ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Johannes Bader

Schlanke Programme mit genetischer Programmierung

Semesterarbeit

Semester: 8
Betreuer: Stefan Bleuler
Abgabedatum: 22.7.2005

TIK Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Inhaltsverzeichnis

Einleitung

Genetische Programmierung beginnt mit mehreren zufällig generierten Programmen, welche eine klar definierte Funktion lösen sollen. Die sog. Fitness gibt dabei an, wie gut dies gelingt. Die aus der Genetik bekannten Operatoren: Kreuzung, Mutation und Rekombination werden danach verwendet, um neue Programme (die *Individuen* der *Population*) zu kreieren. Das darwinistische Prinzip der natürlichen Auslese bestimmt anschliessend anhand der Fitnesswerte darüber, welche Individuen in der Populationen verbleiben und welche entfernt werden (siehe Kapitel *„Einführung in die genetische Programmierung"*).

Auf diese Weise entwickelt sich die Anfangspopulation sukzessive in verschiedenen Generationen weiter.

Leider haben die Programme jedoch die Tendenz, unkontrolliert zu wachsen. Dieses Wachstum bremst die Evolution oder bringt sie sogar ganz zum Erliegen (*siehe Abschnitt „Bloat"*).

Ein möglicher Ansatz, um dieses unbeschränkte Programm Wachstum, das sog. *„Bloat"* zu verhindern, ist die *Mehrzieloptimierung*. Dabei werden Individuen sowohl anhand ihrer Fitness, als auch an der Länge bewertet. Dies verhindert nicht nur Bloat, sondern hält auch eine grosse Bandbreite von Individuen verschiedener Grösse und Fitness aufrecht.

Es wird angenommen, dass die kleinen Individuen häufig Teillösungen des eigentlichen Problems darstellen und als Bausteine für die komplette Lösung verwendet werden können. Ob diese Annahme richtig ist, soll in dieser Arbeit untersucht werden.

1. Einführung in die Genetische Programmierung

Einleitung

Mittels evolutionären Algorithmen lassen sich Lösungen für Probleme finden, die aufgrund ihres zu grossen Suchraumes nicht auf klassischem Wege lösbar sind. Dabei wird eine ganze Gruppe von möglichen Lösungen (eine *Population* von *Individuuen*) parallel untersucht.

Bei der genetische Programmierung (GP) sind die Individuen Programme. Deren Evaluation liefert eine klar definierte Zahl zurück, die sog. *Fitness*. Anhand dieser werden, inspiriert von der Evolution der Lebewesen, die besten Individuen ausgewählt (Selektion) und mittels Kreuzung und Mutation verändert.

Genetische Operatoren

Die Anfangspopulation wird mithilfe dreier genetischer Operatoren transformiert. In diesem Abschnitt sollen diese kurz vorgestellt werden.

1. Kreuzung

Der Kreuzungsoperator kombiniert das genetische Material von zwei Elternteilen. Dabei werden folgende Schritte ausgeführt (*siehe Abbildung 1.1*):

- **Auswahl der Eltern**: Aus der Population werden zwei Elternteile ausgewählt.
- **Bestimmen der Teilbäume**: In beiden Elternteilen wird ein Teilbaum bestimmt (rot eingefärbt).
- **Austauschen der Teilbäume**: Die beiden Teilbäume werden ausgetauscht. Dadurch entstehen zwei Nachkommen.

Notation: Wird von den Eltern eines Individuums gesprochen, so wird unterschieden zwischen Empfänger-Elternteil und Spender-Elternteil. Die Abbildung 1.1 zeigt die Nomenklatur aus Sicht des ersten Kindes. Der Teilbaum, welcher vom Empfänger-Elternteil entfernt wird, wird in dieser Arbeit „weggeschnittener Teilbaum“ genannt, der hinzugefügte Baum entsprechend „hinzugefügter Teilbaum“.

2. Mutation

Mutation wird auf einzelne Individuen angewandt und läuft wie folgt ab:

- **Bestimmen eines Teilbaums**: Im Individuum wird wie bei der Kreuzung zufällig ein Teilbaum bestimmt.

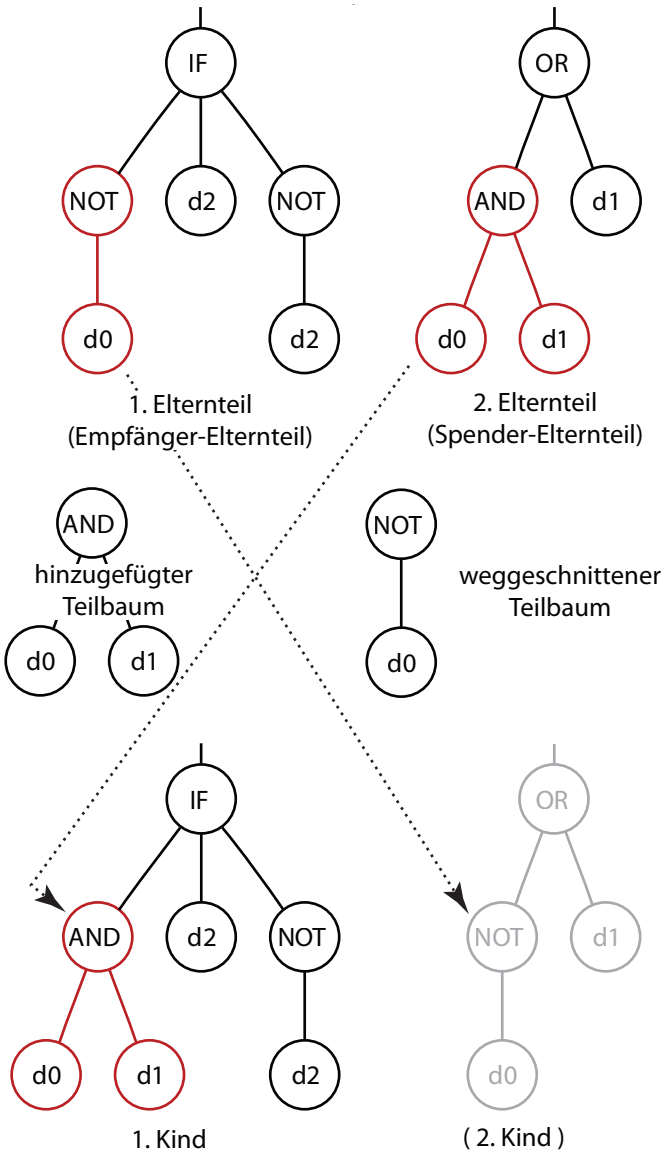


Abbildung 1.1: Illustration der Kreuzungsoperation. Die beiden rot eingefärbten Teilbäume werden ausgetauscht, dadurch entstehen die zwei Nachkommen.

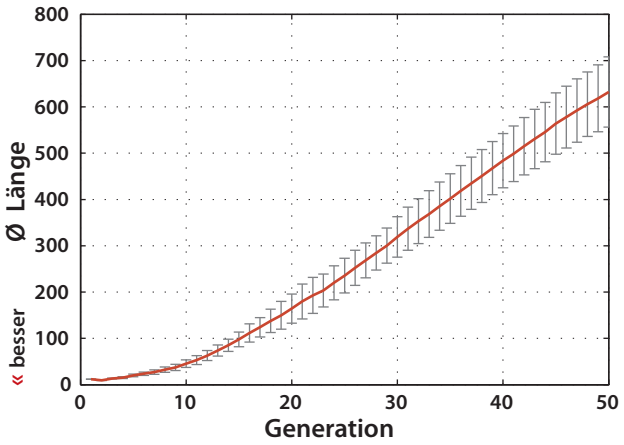
» **Ersetzen des Teilbaums**: Der Teilbaum wird durch einen zufällig generierten Baum ersetzt.

3. Reproduktion

Die Reproduktion kopiert ein ausgewähltes Individuum und platziert es in der Population.

Bloat

Aus Gründen, die im Abschnitt *„Gründe für das Programm-Wachstum“* näher erläutert werden, haben die Programme die Tendenz, stark zu wachsen. Die folgende Abbildung zeigt am 7 - Paritätsproblem (siehe Abschnitt *„Verwendete Testprobleme“)*, wie die durchschnittliche Länge der Population (*rote Kurve*) zunimmt. Dabei wurden die Parameter verwendet, auf die im Kapitel *„Simulationsaufbau“* noch näher eingegangen wird. Lediglich die maximale Tiefe der Bäume sowie die maximale Kantenanzahl wurden erhöht, auf 1ØØ bzw. 5ØØØ; dadurch konnte die Population in den untersuchten ersten 5Ø Generationen ungehindert wachsen.



Als graue Fehlerbalken ist die Standartabweichung der Länge eingezeichnet.

Man sieht schön, wie das Längenwachstum bis ungefähr zur 15. Generation zunimmt. Danach bleibt es in etwa konstant, die Individuen nehmen dann im Schnitt um rund 15 Kanten pro Generation zu.

Nachteile von Bloat

Dieses stetige Wachstum der Programmlänge, das sog. *„Bloat“,* hat mehrere Nachteile:

» **Stagnation**: Werden die Individuen immer länger, beginnt die Evolution häufig zu stagnieren. Das heisst, die Fitness der Population verbessert sich nicht mehr.

» **Ressourcen**: Grosse Programme benötigen mehr Prozessorzeit und Speicherplatz.

» **Generalisation**: Kleine Programme lassen sich besser als Bausteine für komplette Lösungen verwenden.

» **Zu lange Lösungen**: Werden Lösungen gefunden, so sind diese länger als nötig. Dies erschwert die Interpretation und benötigt unnötig Ressourcen.

Gründe für das Programmwachstum

Die meisten Erklärungen dafür, dass die Programme im Verlaufe der Generationen wachsen, gehen von der Annahme aus, dass Kreuzungen, welche die Struktur der Eltern wenig verändern, tendenziell fittere Nachkommen hervorbringen als Kreuzungen, welche einschneidende Änderungen vornehmen.

Grund für diese Annahme ist, dass die Mehrzahl der Kreuzungsoperationen im späteren Stadium des Algorithmus Nachkommen hervorbringt, die weniger fit sind, als ihre Eltern. Nachkommen, die den Eltern ähnlich sind, sind somit im Vorteil gegenüber Kindern, die sich stark von ihren Vorfahren unterscheiden.

Die wichtigsten Theorien zum Code-Wachstum sind die Folgenden:

Intron Theorie

Links in Abbildung 1.2 ist eine Lösung für das 2 Bit Paritätsproblem dargestellt. Wie man am vereinfachten Programm (*rechts*) sieht, lassen sich Teile des Baums streichen oder vereinfachen; solche Elemente werden *Intron* genannt. Es lassen sich davon drei Typen unterscheiden:

» **Hierarchische Introns** : Der linke Eingang der beiden IF-Funktionen in Abbildung 1.2 hat keinen Einfluss auf die Ausgabe des Programms. Jede Kreuzung, die diesen Eingang ersetzt, verändert also auch die Fitness des

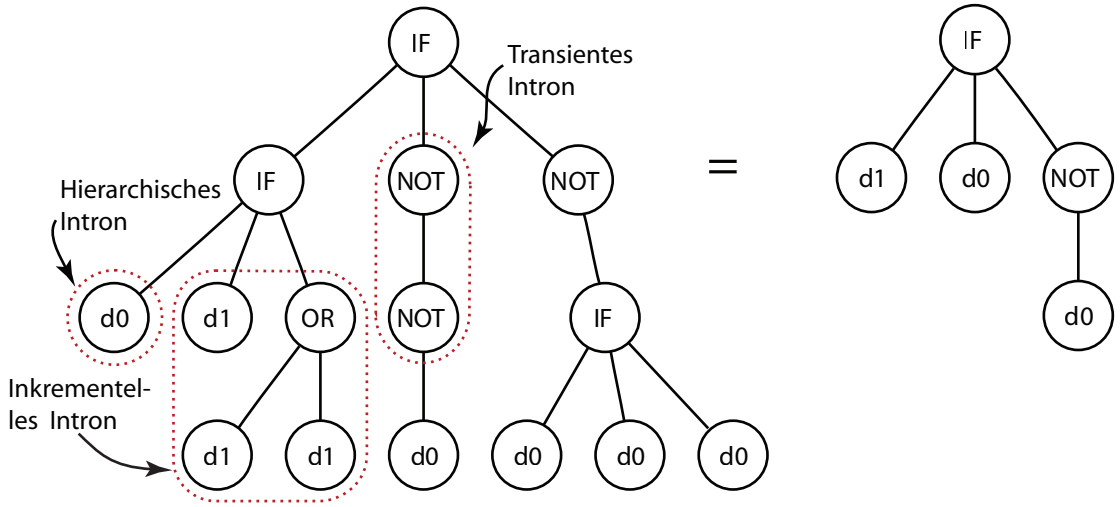


Abbildung 1.2: Der linke Baum lässt sich zum rechten vereinfachen. Rot eingekreist sind die drei im Text erwähnten Arten von Introns.

Individuums nicht (Man nennt solche Teilbäume deshalb auch *„inviable“*). Zusätzlich wird der ganze eingesetzte Teilbaum ebenfalls zum Intron. Hierarchische Introns sind deshalb ein Hauptgrund für Bloat.

» **Transiente Introns**: Der Teil *NOT[NOT[…]* in der Abbildung 1.2 lässt sich komplett durch die Nulloperation ersetzen. Solche sog. *transienten Introns* bieten keinen grossen Schutz vor der Kreuzungsoperation und sind kaum verantwortlich für das Code-Wachstum.

» **Inkrementelle Fitness Introns**: Wird in Abbildung 1.2 ein Eingang der OR-Funktion ersetzt, so hat dies zwar einen Einfluss auf den Programmoutput, allerdings nur, wenn sowohl d_0 als auch d_1 den Wert Null haben. Introns dieser Art sind eine potentielle Ursache für Bloat, wenn auch weniger stark als die transienten Introns.

Theorie des „Removal Bias“

Diese Theorie basiert auf den hierarchischen Introns. Kreuzungen, welche die Fitness des Individuum nicht verändern, produzieren fittere Individuen. Dies ist meist dann der Fall, wenn der entfernte Teilbaum *„inviable“* ist, was häufiger zutrifft, wenn vom Empfänger-Elternteil nur ein kleiner Teilbaum entfernt wird. Der Teilbaum, der anstelle des Introns eingefügt wird, ist hingegen durchschnittlich lang. Die erzeugten Kinder sind somit tendenziell länger als ihr Empfängerelternteil.

Diffusions Theorie

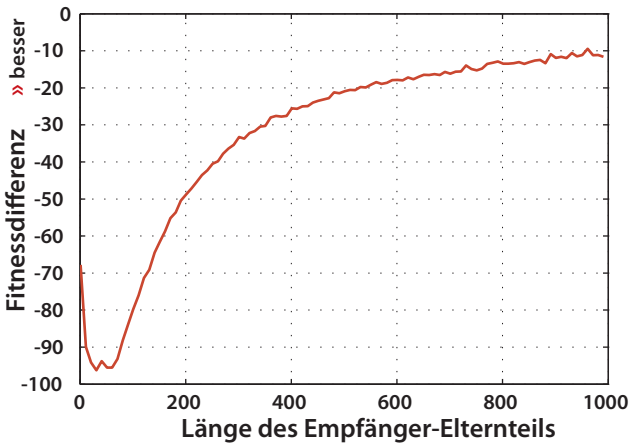
Diese Erklärung für Bloat basiert nicht auf Introns. Sie geht vielmehr davon aus, dass die Population in Bereiche konvergiert, wo die meisten potentiellen Lösungen liegen. Da grosse Bäume mehr Spielraum bieten, wachsen die Programme unaufhörlich [LanØØ, LSP99].

Die „Depth-Correlation“ Theorie

Luke [LØØ-1,LØØ-2] konnte zeigen, dass eine Korrelation besteht zwischen der Tiefe, auf welcher die Kreuzung stattfindet und dem Grad, in dem sich die Fitness des Nachkommen von seinem Empfänger-Elternteil unterscheidet: Je tiefer der Teilbaum entfernt wird, desto besser ist die Fitness des Kindes. Aus dem selben Grund wie in der „Removal Bias“ Theorie kommt es zum Bloat, allerdings ohne den entfernten Teilbaum als *„inviable“* vorauszusetzen.

Zweitens werden bei grösseren Empfänger-Elternteilen die Teilbäume durchschnittlich auf tieferer Ebene entfernt. Dadurch haben gros-

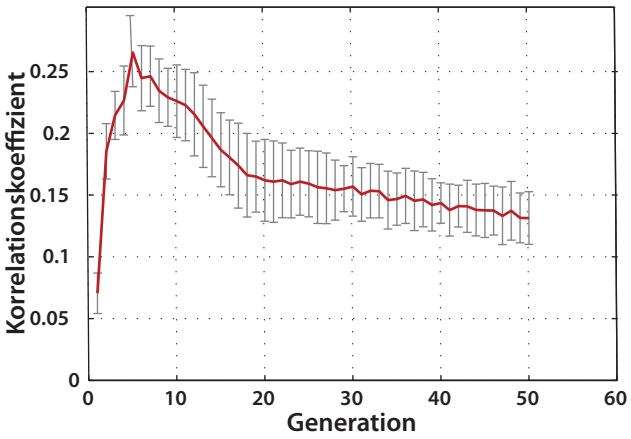
se Empfänger-Elternteile aus oben genannten Gründen auch eine bessere Chance, gute Nachkommen hervorzubringen. Das dies der Fall ist, zeigt folgendes Diagramm:



Die Ordinate zeigt die Differenz zwischen der Fitness des Empfänger-Elternteils und dem ersten Kind; eine negative Fitnessdifferenz steht dabei für ein Verschlechterung der Fitness. Diese Werte sind in Abhängigkeit von der Länge des Empfänger-Elternteils aufgetragen.

Die Fitnessdifferenz fällt für kleine Längen (unter 3Ø Kanten) noch ab, bevor sie zu steigen beginnt. Das hat damit zu tun, dass bei kleinen Empfängerelternteilen die Kreuzung häufig den Grossteil des ersten Elternteils durch das zweite ersetzt. Dadurch unterscheidet sich das Nachkommen nicht gross von seinem Spenderelternteil und ist dadurch tendenziell fitter. Für grössere Längen nimmt dann die Fitnessdifferenz wie erwartet stetig zu.

Die Stärke des Zusammenhangs zwischen Länge des Empfängerelternteils und resultierender Fitnessdifferenz ist abhängig vom Stadium, in dem sich die Population befindet. In dem folgenden Diagramm wurde der Korrelationskoeffizient zwischen den beiden angesprochenen Grössen berechnet und über die gesamte Population gemittelt (rote Kurve). Als graue Fehlerbalken ist zusätzlich die Standartabweichung aufgetragen.



Ein positiver Zusammenhang ist während des ganzen Laufes vorhanden, allerdings ist er etwa in der 5. Generation maximal und nimmt anschliessend ab.

Ansätze gegen Bloat

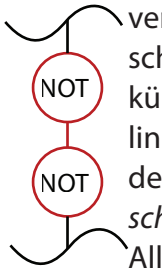
Um Bloat zu verhindern oder mindestens einzuschränken, existieren mehrere Ansätze:

» **Automatisch generierte Funktionen**: Teile des Programms werden „eingekapselt". Solche Teile können durch die Kreuzung nicht geteilt werden. Dadurch soll verhindert werden, das funktionale Blöcke zerstört werden [Koz92].

» **Spezielle Operationen**: Zwei spezielle Operationen, *„size fair"* und *„homologous crossover"*, generieren die Nachkommen, indem sie einen Teil des Programms des ersten Elternteils durch einen sorgfältig ausgewählten, ähnlich langen Teil des zweiten Elternteils ersetzen. Dadurch sind die Nachkommen ähnlich lang, wie ihre Vorfahren [LanØØ].

» **Grössenbeschränkung**: Als zusätzlicher Parameter wird eine maximal erlaubte Grösse der Programme eingeführt. Die Programme wachsen nur noch, bis sie diese Grenze erreicht haben [Koz92].

» **Editing Operation**: In jeder Generation wird versucht, die Programme durch Löschen von unnötigen Teilen zu verkürzen. Zum Beispiel lassen sich im linksstehenden Ausschnitt die beiden NOT Operatoren (*Siehe Abschnitt „Operatoren"*) entfernen. Im Allgemeinen ist es allerdings zu komplex, überflüssige Teile zu erkennen und die Programme zu vereinfachen [Koz92].



» **Constant Parsimony Pressure**: In diesem Algorithmus wird versucht, die Länge des Programms sowie dessen Fitness gleichzeitig zu minimieren. Dies geschieht durch eine Linearkombination der beiden Werte, die Länge wird mit einem konstanten Faktor multipliziert und zur Fitness hinzuaddiert. Der zu minimierende Wert ist also:

$$\text{Zielfunktion} = \text{Fitness} + \alpha \cdot \text{Länge}$$

Eine Schwierigkeit bei diesem Ansatz ist, den Parameter α passend zu wählen. Wird er zu klein gewählt, können die Individuen sehr lange wachsen. Wählt man α hingegen zu gross, erhalten kleine, nicht sehr gute Programme den Vorzug gegenüber längeren Programmen, die das Problem lösen würden [Bli96, ZF99].

» **Adaptive Parsimony Pressure**: Beim adaptiven Parsimony Pressure ist der Parameter α nicht mehr konstant, sondern der Wert einer Funktion g. Festgelegt wird dafür eine Fehlertoleranz ε. Am Anfang des Laufes fliesst die Länge kaum in die Zielfunktion ein. Sobald jedoch die Bäume weniger als ε Eingangskombinationen falsch berechnen, steigt der Längendruck an [ZBØ5].

» **Double Tournament**: In diesem Verfahren werden die Individuen mittels „Tournament"-

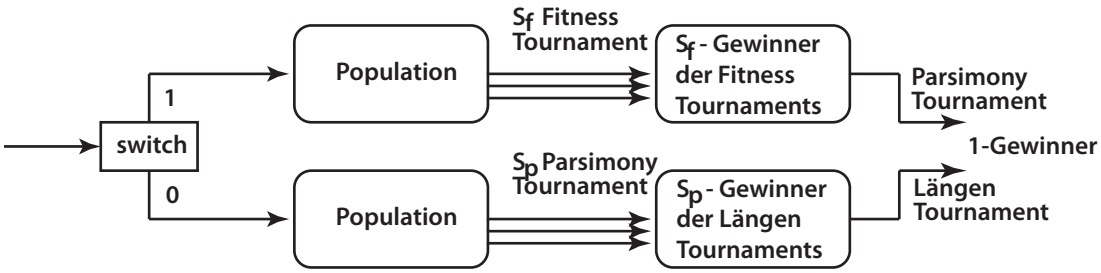


Abbildung 1.3: Dieses Ablaufdiagramm zeigt, wie „Double Tournament" die Individuen aus der Population auswählt.

Selektion ausgewählt. Die Teilnehmer dieses Tournament ist jedoch nicht die gesamte Population, sondern die Gewinner einer zweiten Tournament Selektion. Zuerst wird S_f-mal ein Tournament veranstaltet, bei dem jeweils ein Gewinner anhand der Fitness bestimmt wird. Aus diesen S_f Individuen wird dann der Gewinner anhand der Länge bestimmt (Parsimony Tournament). Falls der Parameter *switch* Ø ist, werden zuerst S_p Gewinner aus „Parsimony Tournaments" bestimmt und anschliessend der finale Gewinner über die Fitness (*siehe Abbildung 1.3*) [LPØ2].

» **Proportional Tournament**: Bei diesem Verfahren werden die Individuen über ein normales Tournament ausgewählt. Ein fester Parameter R entscheidet dabei, ob beim Tournament die Länge oder die Fitness ausschlaggeben ist. [LPØ2].

» **Pseudo-Hillclimbing**: In diesem Verfahren wird die Fitness der Nachkommen direkt mit der der Eltern verglichen; falls die Fitness nicht besser ist als die der beiden Vorfahren, wird das Kind durch das fittere der beiden Elternteile ersetzt. Weil die meisten Kreuzungen die Fitness verschlechtern, können frühe Individuen auf diese Weise sehr lange in der Population verbleiben. Weil frühere Individuen eher klein sind als die, wegen des Bloat Phänomens gewachsenen, späteren Individuen, wächst die Durchschnittslänge weniger schnell [Hag93].

» **Explizit definierte Introns**: Bei dieser Technik findet die Kreuzung und Mutation nicht an zufällig ausgewählten Knoten statt, sondern nur an speziellen Positionen im Baum [NFB96].

Mehrzieloptimierung

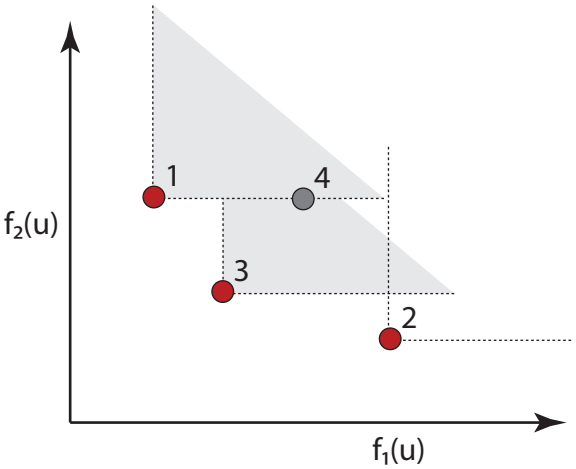
Pareto-Dominanz

In diesem Abschnitt soll nun auf die sogenannte *Mehrzieloptimierung* näher eingegangen werden. Diese behandelt die beiden Ziele, fitte und kleine Individuen, gleichgestellt. Es resultiert dadurch keine eindeutige optimale Lösung, sondern die sog. *Pareto-optimale Menge*. Diese enthält die Individuen, für die es kein anderes gibt, welches nicht in mindestens einem der Ziele unterlegen wäre. Dies lässt sich mit der sog. *Pareto-Dominanz* formalisieren:

Sollen die Funktionen $f_1(x)$, $f_2(x)$, …, $f_n(x)$ optimiert werden, dann *dominiert* ein mögliche Lösung *u* ein Element *v*, falls gilt:

$$\forall i \in \{1,2,...,n\},\ f_i(u) \leq f_i(v) \text{ und}$$
$$\exists i \in \{1,2,...,n\},\ f_i(u) < f_i(v)$$

In der folgenden Abbildung wird der Punkt 4 vom Punkt 1 und 3 dominiert, nicht jedoch von Punkt 2.



Das *Pareto Optimum* besteht aus den nicht dominierten Punkten, im obigen Beispiel die Punkte 1,2 und 3.

Ausserdem *dominiert u* eine Lösung *v schwach*, falls gilt:

$$\forall i \in \{1,2,...,n\},\ f_i(u) \leq f_i(v)$$

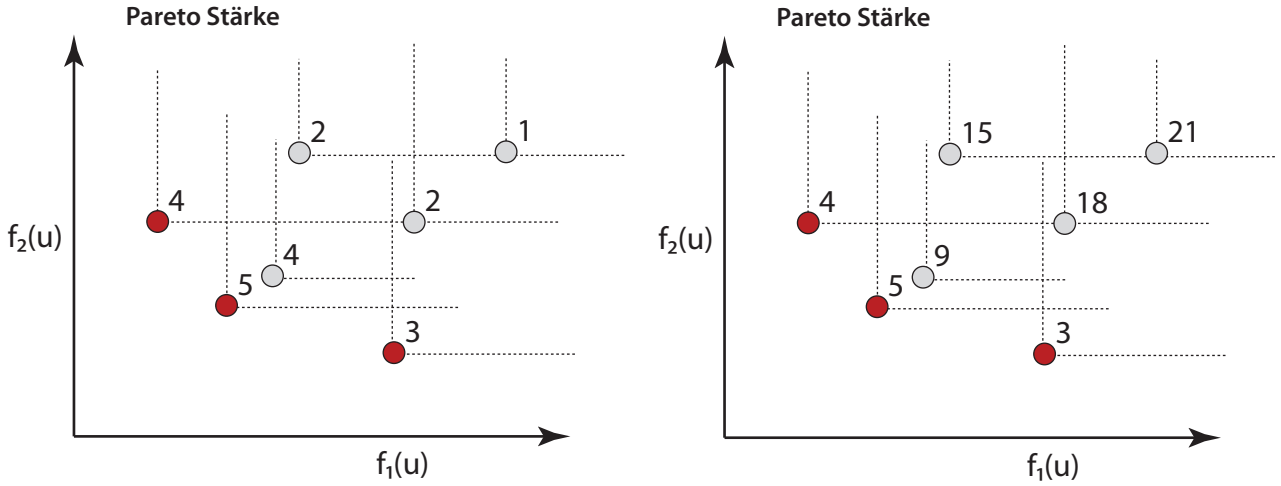


Abbildung 1.4: Illustration zur Berechnung der Pareto-Fitness in SPEA2. Links abgebildet ist die Berechnung der Pareto Stärke. Diese werden anschliessend zur Pareto-Fitness addiert (rechts). Rot eingefärbt sind die Punkte der Pareto-Front

Jeder Punkt dominiert sich, und alle anderen Punkte mit denselben Funktionswerten, also schwach.

SPEA2

Zunächst wird in SPEA2 die Stärke jedes Individuums bestimmt; sie beschreibt, wie viele Punkte ein Individuum schwach dominiert. Da jedes Individuum mindestens sich selbst schwach dominiert, ist die Stärke deshalb ein Wert ≥ 1. Abbildung 1.4 links zeigt die Pareto-Stärken für eine Beispiel Population mit zwei Zielfunktionen.

Die Pareto-Fitness eines Individuums wird nun bestimmt, indem die Stärke-Werte aller Punkte addiert werden, von denen es schwach dominiert wird (*siehe Abbildung 1.4 rechts*). Damit ist die Pareto-Fitness ≥ Pareto-Stärke ≥ 1. Kleine Pareto-Fitnesswerte erhalten die Individuen, für welche gilt:

» **1**: werden von möglichst wenigen anderen Punkten schwach dominiert.

» **2**: dominieren selbst möglichst wenig andere Punkte schwach.

Das führt dazu, das einerseits die Pareto-Front approximiert wird (erster Punkt), andererseits aber auch Nischen (d.h. Orte mit geringer Dichte an Punkten) gefüllt werden (zweiter Punkt).

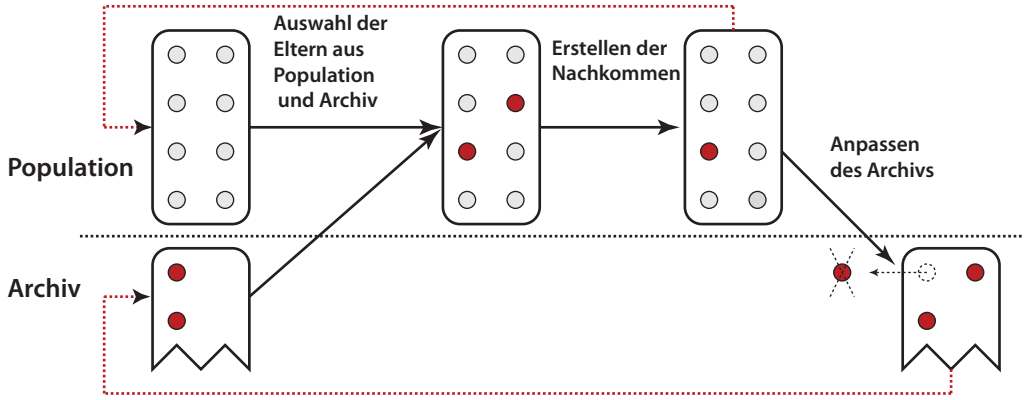


Abbildung 1.5: Ablauf des SPEA2 Algorithmus. Die Kreise bezeichnen Individuen, rot eingefärbt sind dabei die Punkte auf der Pareto-Front, Im Beispiel dominiert einer der Nachkommen ein Individuum des externen Archivs und wird an dessen Stellen ins Archiv aufgenommen.

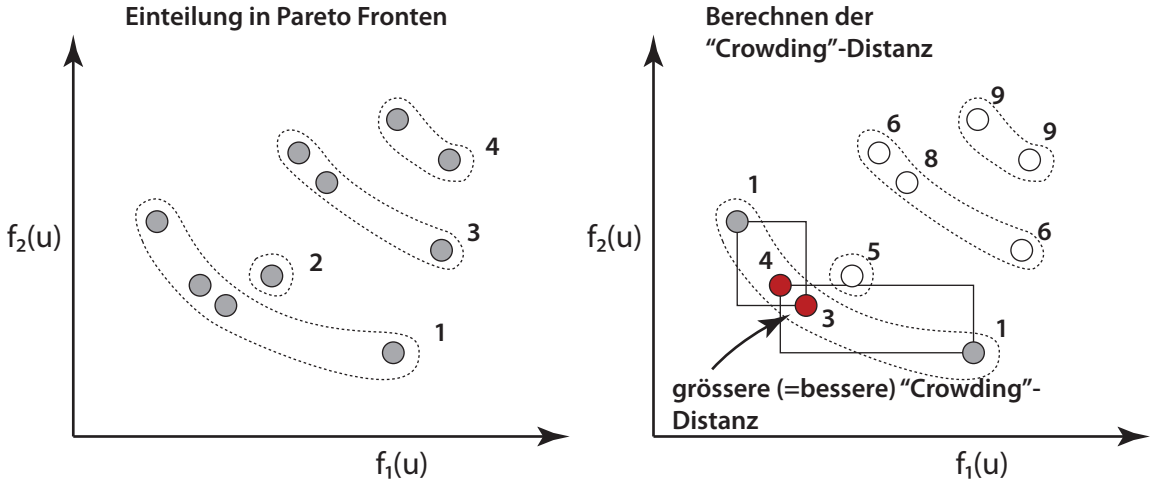


Abbildung 1.6: Ordnen der Individuen in NSGA-II. Zuerst werden die Punkte in Pareto-Fronten eingeteilt (links). Anschiessend folgt innerhalb dieser eine Abstufung nach „Crowding" - Distance (rechts), für die beiden rot eingefärbten Individuen ist die Berechnung durch Rechtecke angedeutet. Die Zahlen im rechten Bild geben die finale Ordnung wieder.

Der Algorithmus läuft wie in Abbildung 1.5 gezeigt ab. Aus dem externen Archiv und der Population werden die Individuen für die Paarung ausgewählt. Anschliessend wird das Archiv angepasst, dieses umfasst alle Individuen der Pareto-Front. Seine Grösse kann somit auch abnehmen. Für den Fall, dass die Anzahl Individuen im externen Archiv die vorgesehene Grösse überschreitet, sieht SPEA2 einen Clustering Algorithmus vor, um das Archiv zu beschneiden. Für die Simulationen in diesem Bericht wurde die Archivgrösse allerdings so gross gewählt, dass dies nie nötig war - weshalb hier auch nicht näher auf das Verfahren eingegangen wird.

NSGA-II

NSGA-II verwendet ein zweistufiges System, um die Individuen einer Population nach „Qualität" zu ordnen:

» **Einteilung in Fronten**: Die Individuen der Paretofront werden in die erste Kategorie (mit erste Paretofront bezeichnet) eingeteilt und aus der Population entfernt. Von dieser reduzierten Population wird erneut die Paretofront bestimmt – diese Individuen wandern in die zweitbeste Kategorie (die zweite Paretofront) und werden wiederum aus der Population entfernt. Diese Schritte werden

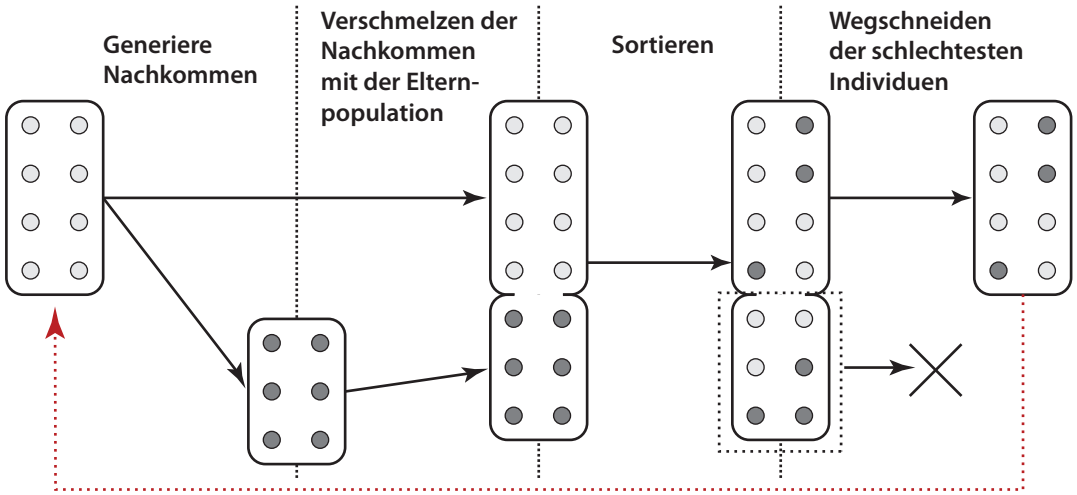


Abbildung 1.7: Ablauf der NSGA-II Algorithmus.

solange wiederholt, bis alle Individuen in Kategorien sortiert sind.

» **Ordnen innerhalb der Kategorie**: Innerhalb der Front werden die Individuen nach der sog. „Crowding"-Distanz geordnet; um diese für einen gegebenen Punkt zu berechnen, werden die Abstände der beiden Nachbarpunkte (Also die Punkte mit nächst höherem und nächst kleinerem Wert) für jede Zielfunktion gemittelt. Falls ein Punkt für irgendeine Zielfunktion den Maximal- oder Minimalwert innerhalb der Kategorie besitzt (also nur einen Nachbarn hat), wird seine „Crowding"-Distanz zu unendlich gesetzt. Je grösser die Crowding Distanz eines Individuums ist, desto besser.

Der NSGA-II Algorithmus läuft nun wie in Abbildung 1.7 gezeigt ab. Die Elternpopulation wird wie oben beschrieben geordnet. Mittels Tournament Selektion wird dann eine Nachkommen Generation kreiert, wobei die Ordnungsrelation anstelle der klassischen Fitness verwendet wird. Primär entscheidet also, in welcher Paretofront ein Individuum liegt und sekundär, innerhalb einer Front, wie stark es „beengt" ist.

FOCUS

Beim FOCUS Verfahren werden alle, aber auch nur, Individuen der Pareto-Front in der Population belassen. Es verwendet nebst den Zielfunktionen Länge und Fitness auch eine Diversitätsgrösse. Diese berechnet sich als quadrierte Distanz zu den anderen Individuen der Population, wobei sich Distanz auf die Baumstruktur bezieht. (Details zur Berechnung dieser Metrik finden sich in [JWPØ1]).

Der Algorithmus läuft wie in Abbildung 1.8 beschrieben ab. Die durch Kreuzung und Mutation generierten Nachkommen werden alle miteinander verglichen. Falls ein Individuum schon vorhanden ist, wird es zunächst mutiert. Fall es dadurch immer noch nicht einzigartig ist, wird es verworfen. Nach diesem Schritt wird nacheinander für jedes Individuum überprüft, ob es schwach dominiert wird. Fall ja, wird es umgehend entfernt. Durch die sequentielle Prüfung verbleibt von Individuen, die den selben Punkt im Raum belegen, nur das letzte, dass getestet wird.

Da nur die Pareto-Front in der Population verbleibt und von jedem Werte-Vektor nur ein Ver-

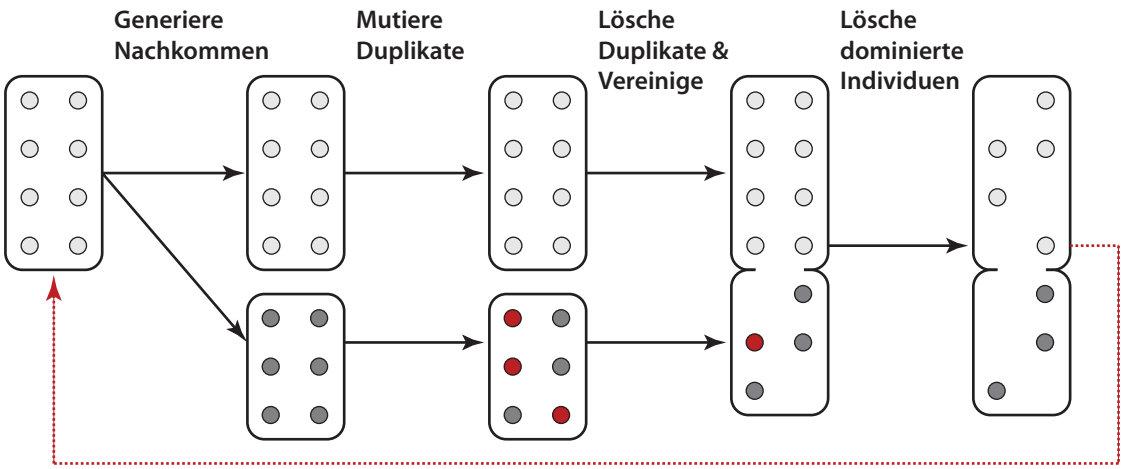


Abbildung 1.8: Ablauf des FOCUS Algorithmus. Aus der Population werden mit Kreuzung, Mutation und Reproduktion Nachkommen produziert. Die Kinder werden mit der ursprünglichen Population verglichen und die Duplikate mutiert (rot). Sind dannach immer noch Duplikate vorhanden, werden diese entfernt. Die verbliebenen Nachkommen werden nun mit der ursprünglichen Population vereinigt. Aus dieser vereinigten Population werden nun sequentiell die schwach dominierten Individuen entfernt.

2. Simulationsaufbau

Verwendete Testprobleme

» **Even-k-Parity**: Die gerade k Paritätsfunktion hat k binäre Eingänge. Sie liefert WAHR (oder 1) zurück, wenn eine gerade Anzahl der Eingänge WAHR sind. Ist einen ungerade Anzahl WAHR, liefert die sie FALSCH (oder Ø) zurück. Die Aufgabe im „even-k-parity"-Problem ist es, eine boolsche Funktion zu finden,

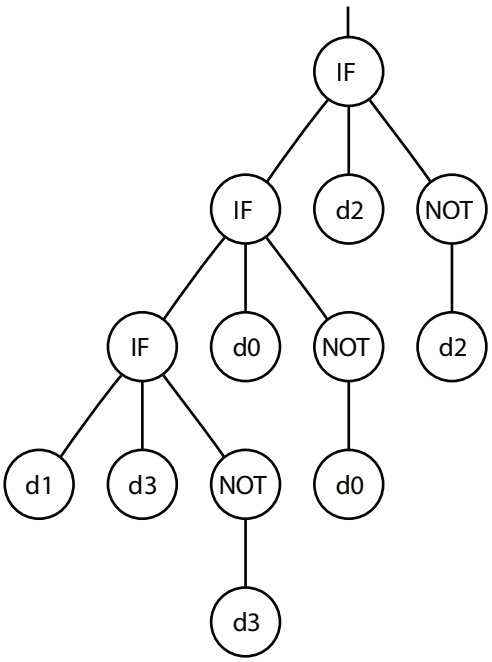


Abbildung 2.1: Beispiellösung für das 4 - Paritätsproblem

welche die beschriebene Paritätsfunktion für jede mögliche Kombination von Eingängen realisiert. Mit den später beschriebenen Operatoren benötigt die kürzeste Lösung je (k-1) IF und NOT Operatoren und (k-1)•4+1 Kanten. Abbildung 2.1 zeigt eine mögliche „even-4-Parity"-Funktion.

» **k-Multiplexer**: Das k-Multiplexer Problem hat *m* Steuer- und *n* Dateneingänge, wobei $n = 2m$ und $k = m + n$, also $k = \{3,6,11,16,37,...\}$. Abbildung 2.2 zeigt einen 6-Multiplexer. Die minimale Lösung benötigt 2^m-1 IF-Operatoren und $3 \cdot (2^m-1)+1$ Kanten.

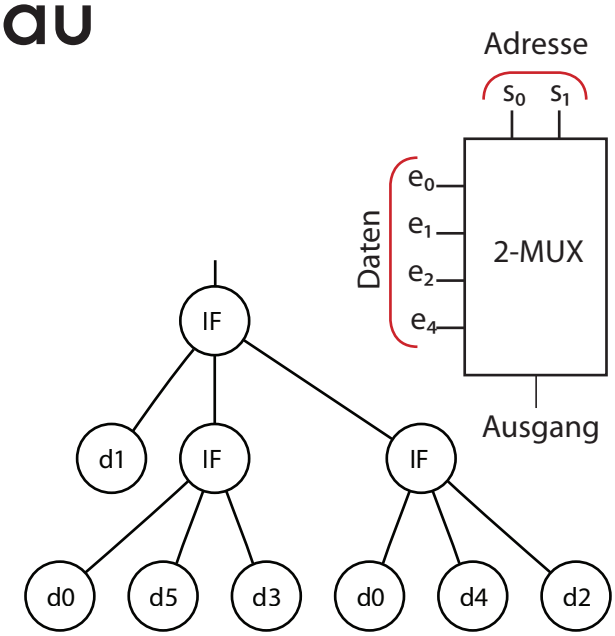


Abbildung 2.2: Kürzester 6 - Multiplexer. Die Terminale dØ und d1 entsprechen den Adresslinien sØ und s1, die Terminale d2 bis d4 den Datenleitungen eØ bis e4

» **k-Hamming Distanz**: Bei diesem Problem soll die Hamming Distanz zwischen der ersten und der zweiten Hälfte der *k* binären Eingänge berechnet werden. *k* muss also gerade sein. Die minimale Lösung benötigt k/2 IF- und NOT-Operatoren, k/2 - 1 Plus-Knoten und 3*k*-1 Kanten. Abbildung 2.3 zeigt eine Lösung des 4-Hamming Problems.

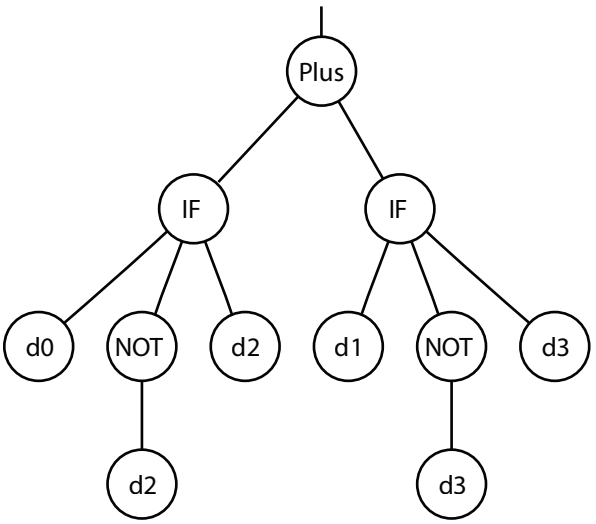
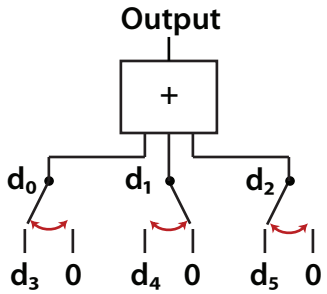


Abbildung 2.3: Beispielbaum für die 4 - Hamming Funktion.

» **k-Addierer**: Dieses Problem ist am Hamming Problem angelehnt, weist aber weniger lokale Minima auf. Die ersten *k*/2 binären Eingänge entscheiden, welche Werte der zweiten Hälfte einer Addition zugeführt werden. Die zweiten k/2 Eingänge haben die Werte {Ø, 1},{Ø, 2}, …, {Ø, $2^{k/2-1}$}. Dadurch bestimmt jeder Eingang der zweiten Hälfte über genau eine Stelle in der binären Darstellung des Ausgangs.



Erläuterung zum 6 - Addierer. Die ersten drei Eingänge, dØ bis d2, entscheiden darüber, welche der Werte d3 bis d5 aufsummiert werden sollen.

Die minimale Anzahl Kanten beträgt (5/2)•k-2. Abbildung 2.4 zeigt eine mögliche Lösung für das 4 - Addierer Problem

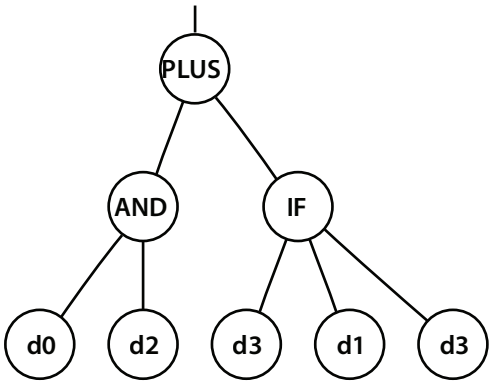


Abbildung 2.4: Beispiellösung für das 4 - Addierer Problem

Knoten

Terminale

Die Terminale werden mit dØ, d2, …, dk-1, bezeichnet. Ausser beim Addiererproblem sind die Eingänge stets Ø oder 1. Um die Anzahl Treffer eines Teilbaums zu bestimmen, wurden alle 2^k Eingangskombinationen durchprobiert und das Ergebnis mit dem der gewünschten Funktion verglichen.

Verwendete Operatoren

In den Paritäts- und Multiplexer-Testproblemen wurden die Operatoren {NOT,AND,OR,IF} verwendet, für das Hamming und das Addierer Problem zusätzlich der Plus-Operator. Im Folgenden sind die Operatoren kurz erläutert (Siehe auch Abbildung 2.5).

» **NOT**: Der NOT Operator hat einen Eingang und liefert 1 zurück, falls dieser den Wert Ø hat. Ist der Eingang 1 (oder allgemein ein Wert ungleich Ø), so wird Ø ausgegeben.

» **AND**: Der AND-Operator liefert genau dann 1 zurück, wenn beide Eingänge ungleich Ø sind. Ist einer der Eingänge Ø, wird Ø zurückgegeben.

» **OR**: Der OR-Operator liefert genau dann 1 zurück, wenn einer der Eingänge ungleich Ø ist. Sind beide Eingänge Ø, wird Ø zurückgegeben.

» **IF**: Der IF-Operator prüft, ob der erste Eingang (dØ) ungleich Ø ist. Falls ja, wird der zweite Eingang (d1) ausgegeben. Falls der er-

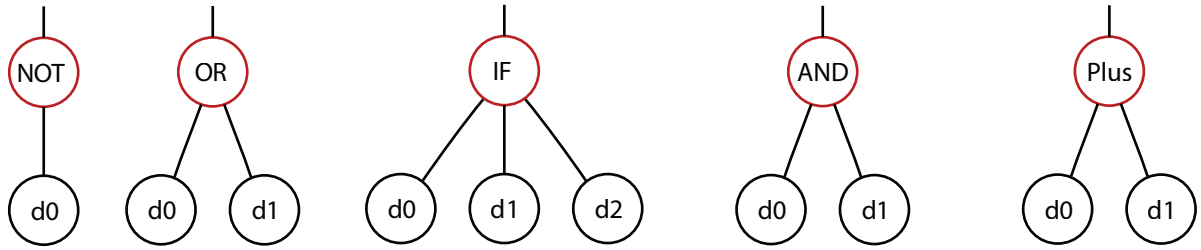


Abbildung 2.5: Die 5 verwendeten Operatoren.

ste Eingang Ø ist, wird der dritte Eingang (d2) ausgegeben.

» **Plus**: Der Plus-Operator addiert seine zwei Eingänge.

Paramater

Falls nicht anders angegeben, wurden in den Simulationen die folgenden Parameter verwendet.

Die Populationsgrösse betrug in den Experimenten 4ØØØ Individuen. Die Läufe wurden nach 2ØØ Generationen abgebrochen. Falls bereits vorher perfekte Lösungen gefunden wurden, wurde der Lauf nicht abgebrochen, so dass sich die Lösungen bezüglich ihrer Grösse noch verbessern konnten. Zusammen mit der Anfangspopulation werden also 8Ø4'ØØØ Individuen untersucht.

Die Bäume der Anfangspopulation haben eine maximale Tiefe von 5; 6Ø% sind dabei vollständig (d.h. die Terminale sind alle auf der selben Ebene). Alle Individuen dürfen eine Tiefe von 2Ø und 7ØØ Kanten nicht überschreiten. In den SPEA2 Läufen wurde eine maximale Archivgrösse von 5ØØØ Individuen festgesetzt, diese Obergrenze wurde nie erreicht.

Die Nachkommen wurden zu 9Ø% durch Kreuzung, und zu 1Ø% durch reine Reproduktion er-

Parametername	**Wert**
Populationsgrösse	4000
Maximale Archivgrösse (Wurde in keinem Spea2 Lauf erreicht)	5000
Anzahl Generationen	200
Abbruchskriterium	Nie
Maximale Tiefe der Bäume (Wurde in keinem Lauf erreicht)	20
Maximale Anzahl Kanten (Wurde in keinem Lauf erreicht)	700
Maximale Anfangstiefe der Bäume	5
Anteil von vollständigen Anfangsbäumen	60%
Kreuzungswahrscheinlichkeit (pc)	90%
Mutationswahrscheinlichkeit (pm)	10%
Reproduktionsmethode	Tournament
„Tournament" - Grösse	10 (für CP), 7 sonst
Operatoren	AND, OR, IF, NOT, (Plus)
Terminale	d0, d1, …, d(k-1)

Tabelle 1: Die Paramtergrössen für die Simulationen. Falls davon abweichende Einstellungen verwendet wurden, so ist dies im Text erwähnt.

zeugt. 1Ø% wurden danach noch mutiert. Die Auswahl der Elternteile erfolgte mittels *„Tournament“*, dessen Grösse in den *„Constant Parsimony“* Läufen 1Ø, für die restlichen Läufe 7 betrug.

Tabelle 1 fasst die verwendeten Parameter zusammen.

3. Baustein - Hypothese

Einleitung

Die Baustein Hypothese (*„Building-Block Hypothesis"*) geht davon aus, dass Individuen, welche einen Teil des Problems lösen, über die Kreuzung kombiniert werden. Dadurch entstehen aus den Teillösungen, den Bausteinen, komplette Lösungen. So könnte aus zwei Individuen, welche das 3-Paritäts-Problem lösen, durch Kreuzung eine 5-Paritäts-Lösung entstehen, wie Abbildung 3.1 zeigt.

Fitnessbereich von kleinen Individuen

Dank der Mehrzieloptimierung sind in SPEA2 Populationen viele kleine Individuen. Primär behaupten sie sich über ihre kurze Länge, um aber ihre Eignung als Bausteine zu untersuchen, interessiert auch deren Fitness.

Solange Individuen alle Terminale verwenden, liegt ihre Fitness im Bereich Ø bis $2^{\#\text{Eingänge}}$. Sobald sie jedoch, und dies ist vor allem bei kleinen Individuen der Fall, nicht mehr alle Eingänge berücksichtigen, reduziert sich der mögliche Fitness-Bereich. Abhängig vom Testproblem wird im Folgenden auf diesen Bereich eingegangen.

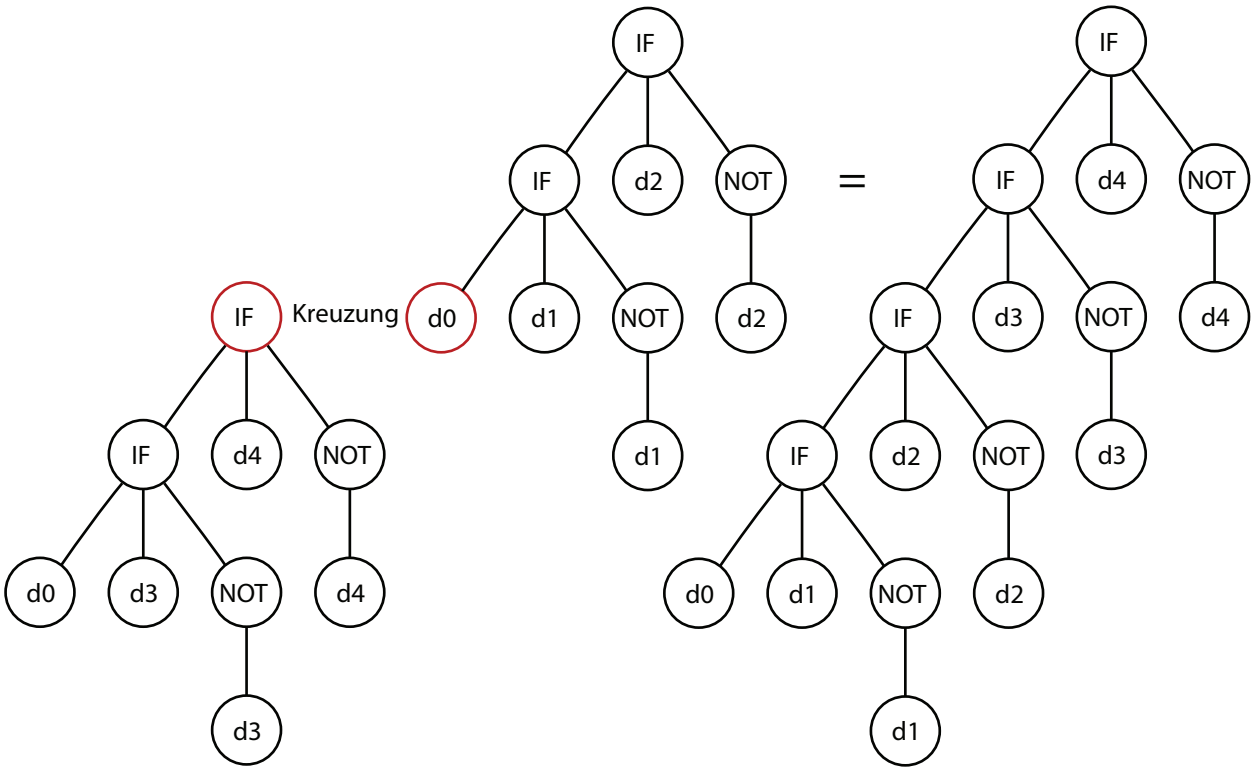


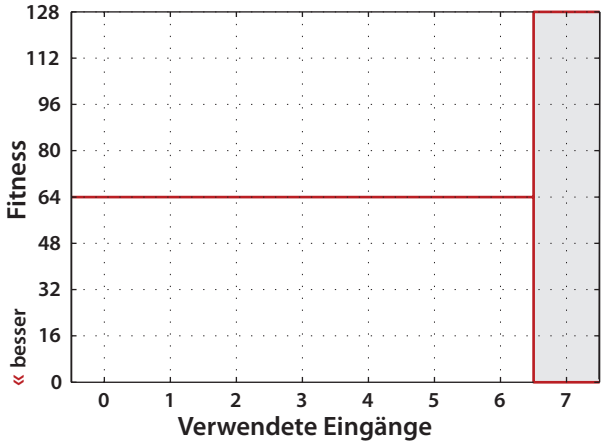
Abbildung 3.1: Durch die Kreuzungsoperation kann der (Teil-) Baum links an die Stelle von dØ im mittleren Baum (das Empfängerindividuum) gesetzt werden. Dadurch resultiert aus den beiden perfekten 3-Paritäts-Funktionen die Lösung für das 5-Paritätsproblem (rechter Baum).

7-Parität

Individuen, welche nicht alle Eingänge verwenden, haben unabhängig von ihrer Struktur immer 5Ø%, also $2^{\#\text{Eingänge}\ -1}$ Treffer. Denn der Ausgang der erwünschten boolschen Funktion ist unabhängig davon, ob ein nicht verwendeter Eingang den Wert Ø oder 1 hat; die gewünschte Parität ist jedoch einmal Ø und einmal 1.

Wird also die Lösung für das *„even-7-Parity"* Problem gesucht, dann wird ein Individuum, welches die gerade Parität von 6 Eingängen perfekt berechnet, kaum in der Population verbleiben. Dies obwohl das Individuum durch eine kleine Ergänzung zu einer Lösung für das komplette Problem werden könnte. Auf eine zweite Eigenheit des Paritätsproblems im Zusammenhang mit der Fitness wird im Abschnitt *„Einfluss der Teilbäume"* eingegangen.

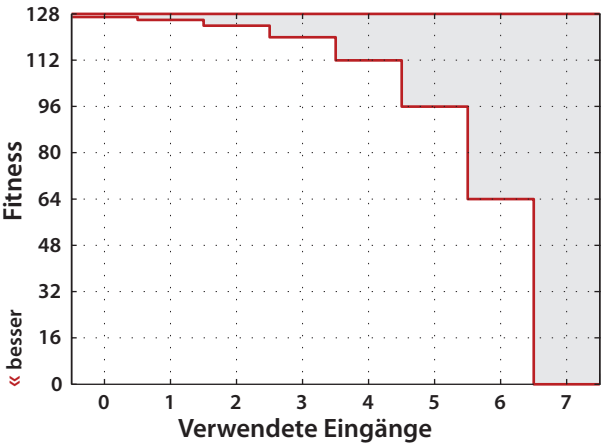
Die folgende Abbildung zeigt diesen Sachverhalt graphisch. Der grau schraffierte Bereich stellt den Fitnessbereich dar.



Um die Qualität der Teilbäume besser einschätzen zu können, wurde ihre Fitness deshalb anders berechnet: Um die Anzahl korrekter Antworten zu bestimmen, werden dabei nur die Eingänge durchprobiert, welche auch wirklich vom Individuum verwendet werden. Dadurch bewegen sich die Fitnesswerte in folgenden Intervallen.

#Eingänge	Minimal	Maximal
0	127	128
1	126	128
2	124	128
3	120	128
4	112	128
5	96	128
6	64	128
7	0	128

Die folgende Graphik zeigt, wie sich der Fitnessbereich dadurch verändert:

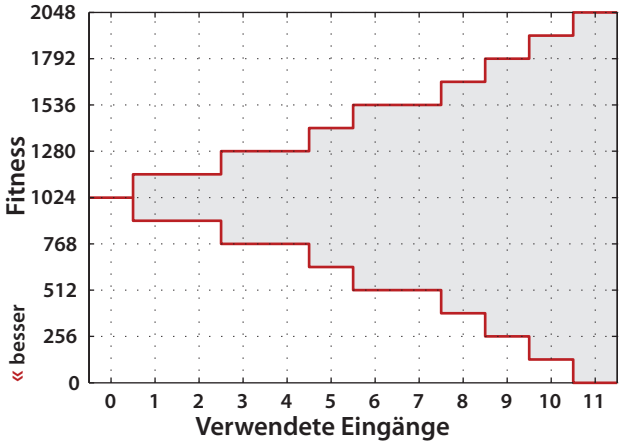


11-Multiplexer

Die Tabelle zeigt, bei welchen Kombinationen von Adress- und Dateneingängen der maximale Fitness Bereich erreicht wird. Fett gesetzt sind die drei gültigen Multiplexer-Lösungen.

#Eingänge	#Addressen	#Daten	Minimal	Maximal
0	0	0	1024	1024
1	0	1	896	1152
2	1	1	896	1152
3	**1**	**2**	**768**	**1280**
4	1	3	768	1280
5	2	3	640	1408
6	**2**	**4**	**512**	**1536**
7	2	5	512	1536
8	3	5	384	1664
9	3	6	256	1792
10	3	7	128	1920
11	**3**	**8**	**0**	**2048**

Die folgende Abbildung zeigt die Fitnessbereiche.



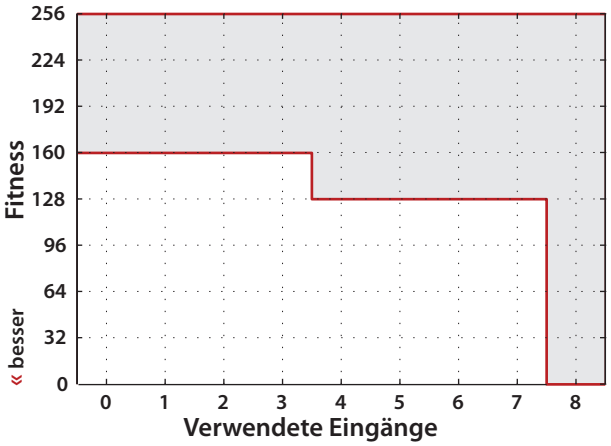
8-Hamming

Da beim Hamming Problem der Plus-Operator zur Verfügung steht, kann jede natürlich Zahl als Ergebnis resultieren. Dadurch ist die minimale Anzahl Treffer, unabhängig von der Anzahl Eingänge, immer Ø und dadurch die maximale Fitness $2^{\#\text{verwendete Eingänge}}$.

Die minimale Fitness erhält man bei 4-7 Eingängen, indem das 4-Hamming Problem gelöst und zum Ergebnis 1 hinzugezählt wird. Dadurch werden 5Ø% der Eingangskombinationen richtig berechnet. Sobald weniger als die Hälfte der Eingänge in die Berechnung einfliessen, erreicht der konstante Wert „2“ mit ⅜ richtigen Antworten die beste Fitness.

#Eingänge	Minimal	Maximal
0	160	256
1	160	256
2	160	256
3	160	256
4	128	256
5	128	256
6	128	256
7	128	256
8	0	256

Graphisch sehen die Bereich so aus:

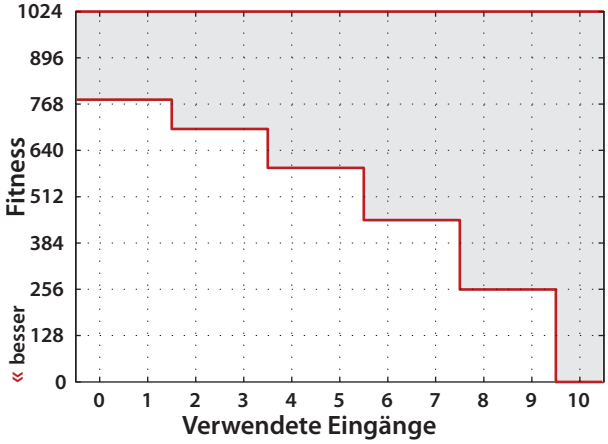


1Ø-Addierer

Aus demselben Grund wie beim Hamming Problem liegt auch beim Addierer die minimale Anzahl Treffer bei Ø und damit die maximale Fitness bei $2^{1Ø}$. Die minimale Fitness hingegen nimmt gleichmässiger ab. Für *e* Eingänge wird das entsprechende $\left\lfloor \frac{e}{2} \right\rfloor$ Addierer Problem gelöst. Die nicht berücksichtigten $n := (k - \left\lfloor \frac{e}{2} \right\rfloor)$ Eingänge liefern in $\left(\frac{3}{4}\right)^{n/2}$ der Fälle den Beitrag Ø, so dass der berechnete Wert korrekt ist.

#Eingänge	Minimal	Maximal
0	781	1024
1	781	1024
2	700	1024
3	700	1024
4	592	1024
5	592	1024
6	448	1024
7	448	1024
8	256	1024
9	256	1024
10	0	1024

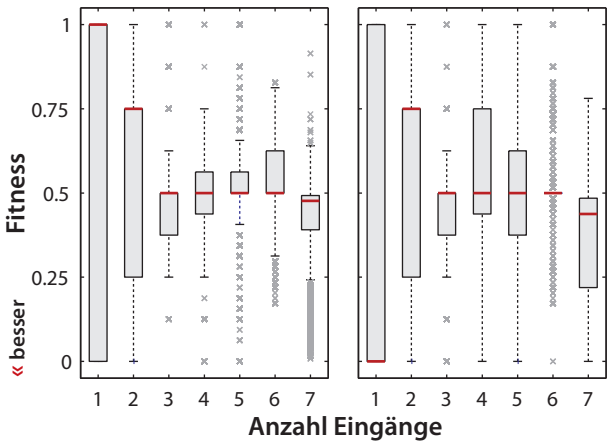
Bildlich sehen die Bereich wie folgt aus:



Fitness der ausgetauschten Teilbäume

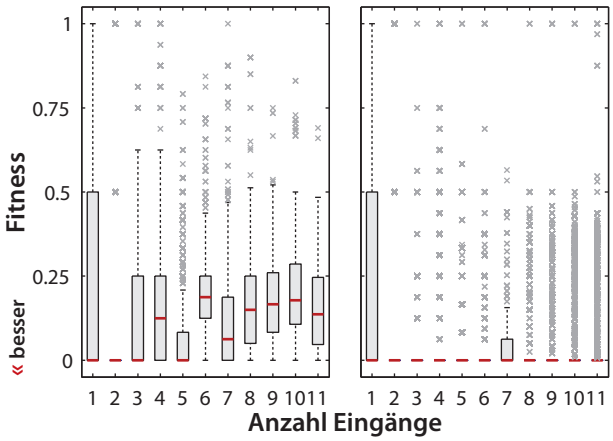
Bei der Kreuzung wird vom ersten Elternteil ein Teilbaum weggeschnitten und durch einen Teilbaum des zweiten Elternteils ersetzt. Abhängig von der Anzahl Eingänge soll nun angegeben werden, wie gut die Fitness dieses zweiten Teilbaums ist. Dazu wird die Fitness auf die im vorherigen Abschnitt beschriebenen Bereiche normiert, um die Qualität besser abschätzen zu können. Die Statistiken wurden dabei einmal für die frühe Phase des Algorithmus erstellt, in der noch kein perfektes Individuum in der Population ist, und einmal für die zweite Phase, in der bereits perfekte Individuen gefunden wurden.

7 - Parität



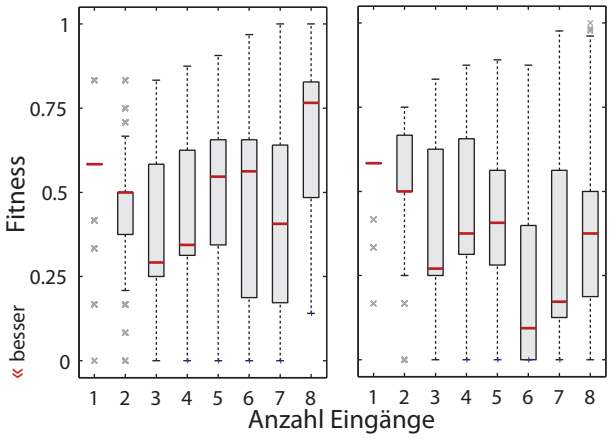
Der Median der Fitness der Teilbäume weichen beim 7-Paritäts Problem kaum von 5Ø% ab. Ausserdem ist nur für Teilbäume mit mehr als 3 Eingänge ist ein kleiner Unterschied zwischen der Verteilung in der ersten und der zweiten Phase zu erkennen.

11 - Multiplexer



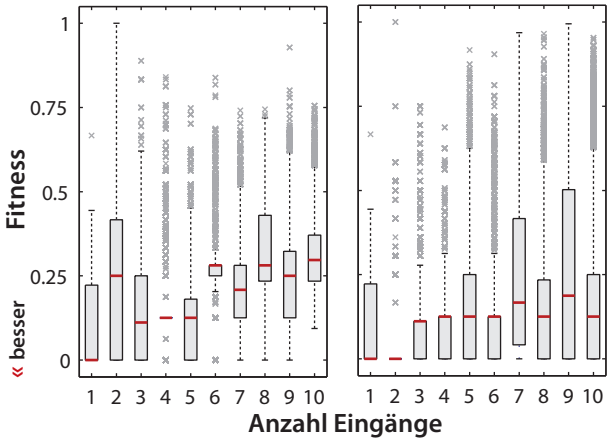
Bereits in der ersten Phase ist die Fitness der ausgetauschten Teilbäume sehr gut. ¾ der Bäume mit mindestens zwei Terminalen erreicht 75% Treffer. In der zweiten Phasen erreicht dann ein Grossteil der Teilbäume die minimal mögliche Fitness.

8 - Hamming



Die mittlere Fitness der Bäume mit 3 und 4 Eingängen ist schon in der ersten Phase der Läufe deutlich besser als Ø.5. In der zweiten Phase sind dann auch die Teilbäume mit 5 oder mehr Eingängen deutlich fitter als 5Ø%. Insbesondere der Median des 6er Teilbaums ist mit rund Ø.1 sehr niedrig.

1Ø - Addierer



Im Vergleich zu den Hamming Läufen sind die Teilbäume für das Addierer Problem noch einmal deutlich besser; sowohl in der ersten, als auch der zweiten Phase.

Einfluss der Teilbäume

Nachdem im vorangegangenen Abschnitt die Verteilung der Fitness der ausgetauschten Teilbäume diskutiert wurde, soll nun näher auf deren Einfluss eingegangen werden. Dabei wird unterschieden zwischen den weggeschnittenen Teilbäumen, die vom Empfänger-Elternteil entfernt werden und den Teilbäumen, welche stattdessen eingefügt werden.

Die Fitness Differenz berechnet sich als (Fitness des Empfänger-Elternteils) minus (Fitness des ersten Nachkommens). Werte grösser als Ø entsprechen also eine Verbesserung der Fitness.

7 – Parität

Weg: Die Fitness des Nachkommen ist stark abhängig vom weggeschnittenen Teilbaum (*Abbildung 3.2 links*). Zwischen Teilbäumen, die eine Fitness kleiner als 64 haben, und der Fitnessdifferenz besteht ein linearer Zusammenhang: Die Fitness des Nachkommens verschlechtert sich umso mehr, je besser die Fitness des weggeschnittenen Teilbaums war. Erzielte dieser z.B. 74 Treffer, so hat das Kind eine um rund 1Ø Treffer (74-64) verschlechterte Fitness. Da Teilbäume alle 7 Terminale enthalten müssen, um überhaupt mehr als 64 Treffer zu erzielen, werden bei dieser Art Kreuzung grosse Teile des Empfängerelterteils entfernt. Diese grossen Veränderungen resultieren meist in einer Fitness von 64 des Nachfolgers (Bis auf den Fall, im dem alle Terminale mindesten doppelt vorhanden waren). Der Teilbaum hingegen entspricht mehr oder weniger dem Elterteil und hat somit auch dessen Fitness. Dadurch resultiert der lineare Zusammenhang.

Für Fitnesswerte grösser als 64 kommt es zu einer Überlagerung von verschiedenen Kurven, die zu den verschiedenen Anzahl Terminalen gehören. Teilbäume mit 6 Eingängen haben beispielsweise eine 5Ø%-Fitness von 96, was zum Kurvenmaximum führt.

Hinzu: Die Fitness des hinzugefügten Teilbaums hat kaum einen Einfluss auf die Fitness des Kindes (*Abbildung 3.2 rechts*). Dies hat mit folgender Eigenheit der Paritätsfunktion zu tun:

Angenommen, das 7-Paritätsproblem soll gelöst werden. Nun sei das Empfängerelternteil die perfekte Lösung, ausser dass von dØ und d1 statt die XOR- die OR-Verknüpfung berechnet wird. Dadurch werden nur ¾ der Eingangskombinationen richtig berechnet, das Individuum hat eine Fitness von 32. Nun werde diese OR-

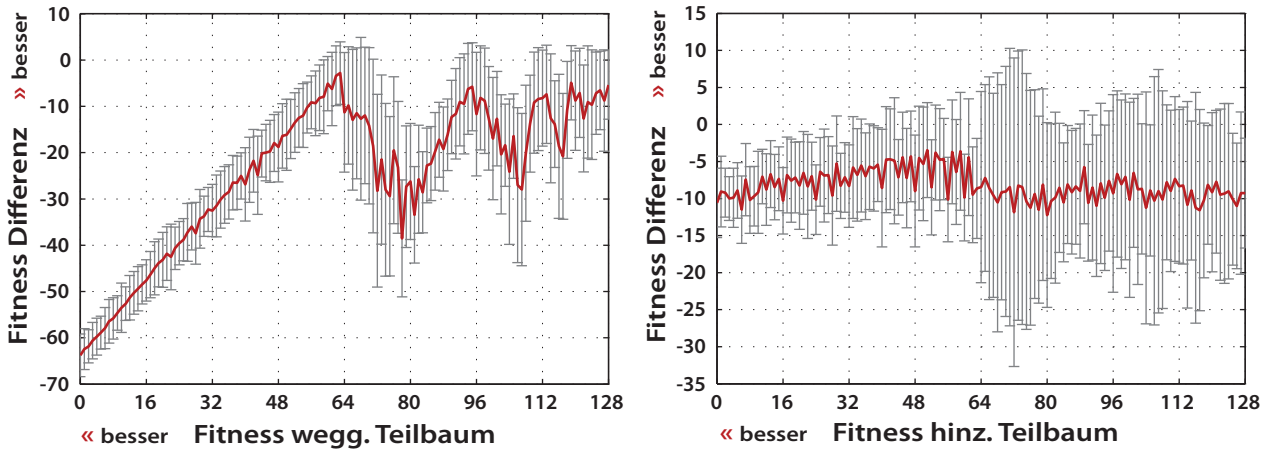


Abbildung 3.2: Das Diagramme zeigen, wie der Einfluss der beiden ausgetauschten Teilbäume auf die Fitness des Nachkommen ist beim 7-Paritätsproblem. Rot eingezeichnet ist der Mittelwert, die grauen Balken zeigen die Standartabweichung.

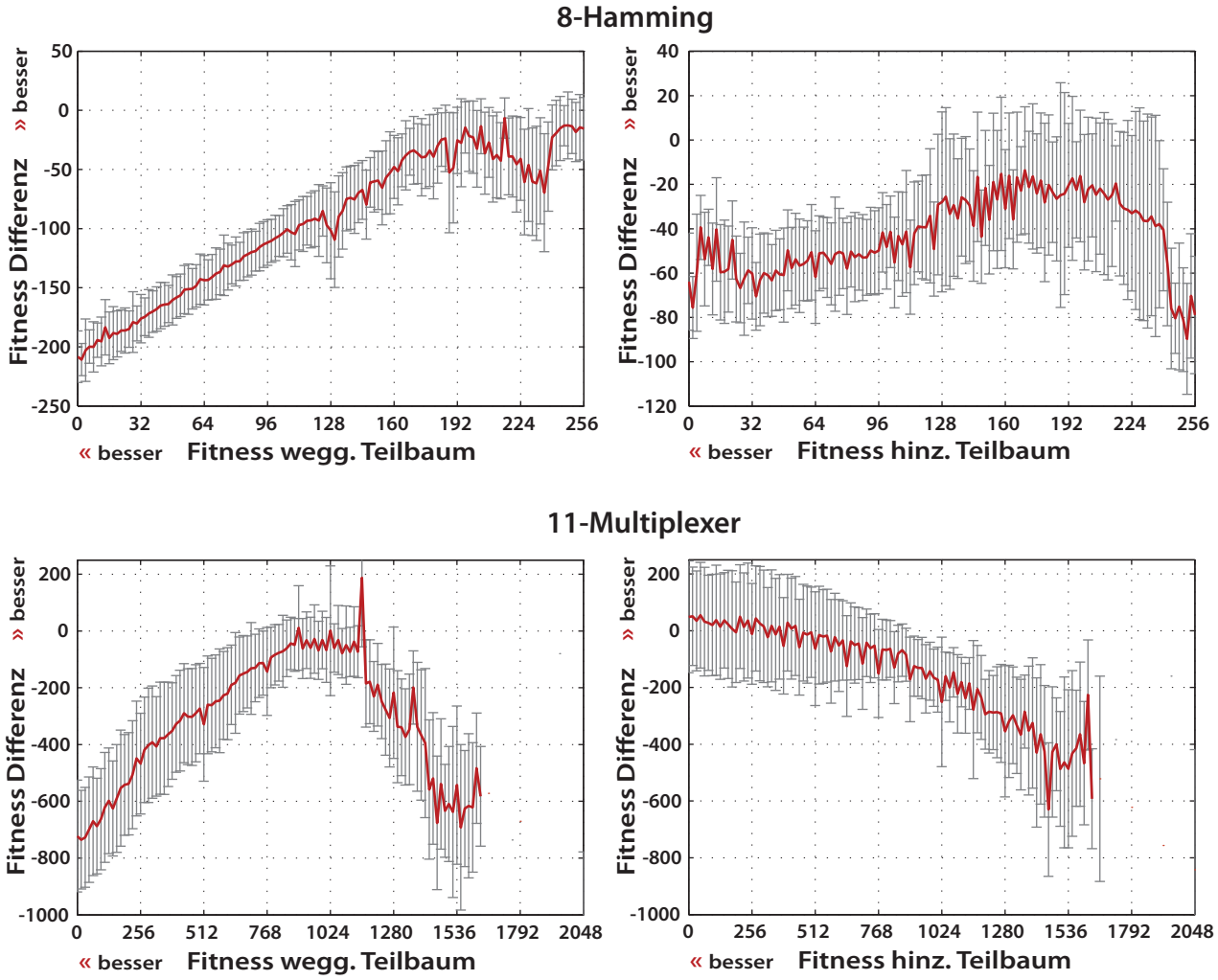


Abbildung 3.3: Die vier Diagramme zeigen, wie der Einfluss der beiden ausgetauschten Teilbäume auf die Fitness des Nachkommen ist. Rot eingezeichnet ist der Mittelwert, die grauen Balken zeigen die Standartabweichung.

Verknüpfung durch Kreuzung durch einen Teilbaum ersetzt, der die 3-Parität von dØ, d1 und d2 perfekt berechnet. Da nun der Eingang d2 zweimal vorkommt, „neutralisiert" er sich. Der Baum hat, aus denselben Gründen wie bei der ersten Paritäts Eigenheit (siehe Abschnitt *„Fitnessbereiche"*), die Fitness 64. Ein perfekter Teilbaum verbessert die Fitness also nur, wenn er genau die Eingänge dØ und d1 verwendet. Die restlichen 254 Kombinationen von Eingängen, die ein perfekter Teilbaum verwenden kann, resultiert in der „neutralen" Fitness 64.

Diese zweite Eigenheit des Paritätsproblem verhindert, dass eine Kombination von Teillösungen häufig von Erfolg gekrönt ist.

11 – Multiplexer

Weg: Wie beim Paritätsproblem verschlechtert sich die Fitness des Kindes umso mehr, je besser die Fitness des weggeschnittenen Teilbaums ist, solange diese mehr als 5Ø% Treffer (1Ø24) erzielt (*Abbildung 3.3 unten links*). Im Unterschied zum Paritätsproblem muss ein Teilbaum nicht alle Terminale benutzen, um eine Fitness unter 1Ø24 zu erzielen, tendenziell sind aber auch hier die Teilbäume mit guter Fitness eher länger und führen zu grösseren und damit destruktiveren Veränderung.

Fitnesswerte von mehr als 1Ø24 treten selten auf. Solche Teilbäume müssen den Ausgang zusätzlich negieren. Da aber NOT-Operatoren für

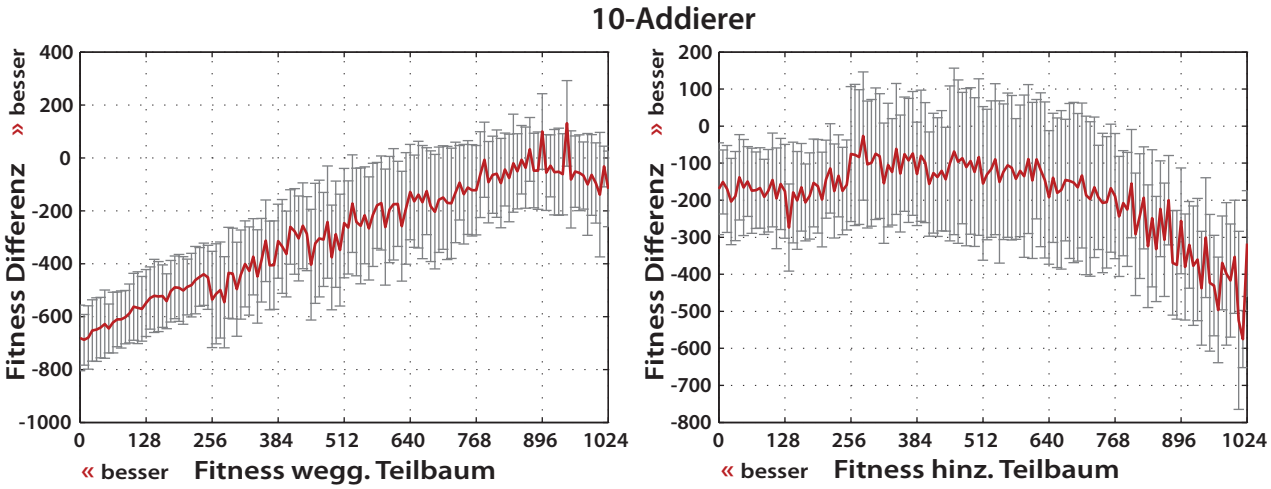


Abbildung 3.4: Das Diagramme zeigen, wie der Einfluss der beiden ausgetauschten Teilbäume auf die Fitness des Nachkommen ist. Rot eingezeichnet ist der Mittelwert, die grauen Balken zeigen die Standartabweichung.

einen perfekten Multiplexer nicht nötig sind, treten diese auch selten auf. Bis auf die Negation sind solche Teilbäume denen mit Fitness kleiner als 1Ø24 ähnlich, weshalb die Kurve auch zur rechten Seite dieses Werts abfällt.

Hinzu: Im Unterschied zum Paritätsproblem hat die Fitness des hinzugefügten Teilbaums deutlich einen Einfluss auf die Qualität des Nachkommen (*Abbildung 3.3 unten rechts*). Wie erwartet, führt eine gute Fitness dabei auch zu guter Fitness im Nachkommen. Der Effekt ist jedoch deutlich schwächer, als der für die weggeschnittenen Teilbäume.

8 - Hamming

Weg: Im Unterschied zu den beiden vorangegangenen Problemen liegt der Anteil korrekter Antworten für ein Individuum mit konstanter Ausgabe nicht bei 5Ø%, sondern im schlechtesten Fall bei Ø Prozent. Die Kurve der Fitnessdifferenz (*Abbildung 3.3 oben links*) hat ihr Maximum entsprechend erst bei hohen Fitnesswerten um 2ØØ. Oberhalb von 128 Fitnesspunkten kommt es zu einer Überlagerung von Teilbäumen mit unterschiedlicher Anzahl Eingänge, weshalb die Kurve zunehmend vom linearen Verlauf abweicht.

Hinzu: Interessanterweise führen Teilbäume mit guter Fitness eher zu einer Verschlechterung, als solche mit hohen Fitnesswerten (*rechte Kurve in Abb. 3.3*). Dies lässt sich dadurch erklären, dass Teilbäume mit guter Fitness eher gross sind und so zu einer starken, und damit tendenziell destruktiven, Veränderung führen. Dieser Effekt ist offenbar stärker, als die eigentlich bessere Qualität der Teilbäume.

Zudem sind in der Population sehr viele Individuen, die den konstanten Wert „2“ ausgeben. (Solche Individuen lassen sich mit nur 9 Kanten realisieren und erzielen einen Fitnesswert von 16Ø, sie gehören damit immer zur Paretofront). Diese Individuen haben mit der Struktur der korrekten Berechnung des Hamming-Werts nicht viel gemein und können deshalb auch nicht durch gute Teilbäume verbessert werden, ausser das Empfänger-Elternteil wird komplett durch den Baum des zweiten Elternteils ersetzt, wodurch es auch dessen Fitness annimmt.

1Ø - Addierer

Weg: Für die weggeschnittenen Teilbäume resultiert ein ähnlicher Fitnessverlauf wie für das Hamming Problem (*Abbildung 3.4 links*)

Hinzu: Im Unterschied zum Hamming Problem fällt die Fitness Differenz wie erwartet ab, je

schlechter die Fitness des hinzugefügten Teilbaums ist, allerdings erst für grosse Fitnesswerte (*Abbildung 3.4 rechts*). Offenbar ist auch hier der Effekt vorhanden, dass gute Teilbäume eher lang sind und dadurch zu grossen Veränderungen führen. Für diese Theorie spricht auch, dass die Kurve beim Fitnesswert 256 absackt. Unter diesem Wert müssen Bäume mindestens 8 Eingänge verwenden und sind entsprechend gross.

Fitness Verlauf

Im vorangegangenen Kapitel wurde untersucht, wie die Fitness der ausgetauschten Teilbäume die Fitness der Nachkommen beeinflusst. In diesem Abschnitt soll nun untersucht werden, wie häufig sich die Individuen der Nachkommen um einen gegebenen Fitnessbetrag verschlechtert bzw. verbessert. Das Zusammensetzen von Teillösungen sollte häufig zu einer sprunghaften Verbesserung der Fitness führen.

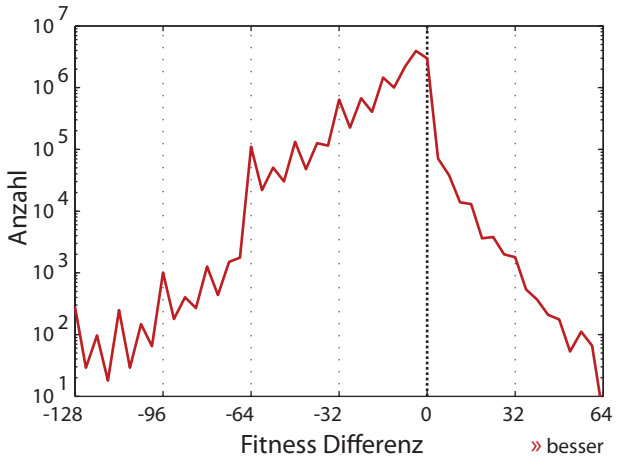
Dazu wurden 25 Läufe pro Lauf analysiert. Die Fitnessdifferenzen wurden in Bereiche aufgeteilt, deren Breite für jedes Problem variiert.

7 - Parität, normal

Die Fitness wurde in Bereiche der Breite 4 eingeteilt. Hauptsächlich aufgrund der zwei erwähnten Probleme sind sprunghafte Verbesserungen der Individuen beim Paritäts Problem sehr selten. Bereits Fitnessdifferenzen von +4 bis +7 treten fast 1ØØ mal seltenere auf, als fitnessneutrale Kreuzungen.

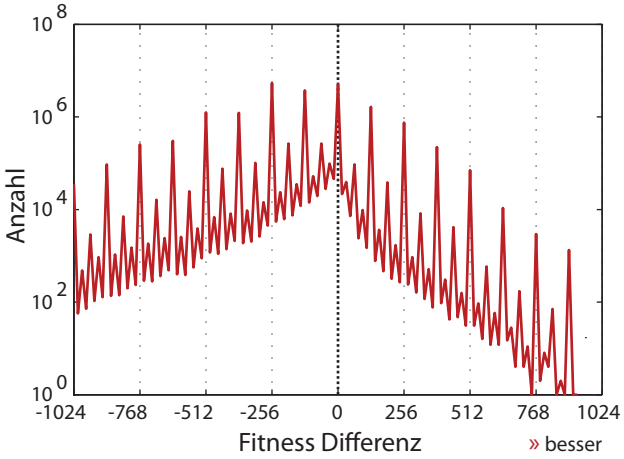
Verschlechterungen treten allgemein weit häufiger auf, als Verbesserungen. Je grösser der Betrag der Fitnessdifferenz, desto seltener tritt sie auf.

Verschlechterungen treten ab einer Stärke von 64 sprunghaft seltener auf.



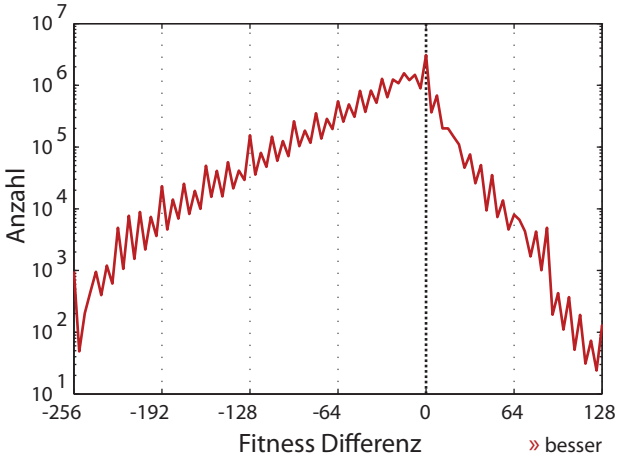
11 - Multiplexer

Beim Multiplexer fällt auf, dass Differenzen, die ein Vielfaches von 128 sind, gehäuft auftreten (gleiches gilt in geringerem Masse für Vielfache von 64). Selbst grössere Verschlechterungen bzw. Verbesserungen um den Betrag dieser Vielfachen sind noch recht häufig. Die Fitnesswerte wurden hier in Bereiche der Breite 16 eingeteilt.



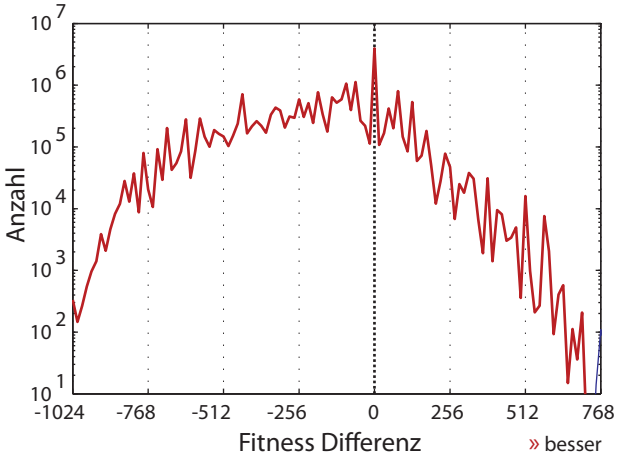
8 - Hamming

Für die folgende Graphik wurde eine Bereichsbreite von 4 gewählt. Die Häufigkeitsverteilung für negative Fitnessdifferenzen nimmt einen ähnlichen Verlauf, wie der des Paritätsproblems. Positive Differenzen treten hingegen häufiger auf.



1Ø - Addierer

Der Verlauf sieht ähnlich aus, wie für das Hamming Problem, wobei die Kurve in die negative Richtung weniger schnell abfällt.

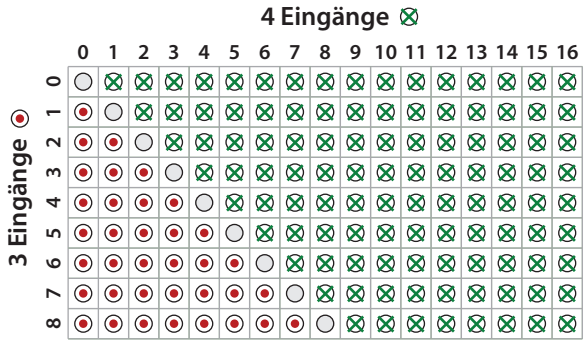


Die Fitnessdifferenzen wurden in Bereiche der Breite 16 eingeteilt.

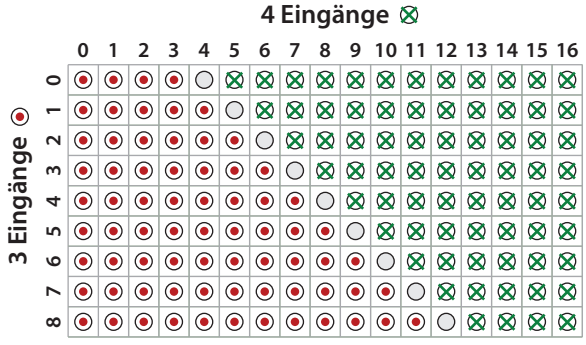
Angepasstes Paritätsproblem

Um die angesprochenen Schwierigkeiten beim Paritätsproblem zu entschärfen, wurde die Berechnung der Fitness der Bäume angepasst. Dabei wurden nur die Eingänge getestet, die auch wirklich im Individuum vorkommen. Das heisst, die Anzahl korrekter Antworten liegt im Bereich Ø bis $2^{\#\text{verwendete Eingänge}}$. Um daraus die Fitness zu berechnen, wurden drei Ansätze getestet. Die drei Illustrationen zeigen dabei, bei welcher Anzahl richtiger Antworten ein Baum mit 3 Eingängen einen Baum mit 4 Eingängen dominiert und umgekehrt.

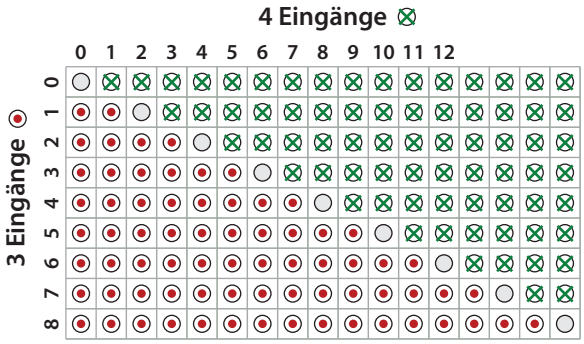
» **1**: Fitness = #korrekte Antworten



» **2**: Fitness = #korrekte Antworten - #falsche Antworten



» **3**: Fitness = #korrekte Antworten/#falsche Antworten



Läufe ohne Kreuzungsoperation

Idee

Als weitere Methode um herauszufinden, ob in SPEA2 Teillösungen zusammensetzt werden, wurden Läufe ohne Kreuzungsoperation durchgeführt und die Ergebnisse mit entsprechenden *„Constant Parsimony"* Läufen verglichen (Der Paramter α betrug dabei Ø.Ø1). Als Testproblem wurde das 7 - Paritätsproblem verwendet.

Falls die bessere Leistung des SPEA2 Algorithmus vor allem auf der Baustein-Hypothese basiert, müssten die Ergebnisse ohne Kreuzung deutlich einbrechen, wohingegen *„Constant Parsimony"* weniger stark betroffen sein dürfte.

Simulationsaufbau

Die drei durchgeführten Simulationen waren (mit jeweils 1Ø Läufen):

» **1**: Im ersten Experiment wurden die Parameter wie im Abschnitt „Simulationsparamter" beschrieben verwendet.

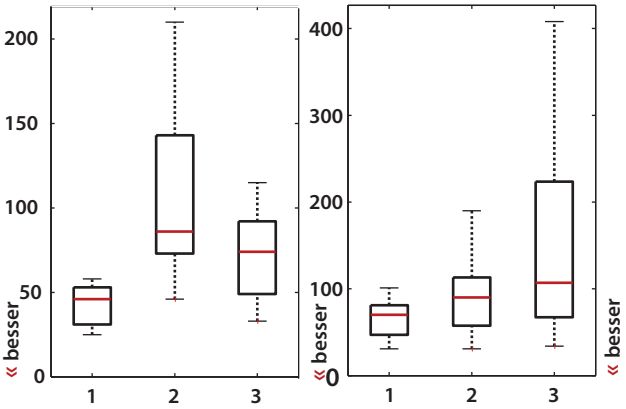
» **2**: In diesen Läufen wurden die Parameter des ersten Experiments verwendet, aber zusätzlich die Mutationswahrscheinlichkeit auf 9Ø% erhöht.

» **3**: Diese Läufe wurden mit den Parameter des zweiten Experiments durchgeführt (also auch mit Mutationswahrscheinlichekeit 9Ø%) aber mit abgeschalteter Kreuzungsoperation.

Ergebnisse

Abbildung 3.5 zeigt die Ergebnisse der Experimente. Wie man erkennen kann, sind finden sowohl *„Constant Parsimony"* als auch SPEA2 ohne Kreuzung fast immer Lösungen. (SPEA2 sogar häufiger, als mit Kreuzung). Allerdings werden die Lösungen deutlich später gefunden (Boxplots).

Die folgende Abbildung zeigt, wie lang das erste perfekte Individuum war. (*links SPEA2, rechts „Constant Parsimony"*)



Hier sieht man, dass „Constant Parsimony" grundsätzlich kürzere Individuen hervorbringt.

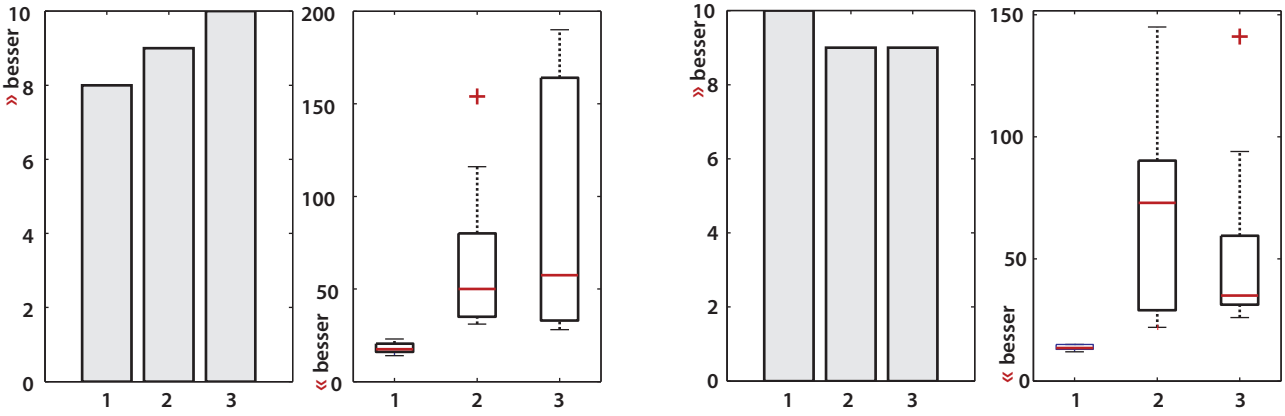
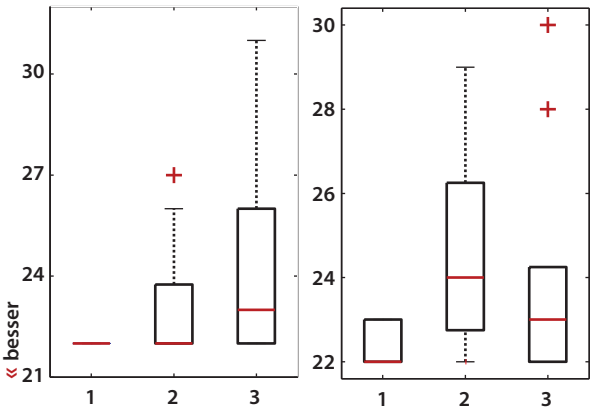


Abbildung 3.5: Balkendiagramme stehen für die Anzahl Läufe, in denen eine korrekte Lösung gefunden wurde. Die Boxplots zeigen, in welcher Generation das erste perfekte Individuum auftrat. Die linke zwei Diagramme sind für einen SPEA2 Lauf, die beiden rechten für einen „Constant Parsimony" Lauf. Die Ziffern stehen für: 1 = normaler Lauf, 2 = Lauf mit Mutationswahrscheinlichkeit 9Ø%, 3 = Lauf ohne Kreuzung und Mutationswahrscheinlichkeit von 9Ø%.

Durch die erhöhte Mutationswahrscheinlichkeit und die fehlende Kreuzung verliert SPEA2 dafür auch stärker.

Als letzter Vergleich ist in den zwei folgenden Diagrammen die Länge des kürzesten, perfekten Individuums aufgetragen.



Auch hier sind sowohl SPEA2 (links) als auch „Constant Parsimony“ (rechts) nicht stark von der erhöhten Mutationswahrscheinlichkeit oder der fehldenden Kreuzungsoperation betroffen.

Fazit

Die Vermutung, dass SPEA2 in den Experimenten 2 und 3 stärker „verliert“ als „Constant Parsimony“, konnte nicht nachgewiesen werden. Offenbar sind die Vorteile von SPEA2 beim Lösen des 7 - Paritätsproblem vor allem die erhöhte Diversität der Individuen in der Population.

4. Oszillationen

Fitness- und Längenverlauf

Im Unterschied zur klassischen genetischen Programmierung können sich Individuen bei SPEA2 nicht nur über die Fitness, sondern auch die Länge profilieren. Dadurch findet sich in der Population eine grosse Diversität: von langen und fitten bis hin zu sehr kurzen, dafür wenig fitten Individuen.

Erklärung zu den verwendeten Diagrammen

Für jedes der vier verwendeten Testprobleme sowie eine Variante des angepassten 7 - Paritäts-Problem wurde die Fitness- und Längenverteilung in Abhängigkeit von der Generation aufgetragen. Die Daten wurden dazu aus jeweils 2Ø Läufen gesammelt. Die Abbildungen zeigen

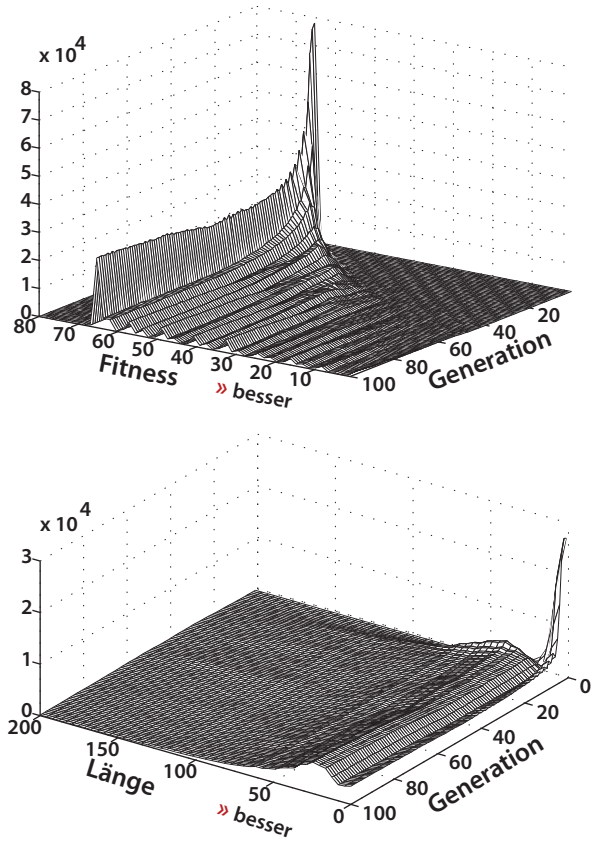


Abbildung 4.1: Die Fitnessverteilung (oben) und die Längenverteilung (unten) in Abhängigkeit der Generationen für das 7 - Paritätsproblem.

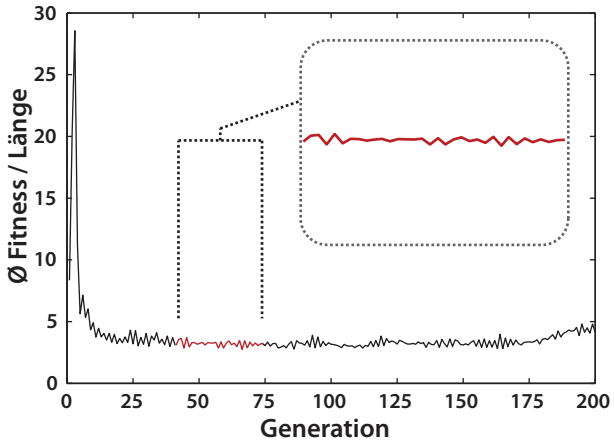


Abbildung 4.2: Mittelwertverlauf der Verhältnisse Fitness zu Länge des 7 - Paritätsproblems

den Verlauf nur für die ersten 1ØØ Generationen, da sich danach die Verteilung kaum mehr ändert. Die Fitness- und Längenwerte wurden in 3Ø gleich grosse Bereich aufgeteilt.

Zusätzlich wurde für alle Individuen das Verhältnis von Fitness zu Länge gebildet. Diese Werte wurde über die gesamte Population gemittelt und nach der Generation aufgetragen. Um Oszillationen hervorzuheben, wurde der rot markierte Ausschnitt zusätzlich dreifach vergrössert dargestellt.

7 - Parität

Fitnessverlauf: Zu Beginn des Laufes haben praktisch alle Individuen einen Fitnesswert von 64. Der Anteil sinkt jedoch relativ schnell auf rund 1/4 der Population. Die restlichen Individuen verteilen sich auf höhere Fitnesswerte, wobei kleine Fitnesswerte selten auftreten.

Längenverlauf: Die Längenverteilung pendelt sich sehr schnell ein, die meisten Individuen sind ähnlich gross wie die optimale Lösung (25 Kanten). Individuen die länger sind als 1ØØ Kanten sind in der Population praktisch nicht vorhanden.

Verhältnis: In der ersten Generation nach der zufällig generierten Initialpopulation schnellt das durchschnittliche Verhältnis auf rund 28 hoch, konvergiert dann aber rasch auf einen Wert von rund 4. Dieser Wert bleibt ohne grössere Schwankungen bis zum Ende des Laufes bestehen.

11 - Multiplexer

Fitnessverteilung: Bereits nach 2Ø Generationen sind in der Populationen perfekte 11-Multiplexer vorhanden. Ihr Anteil nimmt danach jedoch kaum mehr zu.
Es fällt auf, dass die Fitness der Individuen nur wenige verschiedene Werte annimmt. Es handelt sich dabei um die in Abschnitt *„Fitnessbereich von kleinen Individuen"* aufgelisteten Minima für Bäume mit reduzierter Anzahl an Eingängen.

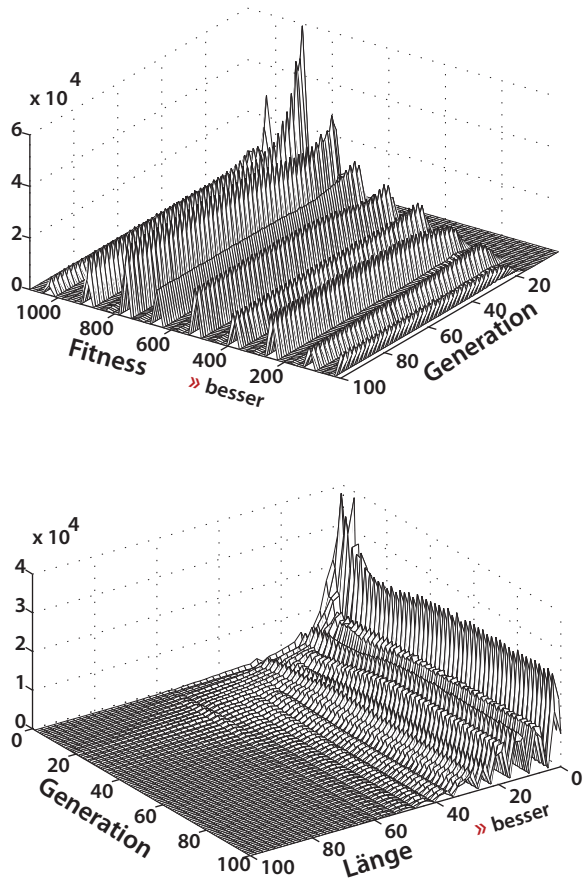


Abbildung 4.3: Die Fitnessverteilung (oben) und die Längenverteilung (unten) in Abhängigkeit der Generationen für das 11 - Multiplexer Problem.

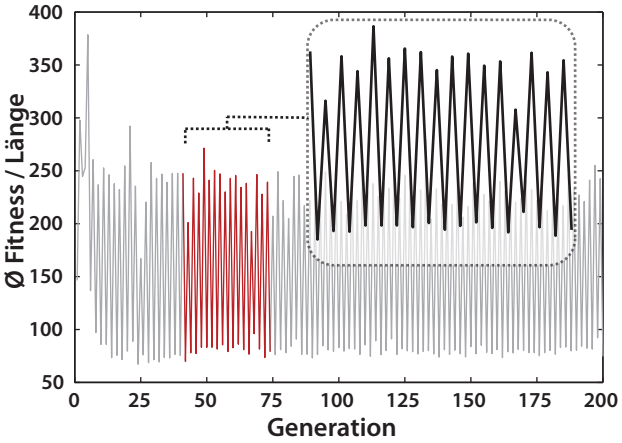


Abbildung 4.4: Mittelwertverlauf der Verhältnisse Fitness zu Länge des 11 - Multiplexer Problems.

Längenverteilung: Auch die Längenverteilung bleibt nach kurzer Einschwingphase praktisch konstant. Wie auch die Fitnesswerte sind die Längen auf ein paar wenige dominante Werte verteilt; wobei besonders häufig kleine Individuen vorkommen.

Verhältnis: Wie man in Abbildung 4.4 sehen kann, oszilliert das Verhältnis von Fitness zu Länge sehr stark. In jeder Generation nimmt das mittlere Verhältnis rund um den Faktor 3 zu, bzw. ab. Man erkennt die Oszillation in Abbildung 4.3, die Kämme sind alle „gezackt".

8 - Hamming

Fitnessverteilung: Die Fitnesswerte verteilen sich über das ganze Spektrum, wobei kleine Werte aber relativ selten auftreten.

Längenverteilung: In der Population ist die besonders viele Individuen mit einer Länge von 1-3 vertreten. Grössere Werte treten mit zunehmend kleinerer Häufigkeit auf.

Verhältnis: Wie beim 11 - Multiplexer oszilliert auch bei diesem Problem das Verhältnis, wenn auch weniger stark. Die Rippel wie in Abbildung 4.3 sind auch in der Abbildung 4.5 für frühe Generationen leicht zu erkennen. Die Werte, zwischen denen das Verhältnis oszilliert, nimmt bis etwa zur 75. Generation ab. Zusätzlich ist

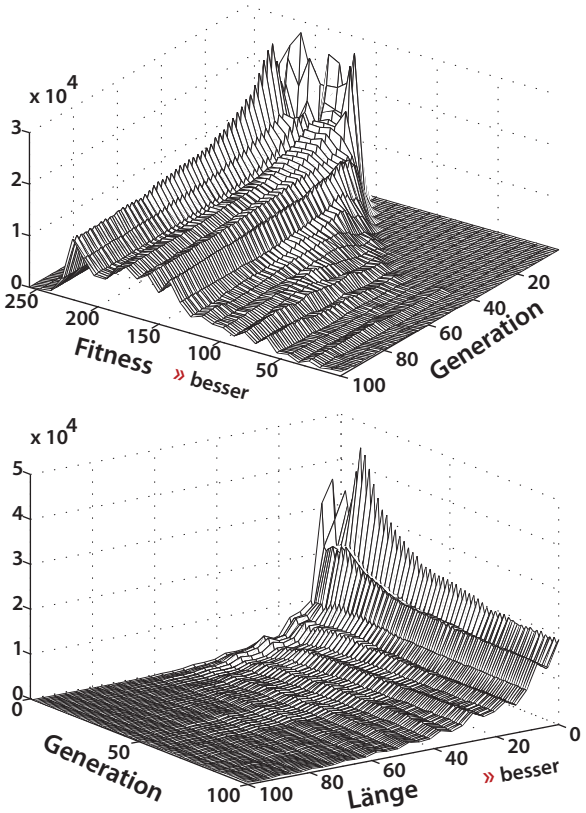


Abbildung 4.5: Die Fitnessverteilung (oben) und die Längenverteilung (unten) in Abhängigkeit der Generationen für das 8 - Hamming Problem.

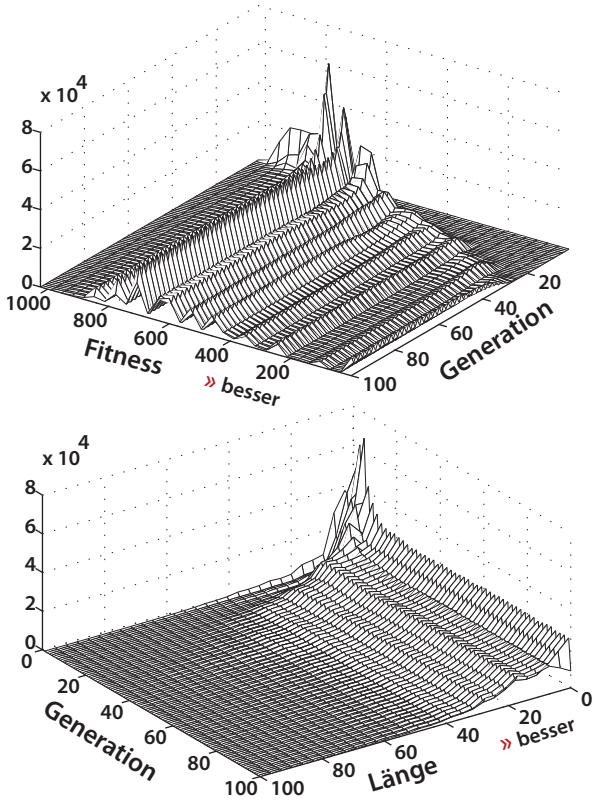


Abbildung 4.7: Die Fitnessverteilung (oben) und die Längenverteilung (unten) in Abhängigkeit der Generationen für das 1Ø - Addierer Problem.

Abbildung 4.6: Mittelwertverlauf der Verhältnisse Fitness zu Länge des 8 - Hamming Problems.

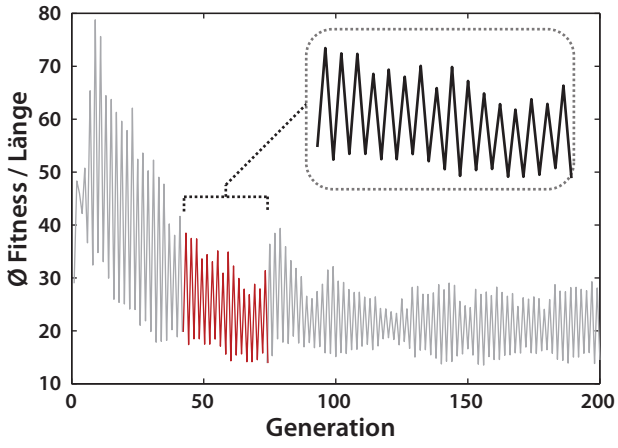
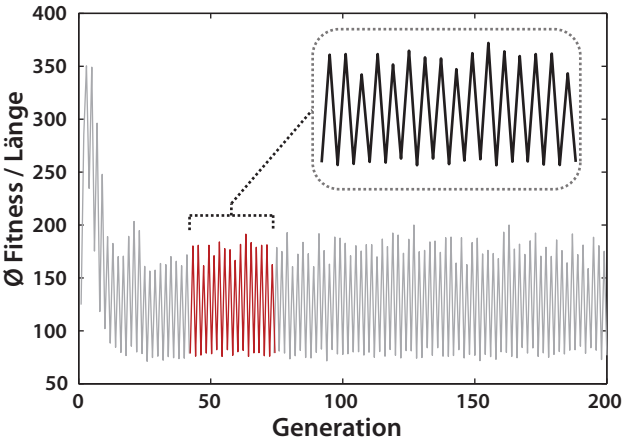


Abbildung 4.8: Mittelwertverlauf der Verhältnisse Fitness zu Länge des 1Ø - Addierer Problems.



das Ausmass der Schwingung unterschiedlich stark.

1Ø - Addierer

Fitnessverteilung: Die Fitnesswerte sind im Unterschied zum 8 - Hamming Problem „flacher" verteilt.

Längerverteilung: Die Längenverteilung ist ähnlich wie die des 8 - Hamming Problems, auch hier treten vorwiegend sehr kurze Individuen auf.

Verhältnis: Auch für das 1Ø - Addierer Problem oszilliert das Verhältnis stark. Der Schwingbereich ist nach kurzer Anfangsphase, in der das Verhältnis kleiner wird, annährend konstant.

7 - Parität, angepasste Variante

Von den 3 Varianten, das Paritätsproblem anzupassen, sei hier nur die erste dokumentiert. (Siehe Abschnitt *„Angepasstes Paritätsproblem"*). Sowohl der Fitness- bzw. Längenverlauf wie auch das gemittelte Verhältnis sieht für alle Versionen ähnlich aus.

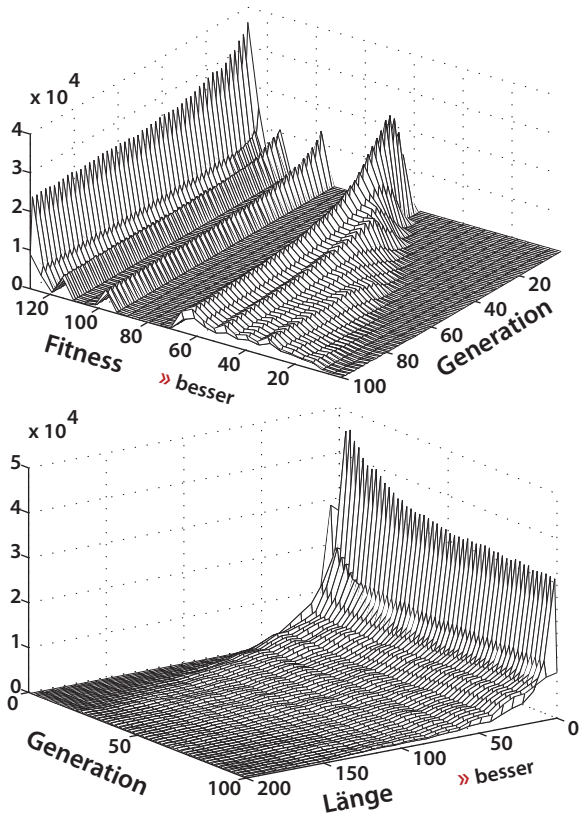


Abbildung 4.9: Die Fitnessverteilung (oben) und die Längenverteilung (unten) in Abhängigkeit der Generationen für das angepasste 7 - Paritätsproblem.

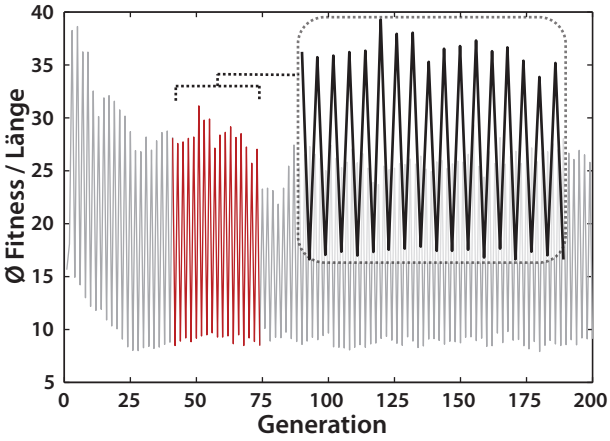


Abbildung 4.1Ø: Mittelwertverlauf der Verhältnisse Fitness zu Länge des angepasste 7 - Paritätsproblem.

Fitnessverteilung: Durch die angepasste Fitness treten nun vermehrt Bäume mit eine Fitness grösser als 64 auf. Unterhalb dieser Marke ist die Verteilung ähnlich wie beim normalen Paritätsproblem.

Längenverteilung: Im Unterschied zum normalen Paritätsproblem treten vor allem sehr kurze Individuen auf. Die grösste Anzahl Bäume ist kürzer als 6 Kanten.

Verhältnis: Durch die Anpassungen der Fitness treten nun auch beim Paritätsproblem sehr starke Oszillationen auf.

Fazit

Bis auf das normale 7-Paritätsproblem treten in allen Testproblemen starke Oszillationen des mittleren Fitness-zu-Länge-Verhältnis auf. Gründe dafür werden im nächsten Abschnitt gegeben.

Gründe für die Oszillation

In diesesem Abschnitt soll näher auf die Gründe für die Oszillationen eingegangen werden. Dazu werden die Fitness- und Längenwerte der Individuen in einem Streudiagramm dargestellt.

Erklärung zu den Streudiagrammen

» **Grösse**: Die Grösse der Punkte beschreibt, wieviele Individuen das entsprechende Wertepaare haben. Dabei ist die Fläche proportional zur Anzahl. D.h. ein doppelt so grosser Radius entspricht einer vierfachen Anzahl. Die folgende Aufstellung zeigt, welche Fläche welcher Anzahl entspricht.

2ØØ, 1ØØ, 5Ø, 1Ø, 3, . 1

» **Farbe**: Die Farbe spiegelt die Pareto - Fitness wieder. Das Spektrum reicht von grün (sehr gute Fitness) über orange zu rot zu schwarz.

7 - Paritätsproblem ohne Anpassung

In der Abbildung 4.11 sind oben die Fitness-Längen-Wertepaare für die Generationen 1Ø2,1Ø3 und 1Ø4 dargestellt. Wie man sieht, kommt es zu einer Häufung der Wertepaare mit einer Fitness von 64 und kurzer Länge. Obschon diese Individuen aufgrund ihrer Länge kaum stark dominiert werden, liegt die Pareto-Fitness im gelben Bereich.

Die Verteilung der restlichen Punkte umfasst wie gewünscht viele verschiedene Wertepaare und ändert sich nicht sprunghaft von einer Geneartion zur andern.

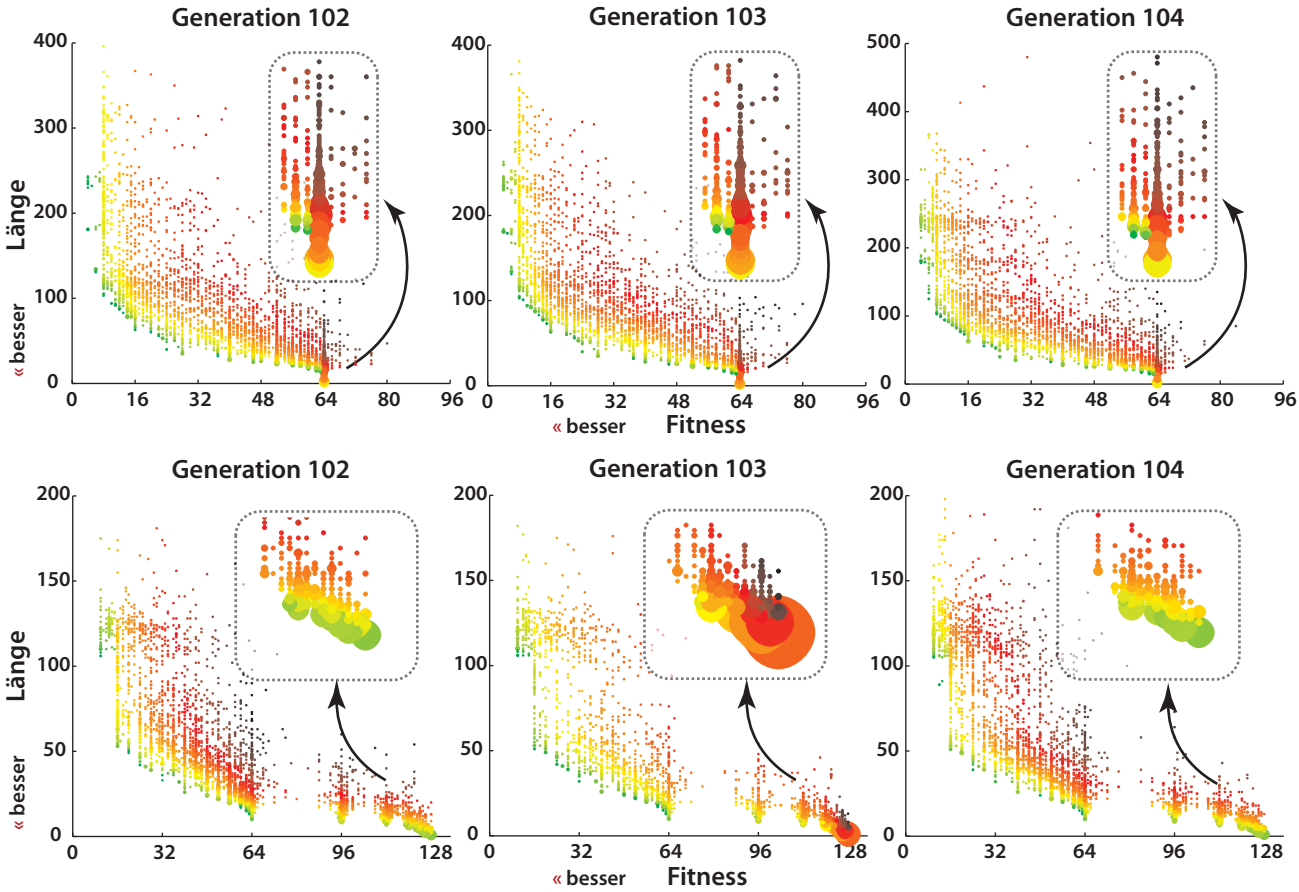


Abbildung 4.11: Verteilung der Individuen des 7 - Paritätsproblem mit normaler (oben) und angepasster Fitness (unten). Die entscheidenden Bereiche wurden zusätzlich dreifach vergrössert dargestellt. Die Farb- und Grössencodierung der Kreise wird im Fliesstext erklärt.

7 - Paritätsproblem mit Anpassung

Abbildung 4.11 unten zeigt die Verteilung der Wertepaare für einen Lauf, in dem die Fitnessberechnung nach der ersten Variante (*siehe Abschnitt „Angepasstes Paritätsproblem“*) angepasst wurde. Die Individuen teilen sich nun entsprechend ihrer Anzahl Eingänge auf, so liegen z.B. die Bäume mit 6 Eingängen vorwiegend rund um den Fitnesswert 96. Diese Verbreiterung in der Längendimension führt aber auch zu einem reduzierten Fitnessbereich – während für das normale Paritätsproblem viele Individuen mit Längen über 2ØØ auftraten sind es in der angepassten Version praktisch keine mehr.

Die sehr kurzen Individuen haben nun nicht mehr alle dieselbe Fitness, sondern sind auf verschiedene Wertepaare rund um 128 aufgeteilt. In der Generation 1Ø2 ist deshalb die Pareto-Fitness der Kreise in diesem Bereich sehr gut. Würden die Kreise alle auf- oder übereinanderliegen, wie dies noch im Paritätsproblem ohne Anpassung der Fall war, wäre die Pareto-Fitness deutlich schlechter.

Die gute Pareto-Fitness im hervorgehobenen Bereich führt dazu, dass sich sehr viele kurze Individuen reproduzieren. Deren Nachkommen sind ebenfalls kurz und haben praktisch immer auch die Fitness 64. Dadurch sind in der 1Ø3. Generation sprunghaft viel mehr Individuen vorhanden, die sich über ihre kleine Länge auszeichnen (An dieser Stelle sei noch einmal erwähnt, dass die Fläche der Kreise der Anzahl Individuen entspricht, eine Verdoppelung des Radius also eine vierfache Anzahl bedeutet.). Diese Zunahme geht auf Kosten der Individuen mit Fitnesswert kleiner 64 (also auch mit allen 7 Terminalen), ihre Anzahl nimmt ab.

Diese extreme Häufung von sehr kleinen Individuen führt nun aber zu einer schlechten Pareto-Fitness, wie man an der roten Färbung erkennen kann. Sie vermehren sich deshalb in der 1Ø3 Generation kaum noch, was wiederum dazu führt, dass die verbliebenenen kleinen Bäume in der 1Ø4 Generation eine sehr gute Pareto-Fitness zugeordnet bekommen.

Die Verteilung der Wertepaare ist ausserdem ähnlich wie die der 1Ø2 Generation - sie schwingt also zwischen den beiden Zuständen der Generationen 1Ø2 und 1Ø3 hin und her.

Massnahmen gegen die Oszillation

In diesem Abschnitt sollen drei Vorschläge gemacht werden, wie sich die Oszillationen reduzieren oder ganz verhindern lassen.

1. Lösung: Anpassen der Fitnessberechnung

Im vorhergehenden Abschnitt haben wir gesehen, dass für das 7-Paritätsproblem ohne Anpassungen keine Oszillation auftritt. Der Hauptgrund dafür ist, dass kleine Individuen alle die gleiche Fitness 64 haben und somit häufig übereinander liegen. Sie erhalten deshalb auch schnell eine schlechte Pareto-Fitness.

Diese Eigenheit lässt sich auf bei den restlichen drei verwendeten Testproblemen ausnützen, indem man die Fitnessberechnung der Bäume anpasst. Dazu wird zuerst bestimmt, ob der Baum alle Terminale verwendet, falls ja, ist seine Fitness unverändert. Falls er jedoch einige Eingänge nicht berücksichtig, wird nicht die Anzahl korrekter Anworten berechnet, sondern abhängig vom Testproblem ein konstanter Wert zugeordnet:

- **11 - Multiplexer**: Die Bäume bekommen die Fitness $2^{11}/2$, also 1Ø24. Dies entspricht der Fitness eines Baumes, der konstante Werte ausgibt.
- **8 - Hamming**: Die Bäume bekommen den maximal möglichen Fitnesswert $2^8 = 256$.
- **1Ø - Addierer**: Die Individuen bekommen ebenfalls den maximal möglichen Fitnesswert, also $2^{1Ø} = 1Ø24$

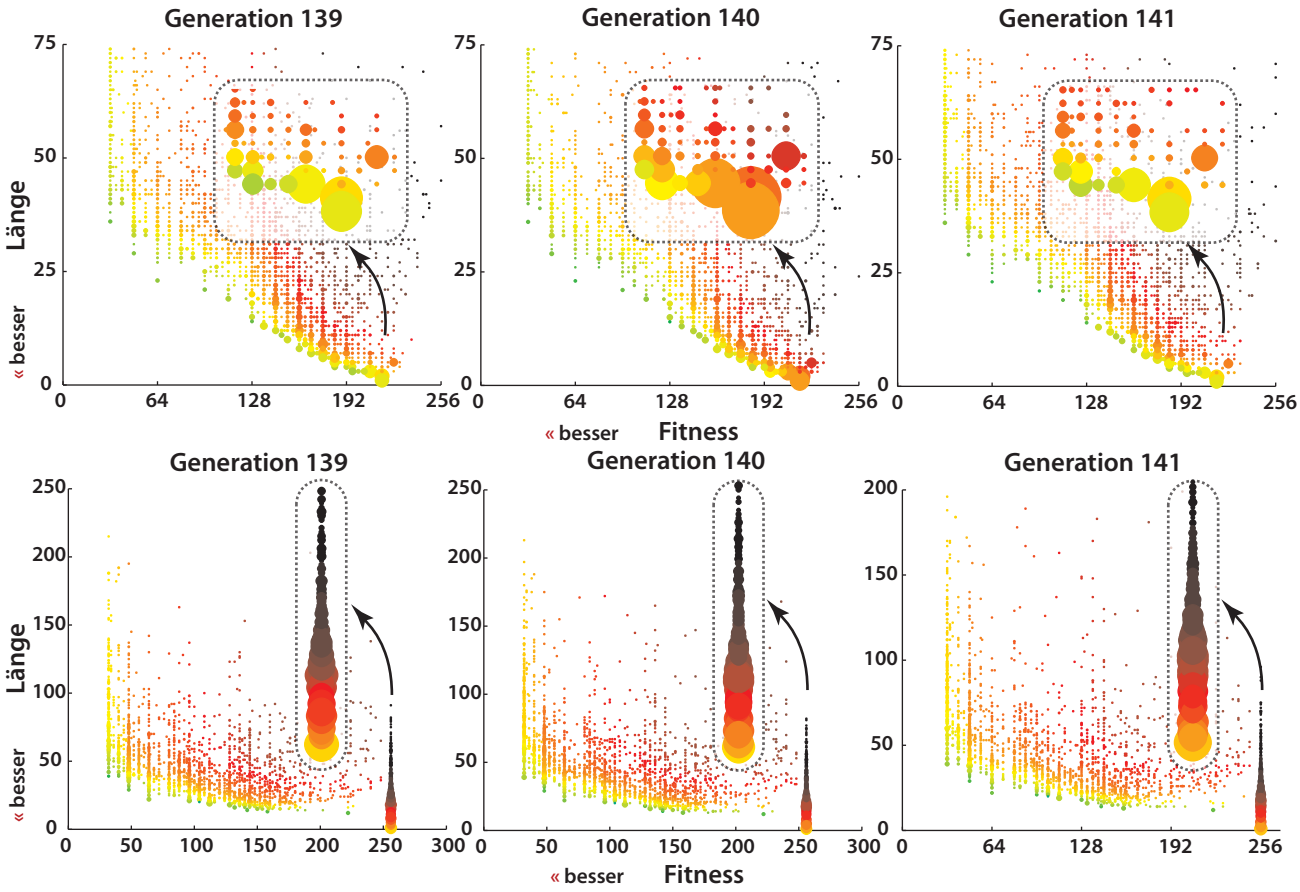
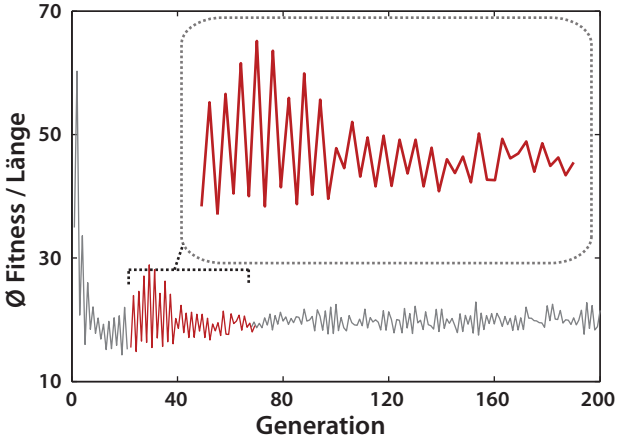


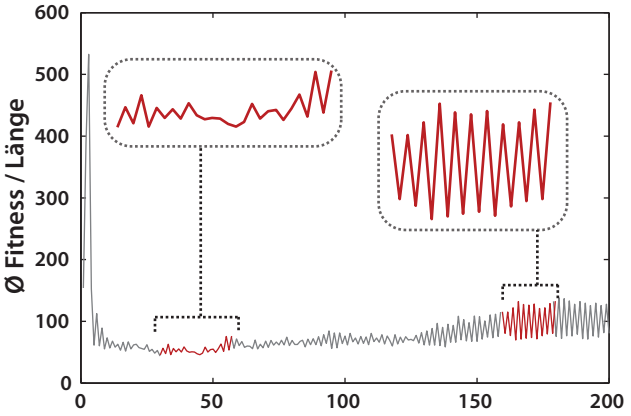
Abbildung 4.12: Verteilung der Individuen des normalen (oben) und des angepassten (unten) 8 - Hamming Problems. Die entscheidenden Bereiche wurden zusätzlich dreifach vergrössert dargestellt. Die Farb- und Grössencodierung der Kreise wird im Fliesstext erklärt.

8 - Hamming: Der Einfluss auf die Verteilung der Individuen sei exemplarisch für das 8 - Hamming Problem angegeben. In Abbildung 4.12 ist oben zunächst das Streudiagramm für das unveränderte Hamming Problem angegeben. Wie man sieht, treten dieselben Effekte wie beim angepassten Paritätsproblem (*Abbildung 4.11 unten*) auf: Die Anzahl der sehr kurzen Individuen nimmt sprunghaft zu (von Generation 139 zu 14Ø) und im nächsten Schritt ebenso wieder ab (von Generation 14Ø zu 141).

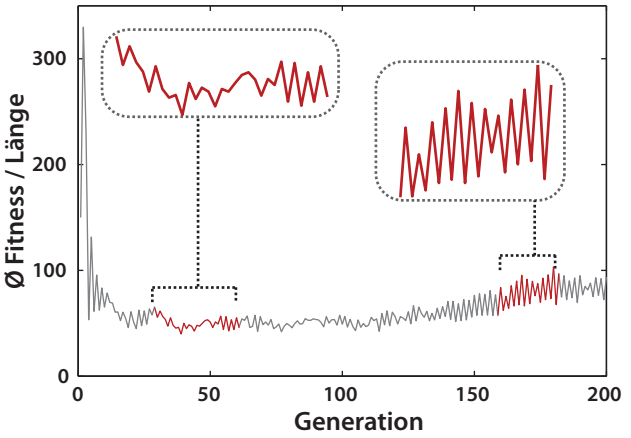
Durch die Anpassung stapeln sich nun die Individuen, ähnlich wie beim normalen 7 - Paritätsproblem, mit Fitness 256 aufeinander (*Abbildung 4.12 unten*). Wie die folgende Abbildung zeigt, tritt besonders am Anfang des Laufes noch ein Oszillationsverhalten auf. Dieses ist jedoch deutlich weniger stark, als noch beim unveränderten 8 - Hammingproblem.



11 - Multiplexer: Der Mittelwertverlauf des Fitness-zu-Länge-Verhältnisses ist in der ersten Hälfte des Laufes praktisch frei von Oszillationen. Diese treten jedoch zunehmend etwa ab der 13Ø. Generation auf. Sie sind jedoch nie so stark, wie in Abbildung 4.4.



1Ø - Addierer: Auch beim Addierer verschwinden die Oszillationen in der ersten Hälfte des Laufes praktisch vollständig und treten erst in der zweiten Hälfte leicht auf.



2. Lösung: IBEA

Eine weiter Möglichkeit, um die Oszillationen einzuschränken, ist IBEA. Dieses Verfahren wird in [ZKØ4] beschrieben. Statt der dichotomen Dominanzrelation wird hier ein kontinuierliches Mass verwendet.

Um die Stärke der Oszillation zu messen, wurde hier die relative Änderung des mittleren Fitness zu Länge Verhältnis verwendet. Diese wurde berechnet, indem der Mittelwert in der x-ten Generation durch den der x+1-ten Generation geteilt wurde. Diese relative Änderungen wurde für 5Ø Läufe bestimmt und gemittelt (rote) Kurve). Als Testproblem wurde das 6 - Hamming Problem verwendet.

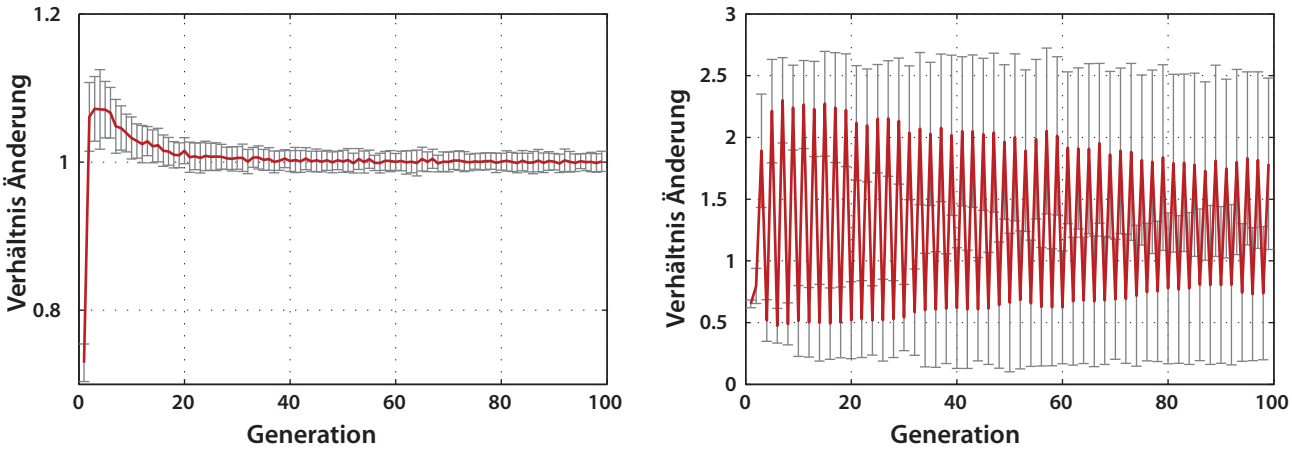


Abbildung 4.13: Diese zwei Diagramme zeigen, wie stark sich der Mittelwert von Fitness/Länge von einer zur nächsten Generation ändert. Rot eingezeichnet ist die mittlere relative Änderung, die grauen Balken stehen für die Standartabweichung der relativen Änderung. Links ist das Ergebnis für IBEA, rechts für SPEA2 abgebildet.

Wie man in Abbildung 4.13 links sehen kann, oszilliert die IBEA Population praktisch überhaupt nicht. Als Vergleich dazu sind rechts die Werte von SPEA2 aufgezeichnet.

3. Lösung: SPEA2 verändern

Ein möglicher Grund für die Oszillationen bei den SPEA2 Läufen ist die Art, wie bei übereinanderliegenden Punkte die Pareto-Fitness berechnet wird:

» **Pareto-Stärke**: Zur Pareto-Stärke zählen alle Punkte, welche schwach dominiert werden – also auch Punkte, die die selben Werte aufweisen. Liegen also n Punkte übereinander, so erhöhen sie sich gegenseitig die Pareto-Stärke um den Wert (n-1).

» **Pareto-Fitness**: Um nun die Pareto-Fitness eines Individuums zu erhalten, werden die Pareto-Stärken aller Punkte zusammengezählt, welche dieses Individuum schwach dominieren, also auch von den Individuen, welche gleich lang und fit sind. Zur Pareto-Fitness dieser Individuen wird also (n-1) mal die um (n-1) erhöhte Pareto-Stärke hinzuaddiert.

Dadurch, dass sich zwei Individuen gegenseitig schwach dominieren, erhöhen sie sich gegenseitig die Pareto-Fitness, diese nimmt bei grossen Punkthaufen annähernd quadratisch mit der Anzahl übereinanderliegender Individuen zu.

Modifikation: Um zu verhindern, dass der obige Effekt eintritt, wurde bei den drei Modifikationen die Dominanzrelation angepasst. Dabei wird auf ID-Werte zurückgegriffen. Um ein Individuum mit gleicher Länge und Fitness zu dominieren, muss nun dieses Nummer kleiner sein (*ähnlich wie beim FOCUS Verfahren, welches im gleichnamigen Abschnitt beschrieben wird.*). Dadurch besteht zwischen zwei Individuen immer eine eindeutige Dominanzrelation.

Diese angepasste Dominanzrelation wurde auf drei verschiedene Arten in SPEA2 eingebaut:

» **1**: Die Pareto-Stärke wird nach alter Dominanzrelation berechnet, die Pareto-Fitness nach neuer.

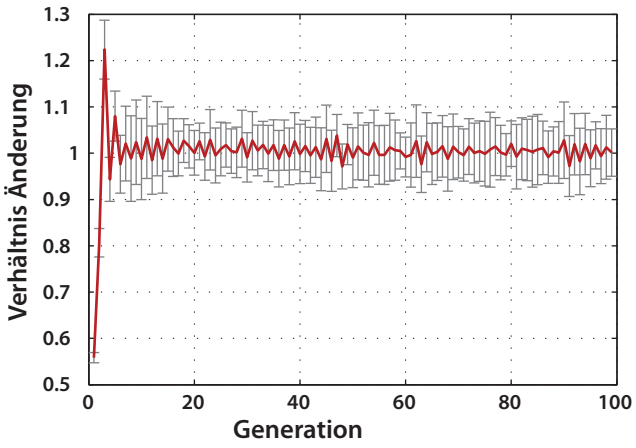
» **2**: Sowohl Pareto-Stärke als auch -Fitness wurden nach neuer Dominanzrelation berechnet.

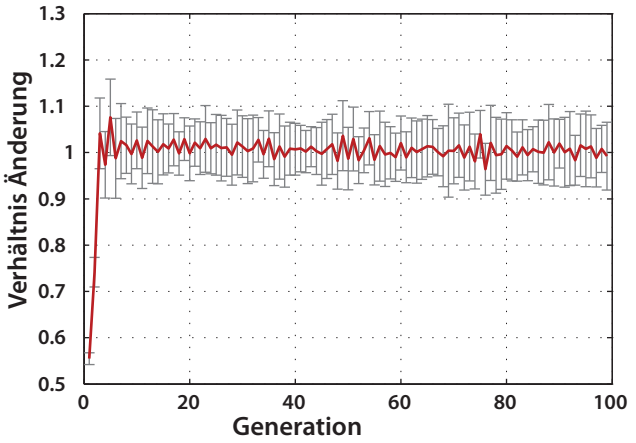
» **3**: Nur die Pareto-Stärke wurde mit der neuen Relation berechnet.

Wie man in Abbildung 4.14 sehen kann, führen die beiden ersten Änderung zu einer starken Reduktion der Oszillationen. Die dritte Modifikation bringt keine Verbesserung, die Oszillation ist ähnlich stark wie im unveränderten SPEA2 Experiment.

Dies lässt darauf schliessen, dass die Berechnung der Pareto - Fitness und nicht der Stärke für das Oszillationsverhalten verantwortlich ist.

Die drei Modifikationen wurden mit dem normalen SPEA2 im Lösen des 1Ø-Addiererproblems verglichen (*Abbildung 4.15*). Die Unterschiede sind in allen Belangen relativ klein. Am meisten Lösungen fanden das normale SPEA2 sowie dessen erste Modifikation.





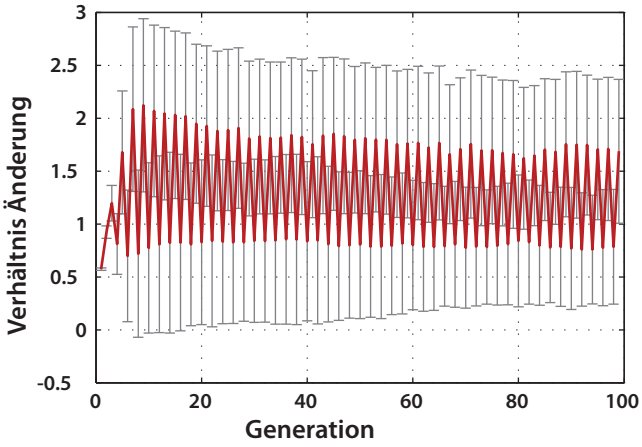


Abbildung 4.14: Oszillationsverhalten der ersten (oben), zweiten (mitte) und dritten (unten) SPEA2 Modifikation. Rot eingezeichnet ist die mittlere relative Änderung des Mittelwertverlaufs des Fitness zu Länge Verhältnisses, die grauen Balken bezeichnen entsprechend die Standartabweichung.

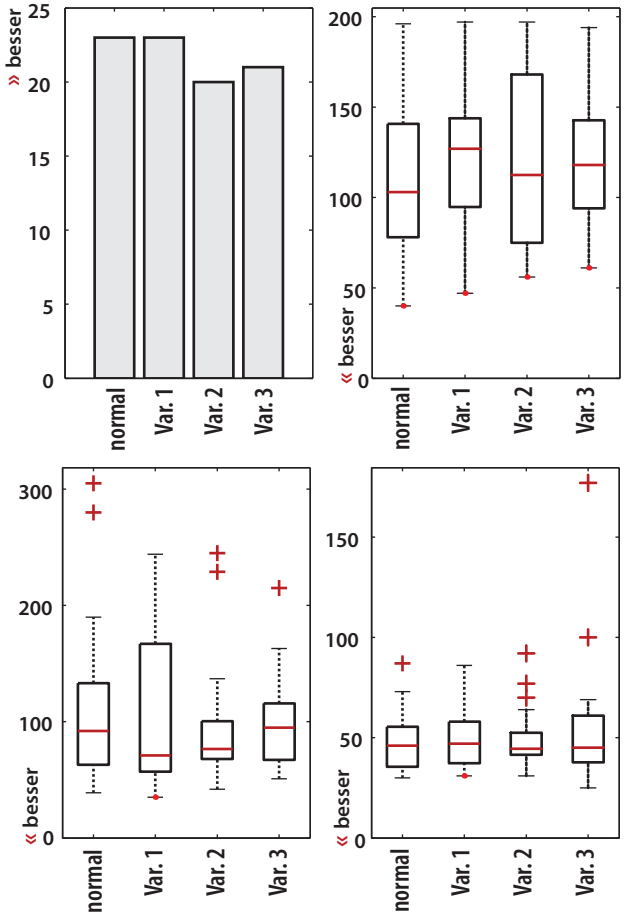


Abbildung 4.15: Vergleich der drei modifizierten mit dem normalen SPEA2.
Anzahl gefundener Lösungen (oben links), Generationen bis zur Lösung (oben rechts). Länge des ersten perfekten (unten links) und des kürzesten perfekten (unten rechts) Individuums. (Siehe nächstes Kapitel)

5. Vergleiche

In diesem Abschnitt sollen die drei Verfahren IBEA, SPEA2 mit erster Modifikation und SPEA2 normal verglichen werden.

Simulationsaufbau

Da die Berechnungen in IBEA sehr aufwändig sind, wurden hier Parameter verwendet, die von den in Abschnitt *„Simulationsparameter"* beschriebenen in drei Punkten abweichen:

» **Populationsgrösse**: Die Populationsgrösse wurde auf 1ØØØ reduziert.

» **Archivgrösse**: Die maximale Archivgrösse wurde auf 1ØØØ Individuen gesetzt. IBEA nutzt das Archiv komplett, während SPEA2 in allen Läufen nur sehr kleine Teile davon belegte.

» **Generationen**: Die Läufe wurden bereits nach 1ØØ Generationen abgebrochen.

Für jede Kombination aus Testproblem und Algorithmus wurden 5Ø Läufe simuliert.

Da statt 8Ø4'ØØØ Individuen nur 1Ø1'ØØØ simuliert wurden, sind auch die Testprobleme zum Teil verkleinert worden.

Diagramme

Um die Algorithmen zu vergleichen, wurden für jedes Testproblem vier verschiedene Diagramme erstellt:

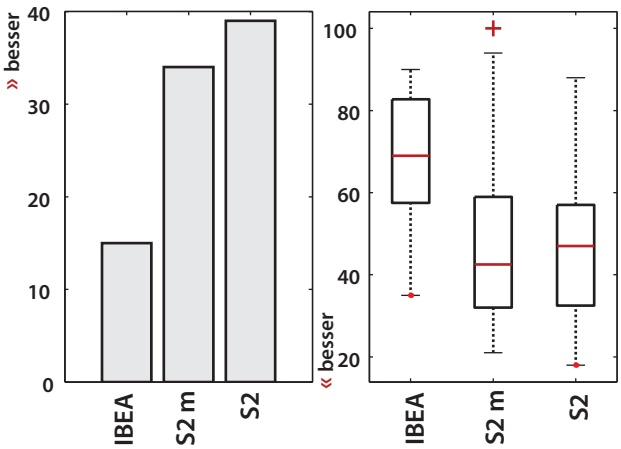
» **Anzahl gefundener Lösungen**: Dieses Balkendiagramm gibt an, in wievielen der 5Ø Läufen eine Lösung gefunden wurde. Es ist somit ein Mass für die Zuverlässigkeit des Algorithmus.

» **Generationen bis zu perfekten Lösung**: Dieser Boxplot zeigt, in welcher Generation die erste perfekte Lösung auftrat. Datengrundlage ist für dieses, wie auch die folgenden zwei Diagramme, die Läufe, in denen tatsächlich auch eine Lösung gefunden wurde. Das Diagramm ist ein Mass für die Geschwindigkeit, mit der der Algorithmus Lösungen findet.

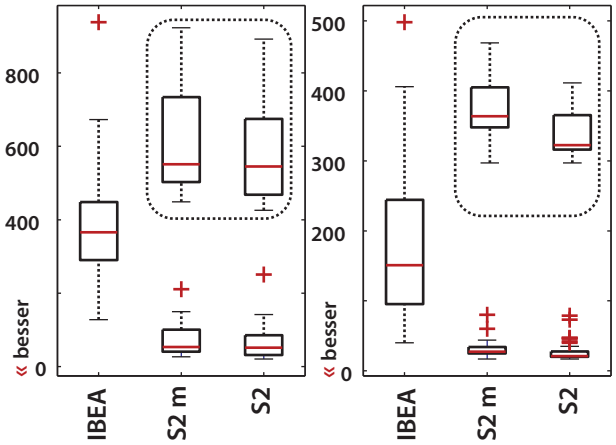
» **Länge der ersten Lösung**: In diesen Diagrammen wird die Länge der ersten gefundenen Lösung dargestellt. Treten mehrere perfekte Lösungen gleichzeitig auf, so floss jeweils nur das Kleinste in die Statistik ein. Diese Statistik soll zeigen, wie gut der Algorithmus Bloat verhindert.

» **Kürzeste gefundene Lösung**: Dieses Diagramm zeigt, wie gut der Algorithmus Lösungen hinsichtlich der Länge optimieren kann. Pro Lauf fliesst, falls vorhanden, die Lösung mit kürzester Länge ein.

Die Boxplots zeigen den Median (rote Linie), den interquartilen Bereich (Rechteck), die Extrema als „Whisker" mit einer Länge von maximal 1.5 fachem Interquartilsabstand (gestrichelte Linien) und die Ausreisser (rote Kreuze).



5 - Parität: Anzahl gefundener Lösungen (links) und verstrichene Generationen bis diese gefunden wurde (rechts)



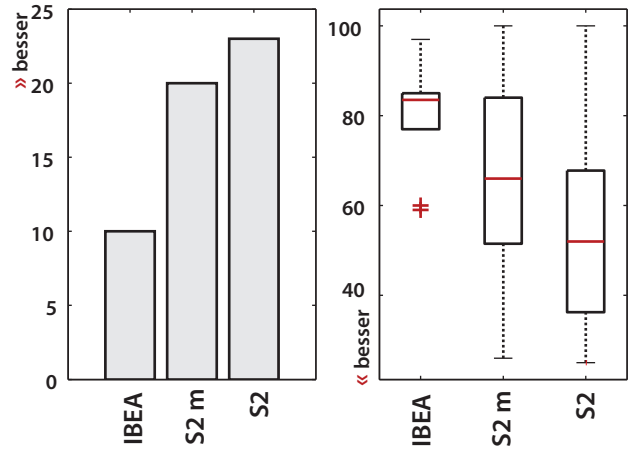
5 - Parität: Länge der ersten (rechts) sowie der kürzesten gefundenen Lösung (links). Die beiden „Boxen" der SPEA Statistik wurden zusätzlich vergrössert dargestellt.

5 - Parität

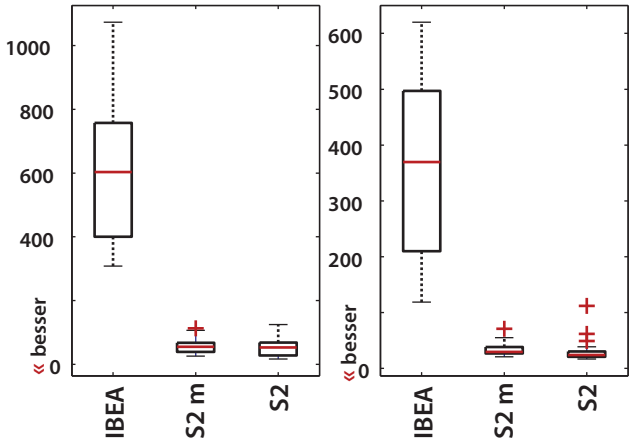
Das normale SPEA2 (Abkürzung S2) findet am meisten Lösungen (39), gefolgt von der Modifikation (Abkürzung S2m) mit 34 Lösungen. IBEA findet nur in 3Ø% der Läufe eine Lösung.

Im zweiten Diagramm sieht man, dass die beiden SPEA2 Varianten die Lösung ähnlich schnell finden, mit leichten Vorteilen bei der Modifikation. IBEA fällt auch hier stark ab.

Sowohl die Länge der ersten als auch der kürzesten Lösung ist in den SPEA2 Versionen recht klein. IBEA hingegen produziert sehr lange Lösungen.



5 - Parität modifiziert: Anzahl gefundener Lösungen (links) und verstrichene Generationen bis diese gefunden wurde (rechts)



5 - Parität modifiziert: Länge der ersten (rechts) sowie der kürzesten gefundenen Lösung (links).

5 - Parität mit modifizierter Fitness

In diesen Läufen wurde die Fitness der Bäume gemäss der ersten Methode aus Abschnitt *„Modifiziertes Paritätsproblem"* berechnet.

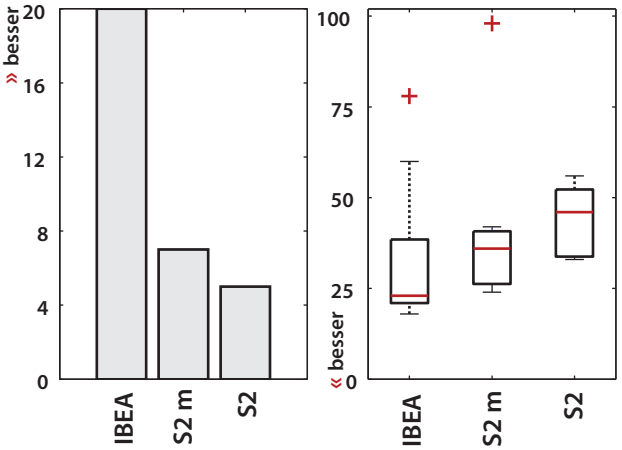
Durch die Modifikation werden mit allen drei Algorithmen deutlich weniger Lösungen gefunden. Wie beim normalen Paritätsproblem findet allerdings auch hier das unveränderte SPEA am meisten Lösungen (23). IBEA findet nur halb so viele Lösungen (1Ø) wie die modifizierte SPEA Version (2Ø).

Auch die Anzahl verstrichener Generationen bis zum ersten perfekten Individuum ist im Schnitt grösser als noch in den unveränderten Paritätsläufen. Das unveränderte SPEA2 findet allerdings hier die Lösungen etwas schneller als die SPEA2 Modifikation.

Wie schon beim Standart Paritätsproblem sind die Lösungen von IBEA sehr lang, wohingegen beide SPEA2 Versionen sehr kurze Lösungen finden.

11 - Multiplexer

Der 11- Multiplexer ist für die verkleinerte Population und die reduzierte Laufzeit ein zu schwieriges Problem. Die beiden SPEA2 Algorithmen fanden nur 7 bzw. 5 mal eine Lösung. und damit deutlich weniger als IBEA (2Ø Lösungen). Auch



11 - Multiplexer: Anzahl gefundener Lösungen (links) und verstrichene Generationen bis diese gefunden wurde (rechts)

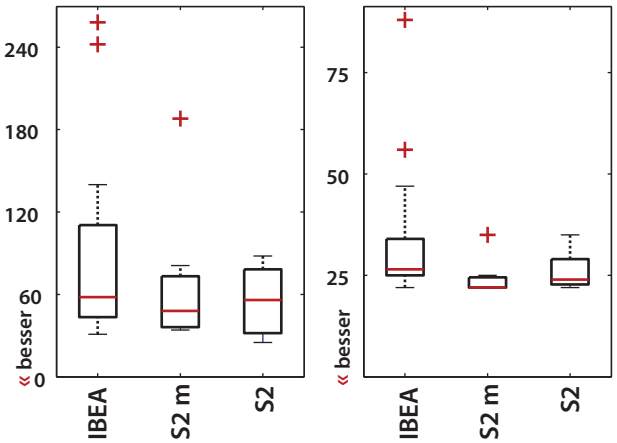
in der Zeit bis zur ersten Lösung liegt IBEA vor S2 m und dem normalen SPEA2.

Was die Länge der ersten und der minimalen Lösung angeht, sind die SPEA2 Algorithmen leicht besser als IBEA.

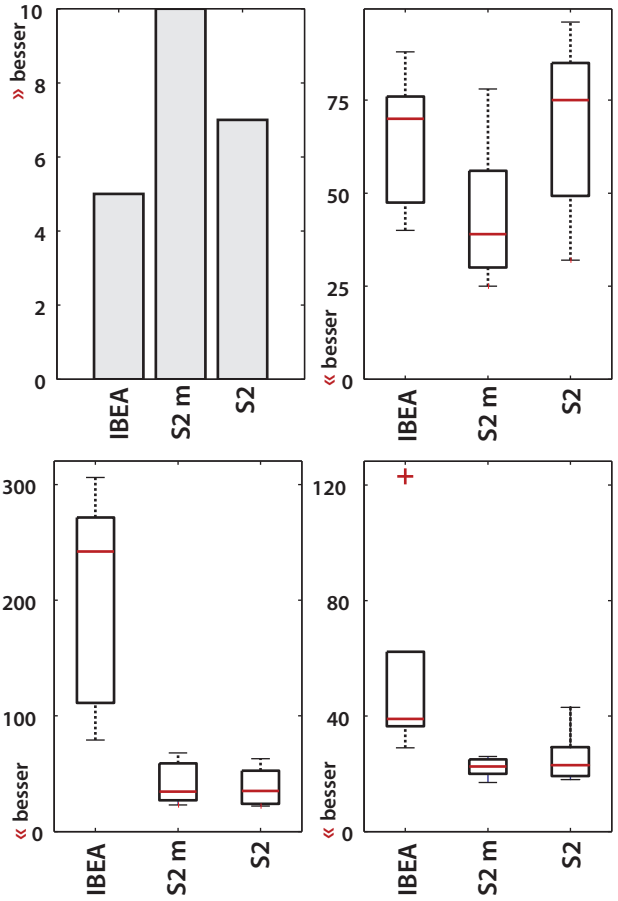
6 - Hamming

Am meisten Lösungen findet der modifizierte SPEA2 Algorithmus (1Ø), gefolgt vom normalen SPEA2 (7 Lösungen) und IBEA mit (5). Auch was die Dauer bis zur Lösung angeht, liegt S2m vorne.

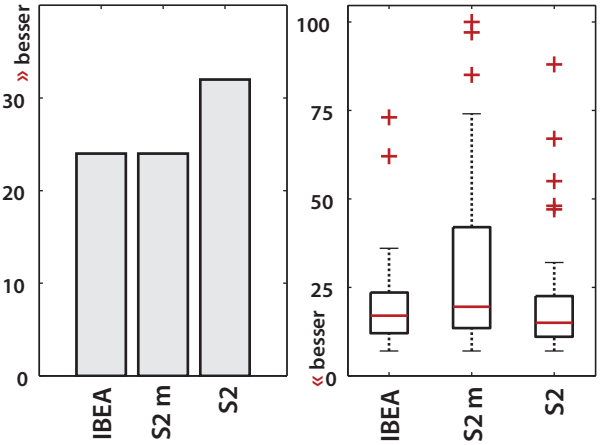
Die Länge der ersten Lösung ist bei IBEA sehr gross, der Algorithmus kann diese aber noch



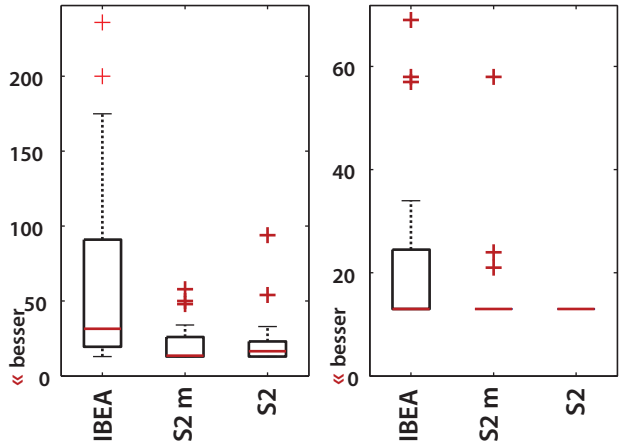
11 - Multiplexer: Länge der ersten (rechts) sowie der kürzesten gefundenen Lösung (links).



6 - Hamming: Anzahl gefundener Lösungen (oben links) und verstrichene Generationen bis diese gefunden wurde (oben rechts).
Länge der ersten (unten rechts) sowie der kürzesten gefundenen Lösung (unten links).



6 - Addierer: Anzahl gefundener Lösungen (links) und verstrichene Generationen bis diese gefunden wurde (rechts)



6 - Addierer: Länge der ersten (rechts) sowie der kürzesten gefundenen Lösung (links).

stark reduzieren (viertes Diagramm). Trotzdem sind die Lösungen deutlich länger als in den SPEA Läufen.

6 - Addierer

Das normale SPEA2 findet am meisten Lösungen (34), die Modifikation und IBEA beide 24 Lösungen. Alle drei finden die Lösungen häufig sehr schnell, wobei die Modifikation etwas länger braucht.

Die Länge ist bei SPEA2 Läufen am kürzesten, es findet den minimalen Baum (13 Kanten) sogar in allen 34 Fällen. Die SPEA2 Modifikation findet diese kürzeste Lösung in 2Ø, IBEA nur in 2 von 24 Läufen.

Fazit

IBEA findet den 11-Multiplexer deutlich öfter als die beiden SPEA2 Varianten, liegt in den restlichen Problemen jedoch hinter den beiden Verfahren. Die Länge der IBEA Lösungen ist durchschnittlich immer grösser als bei den SPEA2 Varianten, bei den zwei Paritätsproblemen und der Hamming Distanz sogar sehr viel länger.

Die beiden SPEA2 Versionen sind ähnlich effektiv im Auffinden von Lösungen, die unveränderte Variante ist in den Paritätsproblemen und im Hammingproblem leicht besser, die Modifikation im Multiplexer- und Addiererproblem. Beide Varianten liefern zudem immer sehr kurze Lösungen.

Schlusswort

Zusammenfassung

Beim häufig verwendeten Paritätsproblem wurde eine Eigenheiten entdeckt: Kleine Individuen, und damit potentielle Bausteine, berechnen immer 50% der Eingangskombinationen richtig. Für das Überleben dieser Individuen in der Population ist somit nur der Wert der Länge, nicht aber der Fitness entscheidend.

Ausgehend von dieser Beobachtung wurde das Paritätsproblem dahingehend angepasst, dass die Fitness der Bäume entsprechend der Anzahl verwendeter Eingänge berechnet wird. Vergleiche zeigten jedoch, dass SPEA2 sich an diesem angepassten Problem schwerer tat.

Da kleine Individuen durch die Modifikation des Testproblems verschiedene Fitnesswerte annehmen können, steigt ihr Anteil an der Population. Zudem nimmt der Anteil abwechselnd stark zu bzw. ab. Diese Oszillationsproblem trat auch bei den restlichen untersuchten Testproblemen auf. Verschiedene Massnahmen wurden untersucht, um dieses Oszillationsverhalten einzuschränken. Einerseits durch Modifikation der Testprobleme und andererseits durch Anpassen der Pareto-Fitness Berechnung in SPEA2 bzw. durch Verwenden der IBEA Indikatoren. Beide Massnahmen konnten die Oszillationen stark reduzieren, was allerdings immer mit einer Verschlechterung beim Auffinden von Lösungen einher ging. Es ist anzunehmen, dass die Oszillationen SPEA2 kaum schaden und vielmehr eine natürlich Folge der Pareto-Fitness Berechnung sind.

Um die Baustein-Hypothese zu prüfen, wurden verschiedene Statistiken zur Fitness der ausgetauschten Teilbäume und deren Einfluss auf die Nachkommen erstellt. Dabei zeigte sich, dass die Fitness des weggeschnittenen Teilbaums einen weit grösseren Einfluss hat, als die des hinzugefügten. Dies kann mehrere Ursachen haben:

Zum einen müssen potentielle Bausteine nicht automatisch ein gute Fitness haben und umgekehrt. Beim unveränderten Paritätsproblem etwa hat selbst ein perfekter 6-Paritätsbaum eine schlechte Fitness, anderseits erreicht ein Baum mit konstanter Ausgabe auf dem Hammming Problem vergleichsweise gute Werte. Zum andern spielt die Grösse der ausgetauschten Teilbäume eine sehr grosse Rolle und verdeckt möglicherweise den Einfluss der Fitness.

Durch Weglassen der Kreuzungsoperation wurde SPEA2 künstlich die Möglichkeit auf Zusammensetzen von Bausteinen genommen. Dass die Leistungsfähigkeit des Algorithmus kaum abnahm bestätigt, dass die Fitness des eingefügten Teilbaums keine grosse Rolle spielt (also auch zufällig sein kann wie bei der Mutation).

Die Verteilung der Fitnessdifferenzen von Eltern zu Nachkommen zeigte, dass sprunghafte Verbesserungen der Fitness äusserst selten sind. Solche Verbesserungen würden für ein Zusammensetzen von Bausteinen sprechen.

Fazit

Keine der Untersuchungen konnte ein Hinweis finden, dass die Vorteile von SPEA2 in der Baustein-Hypothese gründen. Selbst durch komplettes Ausschalten der Kreuzungsoperation verliert das Verfahren nicht stark an Leistungsfähigkeit.

In den untersuchten Testproblemen entwickeln sich die Individuen vielmehr in kleinen Schritten über Mutation. SPEA2 scheint dabei eine besonders grosse Vielfalt an Individuen aufrecht zu erhalten und zieht daraus seinen Vorteil.

Literaturverzeichnis

[LanØØ] W. B. Langdon .Size fair and homologous tree crossover for tree genetic programming. *Genetic Programming Evol. March.*, Vol. 1, 95-119, 2ØØØ.

[Koz92] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press: Cambridge, MA, 1992.

[LSP99] W. B. Langdon, T. Soule, R. Poli und J. A. Foster. *The evolution of size and shape. In Advances in Genetic Programming III*. The MIT Press, Seiten 163-19Ø. Cambridge, MA, 1999.

[LanØ2] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2ØØ2.

[LØØ-1] S. Luke. *Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat*. PhD thesis, University of Maryland, College Park, 2ØØØ.

[LØØ-2] S. Luke. Code growth is not caused by introns. In *Late-Breaking Papers, Proceedings of GECCO 2ØØØ*. Seiten 228-235, 2ØØØ.

[ZB95] B.-T Zhang und H. Mühlenbein. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation, 3(1)*. Seiten 17–38, 1995.

[Bli96] T. Blickle. Evolving compact solutions in genetic programming: A case study. In H.-M. Voigt, W. Ebeling, I. Rechenberg, und H.-P. Schwefel, Editoren, *PPSN IV*, Seiten 564–573. Springer-Verlag, 1996.

[SF99] T. Soule und J. A. Foster. Effects of code parsimony pressure on population in genetic programming. *Evoluationary Computation*, *6(4)*. Seiten 293–3Ø9, 1999.

[LPØ2] S. Luke und L. Panait. Fighting Bloat With Nonparametric Parsimony Pressure. *Parallel Problem Solving from Nature - PPSN VII, Lecture Notes in Computer Science*, *LNCS*. Seiten 411--421, 2ØØ2.

[Hag93] M. Hagiwara. Pseudo-hill climbing genetic algorithm (PHGA) for function optimization. *Proceedings of the International Joint Conference on Neural Networks, v1*. Seiten 713-716, 1993.

[NFB95] P. Nordin, F. Francone und W. Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In P.J. Angeline und K.E. Kinnear Jr., Editoren, *Advances in Genetic Programming 2*, MIT Press, Cambridge, MA, USA. Seiten 111–134, 1995

[JWPØ1] E. D. De Jong, R. A. Watson und J. B. Pollak. Reducing bloat and promoting diversity using multi-objective methods. In L. E. Spector, E. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon und E. Burke, Editoren, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2ØØ1*. Seiten 11-18.

[BFKN98] W. Banzhaf, F. D. Francone, R. E. Keller und P. Nordin. *Genetic Programming: An Introduction*. Morgan Kaufmann, San Francisco, CA, 1998.

[BBTZØ1] S. Bleuler, M. Brack, L. Thiele und E. Zitzler. Multiobjective genetic programming: Reducing bloat by using SPEA2. In *Congress on Evolutionary Computation (CEC-2ØØ1)*. Seiten 536-543, Piscataway, NJ, 2ØØ1. IEEE

[BT94] T. Blickle und L. Thiele. Genetic programming and redundancy. In J. Hopf, Editor, *Genetic Algorithms within the Framework of Evolutionary Computation (Workshop at KI-94, Saarbrücken)*. Seiten 33-38, 1994.

[ENØ1] A. Ekárt und S. Z. Németh. Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 2. Seiten 61-73, 2ØØ1.

[JPØ3] E. D. D. Jong und J. B. Pollack. Multi-objective methods for tree size control. *Genetic Programming and Evolvable Machines*. 4:211-233, 2ØØ3.

[ZKØ4] E. Zitzler und S. Künzli. Indicator-Based Selection in Multiobjective Search. *Parallel Problem Solving from Nature (PPSN VIII)*. Seiten {832-842}, 2ØØ4.