UNIVERSITÄT DORTMUND

Hypervolumen-basierte Selektion in einem evolutionären Algorithmus zur Mehrzieloptimierung

Nicola Beume

Algorithm Engineering Report
TR06-2-002
Mai 2006
ISSN 1864-4503

Universität Dortmund
Fachbereich Informatik
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

Diplomarbeit

Hypervolumen-basierte Selektion in einem evolutionären Algorithmus zur Mehrzieloptimierung

Nicola Beume

INTERNE BERICHTE
INTERNAL REPORTS

Diplomarbeit am
am Fachbereich Informatik
der Universität Dortmund

31. März 2006

Gutachter:
Prof. Dr. Günter Rudolph
Dipl.-Inform. Boris Naujoks

Inhaltsverzeichnis

Vorwort

Diese Diplomarbeit entstand auf ungewöhnliche Weise, da sie auf Arbeiten basiert, die in Kooperation mit zwei wissenschaftlichen Mitarbeitern im Rahmen des Sonderforschungsbereichs 531 *Design und Management komplexer technischer Prozesse und Systeme mit Methoden der Computational Intelligence* durchgeführt wurden. Hier werden die Ergebnisse erstmalig in deutscher Sprache und mit einem Schwerpunkt auf dem Eigenanteil der Diplomandin dargestellt. Die Publikationen der Arbeitsgruppe sind nachfolgend aufgelistet mit Anmerkungen, in welchen Kapiteln dieser Arbeit man die jeweiligen Ergebnisse wiederfindet. Michael Emmerich hat die Grundidee des entwickelten Algorithmus ins Leben gerufen. Die Erarbeitung des Algorithmus wurde gemeinschaftlich von allen drei Autoren durchgeführt. Die Hauptarbeit bei den Anwendungen auf aeronautische Probleme wurde von Boris Naujoks geleistet, daher wird ihre Darstellung, genau wie die Kopplung des Algorithmus mit einem Metamodell von Michael Emmerich, in dieser Arbeit ausgelassen. Die Varianten des Algorithmus wurden hauptsächlich von Nicola Beume und Boris Naujoks erarbeitet. Nicola Beume leistete die Hauptarbeit bei der Entwicklung einer Variante, die im Rahmen der studentischen Projektgruppe 447 erstmals angewendet wurde.

- M. Emmerich, N. Beume und B. Naujoks, An EMO algorithm using the hypervolume measure as selection criterion, In: C. A. Coello Coello et al. (Hrsg.): Proc. Evolutionary Multi-Criterion Optimization: 3rd Int'l Conf. (EMO 2005), Springer, Berlin, 2005, 62–76, LNCS 3410. In den Kapiteln 2 und 4.

- Projektgruppe 447, Metaheuristiken für die Mehrzieloptimierung, Endbericht, Interner Bericht der Universität Dortmund, 2005. In Kapitel 2.

- B. Naujoks, N. Beume und M. Emmerich, Multi-objective optimisation using S-metric selection: Application to three-dimensional solution spaces, In: B. McKay et al. (Hrsg.) Proc. of the 2005 Congress on Evolutionary Computation (CEC 2005), Edinburgh, IEEE Press, Piscataway NJ, 2005, Band 3, 1282-1289. In den Kapitel 2, 3 und 4.

- B. Naujoks, N. Beume und M. Emmerich, Metamodel-assisted SMS-EMOA applied to airfoil optimization tasks, Proceedings EUROGEN'05 *(CD-ROM)*, R. Schilling et al. (Hrsg.), FLM, TU München, 2005. Die Ergebnisse der Studie werden hier nicht dargestellt.

- N. Beume, B. Naujoks und M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume, erscheint in: European Journal of Operational Research. In den Kapiteln 2, 3 und 4.

Meinen Koautoren Michael Emmerich und Boris Naujoks danke ich herzlich für die produktive und erfreuliche Zusammenarbeit. Bei Günter Rudolph, Boris Naujoks und Tobias Wagner bedanke ich mich für ihre hilfreichen Anmerkungen zu dieser Arbeit. Den Kolleginnen und Kollegen des Lehrstuhls für *Algorithm Engineering und Systemanalyse* geleitet von Petra Mutzel bzw. von Hans-Paul Schwefel, sowie des Lehrstuhls für *Effiziente Algorithmen und Komplexitätstheorie* geleitet von Ingo Wegener danke ich für fachliche Ratschläge und interessante Diskussionen.

Kapitel 1

Einführung

Diese Arbeit präsentiert einen evolutionären Algorithmus zur Mehrzieloptimierung, der zur Auswahl der Folgepopulation ein neuartiges Selektionskriterium verwendet. Es wird ein reellwertiges Maß eingesetzt, mit dem die partiell geordneten Lösungsvektoren bewertet und anhand dessen sie vollständig geordnet werden. Dieses Maß ist der Beitrag einer Lösung zum Hypervolumen des von der Population dominierten Zielraumbereichs. Die Lösung, welche den geringsten Beitrag leistet, also das wenigste Hypervolumen exklusiv dominiert, wird nicht in die Folgepopulation aufgenommen. Die Funktionsweise und Eigenschaften des entwickelten *S Metric Selection – Evolutionary Multiobjective Optimisation Algorithm (SMS-EMOA)* werden beschrieben und seine Leistungsfähigkeit auf populären Benchmark-Problemen demonstriert. Es werden alternative Varianten des Selektionskriteriums betrachtet und verglichen. Die Laufzeitkomplexität des SMS-EMOA ist bestimmt durch die Berechnung des Hypervolumens, für die eine Übersicht verschiedener Algorithmen, die teilweise speziell für den SMS-EMOA entwickelt wurden, gegeben wird.

1.1 Motivation und Übersicht

Neben den unzähligen algorithmischen Entwurfsmöglichkeiten und den herausfordernden theoretischen Fragestellungen gibt es vor allem eine Motivation, sich mit evolutionären Algorithmen zu beschäftigen: Sie funktionieren. Nein, nicht immer gut und unter Umständen sogar sehr schlecht, dennoch haben sie – obwohl ihre Eigenschaften aktuell schwierig zu beweisen sind – ihre Existenzberechtigung durch erfolgreiche Anwendungen auf komplexe reale Probleme belegt. Evolutionäre Algorithmen werden auf Optimierprobleme angewendet und in dieser Arbeit speziell auf Mehrzieloptimierprobleme. Bei der Optimierung wird für ein Problem unter allen möglichen Lösungen eine beste Lösung gesucht. Das heuristische Vorgehen zur Optimierung kennt man aus dem Alltag, wenn man beispielsweise eine schnellste PKW-Route zur Arbeitsstelle sucht. Allein durch Betrachtung von Stadtplänen wird man diese nicht bestimmen können, weil die tatsächlichen realen Bedingungen wie Verkehrsaufkommen und Geschwindigkeitsbegrenzungen vor Ort erkundet werden müssen. Man wird also einige Streckenvarianten ausprobieren, dabei die Fahrzeit messen und sich dann für die schnellste Strecke entscheiden. Bei der Mehrzieloptimierung betrachtet man mehre-

re Kriterien gleichzeitig, zusätzlich zur Fahrtdauer beispielsweise die Staugefahr oder die Schönheit der Umgebung. Es gibt eine Fülle zum Teil Jahrzehnte alter Verfahren zur (Mehrziel-)Optimierung, die wir hier nicht aufführen wollen, sondern auf die Literatur verweisen (siehe z. B. [BSMM00, Deb01]). Evolutionäre Algorithmen zeichnen sich dadurch aus, dass sie sowohl schnelle als auch robuste Optimierer sind, d. h. auch bei schwierigen Bedingungen funktionieren. Sie werden vor allem als Black-Box-Optimierer eingesetzt zur Optimierung hoch komplexer Probleme, deren Struktur nicht vollständig verstanden wurde, sodass Funktionsauswertungen oftmals nur durch Simulationen gewonnen werden können. Die Erforschung moderner evolutionärer Algorithmen ist aktuell nur empirisch möglich, exakte Analysen konnten bisher nur für sehr spezielle Betrachtungen durchgeführt werden. Diese Arbeit wählt ebenfalls den empirischen Zugang, um Erkenntnisse über das Verhalten des neu entwickelten Algorithmus zu gewinnen.

Dieses erste Kapitel bietet eine grundlegende Einführung in die wichtigen Themengebiete der Diplomarbeit, nämlich die Funktionsweise evolutionärer Algorithmen, Konzepte der Mehrzieloptimierung, die Vergleichsmöglichkeiten von Algorithmen und Qualitätsmaße zur Bewertung von Ergebnismengen der Mehrzieloptimierung. Kapitel 2 stellt den entwickelten Algorithmus SMS-EMOA mit einigen alternativen Varianten vor. Seine Eigenschaften werden im Vergleich mit anderen Verfahren beschrieben. Das anschließende Kapitel 3 gibt eine Übersicht der aktuell besten Algorithmen zur Berechnung des Hypervolumens, wie es im Selektionsoperator des SMS-EMOA benötigt wird. In Kapitel 4 werden die Ergebnisse empirischer Studien des SMS-EMOA auf akademischen Testfunktionen präsentiert. Schließlich bietet Kapitel 5 eine Zusammenfassung der Arbeit und weitergehende Fragestellungen.

Im Folgenden werden für die beschriebenen Zusammenhänge stets deutsche Begriffe verwendet und die üblichen englischen Fachbegriffe in Klammern hinzugefügt.

1.2 Evolutionäre Algorithmen

Grundlegende Idee

Evolutionäre Algorithmen (evolutionary algorithms, EA) sind randomisierte Suchheuristiken, die zur Optimierung komplexer Probleme eingesetzt werden. Die Verfahren wurden inspiriert durch die darwinistische Vorstellung der natürlichen Evolution als eines Verbesserungsprozesses durch Reproduktion, Variation und Auslese. Entwickelt in den 60er Jahren des vorigen Jahrhunderts wird die biologische Terminologie noch heute verwendet. Übertragen auf die Algorithmik betrachtet man einen syntaktisch definierten Suchraum und das Problem, eine optimale Lösung zu finden, wird zu einem Suchproblem im Raum der möglichen Lösungen. Die Idee ist, ausgehend von beliebigen Lösungen, durch leichte, probabilistisch gesteuerte Variationen, schrittweise zu Verbesserungen zu gelangen.

Als *Heuristiken (heuristics)* bezeichnet man in diesem Zusammenhang Verfahren, die zur Problemlösung eine durchdachte Strategie verfolgen, die aber nicht mit Sicherheit

zu einem guten Erfolg führt (vgl. [Hei99]). Entsprechend kann auch für evolutionäre Algorithmen keine bestimmte Qualität der erzeugten Lösungen garantiert werden. Da man EA zumeist auf sehr komplexe Problemstellungen anwendet, kann selbst in dem Idealfall einer gefundenen optimalen Lösung, diese oftmals nicht als optimal nachgewiesen werden.

Begriffsbestimmung

Als Optimierproblem ist eine Funktion $g : A \rightarrow B$ gegeben, die Elemente eines Problemraums A in einen Bildraum B abbildet, der zumindest partiell geordnet sein muss. Mit Hilfe eines EA wird ein Suchpunkt bzw. eine Lösung $\mathbf{s} \in A$ mit einem möglichst guten Funktionswert $g(\mathbf{s})$ gesucht. Dazu ist eine Codierung der Lösungen festzulegen, die *Repräsentation (representation)* genannt wird. Den Raum A bezeichnet man auch als den Phänotypraum, da die Lösungen in der ursprünglichen Formulierung des Problems erscheinen. Der Genotypraum $G = G_1 \times \ldots \times G_n$ ist der Raum der möglichen Lösungen, auch *Genome (genome)* genannt, auf dem der EA arbeitet. Genotyp- und Phänotypraum können gleich sein, oder bei Phänotypräumen, die aus algorithmischer Sicht unhandlich sind, wählt man einen der üblichen Genotypräume und bildet die Lösung anschließend in den Phänotypraum ab. Diese Genotyp-Phänotyp-Abbildung (genotype-phenotype-mapping) $h_1 : G \rightarrow A$ kann oftmals nicht bijektiv gestaltet werden, sollte aber einen möglichst großen Bereich von A abdecken und dabei wenige Elemente mit gleichem Abbild haben. Eine Abbildung $h_2 : B \rightarrow \mathbb{R}$ ermöglicht eine Bewertung der Phänotypbilder durch reelle Zahlen. Ein Genom ist ein Vektor $\mathbf{x} = (x_1, \ldots, x_n)$ mit $\mathbf{x} \in G$, $x_i \in G_i$ aus n sogenannten *Objektparametern (object parameters)*. Zur kontinuierlichen Optimierung werden die G_i als Intervalle reeller Zahlen gewählt, zur diskreten Optimierung als Intervalle ganzer Zahlen. In der kombinatorischen Optimierung werden binäre Suchräume oder Permutationen verwendet. Es treten auch Mischformen auf, wie beispielsweise die gemischt-ganzzahlige (mixed integer) Repräsentation, die teilweise diskrete und teilweise reellwertige Räume G_i beinhaltet.

Für ein Genom wird eine *Zielfunktion* $f : G \rightarrow \mathbb{R}$ als $f = h_2 \circ g \circ h_1$ ausgewertet. Ein Genom und dessen Zielfunktionswert werden zu einer Einheit zusamengefasst und als *Individuum (individual)* bezeichnet. Zusätzlich kann ein Individuum sogenannte Strategieparameter (strategy parameter) enthalten, die die Variation steuern. Innerhalb eines EA sind die Individuen die zu optimierenden Einheiten. Sie werden mit einer sogenannten *Fitnessfunktion (fitness function)* bewertet. Die Fitnessfunktion ist oftmals identisch zur Zielfunktion, weshalb die Begriffe mitunter synonym verwendet werden. Wir unterscheiden hier explizit zwischen der zu optimierenden Zielfunktion und der Fitness, die nur innerhalb des EA von Interesse ist, um Individuen zu bewerteten und zu selektieren. Diese Unterscheidung ist in der Mehrzieloptimierung besonders detlich, da sich dabei ein Vektor von Zielfunktionswerten ergibt, der durch eine zusätzliche Bewertungsfunktion auf einen reellwertigen Fitnesswert abgebildet wird. Ein Individuum $\mathbf{a} = (\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z})$ besteht also insgesamt aus einem Tupel der Vektoren der Strategieparameter $\mathbf{w}$, des Genoms $\mathbf{x}$, der Zielfunktionswerte $\mathbf{y}$ und der Fitness $\mathbf{z}$. Eine Menge P von Individuen nennt man eine *Population (population)*. Die Population der Größe μ in der Generation t wird bezeichnet als $P(t) = \{\mathbf{a}(t)^{(1)}, \ldots, \mathbf{a}(t)^{(\mu)}\}$.

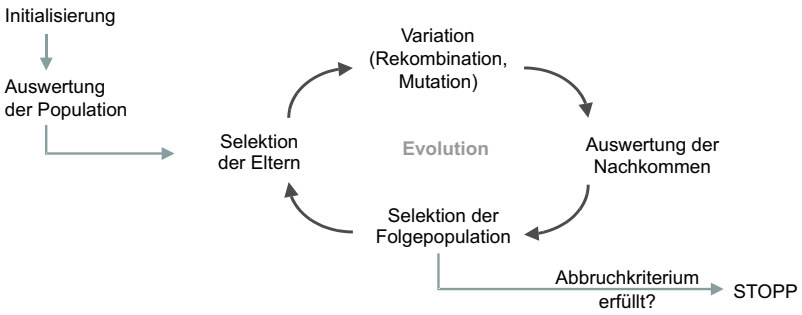

Abbildung 1.1: Schematische Darstellung der Operatoren eines EA.

Beispiel 1.1 (TSP) *Mit einem EA könnte man das berühmte Traveling Salesperson Problem (TSP), eine optimale Rundreise zwischen n Städten zu finden, optimieren. Als Repräsentation kann man n-stellige Permutationen (Genotypraum) verwenden, die die Reihenfolge der Städte darstellen. Der EA versucht nun nicht, eine gute Lösung anhand problemspezifischer Informationen zu konstruieren, sondern beginnt mit irgendeiner Rundreise und „probiert" einfach durch Veränderung der Reihenfolge der Städte (Variation) weitere Lösungen aus. Die Zielfunktion wäre die Minimierung der Summe der Rundreisekosten und die Fitnessfunktion kann ebenso gewählt werden. Formuliert als Mehrzieloptimierproblem könnte man versuchen, gleichzeitig eine schnellste und billigste Rundreise zu finden. Die Zielfunktionen wären die Reisedauer und -kosten und als Fitness müsste eine zusätzliche Bewertungsfunktion dienen, die zwischen beiden Wünschen abwägt.*

Ablauf eines evolutionären Algorithmus

Der Optimierprozess eines EA verläuft in Runden, die als *Generationen (generations)* bezeichnet werden. In jeder Generation werden aus der aktuellen Population Elter-Individuen ausgewählt, aus denen neue *Nachkommen (offspring)* erzeugt werden. Deren Fitnesswert wird bestimmt und anschließend wird eine vielversprechende Auswahl an Individuen zur Folgepopulation ernannt. Die Abfolge der sogenannten Operatoren, aus denen ein EA besteht, ist in Abbildung 1.1 veranschaulicht.

Es gibt unzählige Spezifikationen der Operatoren und deren Entwurf ist ein aktuelles Forschungsgebiet, sodass eine vollständige Wiedergabe unmöglich erscheint. Zum Verständnis wird hier nur ein Basis-Algorithmus beschrieben und ein kurzer Überblick der populärsten etablierten Varianten der Operatoren gegeben werden. Die Darstellung ist dabei so allgemein gehalten, dass sie auf die Einziel- wie auf die Mehrzieloptimierung zutrifft. Spezielle Verfahren, die in evolutionären Mehrzieloptimieralgorithmen eingesetzt werden, sind im folgenden Abschnitt 1.3 beschrieben.

EA arbeiten mit – möglicherweise sehr vielen – Parametern, von denen hier zunächst nur die zwei üblichsten beschrieben werden. Es ist in der Regel eine Populationsgröße $\mu = |P|$ festzulegen, die bestimmt, wie viele Individuen die Population P einer Generation enthalten soll. Pro Generation werden zusätzlich λ Individuen als Nachkommen erzeugt.

Initialisierung Der Algorithmus startet mit der Initialisierung der Population. Diese erfolgt typischerweise zufällig gleichverteilt im Suchraum oder durch Initialisierung von Individuen mit bekannten Lösungen, die es zu verbessern gilt.

Selektion der Eltern Jede Generation beginnt mit der Auswahl von Elter-Individuen für die Reproduktion. Populäre probabilistische Strategien arbeiten dabei fitnessabhängig oder zufällig gleichverteilt. Alternativ kann man deterministisch die besten Individuen wählen.

Variation Aus den Eltern werden durch Variation neue Genome (und eventuell Strategieparameter) für die Nachkommen generiert. Die Erzeugung eines Nachkommen aus mehreren Eltern heißt *Rekombination (recombination, crossover)*, die Variation nur eines Elter wird als *Mutation (mutation)* bezeichnet. Die Spezifizierung der Variationsoperatoren ist stark abhängig von der verwendeten Repräsentation. Die Operatoren dienen der Erforschung des Suchraums und sollten in ihrer grundlegenden Form ungerichtet sein, d. h. keine Richtung im Suchraum bevorzugen.

Rekombination Zur Rekombination werden pro Nachkomme typischerweise zwei Eltern ausgewählt, es gibt aber auch Verfahren bei denen die gesamte Population mit einbezogen wird. Die Werte eines Nachkommen werden entweder durch gewichtete Mittelung (arithmetische Rekombination) der Elternwerte gebildet oder es werden Vektorkomponenten jeweils eines Elter unverändert übernommen (diskrete Rekombination).

Mutation Die Mutation ist eine zufällige leichte Veränderung eines Individuums. Falls eine Rekombination durchgeführt wird, wird sie auf den entstandenen Nachkommen angewendet, ansonsten entsteht durch sie ein Nachkomme als Mutation eines kopierten Elter-Individuums. Bei binären Suchräumen werden beispielsweise Bits invertiert oder bei reellen Genotypräumen normalverteilte Zufallszahlen addiert. Die Strategieparameter dienen bei der reellwertigen Optimierung dazu, die Schrittweiten, also die Größe der Variation für jedes Individuum individuell zu steuern.

Auswertung der Nachkommen Für die λ Nachkommen wird ihr Zielfunktionswert ausgewertet und sie werden durch die Fitnessfunktion bewertet. Diese Auswertung kann sehr zeitaufwändig sein, da es sich um komplexe Probleme handeln kann, für die keine Beschreibung in geschlossener mathematischer Form vorliegt und Funktionswerte von Suchpunkte nur durch Simulationen gewonnen werden können.

Selektion der Folgepopulation Anschließend wird die Folgepopulation der Größe μ ausgewählt, so dass die Populationsgröße bei diesem Basis-Algorithmus konstant bleibt. Werden dazu alle Individuen – die bisherige Population und die Nachkommen – in Betracht gezogen, nennt man das eine Plus-Strategie und schreibt $(\mu + \lambda)$-Strategie. Wählt man nur Nachkommen aus, was natürlich $\mu \leq \lambda$ voraussetzt, nennt man das ein Komma-Strategie, die man als (μ, λ)-Strategie.

Abbruchkriterium Da das Erreichen eines Optimums in der Regel weder garantiert noch verifiziert werden kann, gibt es keinen konzeptionellen Endpunkt des evo-

lutionären Prozesses. Es wird daher ein Abbruchkriterium festgelegt, dass sich typischerweise auf eine Beschränkung von Ressourcen (Anzahl der Generationen, CPU-Zeit) oder auf die Qualität der Lösungen („Erzielten die letzteb Generationen noch eine entscheidende Verbesserung?“) bezieht. Die Bedingung wird zwischen den Generationen überprüft und bei Erfüllung der Optimierprozess gestoppt.

Der evolutionäre Prozess hat das geschickte Absuchen des Suchraums zum Ziel. Dabei sollen die interessante Regionen des Suchraums gefunden und vielversprechende Gebiete verstärkt untersucht werden, um die jeweiligen lokalen Optima zu finden. Es gibt hierbei einen Konflikt zwischen globaler Erforschung des Suchraums und der Konvergenz zu einem lokalen Optimum. Um ein lokales Optimum zu erreichen, steuern zumeist mehrere Individuen darauf zu. Dies kann zu vielen ähnlichen Individuen und damit zu einem Diversitätsverlust der Population führen. Die Konzeption vieler EA beinhaltet, dass besonders gute Individuen ihre Eigenschaften verstärkt vererben und somit die Population eventuell „vereinheitlichen“. Ein Diversitätsverlust kann zur Stagnation des Optimierprozesses führen, insbesondere weil das Verlassen eines lokalen Optimums nicht mehr möglich ist. Wenn alle Individuen sehr ähnlich sind, wird mit hoher Wahrscheinlichkeit in den folgenden Generationen nur noch ein kleiner Teil des Suchraums erreicht. Ist die Population zu einem lokalen Optimum konvergiert, so werden typischerweise nur noch schlechtere Individuen erzeugt und kein globales Optimum gefunden.

Historie evolutionärer Algorithmen

Evolutionäre Algorithmen (EA) ist heute ein Oberbegriff, unter dem sich mehrere Verfahren vereinen. Die Ursprünge sind einerseits die von Hans-Paul Schwefel [Sch65, Sch68] und Ingo Rechenberg in den 60er Jahren des 20. Jahrhunderts entwickelten Evolutionsstrategien [Rec94]. Andererseits wurden zeitnah und unabhängig davon in den USA von John Holland [Hol75] genetische Algorithmen entwickelt. Weitere Ansätze sind das evolutionäre Programmieren (EP) von Lawrence Fogel [FOW66] und die genetische Programmierung (GP) von John Koza [Koz92]. Die Evolutionsstrategien verwendeten eine problemspezifische (oftmals reellwertige) Repräsentation, wohingegen die ersten genetischen Algorithmen stets einen binäre Codierung des Suchraums verwendeten. Das erfordert eine Abbildung des Problemraums (Phänotypraums) in den binären Genotypraum. Bei den Evolutionsstrategien galt die Mutation als zentraler Operator, bei den genetischen Algorithmen wurde die Rekombination als wichtiger angesehen und man formulierte die sogenannte „building block“-Hypothese, nach der sich gute Lösungen aus guten Bestandteilen zusammensetzen. Es gibt weitere Unterschiede, allerdings sind diese im Laufe der Jahre verwischt. Die einzelnen Schulen haben sich gegenseitig inspiriert und aktuelle Forschungsgemeinschaften sind von allen Arbeiten beeinflusst. Moderne evolutionäre Algorithmen lassen sich nicht mehr eindeutig als Evolutionsstrategien oder genetische Algorithmen bezeichnen, so dass es sinnvoll erscheint nur noch den Begriff der evolutionären Algorithmen zu verwenden. Alternativ wird auch der Begriff *Evolutionary Computation (EC)* synonym gebracuht.

In den 60er und 70er Jahren wurden die Verfahren in der breiten wissenschaftlichen Öffentlichkeit eher belächelt und die Verwendung von Zufallsentscheidungen als man-

gelnde Kompetenz der Algorithmendesigner interpretiert. Das änderte sich durch die aufkommende verstärkte Berechnungskraft der Computer in den 80er Jahren, die die experimentelle Optimierung erst wirklich nützlich machte. Seit Mitte der 80er Jahre erleben evolutionäre Algorithmen wachsendes Interesse und einen extremen Aufschwung in den 90er Jahren. Sie werden verstärkt auf reale Problemstellungen angewendet und sind wissenschaftlich anerkannt, so dass sie von einer immer größer werdenden Gemeinschaft erforscht und weiterentwickelt werden. Sie sind heute etabliert als Teilgebiet der hochmodernen *Computational Intelligence (CI)*, der außerdem die Fuzzy Logic und neuronale Netze zugeordnet werden, sowie neuerdings die Verfahren der Schwarm-Intelligenz und der künstlichen Immunnetzwerke. Den CI-Methoden ist gemeinsam, dass sie natur-inspiriert sind, also natürliche Prozesse nicht unbedingt exakt nachahmen, aber sie zum Vorbild nehmen bei der Entwicklung technischer Verfahren.

1.3 Mehrzieloptimierung

Ideen und Modellierungsansätze

Bei alltäglichen Problemen aller Lebensbereiche stehen oftmals Lösungsmöglichkeiten zur Auswahl, die Vor- und Nachteile aufweisen und Wünsche mit unterschiedlicher Intensität erfüllen, was dazu führt, dass keine Lösung eindeutig als die beste angesehen werden kann. Es gibt also verschiedene Anforderungen an die Qualität einer Lösung und die Güte einer Lösung hängt davon ab, wie wichtig einzelne Kriterien für den Entscheidungsfinder sind. Objektiv betrachtet sind zwei Lösungen unvergleichbar, wenn eine Lösung ein Kriterium besser und eines schlechter erfüllt als die andere.

Diese Sichtweise möchten wir nun als *Mehrzieloptimierung (multi-objective optimisation)* formalisieren. Optimiert werden $d \geq 2$ *Zielfunktionen (objectives)* $f_1, ..., f_d$ mit $f_i : S \to \mathbb{R}$ ($i = 1, ..., d$). Aufgrund der Dualität von Minimierungs- und Maximierungsproblemen[1] können wir uns im Folgenden auf die Betrachtung von Minimierungsproblemen beschränken. Die Zielfunktionswerte im *Zielfunktionsraum (objective space)* bilden einen d-dimensionalen Vektor $\mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_d(\mathbf{x}))^T$. Die Ziele sind typischerweise konfliktär und können daher nicht alle gleichzeitig optimal erfüllt werden. Wenn also eine Lösung ein bestimmtes Kriterium bestmöglich erfüllt, gilt das für die restlichen Zielfunktionen typischerweise nicht. Eine Lösung eines Mehrzieloptimierproblems bezeichnet man auch als *Kompromisslösung (compromise solution).*

Bei der Problemmodellierung gibt es zwei gegensätzliche Konzepte, nämlich die a priori Verfahren und den a posteriori Ansatz. Die a priori Verfahren legen im Voraus der Optimierung eine Funktion fest, die eine Kombination der d Zielfunktionen darstellt, sodass daraufhin ein monokriterielles Problem optimiert wird. Diese Vorgehensweise setzt allerdings voraus, dass das Problem und die Beziehungen der Zielfunktionen gut verstanden wurden. Außerdem muss der Entscheidungsfinder seine Präferen-

[1] Es gilt: $\min(f_i) = -\max(-f_i)$.

zen der Zielfunktionen genau kennen. Nur unter diesen Umständen kann die monokriterielle Ersatzfunktion adäquat konstruiert werden und gute Lösungen des ursprünglichen d-kriteriellen Problems erzeugen. Führt man wiederholte Läufe mit verschiedenen Gewichtungen der Zielfunktionen durch, lässt sich auf diese Weise auch eine Lösungsmenge des mehrkriteriellen Problems erzeugen. Die Vorlieben des Entscheidungsfinders können also auch variabel gehalten werden, dennoch gilt die Voraussetzung, dass die Problemstruktur zur erfolgreichen Optimierung bekannt sein muss.

Bei dem a posteriori Ansatz wird tatsächlich ein mehrkriterielles Problem optimiert und die Funktionswerte als Vektoren belassen. Der Name des Konzepts ergibt sich, weil der Entscheidungsfinder keine Präferenzen festlegen muss, sondern erst nach dem Optimierungsprozess eine zu realisierende Lösung nach seinen Vorlieben auswählt. Die Wichtigkeit von Zielfunktionen wird also erst im Nachhinein vom Entscheidungsfinder festgelegt, nachdem eine Lösungsmenge vorliegt und deutlich ist, welche Ziele in welcher Form erfüllt werden konnten und welche Kompromisse möglich sind. Wir betrachten im Folgenden also die sogenannte *Vektoroptimierung (vector optimisation)* [Mie01] oder *Pareto-Optimerung (Pareto optimisation)* [Deb01, CVL02].

Konzepte der Pareto-Optimierung

Um die Zielfunktionsvektoren von Individuen zu einander in Beziehung zu setzen, werden partielle Ordnungsrelationen verwendet, deren grundlegende Definitionen in Anhang A erläutert sind.

Definition 1.1 ((schwache/ starke) komponentenweise Ordnungsrelation) *Seien* $\mathbf{v}$ *und* $\mathbf{v}'$ d*-dimensionale Vektoren. Die schwache komponentenweise Ordnungsrelation bedeutet* $\mathbf{v} \leq \mathbf{v}' \Leftrightarrow \forall i \in d : \mathbf{v}_i \leq \mathbf{v}'_i$ *und ist eine partielle Ordnung. Die (starke) komponentenweise Ordnungsrelation bedeutet* $\mathbf{v} < \mathbf{v}' \Leftrightarrow \mathbf{v} \leq \mathbf{v}' \wedge \mathbf{v} \neq \mathbf{v}'$ *und bildet eine strenge partielle Ordnung.*

Auf die übliche partielle Ordnung von Vektoren wird eine strikte partielle Ordnung aufgesetzt, bei der ein Element nicht mit sich selbst in Beziehung steht. Über die Ordnungsrelationen von Funktionswertvektoren werden sogenannte Dominanzrelationen definiert, deren Beziehung auch auf die Urbildern, also die zugehörigen Vektoren der Objektparameter übertragen wird.

Definition 1.2 (Dominanz (dominance)) *Für zwei* n*-dimensionale Vektoren von Objektparametern* $\mathbf{x}$ *und* $\mathbf{x}'$ *und deren* d*-dimensionale Zielfunktionsvektoren* $\mathbf{y} = f(\mathbf{x})$ *bzw.* $\mathbf{y}' = f(\mathbf{x}')$ *gelten folgende Beziehungen*[2]*:*

$$\begin{aligned}
\mathbf{y} < \mathbf{y}' \quad &\Leftrightarrow \quad \mathbf{y} \prec \mathbf{y}' && [\mathbf{y} \textit{ dominiert } \mathbf{y}'] \\
&\qquad\quad \mathbf{x} \prec \mathbf{x}' && [\mathbf{x} \textit{ dominiert } \mathbf{x}'] && (1.1) \\
\mathbf{y} \leq \mathbf{y}' \quad &\Leftrightarrow \quad \mathbf{y} \preceq \mathbf{y}' && [\mathbf{y} \textit{ dominiert } \mathbf{y}' \textit{ schwach}] \\
&\qquad\quad \mathbf{x} \preceq \mathbf{x}' && [\mathbf{x} \textit{ dominiert } \mathbf{x}' \textit{ schwach}] && (1.2) \\
\mathbf{y} \not\leq \mathbf{y}' \wedge \mathbf{y}' \not\leq \mathbf{y} \quad &\Leftrightarrow \quad \mathbf{y} \parallel \mathbf{y}' && [\mathbf{y} \textit{ und } \mathbf{y}' \textit{ unvergleichbar}] \\
&\qquad\quad \mathbf{x} \parallel \mathbf{x}' && [\mathbf{x} \textit{ und } \mathbf{x}' \textit{ unvergleichbar}] && (1.3)
\end{aligned}$$

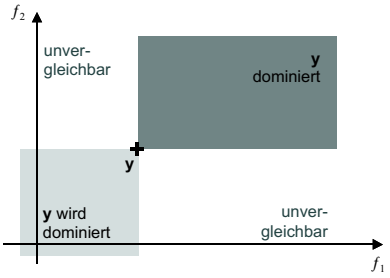

Abbildung 1.2: Veranschaulichung der Dominanzrelation (links) und der nicht-dominierten Sortierung (rechts). Dargestellt ist eine Punktmenge aus drei Fronten, deren Elemente durch verschiedene Grautöne der Punkte markiert sind.

Definition 1.3 (Pareto-Optimum, Pareto-Menge, Pareto-Front) *Eine Lösung* $\mathbf{x} \in S$ *heißt Pareto-optimal, wenn im Suchraum keine Lösung* $\mathbf{x}' \in S$ *existiert, von der* $\mathbf{x}$ *dominiert wird. Für den zugehörige Zielfunktionswertvektor* $\mathbf{y} = f(\mathbf{x})$ *gilt also:* $\nexists \mathbf{x}' \in S : \mathbf{y}' < \mathbf{y}$*; und* $\mathbf{y}$ *wird ebenfalls Pareto-optimal oder effizient genannt. Als Pareto-Front PF bezeichnet man die Menge Pareto-optimaler Zielfunktionsvektoren, die Menge ihrer Urbilder wird Pareto-Menge* $PM = \{\mathbf{x} \mid \mathbf{y} \in PF\}$ *genannt.*

Von einem Pareto-optimalen Zielfunktionsvektor kann also keine Vektorkomponente verbessert werden ohne eine andere zu verschlechtern. Die global minimalen Elemente (vgl. Anhang A) der Dominanzrelation werden als *Pareto-optimal* und die minimalen innerhalb einer Menge als *nicht-dominiert* bezeichnet.

Definition 1.4 (Nicht-Dominanz (non-dominance)) *Eine nicht-dominierte Lösung* $\mathbf{x} \in M \subseteq S$ *ist eine Lösung, die von keiner anderen Lösung in* $\mathbf{x}' \in M$ *dominiert wird. Die nicht-dominierte Menge* $ND(M) = \{\mathbf{x} \in M \mid \nexists \mathbf{x}' \in M \text{ mit } \mathbf{x}' \prec \mathbf{x}\}$ *enthält alle nicht-dominierten Lösungen von M. Die Elemente von* $ND(M)$ *sind unvergleichbar oder dominieren einander schwach. Sei* $F = \{f(\mathbf{x}) \mid \mathbf{x} \in M\}$ *die Menge der Zielfunktionsvektoren, dann bezeichnet man die nicht-dominierten Zielfunktionsvektoren* $ND(F) = \{f(\mathbf{x}) \mid \mathbf{x} \in ND(M)\}$ *als nicht-dominierte Front.*

In der evolutionären Mehrzieloptimierung ist das Konzept der nicht-dominierten Sortierung populär. Es entspricht der Partitionierung einer Menge in Antiketten. Dilworth's Dekompositionstheorem [Dil50] besagt, dass eine Menge der Höhe h in h Antiketten unterteilt werden kann (vgl. Anhang A). Eine algorithmische Beschreibungen der Partitionierung wurde von Goldberg [Gol89] formuliert. Eine effiziente Implementierung von Deb et al. [DAPM00] wird in Kapitel 2 erläutert und in Anhang B vollständig wiedergegeben.

Definition 1.5 (Nicht-dominierte Sortierung (non-dominated sorting)) *Die nicht-dominierte Sortierung ist eine hierarchische Partitionierung einer Lösungsmenge* F *in nicht-dominierte Mengen* $F_1, ..., F_v$*, mit folgenden Eigenschaften.*

1. $F_1 = ND(F)$.

[2] Die schwache Dominanz wird auch als Überdeckung (covering) bezeichnet.

2. $\forall i$ *mit* $2 \leq i \leq v$: $F_i = ND(F \setminus \bigcup_{1 \leq j < i} F_j)$.

3. $ND(F_v) \setminus F_v = \emptyset$.

Die Partitionen F_i werden als Fronten bezeichnet.

Die erste Front entspricht der nicht-dominierten Front. Eine Zielfunktionsvektor $\mathbf{y}$ gehört zur i-ten Front, wenn die Front F_{i-1} einen Vektor $\mathbf{y}'$ enthält, durch den $\mathbf{y}$ dominiert wird. Beispielsweise bedeutet dies, ein Zielfunktionsvektor gehört zur dritten Front, wenn er von einem Vektor (der zweiten Front) dominiert wird, die wiederum selbst dominiert wird von einem nicht-dominierten Zielfunktionsvektor (einem Element der ersten Front). Die Nummer der Front gibt also in gewisser Weise den Grad der Dominanz an. Die Zeilfunktionsvektoren der letzten Front dominieren keine anderen Zielfunktonsvektoren von M, was allerdings nicht als definierende Eigenschaft zu verstehen ist. Die Partitionen bilden eine Dominanz-Hierarchie, da jedes Element in F_i von mindestens einem Element in F_{i-1} dominiert wird. Eine Front mit niedrigerem Index ist also „besser “ als eine Front mit höherem Index, wobei die vergleichende Bewertung von Mengen im folgenden Kapitel 1.4 näher spezifiziert wird.

Zielsetzung der Pareto-Optimierung

Zitzler [Zit99] formuliert drei Ziele, die bei der Mehrzieloptimierung angestrebt werden sollten:

- die Distanz zur Pareto-Front sollte minimiert werden (Konvergenz),
- die gefundenen Lösungen sollten möglichst den gesamten Bereich der Pareto-Front abdecken (Diversität),
- es sollten möglichst viele nicht-dominierte Lösungen gefunden werden, um dem Entscheidungsfinder eine große Auswahl an Kompromisslösungen zu bieten, aus denen er nach seinen Präferenzen auswählen kann.

Die Leistungsfähigkeit von Mehrzieloptimier-Algorithmen bestimmt sich also durch die Qualität und Quantität ihrer produzierten Ergebnismengen und die Zeit- und Speicherplatz-Ressourcen, die sie zu deren Erzeugung benötigen.

Bei der Modellierung wird angestrebt, die Anzahl der Zielfunktionen und damit die Dimension des Suchraums möglichst gering zu halten. Bisherige Forschungsarbeiten der heuristischen Mehrzieloptimierung betrachten meistens zwei oder drei Zielfunktionen.

1.4 Qualitätsmaße zur Bewertung von Mengen

Idee und Verwendung

Das Ergebnis eines Mehrzieloptimierprozess ist bei der Pareto-Optimierung üblicherweise eine Menge von Kompromisslösungen, also eine Approximation der Pareto-Front. Um die Leistungsfähigkeit von Algorithmen zu bewerten, benötigt man Konzepte zum Vergleich der produzierten Ergebnismengen. Zu diesem Zweck wurden

Qualitätsmaße, so genannte *Metriken (metrics)* entwickelt. Dieser Begriff ist in gewisser Weise irreführend, da es sich nur um einen Fachbegriff für Maße und nicht notwendigerweise um Metriken im mathematischen Sinne handelt, also um binäre Relationen, die positiv und symmetrisch sind, sowie die Dreiecksungleichung erfüllen. Die Metriken werden entweder direkt auf die Ergebnismengen angewendet oder auf die nicht-dominierte Teilmenge davon.

Metriken können anhand der Anzahl ihrer Argumente kategorisiert werden. Unäre Metriken sind Abbildungen $m(M) : M \rightarrow \mathbb{R}$, die einer Menge M eine reelle Zahl zuweisen. Binäre Metriken $m(M, M')$ werden eingesetzt um zwei Mengen direkt mit einander zu vergleichen. Unäre Maße haben den Vorteil, beliebig viele Mengen durch Sortierung ihrer Abbilder vergleichen zu können.

Es gibt verschiedene Möglichkeiten der Kategorisierung von Metriken. Deb [Deb01] schlägt entsprechend der Aspekte, die die Metriken messen drei Klassen vor: Metriken für die Konvergenz zur Pareto-Front, Metriken für die Diversität der Menge und Metriken, die beides messen. Zitzler et al. [ZTL$^+$03] sowie Knowles und Corne [KC02] betrachten Metriken hinsichtlich des Grades ihrer Kompatibilität mit den Überlegenheits-Relationen, die im Folgenden beschrieben werden.

Eigenschaften von Metriken

Die tatsächliche Leistungsfähigkeit von Metriken wird durch sogenannte Überlegenheitsrelationen (outperformance relations) bewertet, die von Hansen und Jaszkiewicz [HJ98] definiert wurden. Die Überlegenheitsrelationen beziehen sich auf die Dominanz-Relation, die allen Mengen zugrunde liegt. Es wird bewertet, inwiefern die Ordnung, die eine Metrik auf einer Menge von Ergebnismengen induziert mit der Halbordnung der Dominanzrelation kompatibel ist. Es gibt drei Grade der Überlegenheit (outperformance), nämlich die schwache, die starke und die vollständige.

Definition 1.6 (schwache Überlegenheit (weak outperformance)) *Eine Approximation A der Pareto-Front ist einer anderen Approximation B schwach überlegen (AO_wB), wenn mindestens ein Punkt aus A nicht in B enthalten ist und jeder Punkt in B von einem in A schwach dominiert wird.*
$A\, O_w\, B \;\; :\Leftrightarrow \quad i) \;\; A \neq B \quad ii) \;\; ND(A \cup B) = A.$

Definition 1.7 (starke Überlegenheit (strong outperformance)) *Eine Approximation A der Pareto-Front ist einer anderen Approximation B stark überlegen ($A\, O_s\, B$), wenn mindestens ein Punkt aus B von einem Punkt in A dominiert wird und jeder Punkt in B von einem in A schwach dominiert wird.*
$A\, O_s\, B \;\; :\Leftrightarrow \quad i) \;\; ND(A \cup B) = A \quad ii) \;\; B \setminus ND(A \cup B) = \emptyset.$

Definition 1.8 (vollständige Überlegenheit (complete outperformance)) *Eine Approximation A der Pareto-Front ist einer anderen Approximation B vollständig überlegen ($A\, O_c\, B$), wenn jeder Punkt aus B von einem Punkt in A dominiert wird.*
$A\, O_c\, B \;\; :\Leftrightarrow \quad i) \;\; ND(A \cup B) = A \quad ii) \;\; B \cap ND(A \cup B) = \emptyset.$

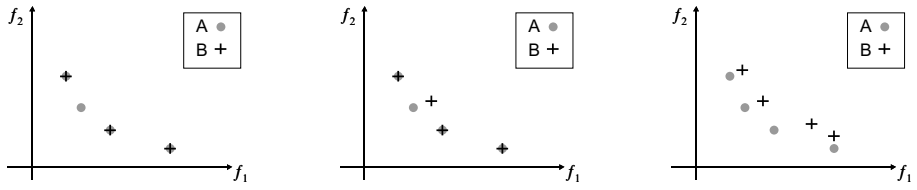

Abbildung 1.3: Veranschaulichung der Überlegenheitsrelationen: Links gilt $A\ O_w\ B$, mittig $A\ O_s\ B$ und rechts $A\ O_c\ B$.

Die drei Überlegenheitsrelationen O_ϕ mit $\phi \in \{w, s, c\}$ haben folgende Eigenschaften.
a) Sie sind transitiv, es gilt: $A\ O_\phi\ B \quad \wedge \quad B\ O_\phi\ C \quad \Rightarrow \quad A\ O_\phi\ C$.
b) Sie sind antisymmetrisch: $A\ O_\phi\ B \quad \wedge \quad B\ O_\phi\ A \quad \Rightarrow \quad A = B$.
c) Sie bilden eine Hierarchie: $A\ O_c\ B \quad \Rightarrow \quad A\ O_s\ B \quad \Rightarrow \quad A\ O_w\ B$,
bzw. $\quad O_w \subset O_s \subset O_c$.

Mengen, die einander nicht vollständig überdecken, also teilweise disjunkte nichtdominierte Elemente enthalten, sind bezüglich der Überlegenheitsrelationen unvergleichbar, diese liefern also in diesen Fällen keinen Zusammenhang. Die schwache und starke Überlegenheitsrelation werden oft implizit in aktuellen Forschungsarbeiten verwendet. Einfache Beispiele der Überlegenheitsrelationen sind in Abbildung 1.3 gezeigt.

Definition 1.9 (Vollständigkeit (completeness)) *Eine Metrik $m(M)$ wird als vollständig bezüglich einer Überlegenheitsrelation O bezeichnet, wenn für alle Mengen A, B gilt:* $A\ O\ B \quad \Rightarrow \quad m(A)$ *besser als* $m(B)$.

Jedes Paar, das mit einer Relation O vergleichbar ist, ist auch mit einer vollständigen Metrik vergleichbar und die induziert für jedes Mengenpaar eine Beziehung, die mit der Ordnung O kompatibel ist.

Definition 1.10 ((schwache) Kompatibilität ((weak) compatibility)) *Eine Metrik m wird als kompatibel zu einer Überlegenheitsrelation O bezeichnet, wenn für alle Mengen A, B gilt:* $m(A)$ *besser als* $m(B) \quad \Rightarrow \quad A\ O\ B$.
Eine Metrik m heißt schwach kompatibel zu einer Überlegenheitsrelation, wenn gilt: $m(A)$ *nicht schlechter als* $m(B) \quad \Rightarrow \quad A\ O\ B$.

Wenn für zwei Mengen $A\ O_s\ B$ gilt, dann bewertet eine zur starken Überlegenheitsrelation kompatible Metrik die Menge A besser als die Menge B. Ist eine Metrik nicht (schwach) kompatibel mit den Überlegenheitsrelationen, so kann ihre Bewertung von Mengen nicht als korrekt angesehen werden.

Zitzler et al. [ZTL+03] zeigen, dass keine unäre Metrik – sogar keine Kombination mehrerer unärer Metriken – existiert, deren implizierte Ordnung äquivalent zur starken Überlegenheitsrelation ist. Das bedeutet, dass es keine Menge unärer Metriken gibt, die gleichzeitig O_s-kompatibel und O_s-vollständig sind. Allerdings existiert theoretisch für den Vergleich endlicher Mengen ein unärer Indikator, der diese Bedingung erfüllt.

Dieser Indikator ist aber nicht in der Lage, die Unterschiede zwischen zwei Mengen zu quantifizieren, sondern nur die bessere zweier Mengen zu bestimmen. Zitzler et al. beschrieben, dass die Anzahl der unären Indikatoren mindestens so groß wie die Anzahl der Zielfunktionen sein muss, um ein Vergleichssystem zu gewährleisten, dass gleichzeitig O_s-vollständig und O_s-kompatibel ist. Dies ist also mit einer festen Anzahl unabhängig vom Problem nicht möglich.

In der vorliegenden Arbeit werden die S-Metrik und die Konvergenzmetrik verwendet, diese beiden Metriken werden daher im Folgenden näher betrachtet.

S-Metrik

Die *S-Metrik (S metric)* wurde erstmals 1998 von Zitzler und Thiele [ZT98] formuliert und als Größe des dominierten oder überdeckten Raums (size of dominated space, size of space covered) bezeichnet. Im mathematischen Zusammenhang entspricht sie dem Lebesgue-Maß (siehe z. B. Bronstein et al. [BSMM00]). Gemäß der Definition der Dominanz-Relation entspricht der von einem d-dimensionalen Punkt $\mathbf{y}$ überdeckte Raum dem (unendlichen) Hyperquader, der durch die d Hyperebenen aufgespannt wird, die jeweils achsen-parallel durch den Punkt verlaufen. Der abgedeckte Raum entspricht damit den Punkten, die in den d-dimensionalen Intervallen $[y_1, \infty) \times \ldots \times [y_d, \infty)$ enthalten sind (vgl. auch Kapitel 3.2.2). Die Größe des abgedeckten Raums zu berechnen, ist nur bei einem endlichen Raum sinnvoll, daher wird ein Referenzpunkt eingeführt, der den Raum begrenzt. Der Referenzpunkt muss dominiert werden von den Punkten, deren überdeckter Raum gemessen werden soll (vgl. Abbildung 1.4). Die Begriffe der S-Metrik und des dominierten bzw. überdeckten Hypervolumens werden im Folgenden synonym verwendet.

Definition 1.11 (S-Metrik (S metric)) *Der Wert der S-Metrik $S(M, \mathbf{r})$ entspricht der Größe des Raums, der von den Punkten einer Menge M überdeckt wird und durch den Referenzpunkt $\mathbf{r}$ beschränkt ist. Der überdeckte Raum einer Menge ist die Vereinigung der jeweils von den Punkten überdeckten Räume, wobei mehrfach überdeckte Bereiche nur einfach gezählt werden. Der überdeckte Hyperquader eines Punktes $\mathbf{y}$ sei $h(\mathbf{y}) = [y_1, r_1] \times \ldots \times [y_d, r_d]$. Der Wert der S-Metrik ist folgendermaßen bestimmt, wobei Λ das Lebesgue-Maß bezeichnet:*

$$S(M, \mathbf{r}) = \Lambda(\{\bigcup h(\mathbf{y}) | \mathbf{y} \in M\}). \tag{1.4}$$

Die S-Metrik zeichnet sich dadurch aus, dass sie als einziges (bekanntes) unäres Maß O_w-vollständig und gleichzeitig kompatibel in dem Sinne ist, dass sie kein Mengenpaar falsch bewertet, für das eine O_s-Relation gilt. Voraussetzung dafür ist, die Beschränktheit des Zielraums, so dass ein Maximalwert der S-Metrik existiert [ZTL$^+$03]. Die S-Metrik unterscheidet zwischen den verschiedenen Überlegenheitsrelationen, sodass eine Menge, die einer anderen überlegen ist, immer besser bewertet wird. Die S-Metrik ist unabhängig von einer linearen Skalierung des Zielraums [Kno02], die induzierte Ordnung von Mengen ändert sich also nicht, wenn man die Zielfunktionswerte mit einem skalaren Wert multipliziert. Sie ist außerdem monoton, was bedeutet, dass jede zusätzliche nicht-dominierte Lösung, den Metrikwert der Menge nicht ver-

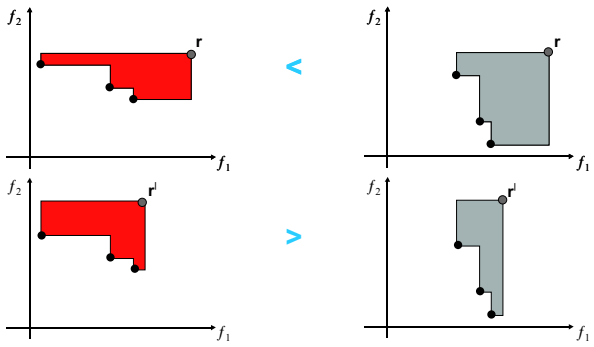

Abbildung 1.4: Das dominierte Hypervolumen (S-Metrik) ist begrenz durch den Referenzpunkt $\mathbf{r}$ bzw. $\mathbf{r}'$. Die rechte Menge hat den größeren Wert bzgl. $\mathbf{r}$ und die linke bzgl. $\mathbf{r}'$.

schlechtert.

Die Ordnung von zwei Mengen, die bezüglich der Überlegenheitsrelationen unvergleichbar sind, kann vom gewählten Referenzpunkt abhängen. Dies ist genau dann der Fall, wenn die zu vergleichenden Mengen sich in ihren Randpunkten unterscheiden.

Definition 1.12 (Randpunkt (boundary point)) *Ein Punkt* $\mathbf{y}$ *heißt Randpunkt einer Menge* M*, wenn gilt:* $\exists y_i$ *mit* $y_i = \max_{\mathbf{y} \in M}\{y_i\}$.

Ein Randpunkt hat also in einer Zielfunktion einen schlechtesten Wert unter den Elementen der Menge. Knowles und Corne [KC02] illustriert den Fall verschiedener Rangordnungen an einem Beispiel, dass in Abbildung 1.4 reproduziert wird. Für zwei Mengen A, B gilt für zwei Referenzpunkte $\mathbf{r}$ und $\mathbf{r}'$: $S(A, \mathbf{r}) < S(B, \mathbf{r}')$ und $S(A, \mathbf{r}') > S(B, \mathbf{r}')$.

Eine Metrik sollte effizient berechenbar sein. Ein negativer Aspekt der S-Metrik ist ihre hohe Laufzeitkomplexität. Eine Übersicht aktueller Berechnungsalgorithmen wird in Kapitel 3 gegeben. Der beste bekannte Algorithmus hat eine worst-case Laufzeit von $O(m^{d/2} \log m)$ (für m Punkte im d-dimensionalen Zielraum), die bezüglich der Kardinalität der Menge polynomiell, aber exponentiell in der Hälfte der Dimension des Suchraums ist.

Fleischer [Fle03b] zeigte einen Zusammenhang zwischen dem Maximum der S-Metrik und der Pareto-Front für die diskrete Mehrzieloptimierung, also im Falle eines endlichen Suchraums mit einer endlichen Pareto-Front. Gegeben sei ein feste Größe l der Punktmenge und ein Referenzpunkt $\mathbf{r}$, der von allen Pareto-optimalen Punkten eines d-kriteriellen Optimierproblems dominiert wird. Eine Punktmenge der Kardinalität l, die unter allen Punktmengen der Kardinalität l den größten S-Metrikwert aufweist, besteht nur aus Pareto-optimalen Punkten. Das Problem, eine Menge zu finden, die den S-Metrikwert maximiert, ist damit äquivalent zu dem Problem, die Pareto-Front PF zu finden. Fleischer unterscheidet genauer zwei Fälle, abhängig von der Größe der Punktmenge.

- Falls $|M| < |PF|$: $N = \text{argmax}_{M \in (\mathbb{R}^d)^l}\{S(M,R)\} \Rightarrow N \subseteq PF$
- Falls $|M| \geq |PF|$: $N = \text{argmax}_{M \in (\mathbb{R}^d)^l}\{S(M,R)\} \Leftrightarrow PF \subseteq N$

Die Beweise der obigen Aussagen sind in [Fle02] zu finden.

Konvergenz-Metrik

Die *Konvergenz-Metrik* misst den mittleren Abstand der Punkte zum jeweils nächst gelegenen Pareto-optimalen Punkt im Zielfunktionsraum. Da die Pareto-Front im Vorfeld der Optimierung im Normalfall nicht bekannt ist und auch bei akademischen Testfunktionen häufig nicht analytisch berechnet werden kann, verwendet man statt der wahren Pareto-Font eine Teilmenge dieser als Referenzmenge. Diese Teilmenge kann durch vorige Experimente (approximativ) gewonnen werden.

Definition 1.13 (Konvergenz-Metrik (convergence metric)) *Der Wert von $K(M,R)$ ist der durchschnittliche Abstand der Elemente in M zu dem jeweils nächst gelegenen Punkt einer Menge Pareto-optimaler Punkte $R \subseteq PF$.*

$$K(M,R) \quad = \quad 1/|M| \cdot \left(\sum_{\mathbf{y} \in M} \min_{\mathbf{v} \in R} \{dist(\mathbf{y}, \mathbf{v})\} \right). \tag{1.5}$$

Die Funktion $dist(\mathbf{y}, \mathbf{v})$ misst den euklidischen Abstand der beiden Punkte $\mathbf{y}$ und $\mathbf{v}$.

Die Konvergenzmetrik ist interessant, weil sie die Annäherung an die Pareto-Front beschreibt. Da nur genau eine Lösung letzlich zur Realisierung ausgewählt wird, ist es wichtig, dass diese Lösung tatsächlich annähernd Pareto-optimal ist.

Die Konvergenzmetrik ist nicht monoton, eine zusätzliche nicht-dominierte Lösung kann den Metrikwert verschlechtern. Ein deutlicher Nachteil ist natürlich, dass die Pareto-Front (oder eine Teilmenge dieser) bekannt sein muss. Die Metrik ist weder vollständig noch kompatibel zu einer der Überlegenheitsrelationen. Das wird an einem einfachen Beispiel deutlich: Eine Menge, die einen nicht-dominierten Punkt enthält, der weit von der Pareto-Front entfernt liegt, wird besser bewertet als eine Menge, die diesen Punkt nicht enthält und ansonsten gleich ist.

Die Konvergenzmetrik kann ganz naiv in Zeit $O(|M| \cdot |R|)$ berechnet werden, indem man für jedes Element in M den Abstand zu jedem Element in R berechnet und diese vergleicht. Effizientere Algorithmen sind aktuell nicht bekannt.

1.5 Leistungsvergleiche von Algorithmen

Als Kriterien der Leistungsfähigkeit von Algorithmen betrachtet man zum Einen die erzeugte Ergebnismenge und zum Anderen die Ressourcen, die zu deren Erzeugung benötigt werden. Vergleiche der Ergebnismengen wurden im vorausgehenden Abschnitt 1.4 behandelt. Im Folgenden werden Modelle zum Laufzeitvergleich beschrieben und es wird auf Besonderheiten von Vergleichsszenarien hingewiesen.

Black-Box-Szenario

Evolutionäre Algorithmen gehören zu den *Black-Box-Optimierern*, d. h. sie gewinnen Informationen über das zu optimierende Problem nur punktuell über Suchpunkte, die sie anfragen und deren Funktionswert sie erhalten. Das zu optimierende System ist für den Algorithmus ansonsten unbekannt, also eine Black-Box. Eine Auswertung der Funktion erfolgt dann unabhängig vom EA, es besteht lediglich eine Informationsschnittstelle zwischen Auswerter und Algorithmus. Die Suchpunkte werden bei allen randomisierten Suchheuristiken probabilistisch in Abhängigkeit der bisherigen Suchpunkte und deren Funktionswerten gewählt (vgl. Wegener [Weg03]).

Die Probleme, auf die evolutionäre Algorithmen angewendet werden, sind oftmals hoch komplex, sodass für sie keine geschlossene mathematische Formulierung vorliegt und sie können tatsächlich nur punktuell und oftmals nur mit Hilfe von Simulatoren ausgewertet werden. Simulationen, wie sie zum Beispiel für physikalische oder soziale Modelle verwendet werden, sind in der Regel sehr zeitaufwändig. Im Vergleich dazu ist die Rechenzeit, die ein evolutionärer Algorithmus für die Durchführung seiner Operatoren benötigt, vernachlässigbar gering und man betrachtet daher die Funktionsauswertung als den Bestandteil des EA, der die Größenordnung der Laufzeit bestimmt. Im Black-Box-Szenario verwendet man als Berechnungsmodell für die Laufzeit die Anzahl von Funktionsauswertungen, alle Rechenoperationen außerhalb der Black-Box werden vernachlässigt.

In der Mehrzieloptimierung gibt es verschiedene Auffassungen darüber, ob als Funktionsauswertung die Auswertung genau einer Zielfunktion oder die Auswertung aller d Zielfunktionen anzusehen ist. Problematisch ist diese Uneinigkeit beispielsweise, wenn man einen Mehrzieloptimier-Algorithmus auf einem mehrkriteriellen Problem vergleicht mit einem einkriteriellen Algorithmus, der ein äquivalentes Problem optimiert, allerdings aufgespaltet in mehrere einkriterielle. Problemspezifisch sollten die Faktoren mit größtem Zeitaufwand, also beispielsweise die Anzahl benötigter Simulation, bei beiden Szenarien gleich sein, um die Vergleichbarkeit zu gewährleisten.

No Free Lunch Theorem

Bei universellen Optimierern wie EA würde man gern in Bezug auf alle Optimierprobleme eine Aussage treffen wie „ein Algorithmus A ist besser als ein Algorithmus B“. Das No Free Lunch (NFL)-Theorem besagt, dass genau dies nicht zutrifft und zeigt dadurch die Grenzen der möglichen vergleichenden Aussagen über das Verhalten von Algorithmen im Black-Box-Szenario.

Theorem 1.1 (NFL-Theorem) *Für endliche Suchräume mit endlichen Bildbereichen gilt, dass im Durchschnitt über alle Funktionen, alle Algorithmen, die keinen Suchpunkt mehrfach anfragen, bezüglich der Leistungsaspekte wie z. B. Optimierzeit oder Erfolgswahrscheinlichkeit alle das gleiche Verhalten aufweisen [WM97]. Für die Algorithmen muss zudem gelten, dass sie jeden Suchpunkt mit positiver Wahrscheinlichkeit anfragen.*

Diese Aussage ist auf den ersten Blick überraschend, besagt sie doch, dass ein durchdachtes Optimierverfahren nicht besser funktioniert als beispielsweise die deterministische Anfrage von Suchpunkten in lexikographischer Reihenfolge oder eine rein zufällige Suche – und dass auch kein besseres entwickelt werden kann. Das NFL-Theorem trifft eine Aussage über alle Probleme. Es stellt sich die Frage, ob das NFL-Theorem tatsächlich auch für praxisrelevante Fälle gilt. Die Einschränkung auf endliche Suchräume ist nicht wesentlich, da man bei der Repräsentation der Daten in einem Rechner immer einen – sehr großen, aber – endlichen Suchraum vorliegen hat. Wesentlich ist die Argumentation über *alle* Funktionen, die den Ansatzpunkt für die Kritik liefert, dass das NFL-Szenario nicht realistisch ist.

In der Praxis kann man nur Funktionen betrachten, deren Ressourcen für die Darstellung und Auswertung im Computer beschränkt sind. Zudem gilt das NFL-Theorem für eine Funktionenklasse genau dann, wenn diese *unter Permutationen abgeschlossen (closed under permutation)* ist [DJW02]. Das bedeutet, dass jede Permutation einer Funktion, wieder in der Funktionenmenge enthalten ist[3]. Bei einer permutierten Funktion bleibt die Menge der Funktionswerte identisch, aber die Zuordnung zwischen Argumenten und Funktionswerten ist umgeordnet. Da durch beliebige Permutationen jedem Argument jeder Funktionswert zugewiesen werden kann, verbietet die Funktionenklassen geradezu eine Struktur auf dem Suchraum, die bei realen Optimierproblem stets angenommen werden kann. Das NFL-Theorem liefert also eine fundamentale Aussage, die allerdings eingeschränkt auf praxisrelevante Fälle nicht zutrifft (vgl. Droste et al. [DJW99]).

Droste et al. [DJW02] formulieren ein *ANFL (almost no free lunch)*, welches auch auf realistische Szenarien zutrifft. Dieses besagt, dass es für jede Funktion, die ein Algorithmus effizient optimieren kann, exponentiell viele andere Funktionen ähnlicher Komplexität (gemessen in z. B. Auswertungszeit oder Schaltkreisgröße) existieren, auf denen er versagt, also exponentielle Laufzeit hat. Auch ein universeller Algorithmus arbeitet also immer nur auf einer bestimmten Menge von Funktionen effizient. Das sollte nicht überraschen, da den Strategien jeder Heuristik gewisse Annahmen über die Problemstruktur zugrunde liegen.

Das NFL-Theorem wurde entwickelt für einkriterielle Optimierungsprobleme, neuere Arbeiten übertragen diese Betrachtung auf die Mehrzieloptimierung. Corne und Knowles [CK03a] zeigen, dass man Mehrzieloptimierprobleme auf einkriterielle Probleme zurückführen kann und übertragen dadurch die Gültigkeit des NFL-Theorems auf die Mehrzieloptimierung. Das NFL-Theorem gilt analog zur einkriteriellen Optimierung, wenn man die von Algorithmen generierten Suchpunkte mit absoluten Leistungsmaßen (Metriken) vergleicht.

Corne und Knowles betrachten außerdem die Einschränkung auf bestimmte Pareto-Fronten, d. h. der Raum der Funktionen wird über deren Wertebereich beschränkt. Gerne möchte man Aussagen treffen wie „ein EMOA A ist besser als EMOA B, im Falle einer konvexen, konkaven oder unzusammenhängenden Front“. Tatsächlich gilt auch

[3] Insbesondere ist die Menge aller Funktionen unter Permutationen abgeschlossen.

hier das NFL-Theorem, denn die Menge aller Funktionen, die eine bestimmte Pareto-Front erzeugen, ist unter Permutationen abgeschlossen. Dies gilt auch dann, wenn man nicht eine bestimmte Punktmenge als Pareto-Front festlegt, sondern eine Klasse von Pareto-Fronten mit einer bestimmten Form betrachtet. Da das NFL-Theorem auf die einzelnen Pareto-Fronten zutrifft, gilt es auch für die gesamte Klasse, die diese Pareto-Fronten bilden. Auch bei dieser Sichtweise ist natürlich wieder zu beachten, dass die Betrachtung aller Funktionen kein realistisches Szenario ist.

Es wurden auch Fälle aufgezeigt, in denen kein NFL-Theorem gilt. Corne und Knowles [CK03b] betrachten symmetrische, vergleichende Leistungsmaße, welche die Ergebnismengen zweier Algorithmen paarweise direkt vergleichen, durch eine binäre, symmetrische Relation. Sie zeigen eine Aussage, die sie als *free leftovers in permutation spaces* bezeichnen: Betrachtet man einen Permutationsraum (das bedeutet, die zu optimierende Funktion ist injektiv), dann gilt für jeden Algorithmus, dass ein anderer Algorithmus existiert, der diesem bezüglich einer gewählten symmetrischen Vergleichsmetrik unterlegen ist. Der Beweis wurde geführt, indem ein Gegenbeispiel konstruiert wurde, welches das NFL-Theorem widerlegt. Es ist zu beachten, dass diese Aussage nur möglich ist, weil die Vergleichsmetrik nicht transitiv ist.

Im Falle absoluter Vergleichsmetriken gilt kein NFL, wenn man einen Mehrzieloptimier-Algorithmus als Kopplung von Suchpunkt-Generator und Archivierer betrachtet. Die Eigenschaften des Archivs, nur eine Auswahl und nicht alle erzeugten Punkte zu betrachten, widersprechen den Voraussetzungen des NFL-Theorems. Da bei der Mehrzieloptimierung nicht eine beste Lösung gesucht ist, sondern eine Menge möglichst guter Kompromisslösungen, speichert man die bisher besten – also die nichtdominierten Punkte – in einem Archiv, das notwendigerweise größenbeschränkt ist. Man benötigt also eine Archivierungsstrategie, die eine Auswahl der gefundenen Lösungen trifft. Wesentlich ist die Erkenntnis, dass diese archivierte Menge nicht unbedingt repräsentativ für die Menge der gesamten generierten Suchpunkte sein muss und daher das NFL für das Archiv nicht gilt. Diese Aussage wird *GRATIS (Generator/Archiver Theorem in Search)-Theorem* genannt. Ein Beweis wird einfach dadurch geführt, dass man Algorithmen mit verschiedenen Archivgrößen betrachtet, sodass es einem Algorithmus möglich ist, die gesamte nicht-dominierte Menge zu speichern und einem anderen nicht, welcher dann im Vergleich schlechter bewertet wird.

Wir halten fest, dass in realistischen Szenarien kein NFL-Theorem gilt und bei Mehrzieloptimierproblemen insbesondere Archivierungsstrategien bzw. die geschickte Auswahl einer guten Teilmenge aus einer nicht-dominierten Menge eine wichtige Rolle spielen. Es sind also nicht alle Algorithmen gleich gut und es ist durchaus sinnvoll, über den Entwurf guter Mehrzieloptimier-Algorithmen nachzudenken.

Laufzeit einer Generation

Wendet man einen EA auf ein Problem an, dessen Funktionsauswertung „effizient“ möglich ist, so spielt die Rechenzeit des EA doch eine wesentliche Rolle. Man betrachtet daher als Gegenstück zum Black-Box-Szenario die Laufzeit des EA für eine Generation. Hierbei zählt man nur die Rechenoperationen, die der EA benötigt und

die Zeit für die Funktionsauswertung wird vernachlässigt (bzw. als konstant angenommen). Für den SMS-EMOA und andere Algorithmen ist die Laufzeit einer Generation in Kapitel 2 jeweils angegeben.

Empirische Studien

Die theoretische Analyse evolutionärer Algorithmen ist hoch komplex und exakte Ergebnisse für Laufzeitverhalten, garantierte Approximationen oder Konvergenzeigenschaften sind bisher nur für sehr spezielle Betrachtungen bestimmter (einfacher) Algorithmen auf bestimmten (einfach strukturierten) Funktionen möglich. Das Hilfsmittel zur Analyse moderner EA auf schwierigen Funktionen sind daher experimentelle Studien.

Nicht nur evolutionäre Algorithmen, sondern auch andere Suchheuristiken arbeiten mit Parametern, die das Laufzeit- und Optimierverhalten wesentlich beeinflussen. Wenn man zwei Algorithmen auf einer Funktion vergleicht, muss man beachten, ob man Parametrisierungen gewählt hat, die für den Algorithmus und die Problemstellung geeignet sind. Es ist schwierig gute Parametereinstellung für einen Algorithmus auf einer Funktion zu finden, die Parametrisierung ist also selbst ein Optimierproblem. Durch Experimente geeignete Parametereinstellungen zu finden ist ein eigenes Forschungsgebiet, das Konzepte zur Versuchsplanung und Auswertung hervorgebracht hat, wie *DoE – Design of Experiments*, *DACE – Design and Analysis of Computer Experiments* oder *SPO – Sequenzielle Parameter Optimierung* [BB06].

Es gibt verschiedene Kriterien, die Parametrisierungen erfüllen sollen. Einerseits könnte man nach einer besten Einstellung eines Algorithmus auf genau einer Funktion suchen. Derartige Experimente belegen, dass eine gute Anpassung des Algorithmus an ein Problem möglich ist. Interessant ist dann insbesondere der Spielraum in der Qualität des Algorithmus bei guten oder schlechten Einstellungen. Andererseits wäre es wünschenswert, Parametrisierungen zu kennen, die auf mehreren Funktionen im Durchschnitt gut funktionieren, sodass man hoffen darf, dass diese auch auf einer Funktion mit unbekannter Struktur erfolgreich sind. Die Einstellung ist also robust, da sie unter verschiedenen Bedingungen funktioniert. Zumeist sind Standard-Parameterwerte bekannt, die vielversprechend oder robust erscheinen. In den empirischen Studien in Kapitel 4 werden die Algorithmen anhand dieser Standardwerte, die von den Algorithmen-Designern vorgegeben wurden, verglichen.

Kapitel 2

S-Metrik Selektion (SMS)– EMOA

Dieses Kapitel ist dem neu entwickelten Algorithmus *S-Metrik Selektion EMOA (S metric selection EMOA)* – kurz SMS-EMOA gewidmet. Zunächst werden im folgenden Abschnitt 2.1 Motivation und Idee des Algorithmendesigns erläutert und anschließend die grundlegende Funktionsweise beschrieben (Abschnitt 2.2). Eigenschaften, die aus der Konzeption des SMS-EMOA resultieren werden in Abschnitt 2.3 erläutert. Danach liefert Abschnitt 2.4 einige mögliche Modifikationen bzw. Varianten des SMS-EMOA, die abhängig vom Problem vorteilhaft sein können. Schließlich wird der SMS-EMOA mit anderen etablierten sowie neuen, ähnlichen Verfahren verglichen (Abschnitt 2.5).

2.1 Motivation und Grundidee

Die Zielsetzung bei der Entwicklung des SMS-EMOA war, einen Algorithmus zu entwerfen, der in der Lage ist, eine kleine Population entlang der Pareto-Front gut zu verteilen. Dies berücksichtigt die Bedürfnisse eines Praktikers, der nur eine geringe Anzahl Funktionsauswertungen durchführen kann und dennoch den Bereich interessanter Kompromisslösungen abdecken möchte. Der Algorithmus soll außerdem einfach zu implementieren sein und sich daher auf möglichst wenige algorithmische Konzepte beschränken. So wird beispielsweise auf die Nutzung eines Archivs verzichtet und die Optimierung allein auf der Grundlage der aktuellen Population durchgeführt.

Die S-Metrik wird verwendet, um die Qualität von multidimensionalen Mengen zu vergleichen (vgl. Abschnitt 1.4). Dieses Maß zeichnet sich durch seine vorteilhaften theoretischen Eigenschaften aus und wird daher als eine der gerechtesten Bewertungsmethoden für Mengen angesehen. Ein guter S-Metrikwert der Ergebnismenge eines EMOA ist demnach wünschenswert. Da man nun weiß, was man erreichen möchte, erscheint es sinnvoll und legitim, direkt darauf hin zu arbeiten. Die Berechnung der S-Metrik wird dazu in den Optimierprozess integriert und die Maximierung des S-Metrikwerts der Population angestrebt. Die Selektion wird so gestaltet, dass stets die Teilmenge von Individuen als Folgepopulation ausgewählt wird, deren S-Metrikwert maximal ist unter den möglichen Teilmengen.

2.2 Basis-Algorithmus

Ablauf einer Generation

Der SMS-EMOA von Emmerich et al. [EBN05] ist als steady-state Algorithmus konzipiert, d. h. es wird pro Generation ein Nachkomme erzeugt und ein Individuum aussortiert. Das Hypervolumen-basierte Selektionskriterium bildet die halbgeordneten Zielfunktionsvektoren auf reelle Fitnesswert ab. Dies ermöglicht eine vollständige Ordnung der Lösungen und eine entsprechende Selektion der besten.

Algorithmus 2.1 SMS-EMOA

1: $P_0 \leftarrow \text{init}()$ — Initialisiere zufällige Start-Population von μ Individuen
2: $t \leftarrow 0$
3: **repeat**
4: $\quad \mathbf{o} \leftarrow \text{generate}(P_t)$ — Erzeuge 1 Nachkommen durch Variationsoperatoren
5: $\quad \{F_1, \ldots, F_v\} \leftarrow \text{fast-nondominated-sort}(P_t \cup \{\mathbf{o}\})$ — v Fronten
6: $\quad \mathbf{a}^* \leftarrow \text{argmin}_{\mathbf{a}=(\mathbf{w},\mathbf{x},\mathbf{y},\mathbf{z}) \in F_v}[\Delta_S(\mathbf{y}, F_v)]$ — $\mathbf{a}^* \in F_v$ mit kleinstem $\Delta_S(\mathbf{y}, F_v)$
7: $\quad P_{t+1} \leftarrow (P_t \cup \{\mathbf{o}\}) \setminus \{\mathbf{a}^*\}$ — entferne gewähltes Element
8: $\quad t \leftarrow t + 1$
9: **until** Abbruchkriterium erfüllt

Der Ablauf des SMS-EMOA ist in Algorithmus 2.1 dargestellt. Für die Startpopulation werden μ Individuen zufällig generiert oder problemspezifisch mit bekannten Lösungen initialisiert. Der evolutionäre Prozess beginnt damit, dass zwei Individuen zufällig gleichverteilt aus der Population gewählt werden. Aus diesen Eltern wird per Rekombination und anschließender Mutation ein Nachkomme erzeugt[1].

Auf der temporären Menge der $(\mu + 1)$ Individuen führt die Methode fast-nondominated-sort die Partitionierung in Fronten gemäß der nicht-dominierten Sortierung durch (vgl. Definition 1.5). Ihre Implementierung ist aus dem von Deb et al. [DPAM02] entwickelten Algorithmus NSGA-II übernommen und in Anhang B dargestellt. In einer Vorverarbeitungsphase wird für alle Individuenpaare die Dominanz-Relation überprüft. Die Informationen werden gespeichert, indem für jedes Individuum eine Liste der dominierten Individuen verwaltet wird. Außerdem gibt es einen Zähler für die Anzahl der dominierenden Lösungen. Die erste Front (Rang 1) besteht nun aus den Individuen, deren Zähler den Wert null hat. Die nicht-dominierte Front wird anschließend gedanklich aus der Menge entfernt, indem die Zähler der dominierten Lösungen – erkennbar aus der Liste – um eins reduziert werden. Man verfährt analog für die folgenden Fronten (Rang 2 bis Rang v) bis die Menge vollständig partitioniert ist.

Anschließend wird aus der schlechtesten Front – also der Front mit dem höchsten Rang – ein Individuum aussortiert. Das Individuum, welches den kleinsten Beitrag zum dominierten Hypervolumen der schlechtesten Front leistet, wird nicht in die Folgepopulation aufgenommen. Die Fitness $\mathbf{z}$ eines Individuums $\mathbf{a}$ ist somit primär der Rang seiner zugehörigen Front und sekundär sein Hypervolumenbeitrag: $\mathbf{z}_1 = i$, mit $\mathbf{a} \in F_i$ und falls $\mathbf{z}_1 = v$ der Hypervolumenbeitrag $\mathbf{z}_2 = \Delta_S(\mathbf{y}, F_v)$, wie nachfolgend definiert.

[1] Diese Variation ist ein häufiges Grundkonzept; es sind auch andere Variationsoperatoren einsetzbar.

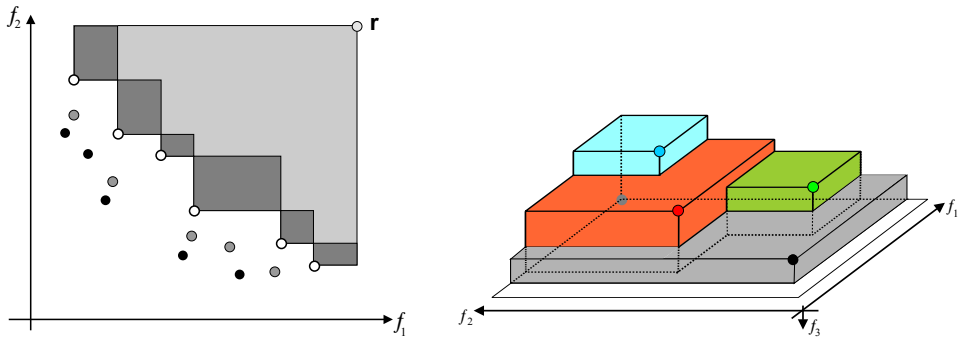

Abbildung 2.1: Hypervolumenbeiträge der dritten Front einer Population in einem zwei-dimensionalen Zielraum (links) und ein Beispiel eines drei-dimensionalen Zielraums (rechts).

Hypervolumenbeitrag als Selektionskriterium

Definition 2.1 (Hypervolumenbeitrag (hypervolume contribution)) $S(M, \mathbf{r})$ *bezeichnet den S-Metrikwert einer Menge M bezüglich eines Referenzpunktes* $\mathbf{r}$ *gemäß Definition 1.11. Der Hypervolumenbeitrag* $\Delta_S(\mathbf{y}, M)$ *eines Punktes* $\mathbf{y} \in \mathbb{R}^d$ *zu einer Menge* M *aus* d*-dimensionalen Punkten ist definiert als*

$$\Delta_S(\mathbf{y}, M) \;=\; S(M, \mathbf{r}) - S(M \setminus \{\mathbf{y}\}, \mathbf{r}). \tag{2.1}$$

Der Hypervolumenbeitrag eines Individuums $\mathbf{a}$ ist definiert über den Beitrag seines Zielfunktionsvektors $\mathbf{y}$ zum S-Metrikwert der zugehörigen Front. Der Beitrag ist die Differenz, die sich zwischen den S-Metrikwerten der entsprechenden Front und der Front ohne den dedizierten Punkt ergibt. Dies entspricht der Größe des Raums, der nur von dem entsprechenden Individuum dominiert wird und von keinem anderen der Front.

Die Beiträge sind für den Fall eines zwei-dimensionalen und eines drei-dimensionalen Zielraums durch zwei Beispiele dargestellt (Abbildung 2.1), um eine Intuition der Definition zu vermitteln. Die linke Abbildung zeigt eine Population aus drei Fronten: die nicht-dominierten Lösungen sind schwarz dargestellt, die zweite Front grau und die Punkte der dritten, schlechtesten Front als unausgefüllte Kreise. Das dominierte Hypervolumen der schlechtesten Front ist hellgrau dargestellt und die Beiträge der Lösungen jeweils durch die dunklen Rechtecke.

In einem zwei-dimensionalen Zielraum ist die Berechnung der Hypervolumenbeiträge effizient möglich. Man sortiert die Zielfunktionsvektoren bezüglich eines Kriteriums, also beispielsweise aufsteigend bezüglich f_1. Die resultierende Punktsequenz ist dadurch automatisch absteigend bezüglich f_2 sortiert, da alle Punkte sich gegenseitig nicht dominieren (vgl. Abbildung 2.1, links). Für das j-te Individuum $a^{(j)}$, $j \in \{2, \ldots, |F_v|\}$, in der sortierten Sequenz der Front F_v berechnet sich die sekundäre Fitness folgendermaßen als Hypervolumenbeitrag:

$$z_2^{(j)} = \Delta_S(\mathbf{y}^{(j)}, F_v) \;=\; \left(y_1^{(j+1)} - y_1^{(j)} \right) \cdot \left(y_2^{(j-1)} - y_2^{(j)} \right). \tag{2.2}$$

In den beiden Dimension des Zielfunktionsraums werden jeweils die Differenzen zum nächst schlechteren Punkt berechnet und diese multipliziert. Für die Randpunkte (vgl. Definition 1.12) der Front, also die Punkte, die aus einem schlechtesten und einem besten Wert bestehen, wird im zwei-dimensionalen Zielraum kein Beitrag berechnet. Die Randpunkte werden als besonders wertvoll angesehen, weil sie die Ausbreitung der Population entlang der Pareto-Front bestimmen. Im Fall zweier Zielfunktionen werden die zugehörigen Individuen daher immer in die Folgepopulation übernommen[2], in höheren Dimension wird allerdings anders verfahren (vgl. Abschnitt 2.4).

Abbildung 2.1, rechts veranschaulicht, dass die Beiträge im drei-dimensionalen Zielraum im Allgemeinen nicht einfache Quader darstellen, sondern komplexere Formen annehmen können. Da die Berechnung der Beiträge bzw. des Hypervolumens einer Menge in hoch-dimensionalen Räumen schwierig und für den SMS-EMOA so wichtig ist, ist ihr ein eigenes Kapitel gewidmet. In Kapitel 3 werden verschiedene Verfahren zur Berechnung des Hypervolumens und für Hypervolumenbeiträge beschrieben.

Variationsoperatoren

Die Variationsoperatoren können aus der Literatur bzw. problemspezifisch gewählt werden, es wurden keine speziellen Operatoren für den SMS-EMOA entworfen. Für die Anwendungen auf Testfunktionen, die in Abschnitt 4.2 vorgestellt sind, wurden als Rekombinationsoperator die *simulierte binäre Rekombination (simulated binary crossover, SBX)* und als anschließender Mutationsoperator die *polynomielle Mutation (polynomial mutation, PM)* eingesetzt (siehe [Deb01]). Diese werden beispielsweise auch von den in Abschnitt 2.5 beschriebenen Algorithmen NSGA-II und IBEA verwendet. Auf den realen aeronautischen Problemstellungen der Studien von Naujoks et al. erwiesen sich einfache traditionelle Operatoren der Evolutionsstrategien [BS02] als vorteilhaft. Da die Variationsoperatoren kein zentrales Thema dieser Arbeit darstellen, wurde ihre Darstellung in den Anhang verschoben. Außerdem ist die Beschreibung der Operatoren SBX und PM komplex. Eine ausführliche Beschreibung mit Hinweisen auf Parametrisierung und Implementierung findet sich in Anhang B.

2.3 Eigenschaften des SMS-EMOA

Laufzeit einer Generation

Die Laufzeit einer Generation des SMS-EMOA wird hier aufgeschlüsselt. Wie in Abschnitt 1.5 bereits diskutiert, wird die Rechenzeit für die Funktionsauswertung der Zielfunktion nicht berücksichtigt. Sie gilt als unbekannt, da sie stark vom Problem abhängig ist. Es wird hier also die reine Berechnungskomplexität der evolutionären Operatoren dargestellt. Die Rechenzeit zur Erzeugung eines Individuums mit Hilfe der Variationsoperatoren ist linear bezüglich der Populationsgröße. Die Laufzeit ist somit bestimmt durch den rechenintensiveren Selektionsoperator.

[2] Falls in einem zwei-dimenionalen Zielraum die schlechteste Front nur aus Randpunkten besteht, wird eines dieser Individuen zufällig aussortiert.

Der Algorithmus fast-nondominated-sort zur nicht-dominierten Sortierung hat eine Laufzeit von $O(d\mu^2)$. Die Vorverarbeitungsphase, in der die Dominanz aller Individuen überprüft wird, benötigt $O(d\mu^2)$. Die Partitionierung in Fronten erfordert pro Front $O(\mu)$ Vergleiche und es kann maximal μ Fronten geben.

Die Anzahl der notwendigen Hypervolumenberechnungen ist abhängig von der verwendeten Größe der Nachkommenpopulation. Um die optimale Nachfolgepopulation zu bestimmen, muss das Hypervolumen jeder möglichen Zusammensetzung der Population berechnet werden. Bei einem $(\mu + \lambda)$-Selektionsschema mit $\lambda > 1$ gäbe es $\binom{\mu+\lambda}{\mu}$ mögliche Populationen, deren Hypervolumen verglichen werden muss. Durch den steady-state-Ansatz ist die Anzahl von Hypervolumenberechnungen mit $\binom{\mu+1}{\mu} = \mu + 1$ linear in der Populationsgröße. Dies ist der Hauptgrund für die Verwendung des steady-state-Konzepts.

Im Falle eines zwei-dimensionalen Zielraums werden die Beiträge gemäß Gleichung 2.2 berechnet und das Verfahren hat in Verbindung mit der vorausgehenden Sortierung eine Laufzeit von $O(\mu \log \mu)$. Auch die nicht-dominierte Sortierung hat für den zweidimensionalen Spezialfall diese Laufzeit, die im wesentlichen durch die Sortierung bestimmt ist. In höher dimensionalen Räumen können die Beiträge bestimmt werden, indem man jeweils einen Punkt aus der Menge ausschließt und die Differenz zum Hypervolumen der Gesamtmenge ist sein Beitrag. Für d-dimensionale Räume $d \geq 3$ existiert ein Verfahren mit Laufzeit $O(\mu^{d/2} \log \mu)$. Bei $\mu + 1$ Aufrufen des Verfahrens pro Generation ergibt sich eine Laufzeit von $O(\mu^{(d/2)+1} \log \mu)$. Die Verfahren zur Hypervolumenberechnung und deren Laufzeiten sind Thema von Kapitel 3.

Schlechtere Selektionskonzepte

Wir diskutieren kurz alternative Designmöglichkeiten für einen Selektionsoperator bei $\lambda > 1$, um die auftretende Problematik zu verdeutlichen. Denkbar wäre die Verwendung einer Greedy-Strategie zur Selektion. Man legt also ein Kriterium fest und wählt als Folgepopulation die μ Individuen, die es am besten erfüllen.

Statt der Beiträge könnte man das absolute Volumen der Zielfunktionsvektoren vergleichen und diesbezüglich die besten Individuen für die Folgepopulation auswählen. Abbildung 2.2 (links) zeigt, dass die ausgewählte Population nicht optimal zusammengesetzt ist, wenn man zwei von drei Lösungen danach auswählt. Nach absolutem Hypervolumen ist der Punkt $\mathbf{y}^{(2)}$ am besten und wird ausgewählt. Hinzu kommt noch einer der Punkte $\mathbf{y}^{(1)}$ oder $\mathbf{y}^{(3)}$, die beide den gleichen S-Metrikwert aufweisen. Ein größeres Hypervolumen ist allerdings erreichbar durch Wahl der Punkte $\mathbf{y}^{(1)}$ und $\mathbf{y}^{(3)}$ ohne $\mathbf{y}^{(2)}$.

Eine andere Möglichkeit ist, die Beiträge der Individuen innerhalb der Menge der $(\mu + \lambda)$ Individuen berechnen. Daraufhin könnte man die μ Individuen mit den größten Beiträgen für die Nachfolgepopulation auswählen. Das Beispiel in Abbildung 2.2 (rechts) veranschaulicht, dass mit dieser Strategie ebenfalls nicht die optimale Nachfolgepopulation gewählt wird für $\lambda > 1$. Für $\lambda = 2$ sind zwei Individuen zu verwerfen und man wählt nach den Beiträgen die Zielfunktionsvektoren $\mathbf{y}^{(2)}$ und $\mathbf{y}^{(3)}$. Dadurch

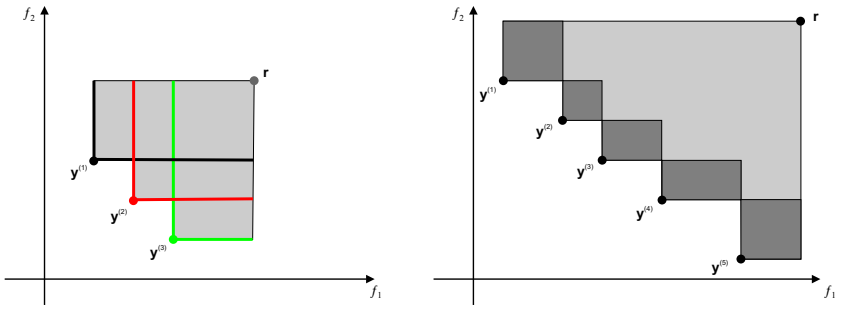

Abbildung 2.2: Illustration alternativer Selektionskonzepte, welche nicht zu einem optimalen S-Metrikwert führen. Links: Gegenbeispiel bei Selektion nach absolutem S-Metrikwert. Rechts: Gegenbeispiel bei Selektion nach Beiträgen bei $\lambda > 1$.

verliert man aber nicht nur die exklusiven Beiträge der jeweiligen Zielfunktionsvektoren, sondern auch das Hypervolumen, was ausschließlich von beiden Punkten abgedeckt wird. Ein schlechtes Ergebnis ergibt sich durch Punkte, die nahe aneinander liegen und daher jeder einzelne nur einen kleinen Beitrag leistet. Entfernt man allerdings alle diese ähnlichen Punkte, so verliert man auch das Hypervolumen, das diese Punkte gemeinsam abdecken, was wesentlich mehr als die Summe der Beiträge sein kann. Eine Population, die mehr Hypervolumen abdeckt, ergibt sich durch Verwerfen von $\mathbf{y}^{(2)}$ und $\mathbf{y}^{(4)}$.

Die beiden angedachten Greedy-Strategien führen nicht zur Auswahl einer Population mit maximalem S-Metrikwert. Um im Falle $\lambda > 1$ eine optimal zusammengesetzte Population zu erhalten, müsste man die S-Metrikwerte jeder möglichen Subpopulation berechnen. Das verlangsamt den Algorithmus wegen der hohen Laufzeit der Hypervolumenberechnung sehr stark. Nur beim steady-state-Ansatz ($\lambda = 1$) kann man das Individuum mit kleinstem Beitrag entfernen und die restlichen Individuen ergeben eine optimal zusammengesetzte Population.

Monotonie des S-Metrikwertes einer Population

Der S-Metrikwert der Population ist schwach monoton steigend. Als Folgepopulation wird immer diejenige ausgewählt, die den größten S-Metrikwert aufweist. Dadurch kann dieser Wert niemals sinken, wenn er mit einem gleich bleibenden Referenzpunkt berechnet wird. Es gilt also

$$\forall t \geq 0 : \quad S(P_t, \mathbf{r}) \leq S(P_{t+1}, \mathbf{r}). \tag{2.3}$$

Der S-Metrikwert ist immer dann echt größer als der vorige, wenn eine nicht-dominierte Lösung eine andere mit niedrigerem Beitrag ersetz hat. Wie in Abschnitt 1.4 erwähnt, wird der maximale S-Metrikwert von der Pareto-Front angenommen.

Konvergenz gegen Pareto-Front

Knowles und Corne [KC03] verwenden den S-Metrik-Beitrag als Selektionskriterium innerhalb eines größenbeschränkten Archivs nicht-dominierter Lösungen. Ist ein endli-

cher Suchraum mit einer endlichen Pareto-Front gegeben und werden alle Suchpunkte mit positiver Wahrscheinlichkeit erzeugt, dann gilt Folgendes: Das Archiv, welches gemäß der Strategie verwaltet wird, konvergiert gegen eine Teilmenge der Pareto-Front. Der Beweis lässt sich auf den SMS-EMOA übertragen, sodass gilt: Die Population des SMS-EMOA konvergiert gegen eine Teilmenge der Pareto-Front. Der Beweis und die Beschreibung der Archivierungsstrategie werden in Abschnitt 2.5 näher erläutert und mit dem SMS-EMOA in Bezug gesetzt.

Diversitätserhaltung

Die Population ändert sich durch das steady-state Konzept nur langsam, nämlich nur um ein Individuum pro Generation. Durch die Verwaltung von dominierten Individuen ist stets eine hohe Diversität gegeben. Würde man nur nicht-dominierte Individuen in der Population behalten, so kann die Populationsgröße im Extremfall auf eins sinken, wenn ein Individuum alle anderen dominiert. Ein derartiger Diversitätsverlust wird durch die konstante Populationsgröße vermieden.

Verwaltung eines Archivs

Der SMS-EMOA benötigt innerhalb des Optimierprozesses kein Archiv. Es ist aber natürlich möglich, ein Archiv aller nicht-dominierten Lösungen zu verwalten. Wir behaupten allerdings, dass die Individuen der Population eine gute Verteilung aufweisen, so dass ein Archiv nicht unbedingt notwendig ist. Bei realen Problemen ist es dennoch empfehlenswert alle gefundenen nicht-dominierten Lösungen zu speichern. Der Entscheidungsfinder kann anhand der finalen Population sehen, welche Kompromisse möglich sind und wird dann sicher Lösungen einer bestimmten Region der Pareto-Front bevorzugen. Dann kann er eventuell auf das Archiv zurück greifen, um aus ähnlichen Lösungen eine auszuwählen, die für die praktische Realisierung am besten geeignet ist. Wir gehen davon aus, dass reale Probleme typischerweise so komplex sind, dass sie nicht vollständig modelliert werden können, sondern ein abstrahiertes bzw. vereinfachtes Problem optimiert wird. Das führt dazu, dass die Eignung ähnlicher Lösungen sehr verschieden sein kann und ein Archiv vieler (auch ähnlicher) Lösungen hilfreich ist.

2.4 Varianten des SMS-EMOA

2.4.1 Selektionen nach Anzahl dominierender Punkte

Zusätzlich zu dem präsentierten Basis-Algorithmus (vgl. Algorithmus 2.1) wurden von Naujoks et al. [NBE05b] verschiedene Varianten des SMS-EMOA entwickelt. Es wurde ein alternatives Selektionskriterium entwickelt, welches in manchen Fällen statt der Selektion bezüglich der Hypervolumenbeiträge verwendet werden kann. Die bisherige Selektion strebt eine Maximierung des Hypervolumens der schlechtesten Front an. Eine neue Idee ist, die Selektion abhängig von der Verteilung der Lösungen auf den vorderen Fronten zu gestalten. In Regionen des Zielraums, in denen die vorderen Fronten dicht besetzt sind, erscheint es nicht sinnvoll, die schlechteren Individuen der hintersten Front zu behalten. Andererseits sind Individuen der schlechtesten

Front interessant, wenn sich in ihrer Region nur wenige bessere Individuen aufhalten. Es soll die Möglichkeit gewahrt bleiben, dass diese Lücken in den vorderen Fronten durch Nachkommen der Individuen hinterer Fronten aufgefüllt werden können. Für das alternative Selektionskriterium wird ein neues Maß definiert, nämlich die Anzahl dominierender Punkte, genannt *Dominanzzahl*.

Definition 2.2 (Dominanzzahl (dominance number)) *Die Dominanzzahl $DZ(\mathbf{y}, M)$ entspricht der Anzahl dominierender Punkte eines Punktes $\mathbf{y} \in \mathbb{R}^d$ innerhalb einer Menge M:*
$DZ(\mathbf{y}, M) = |\{\mathbf{y}' \in M | \mathbf{y}' \prec \mathbf{y}\}|$.

Es wird eine Selektion bezüglich des Maßes $DZ(\mathbf{y}, M)$ durchgeführt, wenn dominierte Individuen auftreten, also die nicht-dominierte Sortierung mehr als eine Front hervorbringt. Es wird das Individuum der schlechtesten Front aussortiert, dessen Zielfunktionswert den größten Wert von $DZ(\mathbf{y}, M)$ aufweist. Liegt dagegen nur eine Front vor, so gilt $DZ(\mathbf{y}, M) = 0$ für alle Zielfunktionsvektoren und es ist keine Rangordnung möglich. In diesem Fall greifen wir wieder auf den Hypervolumenbeitrag als Selektionskriterium zurück. Die modifizierte Selektion ist in Algorithmus 2.2 dargestellt. Sie ersetzt die Zeilen 5 und 6 des Basis-Algorithmus 2.1. Die Selektion bezüglich der Dominanzzahl ist in Abbildung 2.3 im Vergleich zur Selektion anhand der Hypervolumenbeiträge veranschaulicht. Es zeigt sich, dass bezüglich der Dominanzzahl ein Individuum entfernt wird, das von vielen anderen dominiert wird, der Δ_S-Operator aber ein anderes wählen würde.

Algorithmus 2.2 Reduce_domPoints(P)	von SMS-EMOA-domPoints
1: $\{F_1, \ldots, F_v\} \leftarrow$ nondominated-sort(P)	alle v Fronten von P
2: **if** $v > 1$ **then**	
3: $\quad \mathbf{a}^* \leftarrow$ argmax$_{\mathbf{a}=(\mathbf{w},\mathbf{x},\mathbf{y},\mathbf{z}) \in F_v}[DZ(\mathbf{y}, F_v)]$	$\mathbf{a}^* \in F_v$ mit größtem $DZ(\mathbf{y}, F_v)$
4: **else**	
5: $\quad \mathbf{a}^* \leftarrow$ argmin$_{\mathbf{a}=(\mathbf{w},\mathbf{x},\mathbf{y},\mathbf{z}) \in F_1}[\Delta_S(\mathbf{y}, F_1)]$	$\mathbf{a}^* \in F_1$ mit kleinstem $\Delta_S(\mathbf{y}, F_1)$
6: **end if**	
7: **return** $(P \setminus \mathbf{a}^*)$	entferne gewähltes Individuum

Das Maß $DZ(\mathbf{y}, M)$ kann in einer Zeit von $O(d\mu^2)$ bestimmt werden. Im Zielfunktionsraum wird $\mathbf{y}$ mit allen Zielfunktionsvektoren besserer Fronten verglichen, um festzustellen, von welchen $\mathbf{y}$ dominiert wird. Für alle Elemente der schlechtesten Front wird dies durchgeführt. Bei der nicht-dominierten Sortierung mit dem fast-nondominated-sort Algorithmus wird die Anzahl dominierender Punkte ohnehin bestimmt (vgl. Anhang B).

Das Maß der Anzahl dominierender Punkte $DZ(\mathbf{y}, M)$ ähnelt einem Konzept, welches im von Zitzler et al. [ZLT02] entwickelten Strength Pareto Evolutionary Algorithm (SPEA2) verwendet wird. Dabei wird für jeden Punkt die Anzahl der dominierten Punkte bestimmt. Die Fitness eines Punktes ist die Summe dieser Werte über alle dominierenden Punkte. Nicht-dominierte Punkte haben Fitness null und die Fitness ist zu minimieren. Das Schema ist in Abschnitt 2.5.2 näher beschrieben.

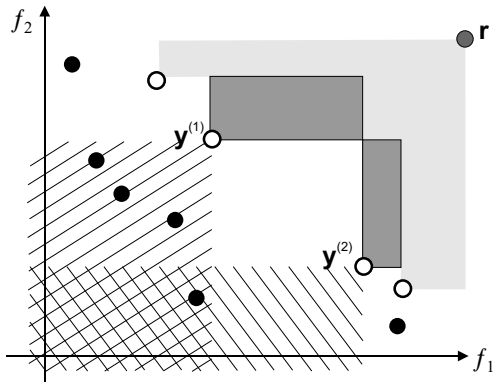

Abbildung 2.3: Vergleich der Dominanzzahl DZ und des Hypervolumenbeitrags Δ_S. Der Punkt $\mathbf{y}^{(1)}$ wird bzgl. Δ_S besser bewertet als $\mathbf{y}^{(2)}$, wie die gefüllten Rechtecke andeuten. Die Schraffuren zeigen die Bereiche, in denen Punkte liegen, die $\mathbf{y}^{(1)}$ bzw. $\mathbf{y}^{(2)}$ dominieren. Bezüglich der Dominanzzahl hat $\mathbf{y}^{(1)}$ den Wert 4 und $\mathbf{y}^{(2)}$ den Wert 1 und wird daher bevorzugt.

Als weitere Alternative wird ein Selektionsoperator betrachtet, das der oben beschriebenen Variante enspricht, aber die nicht-dominierte Sortierung auslässt. Die Dominanzzahl wird für alle Individuen berechnet und die Selektion erfolgt analog. Im Falle dominierter Individuen wird unter diesen das Individuum mit maximaler Dominanzzahl aussortiert. Stellt die Population einer nicht-dominierte Menge dar, ist die Dominanzzahl aller Individuen gleich null und die Selektion wird unter allen Individuen der Population bezüglich des Hypervolumenbeitrags durchgeführt.

2.4.2 Behandlung des Referenzpunktes

Die Wahl des Referenzpunktes bestimmt den absoluten S-Metrikwert einer Menge. Der Beitrag eines Punktes ist genau dann vom Referenzpunkt abhängig, wenn der Punkt ein Randpunkt innerhalb seiner Front ist (vgl. Definition 1.12). Da die Beiträge sich bezüglich der nächst schlechteren Nachbarpunkte berechnen und ein Randpunkt in mindestens einem Kriterium keinen schlechteren Nachbarpunkt hat, ist sein dominierter Raum in der entsprechenden Dimension nur vom Referenzpunkt begrenzt.

Im zwei-dimensionalen Fall wird der Referenzpunkte lediglich benötigt, um die Beiträge der Randpunkte zu berechnen, von denen es in diesem Fall nur zwei gibt. Sie bestehen jeweils aus einem schlechtesten und einem besten Zielfunktionswert. Das disjunkte Hypervolumen aller anderen Punkte ist durch die jeweiligen Nachbarpunkte begrenzt und daher vom Referenzpunkt unabhängig. Der SMS-EMOA bevorzugt im zwei-dimensionalen Fall die Randpunkte gegenüber den restlichen Elementen der Front, sodass sie stets in die Folgepopulation übernommen werden. Die Voraussetzung ist, dass die schlechteste Front mehr als zwei Punkte enthält, ansonsten wird einer der Randpunkte zufällig gewählt und verworfen. Man kann alsi im Falle $d = 2$ auf die Beitragsberechnungen für die beiden Randpunkte verzichten und braucht daher keinen Referenzpunkt zu spezifizieren.

In höher dimensionalen Räumen ist diese Sonderbehandlung der Randpunkte nicht

empfehlenswert, da es sehr viele Randpunkte geben kann. Bereits in einem drei-dimensionalen Zielraum ist es möglich, dass eine Front ausschließlich aus Randpunkten besteht. Beispielsweise können alle Punkte in einer Dimension den gleichen Wert haben und dennoch eine nicht-dominierte Menge darstellen, weil die Nicht-Dominanz bezüglich der beiden anderen Zielfunktionen erfüllt ist. In d-dimensionalen Zielräumen mit $d > 2$ werden daher auch für Randpunkte Hypervolumenbeiträge berechnet und sie werden wie die restlichen Punkte diesbezüglich behandelt.

Die Beiträge der Randpunkte können sehr groß sein, wenn der Referenzpunkt entsprechend weit von der Front entfernt ist. Das würde dazu führen, dass Randpunkte stets in der Population enthalten bleiben, was die Konvergenz zur Pareto-Front behindert. Ein innerer Punkt kann deutlich besser sein als seine Nachbarpunkte und dennoch ist sein Hypervolumenbeitrag im Vergleich zu dem eines Randpunktes klein. Verbesserungen können also bei diesem Vorgehen kaum gewürdigt werden. Wir haben uns daher für eine dynamische Anpassung des Referenzpunktes entschieden, das heißt der Referenzpunkt wird in jeder Generation in Abhängigkeit der aktuellen Population neu bestimmt. Seine Koordinaten werden berechnet als

$$\mathbf{r}_i = \max_{\mathbf{y} \in F} \{y_i\} + 1. \tag{2.4}$$

Er besteht aus dem aktuell schlechtesten Funktionswert in jeder Dimension addiert zu eins. Randpunkte haben also in der Dimension, in der sie einen schlechtesten Wert aufweisen, einen Abstand von eins zum Referenzpunkt. Da sich das Volumen als Produkt über die Dimensionen errechnet, verhält sich somit die Differenz zum Referenzpunkt neutral in der Rechnung und der Beitrag bestimmt sich aus den restlichen Dimensionen. Der dynamische Referenzpunkt wird in den Studien in Abschnitt 4.3.2 verwendet und die Ergebnisse sind deutlich besser als mit einem zuvor angedachtem statischen Referenzpunkt.

2.4.3 Dynamische Populationsgröße

Die Basis-Variante des SMS-EMOA verwendet eine feste Populationsgröße μ. Es wurde eine Abwandlung entwickelt, bei der die Population nur aus nicht-dominierte Lösungen besteht und daher schwankende Größe hat. Dieses Vorgehen soll den evolutionären Prozess am Anfang beschleunigen. Dominierte Lösungen werden direkt verworfen und daher im Reproduktionsprozess nicht berücksichtigt. Die Idee ist, dass eine kleine Population eine schnelle Konvergenz zur Pareto-Front erreicht. Wenn man sich der Pareto-Front angenähert hat, sollen die Populationen anwachsen und sich entlang der Pareto-Front ausbreiten.

Es wird eine maximale Populationsgröße μ_{max} festgelegt und zusätzlich eine initiale μ_{init}. Bei einer sehr kleinen Population droht ein Diversitätsverlust, das heißt, dass sich alle Individuen sehr ähnlich sind. Die initiale Populationsgröße darf daher nicht zu klein gewählt werden, wie die Studien in Abschnitt 4.3.2 zeigen. Es wurden zwei Varianten mit dynamischer Populationsgröße entwickelt. Die eine (genannt *SMS-EMOA-dynPop1*) behält nur nicht-dominierte Individuen (vgl. Algorithmus 2.3). Falls die Anzahl nicht-dominierter Individuen die maximale Populationsgröße μ_{max} überschreitet,

Algorithmus 2.3 Reduce_dynPop1(P) — von SMS-EMOA-dynPop1

```
1: F_1 ← ND(P)                                   nicht-dominierte Individuen von P
2: if |F_1| > μ_max then
3:    a* ← argmin_{a=(w,x,y,z)∈F_1}[Δ_S(y, F_1)]   a* ∈ F_1 mit kleinstem Δ_S(y, F_1)
4:    return (F_1 \ {a*})                        entferne gewähltes Element
5: else
6:    return F_1                                 behalte alle nicht-dominierte Individuen
7: end if
```

wird ein Individuum entsprechend des Hypervolumenbeitrags aussortiert. Die zweite Variante (genannt *SMS-EMOA-dynPop2*) verläuft in zwei Phasen (vgl. Algorithmus 2.4). Solange die maximale Populationsgröße noch nicht erreicht wurde, verhält sich der Algorithmus wie die erste Variante SMS-EMOA-dynPop1. Danach wird die Populationsgröße konstant gehalten und die Selektion wird wie beim originalen SMS-EMOA durchgeführt, also nicht-dominierte Sortierung gefolgt von der Selektion entsprechend des Hypervolumenbeitrags. Die Idee ist, dass nahe der Pareto-Front auch dominierte Lösungen behalten werden. Die erhöhte Diversität soll die Ausbreitung der Population erleichtern.

Algorithmus 2.4 Reduce_dynPop2(P) — von SMS-EMOA-dynPop2

```
 1: if phase == 1 then                                   /* Phase 1 */
 2:    F_1 ← ND(P)                                       nicht-dominierte Individuen von P
 3:    if |F_1| = μ_max then
 4:       phase ← 2                                      Umschaltung auf Phase 2
 5:    end if
 6:    return F_1                                        behalte alle nicht-dominierte Individuen
 7: else                                                 /* Phase 2 */
 8:    {F_1, ..., F_v} ← fast-nondominated-sort(P_t ∪ {o})   v Fronten
 9:    a* ← argmin_{a=(w,x,y,z)∈F_v}[Δ_S(y, F_v)]        a* ∈ F_v mit kleinstem Δ_S(y, F_v)
10:    return (P \ a*)                                   schlechtestes Individuum aussortiert
11: end if
```

Denkbar wäre auch die Spezifikation einer minimalen Populationsgröße, das wurde bisher allerdings nicht untersucht. Zu den anderen vorgestellten Konzepten finden sich Studien in Abschnitt 4.3.2.

2.4.4 Parallelisierbarkeit

Aufgrund des steady-state-Ansatzes kann der SMS-EMOA leicht parallelisiert werden. Die Erzeugung, Fitness-Auswertung und die Hypervolumenberechnung mehrerer Individuen kann gleichzeitig durchgeführt werden. Die Hauptroutine des Algorithmus führt die Selektion anhand der berechneten Werte durch, sobald ein neues Ergebnis vorliegt. Dies entspricht einem Master-Slave-Konzept in der die Hauptroutine die Master-Funktion einnimmt und an einen Slave die aktuelle Population weitergegeben wird, auf der er die Prozeduren generate, evaluate, fast-nondominated-sort und die Hypervolumenberechnung durchführt (vgl. Algorithmus 2.1). Die Parallelisierung der Hypervolumenberechnung ist aufgrund der hohen Laufzeit wünschenswert.

Algorithmus 2.5 NSGA-II

1: $P_0 \leftarrow$ init() Initialisiere zufällige Start-Population von μ Individuen
2: $t \leftarrow 0$
3: **repeat**
4: *Parents* $\leftarrow$ binaryTournamentSelection(P_t) wähle Eltern der Nachkommen
5: *Offspring* $\leftarrow$ SBX(Parents) erzeuge μ Nachkommen durch Rekombination
6: *Offspring* $\leftarrow$ PolynomialMutation(*Offspring*) mutiere μ Nachkommen
7: $F \leftarrow$ fast-nondominated-sort(*Offspring* $\cup P_t$) Fronten $F = \{F_1, ..., F_k\}$
8: **while** $|P_{t+1} \cup F_i| < \mu$ **do**
9: $P_{t+1} \leftarrow P_{t+1} \cup F_i$ füge Fronten in Folgepopulation ein
10: i++
11: **end while**
12: crowdingDistance(F_i) berechne Besiedelungsdichte
13: $\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(|F_i|)}\} \leftarrow$ sortDescending(F_i, crowdingDistance)
14: $P_{t+1} \leftarrow P_{t+1} \cup \{\mathbf{a}^{(j)} \in F_i | j \leq (\mu - |P_{t+1}|)\}$
15: $t \leftarrow t + 1$
16: **until** Abbruchkriterium erfüllt

Zu beachten ist, dass sich die Population der Hauptroutine während der Hypervolumenberechnung eines Slaves ändern kann. Der Beitrag ist somit nicht mehr aktuell bzw. fehlerbehaftet.

2.5 Vergleiche mit anderen Verfahren

Aktuelle state-of-the-art EMOA dienten als Vorlagen bei der Entwicklung des SMS-EMOA. Diese und dem SMS-EMOA ähnliche Verfahren, die zeitnah entstanden, werden hier kurz aufgeführt. Die folgenden Verfahren haben mit dem SMS-EMOA gemeinsam, dass sie eine Abbildung der Zielfunktionsvektoren auf rellwertige Fitnesswerte verwenden. Diese dienen innerhalb des Selektionsoperators zur Indizierung einer vollständigen Ordnung der ansonsten halbgeordneten Zielfunktionsvektoren. Die Selektion entsprechend der S-Metrik wurde auch in andere Heuristiken integriert, die hier beschrieben werden. Weitere Möglichkeiten wurden bisher nur angedacht. Fleischer formulierte in [Fle03b] die Idee, das dominierte Hypervolumen als zu maximierende Zielfunktion bei Simulated Annealing einzusetzen. Außerdem schlägt Fleischer [Fle03a] vor, das Hypervolumen bei Schwarm-Intelligenz-Verfahren als Maß für die Effizienz von Lösungen einzusetzen.

2.5.1 Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)

Der NSGA-II von Deb et al. [DAPM00] wurde als Nachfolger des NSGA (Non-dominated Sorting Genetic Algorithm [SD95]) entwickelt. Der Rang innerhalb der nicht-dominierten Sortierung dient als primäres Selektionskriterium und ein Diversitätsmaß als sekundäres. In diversen Studien hat der NSGA-II seine Leistungsfähigkeit unter Beweis gestellt und ist ein allgemein bekanntes, etabliertes Verfahren.

Die Start-Population der Größe μ wird zufällig initialisiert. Eine Generation des NSGA-II verläuft wie folgt. Zur Reproduktion werden Individuen durch eine binäre Turnierselektion (binary tournament selection) ausgewählt. Dabei werden zwei zufälli-

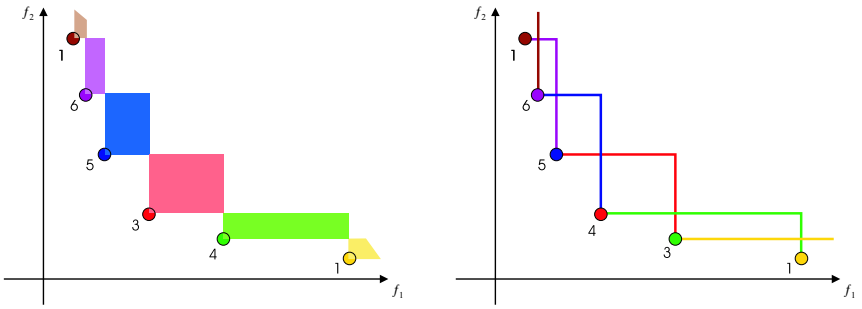

Abbildung 2.4: Vergleich von Hypervolumenbeitrag (links) und Besiedelungsdichte (rechts).

ge Paare von Individuen gewählt und die Individuen, die besser als der Vergleichspartner sind, werden zu Eltern. Mit dem Elternpaar werden durch Simulated Binary Crossover (SBX) zwei Nachkommen erzeugt. Man fährt fort, bis die Nachkommenpopulation die Größe μ erreicht hat. Die Nachkommen werden anschließend durch eine polynomielle Mutation verändert. Nach der Auswertung der Nachkommenpopulation wird sie mit der Elternpopulation vorübergehend vereinigt, sodass die $(\mu + \mu)$-Selektion durchgeführt werden kann. Studien zeigen, dass die elitistische Selektion gegenüber einer Komma-Strategie in der Pareto-Optimierung einen Leistungsvorteil erzielt [Rud01, ZT98]. Zunächst wird die Menge der Individuen mit der nichtdominierten Sortierung partitioniert. Bei der Selektion der Nachfolgepopulation werden nun die Fronten der Reihe nach (mit aufsteigendem Rang) übernommen. Passt eine Front nicht vollständig in die Menge, so wird die *Besiedelungsdichte (crowding distance)* als Selektionskriterium verwendet und die besten Individuen dieser Front entsprechend hinzugefügt, bis die Folgepopulation vollständig ist.

Definition 2.3 (Besiedelungsdichte (crowding distance)) *Die Besiedelungsdichte $G(\mathbf{y}, M)$ ist für Randpunkte einer Menge M definiert als unendlich. Für die anderen Punkte $\mathbf{y} \in \mathbb{R}^d$ in M gilt:*

$$\begin{aligned} G(\mathbf{y}, M) \quad = \quad & \sum_{i=1}^{d} \quad \min\{y'_i - y_i \mid y_i < y'_i \,,\, \mathbf{y}' \in M\} \\ & + \quad \min\{y_i - y''_i \mid y''_i < y_i \,,\, \mathbf{y}'' \in M\}. \end{aligned} \tag{2.5}$$

Eine algorithmische Berechnung des Maßes ist in [DAPM00] zu finden. Dabei wird die Front bezüglich eines Kriteriums sortiert und einem Punkt $\mathbf{y}$ wird der Abstand zwischen dem nächst kleineren ($\mathbf{y}''$ in Formel 2.5) und dem nächst größeren Wert ($\mathbf{y}'$) zugeordnet. Für alle Zielfunktionen wird die Sortierung und Berechnung analog durchgefürt und die Werte über alle d Zielfunktionen addiert. Den Randpunkten wird ein unendlich großer Wert zugewiesen, so dass sie eine besonders gute Bewertung erhalten. Die Laufzeitkomplexität einer Generation des NSGA-II ist bestimmt durch die nicht-dominierte Sortierung und beträgt damit $O(d\mu^2)$, wie im vorigen Abschnitt erläutert.

Die Ähnlichkeit mit dem SMS-EMOA besteht im Einsatz der nichtdominierte Sortierung als primäres Selektionskriterium. Als sekundäres Selektionskriterium steht dem

Hypervolumenbeitrag die Besiedelungsdichte gegenüber. Bei beiden Verfahren werden Randlösungen bevorzugt behandelt. Im Spezialfall eines zwei-dimensionalen Zielraums sind die verwendeten Abstände tatsächlich die gleichen, wie Abbildung 2.4 zeigt. Der Unterschied ist, dass der Hypervolumenbeitrag auf der Multiplikation von Abständen beruht und die Besiedelungsdichte auf der Addition. Der Hypervolumenbeitrag ist größer in Regionen guter Kompromisslösungen und induziert eine Ordnung, die unserer Meinung nach gegenüber der Rangordnung der Besiedelungsdichte vorteilhaft ist. Im Beispiel ist ein Punkt beschriftet mit seinem Rang bezüglich des betrachteten Selektionskriteriums. Die Punkte mit Rang drei stellen also jeweils die am besten bewerteten inneren Punkte dar. Die Besiedelungsdichte eines Punktes ist allerdings nur indirekt abhängig von seiner Position, weil es lediglich auf den Nachbarpunkten basiert und nicht auf der Position des Punktes selbst.

2.5.2 Evolution Strategy with Probabilistic mutation (ESP)

Huband et al. [HHWB03] verwenden in ihrem Algorithmus *ESP (Evolution Strategy with Probabilistic mutation)* eine Approximation des Hypervolumenbeitrags als sekundäres Selektionskriterium. Das Selektionsschema lehnt sich an das des SPEA2 an (vgl. auch Abschnitt 2.4.1).

Algorithmus 2.6 ESP

1: $P_0 \leftarrow$ init() — Initialisiere zufällige Start-Population von μ Individuen
2: $t \leftarrow 0$
3: **repeat**
4: *Parents* $\leftarrow P_t$ — wähle Eltern der Nachkommen
5: *Offspring*.$\mathbf{x} \leftarrow$ uniformCrossover(*Parents*) — erzeuge μ Nachkommen
6: *Offspring*.$\mathbf{w} \leftarrow$ intermediateCrossover(*Parents*)
7: *Offspring* $\leftarrow$ probabilisticMutation(*Offspring*) — mutiere μ Nachkommen
8: evaluate(*Offspring*)
9: $F \leftarrow$ fast-nondominated-sort(*Offspring* $\cup P_t$) — Fronten $F = \{F_1, ..., F_v\}$
10: fitnessCalculation(F) — Grundfitness und Stärke wie bei SPEA2
11: $P_{t+1} \leftarrow F_1$
12: **if** $|P_{t+1}| < \mu$ **then** — Selektion bzgl. Grundfitness
13: $\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(\mu - |P_{t+1}|)}\} \leftarrow$ sortAscending($\bigcup_{i=2}^{v} F_i$, Grundfitness)
14: $P_{t+1} = P_{t+1} \cup \{\mathbf{a}^{(j)} \mid \mathbf{a}^{(j)} \in \bigcup_{i=2}^{v} F_i, j \leq (\mu - |P_{t+1}|)\}$
15: **else if** $|P_{t+1}| > \mu$ **then** — Selektion bzgl. HD$_{approx}$
16: $\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(|F_i|)}\} \leftarrow$ sortDescending(F_1, HD$_{approx}$)
17: $P_{t+1} = \{\mathbf{a}^{(j)} | (\mathbf{a}^{(j)} \in F_1, j \leq \mu\}$
18: **end if**
19: $t \leftarrow t + 1$
20: **until** Abbruchkriterium erfüllt

Es wird eine reellwertige Repräsentation verwendet. Als Eltern-Individuen wird deterministisch die gesamte aktuelle Population gewählt und es werden μ Nachkommen durch Rekombination erzeugt. Man wählt dazu ein Elternpaar und wählt für jede Entscheidungsvariable zufällig einen Elter, dessen Wert übernommen wird. Die Strategievariablen werden als Mittelwerte der jeweiligen Elternwerte übernommen. Das Rekombinationsschema der Entscheidungsvariablen wird als diskret und das der Strategievariablen als intermediär bezeichnet. Wie der Name des ESP-Algorithmus andeu-

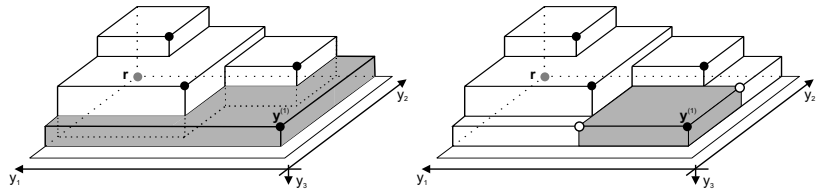

Abbildung 2.5: Vergleich von exaktem Hypervolumenbeitrag Δ_S, der vom SMS-EMOA verwendet wird (links) und dem approximativem Beitrag HD_{approx} des Algorithmus ESP (rechts).

tet, wird ein neuartiger Mutationsoperator namens *probabilistic mutation* verwendet. Mit einer Mutationswahrscheinlichkeit von jeweils $p_m = 1/n$, also dem Kehrwert der Anzahl der Entscheidungsvariablen, wird jede Entscheidungsvariable mutiert. Dieses Vorgehen ist üblich bei eine binären Repräsentation, aber neu für reellwertige Suchräume, bei denen normalerweise jede Entscheidungsvariable mutiert wird. Für $p_m = 1$ entspricht diese Mutation also einem herkömmlichen Operator der Evolutionstrategien. Vor der Mutation werden die Strategievariablen per Selbstanpassung modifiziert nach dem Schema, beschrieben von Bäck et al. [BHS97].

Nach der Funktionsauswertung der Nachkommen wird die nicht-dominierte Sortierung durchgeführt. Dabei werden zwei zusätzliche Maße verwaltet, mit denen nach einem Schema aus dem SPEA2 die Fitness berechnet wird. Die Stärke einer Lösung ist definiert als die Anzahl der Lösungen, die sie dominiert. Die Grundfitness einer Lösung berechnet sich als die Summe der Stärke-Werte der sie dominierenden Individuen. Die Fitness ist somit zu minimieren und nicht-dominierte Lösungen haben eine Fitness von null.

Bei der Selektion wird zunächst die nicht-dominierte Front übernommen. Ist die Menge zu klein, werden dominierte Individuen hinzugefügt in der Reihenfolge ihrer Grundfitness bis die Folgepopulation vollständig ist. Hat die erste Front mehr als μ Elemente werden die *approximativen Hypervolumenbeiträge* berechnet, die Menge entsprechend sortiert und die besten μ Individuen übernommen. Nur bei gleicher Grundfitness werden die approximativen Hypervolumenbeiträge als sekundäres Selektionskriterium verwendet. Dieser Fall trifft im wesentlichen auf die nicht-dominierte Front zu. Es wird also nicht eine der hinteren Fronten vollständig bezüglich des Hypervolumens bewertet, im Unterschied zum SMS-EMOA.

Definition 2.4 (Approx. Hypervolumenbeitrag (app. hypervolume contribution)) *Der approximative Hypervolumenbeitrag HD_{approx} eines Punktes zu einer Menge M ist für Randpunkte definiert als unendlich. Für die anderen Punkte $\mathbf{y} \in \mathbb{R}^d$ in M gilt*

$$HD_{approx}(\mathbf{y}, M) \;=\; \prod_{i=1}^{d} \min\{y'_i - y_i \mid y_i < y'_i\,,\;\; \mathbf{y}' \in M\}. \tag{2.6}$$

Es wird in jeder Dimension des Zielraums die Differenz zum nächsten Nachbarpunkt berechnet und die Werte multipliziert. Für den Fall eines zwei-dimensionalen Zielfunktionsraums entspricht dieses Maß sogar dem exakten Hypervolumen-Beitrag wie er in Definition 2.1 festgelegt ist. In höheren Dimensionen stellt das Maß allerdings

lediglich eine Approximation des Beitrag mit unbekannter Approximationsgüte dar. Der Vorteil besteht darin, dass das Maß effizient zu berechnen ist. Die verwendeten Distanzen erinnern an die Besiedelungsdichte, bei dem Distanzen addiert werden. Der approximative Hypervolumenbeitrag ist, im Unterschied zur Besiedelungsdichte, direkt von der Positionen der Punktes selbst abhängig.

Der ESP-Algorithmus wurde auf den ZDT-Funktionen studiert [HHWB03]. Ein Vergleich der Ergebnisse mit Hilfe von 50%-Attainment-Surfaces mit denen von NSGA-II und SPEA2 zeigt, dass ESP den anderen Algorithmen überlegen ist. Huband et al. schreiben die guten Ergebnisse allerdings dem verwendeten Mutationsoperator zu.

2.5.3 Indicator-Based Evolutionary Algorithm (IBEA)

Zitzler und Künzli [ZK04] stellen fest, dass viele moderne EMOA mehrere Maße zur Selektion heranziehen, nämlich eines, dass auf der Dominanz-Relation basiert und die Konvergenz zur Pareto-Front anstrebt und zusätzlich Techniken, um eine bestimmte Verteilung der Lösungen entlang der Pareto-Front zu erreichen. Präferenzen des Entscheiders können durch diese Techniken direkt in den Optimierprozess integriert werden. Sie verallgemeinern dieses Konzept in ihrem *Indicator-Based Evolutionary Algorithm (IBEA)* [ZK04], der eine beliebige Präferenz-Relation als Selektionskriterium verwenden kann.

Algorithmus 2.7 IBEA

1: $P_0 \leftarrow$ init() — Initialisiere zufällige Start-Population von μ Individuen
2: $t \leftarrow 0$
3: **repeat**
4: *Parents* $\leftarrow$ binaryTournamentSelection(P_t) — wähle μ Eltern der Nachkommen
5: *Offspring* $\leftarrow$ SBX(*Parents*) — erzeuge μ Nachkommen
6: *Offspring* $\leftarrow$ PM(*Offspring*) — mutiere μ Nachkommen
7: evaluate(*Offspring*) — Zielfunktionsauswertung der Nachkommen
8: fitnessCalc($P_t \cup$ *Offspring*) — Fitnessberechnung nach Formel 2.9 mit Indikator
9: $\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(2\mu)}\} \leftarrow$ sortDescending$(P_t \cup$ *Offspring*$)$
10: $P_{t+1} = \{\mathbf{a}^{(j)} \mid \mathbf{a}^{(j)} \in$ *Offspring* $\cup P_t \ , \ j \leq \mu\}$
11: $t \leftarrow t + 1$
12: **until** Abbruchkriterium erfüllt

Indikatorfunktionen I sind binäre Relationen im Zielraum, die einem Individuenpaar eine reelle Zahl zuweisen: $I : (\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) \rightarrow \mathbb{R}$. Binäre Metriken (siehe [HJ98] für eine Übersicht) können als Indikatoren integriert werden, vorausgesetzt, sie sind kompatibel zur schwachen Outperformance-Relation (vgl. Definition 1.6), sodass ihre induzierte Ordnung nicht der Dominanz-Relation widerspricht. Durch den paarweisen Vergleich sind $O(\mu^2)$ Indikatoraufrufe durchzuführen, um eine vollständige Ordnung herzustellen.

In [ZK04] werden zwei Indikatorfunktionen studiert: ein ϵ-Indikator, der Abstände zwischen den Lösungen misst und eine Funktion, die dem Hypervolumenbeitrag entspricht. Der binäre additive ϵ-Indikator $I_{\epsilon+}$ addiert die Verschiebungen im Zielfunktionsraum, die in jeder Dimension möglich bzw. erforderlich sind, so dass eine Lösung

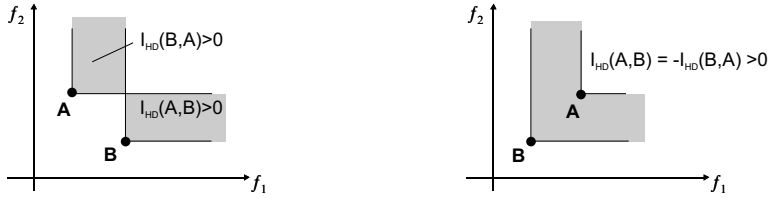

Abbildung 2.6: Binärer Hypervolumen-Indikator I_{HD} des IBEA. Bei zwei unvergleichbaren Punkten sind beide Indikatorwerte positiv (links). Ein negativer Wert bedeutet, dass der erste Punkt den zweiten dominiert (rechts).

$\mathbf{y}^{(1)}$ eine Lösung $\mathbf{y}^{(2)}$ schwach dominiert. Der Indikator wurde in [ZTL$^+$03] für den Vergleich von Mengen eingeführt und wird hier nur für den Spezialfall des Vergleichs zweier Individuen dargestellt.

$$I_{\epsilon^+}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \sum_{i=1}^{d} y_i^{(1)} - y_i^{(2)} \tag{2.7}$$

Der Hypervolumen-Indikator I_{HD} ist definiert als

$$I_{HD}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \begin{cases} S(\{\mathbf{y}^{(2)}\}, \mathbf{r}) - S(\{\mathbf{y}^{(1)}\}, \mathbf{r}) & \text{falls} \quad \mathbf{y}^{(1)} \prec \mathbf{y}^{(2)} \\ S(\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}\}, \mathbf{r}) - S(\{\mathbf{y}^{(1)}\}, \mathbf{r}) & \text{sonst} \end{cases} \tag{2.8}$$

wobei $S(\{\mathbf{y}^{(1)}\}, \mathbf{r})$ den S-Metrik-Wert von $\mathbf{y}^{(1)}$ bezüglich des Referenzpunktes $\mathbf{r}$ bezeichnet gemäß Definition 1.11. Der Indikator I_{HD} entspricht dem Hypervolumenbeitrag einer Lösung, allerdings nur innerhalb der betrachteten zwei-elementigen Teilmenge der Population. Der exakte Beitrags einer Lösung innerhalb der gesamten Population wie er im SMS-EMOA verwendet wird, wird hier nicht berechnet. Bei beiden Indikatoren bedeutet ein negativer Werte, dass die erste Argument-Lösung die zweite dominiert. Der binäre Hypervolumenindikator ist in Abbildung 2.6 veranschaulicht.

Der IBEA verwendet eine binäre Turnier-Selektion zur Auswahl der Eltern-Individuen. Als Variationsoperatoren wurden bei den Studien der ZDT- und DTLZ-Funktionen die SBX-Rekombination und die PM-Mutation eingesetzt. Für die Fitnessfunktion

$$F(\mathbf{x}^{(1)}) = \sum_{\mathbf{x}^{(2)} \in P \setminus \{\mathbf{x}^{(1)}\}} -e^{-I(\mathbf{x}^{(2)}, \mathbf{x}^{(1)})/\kappa} \tag{2.9}$$

ist ein positiver Skalierungsfaktor κ zu spezifizieren. Die Fitness ist zu maximieren und bezüglich ihr wird eine $(\mu + \mu)$-Selektion durchgeführt. Die Fitnessfunktion ist kompatibel zur Dominanz-Relation sofern das für die Indikatorfunktion gilt. Es kommt demnach nicht vor, dass eine Lösung $\mathbf{y}^{(1)}$ einer Lösung $\mathbf{y}^{(2)}$ vorgezogen wird, wenn $\mathbf{y}^{(1)}$ von $\mathbf{y}^{(2)}$ dominiert wird.

Eine Variante des IBEA, der adaptive IBEA normalisiert die Indikatorwerte auf den Wertebereich $[-1, 1]$. Der Parameter κ kann dann für alle Probleme gleich gewählt werden und der Referenzpunkt der S-Metrik kann konstant der Vektor $(2, ..., 2)$ sein.

Bei den Experimenten in [ZK04] auf den Funktionen ZDT6, DTLZ2 und DTLZ6 werden die verwendeten Indikatoren I_{ϵ^+}, I_{HD} als Qualitätsmaße der Ergebnismengen eingesetzt. Es zeigt sich eine signifikante Überlegenheit der IBEA-Varianten gegenüber NSGA-II und SPEA2, was aufgrund der Integration der Indikatoren in den Optimierprozess nicht verwunderlich ist.

2.5.4 Hypervolumenselektion in Archivierungsstrategien

Der SMS-EMOA entspricht weitgehend dem algorithmischen Schema, dass Knowles und Corne [KC03] für EMOA formulierten. Sie beschreiben EMOA als Kopplung von Erzeuger von Suchpunkten und einem Archiv, das die besten Lösungen speichert (vergleiche Abschnitt 1.5). Manche EMOA integrieren ein Archiv in den Optimierprozess, wie z. B. der SPEA2 zur Selektion der Eltern. Der SMS-EMOA benötigt innerhalb des Optimierprozesses kein Archiv. Es ist aber natürlich möglich, ein Archiv aller nichtdominierten Lösungen zu verwalten, wie bereits in Abschnitt 2.3 erwähnt.

Algorithmus 2.8 AA_{Reduce}

1: $P_0 \leftarrow \emptyset$ — Archiv nicht-dominierter Lösungen
2: $t \leftarrow 0$
3: **repeat**
4: $\quad \mathbf{o}_t \leftarrow \text{generate}(t)$ — erzeuge eine neue Lösung
5: $\quad P_{t+1} \leftarrow \text{Reduce}(P_t \cup \{\mathbf{o}_t\})$ — aktualisiere Archiv
6: $\quad t \leftarrow t + 1$
7: **until** Abbruchkriterium erfüllt

Der SMS-EMOA ähnelt dem Algorithmen-Schema AA_{Reduce} wenn man die Selektion der Folgepopulation mit der Aktualisierung eines Archivs vergleicht. AA_{Reduce} verwaltet ein größenbeschränktes Archiv von nicht-dominierten Lösungen. Im Gegensatz dazu kann die Population des SMS-EMOA auch dominierte Lösungen enthalten. Eine Archivierungsstrategie, die als in der Reduzierungsfunktion Reduce den Hypervolumenbeitrag (siehe Definition 2.1) verwendet, heißt AA_S. Die neuartige Idee beim SMS-EMOA ist, diese Funktion in den Selektionsoperator eines EMOA und damit direkt in den Optimierungsprozess zu integrieren.

Knowles und Corne zeigen, dass das Archiv der AA_S-Strategie gegen eine Teilmenge der Pareto-Front konvergiert. Konvergenz bedeutet hier, dass sich der Zustand des Archivs, also die Menge der enthaltenen Lösungsvektoren, nicht mehr verändert bei zukünftigen Iterationen. Voraussetzung für dieses Verhalten ist ein endlicher Raum möglicher Lösungsvektoren. Außerdem muss in jeder Iteration jeder Lösungsvektor mit positiver Wahrscheinlichkeit erzeugt werden.

Zunächst wird die Konvergenz bewiesen. Die archivierte Menge verändert sich nur, falls eine echte Verbesserung erzielt wurde, nicht bei Gleichheit. Dazu muss eine nichtdominierte Lösung gefunden werden, die in das Archiv aufgenommen wird, weil es noch nicht voll ist oder diese Lösung mindestens eine andere dominiert. Falls das vorige Archiv zusammen mit der neuen Lösung eine nicht-dominierte Menge bildet, wird die Lösung genau dann aufgenommen, wenn der Austausch mit einer anderen Archivlösung den S-Metrikwert des Archivs erhöht. Aus den obigen Bedingungen folgt, dass

sich das Archiv nur dann verändert, wenn der S-Metrikwert steigt. Der S-Metrikwert kann niemals sinken und daher kann kein schlechterer Zustand des Archivs erneut angenommen werden. Da die Menge aller Zielfunktionsvektoren als endlich vorausgesetzt wurde, liegt für das Archiv eine endliche Zustandsmenge vor. Die Konvergenz folgt, da kein Zustand wiederholt angenommen werden kann. Aus der Konvergenz folgt, dass eine Teilmenge der Pareto-Front erreicht wurde, wie ein Widerspruchsbeweis zeigt. Falls die konvergierte Menge nicht Teilmenge der Pareto-Front ist, dann gibt es einen Lösungsvektor, der ein Archivelement dominiert. Wird diese Lösung generiert – was nach Voraussetzung mit positiver Wahrscheinlichkeit passiert –, dann wird er in das Archiv übernommen und daher lag noch keine Konvergenz vor.

Wegen der Ähnlichkeit zu AA_S gilt die Eigenschaft auch für den SMS-EMOA. Aufgrund des steady-state-Ansatzes wird auch genau eine neue Lösung pro Iteration erzeugt und anschließend eine Selektion auf dieser und der aktuellen Population durchgeführt. Der Unterschied, dass der SMS-EMOA auch dominierte Lösungen in der Population enthält, beeinflusst weder die Konvergenz noch die Folgerung, der Beweis gilt also analog. Diese Eigenschaft wurde bereits in Abschnitt 2.3 erwähnt.

Wie in Abschnitt 1.4 erwähnt, besteht eine Punktmenge mit optimalem S-Metrikwert, der für Mengen einer festen Kardinalität erzielt werden kann, aus Pareto-optimalen Punkten. Knowles und Corne [KC03] zeigen, dass der S-Metrikwert eines mit der AA_S-Strategie verwalteten Archivs gegen ein lokales Optimum der S-Metrik konvergiert. Sie verwenden dabei eine ungewöhnliche Definition eines lokalen Optimums. Es bedeutet für sie, der Austausch genau eines Archivelements mit einem anderen Punkt bewirkt keine Erhöhung des S-Metrikwerts. Der Austausch mehrerer Punkte gleichzeitig könnte also durchaus noch eine Verbesserung erzielen, aber diese Vorgehen ist in der Strategie nicht vorgesehen. Es ist also nicht bewiesen, dass diese lokalen Optima den bestmöglichen S-Metrikwert annehmen, der mit einer festen Anzahl nichtdominierter Lösungen erzielt werden kann. Die konvergierte Punktmenge sollte genauer als lokales Optimum der Strategie, aber nicht als lokales Optimum der S-Metrik bezeichnet werden.

Kapitel 3

Berechnung des Hypervolumens

Das durch eine Punktmenge dominierte Hypervolumen zu berechnen, ist das zentrale Problem des Selektionsoperators des SMS-EMOA. Die Selektion von Individuen erfolgt auf Grundlage ihrer Hypervolumenbeiträge, die als das ausschließlich von dem Individuum dominierte Hypervolumen definiert sind. Mit einem Algorithmus, der den S-Metrikwert einer Menge berechnet, ist der Beitrag berechenbar als die Differenz zwischen dem Hypervolumen der Gesamtmenge und der Menge ohne den betrachteten Punkt[1]. Für die Spezialfälle zwei- und drei-dimensionaler Zielfunktionsräume wurden Algorithmen entwickelt, die die Hypervolumenbeiträge direkt bestimmen, indem Teile des Hypervolumens einzelnen Individuen zugeordnet werden (Abschnitt 3.2.1). Für Räume höherer Dimension liegen keine derartigen Algorithmen vor, sodass Beiträge als Differenz der Hypervolumenwerte zweier Mengen berechnet werden.

Das dominierte Hypervolumen bzw. die S-Metrik ist bereits als Leistungsmaß zur Bewertung der Ergebnisse von EMOA etabliert. Zur Berechnung der S-Metrik entwickelten Zitzler [Zit01] und Fleischer [Fle03b] Algorithmen, welche beide eine worst case Laufzeit von $O(m^{d+1})$ aufweisen (für m Punkte im d-dimensionalen Raum). Für den Algorithmus, der als der schnellere gilt, wurden Heuristiken entwickelt, die ihn zusätzlich beschleunigen. Die Laufzeit ist stark von der Reihenfolge der Punkte bzw. der Reihenfolge der Dimensionen abhängig. Die Heuristiken sortieren die Eingabemenge so, dass der schlechtest mögliche Fall garantiert vermieden wird. Diese bekannten Algorithmen und Heuristiken werden in Abschnitt 3.1 beschrieben.

Die Berechnung des dominierten Hypervolumens kann als Spezialfall eines allgemeineren geometrischen Problems betrachtet werden, welches als *Klee's measure problem (KMP)* bekannt ist. Für das KMP gibt es einen Algorithmus mit einer Laufzeit von $O(m^{d/2} \log m)$ ($d \geq 3$). Weder das KMP noch der Algorithmus wurden bisher in der Forschungsgemeinschaft der evolutionären Algorithmen erwähnt, sodass mit der vorliegenden Diplomarbeit erstmalig die Berechnung des dominierten Hypervolumens mit den Erkenntnissen der Computational Geometry in Beziehung gesetzt wird. Die Übertragung der Hypervolumenberechnung auf das KMP und der beste bekannte Algorithmus werden in Abschnitt 3.2.2 behandelt.

[1] In diesem Kapitel betrachten wir Zielfunktionsvektoren losgelöst von ihrer algorithmischen Erzeugung durch einen EMOA und bezeichnen sie einfach als d-dimensionale Punkte oder Vektoren.

3.1 Bekannte Algorithmen

Der Algorithmus *Hypervolume by Slicing Objectives (HSO)* (Abschnitt 3.1.1) ist ein einfach zu implementierendes, rekursives Verfahren zur Berechnung des Hypervolumens. Studien wurden allerdings zunächst vernachlässigt, da mit dem *LebMeasure*-Algorithmus (Abschnitt 3.1.3) ein Verfahren präsentiert wurde, das deutlich schneller sein sollte. Es stellte sich heraus, dass die Laufzeitkomplexität von LebMeasure falsch analysiert wurde, sodass sich das Forschungsinteresse wieder dem HSO-Algorithmus zuwandte und zusätzlich Heuristiken (Abschnitt 3.1.2) entwickelt wurden, um diesen zu beschleunigen. Als Überblick über den aktuellen Stand der Forschung werden die erwähnten Algorithmen hier vorgestellt. Als Argumente verwenden die Algorithmen eine Eingabemenge aus nicht-dominierten d-dimensionalen Punkten und einen Referenzpunkt, für den vorausgesetzt wird, dass er von allen Punkten dominiert wird.

3.1.1 Hypervolume by Slicing Objectives (HSO)

Zitzler implementierte für die Optimierumgebung „PISA – Platform and Programming Language Independent Interface for Search Algorithms“ [BLTZ03] einen Algorithmus zur Berechnung des Hypervolumens [Zit01]. Knowles beschreibt davon unabhängig den Algorithmus in seiner Dissertation [Kno02]. While et al. [WBBH05, WHBH06] nannten das Verfahren nachträglich *HSO (Hypervolume by Slicing Objectives)* und studierten es erstmals ausführlich. Der HSO-Algorithmus verfolgt das Prinzip von Sweep-Algorithmen. Das dominierte d-dimensionale Hypervolumen wird rekursiv zerteilt, indem die Punktmenge auf eine kleinere Anzahl Dimensionen projiziert wird. Die Punktmenge wird somit dimensionsweise verarbeitet und nicht punktweise.

In einem zwei-dimensionalen Raum kann man den Wert der S-Metrik einer Menge als Summe der Flächeninhalte von Rechtecken berechnen [Kno02]. Die Punkte werden aufsteigend bezüglich der ersten Dimension sortiert und sind dadurch gleichzeitig bezüglich der zweiten absteigend sortiert, da es sich um eine nicht-dominierte Menge handelt. Der S-Metrikwert einer Menge $M = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m)}\}$ bezüglich eines Referenzpunktes $\mathbf{r} = (r_1, r_2)^T$ berechnet sich dann folgendermaßen.

$$S(M, \mathbf{r}) \;=\; (r_1 - y_1^{(1)})(r_2 - y_2^{(1)}) + \sum_{i=2}^{m} (r_1 - y_1^{(i)})(y_2^{(i-1)} - y_2^{(i)}) \tag{3.1}$$

Die Berechnung ist in Abbildung 3.1 (links) veranschaulicht. Der dominierte Bereich[2] wird in Rechtecke zerteilt, die ausgehend von einem Punkt in der ersten Dimension bis zum Referenzpunkt verlaufen und in der zweiten durch den nächsten Nachbarpunkt begrenzt sind. Der Flächeninhalt ist durch Formel 3.1 leicht zu berechnen.

Der HSO-Algorithmus stellt eine Verallgemeinerung des zwei-dimensionalen Verfahrens auf höher dimensionale Räume dar. Es ist ein rekursives Verfahren, dass die einzelnen Komponenten der Punkte dimensionsweise bearbeitet (vgl. Algorithmus 3.1). Abbildung 3.1 (rechts) zeigt durch die verschiedenen Schattierungen, wie der Raum

[2] Das schwach dominierte Hypervolumen wird hier abkürzend als dominiertes Hypervolumen bezeichnet.

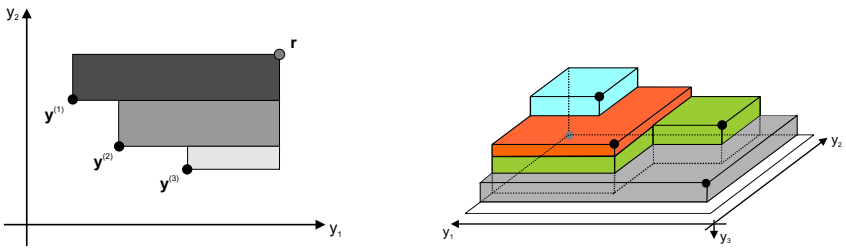

Abbildung 3.1: Veranschaulichung der Berechnung von HSO im zwei- und drei-dimensionalen Zielraum. In der linken Abbildung zeigen die grau markierten Flächen die Zerlegung in Rechtecke. Rechts ist eine andere Menge im drei-dimensionalen Raum dargestellt und eine Zerlegung parallel zur y_1-y_2-Ebene angedeutet.

parallel zur y_1-y_2-Ebene in „Schichten“ unterteilt wird. HSO verwendet als Argumente eine Punktmenge A, einen konstanten Referenzpunkt **r** und die Anzahl zu betrachtender Dimensionen $k \leq d$ der d-dimensionalen Punkte in M. Bei jedem rekursiven Aufruf wird A auf die Menge der *nicht-k-dominierten* Punkte reduziert und k um eins erniedrigt. Die Rekursion bricht bei $k = 2$ ab, da dies dem einfachen zweidimensionalen Sonderfall entspricht.

Definition 3.1 (Nicht-k-dominierte Menge (non-k-dominated set)) *Sei M eine Menge d-dimensionaler Vektoren. Die nicht-k-dominierte Menge $ND_{part}(M,k)$ von M enthält genau die Punkte, die eingeschränkt auf die ersten $k \leq d$ Komponenten des Vektors eine nicht-dominierte Menge bilden.*
$ND_{part}(M,k) = \{y \in M | \nexists z \in M : (z_1, \ldots, z_k) < (y_1, \ldots, y_k)\}$

Das Hypervolumen wird nun berechnet, indem man den dominierten Bereich in „Schichten“ aufteilt und diese in der Dimension rekursiv reduziert. Das $(k-1)$-dimensionale Hypervolumen wird rekursiv bestimmt und multipliziert mit der „Dicke“ der betrachteten Schicht in der ersten Dimension. Man benötigt dazu den Vektor max, der den schlechtesten Wert in k-ter Dimension unter denen in A aufweist (vgl. Algorithmus 3.1). Die Dicke der Schicht in der ersten Dimension ist dann die Differenz zwischen den Komponenten der ersten Dimension von max und dem zuvor schlechtesten Vektor $prev$. Die Punkte, die in erster Dimension einen größeren Wert als max_1 aufweisen, werden anschließend aus der Menge A ausgeschlossen.

Laufzeit

Für die Laufzeitkomplexität des HSO-Algorithmus beschreibt Knowles [Kno02] für eine m-elementige Menge d-dimensionaler Punkte die obere Schranke $O(m^{d+1})$. Es gibt bis zu m Iterationen der While-Schleife pro Aufruf des Algorithmus. Die Funktion ND_{part} erfordert eine Laufzeit von $O(m^2)$. Der HSO-Algorithmus wird m mal auf Ebene k aufgerufen, für $k \geq 3$. Die Rekursion stoppt wie erwähnt bei $k = 2$.

Die Laufzeit wird von While et al. [WHBH06] als Rekursionsgleichung dargestellt. Die Funktion $T(m,d)$ beschreibt die Anzahl 1-dimensionaler Volumenwerte, die innerhalb des HSO berechnet werden. Zur einfacheren Berechnung nimmt man an, dass

Algorithmus 3.1 HSO$(A, \mathbf{r}, k)$

```
 1: vol ← 0                                              Volumen initialisieren
 2: prev_k ← r_k
 3: while (A ≠ ∅) do
 4:    A ←ND_part(A, k − 1)                              Nicht-k-dominierte Menge
 5:    max ← argmax{z_k | z ∈ A}                         Vektor mit schlechtester Komponente k
 6:    if (k < 3) then                                   Sonderfall k = 2
 7:       vol_{k−1} ← max_1                              Ausmaß in erster Dimension
 8:    else
 9:       vol_{k−1} ← HSO(A, r, k − 1)                   Aufruf von HSO mit reduzierter Menge
10:    end if
11:    vol ← vol + vol_{k−1} · |max_1 − prev_1|          Volumenberechnung
12:    prev_1 ← max_1
13:    A ← A \ {z | z_1 ≥ max_1, z ∈ A}
14: end while
15: return vol
```

die Rekursion erst bei $k = 1$ statt bei $k = 2$ abbricht.

$$T(m, 1) = 1 \qquad \text{und} \qquad T(m, d) = \sum_{k=1}^{m} T(k, d-1) \tag{3.2}$$

Die obige Rekursionsgleichung wird mit einem entsprechenden Binomialkoeffizienten gleichgesetzt:

$$T(m, d) = \binom{m+d-2}{d-1}. \tag{3.3}$$

Die Aussage wird in [WHBH06] durch Induktion bewiesen. Es folgt, dass der HSO-Algorithmus eine Laufzeit hat, die polynomiell ist in der Anzahl der Punkte m und exponentiell bezüglich der Dimension des Raums d. Es wird hierbei angenommen, dass $m > d$ gilt.

While et al. [WBBH05] geben ein worst-case Beispiel für den HSO. Der schlechteste Fall tritt ein, wenn während der Ausführung des HSO kein dominierter Punkt auftritt, sodass in jedem Aufruf mit m Punkten gerechnet werden muss. Ein Beispiel ist in Tabelle 3.1 gegeben[3]. Für die dargestellte Punktmenge $M = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(5)}\}$ gilt: $ND_{part}(M, k) = M$, für alle $2 \leq k \leq d$. Der Algorithmus schneidet die Vektorkomponenten von hinten nach vorn ab. Arbeitet man also mit der letzten Dimension, so wird im Folgenden die Menge ohne die d-te Koordinate betrachtet. In unserem Beispiel tritt dabei kein dominierter Punkt auf, die Menge besteht also während der gesamten Bearbeitung aus m Punkten.

Implementierung

While et al. [WHBH06] beschreiben zum besseren Verständnis des Lesers eine ausführlichere Implementierung des HSO als die hier dargestellte. Sie merken an, dass

[3] Das Beispiel ist von While et al. [WBBH05] übernommen und auf den Fall der Minimierung angepasst.

Tabelle 3.1: Worst case Beispiel für HSO mit fünf d-dimensionalen Punkten

	y_1	y_2	$\dots$	y_d	
$\mathbf{y}^{(1)} = ($	5	1	$\dots$	1	$)$
$\mathbf{y}^{(2)} = ($	4	2	$\dots$	2	$)$
$\mathbf{y}^{(3)} = ($	3	3	$\dots$	3	$)$
$\mathbf{y}^{(4)} = ($	2	4	$\dots$	4	$)$
$\mathbf{y}^{(5)} = ($	1	5	$\dots$	5	$)$

der Algorithmus schneller ist, wenn Teilmengen von Punkten direkt verarbeitet werden, ohne sie in zusätzlichen Datenstrukturen zu speichern. Dies entspricht damit der Darstellung von Knowles [Kno02], die in Algorithmus 3.1 wiedergegeben ist. Die Größenordnung der oberen Schranke der Laufzeit der beiden Beschreibungen ist gleich.

3.1.2 Heuristiken zur Beschleunigung des HSO-Algorithmus

Die Punktmenge des worst case Beispiels in Tabelle 3.1 ist nicht prinzipiell schwierig für den HSO. Die Laufzeit ist stark abhängig von der Bearbeitungsreihenfolge der Koordinaten. Auch das worst case Beispiel kann effizient bearbeitet werden, wenn man die erste Dimension mit der d-ten Dimension vertauscht. Hat man diese bearbeitet, so wird die um eine Dimension reduzierte Punktmenge vom ersten Punkt dominiert. Es gilt also $ND_{part}(A, d-1) = \{\mathbf{y}^{(1)}\}$. Die Bearbeitung erfolgt demnach wesentlich schneller. Eine denkbare Strategie zur Aufbrechung der Struktur von worst case Eingaben, ist die zufällige Umsortierung der Eingabemenge. While et al. [WBBH05] entwickelten zwei durchdachte Heuristiken, die eine Umordnung der Dimensionen in eine viel versprechende Reihenfolge vornehmen. Sie werden als Vorverarbeitungsphase des HSO-Algorithmus eingesetzt.

Heuristik: Maximising the number of Dominated Points (MDP)

Die Strategie *MDP (Maximising the number of Dominated Points)* strebt eine Reihenfolge an, bei der möglichst viele Punkte möglichst früh im Bearbeitungsprozess dominiert werden und dadurch im entsprechenden Verarbeitungsschritt wegfallen. Beim HSO-Algorithmus wird die letzte Dimension zuerst bearbeitet. Man berechnet dazu die nicht-k-dominierte Menge mit $k = d - 1$. Die MDP-Heuristik berechnet nun für jede der d Dimensionen jeweils die nicht-dominierte Menge, wobei die entsprechende Dimension nicht betrachtet wird. Die Dimension, bei der die meisten dominierten Punkte auftreten, wenn man sie ignoriert, wird an das Ende des Vektors gestellt. Innerhalb von HSO wird somit die Größe der nicht-k-dominierten Menge für $k = d - 1$ minimiert.

Die Heuristik kann einmalig oder auch iterativ eingesetzt werden. Nach einer einmaligen Anwendung kann man die Dimensionen entsprechend der induzierten Reihenfolge ordnen und den HSO-Algorithmus anwenden. Beim iterativen Vorgehen entfernt man die beste Dimension und wendet MDP erneut an, um die zweit-beste zu bestimmen. While et al. [WBBH05] empfehlen diese mehrfache Anwendung, beispielsweise bis die Anzahl Dimensionen auf vier reduziert ist. Die Laufzeit von MDP beträgt

Tabelle 3.2: Worst case Beispiel für die MDP-Strategie aus fünf drei-dimensionalen Punkten. Links ist die Eingabemenge dargestellt. In der Mitte ist die erste Dimension ignoriert und die grau unterlegten zwei-dimensionalen Punkte werden dominiert. Wird die zweite Dimension ignoriert, so treten nur drei dominierte Punkte auf (rechts). Ignoriert man die dritte Dimension tritt kein dominierter Punkt auf. MDP wählt die erste Dimension als zuerst zu bearbeitende, aber die zweite zu wählen wäre besser.

5	1	1
4	2	5
3	3	4
2	4	3
1	5	2

~~5~~	1	1
~~4~~	2	5
~~3~~	3	4
~~2~~	4	3
~~1~~	5	2

5	~~1~~	1
4	~~2~~	5
3	~~3~~	4
2	~~4~~	3
1	~~5~~	2

$O(m^2d^2)$. Alle Punkte werden in allen Dimensionen miteinander verglichen, was Zeit $O(m^2d)$ einnimmt. Durchgeführt für alle Dimensionen ergibt sich $O(m^2d^2)$. Beim iterativen Vorgehen wird MDP bis zu d mal aufgerufen, daher beträgt die Laufzeit $O(m^2d^3)$.

Es gibt Fälle, in denen im ersten Schritt viele dominierte Punkte auftreten, aber bei den tieferen Verarbeitungsschritten durch die gewählte Reihenfolge sehr viele nicht-dominierte. Das Beispiel[4] in Tabelle 3.2 zeigt einen worst case für diese Strategie. Die linke Tabelle zeigt die vollständige Menge. In den beiden anderen Tabellen sind die Werte der gewählten Dimension durchgestrichen und die jeweils dominierte Menge grau unterlegt. Entfernung der ersten Dimension ergibt vier dominierte Punkte, bei der zweiten sind es drei und bei der dritten keine. Die MDP-Strategie wählt die erste Dimension als erste zu bearbeitende (vgl. Tabelle 3.2, mittig). Der Nachteil ist, dass alle dominierten Punkte sich gegenseitig nicht dominieren. Ihre Weiterverarbeitung ist daher aufwändig. In der rechten Tabelle ist der Fall dargestellt, der sich ergibt, wenn man die zweite Dimension zuerst bearbeitet. Es gibt nur drei dominierte Punkt, von diesen dominiert der beste die anderen und der zweitbeste den dritten. Die Dominanz gilt also auch noch, wenn man nur eine der beiden Dimensionen betrachtet.

Heuristik: Minimising the amount of Worst case Work (MWW)

Eine Strategie, die im beschriebenen worst case der MDP-Strategie besser abschneiden soll, ist die MWW-Heuristik (Minimising the amount of Worst case Work). Diese Strategie simuliert praktisch die ersten Schritte von HSO und berechnet dabei die Anzahl nicht-dominierter Punkte in jeder Scheibe des Raums. Der Bearbeitungsaufwand wird nach oben abgeschätzt durch Gleichung 3.3 des HSO-Algorithmus. Die Werte werden entsprechend Gleichung 3.2 aufsummiert. While et al. [WBBH05] empfehlen auch hier wieder eine iterative Anwendung der Heuristik. Die Laufzeitkomplexität von MWW entspricht der von MDP, also $O(m^2d^2)$ bei einmaliger Anwendung und $O(m^2d^3)$ bei bis zu d Iterationen.

[4] Das Beispiel ist von While et al. [WBBH05] übernommen und auf den Fall der Minimierung angepasst.

Empirische Studien

Die Heuristiken wurden in Verbindung mit dem HSO-Algorithmus auf die Pareto-Fronten einiger DTLZ-Funktionen [DTLZ02] sowie zufällig erzeugte nicht-dominierte Mengen angewendet (vgl. [WBBH05]). Es zeigte sich eine Reduktion der Laufzeit von $25 - 98\%$. Beide Strategien vermeiden garantiert den schlechtesten möglichen Fall. Die durchschnittliche Laufzeit des HSO gekoppelt mit den Strategien ist für fast alle Fälle besser als die Laufzeit des HSO ohne eine vorgeschaltete Heuristik. Die Strategie MWW ergibt in fast allen Testfällen eine bessere Laufzeit als MDP. Auf den DTLZ-Fronten erreicht HSO mit MWW eine nahezu optimale Zeit. Auf den strukturierten Pareto-Fronten der DTLZ-Fronten wurde eine wesentlich stärkere Verbesserung erzielt als auf den zufällig erzeugten Datensätzen.

3.1.3 LebMeasure-Algorithmus

Der *LebMeasure* Algorithmus von Fleischer [Fle03b] unterteilt das dominierte Hypervolumen in Hyperquader. Im Unterschied zu HSO verarbeitet LebMeasure die Eingabemenge punktweise und erzeugt zusätzliche Hilfspunkte um die Struktur der Punktmenge zu vereinfachen.

Zwei-dimensionaler Raum

Der zwei-dimensionale Fall ist in Abbildung 3.2 veranschaulicht. Der Algorithmus verwaltet die Punkte in einer Liste und arbeitet sie der Reihe nach ab. Es wird das dominierte Hypervolumen des ersten Punktes berechnet, das ausschließlich von diesem schwach dominiert wird. Der Bereich ist in beiden Dimensionen jeweils durch seine Nachbarpunkte, also den jeweils nächst schlechteren Wert begrenzt. Falls es keinen schlechteren Wert gibt, wird der Referenzpunkt als Begrenzung herangezogen. Danach wird der Punkt entfernt und man verfährt mit den nächsten analog. Die Hypervolumenwerte ergeben aufaddiert das Hypervolumen der Gesamtmenge. Der zwei-dimensionale Fall ist einfach, weil die zu berechnenden Teilhypervolumina Rechtecke sind.

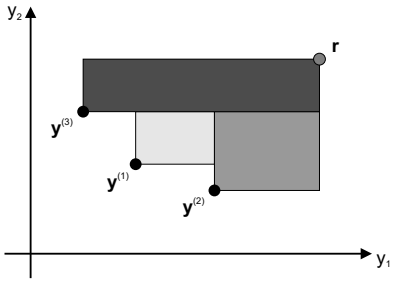

Abbildung 3.2: Ablauf des Algorithmus LebMeasure im zwei-dimensionalen Raum, wenn die Punkte in der Reihenfolge $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{y}^{(3)}$ gegeben sind. Ein graues Rechteck zeigt jeweils den Bereich, der exklusiv vom aktuellen Punkt $\mathbf{y}^{(i)}$ dominiert wird, sobald die Punkte $\mathbf{y}^{(j)}$, mit $j < i$ entfernt sind.

Berechnung der Hypervolumenbeiträge

Das Teilhypervolumen, das für den ersten Punkt berechnet wird, entspricht dem Hypervolumenbeitrag des Punktes. Wie in Abbildung 3.2 deutlich wird, gilt das für die weiteren Punkte nicht. Der erste Punkt ist bereits aus der Menge entfernt, wenn das Teilhypervolumen berechnet wird, das exklusiv vom zweiten Punkt dominiert wird. Somit wird ihm auch der Raum zugeordnet, der vom zweiten und vom ersten Punkt dominiert wird. Die Teilergebnisse des Algorithmus können daher nicht direkt in Hypervolumenbeiträge umgewandelt werden. Um den LebMeasure-Algorithmus im SMS-EMOA einzusetzen, empfehlen wir folgendes Vorgehen, das auch schon von Emmerich et al. [EBN05] und Knowles et al. [KCF03] beschrieben wurde. Der Algorithmus wird mehrfach aufgerufen mit Permutationen der Eingabemenge, so dass jeder Punkt einmal der erste in der Bearbeitungsreihenfolge ist. Der LebMeasure-Algorithmus wird abgebrochen, nachdem die Berechnung des Teilhypervolumens des ersten Punktes abgeschlossen ist. Der berechnete Wert entspricht exakt dem Hypervolumenbeitrag des Punktes.

Ablauf im d-dimensionalen Raum

Im d-dimensionalen Raum stellt das Teilhypervolumen, das von einem Punkt exklusiv abgedeckt wird, im Allgemeinen keinen d-dimensionalen Hyperquader, sondern eine komplexere Form dar. Für die Durchführung von LebMeasure bei drei- und höherdimensionalen Räumen werden daher Hilfspunkte herangezogen. Es wird zunächst wieder das Hypervolumen berechnet, was in jeder Dimension durch die nächst schlechteren Werte begrenzt ist. Das entspricht der Approximation des Hypervolumenbeitrags wie sie im ESP-Algorithmus verwendet wird (vgl. Kapitel 2.5.2). Der bearbeitete Punkt wird aus der Liste entfernt und es werden Hilfspunkte (in Algorithmus 3.2 *spawns* genannt) gesetzt, die gemeinsam das noch nicht bearbeitete Hypervolumen abdecken, das der Punkt exklusiv dominierte. Die neuen Punkte werden an den Anfang der Bearbeitungsreihenfolge gesetzt, sodass mit ihnen analog weiter gearbeitet wird. Das Vorgehen des Algorithmus im drei-dimensionalen Raum ist in Abbildung 3.3 veranschaulicht. Die erste Abbildung zeigt das dominierte Volumen von vier Punkten bezüglich des grau markierten Referenzpunktes $\mathbf{r}$. Die Bearbeitung beginnt mit Punkt $\mathbf{y}^{(1)}$, dessen Hypervolumenbeitrag in der Abbildung 3.3, a) grau markiert ist. In der zweiten Abbildung ist der dominierte Hyperquader des Punktes markiert. Die unausgefüllten Punkte sind die erzeugten Hilfspunkte. In der nächsten Abbildung ist der ursprüngliche Punkt entfernt und die Hilfspunkte dominieren den restlichen Raum entsprechend. In der letzten Abbildung ist die Bearbeitung des von Punkt $\mathbf{y}^{(1)}$ exklusiv dominierten Volumens abgeschlossen und alle Hilfspunkte entfernt.

Das genaue Verfahren ist Algorithmus 3.2 zu entnehmen. Das Verarbeiten der Punkte erfolgt bis die Punktmenge leer ist. Die Punktmenge wird in einer festen Reihenfolge verwaltet. Es wird in jedem Schritt der erste Punkt der Menge verwendet und neue Punkte werden vorn eingefügt, sodass sie als nächste bearbeitet werden. In Zeile 7 werden für den Punkt die nächst schlechteren Werte jeder Dimension bestimmt (nächst größere bei Minimierungsproblemen). Falls es keinen schlechteren Wert gibt, wird der Wert des Referenzpunktes $\mathbf{r}$ verwendet, daher wird vorausgesetzt, dass dieser von allen

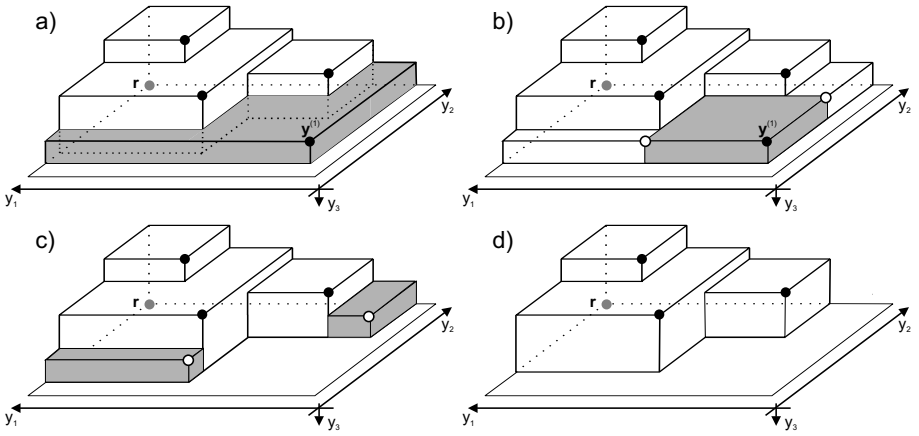

Abbildung 3.3: Ablauf des Algorithmus LebMeasure im drei-dimensionalen Raum. Gezeigt ist die Berechnung des Hypervolumenbeitrags von $\mathbf{y}^{(1)}$, der in Abbildung a) markiert ist. Abbildung b) zeigt den von $\mathbf{y}^{(1)}$ dominierten Hyperquader. Die Hilfspunkte (unausgefüllte Punkte) in Abbildung c) dominieren den restlichen noch nicht abgearbeiteten von $\mathbf{y}^{(1)}$ dominierten Bereich. In der letzten Abbildung ist der Hypervolumenbeitrag von $\mathbf{y}^{(1)}$ vollständig berechnet und der Punkt und alle Hilfspunkte entfernt.

Punkten dominiert wird. Dann wird das Hypervolumen (lopOffVol) des Hyperquaders berechnet, der durch die Nachbarpunkte begrenzt ist. Das Gesamtvolumen ergibt sich als Summe dieser Werte.

Die Prozedur spawnPoints in Algorithmus 3.2 generiert Hilfspunkte, die in einer Liste names spawns gespeichert werden. Für einen Punkt werden mit Hilfe der For-Schleife d Hilfspunkte erzeugt, indem jeweils eine Koordinate auf den nächst schlechteren Wert gesetzt wird und die restlichen übernommen werden. Alle Hilfspunkte werden daher von dem Punkt, aus dem sie erzeugt wurden, dominiert (vgl. Abbildung 3.3, b). Die Hilfspunkte werden der Menge hinzugefügt und der erzeugende Punkt entfernt. In Zeile 12 wird von der Punktmenge die nicht-dominierte Menge bestimmt. Die Prozedur

Algorithmus 3.2 LebMeasure(A, **r**)

```
 1: vol ← 0 ;  list = A                                   Initialisierung
 2: while list ≠ ∅ do
 3:    lopOffVol = 1, 0
 4:    p = getFirst(list)                                 erster Punkt in list
 5:    spawns ← ∅
 6:    for i=0; i<d; i++ do                               für alle Dimensionen
 7:       u_i = min{ q_i − p_i | q_i > p_i , q ∈ {list ∪ r} }   nächst schlechterer Wert
 8:       spawns ← spawnPoint(p, i, u_i)                  erzeuge Hilfspunkt aus p mit u_i statt p_i
 9:       lopOffVol *= u_i − p_i
10:    end for
11:    vol += lopOffVol
12:    list ← NDfilter(list \{p}, spawns, r)              nicht-dom. spawns in list einfügen
13: end while
14: return vol
```

NDfilter entfernt außerdem die Punkte, die eine Koordinate mit dem Referenzpunkt gemeinsam haben. Sie tragen nichts zum Hypervolumenwert bei, weil der Abstand null zum Referenzpunkt als Faktor im Produkt über die Dimensionen auftritt.

Laufzeit

Fleischer [Fle03b] behauptete zunächst, dass die Laufzeit von LebMeasure kubisch in der Anzahl der Punkte ist. Leider ist die Analyse fehlerhaft, da er die Anzahl der generierten Hilfspunkte unterschätzte. While [Whi05] zeigte anhand eines worst case Beispiels, dass die Laufzeit exponentiell in der Dimension des Raums ist. Fleischer argumentierte zuvor, dass die Menge der Hilfspunkte eines „echten“ Punktes niemals d überschreitet. Dies gilt tatsächlich stets für die aktuelle Punktmenge, jedoch nicht über den Gesamtverlauf des Algorithmus. Jeder Hilfspunkt kann höchstens einen neuen Hilfspunkt erzeugen und zwar indem die gegenüber dem Ursprungspunkt verschlechterte Komponente weiter vergrößert wird. Der neue Punkt kann ein nicht-dominierter sein, sobald sein Vorgänger entfernt wurde. Alle anderen von einem Hilfspunkt erzeugten Punkte sind in zwei Komponenten verschlechtert gegenüber dem Ursprungspunkt. Wird der erzeugende Punkt entfernt, so sind die neuen Hilfspunkte immer noch von den anderen älteren Hilfspunkten dominiert, die nur in einer Komponente verschlechtert wurden. Es gibt also pro Punktmenge immer höchstens d nicht-dominierte Hilfspunkte, insgesamt können jedoch exponentiell (bezüglich der Dimension) viele Hilfspunkte erzeugt werden. Fleischer verwechselte also gewissermaßen Zeitkomplexität mit Platzkomplexität.

While [Whi05] zeigt, dass die Anzahl erzeugter Hilfspunkte – und damit die Laufzeit – stark von der Bearbeitungsreihenfolge der Punkte abhängt. Dies kann man auch in Abbildung 3.3 erkennen, denn es gibt eine Reihenfolge, bei der überhaupt keine Hilfspunkte benötigt werden. Das worst case Beispiel des HSO-Algorithmus aus Tabelle 3.1 trifft ebenso auf LebMeasure zu. Durch eine Veränderung der Bearbeitungsreihenfolge der Punkte könnte auch LebMeasure beschleunigt werden. Dies wurde allerdings noch nicht untersucht und ist eine offene Forschungsaufgabe.

3.2 Neue Berechnungsverfahren

3.2.1 Hypervolumenbeiträge in zwei und drei Dimensionen

Für die Fälle zwei-dimensionaler und drei-dimensionaler Zielräume wurden spezielle Algorithmen zur Berechnung der Hypervolumenbeiträge (vgl. Definition 2.1), die der SMS-EMOA benötigt, entworfen. Die beiden Fälle gelten als typisch für reale Anwendungsprobleme, daher ist eine besonders effiziente Behandlung wünschenswert. Außerdem sollten die Algorithmen einfach zu implementieren sein, damit ein Anwender, der nur eine bestimmte Problemdimension bearbeitet, sich nicht in allgemeinere Verfahren einarbeiten muss. Die folgenden Verfahren berechnen die Hypervolumenbeiträge, wie sie im SMS-EMOA zur Selektion benötigt werden, für alle Punkte der gegebenen Menge gleichzeitig. Definiert ist der Beitrag eines Punktes als sein exklusiv dominiertes Hypervolumen. Es kann berechnet werden als Differenz zwischen dem

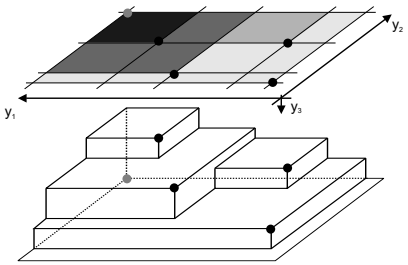

Abbildung 3.4: Veranschaulichung von MHC. Die Punktmenge wird auf die y_1-y_2-Ebene projiziert. Grautöne markieren die verschiedenen maximalen „Höhen“.

Hypervolumen der Gesamtmenge und der Menge bei der der betrachtete Punkt ausgeschlossen ist. Die entwickelten Verfahren ordnen Teile des dominierten Raums den Punkten zu. Mehrfache Aufrufe oder die Berechnung von Differenzen der Werte verschiedener Mengen sind daher nicht notwendig.

Zwei-dimensionaler Raum

Für den zwei-dimensionalen Raum wurde die Berechnung der Beiträge bereits in Gleichung 2.2 vorgestellt. Zur Übersicht wird sie hier nochmals losgelöst vom algorithmischen Überbau dargestellt. Man sortiert die Vektoren einer Menge M bezüglich der Werte einer Dimension, also beispielsweise aufsteigend bezüglich y_1. Die resultierende Punktsequenz ist dadurch automatisch absteigend bezüglich y_2 sortiert, da eine nicht-dominierte Menge vorliegt. Für den j-ten Punkt $\mathbf{y}^{(j)}$, $j \in \{2, \ldots, m-1\}$, in der sortierten Sequenz der Menge M berechnet sich Δ_S wie folgt.

$$\Delta_S(y^{(j)}, M) \;=\; \left(y_1^{(j+1)} - y_1^{(j)} \right) \;\cdot\; \left(y_2^{(j-1)} - y_2^{(j)} \right). \tag{3.4}$$

Die Randpunkte der Menge werden im zwei-dimensionalen Fall in die Folgepopulation des SMS-EMOA übernommen. Es ist daher keine Beitragberechnung erforderlich. Ein Beitrag kann jedoch berechnet werden, indem man den Referenzpunkt als Begrenzung verwendet.

Drei-dimensionaler Raum

Bei einem drei-dimensionalen Raum liegt bereits die Schwierigkeit vor, dass der exklusiv dominierte Raum im Allgemeinen nicht die Form eines Hyperquaders, sondern eine komplexere Form haben kann (vgl. Abbildung 2.1). Für den drei-dimensionalen Fall wurde ein Algorithmus entwickelt, der den dominierten Raum in Quader unterteilt. Ihr Volumen ist einfach durch Multiplikation der Seitenlängen zu berechnen. Die Quader werden dann einzelnen Lösungen zugeordnet, sofern sie zum exklusiv dominierten Raum eines Punktes gehören. Der Hypervolumenbeitrag eines Punktes ergibt sich dann als Summe der Volumina „seiner“ Quader. Der neue Algorithmus *Mesh Hypervolume Contribution (MHC)* ist als Pseudo-Code in Algorithmus 3.3 dargestellt. Abbildung 3.4 veranschaulicht die Funktionsweise an einem Beispiel. Die Grundidee

Algorithmus 3.3 MHC(A, **r**)

```
 1: m ← |A|; A = {s^(1), ..., s^(m)}
 2: (a_1, ..., a_m) ← sortAscending(s_1^(1), ..., s_1^(m))
 3: (b_1, ..., b_m) ← sortAscending(s_2^(1), ..., s_2^(m))
 4: a_{m+1} ← r_1;   b_{m+1} ← r_2
 5: height1[i][j] ← r_3 ;   height2[i][j] ← r_3 ;   for all (i, j) ∈ [m]^2
 6: for all (i, j) ∈ [m]^2 do
 7:    ownerQuantity[i][j] ← 0;  owner ← −1
 8:    for all k = 1; k ≤ m; k++ do
 9:       if s_1^(k) ≤ a_i und s_2^(k) ≤ b_j then
10:          ownerQuantity[i][j]++;   owner ← k          s^(k) dominiert Zelle (i, j)
11:          if s_3^(k) < height1[i][j] then
12:             height2[i][j] ← height1[i][j]; height1[i][j] ← s_3^(k)
13:          else if s_3^(k) < height2[i][j] und s_3^(k) ≠ height1[i][j] then
14:             height2[i][j] ← s_3^(k)
15:          end if
16:       end if
17:    end for
18: end for
19: for all (i, j) ∈ [m]^2 do
20:    if ownerQuantity[i][j] == 1 then          Zelle exklusiv dominiert?
21:       Δ_S(s^(owner), A, r)  += (a_{i+1} − a_i) · (b_{j+1} − b_j) · (height1[i][j]−height2[i][j])
22:    end if
23: end for
24: return Δ_S(s^(1), A, r), ..., Δ_S(s^(m), A, r)
```

ist, die Punkte auf die y_1-y_2-Ebene zu projizieren und die Koordinaten der dritten Dimension als Höhe anzusehen. Für den dominierten Raum wird ein Gitter in der y_1-y_2-Ebene erzeugt, sodass durch jede projizierte Vektorkoordinate eine Kreuzung verläuft.

Der Algorithmus 3.3 besteht aus drei Abschnitten. Bis zur ersten For-Schleife werden Initialisierungen durchgeführt. Die Menge der Eingabepunkte wird in zwei Sequenzen jeweils aufsteigend bezüglich der ersten bzw. der zweiten Dimension gespeichert. An das Ende der Sequenzen wird jeweils die entsprechende Koordinate des Referenzpunktes angehängt. In der For-Schleife ab Zeile 6 wird für jede Zelle des Gitters in der y_1-y_2-Ebene überprüft, welche Punkte die Zelle schwach dominieren. Ein Punkt dominiert eine Zelle schwach, wenn er den besten Punkt der Zelle – also bei Minimierung den Eckpunkt mit kleinsten Koordinaten – bezüglich der ersten beiden Dimensionen schwach dominiert. Die dritte Dimension dient als Höhenattribut der Zelle. Es werden für eine Zelle zwei Werte der dominierenden Punkte gespeichert, nämlich der beste und der zweit-beste Wert bezüglich der dritten Dimension. Diese Höhenwerte wurden zuvor mit der Koordinate des Referenzpunktes initialisiert und werden aktualisiert, sobald ein besserer Punkt gefunden ist. Für jede Zelle wird verwaltet, wie viele Punkte sie dominieren. Außerdem wird bei jeder festgestellten Dominanz der dominierende Punkt als Besitzer (*owner* in Algorithmus 3.3) der Zelle gespeichert. (Bei mehreren dominierenden Punkten wird der Besitzer einfach mehrfach überschrieben, was belanglos ist.) Nachdem die Höhenwerte bestimmt wurden, werden im dritten Abschnitt

des Algorithmus (ab Zeile 19) die Hypervolumenbeiträge zugeordnet. Der Raum zwischen dem besten und dem zweit-besten Wert wird ausschließlich von dem besten Punkt dominiert, sofern keine gleichen Werte auftreten. Das Volumen dieses Raums ist somit Bestandteil des Hypervolumenbeitrags des Punktes. Über die Anzahl dominierender Punkte einer Zelle wird nun festgestellt, ob es genau einen oder mehrere gibt (Zeile 20). Gibt es genau einen dominierenden Punkt, so gilt er als der Besitzer der Zelle und der Raum zwischen bester und zweit-bester Lösung zählt zu seinem Hypervolumenbeitrag. Das Volumen berechnet sich aus der Fläche der Zelle multipliziert mit der Höhendifferenz von bestem und zweitbestem Wert. Der Gesamtwert seines Hypervolumenbeitrags ist somit die Summe „seiner“ Quader.

Laufzeit

Die Laufzeit des Algorithmus ist mit $O(m^3)$ kubisch in der Anzahl der Punkte m und exponentiell in der Dimension des Raums. Die Zeit ergibt sich durch das Traversieren der quadratisch vielen Gitterzellen. Pro Zelle ist die Anzahl der Operationen linear in m, da die gesamte Punktmenge betrachtet werden muss.

3.2.2 Hypervolumen betrachtet als Klee's measure problem

3.2.2.1 Klee's measure problem (KMP)

Klee [Kle77] definierte 1977 das Problem, für eine Menge von m Intervallen auf der Gerade der reellen Zahlen, die Länge der vereinigten Intervalle zu bestimmen, wobei mehrfach enthaltene Bereiche nur einmal gezählt werden. Diese Aufgabe wurde als *Klee's measure problem (KMP)* bekannt. Verallgemeinert auf mehrere Dimensionen werden d Intervalle zu achsen-parallelen Hyperquadern (cuboids) im d-dimensionalen euklidischen Raum, deren vereinigter Rauminhalt oder Hypervolumen (measure, hypervolume) zu berechnen ist[5]. Speziell im zwei-dimensionalen Fall handelt es sich um den Flächeninhalt überlappender Rechtecke und im drei-dimensionalen um das Volumen sich schneidender Quader.

Definition 3.2 (d-dimensionaler Hyperquader (d-dimensional cuboid)) *Ein d-dimensionaler Hyperquader Q ist definiert durch das Kreuzprodukt von d Intervallen $[l_1, u_1] \times \ldots \times [l_d, u_d]$, mit $l_i, u_i \in \mathbb{R}$ für $i \in \{1, \ldots, d\}$.*

In anderer Sichtweise kann man die oberen und unteren Intervallgrenzen jeweils als d-dimensionale Punkte auffassen. Damit ist ein Hyperquader $Q = (\mathbf{l}, \mathbf{u})$ eindeutig bestimmt durch die zwei Eckpunkte $\mathbf{l} = (l_1, \ldots, l_d)$ und $\mathbf{u} = (u_1, \ldots, u_d)$. Für alle Punkte $\mathbf{p}$ des Hyperquaders gilt: $\mathbf{l} \leq \mathbf{p} \leq \mathbf{u}$, gemäß der partiellen Ordnung der Vektoren (siehe Definition 1.1).

Dominiertes Hypervolumen als Klee's measure problem

Das dominierte Hypervolumen, das durch die S-Metrik gemessen wird, kann auf das KMP übertragen werden. Dazu muss das durch eine Punktmenge dominierte Hypervolumen durch eine Menge von Hyperquadern repräsentiert werden. Einem Punkt

[5] In der Computational Geometry sind „area“ oder „measure“ übliche englische Begriffe, „hypervolume“ wird in dieser Forschungsgemeinschaft scheinbar nicht verwendet.

wird der Hyperquader zugewiesen, der durch ihn selbst und den Referenzpunkt der S-Metrik-Berechnung begrenzt ist. Unabhängig von der Dimension des Raums ist ein Hyperquader durch zwei Eckpunkte, die auf derselben Raumdiagonalen liegen, vollständig definiert (vgl. Definition 3.2). Die beiden Eckpunkte legen somit d Intervalle fest und die Problemkodierung des KMP ist erfüllt.

In linearer Zeit kann man also aus einer Punktmenge eine Eingabemenge für Klee's measure problem erzeugen. Die Laufzeit der Algorithmen für das KMP ist von höherer Größenordnung, sodass die Zeit für die Konvertierung der Eingabemenge vernachlässigbar ist. Die Laufzeit der Algorithmen für das KMP ist daher die Laufzeit für die Berechnung des Hypervolumens.

Wird eine nicht-dominierte Menge in eine Menge von Hyperquadern konvertiert, so haben diese eine spezielle Anordnung. Ein Hyperquader ist definiert über seinen zugehörigen nicht-dominierten Punkt und den Referenzpunkt. Daher haben alle Hyperquader den Referenzpunkt als „rechten obere" Ecke, es ist also der größte enthaltene Punkt, gemäß der zugrunde liegenden Ordnung der Vektoren. Außerdem ist kein Hyperquader in einer Menge anderer Hyperquader vollständig enthalten. Dies gilt, weil alle „linken unteren" Eckpunkte nicht-dominierte Punkte sind. Die Frage, inwieweit derzeitige Algorithmen für diesen Spezialfall von Klee's measure problem vereinfacht und beschleunigt werden können, ist eine zukünftige Forschungsaufgabe.

Algorithmen für Klee's measure problem

Klee [Kle77] gibt für das ein-dimensionale KMP einen Algorithmus mit Laufzeit $O(m \log m)$ an und Fredman und Weide [FW78] zeigen eine untere Schranke von $\Omega(m \log m)$. Bentley [Ben77] betrachtet die Erweiterung des Algorithmus auf zwei Dimensionen und zeigte, dass die obere Laufzeitschranke von $O(m \log m)$ auch für diesen Fall gilt. Das ergibt eine Laufzeitkomplexität von $\Theta(m \log m)$ für die ein- und zwei-dimensionalen Versionen des KMP. Für mehr als zwei Dimensionen ist keine spezielle untere Schranke bekannt, so dass man nur $\Omega(m \log m)$ übertragen kann, da das Problem mit zunehmender Dimension nicht einfacher wird. Bentley zeigt durch Verallgemeinerung des Verfahrens für das zwei-dimensionale Problem eine obere Schranke von $O(m^{d-1} \log m)$ für $d > 2$. 1980 verbesserten Leeuwen und Wood [vLW80] die obere Schranke um den Faktor $\log m$ auf $O(m^{d-1})$. Der erstmals 1988 veröffentlichte Algorithmus von Overmars und Yap [OY88, OY91] ist mit einer Laufzeit von $O(m^{d/2} \log m)$ für $d \geq 3$ seither der schnellste bekannte Algorithmus für Klee's measure problem und der Platzbedarf ist mit $O(m)$ sehr klein. Dieses Verfahren wird im Folgenden ausführlich beschrieben.

3.2.2.2 Schnellster Algorithmus für Klee's measure problem

Der Algorithmus von Overmars und Yap [OY91] ist ein Sweep-Verfahren, das ein d-dimensionales Problem in mehrere $(d-1)$-dimensionale Probleme zerlegt. Der $(d-1)$-dimensionale Raum wird durch einen orthogonalen Partitionsbaum repräsentiert. Die Datenstruktur wurde speziell für Klee's measure problem entwickelt und ist auch für andere geometrische Probleme einsetzbar. Zur Berechnung des Hypervolumens wird

innerhalb des Partitionsbaum eine Zerlegung des $(d-1)$-dimensionalen Raums definiert, wobei die Zellen der Partitionierung den Blättern des Baums entsprechen. Einzelne Zellen sind nicht vollständig leer oder vollständig durch Hyperquader abgedeckt, sondern enthalten eine spezielle Gitterstruktur von Hyperquadern. Diese neue Idee ermöglicht eine effiziente Berechnung des überdeckten Hypervolumen innerhalb einer Zelle und die Anzahl benötigter Zellen ist im Vergleich zu früheren Verfahren geringer. Bevor die Details des Algorithmus erläutert werden, führen wir einige Begriffe ein.

Definition 3.3 (i-Grenze, i-Intervall (i-boundary, i-interval)) *Die i-Grenzen $\{l_i, u_i\}$ begrenzen den Hyperquader $Q = (\mathbf{l}, \mathbf{u})$ in der i-ten Dimension. Als i-Intervall $[l_i, u_i]$ bezeichnen wir die Ausdehnung des Hyperquaders in der i-ten Dimension. Es entspricht der Projektion des Hyperquaders auf die i-te Koordinatenachse.*

Definition 3.4 (Scheibe von Grad i (slab at level i)) *Eine Scheibe von Grad i ($i \in \{1, \ldots, d\}$) ist das kartesische Produkt von d Intervallen, wobei i viele endlich sind: $I_1 \times I_2 \times \ldots \times I_i \times \mathbb{R}^{d-i} \in \mathbb{R}^d$, mit $I_1, \ldots, I_i$ Intervalle aus $\mathbb{R}$.*

Definition 3.5 ((Partielle) Überdeckung ((partial) covering)) *Ein d-dimensionaler Hyperquader $Q^{(1)}$ überdeckt einen anderen $Q^{(2)}$ partiell, falls $Q^{(1)}$ das Innere von $Q^{(2)}$ schneidet. Ein d-dimensionaler Hyperquader $Q^{(1)}$ überdeckt einen Hyperquader $Q^{(2)}$ vollständig, falls $Q^{(2)}$ eine Teilmenge von $Q^{(1)}$ ist ($Q^{(2)} \subseteq Q^{(1)}$).*

Aus der Sichtweise der Dominanzrelation können wir sagen: falls $Q^{(2)}$ von $Q^{(1)}$ vollständig überdeckt wird, dann dominiert $Q^{(1)}$ den Hyperquader $Q^{(2)}$ schwach.

Definition 3.6 (i-Säule (i-pile)) *Seien $Q^{(1)}$ und $Q^{(2)}$ zwei Hyperquader. $Q^{(1)}$ heißt i-Säule bezüglich $Q^{(2)}$, falls $Q^{(1)}$ $Q^{(2)}$ partiell überdeckt und für alle j-Intervalle mit $j \neq i$ gilt: Das j-Intervall von $Q^{(2)}$ ist vollständig enthalten im j-Intervall von $Q^{(1)}$.*

Hauptroutine des Algorithmus

Die *räumliche Abtastung (sweep)* ist ein Verfahren, das bei diversen geometrischen Problemen eingesetzt werden kann. Eine $(d-1)$-dimensionale *Hyperebene (sweep hyperplane)* wird entlang einer Koordinatenachse – beispielsweise der y_d-Achse – verschoben. Die Ebene wird diskretisiert über sogenannte *Halte- oder Ereignispunkte (halting points, event points)* bewegt. Bei Klee's measure problem entsprechen die Haltepunkte den Positionen, an denen ein Hyperquader beginnt oder endet, projiziert auf die d-te Koordinatenachse. Konkret hat man eine Menge M von m Hyperquadern gegeben. Die Menge der Haltepunkte $M_d = \{a^{(1)}, \ldots, a^{(m')}\}$ ist die Menge verschiedener d-Grenzen der Hyperquader in M, also Koordinaten der d-ten Dimension der Eckpunkte der Hyperquader. Die Hyperquader, die die Ebene zwischen dem aktuellen und dem nächsten Haltepunkt schneiden, werden als *aktiv* bezeichnet. An jedem Haltepunkt ändert sich also die Aktivität mindestens eines Hyperquaders.

Zur Lösung des d-dimensionalen Problems geht man in der Hauptroutine (vgl. Algorithmus 3.4) folgendermaßen vor. Die Abtastebene des Sweep-Verfahren stoppt an jedem Haltepunkt in M_d und unterteilt dadurch den Raum in m' $(d-1)$-dimensionale

Algorithmus 3.4 OverYap(M, d) Hauptroutine

```
 1: M_d = {a^(1), ..., a^(n')}                 Menge der d-Grenzen von Hyperquadern in M
 2: S ← Partition(M, d − 1)                    konstruiere (d − 1)-dim. orth. Partitionsbaum S
 3: v ← 0
 4: for all i to m' − 1 do                     Sweep entlang d-ter Koordinatenachse
 5:    for all Q ∈ M mit l_d = a^(i) do        Hyperquader Q, die bei a^(i) beginnen
 6:       Insert(Q, root(S))                   füge Hyperquader in Wurzel von S ein
 7:    end for
 8:    v_{d−1} ← (d − 1)-dimensionales Hypervolumen der Hyperquader in S
 9:    v ← v + (a^(i+1) − a^(i)) · v_{d−1}
10:    for all Q ∈ M mit u_d = a^(i+1) do     Hyperquader Q, die bei a^(i+1) enden
11:       Delete(Q, root(S))                   entferne Hyperquader aus Wurzel von S
12:    end for
13: end for
14: return v                                  Hypervolumen n Hyperquader der Menge M
```

„Scheiben“ (vgl. Definition 3.4). Das von Hyperquadern aus M abgedeckte $(d-1)$-dimensionale Volumen einer Scheibe wird jeweils für die aktiven Rechtecke berechnet. Multipliziert mit der Distanz zwischen dem aktuellen und dem nächsten Haltepunkt ergibt sich ein d-dimensionale Hypervolumen. Die Summe dieser Teilwerte ist das Gesamtvolumen der m Hyperquader aus M.

An jedem Haltepunkt werden Hyperquader, die an diesem Punkt beginnen in den orthogonalen Partitionsbaum eingefügt (vgl. Zeile 5 und 6 von Algorithmus 3.4). Nach der Berechnung des $(d-1)$-dimensionalen Hypervolumens werden die Hyperquader, die an dem Haltepunkt enden, aus dem Baum entfernt.

Die Hauptroutine des Algorithmus hat eine Laufzeit von $O(m \log m + mF_{d-1}(m))$. $F_{d-1}(m))$ bezeichnet die Zeit, die für die Aktualisierung des $(d-1)$-dimensionalen orthogonalen Partitionsbaums pro Haltepunkt benötigt wird. Das Einfügen der aktiven und das Löschen der nicht mehr aktiven Hyperquader sowie die Aktualisierung des abgedeckten Hypervolumens erfolgt also innerhalb dieser Zeit.

Konstruktion des orthogonalen Partitionsbaums

Man benötigt eine dynamische Datenstruktur, in der das $(d-1)$-dimensionale Volumen effizient berechnet werden kann. Overmars und Yap [OY91] haben für dieses Problem einen sogenannten *orthogonalen Partitionsbaum (orthogonal partition tree)* als Verallgemeinerung eines k-d-Baums [Ben75] entwickelt. Diese spezielle Datenstruktur kann auch für andere geometrische Probleme wie z. B. den Umfang (perimeter) oder die Konturlinie eingesetzt werden. Allgemeine, nicht-orthogonale Partitionsbäume waren zuvor schon durch Willard [Wil82] und Welzl [Wel88] bekannt.

Definition 3.7 (k-dim. orthogonaler Partitionsbaum (k-dim. orth. partition tree))
Ein k-dimensionaler orthogonaler Partitionsbaum ist ein balancierter binärer Baum, dessen Knoten jeweils eine Region in Form eines k-dimensionalen Hyperquaders zugeordnet ist. Die Wurzel repräsentiert den gesamten k-dimensionalen Raum. Die Re-

gionen von Geschwister-Knoten sind disjunkt und die Vereinigung ihrer Regionen entspricht der Region ihres Eltern-Knotens.

Die Regionen der Knoten einer Ebene bilden jeweils den gesamten k-dimensionalen Raum.

Zur Lösung eines d-dimensionalen KMP wird ein $(d-1)$-dimensionaler orthogonaler Partitionsbaum eingesetzt. Im Folgenden gilt also $k = d - 1$. Es wird eine spezielle Partitionierung des Raums definiert. Diese wird zu Beginn der Hauptroutine (vgl. Algorithmus 3.4) festgelegt und für alle m' Teilprobleme verwendet. Die Struktur des Baums – die Knoten und Kantenbeziehungen – ist also statisch. Der Baum wird als dynamische Datenstruktur bezeichnet, weil er durch Einfügen und Löschen eine variable Anzahl von Hyperquadern verwalten kann. Der k-dimensionale Raum wird in $O(\sqrt{m}^k) = O(m^{k/2})$ Zellen unterteilt, indem man pro Koordinatenachse $O(\sqrt{m})$ Intervalle definiert.

Die zugrunde liegende Partitionierung ist folgendermaßen definiert (vgl. Algorithmus 3.5). Zunächst wird die y_1-Achse in $2\sqrt{m}$ Intervalle unterteilt, wobei jedes höchstens $\sqrt{m}$ 1-Grenzen enthält. Dies ist möglich, da m Hyperquader höchstens $2m$ Grenzen pro Dimension (i-Grenzen) bilden können bzw. genau $2m$, wenn alle Grenzen verschieden sind. Diese Scheiben (vgl. Definition 3.4) werden anschließend bezüglich der zweiten Koordinaten-Achse unterteilt. Dazu betrachtet man für jede Scheibe s jeweils die Menge V_s der Hyperquader, die s partiell abdecken. V_s wird wiederum unterteilt in V_s^1, die Menge der d-dimensionalen Hyperquader, die eine 1-Grenze in der Scheibe s haben, und die Menge V_s^2, für die das nicht zutrifft. Jede Scheibe s wird nun an den 2-Grenzen jedes Hyperquaders in V_s^1 unterteilt. Außerdem teilt man s an jeder $\sqrt{m}$-ten 2-Grenze der Hyperquader aus V_s^2. Für die resultierenden Scheiben s' wird wiederum eine Menge $V_{s'}$ als die Menge der d-dimensionalen Hyperquader betrachtet, die s' partiell abdecken. $V_{s'}^1$ ist die Menge der Hyperquader aus $V_{s'}$, die 1- oder 2-Grenzen im Inneren von s' haben und $V_{s'}^2$ die restliche Menge. Die Scheiben werden bezüglich der dritten Koordinate geteilt an den 3-Grenzen der Elemente in $V_{s'}^1$ und jeder $\sqrt{m}$-ten 3-Grenze der Elemente aus $V_{s'}^2$. So fährt man analog fort, bis man Scheiben von Grad k hat, also Zellen. Allgemein gesprochen unterteilt man die Scheiben von Grad i bezüglich der $(i+1)$-Koordinate. Dazu betrachtet man zwei Teilmengen von Hyperquadern

Algorithmus 3.5 Partition$(M, d-1)$

1: unterteile y_1-Achse in $2\sqrt{m}$ Intervalle — partitioniere y_1-Achse
2: **for all** $i = 2$ to $d-1$ **do** — partitioniere y_i-Achse
3: $V_s = V_s^1 = V_s^2 = \emptyset$
4: **for all** Scheiben s der Ebene $(i-1)$ **do**
5: $V_s \leftarrow \{Q \in M | Q$ überdeckt Scheibe s partiell $\}$
6: $V_s^1 \leftarrow \{Q \in V_s | Q$ hat j-Grenze in Scheibe s, mit $j < i\}$
7: $V_s^2 \leftarrow V_s \setminus V_s^1$
8: unterteile Scheibe s an i-Grenze der $Q \in V_s^1$
9: unterteile Scheibe s an $\sqrt{m}$-ter i-Grenze der $Q \in V_s^2$
10: **end for** — s unterteilt in $O(\sqrt{m})$ Scheiben der Ebene i
11: **end for**

und unterteilt entlang jedes Hyperquaders, der eine j-Grenze ($j < i$) in der Scheibe hat (V_s^1) und entlang jeder $\sqrt{m}$-ten Grenze der restlichen Hyperquader (V_s^2).

Aus der Definition der Partitionierung resultieren Eigenschaften, die sich auf die Struktur des k-dimensionalen orthogonalen Partitionsbaums übertragen. Der Baum enthält die Zellen der Partitionierung in seinen Blättern und den inneren Knoten sind entsprechend größere Regionen zugeordnet.

Satz 3.1 (Eigenschaften des Partitionsbaums) *Für einen d-dimensionalen orthogonalen Partitionsbaum gemäß obiger Partitionierung gelten folgende Eingenschaften.*

1. *Die Partitionierung besteht aus $O(m^{k/2})$ Zellen. Der Baum hat $O(m^{k/2})$ Knoten.*

2. *Jeder k-dimensionale Hyperquader überdeckt partiell höchstens $O(m^{(k-1)/2})$ Zellen, wird also in höchstens $O(m^{(k-1)/2})$ Blättern gespeichert.*

3. *Jede Zelle enthält nur Säulen (vgl. Definition 3.6). Das gilt also insbesondere für die Zellen der Blätter des Baums.*

4. *Jede Zelle wird von höchstens $O(\sqrt{m})$ Hyperquadern partiell abgedeckt. Jedes Blatt enthält somit höchstens $O(\sqrt{m})$ Hyperquader.*

Beweis 3.1 Die Beweise der Aussagen sind in [OY91] zu finden und werden hier nur angerissen.

Zu 1. Es werden $2\sqrt{m}$ Scheiben von Grad 1 festgelegt und jede Scheibe von Grad i in $O(\sqrt{m})$ Scheiben von Grad $i+1$ geteilt, für $i = 1, \ldots k-1$. Es ergeben sich also $O(\sqrt{m}^k) = O(m^{k/2})$ Zellen. Der Baum speichert in jedem Blatt eine Zelle und seine Gesamtzahl an Knoten liegt damit in gleicher Größenordnung.

Zu 2. Überdeckt ein Hyperquader eine Zelle partiell, dann verläuft eine i-Grenze ($i \in \{1, \ldots, k\}$) durch diese Zelle. Es gibt $O(m^{(i-1)/2})$ Scheiben von Grad $i-1$. Jede Scheibe von Grad i wird in den folgenden $(d-i)$ Schritten der Partitionierung in $O(m^{(k-i)/2})$ Zellen unterteilt. Die i-Grenze durchläuft also $O(m^{(i-1)/2} \cdot m^{(k-i)/2}) = O(m^{(k-1)/2})$ Zellen.

Zu 3. Eine Scheibe von Grad i enthält höchstens eine j-Grenze mit $j \leq i$ eines Hyperquaders.

Zu 4. Es können nicht mehr als $O(\sqrt{m})$ Hyperquader eine Zelle abdecken, weil die Zellen dann zu viele Grenzen enthalten würde. Dies würde der Definition der Partitionierung widersprechen. Die Teilmengen V_s^1 haben jeweils höchstens Kardinalität $O(\sqrt{m})$, weil die Intervalle der Scheiben von Grad 1 nach Definition nur $\sqrt{m}$ Grenzen enthalten und das demnach auch für alle weiteren Unterteilungen gilt.

Der orthogonale Partitionsbaum wird nun folgendermaßen aufgebaut. Die Zellen der beschriebenen Partition werden in den Blättern des Baums verwaltet. Ein Eltern-Knoten

Algorithmus 3.6 Insert(Q,δ) und Delete(Q,δ)

```
 1: /* Einfügen/ Löschen eines Hyperquaders Q in Knoten δ des Partitionsbaum S */
 2: /* EINFÜGEN */                                  /* LÖSCHEN */
 3: if δ ist ein Blatt then
 4:    partCov_δ ← partCov_δ ∪ {Q}                  partCov_δ ← partCov_δ \ {Q}
 5:    aktualisiere vol_δ
 6: else if Q deckt reg_δ ab then
 7:    totalCov_δ := totCov_δ + 1                   totalCov_δ := totalCov_δ − 1
 8:                                                 if totalCov_δ > 0 then
 9:    vol_δ = Volumen von reg_δ                       vol_δ = Volumen von reg_δ
10:                                                 else
11:                                                    vol_δ = vol_lson(δ) + vol_rson(δ)
12:                                                 end if
13: else if Q deckt reg_δ partiell ab then
14:    Insert(Q, lson(δ))                           Delete(Q, lson(δ))
15:    Insert(Q, rson(δ))                           Delete(Q, rson(δ))
16:    if totalCov_δ > 0 then
17:       vol_δ = Hypervolumen von reg_δ
18:    else
19:       vol_δ = vol_lson(δ) + vol_rson(δ)
20:    end if
21: end if
```

enthält die vereinigte Region seiner beiden Kinder und damit also die Regionen in seinen Unterbäumen. Der Baum besteht aus k Stufen, die jeweils aus $O(\log m)$ Ebenen bestehen. Die oberste Stufe enthält die $2\sqrt{m}$ Scheiben von Grad 1 auf unterster Ebene, sortiert nach ihren x_1-Koordinaten. Paare benachbarter Scheiben werden jeweils zu Knoten zusammengefasst, so dass ein binärer Teilbaum entsteht. Die tieferen Stufen i enthalten analog die Teilbäume der Scheiben von Grad i.

Einfügen und Löschen von Hyperquadern im orthogonalen Partitionsbaum

Die Prozeduren zum Einfügen und Löschen von Hyperquadern in den bzw. aus dem orthogonalen Partitionsbaum sind in Algorithmus 3.6 dargestellt. Auf der linken Seite steht die Einfüge-Methode Insert. Die Zeilen, in denen sich die Lösch-Methode Delete von ihr unterscheidet, sind rechts hinzugefügt. In der Hauptroutine (Algorithmus 3.4) werden die Methoden für den entsprechenden Hyperquader und den Wurzelknoten des Baums aufgerufen. Es folgen rekursive Aufrufe für die Kinder derjenigen Knoten, deren Region durch den Hyperquader partiell überdeckt ist. Die Rekursion stoppt, wenn die Region eines Knotens durch den Hyperquader vollständig überdeckt ist oder der Knoten ein Blatt ist.

Zur Berechnung des Hypervolumens werden einige Informationen in den Knoten gespeichert. Die Region eines Knotens δ wird als reg_δ bezeichnet. Für Blätter werden in einer Menge $partCov_\delta$ die Hyperquader gespeichert, die die Region von δ schneiden (also partiell oder vollständig überdecken), aber die Region des Eltern-Knoten nur partiell überdecken. Ist δ ein innerer Knoten, so zeigt der Zähler $totalCov_\delta$ an, wie viele Hyperquader die Region des Knotens vollständig abdecken, aber die des Eltern-Knotens nur partiell. Jeder Hyperquader beeinflusst höchstens $O(m^{(k-1)/2} \log m)$ die-

ser *totalCov* Einträge, die entsprechend beim Einfügen oder Löschen entsprechend aktualisiert werden müssen. Diese Anzahl ergibt sich, da ein Hyperquader in höchstens $O(m^{(k-1)/2})$ Blättern gespeichert ist (vgl. Satz 3.1, 3.) und die Tiefe des Baums $O(\log m)$ beträgt. Für jeden Knoten wird das Hypervolumen vol_δ seiner Region reg_δ gespeichert, welches durch Hyperquader überdeckt wird. Falls die Region eines inneren Knoten vollständig überdeckt wird ($totalCov_\delta > 0$), entspricht vol_δ dem Hypervolumen von reg_δ. Ansonten berechnet sich das überdeckte Volumen als Summe der entsprechenden Werte der beiden Kinder. Der Wurzelknoten enthält den gesuchten Wert des überdeckten Hypervolumens, der im Baum gespeicherten Hyperquader. Wie das überdeckte Hypervolumen in einem Blatt berechnet wird, wird nachfolgend beschrieben.

Berechnung der Hypervolumens in einer Zelle

Der k-dimensionale Raum – es sei daran erinnert, dass wir $k = d - 1$ betrachten – ist also in Zellen zerlegt. Einzelne Zellen sind nicht vollständig leer oder vollständig durch Hyperquader abgedeckt, sondern enthalten eine *Gitterstruktur (trellis)* von Hyperquadern. Jeder Hyperquader einer Zelle ist ein Säule (vgl. Satz 3.1, 3.), ein Hyperquader hat also in höchstens einer Dimension eine Grenze innerhalb der Zelle. In den restlichen Dimensionen stimmen die Grenzen des Hyperquaders mit denen der Zelle überein oder der Hyperquader ragt über die Zelle hinaus, so dass er die Zelle bis zu ihren Grenzen überdeckt. Die Grenzen der Hyperquader verlaufen jeweils parallel zu einer Zellwand, da alle Hyperquader achsenparallel sind und die Partitionierung entlang von Hyperquadergrenzen erfolgte. Das Hypervolumen des Gitters innerhalb einer Zelle kann effizient berechnet werden. Sei Z_i die Länge der Zelle in i-ter Dimension. S_i die Summe der Längen der i-Säulen in i-ter Dimension, also das ein-dimensionale Hypervolumen der auf die y_i-Achse projizierten i-Säulen der Zelle. Durch die einander kreuzenden Hyperquader wird ein Gitter gebildet, das aus überdeckten und nicht überdeckten Komponenten besteht. Das Hypervolumen lässt sich berechnen durch das Prinzip des Ein- und Ausschließens.

$$\sum_{1\leq a\leq k} (-1)^{a+1} \left(\sum_{1\leq j_1<\ldots<j_a\leq k} \left(\prod_{1\leq i\leq a} S_{j_i} \prod_{l\in\{j_1,\ldots,j_a\}\wedge l\neq j_i} Z_l \right) \right) \tag{3.5}$$

Zur Veranschaulichung betrachten wir ein drei-dimensionales KMP mit einem zwei-dimensionalen orthogonalen Partitionsbaum. Die Zellen in den Blättern des Baums sind und enthalten zwei-dimensionale Rechtecke. Ein Beispiel ist in Abbildung 3.5 dargestellt. Hier gibt es horizontale und vertikale Rechtecke. Bei horizontalen Rechtecken stimmen die vertikalen Ränder (linker, rechter Rand) und bei vertikalen Rechtecken die horizontalen Ränder (oberer, unterer Rand) mit den Zellwänden überein (oder gehen über diese hinaus). Die Formel 3.5 reduziert sich in diesem Fall auf:

$$Z_1Z_2 - (Z_1 - S_1)(Z_2 - S_2) \quad = \quad Z_2S_1 + Z_1S_2 - S_1S_2. \tag{3.6}$$

Um die Hyperquader einer Zelle zu verwalten, reicht es aus, ein-dimensionale Intervalle zu speichern, die die Ausmaße der Seitenflächen der Hyperquader auf den Zellwänden angeben. Zu diesem Zweck verwendet man jeweils einen Segmentbaum pro

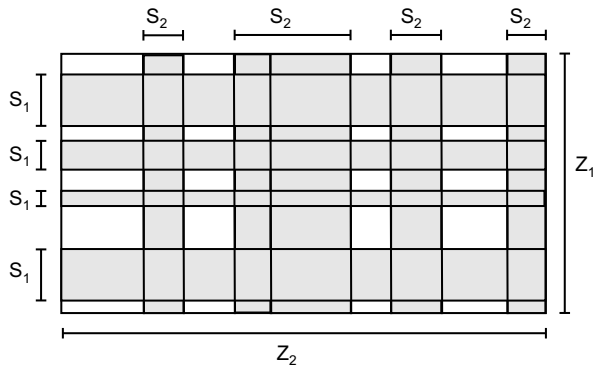

Abbildung 3.5: Zwei-dimensionale Zelle im Blatt des zwei-dimensionalen orthogonalen Partitionsbaums für ein drei-dimensionales KMP. Die enthaltenen Hyperquader sind Säulen und bilden eine Gitterstuktur. Z bezeichnet die Längen der Zelle und die S die Summe der von Hyperquadern überdeckten Intervalle.

Dimension, also k Datenstrukturen $B_1, \ldots, B_k$ pro Zelle. Eine i-Säule wird also entsprechend in den Segmentbaum B_i eingefügt. Das Einfügen oder Löschen von Elementen eines Segmentbaums ist in logarithmischer Zeit möglich. Die Aktualisierung des Flächeninhalts einer Zelle kostet damit $O(\log m)$, da wir die Dimension $k = d-1$ als konstant annehmen. Diese geringe Laufzeit ist nur durch die implizite Repräsentation der Rechtecke möglich.

Übersicht der Laufzeiten

Die Laufzeit des Algorithmus lässt sich analysieren, indem man die Eigenschaften der Partitionierung betrachtet. Wie oben erwähnt, beträgt die Laufzeit des gesamten Algorithmus $O(m \log m + mF_{d-1}(m))$, da man zunächst eine Sortierung der Hyperquadergrenzen bezüglich der d-ten Koordinate durchführt und durch den Sweep-Prozess $m' \leq 2n$ Probleme der Dimension $d-1$ erhält, die man in Zeit $F_{d-1}(m)$ bearbeitet. Es bleibt also die Funktion $F_{d-1}(m)$ zu beschreiben. Die $(d-1)$-dimensionalen Probleme werden mit Hilfe des orthogonalen Partitionsbaums berechnet. Die Aktualisierung eines k-dimensionalen Partitionsbaums kostet $O(m^{(k-1)/2} \log m)$. Für die Lösung des $(d-1)$-dimensionalen Problems ergibt sich also eine Zeit von $F_{d-1}(m) = O(m^{(d-2)/2} \log m)$. Die Berechnung des d-dimensionalen Klee's measure problem für m Hyperquader ist in Zeit $O(m \log m + mF_{d-1}(m)) = O(m^{d/2} \log m)$ möglich.

Reduzierte Platzkomplexität durch Streaming

Ein k-dimensionaler orthogonaler Partitionsbaum benötigt $O(m^{(k+1)/2})$ Speicherplatz. Da stets genau ein $(d-1)$-dimensionaler Baum verwendet wird, hat der Algorithmus einen Speicherplatzbedarf von $O(m^{d/2})$. Durch eine Technik von Edelsbrunner und Overmars [EO85], die *Streaming* genannt wird, kann der Platzbedarf auf $O(m)$ gesenkt werden. Die Idee ist, dass Datenstrukturen nur konstruiert werden, wenn sie für eine Aktualisierung benötigt werden. Es wird immer nur ein kleiner Bereich des d-dimensionalen Raums betrachtet und auf diesem werden Unterteilungen durchgeführt,

so dass man eine Zelle erhält, die zuvor im orthogonalen Partitionsbaum verwaltet wurde. Die Aktualisierungen, die im Laufe des Sweep-Verfahrens auftreten, werden in der Zelle mit Hilfe der Segmentbäume durchgeführt. Nach jedem Schritt wird das $(d-1)$-dimensionale Hypervolumen berechnet. Es wird multipliziert mit dem entsprechenden Wert der d-ten Koordinaten. Anschließend wird die Datenstruktur zerstört und ein neuer Bereich analog betrachtet. Weitere Details dieser Anwendung des Streamings werden von Overmars und Yap [OY91] und grundlegendere Beschreibungen der Streaming Technik von Edelsbrunner und Overmars [EO85] erläutert.

3.3 Übersicht und Anwendung der Algorithmen

Zum Abschluss dieses Kapitels soll eine Übersicht der Algorithmen gegeben und ihr Bezug zum SMS-EMOA erläutert werden. Der SMS-EMOA benötigt innerhalb seines Selektionsoperators die Hypervolumenbeiträge der Individuen. Diese Berechnung ist die aufwändigste Operation des SMS-EMOA – wenn man die Zielfunktionauswertung nicht betrachtet – und bestimmt daher die Größenordnung seiner Laufzeit.

Für zwei-dimensionale Zielräume ist die Berechnung der Beiträge von m Punkten gemäß Formel 3.4 in einer Zeit von $O(m \log m)$ möglich. Die Laufzeit einer Generation des SMS-EMOA (ohne Berücksichtigung der Zielfunktionsauswertung) beträgt für diesen Fall dann $O(\mu \log \mu)$, bei Populationsgröße μ.

Algorithmus 3.3 ist ein Verfahren, dass die Hypervolumenbeiträge für den Spezialfall eines drei-dimensionalen Zielfunktionsraums berechnet. Der Algorithmus hat eine Laufzeit von $O(m^3)$ und für den SMS-EMOA ergibt sich entsprechend die Laufzeit $O(\mu^3)$.

Für die Berechnung der S-Metrik bzw. des dominierten Hypervolumens liegen der HSO-Algorithmus von Zitzler [Zit01], der LebMeasure-Algorithmus von Fleischer [Fle03b] sowie die Algorithmen zur Lösung des KMP vor, unter denen der Algorithmus von Overmars und Yap der schnellste ist. Beim KMP ist das Hypervolumen einer Menge von Hyperquadern zu berechnen. Es beinhaltet damit die Berechnung der Menge von Hyperquadern, die durch eine nicht-dominierte Punktmenge induziert werden als Spezialfall. HSO und LebMeasure haben eine worst case Laufzeit von $O(m^{d+1})$ für m d-dimensionale Punkte. Durch Umsortierung der Eingabemenge kann bei beiden Algorithmen die Laufzeit gesenkt werden. Durchgeführt wurde dies bisher nur für HSO, für den zwei Heuristiken vorliegen, die die Eingabemenge in einer Vorverarbeitungsphase viel versprechend sortieren. Die worst case Komplexität kann dadurch nicht gesenkt werden, allerdings wird der schlechtest mögliche Fall garantiert vermieden und im average case ergibt sich eine deutliche Beschleunigung. Die Heuristiken MDP und MWW haben beide eine Laufzeit von $O(m^2d^2)$ bei einmaliger Anwendung und $O(m^2d^3)$ bei bis zu d iterativen Anwendungen. Der Algorithmus von Overmars und Yap für das KPM hat eine Laufzeitkomplexität von $O(m^{d/2} \log m)$.

Der Hypervolumenbeitrag eines Punktes ist definiert als das Hypervolumen, das er exklusiv dominiert. Der Beitrag kann bestimmt werden, indem man das Hypervolu-

Tabelle 3.3: Übersicht der worst case Laufzeiten der Hypervolumen-Algorithmen

	Laufzeit	Laufzeit des SMS-EMOA
HSO	$O(m^{d+1})$	$O(\mu^{d+2})$
MDP+HSO	$O(m^2d^3) + O(m^{d+1})$	$O(\mu^{d+2})$
MWW+HSO	$O(m^2d^3) + O(m^{d+1})$	$O(\mu^{d+2})$
LebMeasure	$O(m^{d+1})$	$O(\mu^{d+2})$
OverYap	$O(m^{d/2} \log m)$	$O(\mu^{(d/2)+1})$
MHC	$O(m^3)$	$O(\mu^3)$, für $d = 3$
Formel 3.4	$O(m \log m)$	$O(\mu \log \mu)$, für $d = 2$

men der gesamten Punktmenge berechnet, und das Hypervolumen der Menge ohne den betrachteten Punkt subtrahiert. Um die Beiträge einer m-elementigen Menge zu berechnen, sind somit $m + 1$ Aufrufe eines Hypervolumenalgorithmus erforderlich. Die Laufzeit einer Generation des SMS-EMOA ist damit um den Faktor μ größer als die Laufzeit des eingesetzten Hypervolumenalgorithmus (mit $m = \mu$). Bei Einsatz von HSO oder von LebMeasure beträgt die Laufzeit des SMS-EMOA im worst case $O(\mu^{d+2})$. Mit dem Algorithmus von Overmars und Yap ergibt sich eine Laufzeit von $O(\mu^{(d/2)+1} \log \mu)$ für den SMS-EMOA.

Bei zwei- oder drei-dimensionalen Zielräumen sind die speziellen Algorithmen zur Berechnung der Hypervolumenbeiträge empfehlenswert. Der Einsatz des Algorithmus von Overmars und Yap führt für $d = 3$ zu einer etwas niedrigeren Laufzeitkomplexität des SMS-EMOA, ist allerdings aufgrund seiner aufwändigen Datenstrukturen vermutlich erst für Dimensionen $d > 3$ lohnenswert. Für hoch-dimensionale Zielfunktonsräume wird die Anwendung des Algorithmus von Overmars und Yap innerhalb des SMS-EMOA empfohlen. Es ist jedoch möglich, dass der HSO in Verbindung mit einer seiner Heuristiken für viele Eingabemengen effizienter ist. Zukünftige Analysen sollen darüber Aufschluss geben.

Kapitel 4

Empirische Studien

Die Studien in diesem Kapitel geben Aufschluss über die Charakteristika und die Leistungsfähigkeit des SMS-EMOA. Zunächst wird im folgenden Abschnitt 4.1 die Verteilung der Ergebnismengen veranschaulicht. Dazu wird der SMS-EMOA mit Pareto-optimalen Lösungen einer Testfunktion initialisiert und das Augenmerk liegt allein auf der Verteilung der Population auf der Pareto-Front. Die graphische Darstellung der Ergebnisse auf Pareto-Fronten verschiedener Krümmung soll eine Intuition für die Verteilungseigenschaften des SMS-EMOA wecken.

In Abschnitt 4.3 wird ein Benchmarking auf populären akademischen Testfunktionen dargestellt. Hierbei werden die Ergebnisse des SMS-EMOA mit denen etablierten Verfahren verglichen. Die Studien sind denen von Deb et al. [DMM03a, DMM03b] nachempfunden, die zur Präsentation der Algorithmen C-NSGA-II und ϵ-MOEA durchgeführt wurden. Vergleichsstudien auf anderen Algorithmen wurden hier nicht selbst durchgeführt, sondern Ergebnisse aus der Literatur herangezogen. Die Studien werden auf zwei- und drei-kriteriellen Problemen durchgeführt. Diese Fälle erscheinen aktuell als typisch für reale Problemstellungen und sind daher von besonderem Interesse. Die ausgewählten Testprobleme werden zuvor in Abschnitt 4.2 vorgestellt und es wird erläutert, welche Herausforderungen sie an ein Optimierverfahren stellen.

Der SMS-EMOA hat seine Praxistauglichkeit für reale Problemstellungen gezeigt. Bei Anwendungen auf aeronautische Optimierprobleme erzielte der SMS-EMOA sehr gute Ergebnisse. Wir beschränken die Darstellung hier allerdings auf die akademischen Testfunktionen und verweisen für die Praxisanwendungen auf die Studien von Naujoks et al. [NBE05b, NBE05a].

4.1 Verteilung der Ergebnismenge auf der Pareto-Front

Eine Studie soll veranschaulichen, wie der SMS-EMOA Punkte auf der Pareto-Front verteilt. Dazu entwickelten Emmerich et al. [EBN05] eine Familie von Testfunktion namens EBN mit Pareto-Fronten variabler Krümmung. Die Verteilungseigenschaften auf der Pareto-Front sollen hier nicht durch Metriken, sondern nur durch eine optische Darstellung beurteilt werden, um eine Intuition der Struktur des Ergebnismengen des SMS-EMOA zu vermitteln. Die Familie der EBN-Funktionen besteht aus zwei-

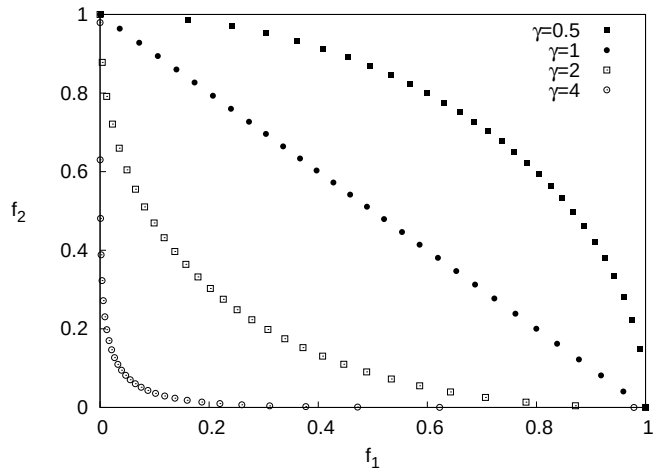

Abbildung 4.1: Veranschaulichung der Verteilung des SMS-EMOA auf Funktionen der EBN-Familie. Stark gekrümmte Regionen sind dichter besetzt.

kriteriellen Zielfunktionen, deren Krümmung durch einen Parameter $\gamma \in \mathbb{R}$ einstellbar ist.

$$f_1(\mathbf{x}) := (\sum_{i=1}^{n} |x_i|)^{\gamma} n^{-\gamma}, \quad f_2(\mathbf{x}) := (\sum_{i=1}^{n} |x_i - 1|)^{\gamma} n^{-\gamma}, \quad \mathbf{x} \in [0,1]^n. \quad (4.1)$$

Der Wert $\gamma = 1$ führt zu einer linearen Pareto-Front, wohingegen $\gamma > 1$ eine konvexe und $\gamma < 1$ eine konkave Pareto-Front erzeugen. Emmerich [Emm05a] zeigt, dass alle Lösungen $\mathbf{x}$ im Intervall $[0,1]^n$ Pareto-optimal sind, also der gesamte Definitionsbereich der Funktion. Die Pareto-Front ist durch die folgende Beziehung der beiden Zielfunktionswerte analytisch exakt gegeben:

$$y_2 = (1 - y_1^{(1/\gamma)})^{\gamma}. \quad (4.2)$$

Die Population des SMS-EMOA wird mit zufälligen Pareto-optimalen Lösungen initialisiert und es interessiert ausschließlich die Verteilung der Punkte. Für die Studie wurde ein Entscheidungsraum mit $n = 20$ Variablen gewählt. Abbildung 4.1 zeigt die Ergebnismengen des SMS-EMOA im Zielfunktionsraum. Die Population ist jeweils konvergiert, sie verändert sich also in nachfolgenden Generationen nicht mehr.

Auf der linearen Pareto-Front ($\gamma = 1$) sind die Lösungen gleichmäßig verteilt, das heißt, sie haben alle den gleichen Abstand von einander. Auf den nicht-linearen Fronten ist die Verteilung nicht gleichmäßig. Die Randlösungen der Menge sind tatsächlich am Rand der Pareto-Front platziert. Die Regionen nahe der beiden Enden der Pareto-Front sind eher spärlich besetzt und die Dichte nimmt zur stärksten Krümmung der Pareto-Front zu. Diese Beobachtungen treffen auf konkave wie konvexe Pareto-Fronten gleichermaßen zu.

Die Verteilung erklärt sich durch die Definition des Hypervolumens bzw. des Hypervolumenbeitrags. Der Hypervolumenbeitrag ist in Regionen, bei denen eine starke Verbesserung bezüglich einer Zielfunktion nur eine minimale Verbesserung bezüglich der

anderen bewirkt, sehr klein. Das Produkt der Abstände zu den Nachbarpunkten wird bei gleicher Summenlänge maximal, wenn die Faktoren gleich groß sind. Daher sind in Regionen, in denen beide Zielfunktionen „ausgeglichen“ verbessert werden können, die Hypervolumenbeiträge größer und es können mehr Punkte platziert werden. Diese Beobachtung erklärt auch, dass die Besiedelungsdichte des NSGA-II, welche auf der Addition von Abständen basiert, eine gleichmäßige Verteilung bewirkt (vgl. Abschnitt 2.5.1). Wie schon in Abschnitt 2.5.4 bei der Betrachtung von AA_S erwähnt, ist nicht bewiesen, dass der S-Metrikwert des SMS-EMOA ein lokales Optimum der S-Metrik darstellt. Jedoch lassen die empirischen Ergebnisse vermuten, dass keine andere Verteilung mit der gleichen Anzahl Pareto-optimaler Punkte einen höheren S-Metrikwert erreicht. Die erzielte Verteilung auf den nicht-linearen Fronten erscheint vorteilhaft gegenüber einer gleichmäßigen Verteilung. Regionen mit Kompromisslösungen, die beide Kriterien relativ gut erfüllen, werden betont und trotzdem wird die gesamte Pareto-Front abgedeckt.

4.2 Zwei- und drei-kriterielle akademische Testfunktionen

Dieser Abschnitt erläutert die akademischen Testfunktionen, die in der Forschungsgemeinschaft verbreitet sind und aktuell verwendet werden, um die Leistungsfähigkeit von EMOA zu vergleichen. Im folgenden Abschnitt werden die zwei-kriteriellen ZDT-Funktionen von Zitzler et al. [ZDT00] und die drei-kriteriellen DTLZ-Funktionen von Deb et al. [DTLZ02] vorgestellt und die Herausforderungen, die sie an ein Optimierverfahren stellen, beschrieben.

ZDT-Funktionen

Deb [Deb99] beschreibt ein allgemeines Schema für die Konstruktion von Testfunktionen. Ein zwei-kriterielles Problem mit n Entscheidungsvariablen wird folgendermaßen definiert.

$$\begin{aligned} \min f_1(\mathbf{x}) &= f_1(x_1, \ldots, x_s), \\ \min f_2(\mathbf{x}) &= g(x_{s+1}, \ldots, x_n) \cdot h(f_1, g) \end{aligned} \tag{4.3}$$

Die Funktion f_1 hängt von den ersten $s < n$ Entscheidungsvariablen ab und g verwendet die restlichen als Argumente. Die Definitionsbereiche der Funktionen sind demnach disjunkt. Die zweite Zielfunktion f_2 basiert auf dem gesamten Argumentvektor. Sie ist definiert als das Produkt von g und einer Funktion h, die wiederum von f_1 und g abhängt.

Durch die Funktionen f_1, g und h können bestimmte Eigenschaft der Pareto-Front erzeugt werden. Die Funktion h steuert die Form der Pareto-Front und generiert bei entsprechender Wahl eine konvexe, konkave oder unzusammenhängende Front. Falls die beiden nachfolgenden Bedingungen erfüllt sind, entspricht die Pareto-Menge den Vektoren, die die Funktion g minimieren. Die Belegung der ersten s Entscheidungsvariablen ist damit beliebig und jeder Wert von f_1 optimal.

1. h ist für feste f_1-Werte schwach monoton steigend im zweiten Argument g.
2. h ist für feste g-Werte streng monoton fallend im ersten Argument f_1.

Die erste Bedingung gewährleistet, dass die Pareto-Front für global minimale Werte von g angenommen wird. Somit kann die Annäherung an die Pareto-Front durch eine multimodale Funktion g erheblich erschwert werden. Die Funktion g testet somit die Fähigkeit eines Algorithmus zur Pareto-Front zu konvergieren. Die zweite Bedingung fordert konfliktäre Zielfunktionen f_1 und f_2, was dazu führt, dass bei festem Wert von g alle Kombinationen von f_1- und f_2-Werten Pareto-optimal sind. Zur Erzeugung von unzusammenhängenden Pareto-Fronten muss die erste Bedingung an die Funktion h relaxiert werden. Da der f_1-Wert unter den oben beschriebenen Bedingungen beliebig ist, kann die f_1-Funktion die Verteilung der Lösungen auf der Pareto-Front bestimmen. Durch eine f_1-Funktion mit einer Tendenz zu bestimmten Werten kann getestet werden, ob ein Algorithmus unter erschwerten Bedingungen eine gleichmäßige Verteilung auf der gesamten Pareto-Front erreicht.

Zitzler et al. [ZDT00] formulieren basierend auf dem obigen Schema sechs Testfunktionen, die nach ihren Entwicklern ZDT1 bis ZDT6 genannt werden. Die Funktion ZDT5 ist auf diskreten Entscheidungsvariablen definiert. Für den SMS-EMOA wurden allerdings bisher nur kontinuierliche Suchräume untersucht, sodass ZDT5 bei der folgenden Präsentation der Funktionen ausgelassen wird. Tabelle 4.1 zeigt eine Übersicht der Funktionen und ihre Eigenschaften werden jeweils kurz umrissen.

Tabelle 4.1: Funktionen ZDT1 bis ZDT4 und ZDT6.
Es gilt $f_2(\mathbf{x}) = g(x_2, \ldots, x_n) \cdot h(f_1, g)$. Die Funktionen f_1 und f_2 sind zu minimieren.

	$f_1(x_1)$ $h(f_1, g)$ $g(x_2, \ldots, x_n)$	n	Pareto-Front, Eigenschaften
ZDT1	x_1 $1 - \sqrt{f_1/g}$ $1 + 9(\ \sum_{i=2}^{n} x_i/(n-1)\ )$	30	konvex
ZDT2	x_1 $1 - (f_1/g)^2$ g wie bei ZDT1	30	konkav
ZDT3	x_1 $1 - \sqrt{(f_1/g)} - (f_1/g) \cdot \sin(10\pi f_1)$ g wie bei ZDT1	30	5 konvexe Teile, nicht zusammenhängend
ZDT4	x_1 h wie bei ZDT1 $1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right)$	10	wie ZDT1, multi-modal
ZDT6	$1 - \exp(-4x_1)\sin^6(6\pi x_1)$ h wie bei ZDT2 $1 + 9(\ \sum_{i=2}^{n} x_i/(n-1)\ )^{0.25}$	10	Teilmenge ZDT2, ungleichmäßige Verteilung

Die Funktion f_1 ist bei allen ZDT-Funktionen nur von der ersten Entscheidungsvariable abhängig und die Funktion g entsprechend von den restlichen. Für die Funktionen ZDT1 bis ZDT3 werden 30 reellwertige Entscheidungsvariablen verwendet, ZDT4 und ZDT6 sind auf 10 Variablen definiert. Alle Funktionen außer ZDT4 ha-

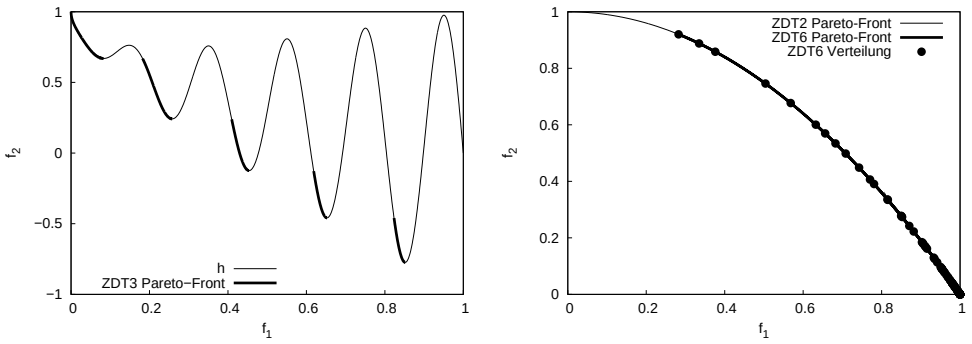

Abbildung 4.2: Pareto-Front von ZDT3 (links) sowie von ZDT2 und ZDT6 (rechts)

ben einen Definitionsbereich von $x_i \in [0, 1], i = 1, \ldots, n$ für alle Entscheidungsvariablen. Bei ZDT4 gilt dies ebenfalls für x_1, aber für die übrigen Variablen gilt $x_i \in [-5, 5], i = 2, \ldots, 10$. Die Pareto-Menge ist für alle ZDT-Funktionen gleich. Sie besteht aus Vektoren für die $x_2 = \ldots = x_n = 0$ gilt. Der Wert von x_1 kann beliebig sein. Für alle Funktionen ist daher der Funktionswert $g(x_2, \ldots, x_n) = 1$ optimal.

ZDT1 und ZDT4 haben die selbe konvexe Pareto-Front. Die Schwierigkeit bei ZDT4 ist, dass es parallel zur Pareto-Front viele lokal optimale Fronten gibt. Dies wird durch die multimodale Funktion g hervorgerufen. Bei 10 Entscheidungsvariablen – also 9 Argumentvariablen – weist sie 21^9 lokal optimale Mengen auf. Die Funktionen ZDT2 und ZDT6 sind konkav, was durch die veränderte Funktion h erreicht wird. Die Pareto-Front von ZDT6 ist eine Teilmenge der Pareto-Front von ZDT2 (vgl. Abbildung 4.2, rechts). Das Besondere an ZDT6 ist die ungleichmäßige Dichte der Punkte im Zielfunktionsraum. Durch die Struktur der Zielfunktion f_1 werden sehr viele Funktionswerte nahe 1 erzeugt. Nur ca. 5% des Definitionsbereichs liefern Zielfunktionswerte, die kleiner als $0{,}5$ sind. Die Abdeckung der vorderen Region der Pareto-Front wird dadurch erschwert. Abbildung 4.2, rechts zeigt durch den Datensatz „ZDT6 Verteilung“ die Verteilung von Pareto-optimalen Zielfunktionswerten, deren Urbilder im Entscheidungsraum gleichmäßig verteilt sind. Gezeigt sind 100 Lösungen mit äquidistanten x_1-Werten. Die Funktion ZDT3 hat eine unzusammenhängende Pareto-Front, die aus fünf leicht konvex gekrümmten Teilen besteht (Abbildung 4.2, links). Durch die enthaltene Sinus-Funktion ist die Funktion h nicht mehr monoton. Wie aus der oben formulierten zweiten Bedingung deutlich wird, sind somit nicht mehr alle Werte von x_1 optimal und es entsteht eine unzusammenhängende Pareto-Front. Für die Pareto-optimalen Punkte gilt $g = 1$ und damit $f_2 = g$. Die in Abbildung 4.2 (links) dargestellte zusammenhängende Kurve entspricht daher genau der Funktion h.

DTLZ-Funktionen

Die DTLZ Testfunktionen wurden nach ihren Entwicklern Deb, Thiele, Laumanns und Zitzler [DTLZ02] benannt. Ausgehend von einer mathematischen Beschreibung der gewünschten Pareto-Front wird eine Testfunktion konstruiert. Das Konzeptionsschema der Funktionen ähnelt stark dem der ZDT-Funktionen und ermöglicht das Erstellen von Testfunktionen mit beliebiger Dimension des Zielfunktionsraums. Die DTLZ-Familie

Tabelle 4.2: Funktionen DTLZ1 bis DTLZ4 für drei-kriterielle Zielfunktionen.
Die $d = 3$ Funktionen f_1, f_2, f_3 sind zu minimieren.

	$f_1(x)$ $f_2(x)$ $f_3(x)$ $g(x_d, \ldots, x_n)$	n ν α	Pareto- Front
DTLZ1	$\frac{1}{2}(1 + g(x_d, \ldots, x_n)) \cdot x_1 x_2$ $\frac{1}{2}(1 + g(x_d, \ldots, x_n)) \cdot x_1(1 - x_2)$ $\frac{1}{2}(1 + g(x_d, \ldots, x_n)) \cdot (1 - x_1)$ $100(\lvert x_M\rvert + \sum_{i=d}^{n}(x_i - 0{,}5)^2 - \cos(20\pi(x_i - 0{,}5)))$	7 5 — 	linear
DTLZ2	$\frac{1}{2}(1 + g(x_d, \ldots, x_n)) \cdot \cos(\frac{\pi}{2}(x_1)^\alpha)\cos(\frac{\pi}{2}(x_2)^\alpha)$ $\frac{1}{2}(1 + g(x_d, \ldots, x_n)) \cdot \cos(\frac{\pi}{2}(x_1)^\alpha)\sin(\frac{\pi}{2}(x_2)^\alpha)$ $\frac{1}{2}(1 + g(x_d, \ldots, x_n)) \cdot \sin(\frac{\pi}{2}(x_1)^\alpha)$ $\sum_{i=d}^{n}(x_i - 0{,}5)^2$	12 10 1 	konvex
DTLZ3	f_1, f_2, f_3 wie bei DTLZ2 g wie bei DTLZ1	12 10 1	wie DTLZ2
DTLZ4	f_1, f_2, f_3 wie bei DTLZ2 mit $\alpha = 100$ g wie bei DTLZ2	12 10 100	wie DTLZ2

umfasst insgesamt neun Funktionen mit verschiedenen Eigenschaften. Wir betrachten in unseren Studien die Funktionen DTLZ1 bis DTLZ4, da für diese die meisten Ergebnisse in der Literatur vorlagen. Wir beschränken uns auf den Fall von drei Kriterien. Die konkreten Bestandteile der drei-kriteriellen DTLZ-Funktionen sind in Tabelle 4.2 dargestellt.

Die Pareto-Front von DTLZ1 ist eine Hyperebene, die durch $\sum_{i=1}^{d}(f_i(\mathbf{x})) = 0{,}5$ beschrieben wird. Die Pareto-Fronten von DTLZ2 bis DTLZ4 sind identisch und entsprechen dem Ausschnitt einer Einheitshyperkugel. Die Pareto-Front kann mit Hilfe der Kugelkoordinaten (θ, φ und Radius 1) für $d = 3$ wie folgt beschrieben werden:

$$\begin{array}{lcll} f_1(\theta,\varphi) = f_2(\theta,\varphi) & = & \cos\theta\cos(\varphi + \frac{\pi}{4}), & \\ f_3(\theta,\varphi) & = & \sin\theta & (4.4) \\ \text{mit} & & 0 \le \theta \le \frac{\pi}{2}, \quad -\frac{\pi}{4} \le \varphi \le \frac{\pi}{4}. & \end{array}$$

Die Pareto-Front ist bei allen DTLZ-Funktionen jeweils auf den ersten Quadranten beschränkt. Die Punkte auf der Hyperkugel bzw. der Hyperebene bilden eine nichtdominierte Menge. Die Testfunktion ist so konzipiert, dass alle weiteren erreichbaren Punkte des Zielfunktionsraums oberhalb der Kugel bzw. Ebene liegen, sodass die oben beschriebenen Mengen die Pareto-Fronten darstellen. Die nicht optimalen Punkte werden erzeugt, indem man die erzeugende Funktion der Pareto-Front skaliert.

Die Pareto-Menge von DTLZ1 ist gegeben durch $x_d, \ldots, x_n = 0{,}5$, die vorderen Entscheidungsvariablen $x_1, \ldots, x_{d-1}$ können beliebige Werte annehmen. Die Pareto-Menge von DTLZ2 bilden die Lösungen $x_d, \ldots, x_n = 0{,}5$, da sie zu $g = 1$ füh-

ren. Die Quadrate der Pareto-optimalen Zielfunktionswerte summieren sich zu Eins: $\sum_{i=1}^{m}(f_i(\mathbf{x}))^2 = 1$. Die Anzahl der Entscheidungsvariablen n ergibt sich aus der Anzahl der Zielfunktionen d und einer Konstante ν als $n = d + \nu - 1$. Für DTLZ1 empfehlen Deb et al. $\nu = 5$, was bei den hier betrachteten drei-kriteriellen Zielfunktionen zu $n = 7$ führt. Für DTLZ2 bis DTLZ4 wird $\nu = 10$ verwendet und führt zu $n = 12$.

Analog zum Schema der ZDT-Funktionen, erschweren Modifikation der Funktion g die Annäherung an die Pareto-Front. So kann eine Funktion mit vielen lokalen Optima dazu führen, dass ein Optimierverfahren solche nicht überwindet, oder die Lösungsdichte kann, analog zu ZDT6, abseits der Pareto-Front erhöht werden. Die gleichmäßige Abdeckung der Pareto-Front wird dadurch zusätzlich erschwert.

Der Vektor wird für eine d-kriterielle Zielfunktion vor dem Objektparameter x_d geteilt. Eine Funktion g verwendet als Argumente die hinteren Variablen. Diese Funktion g wird bei allen Zielfunktionen auf gleiche Weise als Faktor $1/2(1+g)$ eingebunden. Multipliziert wird dies mit einer Komponente, die von den vorderen Entscheidungsvariablen $x_1, \ldots, x_{d-1}$ abhängt und sich über die d Zielfunktionen verändert. Für DTLZ1 sind dies für $d > 3$ Kriterien folgende Komponenten:

$$\begin{aligned} f_1 :&\ \prod_{i=1}^{d-1} x_i \,; \quad f_2 :\ (\prod_{i=1}^{d-2} x_i)(1 - x_{d-1}) \,; \\ f_3 :&\ (\prod_{i=1}^{d-3} x_i)(1 - x_{d-2}) \,; \ \ldots \,; \ f_d :\ (1 - x_1). \end{aligned} \tag{4.5}$$

Für die Zielfunktion f_i wird die hinterste Entscheidungsvariable als Faktor $(1-x_{d-i+1})$ eingebunden und fällt in der folgenden Zielfunktion weg. Für DTLZ2 ergeben sich für $d > 3$ Kriterien folgende Komponenten:

$$\begin{aligned} f_1 :&\ \prod_{i=1}^{d-1} \cos(x_i\pi/2) \,; \quad f_2 :\ \left(\prod_{i=1}^{d-2} \cos(x_i\pi/2)\right) \sin(x_{d-1}\pi/2) \,; \\ f_3 :&\ \left(\prod_{i=1}^{d-3} \cos(x_i\pi/2)\right) \sin(x_{d-2}\pi/2) \,; \ \ldots \,; \ f_d :\ \sin(x_1\pi/2). \end{aligned} \tag{4.6}$$

Hierbei wird für die Zielfunktion f_i die hinterste Entscheidungsvariable als Argument einer Sinusfunktion eingebunden und fällt bei der folgenden Zielfunktion weg. Die DTLZ-Funktionen können mit diesem Schema auf eine beliebige Anzahl von Zielfunktionen skaliert werden.

4.3 Leistungsvergleiche auf Testfunktionen

Mit den im vorigen Abschnitt vorgestellten Testfunktionen wird eine Benchmark-Studie durchgeführt, um die Leistungsfähigkeit des SMS-EMOA mit anderen Verfahren vergleichen zu können. Zunächst werden der Aufbau der Studie und die verwendeten Bewertungsmethoden erläutert. Der darauffolgende Abschnitt präsentiert und bewertet schließlich die vom SMS-EMOA erzeugten Ergebnisse.

4.3.1 Aufbau und Auswertung der Experimente

Design der Experimente

Die Experimente sollen einen Vergleich des SMS-EMOA mit etablierten Verfahren ermöglichen. Die ZDT- und DTLZ-Funktionen sind innerhalb der Forschungsgemeinschaft sehr verbreitet und gelten als anerkannte Benchmark-Funktionen. Ein neues Verfahren muss demnach seine Leistungsfähigkeit auf den Funktionen unter Beweis stellen. Aus den Studien von Deb et al. [DMM03a, DMM03b] liegen unter anderem für die Algorithmen NSGA-II, SPEA2, C-NSGA-II und ϵ-MOEA Testergebnisse vor. Die Experimente für den SMS-EMOA sollten möglichst ähnliche Bedingungen aufweisen, daher wurde die Parametrisierung von Deb et al. [DMM03a] soweit wie möglich übernommen. Für die ZDT- und DTLZ-Funktionen der Studie von Deb et al. wurden jeweils fünf Läufe mit verschiedenen Zufallsgenerator-Saaten durchgeführt, nur für die Funktion DTLZ4 wurden zehn Läufe durchgeführt. Für den SMS-EMOA wurden auf allen Testfunktionen jeweils zehn Läufe durchgeführt. Diese dennoch geringe Anzahl von Testdaten ermöglicht keine zuverlässige Aussage über die, den erzeugten Ergebnismengen zugrunde liegende, Verteilung. Dennoch betrachten wir aufgrund der sehr geringen Standardabweichungen die Werte als aussagekräftig.

Parametrisierung

Für den SMS-EMOA wird die gleiche Anzahl von Funktionsauswertungen durchgeführt wie für die Algorithmen in der Vergleichstudie von Deb et al. [DMM03b]. Bei den ZDT-Funktionen sind dies 20.000 Auswertungen und bei den DTLZ-Funktionen 30.000 Auswertungen, mit Ausnahme von DTLZ3, für die 100.000 Funktionsauswertungen durchgeführt wurden. Die Populationsgröße wurde auf $\mu = 100$ festgesetzt. Die passende Anzahl Generationen ist durch die Populationsgröße und die Anzahl angestrebter Funktionsauswertungen eindeutig festgelegt. Sie beträgt für den SMS-EMOA 19.900 für die ZDT-Funktionen, da die Initialisierung 100 Auswertungen erfordert und in jeder weiteren Generation die Auswertung genau eines Nachkommen durchzuführen ist. Für die DTLZ-Funktionen sind es entsprechend 29.900 bzw. 999.900 Funktionsauswertungen. Der NSGA-II führt beispielsweise 200 Generationen gemäß seines $(100 + 100)$-Selektionsschemas durch, um auf 20.000 Auswertungen zu kommen. Als Variationsoperatoren wurden SBX und PM (vgl. Anhang B) eingesetzt. Die Operatoren wurden identisch zum ϵ-MOEA parametrisiert, nämlich mit $\eta_c = 15$ und $\eta_m = 20$. Die Wahrscheinlichkeit für die Durchführung einer Rekombination wurde als $p_c = 1$ gewählt und die Wahrscheinlichkeit der Mutation einer Entscheidungsvariablen als $p_m = 1/n$, also dem Kehrwert der Länge des Genoms. Bei den SMS-EMOA Varianten mit dynamischer Populationsgröße wurden μ_{init} und μ_{max} als 100 gewählt.

Verwendung von Ergebnissen aus der Literatur

Es wurden keine Studien mit anderen Algorithmen als dem SMS-EMOA durchgeführt, sondern nur Ergebnisse aus der Literatur herangezogen. Daten der Algorithmen liegen nur für bestimmte Parametereinstellungen vor. Es ist möglich, dass ein Algorithmus mit anderen Einstellungen bessere Werte erzielen kann. Die Parameteroptimierung ist

jedoch nicht Bestandteil dieser Arbeit. Wir gehen im Folgenden davon aus, dass die Entwickler der jeweiligen Algorithmen ihre EMOA bestmöglich oder zumindest vielversprechend eingesetzt haben und die gewählten Parametrisierungen daher als repräsentativ für das Verhalten auf den betrachteten Problemen angesehen werden können. Auch für den SMS-EMOA wurde keine Parameteroptimierung angestrebt. Die Parameter der Variationsoperatoren sowie die Populationsgröße wurden nicht in Frage gestellt, sondern vom ϵ-MOEA übernommen, um eine möglichst gute Vergleichbarkeit der Ergebnisse zu gewährleisten. Vergleiche mit den in Abschnitt 2.5 beschriebenen Verfahren, die dem SMS-EMOA ähnlich sind, wären wünschenswert. Außer für den NSGA-II liegen allerdings keine Benchmark-Daten für diese Verfahren vor und es war nicht das Ziel dieser Arbeit, andere Algorithmen zu studieren.

Bewertung durch Metriken

Die Ergebnismengen der Algorithmen wurden mit der S-Metrik und der Konvergenzmetrik bewertet. Allen S-Metrikberechnungen einer Funktion liegt der selbe Referenzpunkt zugrunde. Für die ZDT-Funktionen ist dies der Punkt $(1{,}1; 1{,}1)^T$, für DTLZ1 $(0{,}7; 0{,}7; 0{,}7)^T$ und für die Funktionen DTLZ2 bis DTLZ4 ist es $(1{,}1; 1{,}1; 1{,}1)^T$. Der Referenzpunkt wird von allen Pareto-optimalen Punkten stark dominiert, sodass auch die Randpunkte einen positiven Beitrag zum S-Metrikwert liefern. Wie ihn Abschnitt 1.4 veranschaulicht, beeinflusst die Wahl des Referenzpunktes die, durch die S-Metrik induzierte, Ordnung. Durch einen anderen Referenzpunkt ändern sich also die absoluten S-Metrikwerte und eventuell auch die entsprechende Rangfolge der Algorithmen. Der jeweilige Referenzpunkt erscheint hier sinnvoll gewählt, da er recht nahe an der Pareto-Front liegt und somit keine bestimmte Dimension des Zielraums unverhältnismäßig betont. Durch diese Wahl sind auch bei guten Approximationen Unterschiede in der Qualität von Ergebnismengen zu erkennen.

Bei der Konvergenzmetrik wird eine Referenzmenge Pareto-optimaler Zielfunktionswerte erzeugt. Die Kardinalität der Menge wurde für die ZDT-Funktionen als 1.000 gewählt, genau wie in der Studie von Deb et al. [DMM03a]. Für DTLZ1 verwendeten sie 5.000 Punkte und 8.000 für DTLZ2, wir wählen hier eine Menge aus 5.050 und 7.953 Punkte, die ein regelmäßiges Muster bilden. Für eine gefundene Lösungsmenge wird nicht der Abstand zur wahren Pareto-Front, sondern der Abstand zur Referenzmenge gemessen. Das bedeutet, dass auch ein Pareto-optimaler Punkt einen positiven Abstand von der Referenz-Pareto-Front hat, wenn er nicht genau einem Punkt der Referenzmenge entspricht. Um die Größenordnung diese Fehlers einschätzen zu können, wurden zufällige Mengen von 100 Pareto-optimalen Elementen erzeugt und deren Konvergenzmetrikwert berechnet. Der f_1-Wert ist dabei zufällig gleichverteilt auf der Pareto-Front gewählt. Es wurden für jede Funktion jeweils zehn Datensätze erzeugt, deren Mittelwert und Standardabweichung in den Tabellen 4.3 bis 4.5 als Datensatz „zufällige PF“ (zufällige Teilmenge der Pareto-Front) angegeben sind. Die zufällig erzeugten Pareto-optimalen Mengen haben teilweise einen größeren – und damit schlechteren – Konvergenzmetrikwert als einige der Ergebnismengen der studierten Algorithmen. Die Konvergenzwerte der EMOA sind damit so klein, dass sie die Genauigkeit der Analysemethodik überfordern. Würde für den SMS-EMOA eine größere Referenzmenge gewählt, so würde man sich dem tatsächlichen Konvergenzwert an-

nähern. Die Werte des SMS-EMOA können auf diese Weise nur besser werden, was ihm einen Vorteil gegenüber den Algorithmen der Studie von Deb et al. [DMM03a] verschaffen würde. Es wurde daher auf eine größere Referenzmenge und erhöhte Genauigkeit verzichtet.

Bei der Funktion DTLZ4 tritt mitunter ein starker Diversitätsverlust auf. Das führt dazu, dass die Population nur noch innerhalb einer Ebene verteilt oder sogar auf einen einzigen Punkt kollabiert ist. Läufe, deren finale Population über den gesamten dreidimensionalen Raum verteilt ist, bezeichnen Deb et al. [DMM03a] als erfolgreiche Läufe. Sie berechneten die Metrikenwerte nur für diese Läufe und geben die Anzahl erfolgreicher Läufe jeweils an. Analog werden die Werte auch hier dargestellt. In den Tabellen 4.3 bis 4.5 ist im Falle versagender Läufe die Anzahl erfolgreicher Läufe in Klammern hinzugefügt. Für zwei SMS-EMOA Varianten tritt ein Diversitätsverlust auch bei ZDT2 und ZDT4 auf, der entsprechend angegeben wird.

Übersicht der betrachteten Varianten des SMS-EMOA

Der SMS-EMOA wurde mit verschiedenen Selektionsvarianten eingesetzt, die bereits in Kapitel 2.4 beschrieben wurden. Zur besseren Übrsicht werden für die Varianten kurze, eindeutige Bezeichnungen eingeführt. Der originale SMS-EMOA, der in Algorithmus 2.1 als Basis-Algorithmus vorgestellt wurde, wird hier „SMS 1“ genannt. Der in Abschnitt 2.4.1 präsentierte SMS-EMOA mit der Dominanzzahl als sekundärem Selektionskriterium heißt im Folgenden „SMS 2“. Die ebenfalls in Abschnitt 2.4.1 vorgestellte Abwandlung, die keine nicht-dominierte Sortierung durchführt, wird als „SMS 3“ bezeichnet. „SMS 4“ und „SMS 5“ entsprechen den in Abschnitt 2.4.3 beschriebenen Varianten mit dynamischer Populationsgröße.

Bei der Selektion bezüglich des Hypervolumenbeitrags werden in zwei-dimensionalen Räumen Randpunkte gegenüber den restlichen Punkten einer Front bevorzugt. Für die drei-kriteriellen DTLZ-Funktionen wird der Referenzpunkt der S-Metrik dynamisch verwaltet, wie in Abschnitt 2.4.2 beschrieben. Die Selektion bezüglich der Dominanzzahl führt keine Sonderbehandlung der Randpunkte durch.

4.3.2 Qualität der Ergebnismengen

ZDT-Funktionen

Die Metrikenwerte für die ZDT-Funktionen sind in den Tabellen 4.3 und 4.4 dargestellt, wobei zur Übersicht jeweils Ergebnisse aus den Studien von Deb et al. [DMM03a, DMM03b] eingefügt sind. Die jeweils besten Werte bezüglich einer Metrik sind fett gedruckt. Neben den Mittelwerten der Läufe sind auch die entsprechenden Standardabweichungen angegeben. Zur besseren Lesbarkeit ist zusätzlich der Rang in der durch die jeweilige Metrik induzierte Sortierung angegeben. Die Rangordnung soll dabei nicht bedeutet, dass die Unterschiede unbedingt signifikant sind.

Die Tabellen 4.3 und 4.4 zeigen, dass der SMS-EMOA auf allen ZDT-Funktionen außer ZDT6 einen deutlich besseren (also größeren) S-Metrikwert aufweist als die übrigen etablierten EMOA. Dies ist nicht erstaunlich, da die Optimierprozess des SMS-

Tabelle 4.3: Ergebnisse von EMOA auf den ZDT-Funktionen ZDT1 und ZDT4

		Konvergenz-Metrik			S-Metrik		
	Algorithmus	Mittel	Std. Ab.		Mittel	Std. Ab.	
ZDT1	NSGA-II	0,0005490	6,62e-05	7	0,8701	3,85e-04	9
	C-NSGA-II	0,0006117	7,86e-05	8	0,8713	2,25e-04	6
	SPEA2	0,0010059	12,06e-05	9	0,8708	1,86e-04	7
	ϵ-MOEA	**0,0003955**	1,22e-05	1	0,8702	8,25e-05	8
	SMS 1	0,0004321	1,84e-05	2	**0,8721**	1,20e-05	1
	SMS 2	0,0004400	2,80e-05	3	**0,8721**	1,17e-05	1
	SMS 3	0,0004514	3,23e-05	5	**0,8721**	9,62e-06	1
	SMS 4	0,0004498	3,37e-05	4	**0,8721**	1,33e-05	1
	SMS 5	0,0004712	4,82e-05	6	**0,8721**	1,73e-05	1
ZDT4	NSGA-II	0,0063900	0,0043	8	0,8613	0,00640	6
	C-NSGA-II	0,0061839	0,0744	7	0,8558	0,00301	8
	SPEA2	0,0076928	0,0043	9	0,8609	0,00536	7
	ϵ-MOEA	0,0025906	0,0006	2	0,8509	0,01537	9
	SMS 1	0,0027225	0,0014	3	0,8676	0,00239	2
	SMS 2	0,0036454	0,0025	6	0,8646	0,00512	4
	SMS 3	**0,0022207**	0,0014	1	**0,8682**	0,00214	1
	SMS 4 (5)	0,0029923	0,00074	4	0,8661	0,00365	3
	SMS 5 (5)	0,0034771	0,0016	5	0,8642	0,00628	5
	zufällige PF	*0,0003438*	*2,38e-05*		*0,8654*	*0,00155*	

EMOA genau darauf hin arbeitet. Der SMS-EMOA erreicht außerdem eine sehr gute Annäherung an die Pareto-Front. Bemerkenswert ist, dass über alle Läufe der Studie nur eine einzige dominierte Lösung auftrat, und zwar bei der Funktion ZDT4 und Selektionsvariante 2. Ansonsten stellt die finale Population stets eine nicht-dominierte Menge dar.

Die Ergebnisse des SMS-EMOA bei den verschiedenen Selektionsvarianten unterscheiden sich optisch kaum, daher beschränken sich die Abbildungen 4.3 bis 4.5 auf die Darstellung einer Variante, nämlich des originalen SMS-EMOA („SMS 1“). Gezeigt ist jeweils der Lauf der zehn Läufe, der den Median bezüglich der S-Metrikwerte darstellt. Man sieht die finale Population dieses Laufs. Zusätzlich ist jeweils die Pareto-Front eingezeichnet. Zwischen den einzelnen SMS-EMOA Varianten sind nur minimale Unterschiede zu erkennen. Die einzelnen Varianten sortieren sich oftmals als Gruppe gebündelt zwischen die Werte der anderen EMOA ein.

Auf der Funktion ZDT1 erreichen alle SMS-EMOA Varianten (bei gerundeten Zahlen) den gleichen S-Metrikwert, der besser ist als der aller anderen EMOA. Den besten Konvergenzwert stellt der ϵ-MOEA und die SMS-EMOA Varianten folgen auf den nächsten Plätzen. Alle EMOA erzielen Werte der Konvergenzmetrik, die dem Wert der zufälligen Pareto-optimalen Teilmenge („zufällige PF“) ähnlich sind. Die S-Metrikwerte approximieren den optimalen Wert bis zur zweiten Nachkommastelle, wobei nicht bekannt ist, welcher Wert mit einer Menge von 100 Punkte erreichbar ist.

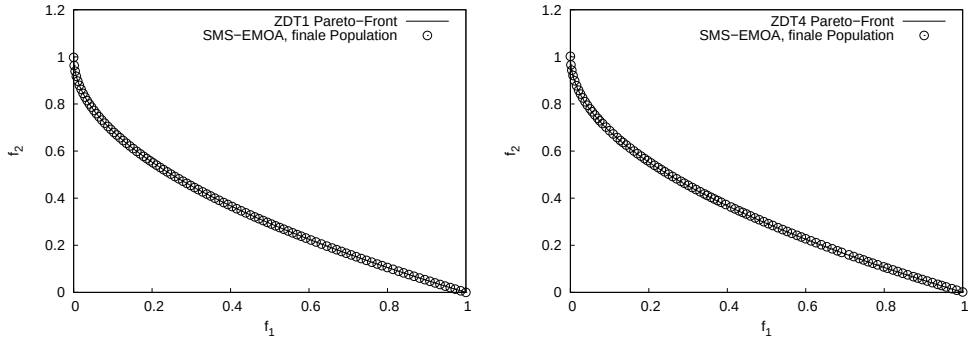

Abbildung 4.3: Finale Population des „Median-Laufs“ des SMS-EMOA auf ZDT1 (links) und ZDT4 (rechts).

In Abbildung 4.3 liegen die Punkte optisch alle auf der Pareto-Front. Die Punkte in der rechten Abbildung zu ZDT4 sind unregelmäßiger verteilt als in der linken Abbildung zu ZDT1. Die Verteilung der Punkte auf der Pareto-Front von ZDT1 lässt erkennen, dass die Punkte in Bereichen starker Krümmung dichter liegen, wie es schon auf den EBN-Funktionen veranschaulicht wurde (vgl. Abschnitt 4.1).

Wie in Abschnitt 4.2 erwähnt, hat die Funktion ZDT4 die gleiche Pareto-Front wie ZDT1, ist aber aufgrund lokaler Optima schwieriger zu optimieren. Die Konvergenzwerte aller Algorithmen sind um etwa eine Zehnerpotenz schlechter als auf der Funktion ZDT1. Auch der S-Metrikwert ist deutlich kleiner. Der SMS-EMOA erreicht bezüglich beider Metriken die besten Werte aller betrachteten Verfahren, und zwar mit „SMS 3“. Den besten S-Metrikwert nach dem SMS-EMOA erzielt der NSGA-II, der allerdings den zweit-schlechtesten Konvergenzmetrikwert erreicht. Der ϵ-MOEA erzielt den zweit-besten Konvergenzmetrikwert, aber auch den schlechtesten Wert bezüglich der S-Metrik. Die zufälligen Pareto-optimalen Mengen sind in Tabelle 4.3 nur einmal dargestellt, da beide Funktionen die gleiche Pareto-Front haben. Der Konvergenz-Metrikwert dieser Mengen ist nur wenig kleiner als der des besten EMOA. Der S-Metrikwert der Mengen ist deutlich schlechter als der der besten EMOA auf der Funktion ZDT1. Bei der Funktion ZDT4 lassen die Konvergenz-Metrikwerte erkennen, dass alle EMOA die lokalen Optima überwunden haben und die Punkte die global-optimale Pareto-Front approximieren. Die kleineren S-Metrikwerte der etablierten Verfahren sind daher auf eine schlechtere Verteilung der Punktmenge zurückzuführen. Die SMS-EMOA Varianten 1, 3 und 4 erzielen als einzige Algorithmen einen besseren S-Metrikwert als die zufällig gleichverteilten Mengen. Bei den Varianten mit dynamischer Populationsgröße zeigt sich bei fünf von zehn Läufen ein Diversitätsverlust, der dazu führt, dass die Population zu einem einzigen Punkt kollabiert. Die Metrikenwerte sind nur für die erfolgreichen Läufe berechnet, wie in der Studie von Deb et al. [DMM03a].

Auf der konkaven Pareto-Front der Funktion ZDT2 erreicht der NSGA-II den besten Konvergenzwert. Die SMS-EMOA Varianten 2 bis 5 sind besser als alle restlichen Verfahren und SMS 1 sortiert sich hinter den C-NSGA-II ein. Bei den Varianten SMS 4 und 5 mit dynamischer Populationsgröße treten, wie auch bei ZDT4, Diversitätsverluste auf und die Metrikenwerte werden nur für die jeweils drei erfolgreichen Läufe

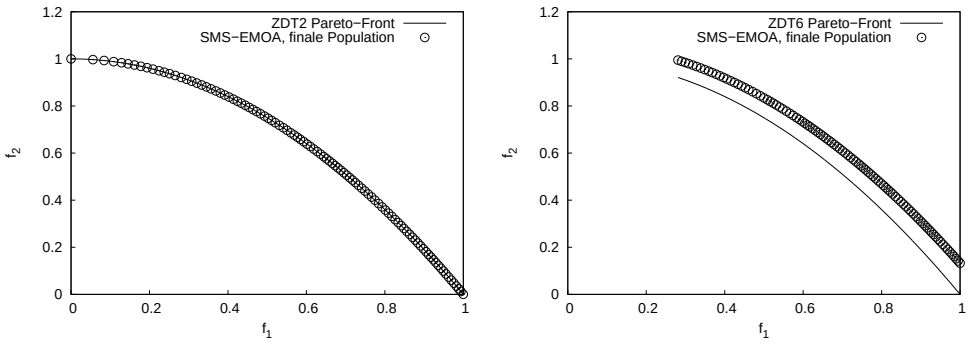

Abbildung 4.4: Finale Population des „Median-Laufs“ des SMS-EMOA auf ZDT2 (links) und ZDT6 (rechts).

berechnet. Bezüglich der S-Metrik weist der NSGA-II den schlechtesten Wert auf und die SMS-EMOA Varianten die besten. Der Wert der Variante 5 ist am größten und der der anderen gleich. Auf der Funktion ZDT6 erzielt der SPEA2 deutlich bessere Werte als alle anderen Verfahren. Wie in Abbildung 4.4 (rechts) zu erkennen ist, erreicht die finale Population des SMS-EMOA nicht die Pareto-Front. Auch die Verteilung der Punkte ist nicht so wie es dem S-Metrikwert zu Gute kommen würde, da die wenig gekrümmte Region hoher f_1-Wert dichter besetzt ist als der stärker gekrümmte Bereich. Die zufälligen Teilmengen der Pareto-Front erzielen einen guten S-Metrikwert, das sie gleichverteilt erzeugt wurden.

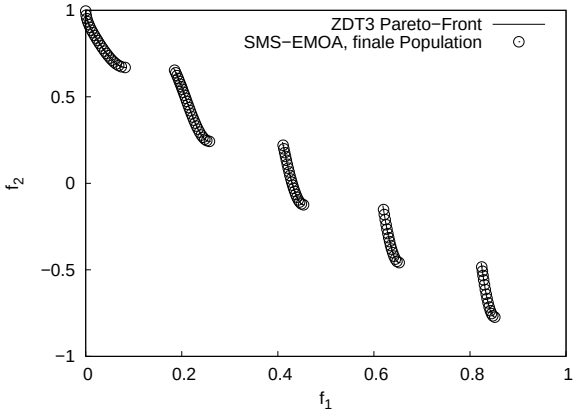

Abbildung 4.5: Finale Population des „Median-Laufes“ des SMS-EMOA auf ZDT3.

Die Ergebnismenge des SMS-EMOA für ZDT3 ist in Abbildung 4.5 dargestellt. Man sieht, dass alle fünf Teile der Pareto-Front abgedeckt sind, wobei die linken Teile dichter besetzt sind. Die SMS-EMOA Varianten 1 und 2 erreichen den gleichen besten S-Metrikwert. Danach folgt der ϵ-MOEA und die restlichen Verfahren. Die übrigen SMS-EMOA Varianten belegen die letzten Plätze. Bezüglich der Konvergenzmetrik erreichen alle SMS-EMOA deutlich bessere Werte als die anderen EMOA. Unter den anderen EMOA erreicht der ϵ-MOEA den besten Konvergenzwert.

Insgesamt scheinen die Funktion ZDT1 und ZDT2 die „einfachsten“ der ZDT-Familie

Tabelle 4.4: Ergebnisse von EMOA auf den ZDT-Funktionen ZDT2, ZDT3 und ZD6

		Konvergenz-Metrik			S-Metrik		
	Algorithmus	Mittel	Std. Ab.		Mittel	Std. Ab.	
ZDT2	NSGA-II	**0,0003785**	1,88e-05	1	0,5372	3,01e-04	9
	C-NSGA-II	0,0004001	1,91e-05	6	0,5374	4,42e-04	7
	SPEA2	0,0008285	1,14e-04	9	0,5374	2,61e-04	7
	ϵ-MOEA	0,0004645	2,47e-05	8	0,5383	6,39e-05	6
	SMS 1	0,0004087	2,91e-05	7	0,5388	2,40e-05	2
	SMS 2	0,0003947	1,79e-05	3	0,5388	1,47e-05	2
	SMS 3	0,0003985	2,07e-05	5	0,5388	3,13e-05	2
	SMS 4 (3)	0,0003922	9,53e-06	2	0,5388	1,61e-05	2
	SMS 5 (3)	0,0003966	4,09e-06	4	**0,5389**	8,78e-06	1
	zufällige PF	*0,0003616*	*2,64e-05*		*0,5296*	*0,00561*	
ZDT6	NSGA-II	0,0789611	0,0067	8	0,3959	0,00894	9
	C-NSGA-II	0,0794067	0,0110	9	0,3990	0,01154	8
	SPEA2	**0,0057358**	0,0009	1	**0,4968**	0,00117	1
	ϵ-MOEA	0,0679280	0,0118	5	0,4112	0,01573	5
	SMS 1	0,0605766	0,0117	3	0,4216	0,01569	3
	SMS 2	0,0597853	0,0077	2	0,4227	0,01041	2
	SMS 3	0,0622466	0,0091	4	0,4194	0,01211	4
	SMS 4	0,0727969	0,0053	7	0,4053	0,00708	7
	SMS 5	0,0716449	0,0056	6	0,4068	0,00742	6
	zufällige PF	*0,0002846*	*2,36e-05*		*0,4984*	*0,00332*	
ZDT3	NSGA-II	0,0023232	13,95e-05	7	1,3285	1,72e-04	4
	C-NSGA-II	0,0023945	12,30e-05	8	1,3277	9,82e-04	5
	SPEA2	0,0026054	15,46e-05	9	1,3276	2,54e-04	6
	ϵ-MOEA	0,0017514	7,45e-05	6	1,3287	1,31e-04	3
	SMS 1	0,0005455	3,95e-05	5	**1,3295**	2,34e-05	1
	SMS 2	0,0005367	3,44e-05	4	**1,3295**	2,89e-05	1
	SMS 3	**0,0005238**	3,02e-05	1	1,3212	0,02641	7
	SMS 4	0,0005362	5,38e-05	3	1,3212	0,02641	7
	SMS 5	0,0005241	4,27e-05	2	1,3128	0,03520	9
	zufällige PF	*0,0004607*	*5,03e-05*		*1,3231*	*0,00680*	

zu sein. Sowohl Konvergenz- als auch S-Metrikwerte sind hier sehr gut, wobei die SMS-EMOA Varianten auf der konvexen ZDT1-Funktion einen deutlichen Vorsprung beim S-Metrikwert zeigen. Bei den anderen Funktionen zeigen sich genau die Schwierigkeiten, die bei der Konzeption der Funktionen intendiert waren. So erschwert die Multimodalität der Funktion ZDT4 deutlich die Konvergenz zur Pareto-Front. Bei ZDT6 wird durch die ungleichmäßige Dichte der Lösungspunkte eine gleichmäßige Abdeckung der Pareto-Front verhindert. Eine Ausnahme bildet hier der SPEA2, der mit Abstand die besten Metrikenwerte erreicht. Allerdings zeigt er auf den anderen Funktionen keine besonderen Leistungen und erreicht mitunter die schlechtesten Werte. Der SMS-EMOA erreicht die besten Approximationen der unzusammenhängenden Pareto-Front von ZDT3, die allen anderen Verfahren deutliche Schwierigkeiten be-

reitet. Auf allen Funktionen außer ZDT6 stellt eine der SMS-EMOA Varianten den besten Wert der S-Metrik. Zudem sind die S-Metrikwerte aller SMS-EMOA besser als die restlichen EMOA auf den Funktionen ZDT1, ZDT2 und ZDT4. Bezüglich der Konvergenz-Metrik erreicht der SMS-EMOA zwei Bestwerte, nämlich auf ZDT3 und ZDT4. Auch auf den übrigen Funktionen sind die Werte des SMS-EMOA auch immer zumindest im Mittelfeld der Verfahren. Insgesamt ist der SMS-EMOA das Verfahren, das auf dem Benchmarking der ZDT-Familie am besten abschneidet durch die hervorragenden S-Metrikwerte und die gute Konvergenz.

DTLZ-Funktionen

Auf den DTLZ-Funktionen zeigt sich, dass der SMS-EMOA die Metrikenwerte der etablierten EMOA deutlich verbessert. Es wurden nur die jeweils besten Werte der Studien von Deb et al. [DMM03a, DMM03b] in Tabelle 4.5 übernommen und es zeigen sich dennoch große Qualitätsunterschiede zu den Ergebnismengen des SMS-EMOA. Leider liegen für die etablierten EMOA kaum S-Metrikwerte vor. Die fehlenden Daten sind in Tabelle 4.5 durch die Abkürzung „n. b.“ (nicht bekannt) gekennzeichnet. Die Spalten enthalten auch hier wieder den Mittelwert, die Standardabweichung und als Orientierungshilfe den Rang in der induzierten Sortierung.

Den besten Wert der Konvergenz-Metrik bei DTLZ1 erreicht der ϵ-MOEA, der sogar den Wert der zufällig gewählten Pareto-optimalen Punkte unterbietet. Er weist allerdings den schlechtesten der dargestellten S-Metrikwerte auf. Den besten S-Metrikwert der Studien von Deb et al. liefert SPEA2. Der Wert ist allerdings noch deutlich schlechter als die Werte der SMS-EMOA Varianten. Abbildung 4.6 zeigt die finale Population des originalen SMS-EMOA. Es ist wie auf den ZDT-Funktionen der Lauf gewählt, der den Medianwert bezüglich der S-Metrik darstellt. Die Punkte sind gleichmäßig über die gesamte Pareto-Front verteilt, wobei einige Punkte exakt die Ränder der Pareto-Front abdecken. Diese gleichmäßige Verteilung korrespondiert mit der Beobachtung, dass auf der linearen ($\gamma = 1$) EBN-Funktion (vgl. Abschnitt 4.1) alle Punkte den gleichen Abstand haben. Die Seitenansicht der Population in der rechten Abbildung zeigt, dass – zumindest optisch – alle Punkte genau auf der Pareto-Front liegen. Hier wie im Folgenden sind nur die Ergebnisse des originalen SMS-EMOA („SMS 1“) abgebildet, da kein Unterschied der Selektionsvarianten zu erkennen ist.

Bei der Funktion DTLZ2 (Abbildung 4.7) ist für die Ergebnismengen des SMS-EMOA ein deutliches Muster zu erkennen. Die drei Eckenpunkte des Kugelausschnitts sind exakt mit Punkten besetzt. Daneben ist in jeder Dimension eine Lücke vorhanden. Neben der Lücke ist der Rand der Pareto-Front dicht mit Punkten besetzt. Im inneren Bereich der Pareto-Front sind die vorhandenen Punkte gleichmäßig verteilt. Dieses Phänomen der Lücke erklärt sich durch die Definition der Hypervolumenbeiträge. Die Punkte am Rand der Pareto-Front – also auf den Ebenen durch jeweils zwei Koordinatenachsen – erhalten einen Hypervolumenbeitrag, der stark von der Wahl des Referenzpunktes abhängt. Wir wählen den Referenzpunkt immer als den Vektor der maximalen Werte zu denen eine Eins addiert wird (vgl. Abschnitt 2.4.2). Die Randpunkte erhalten somit einen tendenziell höheren Beitrag als innere Punkte. Da Nachbarpunk-

Tabelle 4.5: Ergebnisse von EMOA auf den DTLZ-Funktionen

		Konvergenz-Metrik			S-Metrik		
	Algorithmus	Mittel	Std. Abw.		Mittel	Std. Abw.	
DTLZ1	SPEA2	0,003338	3,54e-02	7	0,315981	6,98e-04	6
	ϵ-MOEA	**0,002450**	9,52e-05	1	0,298487	n. b.	
	SMS 1	0,002966	3,42e-04	3	0,316868	1,65e-04	3
	SMS 2	0,002981	1,98e-04	4	0,316875	1,11e-04	2
	SMS 3	0,002871	1,17e-04	2	**0,316902**	7,04e-05	1
	SMS 4	0,003068	6,55e-04	5	0,316855	2,68e-04	4
	SMS 5	0,003094	6,38e-04	6	0,316845	2,94e-04	5
	zufällige PF	*0,002531*	*5,53e-05*		*0,308870*	*1,54e-03*	
DTLZ2	C-NSGA-II	0,009860	8,8e-04	6	n. b.	n. b.	
	SMS 1	0,006428	2,16e-04	2	**0,757988**	3,94e-05	1
	SMS 2	0,006429	3,52e-04	3	0,757943	4,42e-05	5
	SMS 3	0,006458	3,34e-04	4	0,757965	5,32e-05	4
	SMS 4	0,006678	3,50e-04	5	0,757985	5,30e-05	2
	SMS 5	**0,006368**	4,00e-04	1	0,757970	3,08e-05	3
	zufällige PF	*0,006428*	*2,61e-04*	6	*0,691109*	*8,62e-03*	
DTLZ3	ϵ-MOEA	0,012229	2,23e-03	6	n. b.	n. b.	
	SMS 1	**0,006679**	2,63e-04	1	0,756251	1,27e-03	2
	SMS 2	0,007070	5,71e-04	5	0,754958	1,68e-03	5
	SMS 3	0,006727	3,84e-04	2	0,755778	1,28e-03	3
	SMS 4	0,006884	5,94e-04	4	0,755074	1,62e-03	4
	SMS 5	0,006758	4,05e-04	3	**0,756344**	9,00e-04	1
DTLZ4	ϵ-MOEA (6)	0,009776	2,0e-04	6	n. b.	n. b.	
	SMS 1 (5)	0,006550	4,87e-04	3	0,757977	4,41e-05	2
	SMS 2 (5)	0,006537	2,43e-04	2	0,757933	2,70e-05	3
	SMS 3 (4)	0,006579	3,56e-05	4	**0,757999**	7,86e-05	1
	SMS 4 (3)	0,006946	5,64e-04	5	0,757259	1,33e-03	5
	SMS 5 (2)	**0,006246**	1,30e-04	1	0,757900	4,71e-07	4

te im inneren Bereich jeweils ihren exklusiv dominierten Raum begrenzen, kann im inneren Bereich der Pareto-Front nur eine begrenzte Anzahl Punkte platziert werden. Der Abstand zum Rand der Pareto-Front entsteht dadurch, dass dieser Bereich nicht so stark gekrümmt ist (vgl. Abbildung 4.7, rechts). Wie schon bei den EBN-Funktionen veranschaulicht, sind die Bereiche starker Krümmung dichter besetzt. Die Seitenansicht (rechts) der Achtelkugel veranschaulicht auch hier, dass die Punkte genau auf der Pareto-Front liegen. Die Konvergenzwerte des SMS-EMOA sind so klein, dass der beste sogar den Wert der zufälligen Pareto-optimalen Punkte unterbietet. Der S-Metrikwert ist deutlich größer als der der zufällig gleichverteilten Punkte. Daten anderer EMOA liegen leider nicht vor.

Die Pareto-Fronten von DTLZ3 und DTLZ4 sind identisch zu der von DTLZ2. Auf die graphische Darstellung der Ergebnisse wird verzichtet, da sie keine Unterschiede zwischen diesen Funktionen und den SMS-EMOA Varianten erkennen lässt. Die Me-

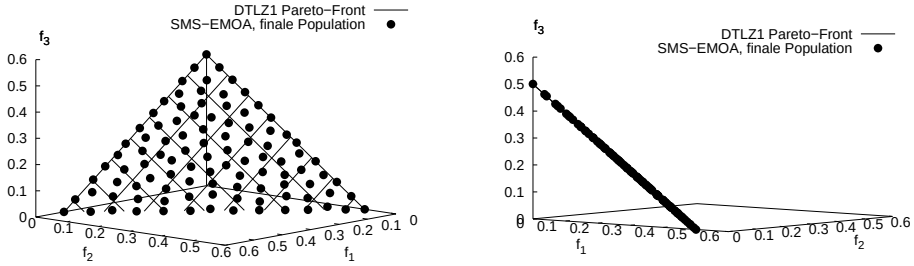

Abbildung 4.6: Finale Population des „Median-Laufes“ des SMS-EMOA auf DTLZ1

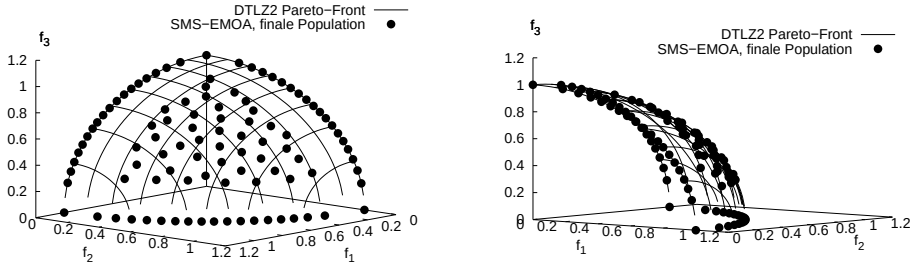

Abbildung 4.7: Finale Population des „Median-Laufes“ des SMS-EMOA auf DTLZ2

trikenwerte bei DTLZ3 sind insgesamt etwas schlechter als bei DTLZ1. Den besten Konvergenzmetrikwert liefert der originale SMS-EMOA und der beste Algorithmus der Studie von Deb et al. ist der ϵ-MOEA, dessen Wert um mehr als eine Zehnerpotenz. Die Variante SMS 5 liefert den besten S-Metrikwert. Abbildung 4.8 veranschaulicht die aufgetretenen Diversitätsverluste bei DTLZ4. Die Population zieht sich mitunter auf eine Ebene des Raums zusammen, wobei das in den vorliegenden Studien auf der f_1-f_2- oder der f_1-f_3-Ebene auftrat. Außerdem passiert es bei den Varianten mit dynamischer Populationsgröße, dass die gesamte Population zu einem Punkt kollabiert. Der ϵ-MOEA hat unter zehn Läufen sechs, die erfolgreich, also auf den ganzen drei-dimensionalen Raum verteilt sind. Die SMS-EMOA Varianten mit dynamischer Populationsgröße erliegen diesem Diversitätsverlust am häufigsten und konnten nur drei bzw. zwei erfolgreiche Läufe durchführen. Deren Metrikenwerte sind allerdings sehr gut.

Vergleich der Varianten

Die Unterschiede zwischen den SMS-EMOA Varianten sind auf den meisten betrachteten Funktionen gering. Es zeigt sich, dass die SMS-EMOA Varianten mit konstanter Populationsgröße (SMS 1 bis 3) im Vergleich zu denen mit dynamischer Populationsgröße (SMS 4 und 5) robuster sind.

Die Selektionsvarianten mit konstanter Populationsgröße unterscheiden sich nur, falls

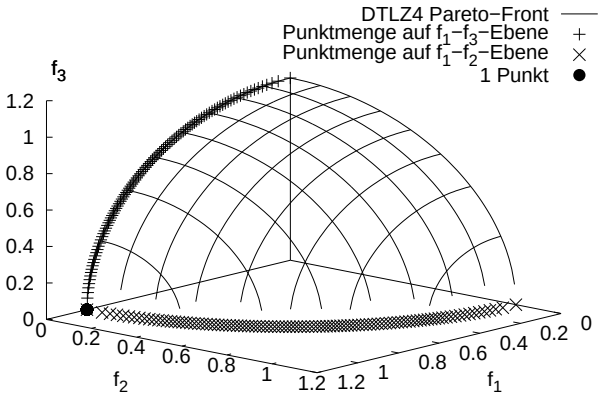

Abbildung 4.8: Aufgetretene Diversitätsverluste bei DTLZ4

die Population dominierte Individuen enthält. Bildet die Population eine nicht-dominierte Menge, so verwenden alle Varianten den Hypervolumenbeitrag als Selektionskriterium. Die Varianten unterscheiden sich daher im wesentlichen nur zu Anfang des Optimierprozesses, bis zum ersten Mal die gesamte Population auf einer Front der nicht-dominierten Sortierung liegt. Danach schwankt die Anzahl der Fronten typischerweise zwischen eins und zwei. Pro Generation wird nur ein Individuum erzeugt und dieses ist entweder nicht-dominiert oder wird von einem Mitglied der Population dominiert und liegt somit auf der zweiten Front. Nur falls ein neu erzeugtes Individuum mehrere Individuen der aktuellen Population dominiert, kann die Anzahl Fronten im Folgenden – kurzfristig und geringfügig – ansteigen.

Die Variante „SMS 3“, die keine nicht-dominierte Sortierung durchführt, hat sich als interessante Alternative herausgestellt. Ihr Verhalten ist sehr ähnlich zu dem von „SMS 2“, aber teilweise sogar wesentlich besser. Die Variante erreicht den Bestwert der Konvergenzmetrik bei ZDT3 und die Bestwerte beider Metriken auf ZDT4. Sowohl die Konvergenzmetrik als auch die S-Metrik überdurchschnittlich zu optimieren, gelingt einem EMOA nur selten. Eine weitere Außnahme in dieser Studie bildet SPEA2, der bei ZDT6 die besten Werte erreicht. Typischerweise gilt für andere EMOA, dass sie bezüglich einen Metrik einen guten und bezüglich der anderen einen umso schlechteren Wert erreichen.

Die Varianten mit dynamischer Populationsgröße wurden im wesentlichen für den Einsatz bei realen Problemstellungen entwickelt, bei denen nur eine kleine Anzahl Funktionsauswertungen durchgeführt werden kann und eine schnelle Konvergenz wünschenswert ist. Um das Potenzial dieser Selektionsvarianten zu analysieren, müsste man den Verlauf der Algorithmen beobachten und Metrikenwerte für eine viel kleinere Anzahl von Generationen berechnen. Bei der hier verwendeten hohen Anzahl von Generationen gelang es vielen Algorithmen, gute Werte zu erreichen und die SMS-EMOA Varianten konnten ihre speziellen Fähigkeiten nicht ausspielen. Es wurden weiterhin ihre Nachteile deutlich. Ihre dynamisch verwaltete Population nicht-dominierter

Individuen kann sehr klein werden und zu einem starken Diversitätsverlust führen, wie es bei den Funktionen ZDT2 und ZDT4, sowie verstärkt bei der Funktion DTLZ4 beobachtet wurde.

Es wurden weitere Varianten des SMS-EMOA entwickelt, die allerdings nur als Vergleichsobjekte dienten und nicht als empfehlenswerte Verfahren präsentiert werden sollen. Zum Vergleich zu Variante „SMS 2“ wurde im Falle dominierter Punkte eine zufällige Selektion unter den Punkten der schlechtesten Front durchgeführt. Diese Variante erzielt schlechtere Ergebnisse als „SMS 2“, die Unterschiede sind allerdings nicht sehr groß. Andererseits wurde die Selektion auf der ersten nicht-dominierten Front durch eine zufällige Wahl ersetzt. Hier zeigen sich deutlich schlechtere Ergebnisse. Die Selektion bezüglich des Hypervolumens wird daher als diejenige angesehen, die die Qualität des Verfahren prägt. Wie die vielen guten SMS-EMOA Varianten zeigen, ist diese Selektion in Verbindung mit verschiedenen Verfahren erfolgreich.

Um deutlichere Unterschiede zwischen den Varianten des SMS-EMOA festzustellen, müsste der Verlauf der Algorithmen genauer untersucht werden. Bei den dreidimensionalen DTLZ-Funktionen ist der SMS-EMOA den anderen Verfahren deutlich überlegen und es stellt sich die Frage, ob dies nicht sogar auch der Fall wäre, wenn der SMS-EMOA eine viel geringere Anzahl Generationen als die anderen Verfahren durchführen würde.

Kapitel 5

Zusammenfassung

Rekapitulation der Arbeit

Das Design eines guten evolutionären Algorihtmus zur Mehrzieloptimierung (EMOA) erscheint selbst als mehrkriterielles Problem. Viele wünschenswerte Anforderungen, wie eine hohe Qualität der Lösungsmenge, eine geringe Laufzeit und die Robustheit des Verfahrens, stehen in Konflikt zu einander. Bei dem Entwurf des *SMS-EMOA (S-Metrik Selektion EMOA)* wurde vor allem auf die Qualität der erzielten Lösungsmenge Wert gelegt, die – bewertet durch die S-Metrik – möglichst gut sein soll. Der SMS-EMOA integriert dazu die Maximierung des S-Metrikwertes der Population in den Optimierprozess. In einem steady-state Selektionsschema wird stets die Menge von Individuen als Folgepopulation ausgewählt, die den maximalen S-Metrikwert erreicht. Das bedeutet, dass das Individuum, welches den kleinsten Beitrag liefert, nicht in die Folgepopulation übernommen wird. Der SMS-EMOA hat außer den problemspezifisch zu wählenden Variationsoperatoren die Populationsgröße als einzigen Parameter.

Im einführenden Kapitel wurden die Eigenschaften der S-Metrik beschrieben, um zu verdeutlichen, warum sie als zu optimierendes Qualitätsmaß ausgewählt wurde. Die S-Metrik misst das von der Population dominierte Hypervolumen und belohnt sowohl eine gute Verteilung als auch die Annäherung an die Pareto-Front. Aufgrund der theoretischen Eigenschaften kann sie als eines der gerechtesten Maße zur Bewertung von Mengen von Vektoren angesehen werden. Daher ist die Maximierung des S-Metrikwertes ein erstrebenswertes Ziel. Der SMS-EMOA zeichnet sich dadurch aus, dass der S-Metrikwert seiner Population monoton steigend ist und es ist bekannt, dass der maximale S-Metrikwert von der Pareto-Front angenommen wird. Der SMS-EMOA erzielt nicht nur hervorragende S-Metrikwerte, sondern erreicht zudem eine sehr gute Annäherung an die Pareto-Front. Die Ergebnismengen sind dabei über die gesamte Pareto-Front bis hin zu ihren Randpunkten verteilt und liegen dichter in Regionen von Kompromisslösungen, die alle Zielkriterien relativ gut erfüllen. Durch diese Eigenschaften erreicht der SMS-EMOA eine höhere Qualität als jeder andere der betrachteten EMOA. Seine Überlegenheit ist bei den drei-kriteriellen Funktionen besonders deutlich.

Verschiedene Varianten des SMS-EMOA wurden entwickelt und können im Vergleich zu etablierten Verfahren als konkurrenzfähig bezeichnet werden. Ein alternatives Se-

lektionskriterium kann eingesetzt werden, falls die Population dominierte Individuen enthält. Es wird das dominierte Individuum aussortiert, welches die größte Anzahl dominierender Individuen aufweist. Dadurch werden Individuen in spärlich besetzten Region des Zielfunktionsraums bevorzugt. Das Selektionsverfahren hat den Vorteil, dass es effizienter zu berechnen ist und empirische Studien zeigen, dass die Variante qualitativ sehr ähnliche Ergebnisse zum originalen SMS-EMOA liefert. In dieser Arbeit wurde eine leichte Abwandlung dieser Variante vorgestellt, bei der zur Selektion keine vorausgehende nicht-dominierte Sortierung der Population durchgeführt wird. Der SMS-EMOA wurde dadurch konzeptionell vereinfacht und diese Variante lieferte ebenfalls sehr gute Ergebnisse. Es liegen außerdem Varianten vor, die dynamisch eine Population nicht-dominierter Individuen verwalten.

Für die Spezialfälle von zwei- bzw. drei-dimensionalen Zielfunktionsräumen wurden bereits spezielle Algorithmen zur Berechnung der Hypervolumenbeiträge entwickelt. In dieser Arbeit wurde die Sichtweise des dominierten Hypervolumens als *Klee's measure problem* eingeführt. Dadurch wurde ein Algorithmus gefunden, der den Wert der S-Metrik bzw. des dominierten Hypervolumens einer m-elementigen Menge d-dimensionaler Zielfunktionsvektoren in einer Zeit von $O(m^{d/2} \log m)$ bestimmt. Zuvor lag die obere Schranke der worst case Laufzeit bei $O(m^{d+1})$.

Stand der Forschung zum SMS-EMOA

Es liegen aktuell mehrere Varianten des SMS-EMOA vor, die in dieser Arbeit dargestellt wurden. Ein Übersichtsartikel von Beume et al. [BNE06] gibt diese Varianten sowie aktuelle Ergebnisse wieder. Der SMS-EMOA hat sowohl in den Leistungsstudien auf zwei- und drei-kriteriellen Testfunktionen als auch bei aeronautischen Optimierproblemen jeden anderen betrachteten EMOA in der Qualität der erzielten Lösungsmenge übertroffen.

Der Schwachpunkt des SMS-EMOA ist seine hohe Laufzeit, die durch die Rechenzeitintensive Berechnung des dominierten Hypervolumens geprägt ist. Für die Laufzeit einer Generation des SMS-EMOA mit Populationsgröße μ gilt aktuell die obere Schranke von $O(\mu^{(d/2)+1} \log \mu)$ für ein d-kriterielles Problem. Obwohl der SMS-EMOA auch in hoch-dimensionalen Zielfunktionsräumen hervorragende Ergebnisse erzielt, kann er aufgrund seiner hohen Laufzeit nur für Probleme mit kleiner Anzahl von Zielfunktionen uneingeschränkt empfohlen werden. Für einen Hypervolumenalgorithmus, dessen worst case Laufzeit exponentiell in der Anzahl der Zielfunktionen ist, entwickelten While et al. [WBBH05] Heuristiken zu dessen Beschleunigung. Mit Einsatz dieser Verfahren könnte sich auch der SMS-EMOA im average case effizient verhalten.

Bei praktischen Anwendungsproblemen sind es oft die Zielfunktionsauswertungen, die aufgrund aufwändiger Simulationen die Laufzeit des Optimierprozesses prägen. Die Rechenzeit des zugrunde liegenden EMOA ist dadurch nebensächlich und der SMS-EMOA kann als viel versprechendes Optimierverfahren empfohlen werden. Durch seinen intelligenten Selektionsoperator sollte er insbesondere zu guten Ergebnissen führen, wenn nur eine geringe Anzahl von Funktionsauswertungen durchge-

führt werden kann, wie es sich auch bei den aeronautischen Problemstellungen zeigte.

Weitergehende Fragestellungen

Detailliertere Studien der SMS-EMOA Varianten im Vergleich zu anderen etablierten EMOA sollen Aufschluss über den Verlauf der Algorithmen geben. Die nähere Untersuchung der SMS-EMOA Varianten soll dazu dienen, Empfehlungen aussprechen zu können, in welchen Anwendungsszenarien oder bei Problemen welcher Struktur eine Variante vorteilhaft ist. Insbesondere ist die Betrachtung bei einer kleinen Anzahl von Funktionsauswertung von Interesse für Anwendungen auf reale Problemstellungen.

Die Selektion bezüglich des dominierten Hypervolumens hat sich in verschiedenen Varianten von Selektionsoperatoren des SMS-EMOA als erfolgreich erwiesen. Zudem wurde von Emmerich et al. [EBN05] bereits die Kopplung des SMS-EMOA mit einem Metamodell untersucht. Außerdem setzte Emmerich [Emm05b] innerhalb des Metamodells ein Hypervolumen-basiertes Kriterium ein, um eine Vorselektion viel versprechender Individuen durchzuführen. Es stellt sich die Frage, ob auch in anderen Verfahren der Computational Intelligence der Einsatz eines Hypervolumen-Indikators möglich ist, wie es beispielsweise für Partikel Schwarm Optimierer (PSO) schon angedacht wurde. Diese Frage ist ein aktueller Forschungsgegenstand genau wie die Fragestellung, wie Präferenzen eines Anwenders durch spezielle Qualitätsindikatoren abgebildet und angestrebt werden können.

Es ist aktuell unklar, inwieweit der Einsatz des besten Algorithmus für Klee's measure problem die Laufzeit des SMS-EMOA tatsächlich verbessert. Zukünftige Laufzeitstudien sollen die Frage klären, wie stark der SMS-EMOA bei welcher Populationsgröße und welcher Dimension des Zielfunktionsraums beschleunigt wird. Eine offene Frage ist, ob sich für die spezielle Anordnung der Hyperquader, die durch eine Menge nicht-dominierter Punkte induziert werden, ein effizienterer Algorithmus finden lässt. Denkbar wäre auch, dass eine Analyse der Laufzeit des verwendeten Algorithmus für diesen Spezialfall bereits eine kleinere obere Laufzeitschranke liefert. Zudem könnte an weiteren Algorithmen für die im SMS-EMOA zu berechnenden Hypervolumenbeiträge gearbeitet werden. Ein aktuell lebendiges Forschungsgebiet ist auch die Thematik, wie weit die Beschleunigung von Hypervolumenalgorithmen durch Heuristiken ausgereizt werden kann.

Anhang A

Mathematische Grundlagen

Partiell geordnete Mengen

Die theoretischen Grundlagen partiell geordneter Mengen werden hier erläutert in einer Darstellung, die sich an die von Trotter ([Tro95], Kapitel 8) anlehnt.

Eine *binäre Relation* O auf zwei Mengen A und B ist ein Teilmenge des kartesischen Produkts der Mengen, also $O \subseteq A \times B$. Sie besteht damit aus geordneten Paaren $(a, b) \in O$, für die man auch $a\ O\ b$ oder $a \leq b$ schreibt. Einer binären Relation kann auch nur eine Menge A zugrunde liegen und damit gilt entsprechend $O \subseteq A \times A$. Zwei Elemente sind *vergleichbar*, falls $x \leq y$ oder $y \leq x$ gilt. Für *unvergleichbare* Elemente schreibt man $x||y$.

Eine *Relation* O auf einer Menge M heißt

- *reflexiv*, falls gilt : $\forall x \in M : xOx$
- *irreflexiv*, falls gilt: $\nexists x \in M : xOx$
- *transitiv*, falls gilt: $\forall x, y, z \in M : xOy\ \wedge\ yOz \Rightarrow xOz$
- *antisymmetrisch*, falls gilt: $\forall x, y \in M : xOy \Rightarrow x = y$.

Eine *partiell geordnete Menge (partially ordered set, poset)* ist ein Paar $\mathbf{P} = (M, O)$ einer Menge X und einer binären Relation O auf M, die reflexiv, antisymmetrisch und transitiv ist. M heißt Grundmenge (ground set) und O partielle Ordnung (partial order). Eine *strikte partiell geordnete Menge* ist irreflexiv (und demnach nicht antisymmetrisch) und transitiv.

Eine partiell geordnete Menge ist eine *Kette (chain)* (auch total oder linear geordnete Menge), falls alle verschiedenen Elemente paarweise vergleichbar sind. In einer *Antikette (antichain)* sind alle verschiedenen Elemente unvergleichbar. Die *Höhe (height)* einer Menge ist definiert als die maximale Kardinalität aller Teilmengen, die Ketten sind. Die *Breite (width)* ist entsprechend die Kardinalität der größten Antikette.

Ein Element $a \in M$ heißt *minimal*, falls es kein Element $b \in M$ mit $b < a$ gibt. Analog heißt ein Element $a \in M$ *maximal*, falls es kein Element $b \in M$ gibt mit $a < b$.

Die Menge der minimalen Elemente und die Menge der maximalen Element bilden jeweils Antiketten.

Dilworth's Dekompositionstheorem (decomposition theorem) [Dil50] liefert einen Zusammenhang für die Partitionierung einer partiell geordneten Menge in Ketten bzw. Antiketten.

Theorem A.1 (Dekompositionstheorem) *Falls eine partiell geordnete Menge* $\mathbf{P} = (M, P)$ *Breite* n *hat, dann gibt es eine Partitionierung von* M *in* n *Ketten. Falls* $\mathbf{P} = (M, P)$ *Höhe* m *hat, dann gibt es eine Partitionierung von* M *in* m *Antiketten.*

Die Beweise der Aussagen sind in [Tro95] zu finden.

Anhang B

Algorithmische Ergänzungen

Nicht-dominierte Sortierung

Algorithmus B.1 ist die von Deb et al. [DPAM02] entwickelte Implementierung der nicht-dominierten Sortierung (vgl. Definition 1.5). Die prinzipiele Funktionsweise des Algorithmus ist Abschnitt 2.2 beschrieben. Die Laufzeit von fast-nondominated-sort beträgt $O(d\mu^2)$ und die Platzkomplexität $O(n^2)$.

Algorithmus B.1 fast-nondominated-sort

```
 1: n_p = 0                                   initialisiere Zähler der dominierenden Lösungen
 2: S_p = ∅                                   initialisiere Liste der dominierten Lösungen
 3: for all p ∈ P_t do
 4:    for all q ∈ P_t do
 5:       if p ≺ q then
 6:          S_p = S_p ∪ {q}                 füge q in S_p ein
 7:       else if q ≺ p then
 8:          n_p = n_p + 1                   erhöhe n_p
 9:       end if
10:    end for
11:    if n_p = 0 then
12:       F_i = F_i ∪ {p}                    p wird nicht dominiert und gehört zur 1. Front
13:    end if
14: end for
15: i = 1
16: while F_i ≠ ∅ do
17:    H = ∅                                 initialisiere nächste Front
18:    for all p ∈ F_i do
19:       for all q ∈ S_p do
20:          n_q = n_q − 1                   entferne Elemente der aktuellen Front
21:          if n_q = 0 then
22:             H = H ∪ {q}                  H enthält die nun nicht dominierten Lösungen
23:          end if
24:       end for
25:    end for
26:    i = i + 1
27:    F_i = H                               neue aktuelle Front
28: end while
```

Variationsoperatoren SBX und PM

Die *simulierte binäre Rekombination (simulated binary crossover, SBX)* entwickelt von Deb und Agrawal [DA94] arbeitet mit zwei Eltern und erzeugt aus ihnen zwei Nachkommen. Der SBX-Operator simuliert die Durchführung einer 1-Punkt-Rekombination wie sie bei einer binären Repräsentation gebräuchlich ist. Die Bitstrings der beiden Eltern werden dabei an einem zufälligen Trennpunkt zerteilt und es entstehen zwei Nachkommen, indem man jeweils den ersten Teil vom einen und den zweiten Teil vom anderen Elter übernimmt. Konkret besteht der SBX-Operator aus folgenden Rechenschritten, die für jede Entscheidungsvariable $x_i^{(o1)}$ und $x_i^{(o2)}$ ($i = 1, \ldots n$) der Nachkommen durchgeführt werden müssen.

1. Erzeuge eine $(0, 1)$-gleichverteilte Zufallsvariable u.

2. Berechne den Ausbreitungsfaktor (spread factor) β_i, der im Folgenden den Abstand der Nachkommen von den Eltern $\mathbf{x}^{(1)}(t)$, $\mathbf{x}^{(2)}(t)$ im Genotypraum bestimmt.

$$\beta_i = \begin{cases} 2u^{\frac{1}{\eta_c+1}} - 1 & \text{falls} \quad u \leq 0,5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}} & \text{sonst} \end{cases} \tag{B.1}$$

3. Berechne die Entscheidungsvariablen $\mathbf{x}_i^{(o1)}$, $\mathbf{x}_i^{(o2)}$ der Nachkommen wie folgt.

$$\begin{aligned} x_i^{(o1)} &= 0{,}5(\ (1+\beta_i)x_i^{(1)}(t) + (1-\beta)x_i^{(2)}(t)\ ) \\ x_i^{(o2)} &= 0{,}5(\ (1-\beta_i)x_i^{(1)}(t) + (1+\beta)x_i^{(2)}(t)\ ) \end{aligned} \tag{B.2}$$

Den Parametern β_i liegt dabei folgende Dichtefunktion zugrunde:

$$P(\beta) = \begin{cases} 0{,}5(\eta_c+1)\beta^{\eta_c} & \text{falls} \quad \beta \leq 1 \\ 0{,}5(\eta_c+1)\frac{1}{\beta^{\eta_c+2}} & \text{sonst.} \end{cases} \tag{B.3}$$

Gleichung B.1 entsteht durch eine inverse Transformation der Dichtefunktion (siehe beispielsweise Bronstein [BSMM00]). So wie der Parameter β_i in den Gleichungen B.2 verwendet wird, entspricht er dem Quotienten des Abstands der Nachkommen und dem Abstand der Eltern im Genotypraum: $\beta_i = \left(x_i^{(o2)} - x_i^{(o1)}\right) / \left(x_i^{(2)}(t) - x_i^{(1)}(t)\right)$.

Die *polynomielle Mutation (polynomial mutation, PM)* von Deb und Goyal [DG96] addiert zu den Entscheidungsvariablen eine Zufallsvariablen, der eine polynomielle Wahrscheinlichkeitsverteilung zugrunde liegt. Man geht bei der Mutation für jede Entscheidungsvariable $x_i^{(o)}$ folgendermaßen vor.

1. Lege untere und obere Grenzen x_i^{min} bzw. x_i^{max} der Werte jeder Entscheidungsvariablen fest.

2. Erzeuge eine $(0, 1)$-gleichverteilte Zufallsvariable u.

3. Berechne einen Störfaktor (perturbance factor) δ, der die Stärke der Mutation bestimmt. Die Dichtefunktion $P(\delta)$ seiner polynomiellen Wahrscheinlichkeitsverteilung lautet:

$$P(\delta) = 0{,}5\,(\eta_m + 1) \cdot (1 - |\delta|)^{\eta_m}. \tag{B.4}$$

Werte für konkrete δ_i werden dann berechnet als:

$$\delta_i = \begin{cases} 2u^{\frac{1}{\eta_m+1}} - 1 & \text{falls} \quad u < 0{,}5 \\ 1 - (2(1-u))^{\frac{1}{\eta_m+1}} & \text{sonst.} \end{cases} \tag{B.5}$$

4. Berechne die mutierten Entscheidungsvariablen des Nachkommen $\mathbf{x}^{(o)}$ als:

$$x_i^{(o')} = x_i^{(o)} + \delta_i(x_i^{max} - x_i^{min}). \tag{B.6}$$

Der SMS-EMOA erzeugt als steady-state-Algorithmus nur einen Nachkommen pro Generation. Von den beiden erzeugten Genomen wird eines zufällig gleichverteilt ausgewählt, das dann für den einen Nachkommen übernommen wird. Die anschließende Mutation wird dann nur auf dieses eine Genom angewendet. Für den SMS-EMOA wird die gleiche Parametrisierung wie beim ϵ-MOEA verwendet, damit die Ergebnisse mit denen aus der Studie von Deb et al. [DMM03b, DMM03a] möglichst gut vergleichbar sind. Bei der Rekombination wird der Parameter $\eta_c = 15$ und für die Mutation $\eta_m = 20$ gewählt. Wir verwendeten die Implementierung der Operatoren, die auf der Homepage des Kanpur Genetic Algorithms Laboratory (KanGAL) als C-Code verfügbar ist unter http://www.iitk.ac.in/kangal/codes.shtml.

ES-Variationsoperatoren

Bei den Studien von Naujoks et al. [NBE05b, NBE05a] des SMS-EMOA auf aeronautischen Problemen wurden Vereinfachungen von Variationsoperatoren eingesetzt, die traditionell in Evolutionsstrategien bei reellwertiger Repräsentation verwendet werden. Es wird eine diskrete Rekombination mit zwei uniform zufällig gewählten Eltern $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$ durchgeführt. Bei der Erzeugung eines Nachkommen wird für jede Entscheidungsvariable zufällig gleichverteilt entschieden, ob sie vom einen oder vom anderen Elter übernommen wird. Es werden konstante Schrittweiten benutzt, diese sind also für alle Individuen gleich und unterliegen keinem Evolutionsprozess. Die i-te Entscheidungsvariable $x_i^{(o)}$ berechnet sich also gemäß

$$x_i^{(o)} = \begin{cases} x_i^{(1)} & \text{falls} \quad u < 0{,}5 \quad , \; u \text{ zufällig gleichverteilt in [0,1]} \\ x_i^{(2)} & \text{sonst} \end{cases} \tag{B.7}$$

Die Entscheidungsvariablen werden anschließend mutiert, indem eine normalverteilte Zufallsvariable zu den Werten addiert wird. Der Wert der i-ten Entscheidungsvariable entspricht damit:

$$x_i^{(o')} \; = \; x_i^{(o)} + s_i \cdot N(0,1). \tag{B.8}$$

Die Verteilung ist dabei abhängig von der gewählten Schrittweite s_i, die wie erwähnt konstant ist. $N(0,1)$ steht für eine normalverteilte Zufallsvariable mit Erwartungswert 0 und Standardabweichung 1. Durch die Multiplikation mit den Schrittweiten wird die Standardabweichung der Verteilung modifiziert.

Symbolverzeichnis

Notation

$\bullet^{(i)}$	Der hochgestellte geklammerte Index i bezeichnet den i-ten Wert einer durchnummerierten Menge
$\bullet_i$	Der tiefgestellte Index i bezeichnet einen Wert in der i-ten Dimension
$\mathbf{b}$	ein kleiner, fett gedruckter, römischer Buchstabe bezeichnet einen Vektor
$i \in [j]$	$i \in \{1, \ldots, j\}, i \in \mathbb{N}$

Parameter von evolutionären Algorithmen

$\mathbf{a} = (\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z})$	Individuum $\mathbf{a}$ mit Strategievariablen $\mathbf{w}$, Genom $\mathbf{x}$, sowie Zielfunktionsvektor $\mathbf{y}$ und Fitness $\mathbf{z}$
η_c	Verteilungsindex der polynomiellen Verteilung bei Simulatd Binary Crossover (SBX)
η_m	Verteilungsindex der polynomiellen Verteilung bei der Polynomiellen Mutation (PM)
λ	Größe der Nachkommenpopulation
μ	Größe der Population
μ_{init}	initiale Populationsgröße bei SMS-EMOA-dynPop1 und SMS-EMOA-dynPop2
μ_{max}	maximale Populationsgröße bei SMS-EMOA-dynPop1 und SMS-EMOA-dynPop2
$\mathbf{w}$	Vektor der Strategie- bzw. Steuervariablen
$\mathbf{x}$	Genom bzw. Objektparameter $\mathbf{x} = (x_1, \ldots x_n)$ eines Individuums
$\mathbf{y}$	d-dimensionaler Vektor der Zielfunktionswerte $\mathbf{y} = (f_1(\mathbf{x}), \ldots, f_d(\mathbf{x}))$ eines Individuums

$\mathbf{z}$	Vektor der Fitnesswerte, der von den Zielfunktionswerten und der aktuellen Popualtion abhängt. Auf $\mathbf{z}$ basiert die Selektion eines Individuums.
n	Anzahl der Objektparameter $\mathbf{x} = (x_1, \ldots, x_n)$
P	Menge von Individuen, verwendet in Prozeduren von EMOA
p_c	Anwendungswahrscheinlichkeit des Rekombinationsoperators
p_m	Mutationswahrscheinlichkeit pro Entscheidungsvariable
P_t	Population der t-ten Generation

Parameter von Testfunktionen und Anwendungsproblemen

γ	Parameter der Testfunktionen EBN
ν	Parameter der Testfunktionen DTLZ
d	Anzahl Kriterien der Zielfunktion $f = (f_1, \ldots, f_d)$, Dimension des Zielfunktionsraums

Bewertungsfunktionen

$\Delta_S(\mathbf{y}, M)$	Hypervolumenbeitrag von $\mathbf{y}$ innerhalb der Menge M
$DZ(\mathbf{y}, M)$	Dominanzzahl von $\mathbf{y}$ innerhalb der Menge M
$\mathbf{r}$	Referenzpunkt der S-Metrik $S(M, \mathbf{r})$
F_i	i-te Front der nicht-dominierten Sortierung
$G(\mathbf{y}, M)$	Besiedelungsdichte (crowding distance) von $\mathbf{y}$ innerhalb der Menge M
$K(M, R)$	Konvergenz-Metrik der Menge M bzgl. der Referenzmenge R
$ND(M)$	Nicht-dominierte Menge von M
$ND_{part}(M, k)$	Nicht-k-dominierte Menge von M
R	Referenzmenge der Konvergenz-Metrik $K(M, R)$
$S(M, \mathbf{r})$	S-Metrik der Menge M bzgl. des Referenzpunkts $\mathbf{r}$
v	Anzahl Fronten der nicht-dominierten Sortierung

Variablen der Hypervolumenberechnung

M	Menge von Punkten oder Hyperquadern
m	Kardinalität einer Menge von Punkten oder Hyperquadern
Q	d-dimensionaler Hyperquader $Q = (\mathbf{l}, \mathbf{u})$ mit unteren Intervallgrenzen $\mathbf{l} = (l_1, \ldots, l_d)$ und oberen Intervallgrenzen $\mathbf{u} = (u_1, \ldots, u_d)$

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[BB06] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation – The New Experimentalism*. Natural Computing Series. Springer, Berlin, 2006. (im Druck).

[Ben75] J. L. Bentley. Multidimensional Binary Search Tree Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Ben77] J. L. Bentley. Algorithms for Klee's rectangle problem. Interner Bericht, Department of Computer Science, CMU, 1977.

[BHS97] T. Bäck, U. Hammel und H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[BLTZ03] S. Bleuler, M. Laumanns, L. Thiele und E. Zitzler. PISA — A Platform and Programming Language Independent Interface for Search Algorithms. In C. M. Fonseca et al., Hrsg., *Evolutionary Multi-Criterion Optimization, 2nd Int'l Conf. (EMO 2003)*, LNCS 2632, Seiten 494 – 508. Springer, Berlin, 2003.

[BNE06] N. Beume, B. Naujoks und M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 2006. (Im Druck).

[BS02] H.-G. Beyer und H.-P. Schwefel. Evolution strategies – A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[BSMM00] I. N. Bronstein, K. A. Semendjajew, G. Musiol und H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt a.M., 2000.

[CK03a] D. Corne und J. Knowles. No Free Lunch and Free Leftovers Theorems for Multiobjective Optimisation Problems. In C. M. Fonseca et al., Hrsg., *Evolutionary Multi-Criterion Optimization, 2nd Int'l Conf. (EMO 2003)*, LNCS 2632, Seiten 327–341. Springer, Berlin, 2003.

[CK03b] D. Corne und J. Knowles. Some Multiobjective Optimizers are Better than Others. In *Proc. of the 2003 Congress on Evolutionary Computation (CEC 2003), Canberra*, Band 4, Seiten 2506–2512. IEEE Press, Piscataway NJ, 2003.

[CVL02] C. A. Coello Coello, D. A. Van Veldhuizen und G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.

[DA94] K. Deb und R. B. Agrawal. Simulated Binary Crossover for continuous search space. Technical Report 94027, Indian Institute of Technology, Kanpur, India, 1994.

[DAPM00] K. Deb, S. Agrawal, A. Pratap und T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer et al., Hrsg., *Proc. of the Parallel Problem Solving from Nature VI Conference, Paris*, LNCS 1917, Seiten 849–858. Springer, Berlin, 2000.

[Deb99] K. Deb. Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[Deb01] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[DG96] K. Deb und M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[Dil50] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–165, 1950.

[DJW99] S. Droste, T. Jansen und I. Wegener. Perhaps Not a Free Lunch But At Least a Free Appetizer. In W. Banzhaf et al., Hrsg., *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99), Orlando FL*, Band 1, Seiten 833–839. Morgan Kaufmann, San Francisco CA, 1999.

[DJW02] S. Droste, T. Jansen und I. Wegener. Optimization with randomized search heuristics—the (A)NFL theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287(1):131–144, 2002.

[DMM03a] K. Deb, M. Mohan und S. Mishra. A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions. KanGAL report 2003002, Indian Institute of Technology, Kanpur, India, 2003.

[DMM03b] K. Deb, M. Mohan und S. Mishra. Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In C. M. Fonseca et al., Hrsg., *Evolutionary Multi-Criterion Optimization. 2nd Int'l Conf. (EMO 2003)*, LNCS 2632, Seiten 222–236. Springer, Berlin, 2003.

[DPAM02] K. Deb, A. Pratap, S. Agarwal und T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[DTLZ02] K. Deb, L. Thiele, M. Laumanns und E. Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Proc. of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Band 1, Seiten 825–830. IEEE Press, Piscataway NJ, 2002.

[EBN05] M. Emmerich, N. Beume und B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In C. A. Coello Coello et al., Hrsg., *Proc. Evolutionary Multi-Criterion Optimization, 3rd Int'l Conf. (EMO 2005)*, LNCS 3410, Seiten 62–76. Springer, Berlin, 2005.

[Emm05a] M. Emmerich. A Rigorous Analysis of Two Bi-Criteria Problem Families with Scalable Curvature of the Pareto Fronts. Technischer Bericht LIACS TR 2005-05, Leiden Institute on Advanced Computer Science, Leiden, NL, 2005.

[Emm05b] M. Emmerich. *Single- and Multi-objective Design Optimization Assisted by Gaussian Random Field Models*. Dissertation, Universität Dortmund, Dortmund, 2005.

[EO85] H. Edelsbrunner und M. H. Overmars. Batched dynamic solutions to decomposable searching problems. *Journal of Algorithms*, 6(4):515–542, 1985.

[Fle02] M. Fleischer. The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics. Interner Bericht 2002-32, Institute for Systems Research, University of Maryland, College Park, MD, 2002.

[Fle03a] M. Fleischer. Foundations of Swarm Intelligence: From Principles to Practice. Technical report CSHCN TR 2003-5, Institute for Systems Research, University of Maryland, College Park, MD, 2003.

[Fle03b] M. Fleischer. The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics. In C. M. Fonseca et al., Hrsg., *Evolutionary Multi-Criterion Optimization, 2nd Int'l Conf. (EMO 2003)*, LNCS 2632, Seiten 519–533. Springer, Berlin, 2003.

[FOW66] L. J. Fogel, A. J. Owens und M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, 1966.

[FW78] M. L. Fredman und B. Weide. The complexity of computing the measure of $\bigcup[a_i, b_i]$. In *Communications of the ACM*, Band 21, Seiten 540–544, 1978.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[Hei99] L. J. Heinrich. Grundlagen der Wirtschaftsinformatik. In P. Rechenberg und G. Pomberger, Hrsg., *Informatik-Handbuch*, Seiten 1019–1034. Hanser, München, 1999.

[HHWB03] S. Huband, P. Hingston, L. While und L. Barone. An Evolution Strategy with Probabilistic Mutation for Multi-Objective Optimisation. In *Proc. of the Congress on Evolutionary Computation (CEC 2003), Canberra, Australia*, Band 4, Seiten 2284–2291. IEEE Press, Piscataway NJ, 2003.

[HJ98] M. P. Hansen und A. Jaszkiewicz. Evaluating the quality of approximations of the non-dominated set. Technical Report IMM-REP-1998-7, Institute of Mathematical Modeling, Technical University of Denmark, 1998.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.

[KC02] J. Knowles und D. Corne. On Metrics for Comparing Nondominated Sets. In *Proc. of the Congress on Evolutionary Computation (CEC 2002)*, Band 1, Seiten 711–716. IEEE Press, Piscataway NJ, 2002.

[KC03] J. Knowles und D. Corne. Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, 2003.

[KCF03] J. D. Knowles, D. W. Corne und M. Fleischer. Bounded Archiving using the Lebesgue Measure. In *Proc. of the 2003 Congress on Evolutionary Computation (CEC 2003), Canberra, Australia*, Band 4, Seiten 2490–2497. IEEE Press, Piscataway NJ, 2003.

[Kle77] V. Klee. Can the measure of $\bigcup[a_i, b_i]$ be computed in less than $O(n \log n)$ steps? In *American Mathematical Monthly*, Band 84, Seiten 284–285, 1977.

[Kno02] J. D. Knowles. *Local Search and Hybrid Evolutionary Algorithms for Pareto Optimization.* Dissertation, University of Reading, Reading, UK, 2002.

[Koz92] J. R. Koza. *Genetic Programming.* MIT Press, Cambridge, MA, 1992.

[Mie01] K. Miettinen. Some Methods for nonlinear multi-objective optimization. In E. Zitzler et al., Hrsg., *Evolutionary Multi-Criterion Optimization, 1st Int'l Conf. (EMO 2001)*, LNCS 1993, Seiten 1–20. Springer, Berlin, 2001.

[NBE05a] B. Naujoks, N. Beume und M. Emmerich. Metamodel-assisted SMS-EMOA applied to airfoil optimization tasks. In R. Schilling et al., Hrsg., *Proc. EUROGEN'05* (CD-ROM). FLM, TU München, 2005.

[NBE05b] B. Naujoks, N. Beume und M. Emmerich. Multi-objective optimisation using S-metric selection: Application to three-dimensional solution spaces. In B. McKay et al., Hrsg., *Proc. of the 2005 Congress on Evolutionary Computation (CEC 2005), Edinburgh*, Band 2, Seiten 1282–1289. IEEE Press, Piscataway NJ, 2005.

[OY88] M. H. Overmars und C.-K. Yap. New upper bounds in Klee's measure problem (extended abstract). In *Proc. of the 29th IEEE Symposium on the Foundations of Computer Science (FOCS'88), Edinburgh*, Seiten 550–556. IEEE Computer Society Press, Washington, D.C., 1988.

[OY91] M. H. Overmars und C.-K. Yap. New upper bounds in Klee's measure problem. *SIAM Journal on Computing*, 20(6):1034–1045, 1991.

[Rec94] I. Rechenberg. *Evolutionsstrategie '94*. Frommann-Holzbook, Stuttgart, 1994.

[Rud01] G. Rudolph. Evolutionary search under partially ordered fitness sets. In M. F. Sebaaly, Hrsg., *Proc. Int'l NAISO Congress on Information Science Innovations (ISI 2001)*, Seiten 818–822. ICSC Academic Press, Millet, Kanada, 2001.

[Sch65] H.-P. Schwefel. *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Diplomarbeit, Technische Universität Berlin, Hermann Föttinger–Institut für Strömungstechnik, 1965.

[Sch68] H.-P. Schwefel. Experimentelle Optimierung einer Zweiphasendüse, Teil I. Bericht Nr. 35 zum Projekt MHD–Staustrahlrohr 11.034/68, AEG Forschungsinstitut, Berlin, 1968.

[SD95] N. Srinivas und K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[Tro95] W. T. Trotter. Partially Ordered Sets. In R. L. Graham, M. Grötschel und L. Lovász, Hrsg., *Handbook of Combinatorics, Volume 1*, Seiten 433–480. Elsevier Science B. V., Amsterdam, Niederlande, 1995.

[vLW80] J. van Leeuwen und D. Wood. The measure problem for rectangular ranges in d-space. In *Journal of Algorithms*, Band 2, Seiten 282–300, 1980.

[WBBH05] L. While, L. Bradstreet, L. Barone und P. Hingston. Heuristics for Optimising the Calculation of Hypervolume for Multi-objective Optimisation Problems. In B. McKay et al., Hrsg., *Proc. of the 2005 Congress on Evolutionary Computation (CEC 2005), Edinburgh*, Band 3, Seiten 2225–2232. IEEE Press, Piscataway NJ, 2005.

[Weg03] I. Wegener. *Komplexitätstheorie – Grenzen der Effizienz von Algorithmen*. Springer, Berlin, 2003.

[Wel88] E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proc. of the 4th ACM Symposium on Computational Geometry, Urbana-Champaign IL*, Seiten 23–33. ACM Press, New York, 1988.

[WHBH06] L. While, P. Hingston, L. Barone und S. Huband. A Faster Algorithm for Calculating Hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, 2006.

[Whi05] L. While. A New Analysis of the LebMeasure Algorithm for Calculating the Hypervolume. In C. A. Coello Coello et al., Hrsg., *Proc. Evolutionary Multi-Criterion Optimization, 3rd Int'l Conf. (EMO 2005)*, LNCS 3410, Seiten 326–340. Springer, Berlin, 2005.

[Wil82] D. E. Willard. Polygon retrieval. *SIAM Journal of Computing*, 11:149–165, 1982.

[WM97] D. H. Wolpert und W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[ZDT00] E. Zitzler, K. Deb und L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[Zit99] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Dissertation, Swiss Federal Institute of Technology (ETH), Zürich, Schweiz, 1999.

[Zit01] E. Zitzler. *Hypervolume Metric Calculation*. Computer Engineering and Networks Laboratory (TIK), Zürich, 2001. ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c.

[ZK04] E. Zitzler und S. Künzli. Indicator-Based Selection in Multiobjective Search. In X. Yao et al., Hrsg., *8th Int'l Conf. on Parallel Problem Solving from Nature (PPSN VIII)*, Seiten 832–842, UK, 2004. Springer, Berlin.

[ZLT02] E. Zitzler, M. Laumanns und L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design, Optimisation, and Control*, Seiten 95–100, Barcelona, Spain, 2002. CIMNE.

[ZT98] E. Zitzler und L. Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Case Study. In A. E. Eiben, Hrsg., *Parallel Problem Solving from Nature V*, LNCS 1498, Seiten 292–301. Springer, Berlin, 1998.

[ZTL+03] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca und V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.