlich kleinste Populationsgröße benötigt hat. Für Keijzer konnte diese Beobachtung nicht gemacht werden. Umgekehrt konnte bei Parity nicht bestätigt werden, dass Two-Point und Uniform Rekombination mit linear fallender Rekombinationsrate ohne Offset einen geringeren λ-Wert

aufweisen. Zusammenfassend lässt sich also auch für die Populationsgröße kein Zusammenhang zu den unterschiedlichen Rekombinationsarten herstellen.
Rekombinationsraten: Bei den Ergebnissen der Hyperparameteroptimierung vom Parity-Testszenario werden von konstanten Rekombinationsalgorithmen besonders hohe und besonders niedrige Rekombinationsraten vermieden. Für die linear fallende Rate und die Rekombinationsrate mit der One-Fifth Regel wird nahezu der gesamte Definitionsbereich genutzt. Wieder im Gegensatz dazu stehen die Ergebnisse des Keijzer-Testszenarios. Hier werden für nahezu alle CGP-Konfigurationen sehr hohe oder niedrige Werte für die Rekombinationsrate verwendet. Somit kann durch die Ergebnisse der Hyperparameteranalyse nicht bewertet werden, welche Bereiche der Rekombinationsrate besonders effektiv sind.
Offset: Bei Parity wird der geringste Offset bei der Two-Point Rekombination und der höchste Offset bei der Uniform Rekombination verwendet. Werden bei Keijzer die originalen Ergebnisse verwendet, bei denen ein Offset von 0 zugelassen wird, kann dieses Verhalten ebenfalls beobachtet werden. Dies könnte bedeuten, dass die Two-Point Rekombination effizienter ist als die Uniform Rekombination, da bei ersterem weniger Rekombinationsschritte ausgesetzt werden. Die für den Offset neu eingefügten Werte, falls ein Offset gleich 0 bestimmt wird, sind bei Parity ähnlich zum allgemeinen Mittelwert des Offsets. Ein anderes Verhalten kann für das Keijzer-Testszenario beobachtet werden, bei dem die zweitbesten Parametersätze sehr hohe Offset-Werte verwenden. Dies ergibt also einen extrem hohen Sprung im Offset zwischen erstbestem und zweitbestem Parametersatz. Das ist ein Indiz dafür, dass die Güte eines CGPs nicht kausal mit der Höhe des Offsets zusammenhängt.
Zu beachten ist, dass die Ergebnisse der Hyperparameteranalyse kritisch begutachtet werden müssen. Um die Ergebnisse und die daraus abgeleiteten Aussagen zu validieren ist es sinnvoll eine umfangreichere Hyperparameteranalyse auszuführen. Diese sollte mit Hilfe von mehr Rechenkapazität ausgeführt werden, damit mehr Hyperparameter getestet werden können. Außerdem kann die Anzahl an ausgeführten CGP-Trainings pro Bewertungsschritt in der Hyperparameteranalyse höher gesetzt werden. Durch diese Schritte könnten sich die Ergebnisse von Parity und Keijzer aneinander angleichen, wodurch bessere Bewertungen stattfinden könnten. Andernfalls könnte die Aussage gestärkt werden, dass die CGP-Konfigurationen keinen Zusammenhang zur Auswahl einzelner Hyperparameter aufweisen.

Folgend sollen die Rohdatenanalysen der vier Testszenarien Parity, Keijzer, Encode und Koza zusammengefasst und miteinander verglichen werden.
Für alle Testszenarien wurde festgestellt, dass deutlich mehr Mutationen zu einer Verbesserung der Fitness geführt haben als es für Rekombinationen der Fall war. Dies kann damit

zusammenhängen, dass die Rekombinationserfolge bei allen Testszenarien in den frühen Testphasen stattgefunden haben. Diese Beobachtung wird bei Parity und Keijzer dadurch gestützt, dass die Rekombination kein einziges mal die Fitness verbessern konnte, wenn ein Offset eingesetzt wurde. Für die Testszenarien Keijzer und Koza wurden mehr Rekombinationserfolge aufgezeichnet als für die booleschen Probleme. Das kann damit zusammenhängen, dass die Fitness in booleschen Problemen einen diskreten Wertebereich aufweist. Für Tests aus der symbolischen Regression ergeben sich damit mehr Möglichkeiten die Fitness zu verbessern, indem die Verbesserungen kleinschrittiger ablaufen können. Zu beachten ist, dass die Bewertungen in diesem Abschnitt 4.1 quantitativ sind und nicht qualitativ. Demnach wird nur betrachtet wie viele Rekombinationen und Mutationen die Fitness verbessert haben. Dabei lässt sich keine Aussage darüber treffen, wie groß die Rekombinations- und Mutationserfolge ausfallen. Dies sollte in weiteren Tests näher untersucht werden, um näher beurteilen zu können, wie gut die Rekombination abschneidet.
Im Koza-Testszenario treten vereinzelt CGP-Konfigurationen mit konstanter Rekombinationsrate auf, in denen die erfolgreichen Rekombinationen etwas später stattfinden. Diese leicht erhöhte Iterationenzahl hat allerdings keinen Zusammenhang zu der Anzahl an Iterationen bis zur Konvergenz. Demnach ist es unwahrscheinlich, dass die Rekombinationserfolge nur später auftreten, weil das Training an sich länger lief. Ein Zusammenhang zwischen den späteren Rekombinationserfolgen und der konstanten Rekombinationsrate kann nicht vollends bestätigt werden und kann in weiteren Tests näher untersucht werden. Falls dieser oder ein ähnlicher Zusammenhang festgestellt werden kann, könnte die Rekombination gezielter verwendet werden. So könnte beispielsweise die linear fallende Rekombinationsrate verwendet werden, die ähnliche Anzahlen einer positiven Rekombination aufweisen, allerdings frühere Rekombinationserfolge bietet. So könnte die Rekombination früher ausgesetzt werden, ohne Erfolge der Rekombination einzubüßen. Allerdings tritt beim Encode-Testszenario so ein Fall mit späteren Rekombinationserfolgen nicht auf. Dies kann aber daran liegen, dass im Encode-Testszenario insgesamt weniger Rekombinationserfolge vermerkt wurden als beim Koza-Testfall. Es ist also empfohlen dieses Verhalten bei weiteren Testfällen mit SR zu prüfen.
Ein weiterer Punkt, der für alle Testszenarien beobachtet wurde ist, dass es Iterationen gibt, in denen die verbesserte Fitness durch die Mutation verloren gegangen ist. Dies kann auftreten, da die Auswahl der Elitisten erst nach der Rekombination ausgeführt wird und somit im Normalfall keine Evaluation nach der Rekombination und vor der Mutation stattfindet. Dieser Missstand könnte dadurch gelöst werden, indem die Fitness bereits nach der Rekombination evaluiert wird und hier bereits Elitisten zwischengespeichert werden. Anschließend könnten die Elitisten nach der Mutation mit ersteren verglichen werden, um so die besten Elitisten zu wählen.
Des weiteren fällt auf, dass die CGP-Konfigurationen mit Rekombination nur gelegentlich

eine schnellere Konvergenz erzielen, obwohl neben der Mutation auch die Rekombination die Fitness verbessert. In den Testfällen Encode und Koza konnte außerdem festgestellt werden, dass die Spalten, die die Rekombination beschreiben keine zusammenhängenden Muster zu den Ergebnis-Spalten aufweisen. In den Hyperparameteranalysen von Parity und Keijzer werden außerdem jeweils 7 aus 9 CGP-Konfigurationen mit einem Offset versehen und so der gesamte Rekombinationserfolg übergangen. Für den Keijzer-Testfall konnte auch beobachtet werden, dass diejenigen CGP-Konfigurationen mit Offset weniger Iterationen bei konvergierenden Durchgängen brauchten als die jeweiligen gleichen CGP-Konfigurationen ohne Offset. Dies könnte ein Indiz dafür sein, dass nicht nur die Mutation die Rekombination negativ beeinflusst, sondern auch umgekehrt die Rekombination den Erfolg der Mutation verringert. Eine Lösungsmöglichkeit wäre womöglich eine Spaltung von Rekombination und Mutation, wie die bereits für GP verwendet wird [ES15]. Dabei würden die beiden genetischen Operatoren nicht nacheinander ablaufen, sondern gleichzeitig. Dafür wird die Population gespalten und ein Teil des Nachwuchs wird durch Rekombination und der andere Teil durch Mutation erzeugt. So würden gleichzeitig die großen Veränderungen der Chromosomen eingebracht werden, die die Rekombination erzeugt, aber auch die kleineren Anpassungen, die aus der Mutation herrühren. So könnte der Lösungsraum gleichzeitig groß- und kleinschrittig abgesucht werden.
Als letzte wichtige Erkenntnis aus diesem Abschnitt kann zusammengefasst werden, dass die Uniform Rekombination deutlich mehr positive Rekombinationen hervorgebracht hat als die anderen Rekombinationsarten. Während die konstante und linear fallende Rekombination ähnlich viele Rekombinationserfolge erzielt haben, hat die Rekombination mit der One-Fifth Regel in deutlich weniger Iterationen die Fitness verbessert. Demnach hat die Uniform Rekombination mit konstanter oder linear fallender Rekombinationsrate das größte Potential die Rekombination an sich effizienter zu gestalten. Interessant ist die Beobachtung, dass nicht nur höhere Rekombinationsraten zu mehr positiven Rekombinationen geführt haben, obwohl eine höhere Rekombinationsrate die Wahrscheinlichkeit für eine effiziente Rekombination steigert. Im Keijzer-Testszenario konnte die konstante Rekombinationsrate von 0,8 mehr positiver Rekombinationen erzielen als die linear fallende Rekombinationsrate, die mit 0,9 initialisiert wird. Auch bei den Testfällen Encode und Koza konnten in den Plots zu positiven Rekombinationen über Rekombinationsraten lokale Maxima gefunden werden. Bei diesen wurden Hochpunkte für positive Rekombinationen gefunden, die bei relativ geringen Rekombinationsraten entstanden sind. Diese Rekombinationsraten könnten weiter in anderen Testszenarien untersucht werden, um Muster zu erkennen und gegebenenfalls eine ideale Rekombinationsrate zu finden oder mehr Verständnis aufzubauen.
Die Ergebnisse in diesem Abschnitt zeigen, dass im Gebrauch der Rekombinationsalgorithmen mehr Potential steckt als mit dem CGP-Aufbau aus Abbildung 2.1 genutzt wird.

Der in diesem Abschnitt erläuterte Ausblick kann verwendet werden, um weitere Erkenntnisse zu sammeln und die Rekombination im allgemeinen weiter zu fördern. Trotzdem, dass das Potential nicht voll genutzt wird, konnten die bisherigen Ergebnisse Grund zu der Vermutung geben, dass bereits mit diesem Stand die Effizienz eines CGPs gesteigert werden kann, wenn die Rekombination richtig gewählt und parametriert wird. Diese Ergebnisse wurden für Encode und Koza auch erzielt, obwohl nur die Hyperparameter der CGP-Konfigurationen ohne Rekombination optimiert wurden. Der folgende Abschnitt 4.2 wertet den Ist-Stand der Standard-Rekombination in CGPs weiter aus.

4.2 Ergebnisse Bayes'sche Analyse

In diesem Abschnitt werden die Ergebnisse der bayes'schen Analyse aufgezeigt und bewertet. Dafür werden die Tabellen aus dem Anhang A.2 verwendet. Aus den Ergebnissen der Rohdatenanalyse von Encode (Abschnitt 4.1.3) und Koza (Abschnitt 4.1.4) geht hervor, dass für Encode die Fitness bei der 10000. Iteration bewertet wird und für Koza die Anzahl der Iterationen bis zur Konvergenz. Für die restlichen Testszenarien Parity und Keijzer wird ebenfalls die Anzahl der Iterationen bis zur Konvergenz analysiert. Ziel der bayes'schen Analyse ist es herauszufinden, ob Rekombination die Effizienz von CGPs verbessern. Aus Abschnitt 4.1.5 geht hervor, dass Rekombination ein vermutlich größeres Potential besitzt als mit Standard-Vorgehensweisen genutzt wird. Die bayes'sche Analyse wird verwendet, um die Effizienz der Standard-Vorgehensweisen mit und ohne Rekombination abzugleichen.
Folgend werden die Spalten der Tabellen näher erläutert:

- **CGP-Konfiguration:** Es handelt sich um die jeweils für das Training verwendete CGP-Konfiguration. Für Parity und Keijzer sind das die durch die Hyperparameteranalyse optimierten CGP-Konfigurationen mit und ohne Offset. Bei Encode und Koza wird wie in den Abschnitten 4.1.3 und 4.1.4 beschrieben, kein Offset betrachtet. Dafür werden jeweils verschiedene Werte der Rekombinationsraten analysiert.
- **HPDI (Iter. / Fitn.):** Diese Spalte gibt den HPDI von 95% an. „Iter." und „Fitn." gibt dabei an, ob dabei die Fitness oder die Anzahl der Iterationen bewertet werden. Die HPDI-Berechnung kann ebenfalls mit zensierten Werten umgehen. Dies wird für die Analyse der Iterationen bis zur Konvergenz relevant. Dabei wird dem Modell mitgegeben wie viele CGP-Modelle nicht konvergiert sind und bei ungefähr welcher Iteration alle Durchläufe das Stopp-Kriterium erfüllen müssten. Diese Bewertung basiert demnach unter anderem auf Schätzwerten und sind mit Vorsicht zu bewerten. Die

Berechnung der Schätzwerte basiert auf der Information wie hoch die Fitness zur Abbruchiteration ist. Dadurch kann die hohe der Fitness zur Abbruchiteration ebenfalls in die Bewertung einfließen. Da für alle bewerteten CGP-Konfigurationen das gleiche Verfahren angewendet wurde, sollten die Ergebnisse miteinander vergleichbar sein.

- **MW:** Diese Spalte gibt den Mittelwert der gefitteten Wahrscheinlichkeitsverteilung an.
- **PL-Platz:** Dabei handelt es sich um die Platzierung der CGP-Konfigurationen mit Hilfe des Plackett-Luce-Modells. Die Platzierung gibt eine Wahrscheinlichkeit an, dass die jeweilige CGP-Konfiguration die besten Ergebnisse liefert. Demnach werden Werte von 0 bis 1 vergeben, die zusammen 100% ergeben. In dem Plackett-Luce-Modell können zensierte Daten nicht betrachtet werden. Deswegen wurden dem Modell die jeweils vorliegenden Ergebnissen zur Abbruchiteration mitgegeben.

4.2.1 Bayes'sche Analyse: Parity

Dieser Abschnitt bezieht sich auf die Tabelle A.11 und bewertet lediglich die Ergebnisse der bayes'schen Analyse von Parity. Jegliche Aussagen können vorerst nicht auf die Allgemeinheit geschlossen werden und werden in Abschnitt 4.2.5 näher bewertet.

Bewertung MW und PL-Plätze Werden die MW der verschiedenen CGP-Modelle betrachtet, wird deutlich, dass nur 2 der 18 CGP-Konfigurationen mit Rekombination schlechter abschneiden als diejenige ohne Rekombination. Das ist ein Zeichen dafür, dass Rekombination durchaus die Effizienz von CGP steigern kann, wenn die Rekombination sinnvoll konfiguriert wird.
Die jeweils drei niedrigsten MW und drei höchsten PL-Plätze sind in der Tabelle A.11 grün markiert, während die drei höchsten MW und drei niedrigsten PL-Plätze rot sind. Zu sehen ist, dass die MW und PL-Plätze nicht genau die gleichen CGP-Konfigurationen als gleich gut oder schlecht bewerten. Dies kann daher kommen, dass in die Berechnung der MW Schätzwerte einfließen, wann ein CGP-Modell konvergiert. Für die Berechnung der PL-Plätze ist das nicht der Fall. Die MW geben demnach mehr Eindrücke darauf wie nah die CGP-Modelle an der Konvergenz sind, wenn sie das Stopp-Kriterium noch nicht erfüllt haben. Allerdings handelt es sich dabei wie bereits erwähnt um einen Schätzwert, der durchaus mit Vorsicht betrachtet werden muss. Die Tabelle A.2 zeigt jedoch, dass Trends ob eine CGP-Konfiguration gute oder schlechte Ergebnisse hervorbringt, durch MW und PL-Plätze

gleich bewertet werden. Bei den Unterschieden handelt es sich nur um einzelne Platzierungen der CGP-Konfigurationen. Durch die gemeinsame Betrachtung von MW und PL-Plätze kann demnach ein gutes Bild davon gegeben werden, wie effizient einzelne CGP-Modelle trainieren. Werden die besten und schlechtesten MW und PL-Plätze in der Tabelle betrachtet, wird deutlich, dass die CGP-Konfigurationen mit der One-Point Rekombination deutlich bessere Ergebnisse liefert als die anderen Rekombinationsarten. Aus Abschnitt 4.1.1 ist bekannt, dass für die One-Point Rekombination deutlich weniger Rekombinationserfolge auftreten als bei den anderen Rekombinationsarten. Deswegen kann davon ausgegangen werden, dass bei diesen CGP-Konfigurationen die Rekombination weniger Einfluss auf das Training hat. Werden die schlechtesten Verfahren nach MW und PL-Plätzen mit Rekombination ohne Offset aus Tabelle A.11 mit der Tabelle A.2 verglichen kann beobachtet werden, dass der Anteil der negativen Mutationen pro positive Rekombinationen relativ hoch ist. Das kann dafür sprechen, dass Rekombination mit dem Standard-CGP-Verfahren mit weniger einflussreicher Rekombination besser funktioniert, da Rekombination und Mutation nicht ausreichend aufeinander abgestimmt sind und damit einflussreichere Rekombination (mit mehr möglichen positiven Rekombinationsschritten) weniger gute Ergebnisse liefern. Wenn sich dieser Ansatz bewahrheitet, könnte demnach durch eine Anpassung des CGP-Verfahrens Rekombination sinnvoller eingesetzt werden, sodass einflussreichere Rekombinationsarten wie die Uniform Rekombination einen größeren Mehrwert generieren.
Des Weiteren werden die Ergebnisse jeweils bezüglich MW und PL-Plätzen von Verfahren mit Offset mit den Ergebnissen der jeweils gleichen CGP-Konfigurationen ohne Offset verglichen. Dabei weisen jeweils sechs aus neun CGP-Konfigurationen ohne Offset bessere Ergebnisse auf als diejenigen mit Offset. Dies spricht dafür, dass ein Rekombinationsoffset der Effizienz eines CGP-Modells schadet. Ein Grund dafür könnte sein, dass damit die positiven Einflüsse der Rekombination gestrichen werden, während die nachteiligen Effekte des Zusammenspiels von Rekombination und Mutation weiterhin bestehen bleiben.

Bewertung HPDI Analysiert werden nun die HPDI-Werte aus Tabelle A.11. Dabei werden die HPDI-Werte mit dem kleinsten Umfang grün markiert und die mit dem größten rot. Diese werden verwendet, um die Ergebnisse stichprobenartig zu bewerten. Es fällt auf, dass die One-Point Rekombination die besten drei Werte beinhaltet, wenn der HPDI-Umfang bewertet wird. Wenn aus allen HPDI-Werten die unteren Grenzen betrachtet und verglichen werden, kann außerdem beobachtet werden, dass die drei kleinsten unteren Grenzen Teil der grün markierten HPDIs sind. Umgekehrt lässt sich beobachten, dass besonders große untere Grenzen zu großen Umfängen der HPDI führen. Bei den drei größten unteren Grenzen ist dies in einem Fall keine rot markierte HPDI, allerdings handelt es sich dabei

um einen ähnlich großen HPDI-Umfang, wodurch ein Trend erkennbar wird. Dieses Verhalten weist darauf hin, dass die frühe Konvergenz der frühesten erfolgreichen Trainings ein Anzeichen dafür ist, dass die meisten CGP-Modelle relativ früh konvergieren. Zu beachten ist, dass es sich hierbei nicht um Ausreißer handelt, da diese in der 95%-igen HPDI nicht erfasst werden. Dies kann dafür sprechen, dass frühe Trainingserfolge besonders wichtig sind, wenn eine verlässliche CGP-Konfiguration generiert werden soll, deren Modelle in einer relativ kleinen Iterationsspanne konvergieren sollen. Da Rekombination laut Abschnitt 4.1.5 besonders in früheren Iterationen ein hohes Potential aufweist die Fitness zu verbessern, ist es demnach sinnvoll die Rekombination und deren Einsatz effizienter zu gestalten.

4.2.2 Bayes'sche Analyse: Keijzer

Dieser Abschnitt beurteilt die bayes'sche Analyse des Keijzer-Testszenarios und bezieht sich dabei auf die Tabelle A.12. Alle Aussagen und Bewertungen der Ergebnisse beziehen sich demnach nur auf den Keijzer-Testfall und können nicht direkt auf die Allgemeinheit bezogen werden. In Abschnitt 4.2.5 werden die Ergebnisse aller Testszenarien zusammengefasst und bewertet.

Bewertung MW und PL-Plätze Im ersten Schritt wird der MW aller CGP-Konfigurationen bewertet. Bei 12 aus 18 Verfahren mit Rekombination sind die MW kleiner und damit besser als die der CGP-Konfiguration ohne Rekombination. Bei Betrachtung der Mittelwerte der Spalte MW über die verschiedenen Rekombinationsarten hinweg ergeben sich folgende Durchschnittswerte: One-Point Rekombination braucht durchschnittlich ca. 8200 Iterationen, Two-Point ca. 8670 und Uniform ungefähr 8650. Damit liegen alle Mittelwerte unterhalb von ca. 9552 Iterationen, die das CGP-Modell ohne Rekombination benötigt. Das heißt, dass es sich beim Standard-CGP-Vorgehen im Durchschnitt lohnt Rekombination anzuwenden.
Die jeweils drei besten MW und PL-Plätze sind in der Tabelle grün markiert, während die schlechtesten rot sind. Zu beobachten ist, dass die jeweils besten und schlechtesten Werte der beiden Spalten nicht immer übereinstimmen. Dies ist der Fall, da bei den PL-Plätzen keine Schätzung darüber abgegeben wird, wann nicht-konvergente CGP-Modelle das Stopp-Kriterium erfüllen. Hier wird nur die maximale Iterationsgrenze (6500 Iterationen) angegeben, falls ein CGP-Modell nicht fertig trainiert wurde. Demnach geben die Werte der Spalte MW besser wieder, wie hoch die Fitness beim Abbruch des CGP-Trainings war. Allerdings besteht bei der Schätzung der Iterationen ein gewisser Unsicherheitsfaktor

hinsichtlich der Entwicklung der Ergebnisse des CGP-Modelle. Deswegen werden beide Spalten in die Bewertung einbezogen. Zu sehen ist, dass die schlechtesten Ergebnisse von MW und PL-Platz in den Rekombinationsraten Two-Point und Uniform auftreten. Allerdings treten ebenfalls die besten Ergebnisse vor allem in diesen Rekombinationsarten auf. Dies zeigt auf, wie wichtig es ist nicht nur die Rekombinationsart richtig zu wählen, sondern auch die Rekombinationsraten-Arten und deren Parametrierungen abzuwägen.
Wird ein näherer Blick auf diejenigen CGP-Konfigurationen aus Tabelle A.4 mit Rekombination ohne Offset geworfen, bei denen die Spalten MW und PL-Platz aus Tabelle A.12 besonders gut oder besonders schlecht ausfallen, fällt auf, dass die schlechten Ergebnisse vor allem bei denjenigen CGP-Konfigurationen auftreten, in denen viele negative Mutationen auftreten (Two-Point Clegg und Uniform Konstant). Dies kann bedeuten, dass in diesen CGP-Modellen die Rekombination und Mutation häufiger gegeneinander gearbeitet haben, anstatt gemeinsam bessere Ergebnisse zu liefern. Die besseren Ergebnisse nach MW und PL-Plätzen treten in denjenigen Konfigurationen auf, in denen die positiven Rekombinationserfolge entweder relativ selten auftreten oder in denen die negativen Mutationen anteilig weniger auftreten (One-Point Konstant und Uniform One-Fifth). Das könnten diejenigen CGP-Konfigurationen sein, in denen die Rekombination weniger einflussreich ist oder sich die Rekombination und Mutation weniger negativ beeinflussen. Da bei der CGP-Konfiguration „Uniform One-Fifth ohne Offset“ in Tabelle A.4 eine sehr frühe Iterationenzahl für positive Rekombination erfasst wird, kann es sein, dass die Rekombination anschließend mit Hilfe der One-Fifth Regel so reduziert wurde, sodass die Mutation nicht mehr negativ beeinflusst werden konnte. Dieses Szenario kann allerdings mit den vorliegenden Daten nicht untersucht und demnach nicht evaluiert werden.
Des Weiteren werden alle Ergebnisse aus MW und PL-Plätzen derjenigen CGP-Konfigurationen mit Offset mit den Ergebnissen der gleichen Konfigurationen ohne Offset verglichen. Für den MW sind die Ergebnisse von sechs aus neun Konfigurationen ohne Offset besser. Werden die PL-Plätze bewertet sind es nur fünf aus neun. Die Ergebnisse zeigen, dass es im Durchschnitt besser ist die Rekombination nicht auszusetzen. Trotzdem zeigen diese Ergebnisse auch, dass das Potential von Rekombination noch weiter ausgebaut werden muss, um eine verlässliche Lösung anzubieten, bei der kein Offset mehr benötigt wird.

Bewertung HPDI Für die nähere Betrachtung der HPDI-Werte aus Tabelle A.12 werden die drei HPDI-Ranges mit dem geringsten Umfang grün, während diejenigen mit dem größten Umfang rot markiert werden. Es fällt auf, dass die Uniform Rekombination relativ kleine HPDI-Ranges aufweist, bis auf die zwei sehr großen Umfänge. Die One-Point Rekombination weist zwei sehr geringe HPDI-Umfänge auf, während die restlichen rela-

tiv hoch ausfallen. In der näheren Analyse wird kein Grund gefunden, wieso diese CGP-Konfigurationen dermaßen niedrige oder hohe HPDI-Ranges aufweisen. Werden die drei höchsten unteren Grenzen der HPDI-Werte näher betrachtet, fällt auf, dass diese zu den HPDI-Werten mit den größten Umfängen gehören. Wenn umgekehrt die niedrigsten unteren Grenzen gesammelt werden, passen zwei von drei zu den HPDI-Werten mit den geringsten Umfängen, während der dritte einen ähnlich kleinen HPDI-Umfang aufzeigt. Dies zeigt, dass wie bereits in 4.2.1 erläutert, frühe Konvergenzen zu einem kleineren HPDI-Umfang führen. So ist es zum Beispiel unwahrscheinlich, dass ein CGP bei mehreren Durchläufen relativ lange bis zur Konvergenz braucht aber dafür alle Durchläufe ähnlich lange benötigen. Demnach ist es wichtig frühe Iterationen sinnvoll für das Training zu nutzen, um eine verlässliche Konvergenzrate zu erzielen. Da Abschnitt 4.1.5 zufolge die Rekombination besonders in frühen Iterationen Potential aufzeigt die Fitness zu verbessern, ist es umso wichtiger die Rekombination effizienter zu gestalten.

4.2.3 Bayes'sche Analyse: Encode

Dieser Abschnitt befasst sich mit der Evaluierung der bayes'schen Analyse von Encode und bezieht sich auf die Tabellen A.13, A.14 und A.15. In der bayes'schen Analyse von Encode wird bewertet wie hoch die Fitness-Werte der CGP-Modelle bei der 10000. Iteration sind. Zu beachten ist, dass sich die Aussagen innerhalb dieses Abschnittes nur auf den Encode-Testfall beziehen und vorerst keine allgemeinen Schlussfolgerungen zulassen. In Abschnitt 4.2.5 werden alle Ergebnisse der bayes'schen Analysen zusammengetragen und verglichen.
Innerhalb der Tabellen A.13, A.14 und A.15 wurden die besten drei Ergebnisse pro Spalte grün markiert und die schlechtesten rot. Zu beachten ist, dass die CGP-Konfiguration ohne Rekombination durch eine Hyperparameteranalyse optimiert wurde. Die restlichen CGP-Konfigurationen haben die Hyperparameter übernommen und die Rekombinationsrate belegt.

Bewertung MW und PL-Plätze Werden die MW der CGP-Konfigurationen verglichen, kann beobachtet werden, dass für die One-Point und Uniform Rekombination jeweils 20 der 22 Verfahren mit Rekombination eine bessere Fitness aufweisen als die CGP-Konfiguration ohne Rekombination. Für die Two-Point Rekombination sind alle Verfahren schlechter, wenn Rekombination ausgeführt wird. Werden die Mittelwerte der Spalte MW über jede Rekombinationsraten-Art gebildet, ergibt sich folgende Tabelle 4.1:

	Konstant	Clegg	One-Fifth
One-Point Rekombination	0,03967	0,04082	0,03803
Two-Point Rekombination	0,05690	0,05618	0,05387
Uniform Rekombination	0,03858	0,03582	0,03748

Tabelle 4.1: Encode: Durchschnittliche Fitness nach 10000. Iteration

Es ist zu erkennen, dass für den Encode-Testfall die Two-Point Rekombination deutlich schlechtere Ergebnisse liefert als die anderen Rekombinationsarten. Für jede Rekombinationsraten-Art bietet die Uniform Rekombination die besten Ergebnisse. Werden die Rekombinationsraten-Arten bewertet, weist die One-Fifth Rekombinationsrate für die One-Point Rekombination das beste Ergebnis auf, gefolgt von der konstanten Rekombinationsrate und zuletzt der linear fallenden Rekombinationsrate. Die Two-Point Rekombination weist für alle Rekombinationsraten sehr ähnliche Werte auf. Die Reihenfolge derer ist: One-Fifth, linear fallend, konstant. Bei der Uniform Rekombination ist die linear fallende Rekombinationsrate die beste. Es folgt die One-Fifth Rekombinationsrate und anschließend die konstante Rekombinationsrate. Es lässt sich also keine einheitliche Reihenfolge bilden, die die Effizienz der Rekombinationsraten-Arten abbildet. Ebenfalls lassen sich durch die Auswertung der Tabellen A.13, A.14 und A.15 keine Muster finden wie die verschiedenen Rekombinationsraten-Arten am besten zu parametrieren sind. Dementsprechend wäre es für das Standard-CGP-Verfahren empfehlenswert verschiedene Rekombinationsraten-Arten innerhalb der Hyperparameteranalyse zu testen.

Bewertung HPDI Für die Testszenarien Parity und Keijzer wurde in den Evaluationen der bayes'schen Analyse in den Abschnitten 4.2.1 und 4.2.2 herausgefunden, dass kleine untere Grenzen in den HPDI-Werten sehr wahrscheinlich eine kleine Intervallgröße hervorbringen. Bei diesen bayes'schen Analysen wurde allerdings die Anzahl der Iteration bis zur Konvergenz betrachtet, im Gegensatz zu Encode, bei dem die Fitness zu einer bestimmten Iteration betrachtet wird. Demnach soll diese Beobachtung für den Fall der Fitness-Bewertung auch betrachtet werden. Allerdings fällt bei der Betrachtung der Tabelle A.14 auf, dass diese Beobachtung für die Fitness-Bewertung nicht zutrifft: Ein Gegenbeispiel ergeben die drei fett gedruckten HPDI-Werte dieser Tabelle, die alle drei die kleinsten untersten Grenzen abbilden. Diese kleinen unteren Grenzen führen allerdings zu relativ großen HPDI-Ranges. Dementsprechend kann keine Aussage darüber getroffen werden, wie gut die Fitnesswerte der anderen Durchläufe zu einer bestimmten Iteration sind, wenn bereits bekannt ist, dass einige Fitness-Werte sehr klein sind.

4.2.4 Bayes'sche Analyse: Koza

Dieser Abschnitt evaluiert die Ergebnisse der bayes'schen Analyse des Koza-Testszenarios und bezieht sich dabei auf die Tabellen A.16, A.17 und A.18. Die bayes'sche Analyse des Koza-Testfalls bewertet wie in Abschnitt 4.1.4 beschrieben die Anzahl der Iterationen bis zur Konvergenz. Alle Aussagen und Ergebnisse in diesem Abschnitt beziehen sich ausschließlich auf die Auswertung des Koza-Testfalls und können nicht als allgemeine Aussagen gewertet werden. In Abschnitt 4.2.5 werden die Evaluationsergebnisse aller Testszenarien für die bayes'sche Analyse zusammengefasst und bewertet.

Bewertung MW und PL-Plätze Wird die Spalte MW aus den jeweils genannten Tabellen betrachtet und einzeln miteinander verglichen, werden folgende Ergebnisse sichtbar: für die One-Point Rekombination sind keine Verfahren mit Rekombination besser als das ohne Rekombination, für die Two-Point Rekombination sind fünf aus 22 CGP-Modelle mit Rekombination besser, für die Uniform Rekombination sind es lediglich drei Verfahren. Dies zeigt, dass für den Koza-Testfall Verfahren mit eingefügtem Rekombinationsschritt deutlich schlechter abschneiden als ohne Rekombination. Dabei ist allerdings zu beachten, dass die CGP-Konfiguration ohne Rekombination mit Hilfe einer Hyperparameteranalyse optimiert wurde, während die Verfahren mit Rekombination lediglich diese Parametrierung übernommen und die Rekombinationsrate ergänzt haben. Demnach kann keine Aussage darüber getroffen werden wie effizient die Rekombination tatsächlich ist. Da für Parity (Abschnitt 4.2.1) und Keijzer (Abschnitt 4.2.2) deutlich bessere Ergebnisse beobachtet werden konnten, könnte es sein, dass eine Hyperparameteranalyse für alle verwendeten CGP-Verfahren die Ergebnisse verbessern könnte. Dies führt zu der Annahme, dass bei einer Hyperparameteroptimierung alle Parameter einer CGP-Konfiguration auf einmal betrachtet werden müssen und somit keine Hyperparameter einer anderen Optimierung verwendet werden können, um lediglich einen einzelnen Parameter zu optimieren. Diese Annahme müsste in weiteren Tests geprüft werden. Es kann allerdings auch sein, dass der Koza-Testfall deutlich schlechtere Werte für Verfahren mit Rekombination aufweist, da in diesem Testfall deutlich mehr Iterationen bis zur Konvergenz benötigt werden als bei Parity und Keijzer. In diesen weiteren Iterationen werden nach Abschnitt 4.1.5 Mutationsschritte relevanter als Rekombinationsschritte für die Effizienz des Trainings. Demnach ist es auch möglich, dass die Rekombinationen die Mutation in den späteren Iterationen so negativ beeinflussen, dass das Verfahren ohne Rekombination besser abschneidet.
Die folgende Tabelle 4.2 zeigt die ungefähren Mittelwerte aller Rekombinationsarten pro Rekombinationsraten-Art.

	Konstant	Clegg	One-Fifth
One-Point Rekombination	22164	19272	14804
Two-Point Rekombination	9510	8509	8444
Uniform Rekombination	12599	7015	11273

Tabelle 4.2: Koza: Durchschnittliche Iterationen bei Konvergenz

Zu beobachten ist, dass die One-Point Rekombination für alle Rekombinationsraten-Arten besonders schlechte Werte hervorbringt. Die Two-Point Rekombination ergibt im Vergleich zu den anderen Rekombinationsarten relativ gute Ergebnisse für alle Rekombinationsraten-Arten. Für die Uniform Rekombination führt nur die linear fallende Rekombinationsrate zu guten Ergebnissen. Diese sind deutlich besser als die Ergebnisse der Two-Point Rekombination. Dieser Erfolg der linear fallenden Rekombinationsrate für die Uniform Rekombination könnte darauf zurückgeführt werden, dass die Start-Rekombinationsrate mit 0,9 hoch gewählt wurde. Allerdings sind die Ergebnisse der konstanten Rekombinationsrate für hohe Rekombinationsraten aus Tabelle A.18 besonders schlecht. Dementsprechend ist das konsequente Verringern der Rekombinationsrate in diesem Fall der relevante Unterschied zu den anderen Rekombinationsraten-Arten. Selbst für eine kleine Delta-Rekombinationsrate von 0,0005 für linear fallende Rekombinationen wird die Rekombination nach spätestens 1800 Iterationen vollständig ausgesetzt. Dies könnte der Grund sein, warum die linear fallende Rekombination für einfachere Testfälle wie Parity oder Keijzer nicht deutlich effizienter als die anderen Rekombinationsraten-Arten abschneidet. Denn diese Testfälle brauchen deutlich weniger Iterationen als der Koza-Testfall bis zur Konvergenz, wodurch die Rekombination womöglich nicht früh genug abgestellt wurde. Werden alle Werte der linear fallenden Rekombination aus Tabelle 4.2 betrachtet, fällt auf, dass die Mittelwerte für komplexere Rekombinationsarten besser ausfallen. Dies könnte daran liegen, dass komplexere Rekombinationsarten laut Abschnitt 4.1.4 deutlich mehr Rekombinationserfolge vermerken können als die weniger komplexen Rekombinationsarten. Wenn die Rekombination also rechtzeitig vollständig unterbrochen wird, könnten besonders komplexere Rekombinationsarten einen höheren Erfolg einbringen, da ihr erhöhter Einfluss auf das Trainingsverhalten in den ersten Trainingsphasen sinnvoll genutzt werden könnte ohne anschließend negative Effekte auf die Mutation zu erzeugen.
Die Betrachtung der PL-Plätze aus den Tabellen A.16, A.17 und A.18 hat keine weiteren Erkenntnisse oder Muster hervorgebracht.

Bewertung HPDI Für die Bewertung der HPDI-Werte wurden jeweils die kleinsten unteren und oberen Grenzen grün markiert, während die größten unteren und oberen Gren-

zen rot sind. Zu beobachten ist, dass in den meisten Fällen die jeweils kleinsten und Größten Grenzen aufeinander treffen. In den Ausnahmen werden ähnlich kleine/große Werte markiert. Für die markieren HPDI-Werte wurden die HPDI-Umfänge berechnet, wobei sich herausgestellt hat, dass die kleinen Grenzen kleine Ranges hervorbringen und die großen Grenzen große Ranges aufweisen. Diese Bewertung ergibt das gleiche Ergebnis wie die Abschnitte 4.2.1 und 4.2.2, obwohl eine anderen Herangehensweise genutzt wurde, um dieses hervorzubringen: Frühe Konvergenzen innerhalb der unteren Grenze des HPDI führen zu kleineren Streuungen der Ergebnisse. Demnach ist es wichtig bereits in den frühen Iterationen gute Trainingsergebnisse zu erzielen, um eine große Streuung der Ergebnisse vorzubeugen. Deswegen ist es sinnvoll die Rekombination und deren Einsatz weiter zu optimieren, die nach Abschnitt 4.1.5 in den ersten Iterationen das größte Potential besitzt einen Mehrwert zu liefern. Des Weiteren können durch diese Erkenntnis aus den HPDI-Werten die Streuungen der Ergebnisse frühestmöglich eingeschätzt werden, wenn noch nicht alle CGP-Modelle trainiert wurden. Wenn für CGP-Konfigurationen bereits einige Durchläufe deutlich früher konvergiert sind als in anderen, ist es wahrscheinlich, dass die folgenden Durchläufe zu einer relativ ähnlichen Iteration konvergieren. Umgekehrt wird es für die folgenden Durchläufe der langsameren CGP-Modelle wahrscheinlicher, dass diese umso später konvergieren.

4.2.5 Bayes'sche Analyse: Zusammenfassung

Dieser Abschnitt vergleicht die Ergebnisse und Aussagen der bayes'schen Analyse aller Testszenarien und fasst die Erkenntnisse zusammen.

Erkenntnisse zum Offset Zuerst werden die Ergebnisse der Analysen aus den Abschnitten 4.2.1 und 4.2.2 betrachtet, um die Verwendung eines Offsets für die Rekombination zu bewerten. Für Parity und Keijzer wurde anhand des MW und PL-Platzes bestimmt, welche jeweils gleichen CGP-Konfigurationen mit oder ohne Offset bessere Ergebnisse geliefert haben. Das Ergebnis zeigt, dass es durchschnittlich besser ist keinen Offset anzuwenden. Vermutlich ist dies darauf zurückzuführen, dass so die positiven (aber auch negativen) Einflüsse der Rekombination zu Beginn des Trainings blockiert werden, während die negativen Folgen zu den späteren Iterationen wieder eintreten. Trotzdem haben ein paar der CGP-Modelle mit Offset bessere Ergebnisse geliefert als ohne Offset. In diesen Fällen überwiegen wahrscheinlich die negativen Einflüsse der Rekombination auf die Mutation in den ersten Iterationen, sodass ein Offset sinnvoll ist. Ein Ziel muss daher al-

so sein, die Rekombination oder die CGP-Trainingsverfahren so zu modifizieren, dass die Rekombination effizienter wird.

Erkenntnisse aus MW und PL-Plätzen Folgend werden die Ergebnisse bezüglich der Spalten „MW“ und „PL-Platz“ ausgewertet. Bei den Testszenarien Parity und Keijzer, bei denen alle CGP-Konfigurationen durch eine Hyperparameteranalyse optimiert wurden, konnte festgestellt werden, dass die Verwendung von Rekombination durchaus Sinn macht und in relativ vielen Fällen zu schnelleren Konvergenzen führt. Dabei war vor allem die One-Point Rekombination effektiver als die anderen Rekombinationsarten. Die Two-Point Rekombination führte zu besonders guten und auch besonders schlechten Ergebnissen. Dies zeigt auf, dass es sinnvoll wäre bei den Hyperparameteroptimierungen auch unterschiedliche Rekombinationsraten-Arten zu testen, da diese den Trainingserfolg stark beeinflussen können. Bei der Untersuchung vom Keijzer-Testfall ist aufgefallen, dass schlechte Ergebnisse vor allem durch diejenigen CGP-Konfigurationen ausgelöst wurden, die viele negative Mutation hervorgerufen haben, also Mutationen die die Fitnessverbesserung der Rekombination teilweise oder vollständig ausgelöscht haben. Dies sind wahrscheinlich diejenigen CGP-Modelle, bei denen die Rekombination und Mutation besonders stark gegeneinander arbeiten. Besonders gute Ergebnisse wurden beobachtet, wenn entweder wenige Rekombinationsschritte zu einer Verbesserung der Fitness geführt haben (positive Rekombinationen) oder anteilig weniger negative Mutationen im Vergleich zu den positiven Rekombinationen festgestellt wurden. Dabei könnte es sich um CGP-Modelle handeln, bei denen die Rekombination entweder weniger Einfluss auf das Training ausübt oder die positiven Einflüsse der Rekombination die negativen Effekte auf die Mutation überwiegen. Diese Beobachtung deckt sich ebenfalls damit, dass die One-Point Rekombination als die durchschnittlich beste Rekombinationsart hervorgegangen ist. Wie in Abschnitt 4.1 zu sehen ist, bringen einfache Rekombinationsarten weniger Trainingserfolge und haben damit einen geringeren Einfluss auf den Trainingserfolg. Dadurch kollidieren die wahrscheinlich weniger oder überhaupt nicht mit den Einflüssen der Mutation auf das Training. Für das Standard-CGP-Verfahren mit Rekombination und anschließender Mutation scheint für weniger komplexe Testszenarien die One-Point Rekombination die effektivste Rekombinationsart zu sein. Allerdings könnten Veränderung des Trainingsverfahrens dazu führen, dass komplexere Rekombinationsarten mit mehr Einfluss auf das Trainingsgeschehen wie die Uniform Rekombination zu noch besseren Ergebnissen führen.
Bei den komplexeren Testszenarien Encode und Koza konnten andere Ergebnisse festgestellt werden. Zu beachten ist, dass in diesen Testfällen nur die CGP-Konfiguration ohne Rekombination mit einer Hyperparameteranalyse optimiert wurden. Die anderen CGP-Modelle haben diese Parameter übernommen und jeweils eine Rekombinationsrate einge-

fügt. Für den Encode-Testfall haben die One-Point und Uniform Rekombination zu sehr guten Ergebnissen geführt, während bei der Two-Point Rekombination alle Tests schlechter ausgefallen sind als bei Tests ohne Rekombination. Da für den Encode-Testfall jedoch nur die Fitness zur 10000. Iteration betrachtet wurde, kann die Effizienz der Rekombination gegenüber des Trainings nicht sinnvoll ausgewertet werden. Dadurch, dass die Mutation in den späteren Iterationen zu mehr Trainingserfolgen führt als die Rekombination, kann es passieren, dass bei der Betrachtung der Iterationen bis zur Konvergenz die CGP-Konfiguration ohne Rekombination bessere Ergebnisse liefert, da die Mutation in diesem Verfahren besser funktionieren könnte. Für den Koza-Testfall haben sehr wenige CGP-Modelle mit Rekombination besser abgeschnitten als diejenigen ohne Rekombination. Einerseits könnten die CGP-Konfigurationen mit Rekombination durch eine Hyperparameteroptimierung verbessert werden. Andererseits kann es auch sein, dass komplexere Testszenarien wie Encode und Koza andere Rekombinationsverfahren benötigen als die einfachen Testfälle. Während die One-Point Rekombination bei den einfachen Testfällen die beste Rekombinationsart zu sein scheint, sind die Ergebnisse der komplexeren Rekombinationsarten bei Koza besser als die der One-Point Rekombination. Für Encode ergibt die Uniform Rekombination für alle Rekombinationsraten-Arten die durchschnittlich besten Ergebnisse. Für den Koza-Testfall führt die Two-Point Rekombination für alle Rekombinationsraten-Arten zu durchschnittlich relativ guten Ergebnissen. Für beide Testfälle (Encode und Koza) werden die durchschnittlich besten Ergebnisse von der Uniform Rekombination mit linear fallender Rekombinationsrate erzeugt. Da für die konstante Rekombinationsrate die meisten Rekombinationsarten durchschnittlich relativ schlechte Ergebnisse liefern, ist es wahrscheinlich dass die komplexen Testszenarien nicht von der hohen Start-Rekombinationsrate, sondern von der Verkleinerung der Rekombinationsrate profitieren. Eine mögliche Ursache dafür, dass komplexere Testszenarien im Vergleich zu den einfachen Testfällen vor allem komplexere Rekombinationsarten mit linear fallender Rekombinationsrate bevorzugen, könnte sein, dass die Testfälle insgesamt mehr Iterationen bis zur Konvergenz benötigen als die einfacheren Testfälle Parity und Keijzer. Bei der bisherigen Belegung der Delta-Rekombinationsrate von 0,0005-0,0055 wird die Rekombination nämlich erst nach 164-1800 Iterationen vollständig abgesetzt. Für die einfachen Testfälle könnte dies ein zu später Zeitpunkt sein, weshalb die komplexeren Rekombinationsarten (wie Uniform Rekombination) noch in relativ späten Iterationen aktiv sind und das Training durch Mutation wahrscheinlich stärker beeinträchtigen als es die One-Point Rekombination tut. Um das volle Potential der Rekombination zu nutzen, ist es also wahrscheinlich für alle Testfälle sinnvoll die Uniform Rekombination zu wählen, wobei die Rekombination nur bis zu einer gewissen Iteration ausgeführt wird. Außerdem wäre ein wichtiger Schritt das Einführen eines neuen CGP-Lernverfahrens, bei dem sich die Rekombination und Mutation nicht mehr gegenseitig negativ beeinflussen können. Eine Möglich-

keit für die Umsetzung könnte sein, Rekombination und Mutation nicht mehr nacheinander auszuführen werden sondern parallel, während beide genetische Operatoren eigene Nachwuchschromosome erzeugen, die anschließend verglichen werden. Aus den beiden Teilpopulationen können anschließend die Elitisten ausgesucht werden.

Erkenntnisse aus HPDI Abschließend werden die Ergebnisse zusammengefasst, die aus der Bewertung der HPDI-Werte folgen. Bei denjenigen Testszenarien, für die die Iterationen bis zur Konvergenz bewertetet wurden (Parity, Keijzer und Koza) konnte festgestellt werden, dass kleine untere Grenzen der HPDI-Werte zu kleinen HPDI-Umfängen führen. Umgekehrt wurde ebenfalls festgestellt, dass große untere Grenzen zu großen HPDI-Ranges führen. Demnach können für CGP-Tests bei denen viele Durchläufe pro CGP-Konfiguration durchgeführt werden, relativ früh Erkenntnisse über ausstehende Durchläufe gewonnen werden. Wenn mehrere Durchläufe besonders früh konvergieren, werden die anderen wahrscheinlich ähnlich früh konvergieren, da eine kleine Streuung der Ergebnisse zu erwarten ist. Umgekehrt wird die Streuung größer, wenn die ersten Durchläufe relativ spät konvergieren. Wichtig dabei ist, dass es sich nicht um Ausreißer handeln darf, da diese bei der 95%-igen HPDI nicht betrachtet werden. Des Weiteren kann daraus abgeleitet werden, dass Trainingserfolge besonders in den frühen Iterationen wichtig sind, da späte Konvergenzen und somit späte Trainingserfolge eine größere Streuung aufweisen und damit unsicherer sind. Da die Rekombination vor allem in den ersten Iterationen einen größeren Trainingsfortschritt erzielen kann, ist es wichtig die Rekombination weiter auszubauen, um die Streuung der Iterationen bis zur Konvergenz zu reduzieren.
Für den Encode-Testfall wurde in der bayes'schen Analyse die Fitness zur 10000. Iteration bewertet. Die gleiche Analyse ergibt hier, dass es keinen Zusammenhang zwischen den unteren Grenzen der HPDI-Werte zu deren Ranges gibt. Demnach kann keine Aussage zu der Streuung der Fitness-Werte in einer Iteration getroffen werden, wenn bekannt ist, dass die kleinsten Fitness-Werte der CGP-Modelle besonders groß oder klein sind.

5 Fazit aus Ausblick

Da das klassische GP bereits in verschiedenen Domänen für Problemlösungen verwendet wird, ist es sinnvoll den Bereich des CGP näher zu betrachten. Laut Miller werde der Rekombinationsschritt in CGP normalerweise nicht ausgeführt, da dieser die Effizienz eines CGP-Modells nicht zielführend steigert [Mil20]. Trotzdem wäre es sinnvoll neben der Mutation einen weiteren genetischen Operator einzubinden, um die Effizienz des CGPs zu steigern und somit komplexere Ausgangsprobleme lösbar zu machen. Die Aussagen, dass Rekombinationsschritte bisher nicht sinnvoll in das Standard-CGP eingepflegt werden können basiert auf Papern von Miller aus den Jahren 1999 und 2011. Diese Aussage wird in dieser Arbeit erneut überprüft. Da der Erfolg der Rekombination stark von der Wahl der Rekombinationsrate abhängt, werden in dieser Arbeit ebenfalls unterschiedliche Arten für Rekombinationsraten betrachtet und bewertet. Für diese Arbeit ergeben sich demnach die folgenden zwei Forschungsfragen:
1. Kann nach heutigem Forschungsstand Rekombination die Effizienz von Standard-CGP steigern?
2. Hat die Art und Weise der Rekombinationsratenberechnung eine Auswirkung auf die Effektivität des CGPs?

Für die Beantwortung der Forschungsfragen wurden vier verschiedene Testprobleme aufgebaut, die durch CGP-Modelle mit verschiedenen Konfigurationen gelöst werden sollten. Die verschiedenen CGP-Konfigurationen beinhalteten unterschiedliche Rekombinationsarten und Varianten der Rekombinationsratenberechnung, sowie den völligen Ausschluss der Rekombination. Für die unterschiedlichen Rekombinationsraten wurde eine zusätzliche Berechnungsvariante eingeführt, die auf der One-Fifth Regel für die Berechnung der Mutationsrate basiert. Bei zwei der vier Testszenarien wurde eine Hyperparameteranalyse ausgeführt, um die Effizienz der besten Ergebnisse für jede CGP-Konfiguration miteinander zu vergleichen. Für die anderen zwei Testfälle wurden die Rekombiantionsraten-Arten facettenreich parametriert, um bewerten zu können, wie sich die Parametrierung der Rekombinationsraten auf die Effizienz des CGPs auswirken.
Für die Analyse der Ergebnisse wurden zwei unterschiedliche Verfahren angewendet. Einerseits wurden die Ergebnisse der Hyperparameteroptimierung und die aufgezeichneten

Rohdaten der Testdurchläufe näher analysiert, andererseits wurde eine Bewertung mit Hilfe der bayes'schen Analyse ausgeführt.
Die Ergebnisse der Evaluierung der Hyperparameteranalyse zeigen vor allem, dass kein ersichtlicher Zusammenhang zwischen den CGP-Konfigurationen und deren Parametrierung besteht. So treten beispielsweise keine einheitlichen Muster auf, wie sich die CGP-Konfigurationen auf die Anzahl der Rechenknoten oder Populationsgröße auswirkt. Durch eine umfangreichere Hyperparameteranalyse mit gegebenenfalls mehr Testszenarien könnten dieses Ergebnis näher untersucht und erneut bewertet werden.
Die Rohdatenanalyse der Trainingsdaten zeigen auf, dass Mutation häufiger die Fitness eines CGP-Modells verbessert als die Rekombination. Dabei handelt es sich allerdings um eine quantitative Bewertung, die in weiteren Analysen näher betrachtet werden sollte. Die Rekombination führt vor allem in den frühen Trainingsphasen zum Erfolg. Für einen Test für symbolische Regression können vereinzelt für konstante Rekombinationsraten spätere Rekombinationserfolge beobachtet werden. Diese Beobachtung könnte in weiteren Tests evaluiert werden, um einen Zusammenhang zwischen späteren Rekombinationserfolgen und der Berechnung der Rekombinationsrate zu überprüfen. Mit diesem Wissen könnte Rekombination gezielter angewendet werden. Die Ergebnisse zeigen außerdem, dass positive Effekte der Rekombination auf die Fitness durch folgende Mutation vermindert oder vollkommen zerstört werden können. Ein zusätzlicher Evaluationsschritt zwischen Rekombination und Mutation könnte somit das Standard-CGP sinnvoll erweitern, sodass Rekombinationserfolge erhalten bleiben. Allerdings scheint es durchaus wahrscheinlich, dass die Rekombination umgekehrt auch die Mutation negativ beeinflusst. Demnach wäre es sinnvoll das Standard-CGP-Verfahren zu überdenken und anzupassen, sodass Rekombination und Mutation ihr volles Potential nutzen können, wenn beide genetische Operatoren angewendet werden sollen. Ein Beispiel dafür wäre eine Teilung der Population, sodass ein Anteil der Nachwuchschromosome durch Rekombination und ein anderer Teil durch Mutation erzeugt werden können. Anschließend könnten die Elitistenauswahl über die gesamte Population hinweg stattfinden, sodass eine Verbesserung der Fitness aus beiden genetischen Operatoren einbezogen werden kann. Des Weiteren weist die Uniform Rekombination einen höheren Einfluss auf die Fitnessverbesserung innerhalb des Trainings auf als die One-Point und Two-Point Rekombination. Die konstante und linear fallende Rekombinationsrate besitzen außerdem das größere Potential zu einer Verbesserung der Fitness zu führen als die One-Fifth Regel für Rekombination. Zusammenfassend lässt sich für die Rohdatenanalyse sagen, dass das Standard-CGP-Verfahren die Kombination aus Rekombination und Mutation nicht optimal nutzt. Wenn das CGP-Verfahren angepasst wird, weist die Uniform Rekombination mit konstanter oder linear fallender Rekombinationsrate das wahrscheinlich größte Potential auf, deutlich bessere Ergebnisse zu erzielen.
Die bayes'sche Analyse zeigt, dass eine Rekombination in den ersten Iterationen meistens

die Effizienz des CGPs verbessert. Für ein paar CGP-Konfigurationen überwiegen die negativen Effekte der Rekombination auf die Mutation die positiven, wodurch es weniger sinnvoll ist Rekombination in den ersten Iterationen auszuführen. Die Verbesserung des CGP-Verfahrens, sodass Rekombination und Mutation sinnvoll gleichzeitig genutzt werden können, wird demnach als wichtiger Meilenstein eingestuft. Außerdem zeigt die Evaluation, dass im Standard-CGP die Rekombination in den meisten Fällen für das gesamte Training durchaus Sinn macht. Dieses Ergebnis wurde für die zwei Benchmark Probleme Parity und Keijzer beobachtet. Es wäre durchaus sinnvoll dies durch weitere Tests zu belegen. Besonders gute Ergebnisse für diese einfachen Benchmarks im Standard-CGP liefert die One-Point Rekombination. Die Evaluationen ergeben, dass es sich dabei um eine Rekombinationsart handelt, die weniger Einfluss auf das Training nimmt und dementsprechend weniger Trainingserfolge liefert, aber auch seltener mit der Mutation kollidiert. Für die komplexeren Benchmark Probleme Encode und Koza resultieren vor allem aus der Uniform Rekombination mit fallender Rekombinationsrate bessere Erfolge. Die Vermutung wird getroffen, dass für die einfachen Benchmark Probleme die Rekombination früher abgesetzt werden sollte und daraufhin die Rekombination mit dem Uniform-Verfahren bessere Ergebnisse liefern könnte. Diese Aussage sollte in weiteren Tests evaluiert werden. Als letzten Punkt wird festgestellt, dass frühere Konvergenzen der CGP-Durchläufe zu weniger Streuung der Ergebnisse führen und umgekehrt. Demnach ist es besonders in den ersten Iterationen wichtig einen hohen Trainingserfolg zu erzielen, um ein verlässliches CGP-Modell zu erstellen. Dadurch wird es umso wichtiger Rekombination effektiv in das CGP-Verfahren einzubinden, da dieser genetische Operator besonders das Training in den ersten Iterationen verbessern kann.

Zusammenfassend lassen sich die zwei Forschungsfragen wie folgt beantworten.
In Standard-CGPs kann Rekombination durchaus sinnvoll eingesetzt werden, sodass dessen Effizienz verbessert wird. Dafür ist es allerdings relevant die richtige Parametrierung mit Hilfe einer Hyperparameteranalyse zu ermitteln. Teil dieser Analyse sollten ebenfalls die Wahl der richtigen Rekombinationsart und Rekombinationsraten-Art sein. Weiter wurde herausgefunden, dass die Rekombination ein größeres Potential für die Verbesserung der Effizienz des CGPs aufweist, das im Standard-CGP-Verfahren nicht vollständig ausgeschöpft wird. Hierzu könnten zukünftig weitere Tests ausgeführt werden, um ein geeigneteres CGP-Verfahren zu entwickeln, das Rekombination und Mutation besser vereinen kann.
Des Weiteren hat die Art und Weise der Rekombinationsratenberechnung einen großen Einfluss auf die Effizienz des CGPs. Hohe Rekombinationsraten zu Beginn des Trainings und ein anschließender Abbruch der Rekombination erscheinen besonders sinnvoll. Zu welcher Iteration die Rekombination im besten Fall ausgesetzt werden soll und ob dies

schleichend oder abrupt stattfinden sollte, könnte durch weitere Forschung näher betrachtet werden.

Literatur

[Ahv+19] Milad Taleby Ahvanooey u. a. „A Survey of Genetic Programming and Its Applications“. en. In: *KSII Transactions on Internet and Information Systems* 13.4 (Apr. 2019). ISSN: 19767277. DOI: 10.3837/tiis.2019.04.002.

[Ara20] Lourdes Araujo. „Genetic programming for natural language processing“. en. In: *Genetic Programming and Evolvable Machines* 21.1-2 (Juni 2020), S. 11–32. ISSN: 1389-2576, 1573-7632. DOI: 10.1007/s10710-019-09361-5.

[Car25] Fredrik Bagge Carlson. *baggepinnen/Hyperopt.jl*. original-date: 2018-08-04T11:25:52Z. Jan. 2025. URL: https://github.com/baggepinnen/Hyperopt.jl (besucht am 02. 02. 2025).

[CCL18] Borja Calvo, Josu Ceberio und Jose A. Lozano. „Bayesian inference for algorithm ranking analysis“. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '18. Kyoto, Japan: Association for Computing Machinery, 2018, S. 324–325. ISBN: 9781450357647. DOI: 10.1145/3205651.3205658.

[CHH24] Henning Cui, Michael Heider und Jörg Hähner. „Positional Bias Does Not Influence Cartesian Genetic Programming with Crossover“. en. In: *Parallel Problem Solving from Nature – PPSN XVIII*. Hrsg. von Michael Affenzeller u. a. Bd. 15148. Series Title: Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2024, S. 151–167. ISBN: 978-3-031-70054-5 978-3-031-70055-2. DOI: 10.1007/978-3-031-70055-2_10.

[ČLM13] Matej Črepinšek, Shih-Hsi Liu und Marjan Mernik. „Exploration and exploitation in evolutionary algorithms: A survey“. en. In: *ACM Computing Surveys* 45.3 (Juni 2013), S. 1–33. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2480741.2480752.

[CMH22] Henning Cui, Andreas Margraf und Jörg Hähner. „Refining Mutation Variants in Cartesian Genetic Programming“. en. In: *Bioinspired Optimization Methods and Their Applications*. Hrsg. von Marjan Mernik, Tome Eftimov und Matej Črepinšek. Bd. 13627. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, S. 185–200. ISBN: 978-3-031-21094-5. DOI: 10.1007/978-3-031-21094-5_14.

[CMH23] Henning Cui, Andreas Margraf und Jörg Hähner. „Equidistant Reorder Operator for Cartesian Genetic Programming:“ en. In: *Proceedings of the 15th International Joint Conference on Computational Intelligence*. Rome, Italy: SCITEPRESS - Science und Technology Publications, 2023, S. 64–74. ISBN: 978-989-758-674-3. DOI: 10.5220/0012174100003595.

[Cui+23] Henning Cui u. a. „Weighted Mutation of Connections To Mitigate Search Space Limitations in Cartesian Genetic Programming“. In: *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. FOGA '23. Potsdam, Germany: Association for Computing Machinery, 2023, S. 50–60. ISBN: 9798400702020. DOI: 10.1145/3594805.3607130.

[Cui24a] Henning Cui. *The Positional Bias might not Influence Cartesian Genetic Programming with Crossover*. März 2024. DOI: 10.5281/zenodo.10830014.

[Cui24b] CuiHen. *CuiHen/CGP_with_Crossover_Strategies*. original-date: 2024-03-15T09:54:31Z. Apr. 2024. URL: https://github.com/CuiHen/CGP_with_Crossover_Strategies (besucht am 11. 06. 2024).

[CWM07] Janet Clegg, James Alfred Walker und Julian Frances Miller. „A new crossover technique for Cartesian genetic programming“. en. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. London England: ACM, Juli 2007, S. 1580–1587. ISBN: 978-1-59593-697-4. DOI: 10.1145/1276958.1277276.

[DDL19] Benjamin Doerr, Carola Doerr und Johannes Lengler. „Self-adjusting mutation rates with provably optimal success rules“. en. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. Prague Czech Republic: ACM, Juli 2019, S. 1479–1487. ISBN: 978-1-4503-6111-8. DOI: 10.1145/3321707.3321733.

[Dem+22] Munise Didem Demirbas u. a. „Stress Analysis of 2D-FG Rectangular Plates with Multi-Gene Genetic Programming“. en. In: *Applied Sciences* 12.16 (Aug. 2022), S. 8198. ISSN: 2076-3417. DOI: 10.3390/app12168198.

[ES15] A.E. Eiben und J.E. Smith. *Introduction to Evolutionary Computing*. en. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. ISBN: 978-3-662-44873-1 978-3-662-44874-8. DOI: 10.1007/978-3-662-44874-8.

[GP15] Brian W. Goldman und William F. Punch. „Analysis of Cartesian Genetic Programming's Evolutionary Mechanisms". In: *IEEE Transactions on Evolutionary Computation* 19.3 (2015), S. 359–373. DOI: 10.1109/TEVC.2014.2324539.

[Has+19] Ahmad Hassanat u. a. „Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach". en. In: *Information* 10.12 (Dez. 2019), S. 390. ISSN: 2078-2489. DOI: 10.3390/info10120390.

[HLS99] Myron E Hohil, Derong Liu und Stanley H Smith. „Solving the N-bit parity problem using neural networks". In: *Neural Networks* 12.9 (1999), S. 1321–1323. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(99)00069-6.

[Kal+23] Roman Kalkreuth u. a. „Towards a General Boolean Function Benchmark Suite". en. In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. Lisbon Portugal: ACM, Juli 2023, S. 591–594. ISBN: 9798400701207. DOI: 10.1145/3583133.3590685.

[Kal20] Roman Kalkreuth. „A Comprehensive Study on Subgraph Crossover in Cartesian Genetic Programming:" en. In: *Proceedings of the 12th International Joint Conference on Computational Intelligence*. Budapest, Hungary: SCITEPRESS - Science und Technology Publications, 2020, S. 59–70. ISBN: 978-989-758-475-6. DOI: 10.5220/0010110700590070.

[KK17] Paul Kaufmann und Roman Kalkreuth. „An empirical study on the parametrization of cartesian genetic programming". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '17. Berlin, Germany: Association for Computing Machinery, 2017, S. 231–232. ISBN: 9781450349390. DOI: 10.1145/3067695.3075980.

[KK20] Paul Kaufmann und Roman Kalkreuth. „On the Parameterization of Cartesian Genetic Programming". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, S. 1–8. DOI: 10.1109/CEC48606.2020.9185492.

[Kom18] Michael Kommenda. „Local Optimization and Complexity Control for Symbolic Regression". Dissertation. Linz, Austria: Johannes Kepler University Linz, 2018. URL: https://resolver.obvsg.at/urn:nbn:at:at-ubl:1-21036 (besucht am 31. 01. 2025).

[Koz95] J. R. Koza. „Survey of genetic algorithms and genetic programming“. en. In: *Proceedings of WESCON'95*. San Francisco, CA, USA: IEEE, 1995, S. 589. ISBN: 978-0-7803-2636-1. DOI: 10.1109/WESCON.1995.485447.

[Kra+13] Krzysztof Krawiec u. a., Hrsg. *Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings*. en. Bd. 7831. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-37206-3. DOI: 10.1007/978-3-642-37207-0.

[Kru12] John Kruschke. „Bayesian Estimation Supersedes the t Test“. In: *Journal of experimental psychology. General* 142 (Juli 2012). DOI: 10.1037/a0029146.

[Li+24] Yanjie Li u. a. *Generative Pre-Trained Transformer for Symbolic Regression Base In-Context Reinforcement Learning*. _eprint: 2404.06330. 2024. URL: https://arxiv.org/abs/2404.06330.

[MC24] Nour Makke und Sanjay Chawla. „Interpretable scientific discovery with symbolic regression: a review“. In: *Artificial Intelligence Review* 57.1 (Jan. 2024), S. 2. ISSN: 1573-7462. DOI: 10.1007/s10462-023-10622-0.

[Mil20] Julian Francis Miller. „Cartesian genetic programming: its status and future“. en. In: *Genetic Programming and Evolvable Machines* 21.1-2 (Juni 2020), S. 129–168. ISSN: 1389-2576, 1573-7632. DOI: 10.1007/s10710-019-09360-6.

[MN18] Nicola Milano und Stefano Nolfi. *Scaling Up Cartesian Genetic Programming through Preferential Selection of Larger Solutions*. arXiv:1810.09485. Okt. 2018. URL: http://arxiv.org/abs/1810.09485 (besucht am 14. 10. 2024).

[Nen80] M. Nenning. „Bayes'sche Inferenz bei der statistischen Auswertung von Marktdaten“. In: *Zeitschrift für Operations Research* 24.8 (Dez. 1980), B259–B273. ISSN: 1432-5217. DOI: 10.1007/BF01918729.

[Oli+18] Luiz Otavio Vilas Boas Oliveira u. a. „Analysing Symbolic Regression Benchmarks under a Meta-Learning Approach“. en. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. arXiv:1805.10365 [cs]. Juli 2018, S. 1342–1349. DOI: 10.1145/3205651.3208293.

[OLM18] Patryk Orzechowski, William La Cava und Jason H. Moore. „Where are we now? a large benchmark study of recent symbolic regression methods“. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '18. Kyoto, Japan: Association for Computing Machinery, 2018, S. 1183–1190. ISBN: 9781450356183. DOI: 10.1145/3205455.3205539.

[Pät24] David Pätzel. *dpaetzel/cmpbayes*. original-date: 2022-03-24T10:52:44Z. Juli 2024. URL: https://github.com/dpaetzel/cmpbayes (besucht am 27. 01. 2025).

[Pey20] Pablo Peyrolón. „Definition des Satzes von Bayes oder das Bayes-Theorem“. de. In: *Der Satz von Bayes*. Series Title: essentials. Wiesbaden: Springer Fachmedien Wiesbaden, 2020, S. 13–21. ISBN: 978-3-658-31022-6 978-3-658-31023-3. DOI: 10.1007/978-3-658-31023-3_2.

[PG17] G. Pavai und T. V. Geetha. „A Survey on Crossover Operators“. en. In: *ACM Computing Surveys* 49.4 (Dez. 2017), S. 1–43. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3009966.

[SB01] S. Sette und L. Boullart. „Genetic programming: principles and applications“. en. In: *Engineering Applications of Artificial Intelligence* 14.6 (Dez. 2001), S. 727–736. ISSN: 09521976. DOI: 10.1016/S0952-1976(02)00013-1.

[SB18] José Eduardo da Silva und Heder S. Bernardino. „Cartesian Genetic Programming with Crossover for Designing Combinational Logic Circuits“. In: *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*. Okt. 2018, S. 145–150. DOI: 10.1109/BRACIS.2018.00033.

[Sys89] Gilbert Syswerda. „Uniform Crossover in Genetic Algorithms“. en. In: *Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, Fairfax, Virginia, USA, June 1989*. Fairfax, Virginia, USA, Jan. 1989. ISBN: 1-55860-066-3. URL: https://www.researchgate.net/publication/201976488_Uniform_Crossover_in_Genetic_Algorithms (besucht am 27. 12. 2024).

[TST22] Ali Torabi, Arash Sharifi und Mohammad Teshnehlab. „Using Cartesian Genetic Programming Approach with New Crossover Technique to Design Convolutional Neural Networks“. en. In: *Neural Processing Letters* (Dez. 2022). ISSN: 1370-4621, 1573-773X. DOI: 10.1007/s11063-022-11093-0.

[Whi+13] David R. White u. a. „Better GP benchmarks: community survey results and proposals“. en. In: *Genetic Programming and Evolvable Machines* 14.1 (März 2013), S. 3–29. ISSN: 1389-2576, 1573-7632. DOI: 10.1007/s10710-012-9177-2.

[YM01] Tina Yu und Julian Miller. „Neutrality and the Evolvability of Boolean Function Landscape“. In: *Genetic Programming*. Hrsg. von Julian Miller u. a. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, S. 204–217. ISBN: 978-3-540-45355-0.

A Anhang

A.1 Tabellen Rohdatenanalyse

A.1.1 Parity: Tabellen Rohdatenanalyse

CGP-Konfigurationen	Anzahl Rechenknoten	λ (Nachwuchs)	Start-Rekombinationsrate	Delta Rekombinationsrate	μ (Elitisten)	Offset
keine Rekombination	1000	20	-	-	18	-
One-Point Konstant kein Offset	1350	52	0,30	-	16	-
One-Point Konstant mit Offset	1500	50	0,8	-	18	25
One-Point Clegg kein Offset	1950	42	-	0,035	14	-
One-Point Clegg mit Offset	1600	58	-	0,01	20	25
One-Point One-Fifth kein Offset	1150	46	0,55	-	6	-
One-Point One-Fifth mit Offset	1100	54	0,35	-	20	30
Two-Point Konstant kein Offset	1900	58	0,4	-	20	-
Two-Point Konstant mit Offset	850	54	0,3	-	16	15
Two-Point Clegg kein Offset	750	56	-	0,02	18	-
Two-Point Clegg mit Offset	1700	28	-	0,04	12	20
Two-Point One-Fifth kein Offset	1800	44	0,55	-	16	-
Two-Point One-Fifth mit Offset	950	58	0,7	-	10	15
Uniform Konstant kein Offset	1800	58	0,8	-	20	-
Uniform Konstant mit Offset	1400	54	0,3	-	20	45
Uniform Clegg kein Offset	1600	50	-	0,015	20	-
Uniform Clegg mit Offset	1500	52	-	0,04	18	25
Uniform One-Fifth kein Offset	650	52	0,75	-	14	-
Uniform One-Fifth mit Offset	250	50	0,35	-	16	40

Tabelle A.1: Parity: Ergebnis Hyperparameteranalyse

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.
keine Rekombination	133	0	0	-	58,5
One-Point Konstant kein Offset	126	8	2	5,5	44
One-Point Konstant mit Offset	136	0	0	-	49
One-Point Clegg kein Offset	123	13	3	4	74
One-Point Clegg mit Offset	126	0	0	-	43,5
One-Point One-Fifth kein Offset	119	2	1	2	34
One-Point One-Fifth mit Offset	127	0	0	-	27,5
Two-Point Konstant kein Offset	113	10	2	3	39,5
Two-Point Konstant mit Offset	129	0	0	-	64
Two-Point Clegg kein Offset	106	20	8	6,5	40
Two-Point Clegg mit Offset	125	0	0	-	70,5
Two-Point One-Fifth kein Offset	126	4	2	3	59,5
Two-Point One-Fifth mit Offset	125	0	0	-	43
Uniform Konstant kein Offset	85	44	15	14	55
Uniform Konstant mit Offset	131	0	0	-	37,5
Uniform Clegg kein Offset	101	41	15	4	31
Uniform Clegg mit Offset	130	0	0	-	51,5
Uniform One-Fifth kein Offset	108	22	12	5	69,5
Uniform One-Fifth mit Offset	123	0	0	-	54

Tabelle A.2: Parity: Auswertung der Rohdaten

A.1.2 Keijzer: Tabellen Rohdatenanalyse

CGP-Konfigurationen	Anzahl Rechenknoten	λ (Nachwuchs)	Start-Rekombinationsrate	Delta Rekombinationsrate	μ (Elitisten)	Offset
keine Rekombination	1850	44	-	-	20	-
One-Point Konstant kein Offset	600	50	0,2	-	20	-
One-Point Konstant mit Offset	1200	50	1	-	4	120
One-Point Clegg kein Offset	1100	60	-	0,05	16	-
One-Point Clegg mit Offset	850	60	-	0,005	16	300
One-Point One-Fifth kein Offset	350	34	0,75	-	18	-
One-Point One-Fifth mit Offset	2000	28	0,65	-	18	180
Two-Point Konstant kein Offset	1700	60	0,5	-	8	-
Two-Point Konstant mit Offset	600	48	0,3	-	16	180
Two-Point Clegg kein Offset	750	34	-	0,005	14	-
Two-Point Clegg mit Offset	800	36	-	0,045	20	210
Two-Point One-Fifth kein Offset	1350	40	0,35	-	20	-
Two-Point One-Fifth mit Offset	700	52	0,7	-	8	30
Uniform Konstant kein Offset	1100	52	0,8	-	8	-
Uniform Konstant mit Offset	1800	60	0,1	-	20	90
Uniform Clegg kein Offset	850	26	-	0,05	18	-
Uniform Clegg mit Offset	2000	44	-	0,05	14	180
Uniform One-Fifth kein Offset	1150	54	0,75	-	18	-
Uniform One-Fifth mit Offset	900	48	0,35	-	20	90

Tabelle A.3: Keijzer: Ergebnis Hyperparameteranalyse

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.
keine Rekombination	2260	0	0	-	624
One-Point Konstant kein Offset	1183	26	16	13	756
One-Point Konstant mit Offset	1531	0	0	-	374
One-Point Clegg kein Offset	1584	59	28	5	243
One-Point Clegg mit Offset	1534	0	0	-	396,5
One-Point One-Fifth kein Offset	1307	35	18	3	766
One-Point One-Fifth mit Offset	1776	0	0	-	229
Two-Point Konstant kein Offset	1854	79	25	15	803
Two-Point Konstant mit Offset	1361	0	0	-	548
Two-Point Clegg kein Offset	1467	106	62	18,5	735
Two-Point Clegg mit Offset	1862	0	0	-	201
Two-Point One-Fifth kein Offset	1788	17	10	4	753
Two-Point One-Fifth mit Offset	1124	0	0	-	158
Uniform Konstant kein Offset	1224	480	239	24	363
Uniform Konstant mit Offset	1890	0	0	-	260,5
Uniform Clegg kein Offset	1386	77	57	6	723
Uniform Clegg mit Offset	2232	0	0	-	385,5
Uniform One-Fifth kein Offset	1491	106	50	8	155
Uniform One-Fifth mit Offset	1749	0	0	-	445,5

Tabelle A.4: Keijzer: Auswertung der Rohdaten

A.1.3 Encode: Tabellen Rohdatenanalyse

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.	Stopp-Kriterium erfüllt
keine Rekombination	1138	-	-	-	3847,5	8
One-Point Konstant: 0,125	1126	20	4	9,5	3362	9
One-Point Konstant: 0,25	1100	42	5	12,5	1963	9
One-Point Konstant: 0,375	1120	43	5	6	4578,5	8
One-Point Konstant: 0,5	1104	34	2	10	936	9
One-Point Konstant: 0,625	1096	45	4	12	3293	6
One-Point Konstant: 0,75	1103	55	11	16	1898,5	6
One-Point Konstant: 0,875	1093	59	6	15	1913	10
One-Point Konstant: 1,0	1086	58	9	8	3754,5	8
One-Point Clegg: 0,0005	1125	48	6	11	1905,5	12
One-Point Clegg: 0,0015	1062	51	5	12	2754,5	6
One-Point Clegg: 0,0025	1063	39	2	9	1530	5
One-Point Clegg: 0,0035	1114	34	2	10,5	2840,5	6
One-Point Clegg: 0,0045	1127	38	2	10	2558	9
One-Point Clegg: 0,0055	1079	55	7	9	4007,5	8
One-Point One-Fifth: 0,125	1138	19	1	7	3950	9
One-Point One-Fifth: 0,25	1135	22	3	7	4111,5	12
One-Point One-Fifth: 0,375	1155	20	1	4	1685	7
One-Point One-Fifth: 0,5	1123	26	0	5	3205	11
One-Point One-Fifth: 0,625	1100	35	3	6	3809	11
One-Point One-Fifth: 0,75	1089	25	1	6	795	7
One-Point One-Fifth: 0,875	1126	32	5	8	5213	5
One-Point One-Fifth: 1,0	1112	42	3	5,5	2863	5

Tabelle A.5: Encode One-Point Rekombination: Auswertung der Rohdaten

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.	Stopp-Kriterium erfüllt
keine Rekombination	1138	-	-	-	3847,5	8
Two-Point Konstant: 0,125	1044	27	2	6	4039,5	2
Two-Point Konstant: 0,25	1062	35	9	7	4292	3
Two-Point Konstant: 0,375	1076	45	4	5	1840	5
Two-Point Konstant: 0,5	1041	50	12	7,5	1666	3
Two-Point Konstant: 0,625	1050	61	15	7	7370	3
Two-Point Konstant: 0,75	1019	61	14	10	3138	1
Two-Point Konstant: 0,875	1023	73	12	9	4445	5
Two-Point Konstant: 1,0	1031	65	10	5	3476,5	6
Two-Point Clegg: 0,0005	1041	70	7	6	3383	5
Two-Point Clegg: 0,0015	1015	53	8	8	2524,5	2
Two-Point Clegg: 0,0025	1026	75	8	9	6024	3
Two-Point Clegg: 0,0035	977	64	9	7	2061	3
Two-Point Clegg: 0,0045	1046	51	9	8	2616	5
Two-Point Clegg: 0,0055	1019	52	8	9,5	3475	7
Two-Point One-Fifth: 0,125	1067	10	3	5	575	2
Two-Point One-Fifth: 0,25	1077	16	4	4	707	1
Two-Point One-Fifth: 0,375	1067	17	1	4	7538	1
Two-Point One-Fifth: 0,5	1063	24	5	5	3363	3
Two-Point One-Fifth: 0,625	1075	32	6	4	4919	5
Two-Point One-Fifth: 0,75	1078	34	8	4	1312	6
Two-Point One-Fifth: 0,875	1088	36	4	5	2529,5	6
Two-Point One-Fifth: 1,0	1049	43	7	4	1140	5

Tabelle A.6: Encode Two-Point Rekombination: Auswertung der Rohdaten

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.	Stopp-Kriterium erfüllt
keine Rekombination	1138	-	-	-	3847,5	8
Uniform Konstant: 0,125	1105	52	12	7	4058	9
Uniform Konstant: 0,25	1099	69	3	13	3831,5	10
Uniform Konstant: 0,375	1076	64	10	9	4071,5	6
Uniform Konstant: 0,5	1105	68	2	12,5	993,5	8
Uniform Konstant: 0,625	1099	78	3	9	1306,5	6
Uniform Konstant: 0,75	1085	81	6	10	3459	8
Uniform Konstant: 0,875	1088	83	1	10	2490,5	6
Uniform Konstant: 1,0	1081	67	5	10	4785	9
Uniform Clegg: 0,0005	1092	76	5	8	2562	13
Uniform Clegg: 0,0015	1099	78	2	10,5	1777	11
Uniform Clegg: 0,0025	1096	74	7	11	2401,5	10
Uniform Clegg: 0,0035	1095	56	4	12	4674,5	4
Uniform Clegg: 0,0045	1080	70	2	13	5118	10
Uniform Clegg: 0,0055	1122	68	10	9,5	4044,5	8
Uniform One-Fifth: 0,125	1152	18	2	6,5	3876	7
Uniform One-Fifth: 0,25	1126	25	6	6	3393	11
Uniform One-Fifth: 0,375	1128	29	2	8	988,5	6
Uniform One-Fifth: 0,5	1156	44	5	7	3968	11
Uniform One-Fifth: 0,625	1122	40	4	8	1781	7
Uniform One-Fifth: 0,75	1104	47	4	6	3385,5	12
Uniform One-Fifth: 0,875	1075	51	10	10	1831,5	6
Uniform One-Fifth: 1,0	1129	55	7	6	4972	9

Tabelle A.7: Encode Uniform Rekombination: Auswertung der Rohdaten

A.1.4 Koza: Tabellen Rohdatenanalyse

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.	Stopp-Kriterium erfüllt
keine Rekombination	1157	-	-	-	208	48
One-Point Konstant: 0,125	1338	9	8	42	616	41
One-Point Konstant: 0,25	1180	23	16	7	611	45
One-Point Konstant: 0,375	1493	42	25	43	2012	47
One-Point Konstant: 0,5	1477	57	40	18	602,5	36
One-Point Konstant: 0,625	1287	48	23	82,5	2302	45
One-Point Konstant: 0,75	1423	65	43	29	989	47
One-Point Konstant: 0,875	1593	85	48	18	848	34
One-Point Konstant: 1,0	1548	136	83	73,5	667	41
One-Point Clegg: 0,0005	1460	98	56	45,5	603	43
One-Point Clegg: 0,0015	1525	78	43	12	659	40
One-Point Clegg: 0,0025	1303	52	31	7	583	41
One-Point Clegg: 0,0035	1233	61	35	6	529	42
One-Point Clegg: 0,0045	1187	62	36	10	511	44
One-Point Clegg: 0,0055	1622	50	28	9,5	1320	42
One-Point One-Fifth: 0,125	1434	2	1	1,5	462	44
One-Point One-Fifth: 0,25	1544	7	6	1	1400	44
One-Point One-Fifth: 0,375	1005	13	8	2	838,5	46
One-Point One-Fifth: 0,5	1137	14	10	2	360,5	41
One-Point One-Fifth: 0,625	1207	18	12	2	947	44
One-Point One-Fifth: 0,75	1387	17	11	3	910	43
One-Point One-Fifth: 0,875	1160	29	14	4	301	43
One-Point One-Fifth: 1,0	1393	38	25	4	568	43

Tabelle A.8: Koza One-Point Rekombination: Auswertung der Rohdaten

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.	Stopp-Kriterium erfüllt
keine Rekombination	1157	-	-	-	208	48
Two-Point Konstant: 0,125	1383	41	24	106	106	45
Two-Point Konstant: 0,25	1475	82	55	47,5	613	48
Two-Point Konstant: 0,375	1207	91	58	75	153	43
Two-Point Konstant: 0,5	1120	85	49	29	194	46
Two-Point Konstant: 0,625	1013	93	38	16	124	49
Two-Point Konstant: 0,75	1373	145	65	21	190	43
Two-Point Konstant: 0,875	1400	140	76	26	731	46
Two-Point Konstant: 1,0	1176	220	140	50,5	207	48
Two-Point Clegg: 0,0005	1446	193	99	30	205	47
Two-Point Clegg: 0,0015	1167	105	59	14	154	45
Two-Point Clegg: 0,0025	1396	106	53	18	504,5	44
Two-Point Clegg: 0,0035	1043	95	45	13	142	46
Two-Point Clegg: 0,0045	1023	100	50	14,5	96	46
Two-Point Clegg: 0,0055	1160	94	43	11,5	223,5	49
Two-Point One-Fifth: 0,125	1422	5	4	2	226	45
Two-Point One-Fifth: 0,25	1348	10	5	4	404,5	44
Two-Point One-Fifth: 0,375	1365	15	4	3	460	46
Two-Point One-Fifth: 0,5	1761	15	4	5	584	47
Two-Point One-Fifth: 0,625	1165	22	13	6	165	48
Two-Point One-Fifth: 0,75	1041	26	13	4	213,5	46
Two-Point One-Fifth: 0,875	1050	42	19	7	185	48
Two-Point One-Fifth: 1,0	1142	35	20	5	348	49

Tabelle A.9: Koza Two-Point Rekombination: Auswertung der Rohdaten

CGP-Konfigurationen	Anzahl pos. Mutationen	Anzahl pos. Rekomb.	Anzahl neg. Mutationen	Median Iter. pos. Rekomb.	Median Iter. bis Konv.	Stopp-Kriterium erfüllt
keine Rekombination	1157	-	-	-	208	48
Uniform Konstant: 0,125	1178	124	75	31,5	316	44
Uniform Konstant: 0,25	1233	157	90	44	479	43
Uniform Konstant: 0,375	1166	251	132	35	257,5	47
Uniform Konstant: 0,5	1084	184	92	27,5	113	45
Uniform Konstant: 0,625	1277	295	152	31	430	47
Uniform Konstant: 0,75	1301	276	141	31	382	43
Uniform Konstant: 0,875	1102	332	181	31	467	44
Uniform Konstant: 1,0	1080	328	199	21	176	43
Uniform Clegg: 0,0005	1155	373	197	40	709	46
Uniform Clegg: 0,0015	793	259	147	17	81	48
Uniform Clegg: 0,0025	887	319	181	25	140	48
Uniform Clegg: 0,0035	1415	318	162	18	162	46
Uniform Clegg: 0,0045	1404	304	153	18	637	48
Uniform Clegg: 0,0055	1364	300	141	19	143	46
Uniform One-Fifth: 0,125	1545	12	6	5	147	41
Uniform One-Fifth: 0,25	1314	24	8	6	153	44
Uniform One-Fifth: 0,375	1384	36	14	8,5	273	47
Uniform One-Fifth: 0,5	1718	61	28	6	848	46
Uniform One-Fifth: 0,625	1244	72	35	6	363	47
Uniform One-Fifth: 0,75	1320	90	49	5	207	44
Uniform One-Fifth: 0,875	1406	92	49	9	470	46
Uniform One-Fifth: 1,0	1444	83	40	6	794,5	47

Tabelle A.10: Koza Uniform Rekombination: Auswertung der Rohdaten

A.2 Tabellen Bayes'sche Analyse

A.2.1 Parity: Tabelle Bayes'sche Analyse

CPG-Konfiguration	HPDI (Iter.)	MW	PL-Platz
Parity keine Rekombination	(237,958; 541,181)	357,440	0,035
Parity One-Point Konstant kein Offset	(72,803; 131,161)	98,059	0,073
Parity One-Point Konstant mit Offset	(117,5867; 241,124)	168,301	0,060
Parity One-Point Clegg kein Offset	(111,330; 204,672)	151,207	0,058
Parity One-Point Clegg mit Offset	(136,755; 318,250)	208,318	0,065
Parity One-Point One-Fifth kein Offset	(305,150; 879,234)	516,320	0,041
Parity One-Point One-Fifth mit Offset	(90,584; 192,603)	132,238	0,073
Parity Two-Point Konstant kein Offset	(161,166; 366,726)	243,306	0,057
Parity Two-Point Konstant mit Offset	(185,680; 398,045)	271,755	0,044
Parity Two-Point Clegg kein Offset	(143,411; 320,218)	214,015	0,056
Parity Two-Point Clegg mit Offset	(279,211; 671,061)	429,804	0,041
Parity Two-Point One-Fifth kein Offset	(165,870; 369,988)	247,230	0,052
Parity Two-Point One-Fifth mit Offset	(187,841; 467,866)	294,748	0,049
Parity Uniform Konstant kein Offset	(182,312; 414,373)	275,283	0,043
Parity Uniform Konstant mit Offset	(158,116; 357,830)	238,048	0,061
Parity Uniform Clegg kein Offset	(120,352; 267,732)	179,796	0,064
Parity Uniform Clegg mit Offset	(147,184; 317,828)	215,359	0,052
Parity Uniform One-Fifth kein Offset	(233,817; 551,718)	356,681	0,040
Parity Uniform One-Fifth mit Offset	(211,377; 519,471)	329,524	0,037

Tabelle A.11: Parity: Bayes'sche Analyse

A.2.2 Keijzer: Tabelle Bayes'sche Analyse

CGP-Konfiguration	HPDI (Iter.)	MW	PL-Platz
Keijzer keine Rekombination	(5446,690; 16772,637)	9551,657	0,047
Keijzer One-Point Konstant kein Offset	(3155,154; 8685,286)	5214,911	0,057
Keijzer One-Point Konstant mit Offset	(5791,946; 19545,587)	10837,605	0,048
Keijzer One-Point Clegg kein Offset	(4055,470; 14388,586)	7754,999	0,053
Keijzer One-Point Clegg mit Offset	(5182,282; 17954,643)	9757,394	0,047
Keijzer One-Point One-Fifth kein Offset	(3746,883; 10479,686)	6233,279	0,050
Keijzer One-Point One-Fifth mit Offset	(4972,228; 16087,780)	9402,618	0,058
Keijzer Two-Point Konstant kein Offset	(4207,924; 13285,672)	7464,243	0,062
Keijzer Two-Point Konstant mit Offset	(3544,144; 10355,412)	6056,680	0,064
Keijzer Two-Point Clegg kein Offset	(5468,641; 17877,607)	10017,892	0,034
Keijzer Two-Point Clegg mit Offset	(4105,772; 13578,136)	7663,952	0,056
Keijzer Two-Point One-Fifth kein Offset	(4408,582; 12408,349)	7381,426	0,053
Keijzer Two-Point One-Fifth mit Offset	(6970,619; 22911,685)	13432,511	0,052
Keijzer Uniform Konstant kein Offset	(8010,603; 23268,991)	14688,052	0,049
Keijzer Uniform Konstant mit Offset	(3406,437; 10751,973)	6052,727	0,070
Keijzer Uniform Clegg kein Offset	(3856,791; 10850,478)	6440,975	0,061
Keijzer Uniform Clegg mit Offset	(6067,492; 20498,984)	11238,547	0,044
Keijzer Uniform One-Fifth kein Offset	(3511,828; 12119,350)	6587,897	0,071
Keijzer Uniform One-Fifth mit Offset	(4025,084; 11844,356)	6893,370	0,025

Tabelle A.12: Keijzer: Bayes'sche Analyse

A.2.3 Encode: Tabellen Bayes'sche Analyse

CGP-Konfiguration	HPDI (Fitn.)	MW	PL-Platz
Encode keine Rekombination	(0,02727; 0,06853)	0,04351	0,038
Encode One-Point Konstant: 0,125	(0,02416; 0,06463)	0,03969	0,038
Encode One-Point Konstant: 0,25	(0,02342; 0,06255)	0,03837	0,042
Encode One-Point Konstant: 0,375	(0,02216; 0,05684)	0,036	0,052
Encode One-Point Konstant: 0,5	(0,02581; 0,06657)	0,04185	0,040
Encode One-Point Konstant: 0,625	(0,02703; 0,06417)	0,04188	0,037
Encode One-Point Konstant: 0,75	(0,02484; 0,05755)	0,03791	0,043
Encode One-Point Konstant: 0,875	(0,02311; 0,06519)	0,03911	0,043
Encode One-Point Konstant: 1,0	(0,02634; 0,06865)	0,04257	0,039
Encode One-Point Clegg: 0,0005	(0,01874; 0,05612)	0,03254	0,056
Encode One-Point Clegg: 0,0015	(0,03085; 0,07312)	0,04756	0,034
Encode One-Point Clegg: 0,0025	(0,03174; 0,07112)	0,04783	0,036
Encode One-Point Clegg: 0,0035	(0,0259; 0,05957)	0,03961	0,042
Encode One-Point Clegg: 0,0045	(0,02262; 0,05981)	0,03682	0,049
Encode One-Point Clegg: 0,0055	(0,02478; 0,06459)	0,04044	0,042
Encode One-Point One-Fifth: 0,125	(0,02234; 0,06005)	0,03702	0,045
Encode One-Point One-Fifth: 0,25	(0,02132; 0,06241)	0,0368	0,048
Encode One-Point One-Fifth: 0,375	(0,02423; 0,05867)	0,03784	0,046
Encode One-Point One-Fifth: 0,5	(0,01903; 0,05371)	0,03227	0,059
Encode One-Point One-Fifth: 0,625	(0,02184; 0,0643)	0,03777	0,045
Encode One-Point One-Fifth: 0,75	(0,02746; 0,06625)	0,04286	0,039
Encode One-Point One-Fifth: 0,875	(0,02436; 0,05364)	0,03622	0,051
Encode One-Point One-Fifth: 1,0	(0,02887; 0,06487)	0,04343	0,037

Tabelle A.13: Encode One-Point Rekombination: Bayes'sche Analyse

CGP-Konfiguration	HPDI (Fitn.)	MW	PL-Platz
Encode keine Rekombination	(**0,02727**; 0,06853)	0,04351	0,061
Encode Two-Point Konstant: 0,125	(0,04657; 0,08164)	0,06176	0,035
Encode Two-Point Konstant: 0,25	(0,0417; 0,08229)	0,05853	0,036
Encode Two-Point Konstant: 0,375	(**0,03172**; 0,07007)	0,04742	0,052
Encode Two-Point Konstant: 0,5	(0,04318; 0,08532)	0,06094	0,035
Encode Two-Point Konstant: 0,625	(0,03854; 0,07511)	0,05397	0,045
Encode Two-Point Konstant: 0,75	(0,05211; 0,08389)	0,06618	0,032
Encode Two-Point Konstant: 0,875	(0,03614; 0,08027)	0,05438	0,041
Encode Two-Point Konstant: 1,0	(0,03386; 0,07925)	0,052	0,047
Encode Two-Point Clegg: 0,0005	(0,03588; 0,07852)	0,05322	0,044
Encode Two-Point Clegg: 0,0015	(0,0457; 0,08078)	0,06077	0,038
Encode Two-Point Clegg: 0,0025	(0,03801; 0,07255)	0,05258	0,046
Encode Two-Point Clegg: 0,0035	(0,04651; 0,09123)	0,06523	0,033
Encode Two-Point Clegg: 0,0045	(0,0321; 0,07079)	0,04795	0,054
Encode Two-Point Clegg: 0,0055	(0,03634; 0,08965)	0,0573	0,038
Encode Two-Point One-Fifth: 0,125	(0,04561; 0,08148)	0,06082	0,039
Encode Two-Point One-Fifth: 0,25	(0,04353; 0,0687)	0,05485	0,047
Encode Two-Point One-Fifth: 0,375	(0,04642; 0,07558)	0,05903	0,038
Encode Two-Point One-Fifth: 0,5	(0,03834; 0,07567)	0,0539	0,042
Encode Two-Point One-Fifth: 0,625	(0,036; 0,07918)	0,0536	0,045
Encode Two-Point One-Fifth: 0,75	(**0,03114**; 0,07277)	0,04805	0,053
Encode Two-Point One-Fifth: 0,875	(0,03244; 0,07634)	0,05012	0,051
Encode Two-Point One-Fifth: 1,0	(0,03436; 0,07444)	0,05057	0,047

Tabelle A.14: Encode Two-Point Rekombination: Bayes'sche Analyse

CGP-Konfiguration	HPDI (Fitn.)	MW	PL-Platz
Encode keine Rekombination	(0,02727; 0,06853)	0,04351	0,034
Encode Uniform Konstant: 0,125	(0,02473; 0,06504)	0,04031	0,038
Encode Uniform Konstant: 0,25	(0,02356; 0,06581)	0,03963	0,039
Encode Uniform Konstant: 0,375	(0,03039; 0,07186)	0,047	0,033
Encode Uniform Konstant: 0,5	(0,02241; 0,05759)	0,03583	0,046
Encode Uniform Konstant: 0,625	(0,02416; 0,05673)	0,03719	0,046
Encode Uniform Konstant: 0,75	(0,02399; 0,06258)	0,03882	0,047
Encode Uniform Konstant: 0,875	(0,02203; 0,05148)	0,03393	0,047
Encode Uniform Konstant: 1,0	(0,022; 0,05696)	0,03592	0,045
Encode Uniform Clegg: 0,0005	(0,01801; 0,0552)	0,03171	0,050
Encode Uniform Clegg: 0,0015	(0,02101; 0,06014)	0,03576	0,050
Encode Uniform Clegg: 0,0025	(0,01737; 0,04703)	0,02884	0,056
Encode Uniform Clegg: 0,0035	(0,02831; 0,05921)	0,04102	0,042
Encode Uniform Clegg: 0,0045	(0,0247; 0,06962)	0,04197	0,035
Encode Uniform Clegg: 0,0055	(0,02211; 0,05691)	0,03563	0,041
Encode Uniform One-Fifth: 0,125	(0,02138; 0,05172)	0,03338	0,047
Encode Uniform One-Fifth: 0,25	(0,02075; 0,06059)	0,03566	0,046
Encode Uniform One-Fifth: 0,375	(0,02667; 0,06216)	0,04111	0,040
Encode Uniform One-Fifth: 0,5	(0,01608; 0,04588)	0,02738	0,061
Encode Uniform One-Fifth: 0,625	(0,02459; 0,06074)	0,03881	0,047
Encode Uniform One-Fifth: 0,75	(0,02107; 0,06167)	0,03642	0,040
Encode Uniform One-Fifth: 0,875	(0,03182; 0,07718)	0,04969	0,030
Encode Uniform One-Fifth: 1,0	(0,02304; 0,06027)	0,03742	0,041

Tabelle A.15: Encode Uniform Rekombination: Bayes'sche Analyse

A.2.4 Koza: Tabellen Bayes'sche Analyse

CPG-Konfiguration	HPDI (Iter.)	MW	PL-Platz
Koza keine Rekombination	(3306,396; 9338,152)	5573,228	0,066
Koza One-Point Konstant: 0,125	(11989,956; 39904,352)	22076,287	0,039
Koza One-Point Konstant: 0,25	(5686,475; 17041,592)	9857,749	0,051
Koza One-Point Konstant: 0,375	(7670,19; 19745,987)	12308,128	0,041
Koza One-Point Konstant: 0,5	(21220,642; 77451,789)	41120,884	0,033
Koza One-Point Konstant: 0,625	(8850,277; 22781,195)	14213,975	0,040
Koza One-Point Konstant: 0,75	(6558,793; 17538,735)	10732,756	0,046
Koza One-Point Konstant: 0,875	(22884,074; 83707,754)	44778,188	0,030
Koza One-Point Konstant: 1,0	(11999,809; 40932,105)	22223,655	0,041
Koza One-Point Clegg: 0,0005	(8804,961; 28842,332)	16070,797	0,049
Koza One-Point Clegg: 0,0015	(12897,221; 44229,228)	24132,461	0,041
Koza One-Point Clegg: 0,0025	(12097,518; 42109,887)	22745,754	0,034
Koza One-Point Clegg: 0,0035	(10713,693; 36149,782)	19751,18	0,040
Koza One-Point Clegg: 0,0045	(8152,674; 25944,801)	14620,549	0,045
Koza One-Point Clegg: 0,0055	(10396,969; 32228,794)	18312,577	0,041
Koza One-Point One-Fifth: 0,125	(7042,407; 22949,118)	12782,824	0,051
Koza One-Point One-Fifth: 0,25	(8698,915; 24772,687)	14733,52	0,043
Koza One-Point One-Fifth: 0,375	(5766,106; 16551,173)	9801,34	0,051
Koza One-Point One-Fifth: 0,5	(10523,665; 39039,178)	20390,489	0,047
Koza One-Point One-Fifth: 0,625	(7477,612; 22329,85)	13015,454	0,043
Koza One-Point One-Fifth: 0,75	(9605,964; 28910,079)	16579,746	0,042
Koza One-Point One-Fifth: 0,875	(7844,464; 27078,529)	14728,439	0,047
Koza One-Point One-Fifth: 1,0	(9099,566; 29587,091)	16404,442	0,037

Tabelle A.16: Koza One-Point Rekombination: Bayes'sche Analyse

CPG-Konfiguration	HPDI (Iter.)	MW	PL-Platz
Koza keine Rekombination	(3306,396; 9338,152)	5573,228	0,042
Koza Two-Point Konstant: 0,125	(5778,032; 20659,535)	11104,449	0,043
Koza Two-Point Konstant: 0,25	(4377,353; 12699,617)	7467,417	0,041
Koza Two-Point Konstant: 0,375	(7682,846; 29668,73)	15237,451	0,042
Koza Two-Point Konstant: 0,5	(4545,7; 15211,868)	8448,031	0,045
Koza Two-Point Konstant: 0,625	(2506,268; 7690,606)	4414,866	0,051
Koza Two-Point Konstant: 0,75	(8106,536; 30489,527)	15831,485	0,036
Koza Two-Point Konstant: 0,875	(5275,995; 16141,93)	9274,254	0,038
Koza Two-Point Konstant: 1,0	(2467,382; 7467,194)	4300,791	0,055
Koza Two-Point Clegg: 0,0005	(3539,092; 10910,749)	6214,133	0,045
Koza Two-Point Clegg: 0,0015	(5136,595; 18389,727)	9803,362	0,043
Koza Two-Point Clegg: 0,0025	(7068,983; 23388,559)	12866,464	0,032
Koza Two-Point Clegg: 0,0035	(4920,07; 17123,139)	9196,382	0,050
Koza Two-Point Clegg: 0,0045	(5065,402; 17637,589)	9552,406	0,045
Koza Two-Point Clegg: 0,0055	(2005,715; 5792,2)	3420,063	0,056
Koza Two-Point One-Fifth: 0,125	(5549,634; 19765,485)	10495,382	0,041
Koza Two-Point One-Fifth: 0,25	(8535,956; 28712,98)	15794,048	0,032
Koza Two-Point One-Fifth: 0,375	(6383,073; 19877,145)	11333,016	0,032
Koza Two-Point One-Fifth: 0,5	(4715,287; 13037,66)	7854,211	0,035
Koza Two-Point One-Fifth: 0,625	(3209,936; 9921,136)	5663,115	0,049
Koza Two-Point One-Fifth: 0,75	(4615,524; 15765,276)	8647,856	0,043
Koza Two-Point One-Fifth: 0,875	(2480,372; 7460,089)	4313,098	0,052
Koza Two-Point One-Fifth: 1,0	(2069,47; 5715,614)	3450,456	0,052

Tabelle A.17: Koza Two-Point Rekombination: Bayes'sche Analyse

CPG-Konfiguration	HPDI (Iter.)	MW	PL-Platz
Koza keine Rekombination	(3306,396; 9338,152)	5573,228	0,045
Koza Uniform Konstant: 0,125	(7116,083; 25435,336)	13593,719	0,039
Koza Uniform Konstant: 0,25	(9984,852; 33237,559)	18247,623	0,029
Koza Uniform Konstant: 0,375	(3172,789; 9799,001)	5586,763	0,052
Koza Uniform Konstant: 0,5	(5845,776; 21080,773)	11192,411	0,048
Koza Uniform Konstant: 0,625	(5056,513; 15161,941)	8855,129	0,043
Koza Uniform Konstant: 0,75	(7678,984; 28599,747)	14989,121	0,042
Koza Uniform Konstant: 0,875	(8075,534; 25962,001)	14529,689	0,037
Koza Uniform Konstant: 1,0	(6997,917; 26553,7)	13799,512	0,046
Koza Uniform Clegg: 0,0005	(5887,322; 17725,061)	10263,634	0,039
Koza Uniform Clegg: 0,0015	(1820,855; 5880,234)	3284,824	0,067
Koza Uniform Clegg: 0,0025	(2105,564; 6348,966)	3669,648	0,058
Koza Uniform Clegg: 0,0035	(6267,561; 20173,041)	11336,11	0,039
Koza Uniform Clegg: 0,0045	(3259,699; 8595,465)	5275,12	0,044
Koza Uniform Clegg: 0,0055	(4447,802; 15210,844)	8263,146	0,044
Koza Uniform One-Fifth: 0,125	(9555,822; 36197,611)	18956,129	0,040
Koza Uniform One-Fifth: 0,25	(6369,195; 24041,614)	12519,802	0,045
Koza Uniform One-Fifth: 0,375	(4284,718; 13099,952)	7548,331	0,045
Koza Uniform One-Fifth: 0,5	(6205,67; 18180,546)	10642,776	0,038
Koza Uniform One-Fifth: 0,625	(5184,312; 15589,696)	9047,982	0,037
Koza Uniform One-Fifth: 0,75	(7626,548; 26532,94)	14283,761	0,037
Koza Uniform One-Fifth: 0,875	(4522,054; 14978,774)	8243,583	0,049
Koza Uniform One-Fifth: 1,0	(5183,432; 15319,365)	8941,343	0,036

Tabelle A.18: Koza Uniform Rekombination: Bayes'sche Analyse