Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Technische Informatik und Kommunikationsnetze
System Optimization Group

Semesterarbeit

Mehrzieloptimierung unter Berücksichtigung struktureller Eigenschaften von Lösungen

Herbstsemester 2009

Martin Baumgartner 05-912-985 <martibau@ee.ethz.ch>

Betreuerin: Tamara Ulrich
Professor: Prof. Dr. Eckart Zitzler

Zürich
4. November 2009

Zusammenfassung

Evolutionären Algorithmen werden zur Optimierung von komplexen Problemen verwendet. Die Lösungen wurden dabei bisher ausschliesslich gemäss ihren Zielfunktionswerten bewertet. Ziel dieser Arbeit ist es, die Qualität der bisherigen Algorithmen zu erhalten und dabei auch die Struktur der Lösungen in die Optimierung miteinzubeziehen.

Dazu werden zwei Methoden untersucht. Die erste verwendet eine Linearkombination von einem herkömmlichen zielfunktionswert-basierten Verfahren und einer strukturbasierten Güte. Die zweite Methode verwendet die strukturelle Güte als Gewichtung in einem herkömmlichen Verfahren.

Es werden dabei zwei strukturelle Kriterien untersucht. Das eine ist die strukturelle Verschiedenheit zu den anderen Lösungen, das andere die Abweichung von einer vordefinierten Teilstruktur.

Bei Experimenten mit dem zweidimensionalen Rucksackproblem hat sich gezeigt, dass die zweite Methode in den Zielfunktionswerten und in den strukturellen Werten bessere Resultate liefert als die erste. Ebenfalls hat sich jedoch gezeigt, dass für gute strukturelle Werte schlechtere Zielfunktionswerte hingenommen werden müssen.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

1.1.1 Mehrzieloptimierung

Als Mehrzieloptimierung bezeichnet man das Finden einer möglichst guten Lösung, die mehreren Zielen[1] möglichst gut Rechnung trägt. Dabei kommt es meistens zu Zielkonflikten[2]. Das Problem wird mit jedem hinzukommenden Ziel (Dimension) komplexer.

Die Mehrzieloptimierung erlaubt es, während dem Lösen des Problems über das Problem selbst zu lernen: Man erfährt zum Beispiel beim Analysieren der Front, wie die einzelnen Ziele im Zusammenhang stehen.

Mindestens so spannend wie die Veranschaulichung der Zielwerte sind jedoch die Lösungsstrukturen, welche zu diesen Werten führen. Deshalb ist es wünschenswert, die Struktur in die Optimierung einbauen zu können. Damit kann man zum Beispiel folgendes suchen:

- möglichst verschiedene Lösungen (Kapitel 2 und 3). Damit erhalten zum Beispiel Entwicklungsengineure einen wichtigen Überblick über ein möglicht breites Spektrum an Lösungen, was zu komplett neuen Lösungsansätzen führen kann.

- Lösungen, die einer gegebenen Struktur (Template) möglichst ähnlich sind. (Kapitel 4). Dies wird zum Beispiel in der Autoindustrie genutzt, weil vom Hersteller gewisse Bauteile vorgegeben sind: Duch Lieferantenverträge oder aufgrund von Sicherheitstandars müssen gewisse Teile[3] verwendet werden.

1.2 Problembeschreibung

Diese Arbeit befasst sich mit Evolutionären Algorithmen (Kapitel 1.3).

Dabei sind pro Lösung Zielfunktionswerte vorgegeben, die optimiert werden müssen. Zusätztlich zu den Zielfunktionswerten ist ein auf dem Entscheidungsraum basierender Gütewert gegeben.

Es sollen Algorithmen in Evolutionäre Algorithmen eingebaut werden, die

[1] alle Ziele zusammen ergeben die Qualität einer Lösung

[2] z.B. soll die Grösse eines Computerspeichers maximiert und die Kosten minimiert werden

[3] Busse, Prozessoren, Controller, ...

1. die Zielfunktionswerte optimieren und dabei nicht schlechter sind als herkömmliche Algorithmen
2. den strukturellen Gütewert gegenüber herkömmlichen Algorithmen verbessern.

Der strukturelle Gütewert kann nicht einfach als weitere Zielfunktion übernommen werden, da dann im ursprünglichen Zielfunktionsraum beliebig schlechte Lösungen zulässig würden, was man nicht wünscht.

Es werden zwei Ansätze geprüft, die Struktur in die Optimierung miteinzubeziehen:

Ansatz 1: Es wird nicht stikt nach der Zielfunktion optimiert, sondern nach einer Linearkombination von Zielfunktionen und strukturellem Gütewert. Der Ansatz ist in Kapitel 2 beschrieben.

Ansatz 2: Der strukturbasierte Gütewert wird als Gewichtungsfaktor in einen auf den Zielfunktionen basierenden Algorithmus eingesetzt. Der Ansatz wird mit zwei unterschiedlichen strukturellen Gütewerten in den Kapiteln 3 und 4 beschrieben.

1.3 Evolutionäre Algorithmen

Evolutionäre Algorithmen arbeiten nach dem Vorbild der Evolutionstheorie von Charles Darwin [1].

Sie werden für komplexe Problemstellungen verwendet, die

- keine Aussagen über die Struktur des Suchraums geben
- meistens keine dominante Lösung haben[4]
- mehrere Zielfunktionen haben (Kapitel 1.1.1 und 1.5.2).F

Eine Population, bestehend aus Individuen, wird zu Beginn initialisiert und ist dann in jeder Generation folgenden Schritten ausgesetzt:

- Variation (Kapitel 1.3.1)
- Fitness: (Kapitel 1.3.2)
- Selektion (Kapitel 1.3.2)

[4] es hat mehr als ein Element in der Pareto-Front

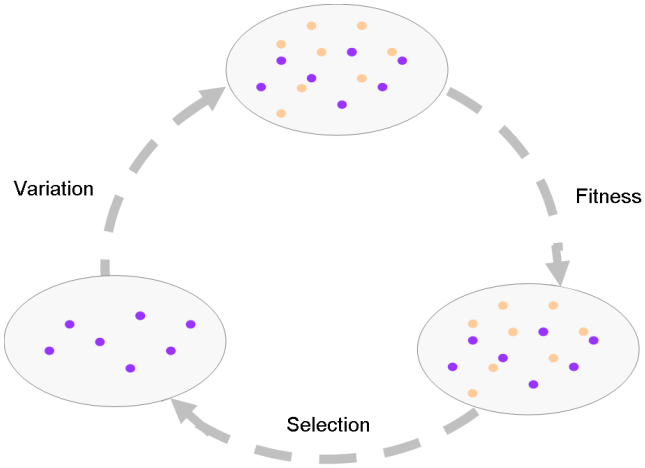


Abbildung 1: Evolutionärer Algorithmus: Eine Generation

1.3.1 Variation

Durch Variation werden aus den bestehenden Individuen neue generiert. Dies geschieht oft durch

Rekombination: Aus den Eigenschaften von mehreren bestehenden Individuen werden neue Individuen generiert.

Mutation: Durch kleine Änderungen werden einzelne Individuen verändert.

1.3.2 Fitness und Selektion

Das Hauptaugenmerk der Arbeit ist auf die Berechnung der Fitness und der daraus resultierenden Selektion gerichtet.

Fitness bezeichnet die Zuweisung eines eindeutigen Gütewerts zu jedem Individuum.

Selektion ist das Auswählen der Individuen aufgrund der Fitness für die nächste Generation.

Im weiteren Verlauf der Arbeit wird **Selektion** für den ganzen Prozess von Fitness und Selektion verwendet, da sich Fitness und Selektion in allen in der Arbeit vorkommenden Algorithmen abwechseln[5].

[5]Die Individuen werden schrittweise aus der Population entfernt und was übrig bleibt ist die nächste Generation.

Durch Variation werden in jeder Generation neue Individuen generiert. Damit *die Welt nicht kollabiert* (der Algorithmus auch über Generationen gleich effizient bleibt), müssen in jeder Generation gleich viele Individuen vorhanden sein. Mit dem Auswählen der Individuen für die nächste Generation wird die Grundlage für den Erfolg des Algorithmus gelegt.

Die meisten der neu generierten Individuen werden schlechtere Eigenschaften haben als die bestehenden, da diese schon optimiert sind. Es gibt - nach dem Zufallsprinzip - jedoch einige wenige, die der Population einen Mehrwert bringen, wenn sie Individuen der bestehenden Population ersetzen.

Je mehr Generationen eine Population lebt, desto besser sollte sie werden (was nicht unbedingt nur heisst, dass die einzelnen Individuen auch gut sind, sondern, dass sich diese gut ergänzen) und desto weniger Individuen werden *ersetzt*.

Die Selektion erlaubt es, nur diejenigen Individuen weiterleben zu lassen, welche auch wirklich gut sind. So können einige besonders gute Individuen auch viele (oder sogar alle) Generationen überleben, ohne *eines natürlichen Todes zu sterben*.

Die Kriterien für die Selektion werden im weiteren Verlauf der Arbeit diskutiert.

Bisherige Algorithmen zur Selektion stützen sich auf die Zielfunktionswerte. In dieser Arbeit werden Ansätze geprüft, bei denen auch der Entscheidungsraum berücksichtigt wird.

1.4 verwendetes Beispiel: Rucksackproblem

Das Rucksackproblem ist auch unter dem Namen 'Knapsack' bekannt [2].

Durch Wählen der *dim* des Problems ergeben sich im Entscheidungsraum $dim * m$ Elemente, je bestehend aus einem $Profit \in [10, 11, \ldots, 100]$. und einem $Gewicht \in [10, 11, \ldots, 100]$. Der Entscheidungsraum der Individuen E_i besteht nur aus '0' (Element mit $Profit_{j,1}$, $Profit_{j,2}$, $Gewicht_{j,1}$ und $Gewicht_{j,2}$ ist in Individuum nicht enthalten) und '1' (ist enthalten): $e_{i,j} \in \{0, 1\}$.

Das Ziel ist es, die Summe aller Profite zu maximieren, ohne dass das Gewicht die gegebene Kapazität übersteigt. Die Kapazität in Dimension d ist

$$K_d = \frac{\sum_{i=1}^{m} Gewicht_{i,d}}{2 * m}.$$

Die Zielfunktion[6] von Individuum s_i in Dimension d ist wie folgt definiert:

$$z_{i,d} = \sum_{k=1}^{k_{max}} Profit_{k,d}$$

$$k_{max} = min(k_{max_d}), d \in \{1, 2\}$$

$$k_{max_d} = max(l_{max_d}) \| \sum_{l=1}^{l_{max_d}} Gewicht_{l,d} < K_d$$

Der zugrundeliegende Code ist in C geschrieben [3].

Der Entscheidungsraum ist in Tabelle 1 (Kapitel A.2.1) abgebildet.

1.5 Begriffe

Einige in der Arbeit verwendeten Begriffe und Zeichen werden hier eingeführt.

1.5.1 Entscheidungsraum

Der Entscheidungsraum spannt den gesamten Suchraum auf und definiert jede Lösung. Er ist die Abbildung der Struktur der Lösungen.

Der Entscheidungsraum ist für jedes Problem unterschiedlich. Er beschreibt das eigentliche Problem und bildet gleichzeitig Lösungen ab. Die Repräsentation des Entscheidungsraums kann ganz unterschiedlich sein: einzelne Bits[7], natürliche Zahlen, reelle Zahlen, Bäume, ... und Kombinationen davon.

Er bildet das Fundament, welches schliesslich zu den Zielfunktionen (Kapitel 1.5.2) führt.

In dem in dieser Arbeit untersuchten Problem (Kapitel 1.4) ist der Entscheidungsraum binär aufgebaut: Es gibt Elemente, die entweder in einer Lösung enthalten sind oder nicht.

1.5.2 Zielfunktionsraum

Der Zielfunktionsraum besteht aus einer oder mehreren Zielfunktionen. Eine Zielfunktion ist eine aufgrund des Entscheidungsraums berechnete Funktion, die optimiert[8] wird (z.B. Kosten, Gewicht, Zeit, ...).

[6] Dies ist die interne Zielfunktion des Rucksackproblems, die einem Maximierungsproblem entspricht. Dem Selektor wird ein Wert $\alpha - z_{i,d}$ übergeben, damit es sich um ein Minimierungsproblem handelt.

[7] wird in dieser Arbeit verwendet, siehe Kapitel 1.4

[8] es wird das Maximum/Minimum davon gesucht

1.5.3 Hypervolumen

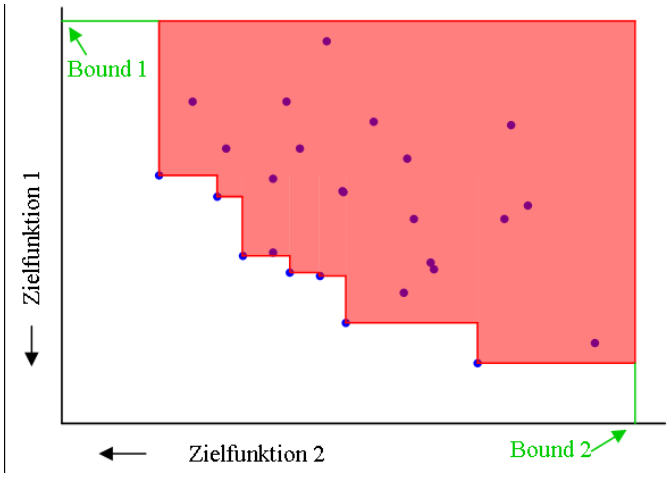


Abbildung 2: (ungewichtetes) Hypervolumen

Das Hypervolumen ([4], siehe Abbildung 2) ist ein Indikator (I_{hyp}) zur Beurteilung der Güte einer Lösungsmenge und definiert durch

$$Hypervolumen = \text{von Individuen dominiertes Volumen.}$$

Die Dimension des Hypervolumens entspricht dim.

Es wird ein $Bound$[9] eingeführt. Dieser dient dazu, dass das Hypervolumen nicht unendlich[10] wird und sollte grösser sein als alle in der Dimension vorkommenden Werte.

Der Hypervolumenindikator wird bei der Auswertung der Resultate (Kapitel 5) verwendet, um die Qualität der Lösungsmenge des jeweiligen Algorithmus zu beurteilen.

Es gibt Algorithmen (1, 7), die das Hypervolumen als Entscheidungskriterium benutzen, indem man sich die Eigenschaft zunutze macht, dass es sich beim Entfernen einzelner Individuen unterschiedlich stark ändern kann.

1.5.4 Glossar

Entscheidungsraum $E = \{e_i | i \in \{1, \ldots, m\}\}$ besteht aus **m** Elementen.

Diversität D ist ein Mass, die Verschiedenheit von Lösungen auszudrücken (Kapitel 2.2).

[9] eine Begrenzung. Der Bound kann für jede Dimension verschieden sein.
[10] Der dominierte Bereich eines Punktes geht bei Minimierungsproblemen bis unendlich.

Template $T = \{t_i \in \{0, 1, x\} | i \in \{1, \ldots, m\}\}$ ist eine im binären Entscheidungsraum (teilweise) vorgegebene Struktur von E; x bedeutet dabei *nicht vorgegeben.*

Zielfunktionsraum Z ist die Abbildung des Entscheidungsraums in die zu optimierenden Eigenschaften der Lösung.

Dimension dim ist die Anzahl der Zielfunktionen, die Z aufspannen. In dieser Arbeit gilt immer $dim = 2$.

Minimierungsproblem: alle Zielfunktionen müssen minimiert werden. Wenn nicht ausdrücklich anders erwähnt, wird in dieser Arbeit immer von einem Minimierungsproblem ausgegangen.

Maximierungsproblem: alle Zielfunktionen müssen maximiert werden.

Indikator ist ein Operand, der etwas über die Qualität der Lösung oder einer ganzen Lösungsmenge aussagt.

Front siehe Pareto-Front.

Pareto-Front besteht aus den nicht-dominierten Individuen (siehe Kapitel 1.5.5).

Aufteilung in Fronten siehe Kapitel 2.3.1.

Edge Ed siehe Kapitel 2.3.2.

Population $S = \{s_i | i \in \{1, \ldots, n\}\}$besteht aus **n** Individuen.

Lösungsmenge siehe Population.

Individuum $s_i = \{E_i, Z_i\} \in S$ besteht aus $E_i = \{e_{i,1}, \ldots, e_{i,m}\}$, welche zu den Zielfunktionswerten $Z_i = \{z_{i,1}, \ldots, z_{i,dim}\}$ führen.

Lösung siehe Individuum.

1.5.5 Präferenzrelationen

Das Individuum s_i ist

dominant: $\nexists s_j \in \{S \setminus s_i\}, k \in \{1, \ldots, dim\} | z_{j,k} \prec z_{i,k}$

dominiert: $\exists s_j \in S | \forall k \in \{1, \ldots, dim\} : z_{j,k} \preceq z_{i,k} \wedge \exists l \in \{1, \ldots, dim\} : z_{j,l} \prec z_{i,l}$

Pareto-optimal, wenn es **nicht dominiert** ist.

unvergleichbar zu Individuum s_j, iff $\exists k, l \in \{1, \ldots, dim\} | (z_{j,k} \prec z_{i,k}) \wedge (z_{i,l} \prec z_{j,l})$. Alle Pareto-optimalen Individuen sind unvergleichbar zueinander.

2 Ansatz 1: Diversität

2.1 Algorithmus

Algorithm 1 Ansatz 1

Require: *alpha* {Grösse der Endpopulation}
Require: Population S {bestehend aus $n \geq alpha$ Individuen s_i}
Require: *gc* {Faktor für $distance\_to\_front$}
Require: *gd* {Faktor für *diversity*}
1: entferne alle Duplikate
2: **if** #Individuen in Front $\geq alpha$ **then**
3: entferne alle dominierten Individuen s_i aus S
4: reduziere S mit Hilfe des Hypervolumens
5: **else**
6: $calculate\_distance\_to\_front()$ {berechnet für jedes Individuum die Distanz zur Front (siehe Kapitel 2.3) }
7: $C \leftarrow \frac{gc}{1131}$ {1131 ist die grösste Distanz in der Initialpopulation mit Seed 0, siehe Testaufbau in Kapitel 5.1}
8: **while** $size(S) > alpha$ **do**
9: $calculate\_diversity()$ {berechnet für jedes Individuum die Diversität (siehe Kapitel 2.2) }
10: $D \leftarrow \frac{gd}{m*n}$ {(analog zu C) es wird hier versucht, $I_{diversity}$ mit $I_{dist\_to\_front}$ vergleichbar zu machen.}
11: entferne Individuum $s_i \notin$ Front aus S , das $D * Diversity(s_i) - C * dist\_to\_front(s_i)$ minimiert
12: **end while**
13: **end if**
14: **return** S

Der strukturelle Gütewert ist in diesem Kapitel die Diversität der Individuen (Unterkapitel 2.2). Als zielfunktionswertbasierter Gütewert wird hier die Distanz zur Front genommen (Unterkapitel 2.3).

Um die Diversität zu erhalten und gleichzeitig die Zielfunktionswerte zu optimieren, kann man den Algorithmus 1 verwenden. Er basiert auf den zwei Indikatoren $I_{Distanz\_zur\_Front}$ und $I_{Diversity}$, die je mit einem Gewichtungsfaktor *gc* respektive *gd* multipliziert werden. In jeder Runde wird das Individuum s_i aus der Population S gelöscht, das

$$Fitness_i = gd * D_i - gc * Distanz_i$$

minimiert, jedoch nicht zur Front gehört. Dass kein[11] Individuum aus der Front entfernt wird ist wichtig, um die erste Bedingung der Problembeschreibung (Kapitel 1.2) zu gewährleisten.

Zu Beginn jedes Evolutionären Algorithmus ist die Diversität normalerweise sehr hoch[12]. Sie nimmt dann ab, weil sich die guten Lösungen meistens nicht sehr voneinander unterscheiden. Die Diversität wird hier durch Einbussen bei der Zielfunktionsoptimierung hoch gehalten.

Es besteht bereits ein Ansatz zur Erhaltung der Diversität [5]. Dieser verwendet Niching, welches das Problem (mehrmals) in Teilprobleme aufteilt und diese optimiert. Die Diversität bleibt durch die Optimierung von unterschiedlichen Teilproblemen ebenfalls erhalten, ohne sie explizit zu optimieren.

2.2 Diversität

Die Diversität drückt die Verschiedenheit aus.

Die Diversität D von Individuum i ist im binären Entscheidungsraum (Kapitel 1.4) wie folgt definiert:

$$D_i = \frac{\sum_{j=1}^{n} HammingDistanz(E_i, E_j)}{m * (n-1)}$$

.

Die Hamming Distanz ist die Anzahl Stellen, in denen sich E_i und E_j unterscheiden [6]. Darum gilt $HammingDistanz(E_i, E_i) = 0$.

2.2.1 Diversität einer Population

In dieser Arbeit wird die Diversität D einer Population S als Durchschnitt aller D_i angenommen:

$$D = \frac{\sum_{i=1}^{n} D_i}{n} \ .$$

Bei grossen Populationsgrössen ($n \gg 1$) ist $D \in [0, 0.5]$.

2.2.2 Beispiel

Im Beispiel gilt $n = 3$ und $m = 5$. Der Entscheidungsraum

[11] ausser, wenn es zuviele in der Front hat

[12] Die Initialpopulation wird üblicherweise durch randomisierte Verfahren initialisiert und dadurch ist die 'Bit-Diversität' (Wahrscheinlichkeit, dass das selbe Bit bei einem anderen Individuum gleich ist) 0.5.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

ergibt die Diversitäten

$$\begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} = \begin{bmatrix} (3+2)/(5*(3-1)) \\ (3+5)/(5*(3-1)) \\ (2+5)/(5*(3-1)) \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.8 \\ 0.7 \end{bmatrix}$$

für die einzelnen Individuen und

$$D = \frac{0.4 + 0.8 + 0.7}{3} = 0.6\overline{3}$$

für die Population.

2.3 Distanz zur Front

Während *Diversität* ein Mass ist, das auf dem Entscheidungsraum basiert, ist *Distanz zur Front* auf dem Zielfunktionsraum aufgebaut. Es könnte auch ein anderer Indikator verwendet werden.

Die Funktion $calculate\_distance\_to\_front()$ implementiert den Indikator $I_{Distanz\_zur\_Front}$. Dieser besteht aus den jeweiligen Distanzen der einzelnen Individuen zur Front.

Es wurden mehrere Möglichkeiten ausprobiert, die Distanz zur Front *Distanz* festzulegen:

2.3.1 FrontNumber

Die Population wird wie folgt in Fronten aufgeteilt und den Individuen s_i werden die entsprechenden $Distanz_i = FrontNumber$ zugewiesen:

FrontNumber=0: alle Individuen, die nicht dominert werden (auf der Front liegen).

FrontNumber=1: alle Individuen, die nur von Individuen mit FrontNumber 0 dominiert werden.

FrontNumber=k: alle Individuen, die nur von Individuen mit FrontNumber 0 bis $k-1$ dominiert werden.

Je tiefer die FrontNumber des Individuums, desto näher ist es an der Front.

Dieser Ansatz ist wegen zu hoher Ungenauigkeit und Unvorhersehbarkeit nicht im Algorithmus 1 enthalten.

2.3.2 Absolute Distanz

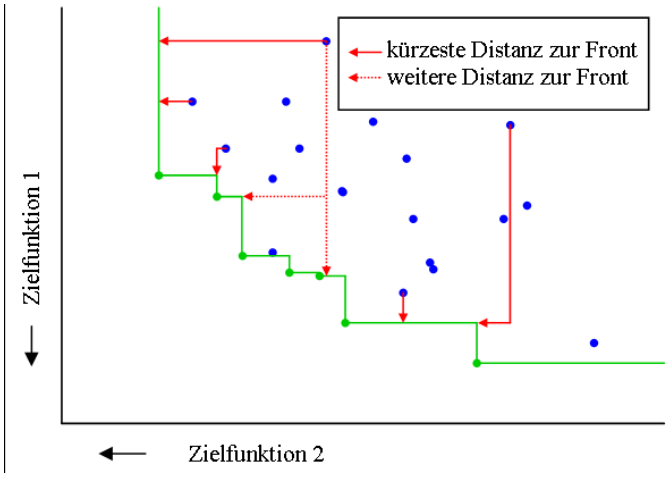


Abbildung 3:
Distanz zur Front: es wird immer die kürzeste Distanz genommen.

Zur Berechnung der absoluten Distanz ist der Vekor Ed nötig:

Edge Ed Annahmen:

- $dim = 2$
- $front$ besteht aus allen l Individuen der Front. Diese sind geordnet (kleinster Wert von erster Dimension zuerst)
- $front_{i,j}$ sei der Zielfunktionswert z_{ij} des Individuums $s_i \in front$

Dann gilt:

$$Ed = \begin{bmatrix} front_{0,1} & \infty \\ front_{1,1} & front_{0,2} \\ \vdots & \vdots \\ front_{k,1} & front_{k-1,2} \\ \vdots & \vdots \\ front_{l,1} & front_{l-1,2} \\ \infty & front_{l,2} \end{bmatrix}$$

Grafisch gesehen sind die Punkte von Ed dort auf der Front, wo sich die von verschiedenen Individuen dominierten Bereiche kreuzen.

Die Distanz ist der kürzeste Abstand zur Front, wobei der Abstand in der Betragsummennorm gemessen wird [7]. Das heisst: Wie weit muss Z_i im

Zielfunktionsraum verschoben werden, um auf der Front zu sein:

$$Distanz_i = min_{(j \in \{1,\ldots,l+1\})}(\sum_{d=1}^{dim} max(0,\ z_{i,d} - Ed_{j,d}).$$

Der Code hierzu ist in Algorithmus 2 beschrieben; ein grafisches Beispiel ist in Abbildung 3.

Algorithm 2 Distanzberechnung mit absoluter Distanz

Require: Population S
Require: $front$ {Vektor, der die Zielfunktionswerte der Front beinhaltet, GEORDNET}
Require: Vektor Ed {Edge}

```
for all Individuen s_i aus S do
    minDist ← ∞ {aktuell kleinste Distanz}
    for all Elemente Ed_j ∈ Ed do
        tempDist ← 0 {temporäre Variable}
        for all Dimensionen d des Zielfunktionsraumes do
            if Ed_{j,d} ≺ z_{i,d} then {≺ entspricht '<'}
                tempDist ← (tempDist + (z_{i,d} − Ed_{j,d}))
            end if
        end for
        minDist ← min(minDist, tempDist)
    end for
    setDist2Front(s_i, minDist)
end for
```

3 Ansatz 2: Diversität

3.1 Algorithmus

Der Algorithmus von Ansatz 2 berechnet die nächste Generation aufgrund des mit der Diversität, einem auf der Struktur basierendem Gütewert, gewichteten Hypervolumens (Unterkapitel 3.2). Dabei werden in jeder Iteration alle Individuen nacheinander provisorisch ausgeschlossen, das gewichtete Hypervolumen berechnet und anschliessend dasjenige Individuum aus der Population entfernt, welches beim Ausschluss zum höchsten (verbleibenden) gewichteten Hypervolumen führt. Bei gleichem gewichtetem Hypervolumen wird das Individuum entfernt, welches die *geringste Diversität* hat. Dieser Prozess wird solange wiederholt, bis genügend Individuen entfernt wurden.

Die erste Bedingung der Problemstellung (Kapitel 1.2) ist dahingegend erfüllt, dass die Individuen der Front immer[13] zum gewichteten Hypervolumen beitragen und sich eine hohe Gewichtung stärker auswirkt, als bei dominierten Individuen.

Es gibt dabei zwei leicht unterschiedliche Algorithmen:

Dynamischer Algorithmus: Die Diversität wird *nach dem provisorischen Entfernen* für die *verbleibenden* Individuen neu berechnet (Zeile 7 im Algorithmus 3). Da die auszuschliessenden Individuen keine Diversität haben, ist *geringste Diversität* die Summe aller Diversitäten der verbleibenden Individuen.

Statischer Algorithmus: Die Diversität wird in jeder Runde nur zu Beginn berechnet (Zeile 5 im Algorithmus 4).

3.1.1 Dynamischer Algorithmus

Im Vergleich zum statischen Algorithmus (Kapitel 3.1.2, Algorithmus 4) führt der dynamische Algorithums 3 zu sehr schlechten Resultaten in der Zielfunktionsoptimierung und in der Diversität.

Der Grund dafür ist, dass das gewichtete Hypervolumen von den Pareto-optimalen Punkten und den Punkten mit einer hohen Diversität abhängt. Es ist wahrscheinlich, dass die Pareto-optimalen Punkte auch eine hohe[14] Diversität aufweisen. Der Algorithmus maximiert das gewichtete Hypervolumen. Dieses kann nun dadurch beeinflusst werden, dass durch Entfernen eines einem dominanten Punkt ähnlichen Punktes, die Gewichtung des do-

[13] ausser ihre Gewichtung wäre 0

[14] *durchschnittlich* reicht, um die Kettenreaktion auszulösen

Algorithm 3 Ansatz 2, dynamischer Algorithmus

Require: *alpha* {Grösse der Endpopulation}
Require: Population S {bestehend aus $n \geq alpha$ Individuen}
1: **while** $size(S) > alpha$ **do**
2: $maxHyp \leftarrow 0$
3: $maxDiversity \leftarrow 0$
4: $bound \leftarrow 1.1 * z_{max}$ {maximaler Zielfunktionswert in S}
5: **for all** Individuen $s_i \in S$ **do**
6: $newPop \leftarrow S \setminus s_i$
7: $totDiversity \leftarrow calculateDiversity(newPop)$
8: **for all** Dimensionen d **do**
9: $sPop_d \leftarrow sortPop(newPop, d)$ {sortiert $newPop$ nach Dimension d}
10: **end for**
11: $hyp \leftarrow gewHypervolume(dim, bound, size(newPop))$ {Algorithmus 6, hat Zugriff auf $sPop$}
12: **if** $(hyp > maxHyp \vee (hyp \equiv maxHyp \wedge totDiversity > maxDiversity))$ **then**
13: $maxHyp \leftarrow hyp$
14: $maxDiversity \leftarrow totDiversity$
15: $individual\_to\_remove \leftarrow s_i$
16: **end if**
17: **end for**
18: $S \leftarrow S \setminus individual\_to\_remove$
19: **end while**
20: **return** S

Algorithm 4 Ansatz 2: statischer Algorithmus

Require: *alpha* {Grösse der Endpopulation}
Require: Population S {bestehend aus $n \geq alpha$ Individuen}
while $size(S) > alpha$ **do**
  $maxHyp \leftarrow 0$
  $maxDiversity \leftarrow 0$
  $bound \leftarrow 1.1 * z_{max}$ {maximaler Zielfunktionswert in S}
  $calculateDiversity(S)$
  for all Individuen s_i in S **do**
    $newPop \leftarrow S \setminus s_i$
    for all Dimensionen d **do**
      $sPop_d \leftarrow sortPop(newPop, d)$ {sortiert $newPop$ nach Dimension d}
    end for
    $hyp \leftarrow gewHypervolume(dim, bound, size(newPop))$ {Algorithmus 6, hat Zugriff auf $sPop$}
    if $hyp > maxHyp \vee$
    $(hyp \equiv maxHyp \wedge diversity(s_i) > maxDiversity)$ **then**
      $maxHyp \leftarrow hyp$
      $maxDiversity \leftarrow diversity(ind)$
      $individual\_to\_remove \leftarrow s_i$
    end if
  end for
  $S \leftarrow S \setminus individual\_to\_remove$
end while
return S

minanten Punktes steigt und damit auch das gewichtete Hypervolumen[15]. Die Diversität hingegen leidet sehr stark unter diesem Phänomen. Es werden nicht diejenigen Punkte entfernt, welche relativ zur *gesamten Population* divers sind, sondern jene, die divers zu den *dominierenden Punkten* sind. Das wiederum bedeutet, dass sich die dominierten Punkte immer mehr ähnlich werden und die Diversität der Population sinkt. Die Zielfunktionsoptimierung ist nicht mehr gewährleistet.

Im schlimmsten Fall gibt es nur ein dominantes Individuum[16] mit Diversität 1. Damit fällt die (durchschnittliche) Diversität der Gesamtpopulation auf

$$\frac{1 + \frac{n-1}{n-1}}{n} = \frac{2}{n} \approx 0 \ll 0.5 \ ,$$

da alle anderen Individuen identisch sind. Das gewichtete Hypervolumen ist in diesem Fall durch die Gewichtung hoch, das ungewichtete jedoch sehr schlecht, weil nicht mehr optimiert wird.

3.1.2 Statischer Algorithmus

Durch das statische[17] Berechnen der Diversität (Algorithmus 4) wird diese zwar nicht exakt[18] berechnet, dafür tritt das Problem des dynamischen Algorithmus (Kapitel 3.1.1, Algorithmus 3) nicht auf.

3.1.3 Beispiel

Der Algorithmus wird anhand eines einfachen Bespiels veranschaulicht. Die Population in Abbildung 4, bestehend aus vier Individuen mit (relativen) Diversitäten[19] 2, 5, 7 und 10, soll auf eine Population mit zwei Individuen reduziert werden.

In der ersten Iteration (siehe Abbildung 5) wird für jedes Individuum ein gewichtetes Hypervolumen berechnet, das alle Individuen ausser dem betreffenden berücksichtigt. Das Individuum, welches den kleinsten Verlust des Hypervolumens aufweist, wird entfernt; hier ist es das mit Diversität '2'.

In der zweiten Iteration (siehe Abbildung 6) stehen nur noch drei Individuen zur Verfügung. Die Diversitäten (Gewichtungen) ändern sich durch das

[15] zur Veranschaulichung: s_d ist das dominante Individuum, s_i hat eine ähnliche Struktur wie s_d ($Z_d \cong Z_i$) und wird entfernt $\Rightarrow$ Die Gewichtung von s_d steigt $\Rightarrow$ das gewichtete Hypervolumen steigt.

[16] alle anderen Punkte unterscheiden sich zu 100% von diesem.

[17] pro zu entfernendem Individuum nur ein Mal

[18] es wird nunmehr mit einer Näherung der Diversität gerechnet, weil sich diese ja mit dem Entfernen eines Individuums wieder ändert.

[19] das gleiche Beispiel ist auch auf den Algorithmus mit vorgegebenen Strukturen (Kapitel 4) anwendbar, wenn man anstatt der Diversität die Nähe zu T einsetzt.

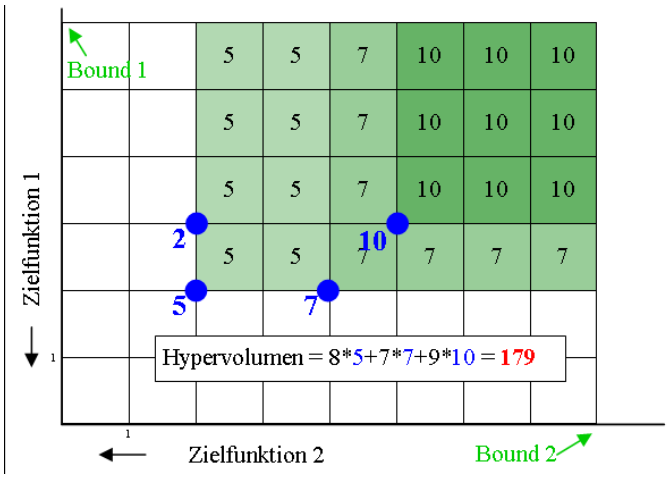
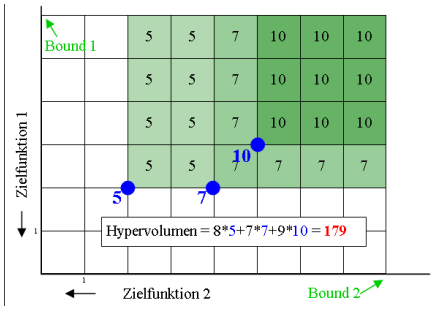
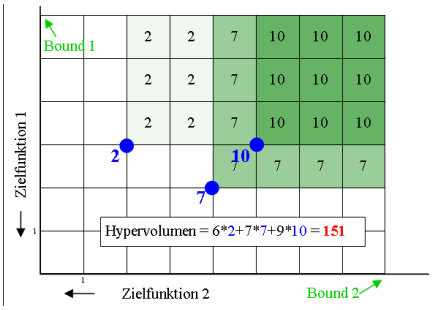


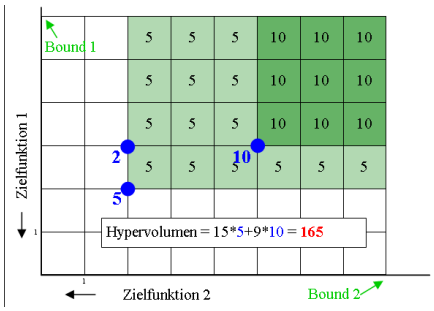
Abbildung 4:
Beispiel zu Algorithmus 4: Initialpopulation



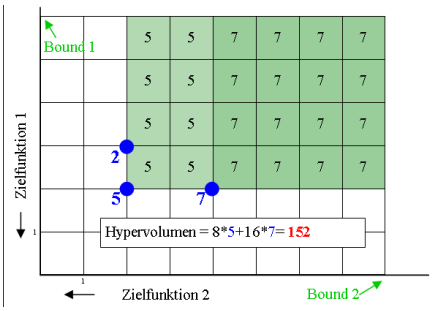
(a) '2' entfernt



(b) '5' entfernt

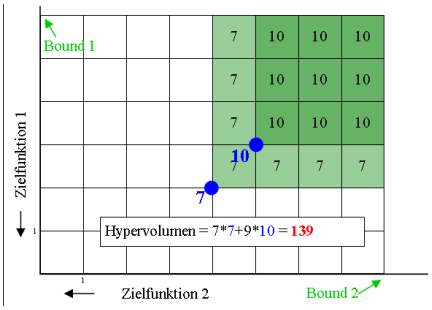


(c) '7' entfernt

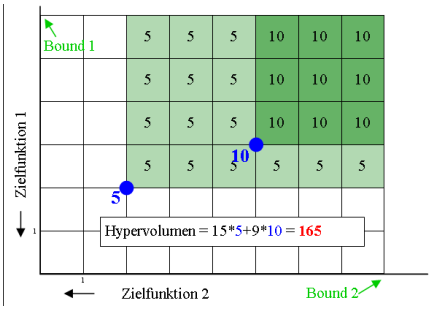


(d) '10' entfernt

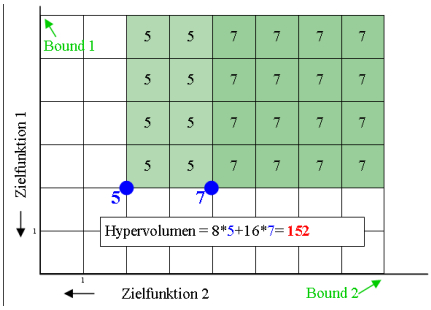
Abbildung 5:
Beispiel zu Algorithmus 4: erste Iteration:
Das beste Hypervolumen wird in (a) erreicht.



(a) '5' entfernt



(b) '7' entfernt



(c) '10' entfernt

Abbildung 6:
Beispiel zu Algorithmus 4: zweite Iteration:
Das beste Hypervolumen wird in (b) erreicht.

Entfernen eines Individuums, was im Beispiel der Einfachheit halber nicht berücksichtigt wurde. Das beste Hypervolumen wird bei Entfernen des Individuums mit Diversität '7' erzielt. Somit sind die beiden verbleibenden Individuen für die nächste Generation diejenigen mit Diversität '5' und '10'.

3.1.4 effizienterer Algorithmus

Der statische Algorithmus 4 kann unter einer Bedingung[20] um ein vielfaches effizienter gemacht werden. Dabei wird berücksichtigt, dass Individuen s_i, die

- nicht zur Front gehören und
- eine kleinere Diversität haben als ein s_i dominierendes Individuum

selber keinen Beitrag zum Hypervolumen[21] leisten und somit entfernt werden können, ohne alle Hypervolumen zu berechnen. Die Funktionalität ist in Algorithmus 5 beschrieben.

Algorithm 5 Modifikation zur Effizienzsteigerung, Teil von Algorithmus 4

$\vdots$ {while...}
if $\exists$ Individuen $s_a, s_b \in S \| (s_a$ dominiert $s_b \wedge diversity(s_a) \geq diversity(s_b))$
{Ind ist Menge aller Individuen s_b, die die Bedingung erfüllen} **then**
  $pop \leftarrow pop \setminus s_i \| s_i \in Ind \wedge diversity(s_i) \equiv min(diversity(Ind))$
else
$\vdots$
  {$calculateHypervolume()$, $removeIndividual()$}
end if

3.2 Gewichtetes Hypervolumen

Das gewichtete Hypervolumen ([8], siehe Abbildung 7) ist wie das Hypervolumen (Kapitel 1.5.3) ein Indikator zur Beurteilung einer Güte einer Population. Im Unterschied zum ungewichteten Hypervolumen können dominierte Punkte einen Beitrag zum gewichteten Hypervolumen leisten.

Jedes Individuum erhält neben den Zielfunktionswerten auch einen Gewichtungswert *Gewicht*[22]. Jedem Punkt p im Raum wird nun ein Wert $g(p)$ wie

[20] Die Experimente haben gezeigt, dass diese bei der Berechnung mit Algorithmus 4 in 70 bis 100% der Iterationen erfüllt ist.
[21] gewichtet und ungewichtet
[22] hier ist es die Diversität, in Kapitel 4 die Nähe zum Template

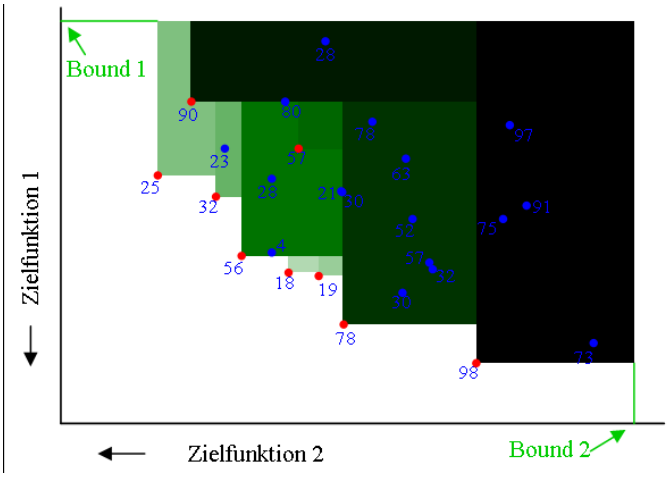


Abbildung 7: gewichtetes Hypervolumen

folgt zugewiesen:

$$g(p) = \begin{cases} 0, & \text{wenn } p \text{ nicht dominiert} \\ max_{p \text{ dominierende Individuen } s_i}(Gewicht_i), & \text{wenn } p \text{ dominiert} \end{cases}$$

.

Das gewichtete Hypervolumen berechnet sich wie folgt:

$$gewichtetesHypervolumen = \int_{alleDimensionen}^{bound} g(p)$$

.

Der Algorithmus 6 dazu basiert auf Rekursion. Die Implentierung erfolgte nach einer Vorlage (Kapitel B.7).

Algorithm 6 gewichtetes Hypervolumen: rekursiver Algorithmus

Require: d {die aktive Dimension}
Require: $bound$
Require: $limit$ {gewünschter Stand der Berechnung des Aufrufs}
Require: Zugriff auf $sPop_i$ {nach Dimension i geordnete Population}
Ensure: $bound > max(z_{i,j}),\ \forall i \in \{1,\ldots,n\}, j \in \{1,\ldots,dim\}$
1: $hyp \leftarrow 0$
2: $maxDiv \leftarrow 0$
3: $S \leftarrow sPop_d$
4: **for** $i = 0 \ldots limit$ **do**
5: $maxDiv \leftarrow max(maxDiv, diversity(s_i \in S))$
6: **if** $i < limit$ **then**
7: $extrusion \leftarrow z_{(i+1),d} - z_{i,d}$
8: **else**
9: $extrusion \leftarrow bound - z_{i,d}$
10: **end if**
Ensure: $extrusion >= 0$
11: **if** $d = 0$ **then**
12: $hyp \leftarrow (hyp + extrusion * maxDiv)$
13: **else**
14: $hyp \leftarrow (hyp + extrusion * gewHypervolume(d-1, bound, i))$
15: **end if**
16: **end for**
17: **return** hyp

4 Ansatz 2: Templates - vorgegebene Strukturen

Wie im vorherigen Kapitel 3 wird hier ein auf der Struktur basierender Gütewert als Gewichtungsfaktor verwendet. Der Code zum Ansatz mit *Templates* entspricht fast vollständig demjenigen von Algorithmus 4, der in Kapitel 3.1 beschrieben ist. Der effizienzsteigernde Zusatz (Kapitel 3.1.4) ist ebenfalls implementiert. Anstelle der Diversität D_i (Kapitel 2.2) wird jetzt jedoch die Distanz zur gegebenen Struktur V_i als Gewichtung verwendet, welche nicht in jeder Iteration neu berechnet werden muss: $calculateDistToTemplate(S, T)$ wird anstelle von $calculateDiversity(S)$ gebraucht, jedoch bereits in Zeile 1.

Die Methode der Linearkombination, wie sie im Algorithmus 1 verwendet wird, wird hier nicht weiterverfolgt, weil die Experimente (Kapitel 5.3 und 5.4) die Überlegenheit des zweiten Ansatzes gezeigt haben.

Das Problem muss immer das selbe sein. Das heisst, der Entscheidungsraum darf sich nicht ändern (Tabelle 1 in Kapitel A.2.1).

4.1 Distanz zum Template

Eine vorgegebene Struktur T kann als Individuum angesehen werden, das jedoch $|x|$ *Don't Care*-Werte x beinhalten darf. Die Eigenschaften, die die Individuen an den Stellen der *Don't Care* aufweisen, sind vom Template unabhängig und können zur Optimierung der Zielfunktionswerte frei verwendet werden.

Die Distanz des Individuums s_i zum Template T ist wie folgt definiert:

$$V_i = \frac{\sum_{\substack{j=1,\\ T_j \neq x}}^{m} |T_j - e_{i,j}|}{|T|}$$

wobei $|T| = m - |x|$.

4.1.1 Beispiel

Wie im Beispiel in Kapitel 2.2.2 gilt $m = 5$ und $n = 3$. Die vorgegebene Struktur sei: $T = \begin{bmatrix} 1 & x & x & 0 & 1 \end{bmatrix}$.

Für die Individuen $\begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ ergeben sich die Distanzen $\begin{bmatrix} 1/3 \\ 1/3 \\ 2/3 \end{bmatrix}$.

4.2 Effizienz

Der Rechenaufwand des Algorithmus ist im Rahmen desjenigen der Diversität 2, jedoch strikt kleiner, da die Berechnung von V_i weniger rechenintensiv ist als die von D_i.

Dennoch benötigen Testläufe mit dem Template-Algorithmus (Kapitel 5.5) wesentlich länger als der Testlauf mit Diversität 2 (Kapitel 5.4). Der Grund ist, dass bei den Templates viel weniger oft der effiziente Zusatz (Algorithmus 5) zur Anwendung kommt. Je grösser das Template ist, desto öfter kann die Berechnung der Hypervolumen übersprungen werden.

5 Experimente

5.1 Testaufbau

Sämtliche[23] Experimente und das Vergleichsexperiment (Kapitel 5.2) haben ein gemeinsames Grundgerüst, damit sie auch miteinander vergleichbar sind:

- auf PISA basierend ([9])
- Populationsgrösse[24]: n = 100
- 11 Testläufe pro Algorithmus[25]
- Anzahl Generationen: 200
- Anzahl Elemente im Entscheidungsraum[26] des Rucksackproblems: m = 100

Da nur das Rucksackproblem (Kapitel 1.4) für sämtliche Experimente verwendet wird, sind auch die Einstellungen für die Variation immer die selben:

- Mutationstyp: 1 (1-bit-Mutation)
- Rekombinationstyp: 1 (1-Punkt-Crossover)
- Mutationswahrscheinlichkeit: 1
- Rekombinationswahrscheinlichkeit: 0.5

Der Gütewert wird immer mit dem Hypervolumenindikator angegeben (Kapitel 1.5.3). Es werden Boxplots der 11 Testläufe ausgegeben, die die Verteilung der Werte wiedergeben [10].

5.2 Referenzalgorithmus: SIBEA

Um etwas über die Güte der entwickelten Algorithmen sagen zu können, muss man sie nicht nur mit sich selbst vergleichen, sondern auch mit einem bestehenden Algorithmus. SIBEA[27] [8] ist ein einfacher Selektor, der wie in Algorithmus 7 beschrieben funktioniert. Er wird als Vergleichsalgorithmus verwendet, um etwas über die Qualität der entwickelten Algorithmen aussagen zu können.

[23] ausser das Experiment in Kapitel 5.5.3

[24] es werden in jeder Generation von 100 Individuen 100 neue generiert und aus diesen 200 wieder 100 selektiert.

[25] mit Seed = {0,1,...,10}

[26] diese sind immer die selben (Tabelle 1)

[27] Abkürzung für *Simple Indicator Based Evolutionary Algorithm*

Algorithm 7 SIBEA

Require: $alpha$\{Grösse der Endpopulation\}
Require: Population S
1: $frontPart \leftarrow partition(S)$\{teilt S in Fronten auf\}
2: entferne Fronten von hinten, bis keine ganze Front mehr entfernt werden kann, ohne dass $size(S) < alpha$
3: **while** $size(S) > alpha$ **do**
4: entferne Individuum s_i aus S, welches geringsten Einfluss auf das Hypervolumen der hintersten verbleibenden Front hat
5: **end while**
6: **return** S

Der Sourcecode ist in JAVA [3].

Das Vergleichsexperiment wird mit den gleichen Parametern wie die entwickelten Algorithmen durchgeführt.

5.3 Experimente mit Ansatz 1 (Diversität)

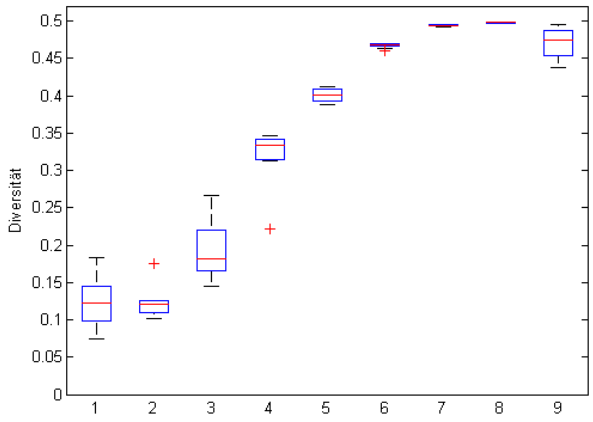
Der Algorithmus der Diversität 1 (Algorithmus 1) bietet zwei Freiheitsgrade:

- gc - Gewichtung der Distanz zur Front (Gütewert)
- gd - Gewichtung der Diversität (struktureller Gütewert).

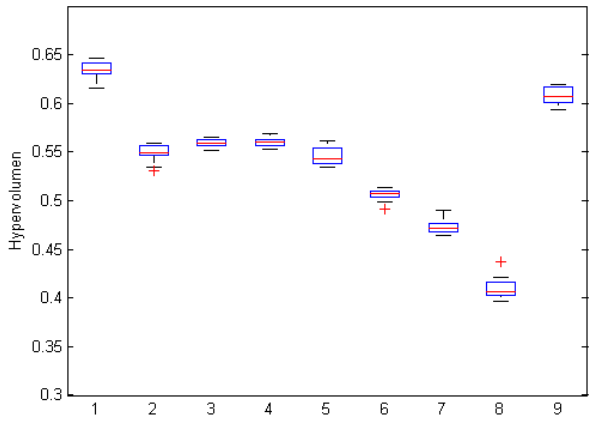
Die hier durchgeführten Experimente zeigen den Zusammenhang dieser beiden Faktoren:

- $gc = 1$ und $gd = 0$ (Kapitel 5.3.1)
- $gc = 0.75$ und $gd = 0.25$ (Kapitel 5.3.2)
- $gc = 0.5$ und $gd = 0.5$ (Kapitel 5.3.3)
- $gc = 0.25$ und $gd = 0.75$ (Kapitel 5.3.4)
- $gc = 0.125$ und $gd = 0.875$ (Kapitel 5.3.5)
- $gc = 0.05$ und $gd = 0.95$ (Kapitel 5.3.6)
- $gc = 0$ und $gd = 1$ (Kapitel 5.3.7)

Eine Zusammenfassung der Experimente mit Diversität findet sich den Abbildungen 8 und 9.



(a) Diversität



(b) Hypervolumen

Abbildung 8:
Resultate der Algorithmen von Diversität 1 und 2:
1: Vergleichsexperiment (Kapitel 5.2)
2: Diversität 1: $gc = 1$ und $gd = 0$ (Kapitel 5.3.1)
3: Diversität 1: $gc = 0.75$ und $gd = 0.25$ (Kapitel 5.3.2)
4: Diversität 1: $gc = 0.5$ und $gd = 0.5$ (Kapitel 5.3.3)
5: Diversität 1: $gc = 0.25$ und $gd = 0.75$ (Kapitel 5.3.4)
6: Diversität 1: $gc = 0.125$ und $gd = 0.875$ (Kapitel 5.3.5)
7: Diversität 1: $gc = 0.05$ und $gd = 0.95$ (Kapitel 5.3.6)
8: Diversität 1: $gc = 0$ und $gd = 1$ (Kapitel 5.3.7)
9: Diversität 2 (Kapitel 5.4)

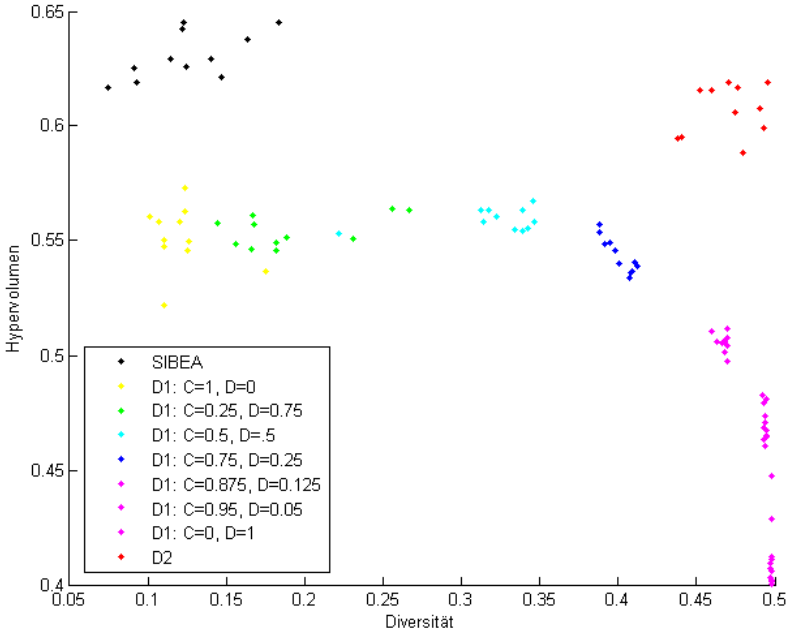


Abbildung 9:
Resultate der Algorithmen von Diversität 1 und 2
D1: Diversität 1, D2: Diversität 2
C $\equiv gc$, D $\equiv gd$

5.3.1 gc = 1, gd = 0

Das Resultat von Diversität 1 verglichen mit dem Vergleichsexperiment ist in Abbildung 8 dargestellt. Es ist erkennbar, dass die Front (das Hypervolumen) schlechter ist. Der Grund dafür ist, dass die Zielfunktionsmethode des *Fronten abschälen mit Hypervolumen* des SIBEA besser ist als *Distanz zur Front*.

Obwohl im Algorithmus 1 alle Duplikate, die die Diversität negativ beeinflussen, entfernt werden, ist die Diversität nicht höher als beim Vergleichsexperiment. Das lässt darauf schliessen, dass Duplikate nur selten vorkommen.

5.3.2 gc = 0.75, gd = 0.25

Das Hypervolumen nimmt im Vergleich zum vorherigen Experiment nicht ab. Durch Miteinbeziehen der Diversität kann dieses jedoch gesteigert werden.

5.3.3 gc = 0.5, gd = 0.5

Ähnlich dem letzten Experiment nimmt die Diversität durch die stärkere Gewichtung abermals zu, ohne Einbussen beim Hypervolumen hinnehmen zu müssen.

5.3.4 gc = 0.25, gd = 0.75

Das Hypervolumen bleibt auf einem mit dem ersten Experiment vergleichbaren Niveau, während die Diversität erneut zunimmt.

5.3.5 gc = 0.125, gd = 0.875

Das Experiment wird durchgeführt, weil zwischen dem letzten (Kapitel 5.3.4) und demjenigen mit $gd = 0$ (Kapitel 5.3.7) ein unausgefüllter Raum[28] ist, siehe Abbildung 9. Das Verhalten zwischen guter Diversität und schlechtem Hypervolumen soll untersucht werden.

Es zeigt sich, dass das Hypervolumen nun abnimmt.

[28] Das Hypervolumen nimmt zwischen dem letzten Experiment und demjenigen mit $gd = 0$ stark ab.

5.3.6 gc = 0.05, gd = 0.95

Analog zum letzten Experiment ist dieses Experiment dazu da, die *Lücke* zu untersuchen.

Erwartungsgemäss nimmt das Hypervolumen wieder stark ab, ohne grosse Zunahme der Diversität.

5.3.7 gc = 0, gd = 1

Wie erwartet ist die Diversität nun maximal[29], das Hypervolumen jedoch nicht optimiert und vergleichbar mit der Initialpopulation aus Tabelle 1.

5.4 Experiment mit Ansatz 2 (Diversität)

Es wurde nur ein Experiment mit dem (statischen, effizienteren) Diversität-2-Algorithmus 4 durchgeführt.

Der Vergleich mit SIBEA (Kapitel 5.2) und Diversität 1 ist in den Abbildungen 8 und 9 dargestellt.

Der Algorithmus von Diversität 2 weist eine viel höhere Diversität auf als der Referenzalgorithmus, jedoch auch ein kleineres Hypervolumen.

Im Vergleich zum Algorithmus von Diversität 1 hingegen ist das Hypervolumen immer grösser, die Diversität erreicht jedoch nicht ganz so hohe Werte.

5.5 Experimente mit Ansatz 2 (Templates)

5.5.1 Auswahl der Strukturen

Da E immer gleich ist (Tabelle 1), kann man statisch Individuen auswählen. Für Testzwecke wird jedem Individuum folgendes zugewiesen:

- *Gut*: '1', wenn Element Mehrwert[30] verspricht, '0' sonst
- *Top 20*: Element gehört zu den extremsten 20 Elementen ('1', '0' sonst)
- *Top 60*: Element gehört zu den extremsten 60 Elementen ('1', '0' sonst)
- *Top 100*: alle Elemente haben '1'

[29] Maximum ≈ 0.5, bei $100 \gg 1$ Individuen
[30] Profit/Gewicht ist *gut*

Dabei wurde der *Gut*-Wert wie folgt bestimmt:

$$Gut_i = \begin{cases} 1, & \alpha_i > Median(\alpha_j) \forall j \in \{1, \ldots, m\} \\ 0, & \text{sonst} \end{cases}$$

, wobei $\alpha_k = \frac{p1_k * p2_k}{w1_k * w2_k}$[31].

Die extremsten Elemente wurden von vorne aus dem Vektor *extrem* ausgewählt:

$$extrem = ordne(\beta_i) \| i \in \{1, \ldots, m\} \| \beta_i = \max(\alpha_i, {\alpha_i}^{-1})$$

5.5.2 Experimente

Es werden neun Experimente durchgeführt:

- für Templates der Länge
  - 20
  - 60
  - 100
- und für Elemente die
  - möglichst *gut*
  - möglichst *schlecht*
  - *zufällig*[32]

  gewählt sind.

20 schlechte Elemente entspricht dabei einer Struktur T, die an den Stellen der ersten 20 Elemente in *extrem* je den Wert $T_{extrem_i} = (1 - Gut_{extrem_i})$ hat und x sonst.

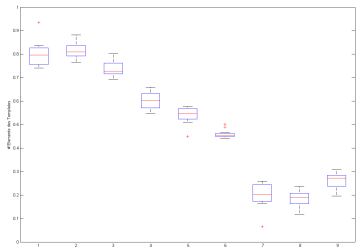
Die Resultate aller neun Experimente sind in Abbildung 10 dargestellt.

Die vorgegebenen Strukturen werden tatsächlich ziemlich gut erreicht, dies jedoch auf Kosten des Hypervolumens. Der Zielfunktionsraum für das Vergleichsexperiment und das mit *100 guten Elementen* ist in Abbildung 11 dargestellt. Es ist erkennbar, dass die Front zwar nicht weniger breit ist, jedoch bedeutend schlechtere Zielfunktionswerte aufweist.

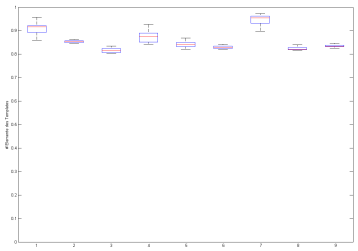
Viel erstaunlicher hingegen ist, dass das Hypervolumen bei allen Templates ungefähr gleich ist. Eine mögliche Erklärung dafür könnte sein, dass es

[31] p ist der Profit, w das Gewicht

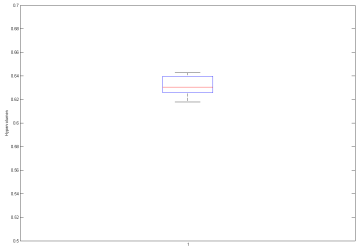
[32] es werden die ersten Elemente genommen, da alle Elemente zufällig sind



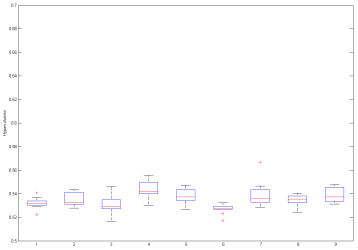
(a) Nähe zum Template des Vergleichsexperiments



(b) Nähe zum Template mit entwickeltem Algorithmus



(c) Hypervolumen des Vergleichsexperiment



(d) Hypervolumina mit entwickeltem Algorithmus

Abbildung 10:
1: Template = 20 gute Elemente
2: Template = 60 gute Elemente
3: Template = 100 gute Elemente
4: Template = 20 zufällige Elemente
5: Template = 60 zufällige Elemente
6: Template = 100 zufällige Elemente
7: Template = 20 schlechte Elemente
8: Template = 60 schlechte Elemente
9: Template = 100 schlechte Elemente

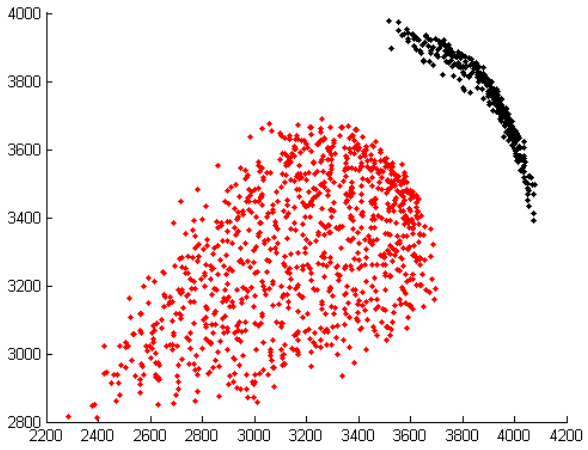


Abbildung 11: Zielfunktionsraum mit den Experimenten:
schwarz: SIBEA
rot: Template = 100 gute Elemente
es handelt sich hier um ein Maximierungsproblem.

zwei Sorten Individuen gibt: diejenigen, welche der vorgegebenen Struktur fast vollständig entsprechen, jedoch keine guten Zielfunktionswerte aufweisen und wenige[33], die zwar gute Zielfunktionswerte haben, jedoch weit von der Struktur entfernt sind. Dass die 'guten' Templates auch wirklich besser sind als die 'schlechten', zeigt sich daran, dass diese viel besser im Resultat des Vergleichsexperimentes enthalten sind.

Aufgrund dieser Resultate der neun Experimente werden zwei zusätzliche Experimente durchgeführt: Kapitel 5.5.3 und 5.5.4.

5.5.3 weniger Individuen

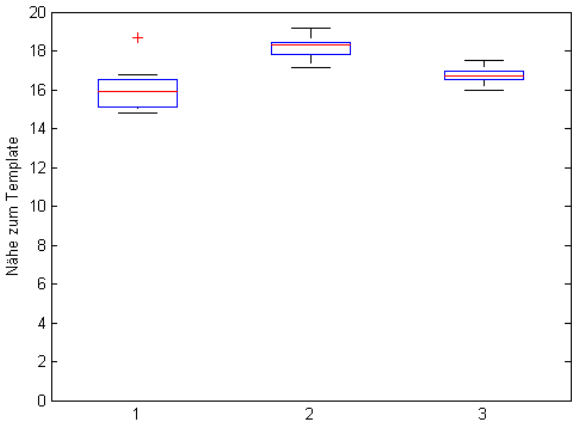
Um den Einfluss der Populationsgrösse auf das Hypervolumen zu untersuchen, wird ein weiteres Experiment durchgeführt.

Es baut auf dem Szenario *20 gute Elemente im Template* auf. Dabei werden folgende Parameter verändert:

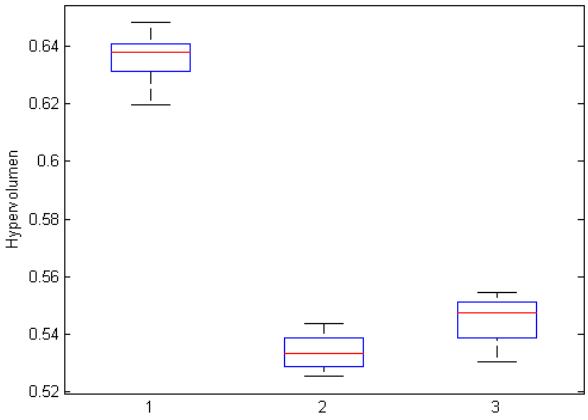
- Populationsgrösse: 50
- Anzahl Generationen: 400.

Die Anzahl Vergleiche zwischen Individuen bleibt dabei gleich.

[33] das erklärt, weshalb der *Durchschnitt der Abweichung zur Struktur* trotzdem gering ist



(a) Nähe zum Template (von 20)



(b) Hypervolumen

Abbildung 12: Template = 20 gute Elemente
1: SIBEA, 100 Individuen, 200 Generationen
2: Template-Algorithmus, 100 Individuen, 200 Generationen
3: Template-Algorithmus, 50 Individuen, 400 Generationen

Das Resultat dieses Experiments ist in Abbildung 12 gezeigt: beim neuen Experiment ist zwar das Hypervolumen leicht höher, das Template dafür weniger gut erfüllt. Die beiden Populationsgrössen sind somit unvergleichbar.[34]

5.5.4 anderes Template

Analog zum Experiment im vorherigen Kapitel 5.5.3 wird ein Experiment durchgeführt, bei dem andere Elemente als in Kapitel 5.5.1 beschrieben als *gut* eingestuft werden.

Durch beobachten der Resultate von SIBEA kann festgestellt werden, dass mehr (ca. 70 %) Elemente *im* Rucksack sind als *nicht*. Der *Gut*-Wert sei nun wie folgt bestimmt:

$$Gut_i = \begin{cases} 1, & \alpha_i \in A \| i \in \{1, \ldots, m\} \\ 0, & \text{sonst} \end{cases}$$

, wobei $\alpha_i = \frac{p_{1_i}}{w_{1_i}} + \frac{p_{2_i}}{w_{2_i}}$ und A aus den grössten $2/3$ aller α_i besteht.

Die Auswahl der extremsten ex Elemente ist nun wie folgt:

$$extremG = \text{alle Individuen nach } \alpha \text{ geordnet, grösste zuerst}$$

$$extrem = \{extremG_1, \ldots, extremG_{\frac{2*ex}{3}}, extremG_{m-\frac{ex}{3}}, \ldots, extremG_m, \}$$

Das Resultat[35] mit dem Szenario *100*[36] *gute* Elemente ist in Abbildung 13 dargestellt.

Auch mit dieser Änderung der Templates vermag man das Hypervolumen nicht merklich steigern.

[34] mit nur 50 Individuen ist der Algorithmus jedoch bedeutend schneller.
[35] nur Hypervolumen
[36] dafür ist *extrem* nicht nötig.

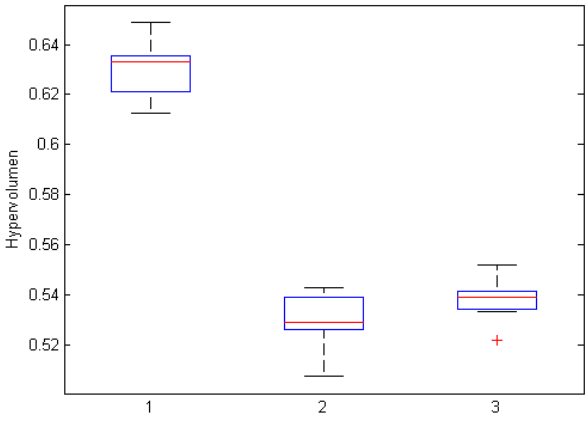


Abbildung 13: Hypervolumen
Template = 100 gute Elemente
1: SIBEA
2: *gut* wie in Kapitel 5.5.1
3: *gut* wie in Kapitel 5.5.4

6 Zusammenfassung

Es wurden mehrere Möglichkeiten untersucht, den Entscheidungsraum in die Optimierung miteinzubeziehen.

6.1 Diversität

Es ist gelungen, die Diversität in der Lösung hoch zu halten.

Die beiden gewählten Ansätze führten jedoch zu unterschiedlichen Resultaten:

- Der erste Ansatz (Kapitel 2) arbeitet mit einer Kombination von je einer Fitnessfunktion im Entscheidungsraum und Zielfunktionsraum. Das ermöglicht es, mit Hilfe der Gewichtungsfaktoren das Resultat zu beeinflussen. Die Diversität kann dadurch maximiert werden, die Qualität der Lösungen ist jedoch schlechter als mit herkömmlichen Algorithmen.
- Der zweite Ansatz (Kapitel 3) greift auf das Prinzip des gewichteten Hypervolumens zurück, was zwar rechenintensiver ist, jedoch auch gute Resultate bringt: die Diversität ist sehr hoch bei nur geringfügig kleinerem Hypervolumen als beim Referenzalgorithmus.

6.2 Templates

Das gewichtete Hypervolumen eignet sich auch für die Berechnung von Lösungen, die einer gegebenen Struktur möglichst nahe kommen und trotzdem gute Zielfunktionswerte aufweisen.

7 weiterführende Arbeiten

Die Arbeit kann mit folgenden Ansätzen weitergefürt oder genauer untersucht werden:

- mehr Experimente mit verschiedenen Populationsgrössen oder Generationen
- mehr Experimente mit Templates: andere Strukturen (z.B. Resultat eines anderen Selektors, alles '1', ...)
- andere Gewichtung. Zum Beispiel höhere *Strafe* bei Nichterfüllen gewisser Strukturen bei Templates oder ein nichtlinearer Verlauf der Gewichtung der Diversität.
- Experimente mit anderen Problemen als dem (einfachen) Rucksackproblem
- andere Eigenschaften der Struktur untersuchen (dazu muss der Entscheidungsraum bekannt sein)
- Ein weiterer Ansatz zum gewichteten Hypervolumen: teile das Problem wie ein $dim + 1$-Problem in Fronten auf und schäle ab (analog zu SIBEA, Kapitel 5.2), anstatt bei dominierten Individuen nur nach der Gewichtung zu eliminieren.
- untersuchen, weshalb das Hypervolumen bei den Experimenten mit vorgegebener und sehr unterschiedlicher Struktur annähernd konstant ist.

A Ändern des Variators

Der Variator ist für das Generieren von neuen Individuen zuständig.

Zum einen muss das PISA-Protokoll [9] verändert werden, damit der Selektor auch auf die Daten des Entscheidungsraums zugreifen kann. Zum anderen kann man die Testläufe automatisieren, indem eine Simulation erneut - diesmal mit einem anderen Seed[37] - und automatisch gestartet werden kann.

Dafür muss der Variator verändert werden.

A.1 Automatisierung

Der Variator ist so verändert worden, dass Läufe mit mehreren Seeds automatisch hintereinander gestartet werden.

A.2 Entscheidungsraum

Die Informationen werden analog der *PISA_var-Datei* in die *PISA_dcs*[38]-*Datei* geschrieben.

Um für die Auswertung die Daten des Entscheidungsraum ebenfalls zur Verfügung zu haben, ist die Funktion $write\_output\_file()$ ebenfalls verändert worden: nach den Daten der Zielfunktionen werden gleich noch die Daten des Entscheidungsraums angehängt. Es gibt nur eine Datei für die Auswertung, in dem ein Individuum nur eine Zeile benötigt.

Damit Aussagen über die Geschwindigkeit der Konvergenz der untersuchten Algorithmen getroffen werden kann, wird für jede Generation ein Outputfile geschrieben.[39]

[37] anders als der Selektor ist der Variator nicht deterministisch

[38] *dcs* für Decision Space = Entscheidungsraum

[39] Diese Daten sind nicht Bestandteil der Arbeit, finden sich jedoch auf dem beiliegenden Datenträger, inklusive einiger Plots der verschiedenen Algorithmen.

A.2.1 Initialpopulation

Die Initialpopulation für sämtliche Experimente ist in Tabelle 1 aufgelistet. Die Spalten *Gut, Top 20* und *Top 60* sind dabei nur für die Experimente mit Templates, ohne dem Experiment von Kapitel 5.5.4 relevant.

Element	Profit 1	Gewicht 1	Profit 2	Gewicht 2	Gut	Top 20	Top 60
1	85	48	26	22	1	0	1
2	81	45	77	51	1	0	1
3	68	38	36	66	0	0	0
4	81	94	14	57	0	1	1
5	46	86	65	82	0	0	1
6	82	39	88	89	0	0	1
7	18	60	38	29	1	0	1
8	97	72	67	32	1	0	1
9	87	37	99	20	1	1	1
10	39	95	32	74	0	1	1
11	51	56	44	54	0	0	0
12	72	98	60	27	1	0	0
13	59	85	37	42	0	0	0
14	66	44	14	65	0	0	1
15	74	89	45	83	0	0	1
16	54	90	37	26	1	0	0
17	33	31	27	39	0	0	0
18	44	82	52	29	1	0	0
19	69	52	13	57	0	0	1
20	58	90	96	89	1	0	0
21	63	59	44	51	0	0	0
22	61	23	39	84	0	0	0
23	28	30	25	61	0	0	1
24	90	28	12	51	0	0	0
25	59	35	39	29	1	0	1
26	21	86	28	51	0	1	1
27	47	95	33	19	1	0	0
28	11	10	28	38	0	0	0
29	88	54	93	62	1	0	1
30	73	66	18	21	0	0	0
31	41	11	60	92	0	0	1
32	20	93	71	17	1	0	0
33	42	95	70	55	1	0	0
34	20	50	32	50	0	0	1
35	30	57	88	20	1	0	1

36	28	93	39	31	1	0	1
37	60	52	87	77	1	0	0
38	27	73	87	46	1	0	0
39	85	27	67	37	1	1	1
40	87	34	92	83	1	0	1
41	68	30	63	86	0	0	0
42	36	90	53	20	1	0	0
43	10	67	65	61	1	1	1
44	100	84	68	42	1	0	0
45	62	30	63	52	1	0	1
46	18	68	82	73	1	0	1
47	27	14	17	79	0	0	1
48	48	58	65	64	1	0	0
49	68	88	80	16	1	0	1
50	100	97	38	99	0	0	1
51	14	62	51	14	1	0	0
52	77	46	54	32	1	0	1
53	75	68	95	18	1	1	1
54	36	61	26	38	0	0	1
55	34	99	39	44	0	0	1
56	60	87	80	94	0	0	0
57	85	26	97	11	1	1	1
58	11	38	43	37	1	0	1
59	65	34	34	33	1	0	1
60	63	55	37	61	0	0	0
61	38	62	69	43	1	0	0
62	61	41	88	80	1	0	0
63	71	90	96	98	0	0	0
64	21	59	79	69	1	0	1
65	81	13	61	74	0	1	1
66	50	29	95	19	1	1	1
67	97	70	91	37	1	0	1
68	50	76	69	80	0	0	0
69	83	11	95	67	1	1	1
70	63	49	14	74	0	1	1
71	17	93	46	21	1	0	1
72	95	59	68	35	1	0	1
73	77	73	81	97	0	0	0
74	71	81	31	96	0	0	1
75	75	48	47	71	0	0	0
76	50	19	52	48	1	0	1

77	10	22	46	76	0	0	1
78	13	25	37	100	0	1	1
79	38	66	54	46	1	0	0
80	23	91	34	66	0	1	1
81	87	28	68	13	1	1	1
82	88	54	69	41	1	0	1
83	51	26	61	30	1	0	1
84	84	95	60	89	0	0	0
85	87	100	62	39	1	0	0
86	43	46	71	73	0	0	0
87	17	83	43	20	1	0	1
88	92	76	80	100	0	0	0
89	62	28	81	22	1	1	1
90	13	90	14	26	0	1	1
91	32	37	47	87	0	0	1
92	21	23	98	69	1	0	0
93	80	97	64	40	1	0	0
94	85	74	53	25	1	0	1
95	22	70	11	44	0	1	1
96	82	12	43	20	1	1	1
97	85	84	80	43	1	0	0
98	13	16	38	38	0	0	0
99	80	98	98	26	1	0	1
100	77	12	94	10	1	1	1

Tabelle 1: Initialpopulation aller Experimente:
Element: Index des Elements
Profit und Gewicht: Grundlage zur Berechnung der Zielfunktion
Gut: 1: (erwartetes) Resultat besser, wenn Element drin ist, 0 wenn nicht
Top 20: wird für die besten/schlechtesten 20 berücksichtigt.
Top 60: wird für die besten/schlechtesten 60 berücksichtigt.

B Auswertung mit MATLAB

Die Auswertung der Resultate erfolgt aussschliesslich mit MATLAB, welches auf die vom Variator produzierten Daten (Kapitel 1.5.1) zugreift.

B.1 div_tot.m

div_tot.m braucht als Input alle Endpopulationen der Experimente der Diversität und dem Vergleichsexperiment und produziert als Output die beiden

Grafiken, die in Abbildung 8 dargestellt sind. Viele der Plots arbeiten mit Boxplots [10], welche die statistische Verteilung gut wiederzugeben versuchen.

B.2 hyp_vs_div.m

hyp_vs_div.m ist eine andere Form von div_tot.m. Alle Resultate werden nicht mehr in Boxplots dargestellt, sondern in der Diversität-Hypvervolumen-Ebene. So lässt sich z.B. der Einfluss der Faktoren *gc* und *gd* besser verfolgen.

B.3 tem_50ind.m

tem_50ind.m wird in Kapitel 5.5.3 verwendet.

B.4 tem_tot.m

tem_tot.m nimmt alle Endgenerationen der (regulären) Versuche mit Templates und dem Vergleichsexperiment und plottet

- für das Vergleichsexperiment den erfüllten Anteil der Templates
- für den Template-Algorithmus den erfüllten Anteil der Templates
- für das Vergleichsexperiment das Hypervolumen
- für den Template-Algorithmus die jeweiligen Hypervolumen.

Das Resultat ist in Abbildung 10 zu sehen.

B.5 div_verlauf.m

div_verlauf.m plottet den Verlauf von verschiedenen Algorithmen, wobei pro Algorithmus je ein Plot mit der Diversität und einer mit dem Hypervolumen ausgegeben wird. Es kann gewählt werden, nach wie vielen Generationen jeweils ein neuer Boxplot ausgegeben wird.

B.6 hypervolume.m

hypervolume.m stammt von Mathworks[40] und wird zur Berechnung der Hypervolumen gebraucht. Es wird eine Monte-Carlo-Approximation verwendet [11].

[40] http://www.mathworks.com/matlabcentral/fileexchange/19651-hypervolume-indicator

B.7 hypeIndicatorExact.m

hypeIndicatorExact.m stammt von Johannes Bader[41] und wird nicht für die Auswertung gebraucht, hat jedoch als Grundlage für die Implementierung des Algorithmus 6 gedient.

[41] johannes.bader@tik.ee.ethz.ch

Literatur

[1] R. J. Bauer, *Genetic algorithms and investment strategies*, M. Thompson, Ed. Karl Weber, 1994.

[2] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[3] E. Z. Institut TIK, "http://www.tik.ee.ethz.ch/sop/pisa/," Jun 2009.

[4] J. Bader and E. Zitzler, "HypE: Fast Hypervolume-Based Multiobjective Search Using Monte Carlo Sampling," Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, TIK Report 286, Oct. 2006.

[5] O. M. Shir, M. Preuss, B. Naujoks, and M. Emmerich, "Enhancing decision space diversity in evolutionary multiobjective algorithms," in *EMO '09: Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 95–109.

[6] H. Klimant, R. Piotraschke, and D. Schönfeld, *Informations- und Kodierungstheorie*. Teubner, 2006.

[7] R. Walter, *Einführung in die Analysis, Band 2*. Berlin: Walter de Gruyter, 2007.

[8] E. Zitzler, D. Brockhoff, and L. Thiele, "The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration," in *Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, ser. LNCS, S. Obayashi *et al.*, Eds., vol. 4403. Berlin: Springer, 2007, pp. 862–876.

[9] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA — a platform and programming language independent interface for search algorithms," in *Evolutionary Multi-Criterion Optimization (EMO 2003)*, ser. Lecture Notes in Computer Science, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds. Berlin: Springer, 2003, pp. 494 – 508.

[10] N. Henze, *Stochastik für Einsteiger: Eine Einführung in die faszinierende Welt des Zufalls*, 7th ed. Wiesbaden: Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH, 2008.

[11] G. S. Fishman, *Monte Carlo: Concepts, algorithms, and applications*, ser. Springer Series in Operations Research. New York: Springer-Verlag, 1996.