

Bauhaus-Universität Weimar
Fakultät Bauingenieurwesen
Institut für Strukturmechanik



Aufgabenstellung

Implementation und Anwendung von Mehrzieloptimierung

Auf der Basis von evolutionären Algorithmen sollen Verfahren zur Mehrzieloptimierung in der Literatur recherchiert werden. Davon soll eines ausgewählt, in OptiSLang implementiert und auf ein mechanisches Optimierungsproblem angewendet werden.

Bearbeiter	Stephan Blum
Erstprüfer	Prof. Dr. Christian Bucher
Zweitprüfer	Dipl.-Ing. Jörg Riedel
Ausgabedatum	26.04.2004
Abgabedatum	26.07.2004

Bauhaus-Universität Weimar
Fakultät Bauingenieurwesen
Institut für Strukturmechanik



Diplomarbeit

Implementation und Anwendung von Mehrzieloptimierung

eingereicht von

Stephan Blum
geboren am 3. August 1976
in Halle (Saale)

Seminargruppe B/96/P
Matrikelnummer 960016

Erstprüfer Prof. Dr. Christian Bucher
Zweitprüfer Dipl.-Ing. Jörg Riedel

Ausgabedatum 26.04.2004
Abgabedatum 26.07.2004

Vorsitzender des Prüfungsausschusses

Inhaltsverzeichnis

1	Einleitung	1
2	Mehrzieloptimierung	2
2.1	Problemformulierung	2
2.2	Pareto-Optimum	3
2.3	Methoden zur Mehrzieloptimierung	7
3	Evolutionäre Strategien zur Mehrzieloptimierung	11
3.1	Grundlagen Evolutionärer Algorithmen	11
3.2	Multikriterielle Evolutionäre Algorithmen	13
3.3	Bewertung und Selektion	14
3.3.1	Selektion durch Vektorauswertung	14
3.3.2	Dominanzbasierte Verfahren	14
3.4	Diversität der Population	18
3.4.1	Fitneßteilung	18
3.4.2	Dichteabschätzung	19
4	Strength Pareto Evolutionary Algorithm	21
4.1	Ablauf des Algorithmus	21
4.2	Fitneßzuweisung	22
4.3	Selektion durch Dichteabschätzung	24
5	Erweiterungen	27
5.1	Simulated Binary Crossover	27
5.2	Berücksichtigung von Nebenbedingungen	31
6	Beispiele	34
6.1	Konstruierte Testprobleme	34
6.2	Strukturmechanisches Optimierungsproblem	42
7	Zusammenfassung	46

A	Dokumentation der SLang-Eingabedateien	49
A.1	Definition der SPEA2-Parameter (<code>InitSPEA.s</code>)	49
A.2	Definition der Problemparameter (<code>Problem.s</code>)	50
A.3	Implementation des SPEA2-Algorithmus (<code>SPEA.s</code>)	50
B	Dokumentation des SLang-Kommandos	56

1

Einleitung

Die meisten praktischen Optimierungsprobleme beinhalten mehr als nur eine Zielfunktion und eine Vielzahl von Designvariablen. Die gleichzeitige Berücksichtigung aller Zielfunktionen führt zu einer Menge Pareto-optimaler Lösungen für das Optimierungsproblem. Es existieren zahlreiche Methoden zur Mehrzieloptimierung, von denen die evolutionären Algorithmen in den letzten 15 Jahren zunehmend an Bedeutung gewonnen haben.

Im Rahmen dieser Arbeit wird zunächst das Mehrzieloptimierungsproblem formuliert, die Terminologie der Pareto-Dominanz erläutert und die existierenden Methoden zur Mehrzieloptimierung klassifiziert. Die evolutionären Strategien zur Lösung von Mehrzieloptimierungsproblemen werden bezüglich ihrer Bewertungs- und Selektionsstrategien charakterisiert. Dabei liegt der Schwerpunkt auf Algorithmen, die sich für Ingenieur Anwendungen als robust und effizient erwiesen haben. Zur Implementierung in OptiSLang wird der *Strength Pareto Evolutionary Algorithm* (SPEA) ausführlich beschrieben und Erweiterungen für dessen Anwendung vorgeschlagen. Anhand verschiedener Testprobleme wird die Eignung des Algorithmus zur Lösung von Mehrzieloptimierungsproblemen untersucht.

Einen wesentlichen Bestandteil dieser Arbeit bildet die Implementierung des Optimierungsalgorithmus in OptiSLang. Die Dokumentation der zugehörigen Eingabedateien befindet sich im Anhang.

2

Mehrzieloptimierung

2.1 Problemformulierung

Ein Mehrzieloptimierungsproblem zeichnet sich durch das Vorhandensein einer Anzahl verschiedener Zielfunktionen aus, die es entweder zu minimieren oder zu maximieren gilt. Zusätzlich können auch Zwangsbedingungen formuliert werden, die von jeder möglichen Lösung erfüllt werden müssen. Ein Mehrzieloptimierungsproblem kann in seiner allgemeinen Form wie folgt beschrieben werden:

$$\begin{aligned} \text{minimize/maximize: } & f_m(\mathbf{x}), & m = 1, 2, \dots, M; \\ \text{subject to: } & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \dots, J; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \dots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \dots, n. \end{aligned} \quad (2.1)$$

Hierbei ist $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ der Vektor der Designvariablen. Er enthält die n Entscheidungsgrößen, die eine Lösung beschreiben. Die Variablen können kontinuierlich, diskret oder binär formuliert werden und sind oftmals durch Nebenbedingungen beschränkt. Dies können sowohl Bedingungen der Ungleichheit $g_j(\mathbf{x}) \geq 0$ als auch Bedingungen der Gleichheit $h_k(\mathbf{x}) = 0$ sein, wobei letztere auch durch zwei Ungleichheitsbedingungen ausgedrückt werden können. Weiterhin sind Designvariablen meist durch eine untere und eine obere Schranke begrenzt. Nebenbedingungen spiegeln oft begrenzte Ressourcen, Benutzeranforderungen oder Grenzen für die Gültigkeit des Berechnungsmodells bezüglich des Optimierungsproblems wider. Die Lösungen, die sowohl die Nebenbedingungen als auch die Variablen Grenzen erfüllen, bilden zusammen den möglichen n -dimensionalen Designvariablenraum X_f :

$$X_f := \{\mathbf{x} \in X : g_j(\mathbf{x}) \geq 0 \wedge h_k(\mathbf{x}) = 0\}. \quad (2.2)$$

Jeder Lösung \mathbf{x} wird über die M Zielfunktionen ein Vektor $\mathbf{f}(\mathbf{x}) = \mathbf{z} = (z_1, z_2, \dots, z_M)^T$ zugeordnet, der einen Punkt im M -dimensionalen Zielfunktionsraum darstellt.

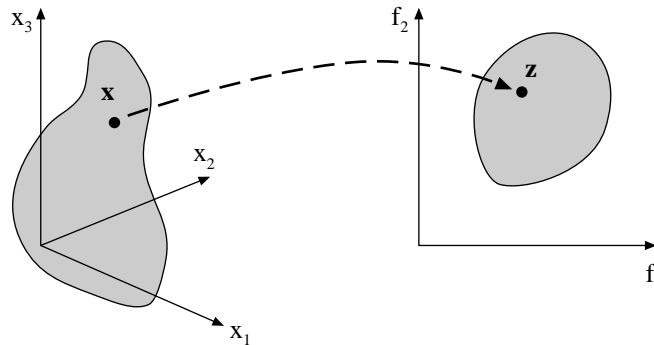


Abbildung 2.1: Zusammenhang zwischen Designvariablenraum (links) und Zielfunktionsraum (rechts).

tionsraum beschreibt. Dies bildet den entscheidenden Unterschied zum Einzielloptimierungsproblem, bei dem nur eine Zielfunktion existiert. Der Zusammenhang zwischen Designvariablenraum und Zielfunktionsraum soll durch Abb. (2.1) verdeutlicht werden.

Das Ziel des Optimierungsprozesses ist es schließlich, Vektoren aus dem Designvariablenraum zu finden, die die Randbedingungen erfüllen und für deren Parameter die Zielfunktionen optimale Werte annehmen. Im Rahmen dieser Arbeit soll dieses Optimum immer durch ein Minimierungsproblem der Zielfunktionen charakterisiert werden. Diese Festlegung schließt jedoch Maximierungsprobleme nicht aus, da diese folgendermaßen durch ein Minimierungsproblem formuliert werden können:

$$\max(f(\mathbf{x})) = -\min(-f(\mathbf{x})). \quad (2.3)$$

2.2 Pareto-Optimum

Um das Optimierungsproblem (2.1) lösen zu können, ist es erforderlich, ein Qualitätskriterium zu finden, das die verschiedenen Zielfunktionen berücksichtigt. Ein solches Kriterium ist das der Pareto-Dominanz, die nach dem italienischen Ökonomen Vilfredo Pareto¹ benannt ist. Für zwei Entscheidungsvektoren \mathbf{a} und \mathbf{b} gilt folgendes:

Definition 1: Eine Lösung $\mathbf{a} \in X$ dominiert eine Lösung $\mathbf{b} \in X$ dann und nur dann, wenn sie in allen Zielfunktionen überlegen oder gleich ist und in mindestens einer Zielfunktion überlegen ist.

¹Vilfredo Pareto (1848–1923) trug durch Studien zur Einkommensverteilung und durch die Einführung des Konzepts der Pareto-Effizienz entscheidend zur Entwicklung der Wirtschaftswissenschaften bei. Er entwickelte u.a. das Gebiet der Mikroökonomie.

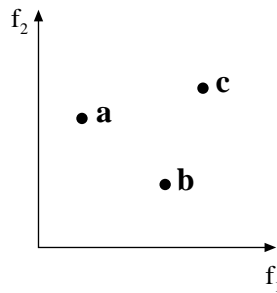


Abbildung 2.2: Das Prinzip der Pareto-Dominanz für ein Minimierungsproblem mit zwei Zielfunktionen: Lösung **a** dominiert Lösung **c**, da **a** in beiden Kriterien überlegen ist. Lösung **a** ist indifferent bezüglich **b**, da jede Lösung in jeweils einem Kriterium der anderen überlegen ist.

Dieser Zusammenhang wird folgendermaßen ausgedrückt:

$$\mathbf{a} \succ \mathbf{b} \Leftrightarrow \begin{aligned} &\forall i \in \{1, 2, \dots, m\} : f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \\ &\wedge \exists j \in \{1, 2, \dots, m\} : f_j(\mathbf{a}) < f_j(\mathbf{b}) \end{aligned} \quad (2.4)$$

Definition 2: Eine Lösung **a** ist indifferent gegenüber einer Lösung **b** dann und nur dann, wenn keine der beiden die jeweils andere dominiert.

Durch das Kriterium der Pareto-Dominanz läßt sich unter verschiedenen Entscheidungsvektoren eine partielle Ordnung herstellen, was durch Abb. (2.2) verdeutlicht werden soll. Pareto-optimal ist eine Lösung dann, wenn es keinen Entscheidungsvektor gibt, der für ein Zielkriterium eine Verbesserung darstellen würde, ohne gleichzeitig mindestens ein anderes zu verschlechtern. Anders ausgedrückt ist eine Lösung dann Pareto-optimal, wenn sie von keiner anderen Lösung dominiert wird. Für eine nicht dominierte Lösung $\mathbf{x} \in X_f$ in bezug auf eine Menge $A \subseteq X_f$ gilt:

$$\nexists \mathbf{a} \in A : \mathbf{a} \succ \mathbf{x} \quad (2.5)$$

Sind alle Zielfunktionen gleichrangig und wird a priori kein Vorzug eines der Optimierungskriterien festgelegt, ist die Dominanz einer Lösung die einzige Möglichkeit festzustellen, ob sie einer anderen überlegen ist. Demzufolge enthält die nicht dominierte Teilmenge aus den möglichen Lösungen X_f die besten Lösungen für das Mehrzieloptimierungsproblem. Diese Lösungen werden auch als *Pareto-Set* bezeichnet. Die korrespondierenden Punkte im Zielfunktionsraum bilden die *Pareto-Front*. Abhängig von der Relation der verschiedenen Zielfunktionen können eine oder mehrere Pareto-Lösungen existieren.

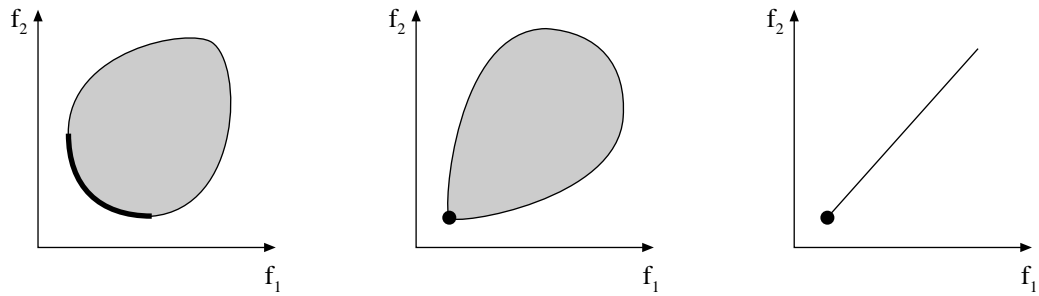


Abbildung 2.3: Das Verhältnis zweier Zielfunktionen für ein Minimierungsproblem: konkurrierende Zielfunktionen mit einer Pareto-Front, bestehend aus verschiedenen Lösungen (links), und korrelative Zielfunktionen mit nur einer Pareto-optimalen Lösung (Mitte und rechts).

Definition 3: Das Verhältnis zweier Zielfunktionen wird als korrelativ bezeichnet, wenn das zugehörige Pareto-Set nur eine Lösung enthält. Das Verhältnis ist konkurrierend, wenn mehr als eine Lösung Pareto-optimal ist.

Das Verhältnis zwischen verschiedenen Zielfunktionen ist in Abb. (2.3) dargestellt. Anhand der Anzahl der Lösungen des Pareto-Sets läßt sich eine Aussage über die Relation verschiedener Zielfunktionen treffen. Für korrelative Zielfunktionen führt die Optimierung eines Kriteriums zwangsläufig auch zur Optimierung des anderen Kriteriums. Das Ergebnis ist eine einzige Pareto-optimale Lösung. Demnach ist dieses Optimierungsproblem identisch mit einem Einzielloptimierungsproblem. Im Rahmen dieser Arbeit sollen deshalb nur Optimierungsprobleme mit konkurrierenden Zielfunktionen behandelt werden. Da es sich bei derartigen Problemen um eine Menge Pareto-optimaler Lösungen handelt, welche jeweils einen Kompromiß zwischen den konkurrierenden Zielfunktionen darstellen, lassen sich für die Mehrzieloptimierung folgende Anforderungen formulieren:

- Finde eine Anzahl an Lösungen nahe den Pareto-optimalen Lösungen.
- Finde Lösungen, die verschieden genug sind, um die gesamte Ausbreitung der Pareto-Front zu repräsentieren.

Obwohl die Mehrzieloptimierung viele optimale Lösungen hervorbringt, ist der Anwender wie auch bei der Einzielloptimierung meist nur an einer Lösung interessiert. Da die Pareto-Front verschiedene Kompromißlösungen enthält, stellt sich für den Optimierer die Frage, welche der Lösungen zu wählen ist. Um die Entscheidung für eine Lösung zu ermöglichen, müssen a posteriori Präferenzen definiert werden. Dies erfordert der Problemstellung übergeordnete Informationen, die oft nicht technischer Natur und erfahrungsbasiert

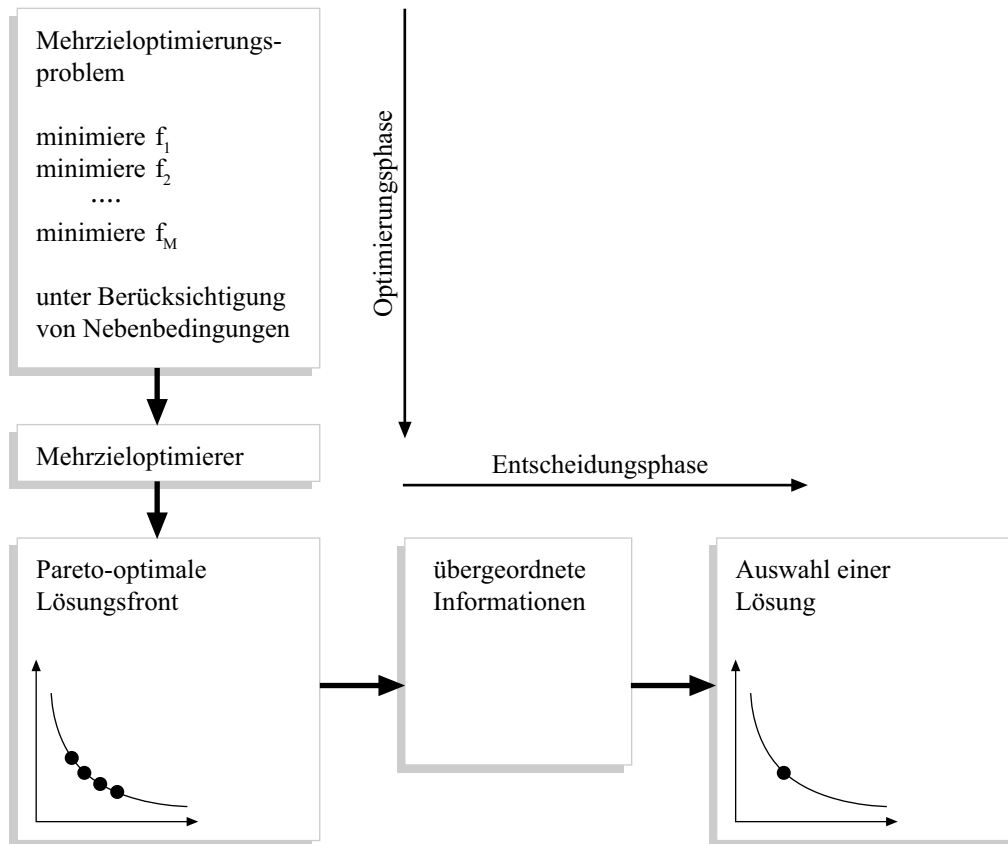


Abbildung 2.4: Schematischer Ablauf einer Mehrzieloptimierung.

sind. Eine Entscheidungsfindung anhand einer Pareto-Front erfordert zwar ein hohes Maß an Problemkenntnis, das Wissen um den Zusammenhang der Zielfunktionen für Pareto-optimale Kompromißlösungen ermöglicht es jedoch auch, eine den eigenen Anforderungen entsprechende Lösung auszuwählen. Die ideale Mehrzieloptimierung verläuft demnach wie folgt:

- Finde verschiedene optimale Kompromißlösungen mit einer großen Bandbreite der Werte der Zielfunktionen.
- Wähle eine dieser Lösungen anhand übergeordneter Informationen aus.

Abb. (2.4) veranschaulicht schematisch den Ablauf einer Mehrzieloptimierung. Würde man das Optimierungsproblem durch eine einzige Zielfunktion ersetzen, würde Schritt 1 zu einer optimalen Lösung führen, und die Entscheidungsfindung aus Schritt 2 würde entfallen, weshalb man die Einzeloptimierung auch als Spezialfall der Mehrzieloptimierung betrachten kann.

2.3 Methoden zur Mehrzieloptimierung

Das Gebiet der Mehrzieloptimierung hat innerhalb der letzten Jahre stark an Bedeutung gewonnen und viele Wissenschaftler dazu veranlaßt, Lösungsmethoden zu entwickeln. Auch wenn die Konzepte zur Mehrzieloptimierung heute interdisziplinär entwickelt werden, liegen die Ursprünge in den Forschungsgebieten des ökonomischen Gleichgewichts, der Spieltheorie² und der mathematischen Optimierung. Die Anwendung der Mehrzieloptimierung in Bereichen der Ingenieurwissenschaften ist durch die wachsenden Anforderungen der Optimierungsprobleme heute weitestgehend etabliert.

Eine detaillierte Zusammenstellung der existierenden Konzepte und Algorithmen zur Mehrzieloptimierung liefern (Marler und Arora (2003)). Sie klassifizieren die Methoden nach der Art der Festlegung von Präferenzen seitens des Entscheidungsfinders:

A-priori-Artikulation von Präferenzen impliziert die Festlegung der relativen Bedeutung verschiedener Zielfunktionen vor Durchführung der Optimierung. Dabei wird das Mehrziel- in ein Einzieloptimierungsproblem überführt.

A-posteriori-Artikulation von Präferenzen erfordert die Selektion einer einzelnen Lösung aus einer Menge mathematisch äquivalenter Lösungen nach Ablauf der Optimierung. Die gewählte Lösung spiegelt dann direkt die Präferenzen des Entscheidungsfinders wider.

Progressive Artikulation von Präferenzen verlangt eine kontinuierliche Interaktion des Entscheidungsfinders während des Ablaufs des Optimierungsalgorithmus. Durch Präferenzen des Anwenders wird die Menge potentieller Lösungen reduziert und nur eine Teilmenge berücksichtigt, was zur Verringerung des Rechenaufwands führt. Ein Nachteil dieser Verfahrensweise ist, daß der Optimierungsprozeß nicht unabhängig vom Anwender ablaufen kann. Diese Art der Interaktion ist deshalb nicht für häufig zu wiederholende Optimierungsaufgaben geeignet.

Da die Vielzahl der existierenden Methoden eine vollständige Erläuterung an dieser Stelle nicht erlaubt, soll die folgende Auswahl grundlegende Prinzipien verdeutlichen.

Methoden mit A-priori-Artikulation von Präferenzen erfordern vom Entscheidungsfinder die Festlegung von Restriktionen und Wichtungen. Die meisten dieser Methoden verwenden eine Nutzwertfunktion, die für jede Zielfunktion definiert wird. Sie repräsentiert die relative Bedeutung der jeweiligen

²Die Spieltheorie (engl. game theory) ist ein Teilgebiet von Mathematik, Operations Research und der Wirtschaftswissenschaften. Sie beschäftigt sich mit der Analyse von Handlungsstrategien in Systemen mit vorgegebenen Regeln („Spielen“). Dazu untersucht die Spieltheorie vorhergesagtes und tatsächliches Verhalten von Akteuren in Spielen und leitet optimale Strategien her.

Zielfunktion. Eine Nutzwertfunktion für ein Mehrzieloptimierungsproblem wird wie folgt definiert:

$$U = \sum_{i=1}^M u_i [F_i(\mathbf{x})] \quad (2.6)$$

Hierbei ist $u_i [F_i(\mathbf{x})]$ die Nutzwertfunktion einer Zielfunktion.

Methoden des gewichteten globalen Kriteriums verwenden Wichtungen als skalare Faktoren der einzelnen Zielfunktionen. Die gebräuchlichste Form der Nutzwertfunktion ist die der *gewichteten exponentialen Summen*:

$$U = \sum_{i=1}^M w_i [F_i(\mathbf{x})]^p \quad (2.7)$$

$$U = \sum_{i=1}^M [w_i F_i(\mathbf{x})]^p \quad (2.8)$$

Die Wichtungen \mathbf{w} und der Exponent p werden vom Anwender a priori festgelegt. Durch die Wichtungen können Präferenzen für bestimmte Zielfunktionen definiert, durch deren Variation aber auch eine größere Vielfalt an Pareto-Lösungen erreicht werden. Bei der Bestimmung der Wichtungsfaktoren müssen folgende Bedingungen erfüllt sein:

$$\sum_{i=1}^M w_i = 1; \quad w_i > 0, \quad i = 1, \dots, M. \quad (2.9)$$

Die Minimierung der Nutzwertfunktion führt zu einer Pareto-optimalen Lösung. Am Beispiel der gewichteten Summe zweier Zielfunktionen soll dies verdeutlicht werden.

$$U = w_1 F_1(\mathbf{x}) + w_2 F_2(\mathbf{x}) \quad (2.10)$$

Damit U ein Minimum annimmt, muß der Gradient von U null sein.

$$\nabla_x U = w_1 \nabla_x F_1(\mathbf{x}) + w_2 \nabla_x F_2(\mathbf{x}) = 0 \quad (2.11)$$

Setzt man die positive Definitheit der Wichtungen (2.9) voraus, sind die Gradienten der Zielfunktionen für einen Pareto-optimalen Punkt kollinear mit entgegengesetzten Richtungen. Somit ist die Linearkombination beider Gradienten gleich null. In Übereinstimmung mit der Definition des Pareto-Optimums bedeutet für eine Lösung von (2.11) die Verbesserung eines Kriteriums eine Verschlechterung des anderen Kriteriums.

Eine grafische Interpretation der Nutzwertfunktion liefert Abb. (2.5). Für festgelegte Wichtungen wird eine Pareto-optimale Lösung dort gefunden, wo die Nutzwertfunktion mit einem kleinstmöglichen Wert den möglichen Zielfunktionsraum schneidet. Um verschiedene Punkte der Pareto-Front zu erhalten, muß die Optimierung der Nutzwertfunktion für verschiedene Wichtungskombinationen durchgeführt werden. Ist die Pareto-Front jedoch nicht

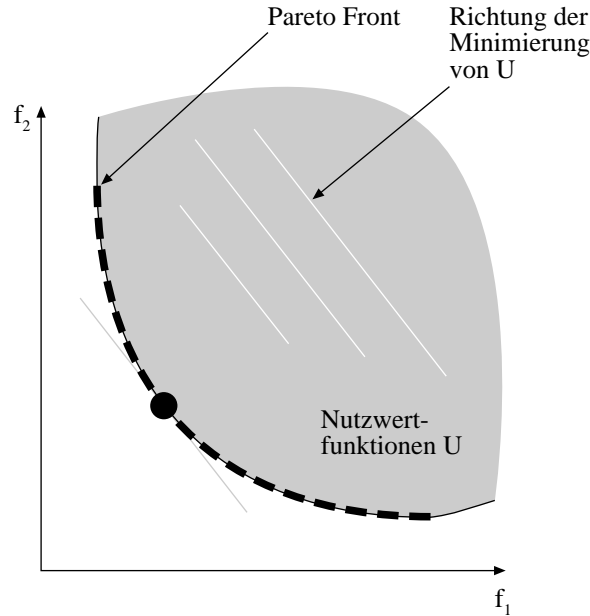


Abbildung 2.5: Nutzwertfunktionen für zwei gewichtete Kriterien im Zielfunktionsraum.

konvex, können durch diese Methode nicht alle Pareto-Lösungen gefunden werden. Weiterhin ist die Methode von den Konvergenzeigenschaften des verwendeten Einzieloptimierers abhängig.

Soll die Front der Pareto-optimalen Lösungen vollständig und gleichmäßig abgebildet werden, sind die eben beschriebenen Methoden nur bedingt geeignet. Sie erfordern eine Vielzahl von Optimierungsläufen mit einer kontinuierlichen Variation der Methodenparameter. Verfahren mit A-posteriori-Artikulation von Präferenzen haben ein Set von Lösungen zum Ziel, die die gesamte Pareto-Front repräsentieren.

Eines dieser Verfahren ist die *normal bounded intersection* Methode (Das und Dennis (1996)). Sie wird folgendermaßen formuliert:

$$\begin{aligned}
 &\text{find : } \mathbf{x} \in \mathbf{X}, \lambda \\
 &\text{to maximize : } \lambda \\
 &\text{subject to : } \mathbf{\Phi} \mathbf{w} + \lambda \mathbf{n} = \mathbf{F}(\mathbf{x}) - \mathbf{F}^\circ
 \end{aligned} \tag{2.12}$$

Hierbei ist $\mathbf{\Phi}$ eine $k \times k$ Matrix, die spaltenweise mit den Vektoren $\mathbf{F}(\mathbf{x}_i^*) - \mathbf{F}^\circ$ belegt ist. Der Vektor $\mathbf{F}(\mathbf{x}_i^*)$ enthält die Zielfunktionswerte, die an dem Punkt ermittelt werden, an dem die i -te Zielfunktion ein Minimum besitzt. Der Vektor \mathbf{F}° bezeichnet den *idealen Punkt* im Zielfunktionsraum, an dem jede Zielfunktion einen minimalen Wert annimmt.

$$\forall i = 1, \dots, k : F_i^\circ = \text{minimum}\{F_i(\mathbf{x}) | \mathbf{x} \in \mathbf{X}\} \tag{2.13}$$

Demnach sind die Elemente der Hauptdiagonale der Matrix Φ gleich null. Der Vektor \mathbf{w} enthält Wichtungen, die die Bedingungen aus (2.9) erfüllen müssen. Der *Quasi-Normalenvektor* \mathbf{n} ergibt sich aus $\mathbf{n} = -\Phi\mathbf{e}$, wobei \mathbf{e} ein Einheitsvektor im M -dimensionalen Zielfunktionsraum ist. Da alle Elemente von Φ positiv sind, zeigt \mathbf{n} auf den Ursprung des Zielfunktionsraums. Durch die schrittweise Veränderung von \mathbf{w} erzielt die Methode Pareto-optimale Punkte als Lösung von (2.12), die gleichmäßig verteilt sind und die Pareto-Front vollständig repräsentieren.

Weitere Methoden, die die gesamte Pareto-Front abbilden können sind *Physical Programming*, *Fuzzy Methods* und *genetische Algorithmen*. Letztere sollen im folgenden Kapitel auf ihre Eignung zur Lösung von Mehrzieloptimierungsproblemen untersucht werden.

3

Evolutionäre Strategien zur Mehrzieloptimierung

3.1 Grundlagen Evolutionärer Algorithmen

Evolutionäre Algorithmen (EA) sind stochastische Suchverfahren, die nach dem Vorbild der biologischen Evolution Prinzipien der Anpassung, Selektion und Variation verwenden. In einem EA durchsucht eine Population künstlicher Individuen den Variablenraum. Ziel ist es, schrittweise bessere Lösungen zu finden und auf diese Weise Optima zu bestimmen oder zu approximieren. Jedes Individuum repräsentiert eine mögliche Lösung des Optimierungsproblems über einen Satz von Genen, der als Chromosom bezeichnet wird. Ein Gen enthält die codierte Information über eine Designvariable. Die Codierung erfolgt meist über eine binäre Zahlenfolge einer bestimmten Länge. Es ist jedoch auch möglich, die Variablen durch reelle Zahlen zu repräsentieren.

Ein EA beginnt mit der zufälligen Initialisierung einer Population der Größe μ . Daraufhin erfolgt die Bewertung der Individuen. Dabei wird jedem Individuum ein Fitneßwert, basierend auf dem Zielfunktionswert im Vergleich zu allen anderen Individuen der Population, zugewiesen. Ein hoher Fitneßwert bedeutet eine gute Anpassung an die Problemstellung. Nach der Initialisierungsphase beginnt der eigentliche Zyklus des Algorithmus. Dabei repräsentiert jeder Durchlauf eine Generation g .

Zunächst erfolgt die Selektion der Individuen, die für die Reproduktion vorgesehen sind. Über den Selektionsoperator wird der Evolution die eigentliche Richtung verliehen, da gute Zustände konserviert werden. Allerdings wirkt die Selektion gegensätzlich zur Diversität der Population, da einzelne Individuen mehrfach zur Reproduktion ausgewählt werden können. Gebräuchliche Selektionsverfahren sind Abschneideselektion¹, Turnierselektion² und Roulette-Selektion.

¹engl. truncation selection.

²engl. tournament selection.

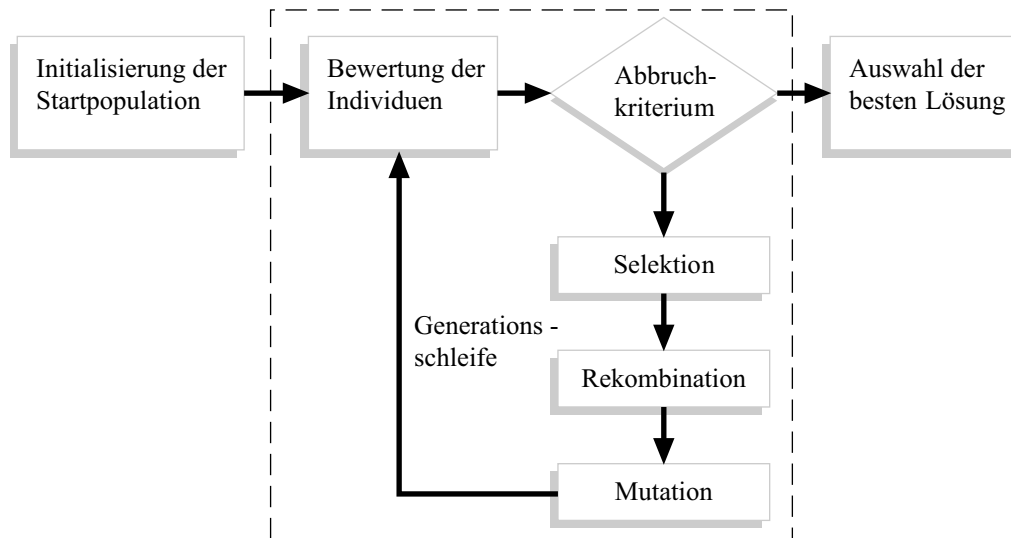


Abbildung 3.1: Prinzipieller Ablauf eines evolutionären Algorithmus.

Danach werden die Variationsoperatoren Rekombination und Mutation angewandt, die der Diversität der Population dienen. Durch die Rekombination der ausgewählten Elternindividuen werden deren genetische Informationen gemischt und λ Nachkommen erzeugt. Als Rekombinationsoperatoren finden häufig das n -Punkt-Cross-over und das uniforme Cross-over Anwendung. Anschließend werden durch die Mutation die genetischen Informationen der Nachkommen oder der gesamten Population gemäß einer vorgegebenen Wahrscheinlichkeitsverteilung zufällig abgeändert. Schließlich erfolgt die Bewertung der Nachkommen, und eine neue Population von μ Individuen wird anhand von Fitneßwerten selektiert. Der Generationszähler wird um eins erhöht. Abb. (3.1) veranschaulicht den prinzipiellen Ablauf eines EA.

Ehemals unabhängig voneinander entstandene Hauptrichtungen der EA sind *Genetische Algorithmen* (GA), nach Grundlagen von Holland (1975) und weiterentwickelt von (Goldberg (1989)), *Evolutionäre Strategien* (ES), eingeführt von Rechenberg (1973) und Schwefel (1981), sowie *Evolutionäre Programmierung* (EP), entwickelt von Fogel et al. (1966). Basis der Theorie der GA ist das Schematheorem und die daraus ableitbare Baustein-Hypothese³. Die Theorie der ES ist zielfunktionsbezogen und baut auf der analytischen Behandlung der Erfolgswahrscheinlichkeit und der Fortschrittsgeschwindigkeit auf. Zielsetzung von EP ist das Erreichen von emergentem Verhalten durch Selbstorganisation. Eine detaillierte Beschreibung der genannten Verfahren ist u.a. in (Bäck und Schwefel (1996)) zu finden.

Alle drei Algorithmen haben sich als geeignet erwiesen, für gegebene kom-

³engl. building block hypothesis.

plexe, multimodale, nicht differenzierbare und unstetige Suchräume näherungsweise optimale Lösungen zu finden. Hierbei erweist sich die Unabhängigkeit dieser Methoden von Gradienteninformation als entscheidender Vorteil gegenüber gradientenbasierten Optimierungsverfahren.

3.2 Multikriterielle Evolutionäre Algorithmen

Die Verwendung von EA zur Lösung von Mehrzieloptimierungsproblemen hat sich heute sowohl als Forschungsgebiet als auch in der Praxis etabliert. Der Vorteil liegt in der parallelen Suche nach mehreren Pareto-optimalen Lösungen in einem einzigen Simulationslauf. Zahlreiche wissenschaftliche Publikationen und unterschiedliche Implementierungen verdeutlichen die Bemühungen, aber auch den Bedarf nach einem stabilen, anwendbaren Algorithmus. Ein Überblick über die Vielzahl von Veröffentlichungen wurde von (Coello (2004)) im Internet zusammengestellt.

Der erste EA zu Mehrzieloptimierung (MOEA⁴) war der *Vector Evaluated GA* (VEGA) (Schaffer (1985)). Der Algorithmus verwendet einen im Vergleich zur Einzieloptimierung modifizierten Selektionsoperator. In jeder Generation werden proportional zu den m Zielfunktionen m Subpopulationen generiert. Die Selektion erfolgt nach dem Prinzip der Spezialisierung. Aus jeder Subpopulation werden die Individuen ausgewählt, die bezüglich der zugehörigen Zielfunktion die beste Fitneß besitzen. Eine gleichzeitige Bewertung der Individuen bezüglich aller Kriterien ist nicht vorgesehen.

Weitere MOEA basieren auf einem Dominanzkriterium zur Bewertung der Individuen. Hierbei werden die Werte aller Zielfunktionen und alle Individuen der Population berücksichtigt. Zusätzlich werden verschiedene Strategien angewandt, die die Diversität der Population im Designvariablen- oder Zielfunktionsraum gewährleisten sollen. Die wichtigsten MOEA-Realisierungen sind der *Multiobjective GA* (MOGA) (Fonseca und Fleming (1993)), der *Niched Pareto GA* (NPGA) (Horn und Nafpliotis (1993)), der *Nondominated Sorting GA* (NSGA-II) (Deb u. a. (2000)) und der *Strength Pareto EA* (SPEA) (Zitzler und Thiele (1998)). In den folgenden Abschnitten sollen die verschiedenen Konzepte zur Bewertung, Selektion und Gewährleistung der Diversität der genannten Algorithmen erläutert werden. Eine verbesserte Variante des Strength Pareto EA (SPEA2) (Zitzler u. a. (2001)) und dessen Implementierung wird im Kapitel (4) detailliert beschrieben.

⁴engl. Multiobjective Evolutionary Algorithm.

3.3 Bewertung und Selektion

3.3.1 Selektion durch Vektorauswertung

Eine erste Strategie zur Bewertung und Selektion von Individuen in einem populationsbasierten Suchprozeß verzichtet auf die Anwendung des Kriteriums der Pareto-Dominanz. Statt dessen erfolgt eine *fitneßproportionale Selektion durch wechselnde Zielfunktionen*, implementiert im Vector Evaluated GA (VEGA) (Schaffer (1985)). Bei einem Optimierungsproblem mit M Kriterien wird die Population aus N Individuen in M Subpopulationen der Größe N/M aufgesplittet. Für jede der Subpopulationen wird als Selektionskriterium jeweils nur eine Zielfunktion betrachtet. Cross-over- und Mutationsoperatoren werden angewandt, nachdem die selektierten Individuen wieder zusammengeführt und vermischt worden sind. Dabei erfolgt eine Mittelwertbildung der Fitneßkomponenten für die einzelnen Zielfunktionen. Es konnte gezeigt werden, daß die voraussichtliche Anzahl an Nachkommen eines Elternindividuums mit der Summe der erwarteten Nachkommen dieses Individuums bezüglich jeder Zielfunktion übereinstimmt. Dieser proportionale Zusammenhang resultiert in einer zu erwartenden Fitneß, die einer gewichteten Linearkombination der Zielfunktionen entspricht, wobei die Wichtungen abhängig von der Verteilung der Population in der jeweiligen Generation sind.

Ein Nachteil dieser Selektionsstrategie ist die bevorzugte Auswahl von Individuen, die für eine einzelne Zielfunktion besonders gute Fitneßwerte besitzen. Dabei werden Lösungen, die bezüglich aller Kriterien nur mittelmäßige Fitneß besitzen, für das Erzielen von Kompromißlösungen aber hilfreich wären, nicht berücksichtigt. Dies führt zur unerwünschten Spezialisierung der Population auf extreme Bereiche der optimalen Front. Dies wird besonders bei konkaven Pareto-Fronten deutlich (s. Abb. (3.2)), bei denen sich die mangelnde Fähigkeit der Methode, alle Lösungen in konkaven Bereichen der Front finden zu können, als entscheidender Schwachpunkt erweist.

3.3.2 Dominanzbasierte Verfahren

Fitneßzuweisung durch Rangbildung

Das am häufigsten verwendete Selektionsverfahren in MOEA ist das der dominanzbasierten Rangbildung. Eine Variante dieses Verfahrens wurde von (Fonseca und Fleming (1993)) für den Multiobjective GA (MOGA) formuliert. Für ein Individuum \mathbf{x}_i der Generation t , das von einer Anzahl $p_i^{(t)}$ von Individuen der aktuellen Generation dominiert wird, ergibt sich der Rang wie folgt:

$$\text{Rang}(\mathbf{x}_i, t) = 1 + p_i^{(t)} \quad (3.1)$$

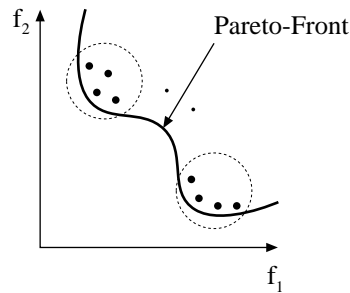


Abbildung 3.2: Konzentration der selektierten Individuen auf lokal konvexe Bereiche der Pareto-Front bei der Selektion durch wechselnde Zielfunktionen.

Demnach besitzen alle nicht dominierten Individuen der Population den Rang 1. Alle dominierten Lösungen erhalten einen Rang, der größer 1 ist und von der jeweiligen Anzahl der sie dominierenden Lösungen abhängig ist. Daß nicht alle möglichen Werte eines Ranges zwischen bestem und schlechtestem Individuum vorkommen müssen, wird anhand Abb. (3.3) deutlich, wo Rang 4 nicht existiert. Für die rangbasierte Fitnesszuweisung wurde folgender Ablauf vorgeschlagen:

1. Sortiere die Individuen der Population entsprechend ihres Ranges.
2. Weise den Individuen Fitneßwerte durch Interpolation zwischen dem Besten (Rang 1) und dem Schlechtesten (Rang $n^* \leq N$) zu. Die Interpolation erfolgt üblicherweise linear.
3. Bilde für Individuen mit identischem Rang den Mittelwert ihrer Fitneßwerte. Dies gewährleistet gleichmäßigen Selektionsdruck, formuliert durch die Interpolationsfunktion, für gleichrangige Individuen.

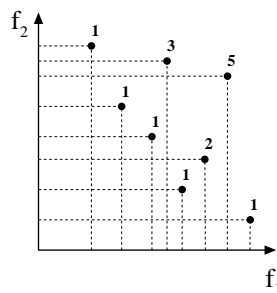


Abbildung 3.3: Dominanzbasierte Rangbildung für ein Minimierungsproblem.

Pareto Domination Tournaments

Das von (Horn und Nafpliotis (1993)) entwickelte Selektionsverfahren für den Niche Pareto GA (NPGA) verwendet das Pareto-Dominanzkriterium im Rahmen einer Turnierselktion. Ausgehend von der binären Relation der Dominanz, wurde die binäre Turnierselktion so erweitert, daß sich nicht nur zwei potentielle Lösungen im direkten Vergleich gegenüberstehen. Statt dessen erfolgt die Selektion eines der beiden Lösungskandidaten durch Dominanzanalyse bezüglich einer zufällig aus der Population ausgewählten Vergleichsmenge. Durch die Anzahl t_{dom} der Individuen der Vergleichsmenge wird hierbei der Selektionsdruck gesteuert. Folgender Pseudocode soll den Ablauf der Pareto-Dominanz Turnierselktion verdeutlichen. Hierbei ist S das Array der N Individuen der Population und $random_pop_index$ das Array der zufällig angeordneten Indizes der Individuen.

```

function selection
/* Returns an individual from the current population  $S$  */
begin
  shuffle(random_pop_index); /* Re-randomize random index array */
  candidate_1 = random_pop_index[1];
  candidate_2 = random_pop_index[2];
  candidate_1_dominated = false;
  candidate_2_dominated = false;
  for comparison_set_index = 3 to  $t_{dom} + 3$  do
/* Select  $t_{dom}$  individuals randomly from  $S$  */
    begin
      comparison_individual = random_pop_index[comparison_set_index];
      if  $S[comparison\_individual]$  dominates  $S[candidate\_1]$ 
        then candidate_1_dominated = true;
      if  $S[comparison\_individual]$  dominates  $S[candidate\_2]$ 
        then candidate_2_dominated = true;
    end /* end for loop */
  if ( candidate_1_dominated AND  $\neg$  candidate_2_dominated )
    then return candidate_2;
  else if (  $\neg$  candidate_1_dominated AND candidate_2_dominated )
    then return candidate_1;
  else
    do sharing;
  end

```

Das Verfahren verzichtet auf eine Pareto-Rangbildung für die gesamte Population. Ein selektiertes Individuum ist nur bezüglich einer Vergleichsmenge nicht dominiert. Liefert die beschriebene Turnierselktion keinen Gewinner, erfolgt die Selektion über ein Fitneßteilverfahren. Die Methode erfordert für die Größe der Vergleichsmenge einen zusätzlich festzulegenden Parameter.

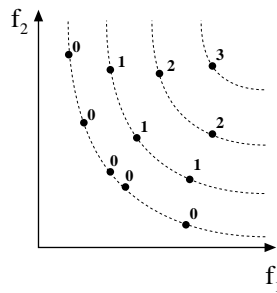


Abbildung 3.4: Bewertung der Individuen durch Bildung nicht dominierter Fronten (NSGA-II).

Nondominated Sorting

Der Nondominated Sorting GA (NSGA-II) (Deb u. a. (2000)) untersucht zur Bewertung eines Individuums dessen Dominanzrelation bezüglich jedes anderen Individuums in der Population. Alle nicht dominierten Lösungen erhalten den Rang 0 und werden vorübergehend aus der Population entfernt. Die wiederum nicht dominierten Lösungen der verbliebenen Population erhalten Rang 1. Dies wird so lange wiederholt, bis allen Lösungen ein Rang zugeordnet ist. Als Ergebnis liefert dieses Bewertungsverfahren Fronten von Lösungen, die jeweils nicht dominiert bezüglich aller Individuen mit höherem Rang sind (s. Abb. (3.4)). Dieses Sortierverfahren wird in (Deb u. a. (2000)) folgendermaßen skizziert.

$F = \text{fast-nondominated-sort}(P)$	F is a set of nondominated fronts
$i = 1$	i is the front counter
until $P \neq \emptyset$	
$F_i = \text{find-nondominated-front}(P)$	find the nondominated front
$P = P \setminus F_i$	remove nondominated solutions
$i = i + 1$	increment the front counter

Der Algorithmus verwendet das Prinzip des Elitismus, um den Verlust von nicht dominierten Lösungen zu vermeiden. Dabei besteht die jeweils aktuelle Generation aus der Population des Archivs und der Population der Nachkommen. Nach der Bewertung aller Individuen wird von den Lösungen mit dem höchsten Rang jeweils die entfernt, die für ein bestimmtes Dichtemaß den geringsten Wert besitzt. Ist die festgesetzte Archivgröße erreicht, bilden die verbliebenen Individuen das Archiv der nächsten Generation. Aus dem Archiv werden dann durch binäre Turnierselktion Individuen zur Reproduktion ausgewählt.

3.4 Diversität der Population

Bei der Mehrzieloptimierung geht es nicht nur darum, eine Anzahl Pareto-optimaler Lösungen zu finden. Zusätzlich sollen diese die Pareto-Front auch näherungsweise repräsentieren. Deshalb soll eine Konzentration der Lösungen auf lokale Optima vermieden werden. Dazu existieren verschiedene Strategien zur Gewährleistung der Diversität der Population, um der u.a. durch Selektionsdruck hervorgerufenen Konzentration von Individuen entgegenzuwirken.

3.4.1 Fitneßteilung

Eine etablierte Methode zur Gewährleistung der Diversität ist die Fitneßteilung, entwickelt von (Goldberg und Richardson (1987)). Das Ziel des Verfahrens ist die Verteilung der Population über eine bestimmte Anzahl verschiedener Hochpunkte im Zielfunktionsraum. Jedem dieser Punkte wird proportional zu seiner Höhe ein Teil der Population zugewiesen.

Um diese Verteilung zu erreichen, wird die Fitneß f_i jedes Individuums durch einen Nischenzähler m_i herabgesetzt. Die geteilte Fitneß ergibt sich durch f_i/m_i . Der Nischenzähler m_i ist ein Maß für die Anhäufung von benachbarten Individuen und wird für jede Lösung bezüglich der gesamten Population folgendermaßen bestimmt:

$$m_i = \sum_{j \in Pop} s[d[i, j]]. \quad (3.2)$$

Hierbei ist $d[i, j]$ die Entfernung zwischen zwei Individuen i und j und $s[d[i, j]]$ eine abnehmende Teilungsfunktion.

$$s[d[i, j]] = \begin{cases} 1 - \frac{d[i, j]}{\sigma_{share}} & \text{für } d[i, j] \leq \sigma_{share} \\ 0 & \text{für } d[i, j] > \sigma_{share} \end{cases} \quad (3.3)$$

Der Parameter σ_{share} entspricht demnach der Entfernung, die zwei Individuen zueinander besitzen müssen, um ihre Fitneß gegenseitig nicht negativ zu beeinflussen. Er wird auch als Nischenradius bezeichnet und sollte anhand der vom Anwender verlangten Abstände zwischen den Punkten der Pareto-Front festgelegt werden. Da Informationen über das Maß dieser Abstände a priori meist nicht zur Verfügung stehen, ist es schwierig, einen Wert für σ_{share} zu wählen. Einen Vorschlag zur Bestimmung des Parameters liefern (Fonseca und Fleming (1993)) indem sie für eine aktuelle Population die von den Pareto-optimalen Lösungen aufgespannte Hyperebene betrachten. Fitneßteilung kann sowohl in Zusammenhang mit fitneßproportionaler Selektion als auch mit Turnierselektion verwendet werden.

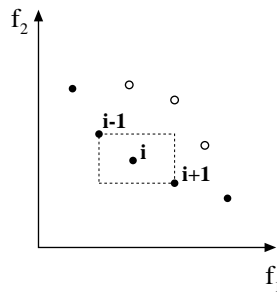


Abbildung 3.5: Dichteabschätzung durch Ermittlung der Abstände eines Punktes i zu seinen benachbarten Punkten $i - 1$ und $i + 1$ im Zielfunktionsraum.

3.4.2 Dichteabschätzung

Das Prinzip der Dichteabschätzung basiert auf der Konstruktion einer näherungsweise Wahrscheinlichkeitsdichte ϕ für eine zufällige Punktmenge X . Daraus lassen sich Wahrscheinlichkeiten für bestimmte Intervalle anhand der Verteilungsfunktion Φ bezüglich X folgendermaßen ermitteln:

$$\begin{aligned} P(a \leq X < b) &= \Phi(b) - \Phi(a) \\ &= \int_a^b \phi(\xi) d\xi; \quad (a < b). \end{aligned} \quad (3.4)$$

In (Silverman (1986)) wird ein Überblick über existierende nichtparametrische Methoden zur Dichteabschätzung gegeben. Ausgehend von der Voraussetzung, daß die Verteilung eine Wahrscheinlichkeitsdichtefunktion ϕ besitzt, wird durch Analyse der gegebenen Daten versucht, diese näherungsweise zu bestimmen.

Eine lokale Variante der Dichteabschätzung wird von (Deb u. a. (2000)) für den NSGA-II verwendet. Für einen Punkt im Zielfunktionsraum wird die durchschnittliche Entfernung der beiden benachbarten Punkte bezüglich jeder Zielfunktion ermittelt. Der daraus resultierende Wert i_{dist} dient zur Abschätzung der Größe des größten achsenparallelen Hyperquaders im M -dimensionalen Zielfunktionsraum, der einen Punkt i umgibt, ohne einen anderen Punkt der Population mit einzuschließen (s. Abb. (3.5)).

Die Ermittlung der Anhäufungsentfernung i_{dist} erfordert das Sortieren der Population bezüglich der Werte jeder Zielfunktion in aufsteigender Reihenfolge. Folgender Algorithmus skizziert dieses Verfahren.

```

crowding-distance-assignment( $\mathcal{I}$ )
 $l = |\mathcal{I}|$                                 number of solutions in  $\mathcal{I}$ 
for each  $i$ , set  $\mathcal{I}[i]_{dist} = 0$           initialize distance
for each objective  $m$ 
     $\mathcal{I} = \text{sort}(\mathcal{I}, m)$           sort using each objective
     $\mathcal{I}[1]_{dist} = \mathcal{I}[l]_{dist} = \infty$     so that boundary points are always selected
    for  $i = 2$  to  $(l - 1)$                 for all other points
         $\mathcal{I}[i]_{dist} = \mathcal{I}[i]_{dist} + (\mathcal{I}[i + 1].m - \mathcal{I}[i - 1].m)$ 

```

Nachdem allen Individuen ein Wert für die Anhäufungsentfernung zugewiesen ist, steht neben dem Fitneßwert ein weiteres Selektionskriterium zur Verfügung. Lösungen mit einer großen Anhäufungsentfernung sind von wenigen Individuen in unmittelbarer Nachbarschaft umgeben und sind zur Gewährleistung der Diversität bei der Selektion zu bevorzugen. Gleiches gilt auch für die Randpunkte der Population.

4

Strength Pareto Evolutionary Algorithm

Der Strength Pareto Evolutionary Algorithm (SPEA), eingeführt von (Zitzler und Thiele (1998)), verbindet verschiedene etablierte Techniken aus existierenden EAs in einem einzigen Algorithmus. Er hat sich anhand vergleichender Studien als geeigneter Algorithmus zur evolutionären Mehrzieloptimierung erwiesen. Er erfordert dabei eine minimale Anzahl von Steuerungsparametern. SPEA läßt sich durch folgende Grundprinzipien charakterisieren:

- Das Prinzip des Elitismus wird durch Speicherung der nicht dominierten Individuen in einem externen Archiv angewendet.
- Die Fitneßzuweisung erfolgt nach dem Konzept der Pareto-Dominanz.
- Die Gewährleistung der Diversität der Population wird durch Dichteabschätzung realisiert.

Obwohl SPEA in seiner ursprünglichen Formulierung bereits gute Ergebnisse lieferte, wurde der Algorithmus von (Zitzler u. a. (2001)) weiterentwickelt. Die folgenden Erläuterungen zu Ablauf, Bewertungs- und Selektionsverfahren beziehen sich deshalb auf SPEA2.

4.1 Ablauf des Algorithmus

SPEA2 Main Loop

Input: α Archivgröße
 μ Anzahl der Eltern = Anzahl der Nachkommen
 T maximale Anzahl der Generationen
Output: A nicht dominierte Lösungsmenge

- Schritt 1: **Initialisierung:** Erzeuge eine zufällig initialisierte Population P_0 der Größe α und ein leeres externes Archiv $\bar{P}_0 = \emptyset$. Setze den Generationszähler $t = 0$.
- Schritt 2: **Fitneßzuweisung:** Ermittle die Fitneßwerte aller Individuen in P_t und \bar{P}_t .
- Schritt 3: **umgebungsbedingte Selektion:** Kopiere alle nicht dominierten Individuen aus P_t und \bar{P}_t nach \bar{P}_{t+1} . Übersteigt die Größe von \bar{P}_{t+1} den Wert α , reduziere \bar{P}_{t+1} mittels des Abschneideoperators. Ist die Größe von \bar{P}_{t+1} kleiner als α , fülle \bar{P}_{t+1} mit dominierten Individuen aus P_t und \bar{P}_t auf.
- Schritt 4: **Abbruch:** Wenn $t \geq T$ oder ein anderes Abbruchkriterium erfüllt ist, bilden die nicht dominierten Lösungen in \bar{P}_{t+1} die nicht dominierte Lösungsmenge A .
- Schritt 5: **Elternselektion:** Wähle aus \bar{P}_{t+1} Individuen zur Reproduktion durch binäre Turnirselektion mit Zurücklegen aus.
- Schritt 6: **Variation:** Wende Rekombinations- und Mutationsoperatoren auf die Elternpopulation an. P_{t+1} ergibt sich aus der daraus resultierenden Population. Erhöhe den Generationszähler ($t = t + 1$) und gehe zu Schritt 2.

Der Algorithmus arbeitet mit einem Archiv festgelegter Größe, in dem die besten Individuen der vorhergehenden Generation enthalten sind. Sie überleben den Generationssprung, ohne durch Variationsoperatoren verändert zu werden. Das Archiv bildet einen Teil der aktuellen Population. Aus dem Archiv werden wiederum eine bestimmte Anzahl von Individuen selektiert, die zur Reproduktion bestimmt sind. Durch Rekombination und Mutation werden daraus Nachkommen generiert, die die Eltern ersetzen, was jedoch keinen Einfluß auf das Archiv hat. Die Nachkommen bilden den zweiten Teil der aktuellen Population. Zur Bewertung der Individuen einer Generation wird jeweils die Gesamtheit aus Archiv und Nachkommen betrachtet.

4.2 Fitneßzuweisung

Das in SPEA2 verwendete Verfahren der Fitneßzuweisung basiert auf dem Prinzip der Dominanz-basierten Rangbildung. Allerdings berücksichtigt diese verbesserte Variante nicht nur die Anzahl der Individuen, von denen eine Lösung dominiert wird, sondern auch die Anzahl der Individuen, die diese Lösung dominiert. Dies führt zu einer feineren Staffelung der Fitneßwerte einer Population und verhindert, daß Lösungen, die von denselben Individuen dominiert werden, identische Fitneßwerte besitzen.

Die Bewertung der Individuen einer Population, bestehend aus P_t und \bar{P}_t , erfolgt in zwei Schritten. Zuerst wird für jedes Individuum i ein Wert $S(i)$ für dessen *strength* bestimmt, der die Anzahl der Lösungen repräsentiert, die

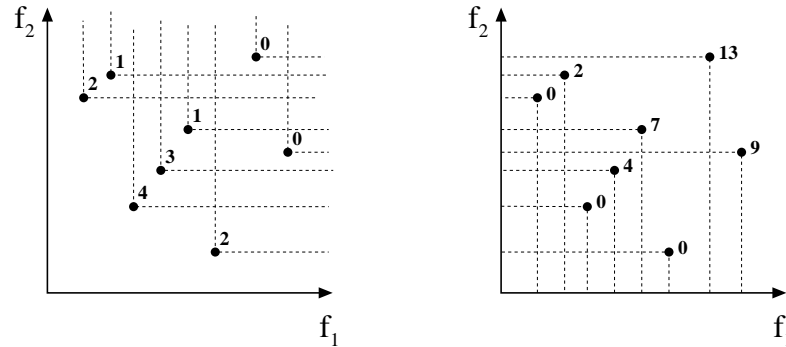


Abbildung 4.1: SPEA2 Fitneßzuweisung für ein Minimierungsproblem. Ermittlung der strength-Werte (links) und der raw fitness (rechts).

von diesem Individuum dominiert werden.

$$\begin{aligned} S(i) &= \text{card } J \\ J &:= \{j \in P_t \cup \bar{P}_t : i \succ j\} \end{aligned} \quad (4.1)$$

Lösungen, die bezüglich der Population nicht dominiert sind, erhalten demnach einen großen Wert für $S(i)$, während der Wert für den *strength* einer Lösung, die kein Individuum dominiert, null ist.

Die Fitneß (*raw fitness*) $R(i)$ einer Lösung i ergibt sich schließlich aus der Summe der *strength*-Werte der Individuen, von denen i dominiert wird.

$$\begin{aligned} R(i) &= \sum_{j \in J} S(j) \\ J &:= \{j \in P_t \cup \bar{P}_t : j \succ i\} \end{aligned} \quad (4.2)$$

Für nicht dominierte Lösungen ergibt sich eine Fitneß von $R(i) = 0$. Lösungen, die von vielen Individuen dominiert werden, erhalten höhere Werte für die Fitneß. Die Fitneß $R(i)$ ist somit ein Wert, den es zu minimieren gilt. Die beiden Schritte zur Bewertung der Individuen sind in Abb. (4.1) für ein Minimierungsproblem mit zwei Zielfunktionen dargestellt.

Die Ermittlung der *strength*-Werte und der Fitneßwerte erfordert jeweils die paarweise Untersuchung der Pareto-Dominanzrelation jedes Individuums bezüglich jedes anderen Individuums der Population. In der `Stang`-Implementierung ist die Fitneßzuweisung in den Routinen `calculateStrength` und `calculateFitness` enthalten. Die Relation zweier Individuen bezüglich des Kriteriums der Pareto-Dominanz wird in der Routine `dominates` (s. Anhang A) untersucht.

4.3 Selektion durch Dichteabschätzung

Die Aktualisierung des Archivs der nächsten Generation erfolgt durch umgebungsbedingte Selektion. Dazu werden zunächst alle nicht dominierten Individuen in das Archiv der Größe α kopiert.

$$\bar{P}_{t+1} := \{i \in P_t \cup \bar{P}_t : R(i) = 0\} \quad (4.3)$$

Ist die Anzahl der nicht dominierten Lösungen gleich der Größe des Archivs ($\text{card } \bar{P}_{t+1} = \alpha$), ist die Selektion beendet. Andernfalls sind zwei verschiedene Situationen möglich:

- Die Anzahl der nicht dominierten Lösungen ist kleiner als die Größe des Archivs ($\text{card } \bar{P}_{t+1} < \alpha$).
- Die Anzahl der nicht dominierten Lösungen übersteigt die Größe des Archivs ($\text{card } \bar{P}_{t+1} > \alpha$).

Im ersten Fall wird das Archiv mit dominierten Individuen aus P_t und \bar{P}_t aufgefüllt. Im zweiten Fall müssen Individuen aus der Menge der nicht dominierten Lösungen entfernt werden, um die Größe des Archivs zu erreichen. An diesem Punkt ist es notwendig, zusätzlich zur Fitness ein weiteres Selektionskriterium einzuführen. Es soll den Selektionsdruck auf Individuen, wie z.B. Randlösungen, die der Diversität der Population dienlich sind, erhöhen.

Das in SPEA2 verwendete Selektionskriterium zur Gewährleistung der Diversität ist der Abstand eines Individuums zu seinem k -ten Nachbarn. Es wird in (Silverman (1986)) als lokales Verfahren zur Dichteabschätzung von Daten beschrieben. Dazu muß für jede Lösung der Abstand zu allen Lösungen im Zielfunktionsraum ermittelt werden. Sind \mathbf{i} und \mathbf{j} Lösungsvektoren im Zielfunktionsraum, ergibt sich deren Abstand aus der Euklidischen Norm der Differenz beider Vektoren:

$$d(\mathbf{i}, \mathbf{j}) = \sqrt{\sum_m |i_m - j_m|^2} \quad (4.4)$$

Die ermittelten Abstände von \mathbf{i} zu den n Lösungen werden in aufsteigender Reihenfolge geordnet.

$$d_1(\mathbf{i}) \leq \dots \leq d_k(\mathbf{i}) \leq \dots \leq d_n(\mathbf{i}) \quad (4.5)$$

Hierbei repräsentiert $d_k(\mathbf{i})$ den Abstand von \mathbf{i} zu seinem k -ten Nachbarn. Weiterhin ist zu beachten, daß $d_1(\mathbf{i})$ der Abstand von \mathbf{i} zu sich selbst und damit null ist. Die Ermittlung des Abstands eines Individuums zu seinem k -ten Nachbarn erfolgt durch die `Stang`-Routine `getNND`.

Mittels dieses Kriteriums erfolgt das Auffüllen oder Reduzieren des Archivs der nächsten Generation. Für den ersten Fall ($\text{card } \bar{P}_{t+1} < \alpha$) läßt sich der Ablauf folgendermaßen beschreiben:

SPEA2 Auffüllen des Archivs

- Schritt 1: Sortiere alle dominierten Individuen aus P_t und \bar{P}_t in aufsteigender Reihenfolge bezüglich ihrer Fitneß.
- Schritt 2: Ermittle die Anzahl der freien Plätze im Archiv ($\alpha - \text{card } \bar{P}_{t+1}$).
- Schritt 3: Kopiere die dominierten Individuen mit der niedrigsten Fitneß in das Archiv, bis die Archivgröße erreicht ist. Steht eine Anzahl von Individuen mit gleicher Fitneß zur Auswahl, die die Anzahl der freien Plätze übersteigt, wähle diejenigen aus, die den größten Abstand zu ihrem k -ten Nachbarn in P_t und \bar{P}_t besitzen. Setze dazu $k = 2$.
- Schritt 4: Markiere alle Individuen aus P_t und \bar{P}_t , die nicht für \bar{P}_{t+1} ausgewählt wurden, als gelöscht.

Das beschriebene Auffüllen des Archivs mit dominierten Individuen ist in den Routinen `truncate_dominated` und `population_density` der `SIANG`-Implementierung enthalten.

Das Reduzieren des Archivs für den Fall $\text{card } \bar{P}_{t+1} > \alpha$ verläuft nach folgendem Schema:

SPEA2 Reduzieren des Archivs

- Schritt 1: Markiere alle dominierten Individuen aus P_t und \bar{P}_t als gelöscht.
- Schritt 2: Ermittle die maximale Häufigkeit des Vorkommens der Individuen in \bar{P}_{t+1} .
- Schritt 3: Markiere aus den Individuen, die mit der maximalen Häufigkeit in \bar{P}_{t+1} vorkommen, jenes als gelöscht, das den kleinsten Abstand zu seinem k -ten Nachbarn in \bar{P}_{t+1} besitzt. Beginne mit $k = 2$. Falls mehrere Individuen den kleinsten Abstand besitzen, untersuche den Abstand zum $(k + 1)$ -ten Nachbarn usw.
- Schritt 4: Ermittle die aktuelle Größe des Archivs. Falls $\text{card } \bar{P}_{t+1} > \alpha$, gehe zu Schritt 2.

In den Routinen `truncate_nondominated` und `archive_density` ist das Reduzieren des Archivs implementiert.

Nach Abschluß der umgebungsbedingten Selektion ist das Archiv mit α Individuen gefüllt, die unverändert in die nächste Generation übergehen. Sie bilden gleichzeitig den Pool, aus dem μ Individuen zur Reproduktion ausgewählt werden. Dies erfolgt durch binäre Turniers Selektion und kann wie folgt formuliert werden:

SPEA2 binäre Turniererlektion

- Schritt 1: Wähle aus \bar{P}_{t+1} ein Individuum i zufällig aus und setze den Turnierzähler gleich null.
- Schritt 2: Wähle aus \bar{P}_{t+1} ein Individuum j zufällig aus und erhöhe den Turnierzähler um eins.
- Schritt 3: Vergleiche die Fitneßwerte von i und j . Ist $R(i) < R(j)$, gehe zu Schritt 6. Ist $R(i) > R(j)$, gehe zu Schritt 5. Sind die Fitneßwerte identisch, gehe zu Schritt 4.
- Schritt 4: Ermittle für i und j den jeweiligen Abstand zu ihrem k -ten Nachbarn ($k = 2$). Ist $d_2(j) > d_2(i)$, gehe zu Schritt 5, ansonsten gehe zu Schritt 6.
- Schritt 5: Kopiere j nach i .
- Schritt 6: Ist der Turnierzähler kleiner als die Turniergröße, gehe zu Schritt 2. Ist die Turniergröße erreicht, wähle i als Elternindividuum aus und lege es in den Pool der möglichen Eltern zurück.
- Schritt 7: Ist die Anzahl der bisher selektierten Eltern kleiner als μ , gehe zu Schritt 1.

Die binäre Turniererlektion ermöglicht die mehrfache Auswahl von Individuen zur Reproduktion. Der Selektionsdruck auf bessere Individuen bezüglich der Kriterien Fitneß und Abstand zum k -ten Nachbarn läßt sich durch Erhöhung der Turniergröße verstärken. Durch Einbeziehung der Dichteabschätzung ist der Selektionsdruck für Randlösungen generell höher, was sich positiv auf die Diversität der Population auswirkt, weil dadurch die Suche in diese Bereiche gelenkt wird. Der beschriebene Ablauf der Elternselektion ist in der Routine `matingSelection` implementiert.

Zuletzt werden auf die selektierten Elternindividuen genetische Variationsoperatoren angewendet. Die Wahl des Rekombinationsoperators ist abhängig von der Art der Codierung der Chromosomen. Für binär codierte Chromosomen stehen Varianten wie n -Punkt-Cross-over und uniformes Cross-over zur Verfügung. Für kontinuierliche Suchräume mit reeller Variablenkodierung kann der *simulated binary crossover*-Operator (Deb und Agrawal (1995)), dessen Eigenschaften im folgenden Kapitel erläutert werden sollen, verwendet werden. Die Mutation der Nachkommen erfolgt mit einer festzulegenden Mutationsrate auf der Basis einer Zufallsvariable, die eine gewählte Wahrscheinlichkeitsdichte besitzt.

5

Erweiterungen

5.1 Simulated Binary Crossover

Der Cross-over-Operator ist für den Suchprozeß eines EA von entscheidender Bedeutung. Zwischen Chromosomen von Individuen mit guter Fitneß werden dabei Gene ausgetauscht, um Chromosomen der Nachkommen zu erhalten. Die Absicht ist es, durch Rekombination der Chromosomen guter Individuen Nachkommen mit besseren Eigenschaften zu erzeugen. Der Erfolg des Suchprozesses hängt auch von der Art der Variablencodierung und des Suchraums ab. Für binär codierte Variablen in Zusammenhang mit diskreten Suchräumen hat sich das Einpunkt-Cross-over als effizienter Operator erwiesen. Für kontinuierliche Suchräume beinhaltet die Umwandlung reeller Variablen in binäre Zahlenketten Schwierigkeiten für den Suchprozeß. Durch die Diskretisierung des Suchraumes ist es nicht möglich, Lösungen mit beliebiger Genauigkeit zu erzielen. Ausgehend von der Suchcharakteristik des Einpunkt-Cross-over, wurde von (Deb und Agrawal (1995)) das *simulated binary crossover* (SBX) für reell codierte Variablen kontinuierlicher Suchräume entwickelt.

Die Suchleistung eines Cross-over-Operators zeigt sich anhand der Wahrscheinlichkeit, aus einem gegebenen Paar von Elternchromosomen (p_1, p_2) beliebige Nachkommen (c_1, c_2) erzeugen zu können. Anhand eines Streukoeffizienten β , der das Verhältnis der Streuung der Nachkommen zur Streuung der Eltern beschreibt, lassen sich zunächst verschiedene Cross-over-Charakteristika klassifizieren.

$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right| \quad (5.1)$$

Kontrahierendes Cross-over: für $\beta < 1$. Der Abstand der Nachkommen ist kleiner als der der Eltern. Die Nachkommen befinden sich zwischen den Eltern.

Expandierendes Cross-over: für $\beta > 1$. Der Abstand der Nachkommen ist größer als der der Eltern. Die Nachkommen liegen außerhalb des Bereiches, der von den Eltern begrenzt wird.

Stationäres Cross-over: für $\beta = 1$. Die Nachkommen sind mit Eltern identisch.

Die Eigenschaft der Kontraktion oder Expansion eines Cross-over-Operators läßt sich anhand des Streuungskoeffizienten β ermitteln. Für den SBX-Operator wird für β folgende Wahrscheinlichkeitsdichte formuliert (Deb und Agrawal (1995)):

$$\varphi(\beta) = 0.5(n_c + 1)\beta^{n_c} \quad ; \quad \text{für } 0 \leq \beta \leq 1 \quad (5.2)$$

$$\varphi(\beta) = 0.5(n_c + 1)\frac{1}{\beta^{n_c+2}} \quad ; \quad \text{für } \beta \geq 1. \quad (5.3)$$

Hierbei ist n_c ein Verteilungsparameter, der alle nicht negativen Werte annehmen kann. Für diese Wahrscheinlichkeitsdichte ist die Wahrscheinlichkeit eines kontrahierenden Cross-over gleich der Wahrscheinlichkeit eines extrahierenden Cross-over:

$$P(0 \leq \beta \leq 1) := \int_0^1 \varphi(\beta) d\beta = 0.5 \quad (5.4)$$

$$P(\beta \geq 1) := \int_1^\infty \varphi(\beta) d\beta = 0.5. \quad (5.5)$$

In Abb. (5.1) ist $\varphi(\beta)$ für verschiedene Verteilungsparameter n_c dargestellt. Für $n = 0$ ergibt sich im Bereich $0 \leq \beta \leq 1$ eine Gleichverteilung der Wahrscheinlichkeitsdichte. Für große Werte von n_c ist die Wahrscheinlichkeit größer, daß β nahe dem Wert 1 liegt.

Unter Verwendung der beschriebenen Wahrscheinlichkeitsdichte generiert der SBX-Operator zwei Nachkommen (c_1, c_2) aus zwei Eltern (p_1, p_2) nach folgendem Prinzip:

Schritt 1: Erzeuge eine Zufallszahl u zwischen 0 und 1.

Schritt 2: Ermittle den Parameter $\bar{\beta}$ wie folgt:

$$\bar{\beta} = \begin{cases} (2u)^{\frac{1}{n_c+1}} & \text{für } u \leq 0.5, \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{n_c+1}} & \text{sonst.} \end{cases} \quad (5.6)$$

Schritt 3: Generiere die Nachkommen wie folgt:

$$\begin{aligned} c_1 &= 0.5 \left[(p_1 + p_2) - \bar{\beta} |p_2 - p_1| \right], \\ c_2 &= 0.5 \left[(p_1 + p_2) + \bar{\beta} |p_2 - p_1| \right]. \end{aligned} \quad (5.7)$$

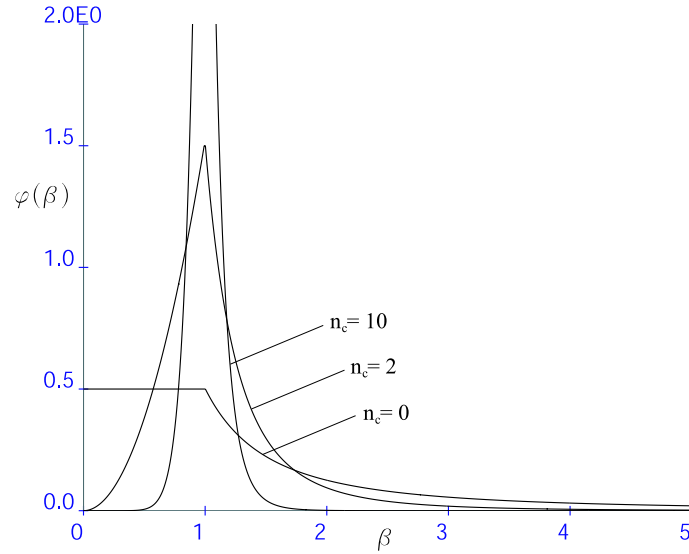


Abbildung 5.1: Wahrscheinlichkeitsdichte von β für verschiedene Verteilungsparameter n_c .

Die Entfernung der Nachkommen von den Eltern kann über den Verteilungsparameter n_c beeinflusst werden. Durch einen kleinen Wert für n_c ist es möglich, Nachkommen weit entfernt von den Eltern zu generieren, während ein großer Wert für n_c dies auf den Bereich nahe der Eltern beschränkt.

Sind für die Variablen die untere und obere Schranke (x_l und x_u) bekannt, wird Gl. (5.6) folgendermaßen verändert:

$$\bar{\beta} = \begin{cases} (\alpha u)^{\frac{1}{n_c+1}} & \text{für } u \leq \frac{1}{\alpha}, \\ \left(\frac{1}{2-\alpha u}\right)^{\frac{1}{n_c+1}} & \text{sonst.} \end{cases} \quad (5.8)$$

Hierbei ergibt sich α aus:

$$\alpha = 2 - \delta^{-(n_c+1)}. \quad (5.9)$$

Unter der Voraussetzung, daß $p_1 < p_2$, läßt sich δ folgendermaßen ermitteln:

$$\delta = 1 + \frac{2}{p_2 - p_1} \min [(p_1 - x_l), (x_u - p_2)]. \quad (5.10)$$

Für diese Verfahrensweise ist die Wahrscheinlichkeit, daß Nachkommen außerhalb der Variablengrenzen generiert werden, gleich null. Außerdem wird vorausgesetzt, daß $p_2 - p_1 \neq 0$ ist. Enthalten die Elternchromosomen mehrere Gene, werden die einzelnen Gene schrittweise mit der Wahrscheinlichkeit 0.5 für das Cross-over ausgewählt. Der SBX-Operator wurde im Rahmen dieser Arbeit in die `genetic`-Kommandogruppe von `SBang` implementiert. Die

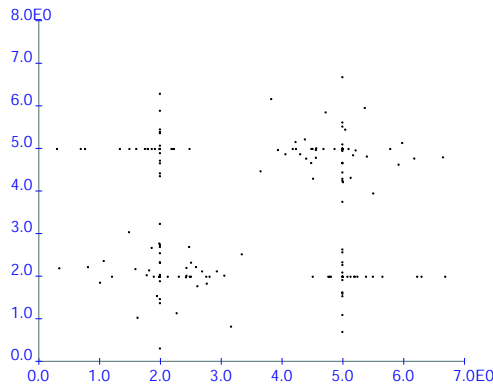


Abbildung 5.2: Verteilung von c_1 und c_2 ; 100mal SBX von $p_1=(2,2)$ und $p_2=(5,5)$; $n_c=1$.

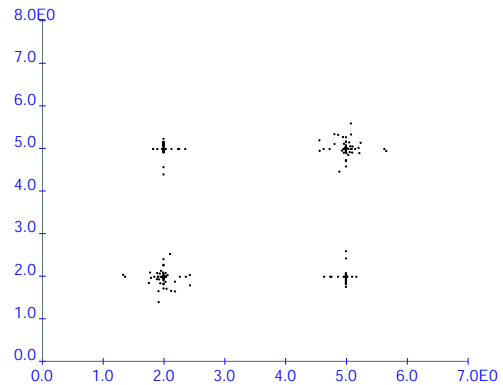


Abbildung 5.3: Verteilung von c_1 und c_2 ; 100mal SBX von $p_1=(2,2)$ und $p_2=(5,5)$; $n_c=10$.

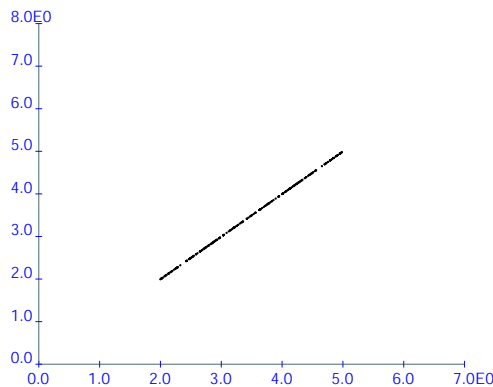


Abbildung 5.4: Verteilung von c_1 und c_2 ; 100mal arithmetisches Cross-Over von $p_1=(2,2)$ und $p_2=(5,5)$.

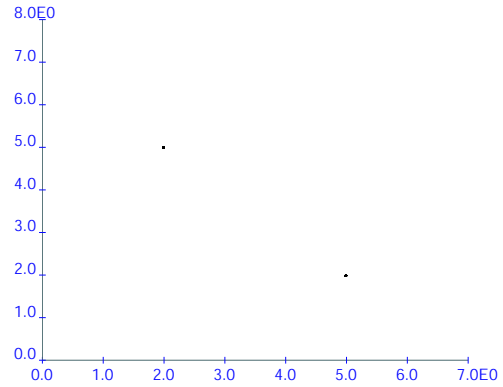


Abbildung 5.5: Verteilung von c_1 und c_2 ; 100mal Einpunkt-Cross-Over für reelle Codierung von $p_1=(2,2)$ und $p_2=(5,5)$.

Dokumentation des `Slang`-Kommandos `genetic crossover`, `sbx` findet sich im Anhang B.

Die Abbildungen (5.1–5.5) zeigen die Ergebnisse der mehrfachen Anwendung verschiedener in `Slang` implementierter Cross-over-Operatoren. Ausgangspunkt ist jeweils ein Elternpaar ($p_1 = (2, 2)$; $p_2 = (5, 5)$) mit reeller Variablencodierung. Die Variablengrenzen sind für beide Variablen durch $x_l = 0$ und $x_u = 7$ festgelegt. Der Cross-over-Operator wurde jeweils 100mal auf das Elternpaar angewendet. Dazu wurde die Population der Eltern nach jedem Cross-over-Vorgang erneut eingelesen.

Die Ergebnisse zeigen die Abhängigkeit des SBX-Operators von dem Verteilungsparameter n_c . Für $n_c = 10$ befinden sich die Nachkommen näher im Bereich der Eltern als für $n_c = 1$. Weiterhin wird deutlich, daß SBX

sowohl kontrahierende als auch expandierende Charakteristik besitzen kann, während das arithmetische Cross-over eine gewichtete Mittelwertbildung zwischen den Genen der Eltern durchführt und somit nur kontrahierend wirkt. Das Einpunkt-Cross-over erzeugt lediglich Nachkommen mit den Genkombinationen (2, 5) und (5, 2).

5.2 Berücksichtigung von Nebenbedingungen

Viele praktische Mehrzieloptimierungsprobleme beinhalten lineare oder nichtlineare Nebenbedingungen der Gleichheit oder Ungleichheit. Zur Berücksichtigung dieser Nebenbedingungen in Einzieloptimierungsproblemen wird häufig eine *penalty*-Funktion verwendet, um die Fitneß von Individuen, die die Nebenbedingungen verletzen, zu verschlechtern. Für ein Minimierungsproblem wird dazu der Wert der Zielfunktion um einen *penalty*-Wert erhöht, der vom Maß der Verletzung der Nebenbedingungen abhängig ist. Die Fitneßfunktion $F(\mathbf{x})$ ergibt sich dann aus:

$$F(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^J R_j \langle g_j(\mathbf{x}) \rangle^2, \quad (5.11)$$

wobei der Operator $\langle \rangle$ den Betrag des Operanden liefert, falls dieser negativ ist. Liegt keine Verletzung der Nebenbedingungen vor, liefert der Operator den Wert 0. Der *penalty*-Parameter R_j soll die Verletzung der jeweiligen Nebenbedingung an die Größenordnung des Zielfunktionswertes anpassen. Die Schwierigkeit bei der Anwendung der *penalty*-Funktion besteht in der Wahl des Parameters R_j . Da er Einfluß auf den Zielfunktionswert hat, kann die falsche Wahl des Parameters zu einer unkontrollierten Entwicklung der Population führen.

Eine Methode zur Berücksichtigung von Nebenbedingungen, die ohne einen zusätzlichen Parameter auskommt, wird in (Deb (2000)) beschrieben. Die Methode verwendet binäre Turnierselektion, wobei jeweils zwei Lösungen aus der Population miteinander verglichen werden. Für Einzieloptimierungsprobleme wird eine der Lösungen nach folgenden Kriterien ausgewählt:

- Jede gültige Lösung wird gegenüber jeder ungültigen Lösung bevorzugt.
- Sind beide Lösungen gültig, wird die mit dem besseren Zielfunktionswert ausgewählt.
- Sind beide Lösungen ungültig, wird die ausgewählt, die die Nebenbedingungen am wenigsten verletzt.

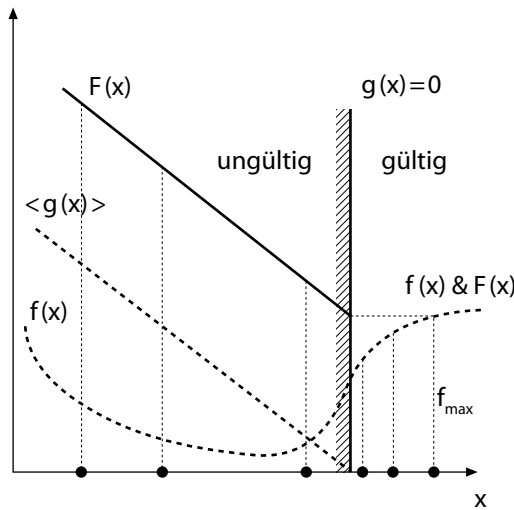


Abbildung 5.6: Zusammengesetzte Fitnessfunktion bei der Berücksichtigung von Nebenbedingungen.

Bereits die Formulierung der Kriterien offenbart die entkoppelte Betrachtungsweise von Zielfunktion und Verletzung der Nebenbedingungen. Dies resultiert in einer zusammengesetzten Fitnessfunktion für jede Lösung \mathbf{x} :

$$F(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{für } g_j(\mathbf{x}) \geq 0, \forall j = 1, 2, \dots, J, \\ f_{max} + \sum_{j=1}^J \langle g_j(\mathbf{x}) \rangle & \text{sonst.} \end{cases} \quad (5.12)$$

Hierbei ist f_{max} der Zielfunktionswert der schlechtesten gültigen Lösung der Population. Abb. (5.6) veranschaulicht die Konstruktion der Fitnessfunktion $F(\mathbf{x})$ aus der Zielfunktion $f(\mathbf{x})$ und der Nebenbedingung $g_j(\mathbf{x})$ für ein Minimierungsproblem mit einer Designvariablen. Es ist erkennbar, daß die Fitness von ungültigen Lösungen von der Verletzung der Nebenbedingungen abhängig ist, während die Fitness von gültigen Lösungen allein auf deren Zielfunktionswert beruht. Da in keinem Fall beide Kriterien im Zusammenhang auftreten, ist auch kein zusätzlicher Parameter nötig, der die Skalierung eines der Kriterien beeinflusst. Statt dessen erfordert die Methode die Aufteilung der Population in gültige und ungültige Lösungen durch die Überprüfung der Einhaltung der Nebenbedingungen für jede Lösung. Für die gültigen Lösungen muß dann noch der Wert der Zielfunktion ermittelt werden. Das ist für die ungültigen Lösungen nicht erforderlich, wodurch Rechenleistung eingespart werden kann.

Die Übertragung der genannten Präferenzkriterien in Zusammenhang mit der formulierten Fitnessfunktion auf Probleme der Mehrzieloptimierung erfolgt durch die Erweiterung des Dominanzkriteriums. Eine Lösung i dominiert eine Lösung j unter Nebenbedingungen, wenn eine der folgenden

Bedingungen erfüllt ist:

- Lösung i ist gültig, und Lösung j ist ungültig.
- Beide Lösungen sind ungültig, aber Lösung i besitzt eine geringere Verletzung der Nebenbedingungen.
- Beide Lösungen sind gültig, aber i dominiert j nach dem Kriterium der Pareto-Dominanz.

Somit kann die Berücksichtigung von Nebenbedingungen in die dominanzbasierte Rangbildung einbezogen werden. Aus den Bedingungen der Dominanz unter Nebenbedingungen resultiert für jede gültige Lösung ein besserer Rang als für jede ungültige Lösung. Die Rangfolge unter gültigen Lösungen wird anhand ihrer Pareto-Dominanz, basierend auf den Zielfunktionswerten, ermittelt, während die Rangfolge unter ungültigen Lösungen von deren jeweiliger Verletzung der Nebenbedingungen abhängig ist.

Dieses Verfahren kann in einen EA zur Mehrzieloptimierung folgendermaßen integriert werden:

1. Überprüfung aller Individuen bezüglich deren Verletzung der Nebenbedingungen.
2. Aufteilung der Population in gültige und ungültige Individuen.
3. Ermittlung der Zielfunktionswerte für gültige Individuen.
4. Rangbildung aller Individuen auf Basis der Kriterien zur Dominanz unter Nebenbedingungen.

Die Berücksichtigung von Nebenbedingungen durch ein modifiziertes Dominanzkriterium wurde im Rahmen dieser Arbeit in den SPEA2-Algorithmus integriert und für das strukturmechanische Optimierungsproblem aus Abschnitt (6.2) angewendet.

6

Beispiele

6.1 Konstruierte Testprobleme

Der in Kapitel (4) erläuterte Algorithmus zur Lösung von Mehrzieloptimierungsproblemen SPEA2 soll im Folgenden auf verschiedene Testprobleme mit mehr als einer Zielfunktion angewendet werden. Dabei soll gezeigt werden, daß SPEA2 sowohl Lösungen nahe der Pareto-optimalen Front findet als auch die Diversität der Lösungen gewährleistet. Weiterhin soll der Einfluß der Populationsgröße und des Cross-over-Operators auf die Ergebnisse untersucht werden. Für alle nachfolgend dokumentierten Simulationen wird für die Selektion der Elternindividuen eine Turniergröße von $tournament=2$ verwendet. Die Mutation der Nachkommen erfolgt mit einer Wahrscheinlichkeit von 0.1 und basiert auf einer Normalverteilung, deren Standardabweichung mit zunehmender Generationszahl abnimmt.

Testproblem 1 (SCH1)

SCH1 ist ein einfaches, häufig verwendetes Testproblem zur Mehrzieloptimierung. Es wird aus zwei Zielfunktionen als Minimierungsproblem folgendermaßen formuliert:

$$\begin{aligned} \text{minimiere: } f_1(x) &= x^2, \\ f_2(x) &= (x - 2)^2 \end{aligned} \tag{6.1}$$

Da das Optimum von f_1 nicht mit dem Optimum von f_2 übereinstimmt, existiert mehr als eine Pareto-optimale Lösung. Das Minimum von f_1 existiert für $x = 0$, während f_2 für $x = 2$ einen minimalen Wert annimmt. Alle Lösungen des Pareto-Sets liegen demnach im Bereich $0 \leq x \leq 2$. Daraus ergeben sich folgende mögliche Intervalle für die Zielfunktionswerte der Lösungen des Pareto-Sets: $0 \leq f_1(x) \leq 4$ und $0 \leq f_2(x) \leq 4$. Ziel der Optimierung ist es, die gesamte Bandbreite der Lösungsfront abzubilden.

Die Abbildungen (6.1) und (6.2) zeigen jeweils die Lösungen des Archivs nach 50 Generationen im Zielfunktionsraum. Der Variablensuchraum wurde

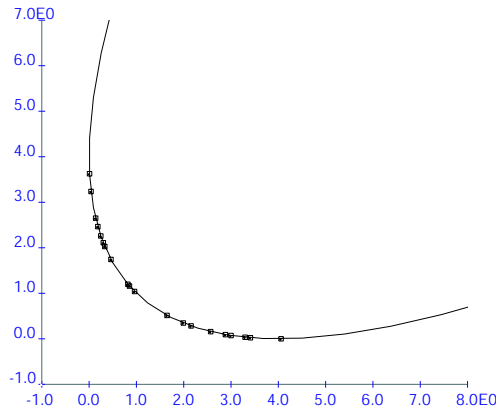


Abbildung 6.1: (SCH1) SPEA2-Archiv nach $T=50$ ($\alpha=20$, $\mu=10$, SBX ($n_c=5$)).

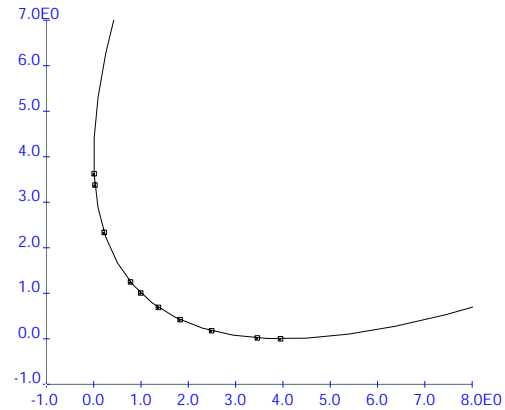


Abbildung 6.2: (SCH1) SPEA2-Archiv nach $T=50$ ($\alpha=10$, $\mu=20$, SBX ($n_c=5$)).

durch $-10 \leq x \leq 10$ begrenzt. Für das Verhältnis zwischen Archivgröße und Anzahl der Nachkommen wurden die Kombinationen $\alpha/\mu = 20/10$ und $\alpha/\mu = 10/20$ verwendet. In beiden Fällen liefert SPEA2 eine gut verteilte Menge Pareto-optimaler Lösungen, wobei auch die Randbereiche der Front berücksichtigt werden.

Um das Konvergenzverhalten von SPEA2 für die Anwendung unterschiedlicher Rekombinationsoperatoren zu untersuchen, wurde der Variablensuchraum auf $-100 \leq x \leq 100$ erweitert und die Anzahl der Generationen auf $T = 20$ verringert. Die Abb. (6.3–6.6) zeigen die Lösungen des Archivs am Ende des jeweiligen Suchlaufs. Das arithmetische Cross-over liefert die größte Anzahl von Lösungen im Bereich der Pareto-Front, während SBX mit dem Verteilungsparameter $n_c = 0$ eine höhere Diversität der Front erreicht. Die Anwendung des uniformen Cross-overs liefert lediglich eine Lösung im Bereich der Pareto-Front. Es wird deutlich, daß eine schnelle Konvergenz maßgeblich von der Suchleistung der genetischen Operatoren abhängig ist. Aufgrund des Vorhandenseins lediglich einer Designvariablen lassen sich anhand dieser Erkenntnisse jedoch keine Aussagen über das Konvergenzverhalten von SPEA2 für höherdimensionale Mehrzieloptimierungsprobleme treffen.

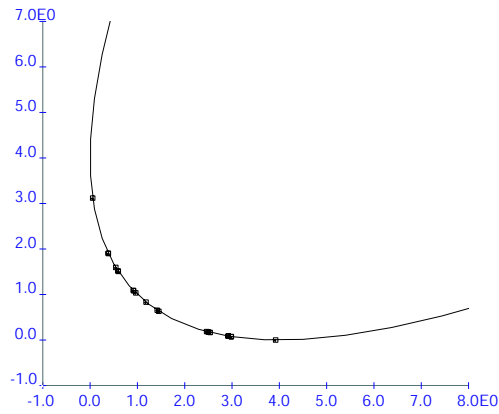


Abbildung 6.3: (SCH1) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, SBX ($n_c=0$)).

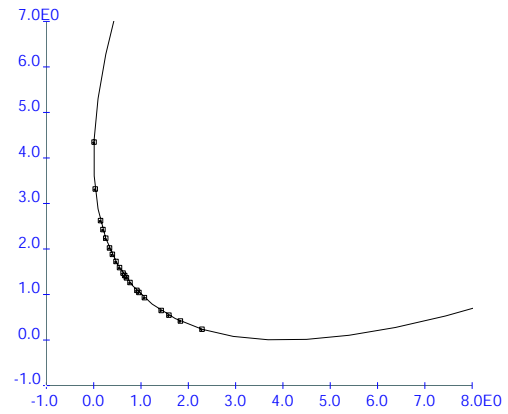


Abbildung 6.4: (SCH1) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, arithmetisches XO ($p_c=0.5$)).

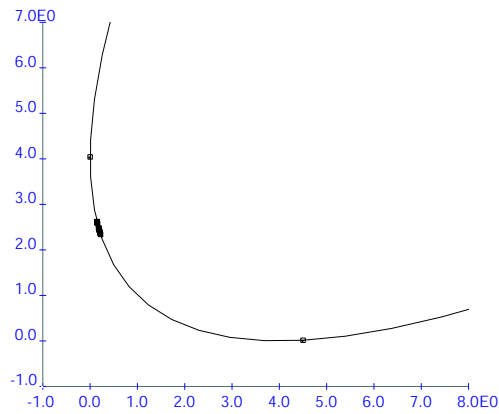


Abbildung 6.5: (SCH1) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, SBX ($n_c=5$)).

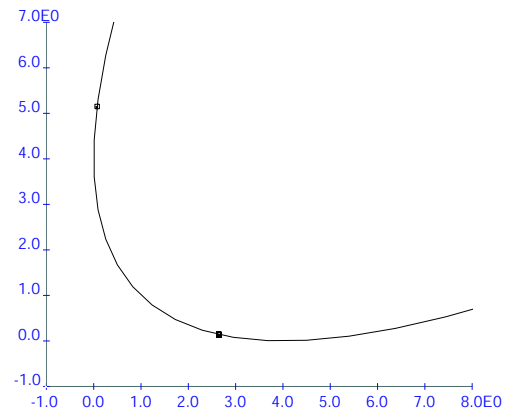


Abbildung 6.6: (SCH1) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, uniformes XO ($p_c=0.5$)).

Testproblem 2 (KUR)

Das Testproblem KUR besitzt eine disjunkte Pareto-Front in einem zwei-dimensionalen Zielfunktionsraum. Dabei besteht die Schwierigkeit darin, in allen Bereichen der Front Pareto-optimale Lösungen zu finden. Die Formulierung des Testproblems KUR lautet wie folgt:

$$\begin{aligned} \text{minimiere: } f_1(\mathbf{x}) &= \sum_{i=1}^2 \left[-10 e^{(-0.2\sqrt{x_i^2+x_{i+1}^2})} \right], \\ f_2(\mathbf{x}) &= \sum_{i=1}^3 [|x_i|^{0.8} + 5 \sin(x_i^3)], \\ -5 &\leq x_i \leq 5, \quad i = 1, 2, 3. \end{aligned} \quad (6.2)$$

Die Pareto-Front der Funktion besitzt sowohl konkave als auch konvexe Bereiche. Abb. (6.7) zeigt die Front für einen hochgradig konvergierten Suchlauf von SPEA2. Aufgrund der Populationsgröße und der hohen Anzahl von Generationen ist für ein solches Ergebnis der Aufwand an Rechenleistung jedoch unökonomisch. Für verschiedene Parametereinstellungen von SPEA2 soll gezeigt werden, wie mit einer begrenzten Anzahl von Zielfunktionsauswertungen gute Näherungen der Pareto-Front gefunden werden können.

Die in Abb. (6.8–6.10) dargestellten Ergebnisse zeigen jeweils die Lösungen des Archivs nach 20 Generationen. Ausgehend von identischen Startpopulationen wurden verschiedene Cross-over-Operatoren zur Rekombination verwendet. Obwohl sich die erzielte nicht dominierte Front in allen drei Fällen deutlich von der Pareto-Front in Abb. (6.7) unterscheidet, liefert der Suchlauf mit SBX (Abb. (6.8)) sichtbar die beste Approximation. Alle Bereiche der disjunkten Front sind mit Lösungen besetzt, und der Verlauf ist durch die hohe Diversität der Lösungen erkennbar.

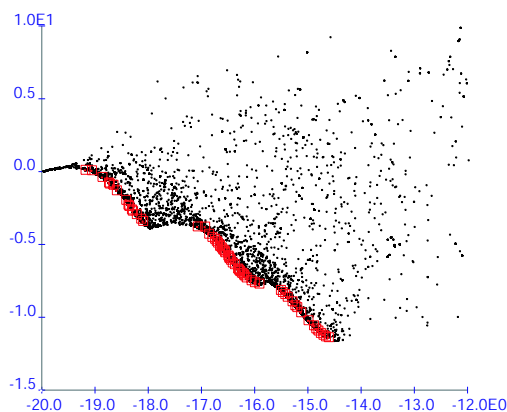


Abbildung 6.7: (KUR) SPEA2-Archiv nach $T=100$ ($\alpha=60$, $\mu=40$, SBX ($n_c=1$)).

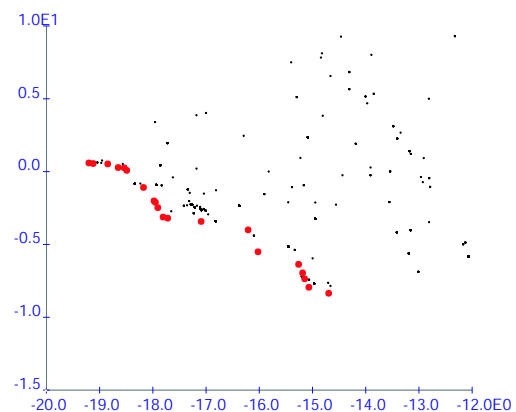


Abbildung 6.8: (KUR) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, SBX ($n_c=0$)).

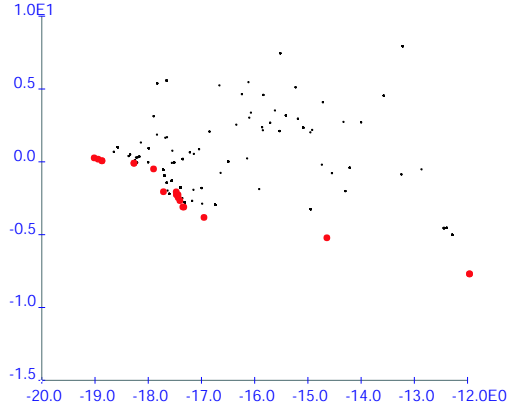


Abbildung 6.9: (KUR) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, arithmetisches XO ($p_c=0.5$)).

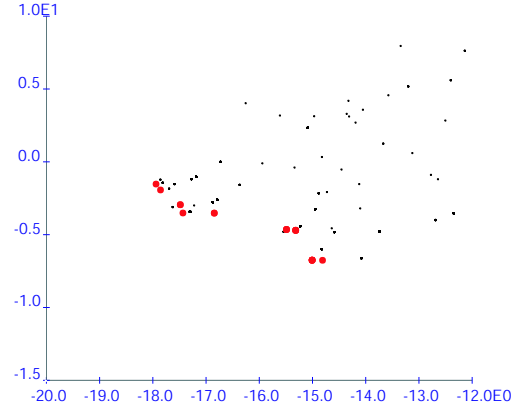


Abbildung 6.10: (KUR) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, uniformes XO ($p_c=0.5$)).

Testproblem 3 (DTLZ2)

Eine Reihe verschiedener skalierbarer Testprobleme zur Mehrzieloptimierung werden in (Deb u. a. (2002)) beschrieben. Die Skalierbarkeit bezieht sich dabei sowohl auf die Dimension des Designvariablenraums als auch auf die Anzahl der Zielfunktionen. Die formulierten Testprobleme basieren auf einer funktionalen Beschreibung der jeweiligen Pareto-Front, woraus der zugehörige Zielfunktionsuchraum konstruiert wird. Das Testproblem DTLZ2 wird folgendermaßen definiert:

$$\begin{aligned}
 \text{minimiere: } \quad & f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos(x_1 \pi/2) \cdots \cos(x_{M-1} \pi/2), \\
 & f_2(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos(x_1 \pi/2) \cdots \sin(x_{M-1} \pi/2), \\
 & \vdots \\
 & f_M(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(x_1 \pi/2), \\
 & 0 \leq x_i \leq 1, \quad i = 1, 2, \dots, n, \\
 \text{wobei: } \quad & g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2.
 \end{aligned} \tag{6.3}$$

Die Anzahl der Designvariablen beträgt $n = M + k - 1$, wobei $k = |\mathbf{x}_M|$ die Anzahl der von $g(\mathbf{x}_m)$ verwendeten Variablen ist. Die nachfolgenden Simulationsergebnisse basieren auf $k = 8$ und $M = 3$ und einer sich daraus ergebenden Dimension $n = 10$ des Designvariablenraums. Die Pareto-optimalen Lösungen ergeben sich für $x_i = 0.5$ ($x_i \in \mathbf{x}_M$). Die zugehörigen Zielfunktionswerte bilden die sphärische Hyperebene $\sum_{m=1}^M (f_m)^2 = 1$. In Abb. (6.11) ist die Pareto-Front für die optimalen Werte von $(x_i \in \mathbf{x}_M)$ dargestellt. Die Werte für x_1, \dots, x_{M-1} wurden mit einer gleich verteilten Wahrscheinlichkeitsdichte zwischen 0 und 1 generiert.

Durch die Art der Problemformulierung läßt sich die theoretische Pareto-Front durch einen funktionalen Zusammenhang ausdrücken. Da für dieses

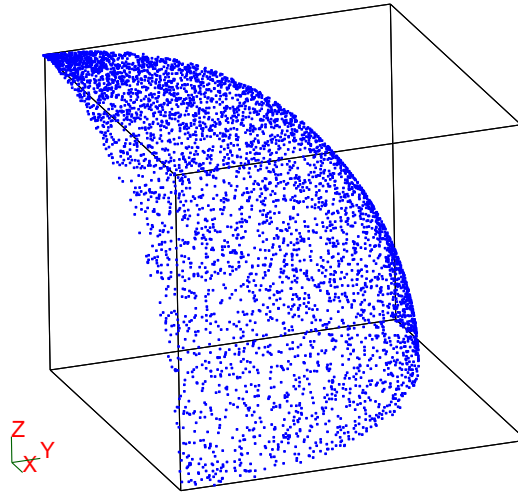


Abbildung 6.11: Theoretische Pareto-Front für TP DTLZ2.

Beispiel alle Pareto-Lösungen auf einer sphärischen Hyperebene mit konstantem Radius liegen, läßt sich für jede während des Optimierungsprozesses gefundene Lösung der Abstand zur tatsächlichen Pareto-Front bestimmen. Für die nachfolgend dokumentierten Simulationen wurde für jede Generation der mittlere Radius \bar{r} der von den Individuen des Archivs aufgespannten Hypersphäre ermittelt.

$$\bar{r} = \left(\sum_{i=1}^{\alpha} \sqrt{\sum_{m=1}^M f_m(\mathbf{x}_i)^2} \right) / \alpha \quad (6.4)$$

Dadurch ergibt sich ein numerisches Kriterium für den Grad der Konvergenz zur optimalen Front. Die Abb. (6.12–6.19) zeigen die Ergebnisse der Optimierung des Testproblems DTLZ2 durch den SPEA2-Algorithmus. Neben den Lösungen des Archivs im Zielfunktionsraum am Ende des jeweiligen Suchlaufs ist der Verlauf des Kriteriums des mittleren Radius des Archivs über die Anzahl der Generationen dargestellt.

Für die beiden ersten Suchläufe mit hoher Populationsgröße und einer hohen Anzahl von Generationen zeigt sich die von SPEA2 erreichbare Genauigkeit der Approximation der Pareto-Front für dieses Beispiel. Durch Anwendung des SBX-Operators erreicht SPEA2 die theoretische Pareto-Front mit einer durchschnittlichen Abweichung von 2.22%, während die Ergebnisse bei Verwendung des arithmetischen Cross-over um 8.34% vom tatsächlichen Optimum abweichen. In den Abb. (6.16–6.19) sind die Ergebnisse für eine geringere Populationsgröße und Generationszahl dargestellt. Wiederum liefert

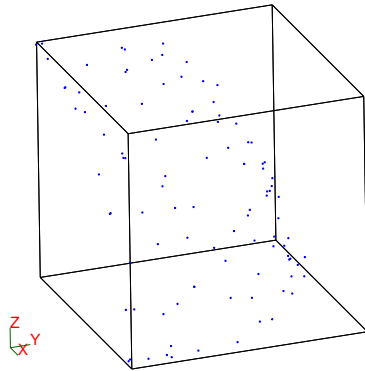


Abbildung 6.12: (DTLZ2) SPEA2-Population nach $T=100$ ($\alpha=60$, $\mu=40$, SBX ($n_c=1$)).

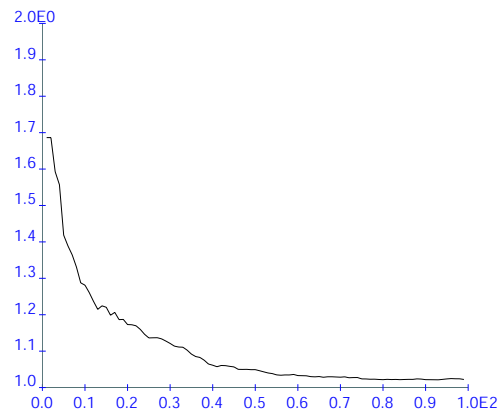


Abbildung 6.13: Verlauf von \bar{r} des SPEA2-Archivs über die Anzahl der Generationen ($\bar{r}(100)=1.022$).

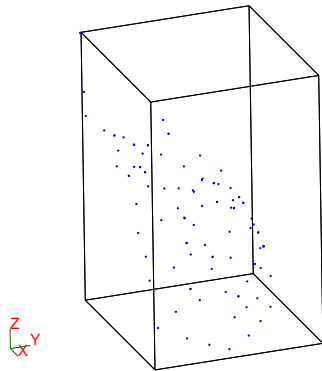


Abbildung 6.14: (DTLZ2) SPEA2-Population nach $T=100$ ($\alpha=60$, $\mu=40$, arithmetisches XO ($p_c=0.5$)).

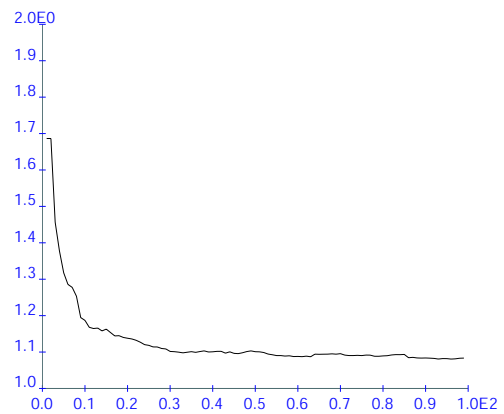


Abbildung 6.15: Verlauf von \bar{r} des SPEA2-Archivs über die Anzahl der Generationen ($\bar{r}(100)=1.083$).

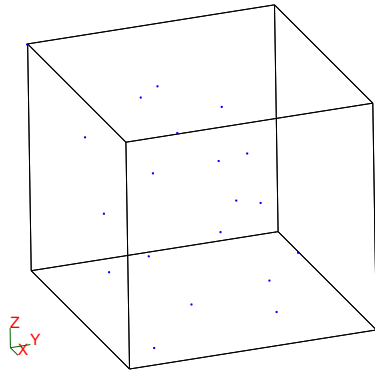


Abbildung 6.16: (DTLZ2) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, SBX ($n_c=0$)).

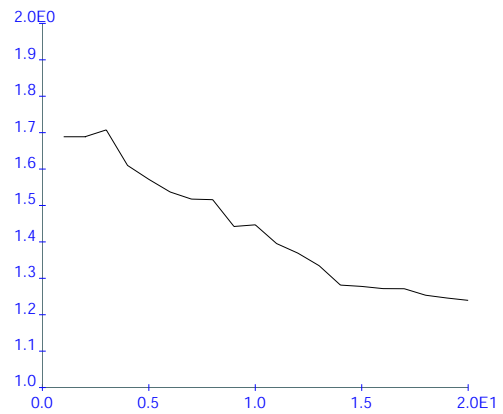


Abbildung 6.17: Verlauf von \bar{r} des SPEA2-Archivs über die Anzahl der Generationen ($\bar{r}(20)=1.239$).

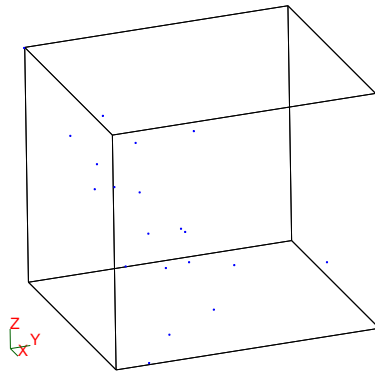


Abbildung 6.18: (DTLZ2) SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, arithmetisches XO ($p_c=0.5$)).

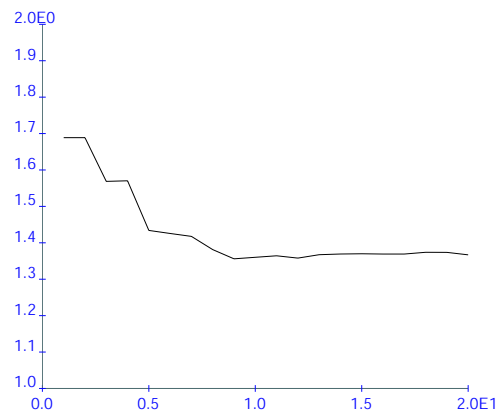


Abbildung 6.19: Verlauf von \bar{r} des SPEA2-Archivs über die Anzahl der Generationen ($\bar{r}(20)=1.367$).

der SBX-Operator ein höheres Maß der Konvergenz, auch wenn die Abweichung von der optimalen Front mit 23.9% doch erheblich ist. Alle erzielten Ergebnisse zeigen eine hohe Diversität der Verteilung der Lösungen im Zielfunktionsraum.

Anhand der Ergebnisse wird erkennbar, daß die Geschwindigkeit der Konvergenz mit zunehmender Generationszahl abnimmt. Dies ist darauf zurückzuführen, daß die Anzahl der nicht dominierten Individuen schon nach wenigen Generationen sehr hoch ist. Damit läßt sich unter den Lösungen kaum noch eine dominanzbasierte Rangordnung herstellen. Das Fortschreiten der Konvergenz hängt nun hauptsächlich vom Suchpotential der Variationsoperatoren ab. Dieses Problem gewinnt mit zunehmender Dimension des Zielfunktionsraums an Bedeutung, da es dann immer schwieriger wird, besonders unter wenigen Individuen, eine differenzierte Bewertung auf Basis der Pareto-Dominanz durchzuführen.

Im folgenden Abschnitt soll der SPEA2-Algorithmus auf ein praktisches Mehrzieloptimierungsproblem angewendet und dabei auch die Komponente der Entscheidungsfindung im Ablauf des Optimierungsprozesses verdeutlicht werden.

6.2 Strukturmechanisches Optimierungsproblem

Das in diesem Abschnitt dokumentierte Beispiel ist ein strukturmechanisches Optimierungsproblem mit zwei konkurrierenden Zielfunktionen. Die Struktur besteht aus einem ebenen 10-Stab-Fachwerkbinder. Die Designvariablen sind die zehn Querschnitte der Fachwerkstäbe. Die zu minimierenden Zielfunktionen sind einerseits das Gewicht der Konstruktion und andererseits die maximale vertikale Knotenverschiebung der Fachwerkknoten.

Für den Fachwerkbinder wurden folgende Geometrie- und Materialparameter verwendet:

Elementlänge L	9.0	m
E-Modul	70000	N/mm ²
Dichte	2700	kg/m ³
Last F	45000	N
maximale Spannung	±170	N/mm ²

Die Materialparameter sind für alle Elemente des Fachwerkbinders identisch. Die Radien der verschiedenen Stabquerschnitte sind durch die Variablen-schranken $4 \text{ mm} \leq r_i \leq 90 \text{ mm}$ beschränkt. Die in den Fachwerkstäben auftretenden Zug- oder Druckspannungen sollen eine maximal zulässige Spannung nicht überschreiten. Die Berücksichtigung des Überschreitens der zulässigen Spannung erfolgt durch die Formulierung einer Nebenbedingung.

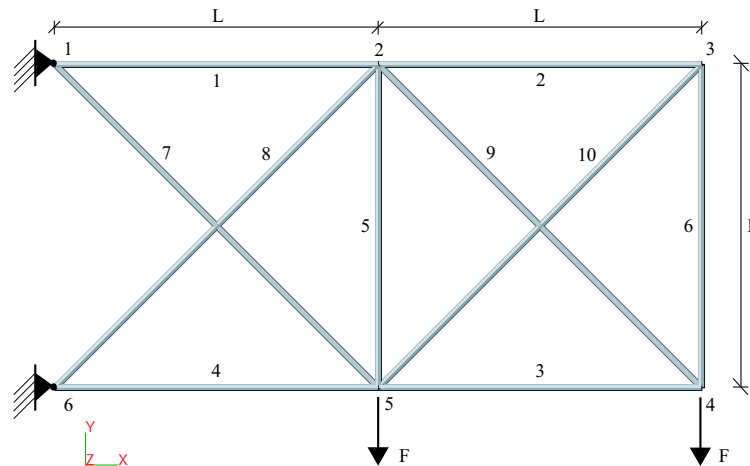


Abbildung 6.20: 10-Stab-Fachwerk: Geometrie, Elemente, Knoten, Lasten.

Abb. (6.20) zeigt die Geometrie der Struktur, die Lagerungsbedingungen und die beiden Knotenlasten.

Für die Modellierung der Struktur in `Stang` wurden 2-Knoten-Stabelemente mit Kreisquerschnitt (ROD) und linear-elastischem Materialgesetz verwendet. Die Berechnung der Knotenverschiebungen und der Spannungen erfolgte durch eine lineare Finite-Elemente-Analyse.

Die Abb. (6.21) zeigt die Pareto-Front der Lösungen des Archivs (markierte Punkte) für einen hochgradig konvergierten Suchlauf von SPEA2 über 100 Generationen, einer Populationsgröße von 100 und einer Archivgröße von 60 Individuen. Hierbei repräsentiert die x-Achse des Zielfunktionsraums das Gewicht der Struktur [kg] und die y-Achse die maximale vertikale Knotenverschiebung [mm]. Die erzielten Pareto-Lösungen zeichnen aufgrund ihrer gleichmäßigen Verteilung den Verlauf der Pareto-Front deutlich nach.

Die Ergebnisse für eine Populationsgröße von 30 Individuen über 20 Generationen sind in den Abb. (6.22–6.25) für verschiedene Cross-over-Operatoren dargestellt. Wie schon bei den konstruierten Testproblemen im vorangegangenen Abschnitt liefert SPEA2 unter Verwendung des SBX-Operators eine gute Approximation und eine hohe Diversität der Pareto-Front im Zielfunktionsraum. Die folgende Tabelle beinhaltet die aufsteigend nach der Vertikalverschiebung sortierten Zielfunktionswerte der Individuen des Archivs am Ende des Suchlaufs aus Abb. (6.22).

Ind	Gewicht [kg]	max v [mm]
1	3723.8	4.875
2	3585.8	5.279
3	3594.8	5.390
4	3394.8	5.472
5	3199.0	5.747
6	3239.7	5.919
7	2982.6	6.378
8	2443.4	6.625
9	2081.2	7.907
10	2045.6	8.331
11	1756.9	9.007
12	1692.1	10.228
13	1687.1	11.217
14	1511.7	11.512
15	1399.9	11.852
16	1156.5	12.713
17	1120.7	15.871
18	1105.8	17.625
19	909.8	17.960
20	527.1	37.762

Die Lösung 16 stellt hierbei einen guten Kompromiß aus Gewicht und Vertikalverschiebung der Konstruktion dar. Das decodierte Chromosom der gewählten Lösung liefert die Radien der verwendeten Kreisquerschnitte der zehn Stäbe:

Element	1	2	3	4	5
r [m]	0.05012	0.03081	0.04072	0.04956	0.01998
Element	6	7	8	9	10
r [m]	0.00488	0.01761	0.05593	0.03913	0.00933

Die gezeigte Auswahl einer Lösung aus dem Pareto-Set bildet zwar den Abschluß des Mehrzieloptimierungsprozesses, oftmals ist aber gerade der Verlauf der Pareto-Front und die daraus ableitbaren Zusammenhänge der Zielfunktionsgrößen das gewünschte Ziel des Entscheidungsfinders. Um dies zu erreichen, bietet sich die Anwendung von Antwortflächenverfahren an, die auf der Basis der ermittelten Pareto-Lösungen eine kontinuierliche Approximation der Pareto-Front liefern.

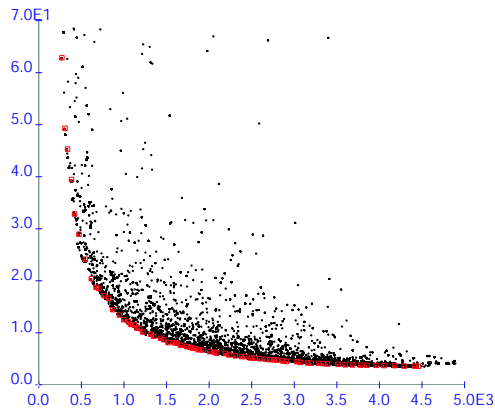


Abbildung 6.21: SPEA2-Archiv nach $T=100$ ($\alpha=60$, $\mu=40$, SBX ($n_c=1$)).

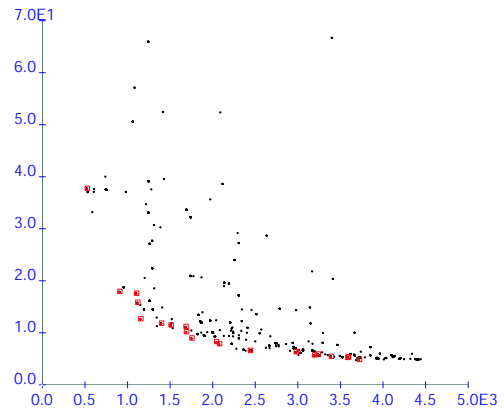


Abbildung 6.22: SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, SBX ($n_c=0$)).

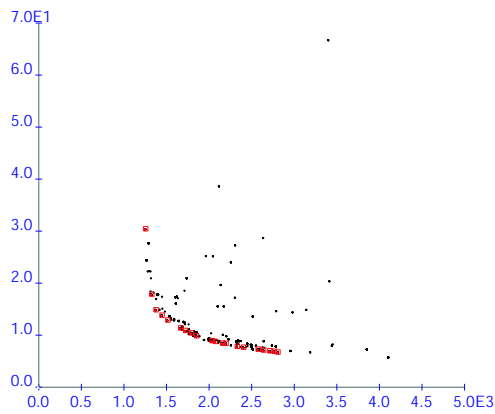


Abbildung 6.23: SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, uniformes XO ($p_c=0.5$)).

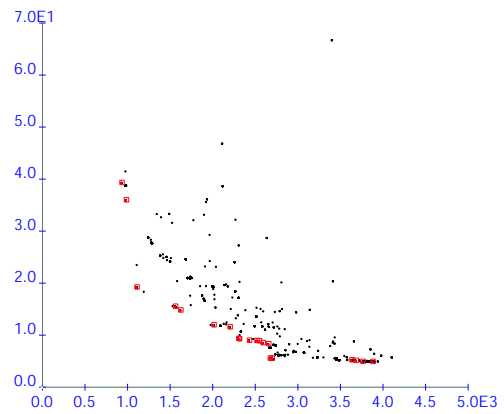


Abbildung 6.24: SPEA2-Archiv nach $T=20$ ($\alpha=20$, $\mu=10$, singlepoint XO (reelle Codierung)).

7

Zusammenfassung

Die Lösung eines Mehrzieloptimierungsproblems besteht aus einer Front Pareto-optimaler Lösungen, die durch die Anwendung evolutionärer Algorithmen in einem einzigen Suchlauf ermittelt werden können. Der SPEA2-Algorithmus hat sich als geeignet erwiesen, die Pareto-Front mit einer hohen Diversität der Lösungen zu approximieren. Allerdings ist das Maß der Konvergenz abhängig von der Populationsgröße und der Anzahl der Generationen und somit von den zur Verfügung stehenden Computer-Ressourcen.

Es konnte gezeigt werden, daß auch für eine begrenzte Anzahl von Zielfunktionsauswertungen eine aussagekräftige Approximation der Pareto-Front erreicht werden kann. Die Wahl des Cross-over-Operators und dessen Suchleistung haben dabei großen Einfluß auf die Qualität der Ergebnisse. Ein fehlendes numerisches Konvergenzkriterium für den Fall, daß der Verlauf der Pareto-Front a priori nicht bekannt ist, macht eine qualitative Bewertung von Ergebnissen jedoch kaum möglich.

Literaturverzeichnis

- BAYER, V. ; BUCHER, C. ; EBERT, M. ; HUTH, O. ; PURKERT, G. ; RIEDEL, J. ; ROOS, D. ; SCHORLING, Y. ; ZABEL, V.: *SLang* - the Structural Language. Version 5.0. Weimar, Germany : Institute of Structural Mechanics - Bauhaus-University Weimar, 2004
- BÄCK, T. ; SCHWEFEL, H.-P.: Evolutionary Computation: An Overview. In: *Proceedings of 1996 IEEE International Conference on Evolutionary Computation (ICEC '96)*. Nayoya University, Japan : IEEE, 1996, S. 20–29
- COELLO, C. A. C.: List of references on evolutionary multi-objective optimization. (2004). – URL <http://www.lania.mx/~ccoello/EMOO/EMOObib.html>
- DAS, I. ; DENNIS, J. E.: *Normal-Boundary Intersection: An alternate method for generating the pareto surface in nonlinear multicriteria optimization problems*. Hampton VA, USA : Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, 1996 (ICASE Report 96-62)
- DEB, K.: An efficient constraint handling method for genetic algorithms. In: *Applied Mechanics and Engeneering* 186 (2000), Nr. 2–4, S. 311–338
- DEB, K. ; AGRAWAL, R. B.: Simulated binary crossover for continuous search space. In: *Complex Systems* 9 (1995), S. 115–148
- DEB, K. ; AGRAWAL, S. ; PRATAB, A. ; MEYARIVAN, T.: *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II*. Kanpur, India : Indian Institute of Technology, 2000 (KanGAL Report 200001)
- DEB, K. ; THIELE, L. ; LAUMANN, M. ; ZITZLER, E.: Scalable Multi-Objective Optimization Test Problems. In: *Congress on Evolutionary Computation (CEC'2002)* Bd. 1. Piscataway, New Jersey : IEEE Service Center, 2002, S. 825–830

- FONSECA, C. M. ; FLEMING, P. J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1993, S. 416–423
- GOLDBERG, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts, USA : Addison-Wesley Publishing Company, 1989
- GOLDBERG, D. E. ; RICHARDSON, J.: Genetic algorithms with sharing for multimodal function optimization. In: *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum, 1987, S. 41–49
- HORN, J. ; NAFPLIOTIS, N.: *Multiobjective Optimization using the Niche Pareto Genetic Algorithm*. Urbana, Illinois, USA : University of Illinois at Urbana-Champaign, 1993 (IlliGAI Report 93005)
- MARLER, R. T. ; ARORA, J. S.: *Review of Multi-Objective Optimization Concepts and Algorithms for Engineering*. University of Iowa, USA : Optimal Design Laboratory, 2003 (Technical Report ODL-01.03)
- SCHAFFER, J. D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*. Lawrence Erlbaum, 1985, S. 93–100
- SILVERMAN, B. W.: *Density estimation for statistics and data analysis*. London : Chapman and Hall, 1986
- ZITZLER, E. ; LAUMANN, M. ; THIELE, L.: *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Zurich, Switzerland : Swiss Federal Institute of Technology (ETH) Zurich, 2001 (Computer Engineering and Networks Laboratory (TIK), Technical Report 103)
- ZITZLER, E. ; THIELE, L.: *An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach*. Zurich, Switzerland : Swiss Federal Institute of Technology (ETH) Zurich, 1998 (Computer Engineering and Networks Laboratory (TIK), Technical Report 43)

Anhang A

Dokumentation der SLang-Eingabedateien

Im Folgenden werden die `SLang`-Eingabedateien, die für die Mehrzieloptimierung mit SPEA2 erstellt worden sind, zusammengestellt und erläutert. Im einzelnen sind dies:

<code>MainSPEA.s</code>	Steuerung des Optimierungsablaufes
<code>InitSPEA.s</code>	Definition der SPEA2-Parameter
<code>Problem.s</code>	Definition der Parameter des Mehrzieloptimierungsproblems
<code>SPEA.s</code>	Implementation des SPEA2-Algorithmus
<code>MOTestProblems.s</code>	Testprobleme zur Mehrzieloptimierung
<code>10bar-fem.s</code>	10-Stab-Fachwerk FE-Analyse
<code>10bar-truss.s</code>	10-Stab-Fachwerk Strukturmodell

Die `SLang`-Eingabedateien `BinFile.s`, `Common.s`, `Functions.s`, `InitMain.s` und `SLangSettings.s` sind Bestandteil der `OptiSLang`-Umgebung und im Rahmen dieser Arbeit nur teilweise eingebunden. Die verwendeten Kommandos entsprechen der `SLang`-Version 5.0. Für die Kommandos, die im Folgenden nicht erläutert sind, sei auf die `SLang`-Dokumentation (Bayer u. a. (2004)) hingewiesen.

A.1 Definition der SPEA2-Parameter (`InitSPEA.s`)

```
#label InitSPEA
int_def  alpha      20
int_def  mu         10
int_def  tournament  2
int_def  nGnr       20
```

—— Festlegung von Archivgröße (α), Anzahl der Elternindividuen (μ), Turniergröße für die Elternselektion (`tournament`) und Anzahl der Generationen (`nGnr`).

```
real_def  MutRate          0.1
real_def  MutStddevBegin  0.1
real_def  MutStddevEnd    0.0010
```

—— Definition der Mutationsrate und der Standardabweichung zu Beginn und Ende des Suchlaufs.

A.2 Definition der Parameter des Mehrzieloptimierungsproblems (`Problem.s`)

```
#label initProblem
int_def  tpId  4
int_def  dim   2
int_def  nPar  10
```

—— Auswahl des Testproblems (`tpID`) und Festlegung der Anzahl der Zielfunktionen (`dim`) und der Anzahl der Designvariablen (`nPar`).

```
object create, real vector, nPar, UpperBound /
object initialize, , UpperBound 90e-3, /
object create, real vector, nPar, LowerBound /
object initialize, , LowerBound 4e-3, /
```

—— Definition der oberen und unteren Variablen-
grenzen. Außerdem erfolgt die Formulierung von Nebenbedingungen.

A.3 Implementation des SPEA2-Algorithmus (`SPEA.s`)

Das Label `SPEA_Algorithm` steuert den Ablauf des gesamten Optimierungsalgorithmus. Zunächst werden die Zufallsvariablen für die Mutation erzeugt und der Startwert des Zufallszahlengenerators gesetzt.

```
#label SPEA_Algorithm
```

```
  genetic allocate, replace set_seed, 1 SimSeedGA, /

  object copy, , LowerBound, gendata /
  object append, add_columns, gendata UpperBound , /
  genetic initialize, replace, 1 gendata alpha, /
```

— Die zufällige Initialisierung der Startpopulation der Größe α erfolgt anhand der festgelegten Variablen Grenzen.

```
#label nextGnr
```

```
  genetic extract, replace population, 1, pp_all /
  control gosub, , GetObjectiveVector, /
  control gosub, , calcConstraintViolation, /
```

— Nach der Decodierung der Chromosomen der Individuen der Population werden die Zielfunktionswerte und das Maß der Verletzung der Nebenbedingungen für alle Individuen ermittelt.

```
  control gosub, , calculateStrength, /
  control gosub, , calculateFitness, /
  control gosub, , calculateDistances, /
```

— Durch Aufruf der Routinen zu Fitneßzuweisung erfolgt die dominanzbasierte Rangbildung der Individuen. Weiterhin werden Abstände aller Lösungen zueinander im Zielfunktionsraum ermittelt.

```
  control gosub, , environmentalSelection, /
  control gosub, , matingSelection, /
```

— Die Selektion der Individuen des Archivs der nächsten Generation und der Elternindividuen erfolgt in den Routinen zur Selektion.

```
  control gosub, , recombination, /
  control gosub, , mutation, /
  genetic next_generation, replace, 1, Gnr /
control if, less, Gnr nGnr nextGnr, /
```

— Schließlich werden die Variationsoperatoren auf die selektierten Elternindividuen angewandt und der Sprung zur nächsten Generation vollzogen. Die Generationsschleife wird so lange durchlaufen, bis die maximale Anzahl von Generationen erreicht ist.

Die im Verlauf des SPEA2-Algorithmus aufgerufenen Subroutinen sollen nachfolgend anhand ihrer Absicht und der wichtigsten Ein- und Ausgabegrößen erläutert werden.

#label GetObjectiveVector

Input: *pp_all, tpID*

Output: *ObjectiveVectors, constraintResponse*

Absicht: Für die Individuen der Population werden die Zielfunktionswerte und die Werte für die Nebenbedingung für das gewählte Testproblem ermittelt. Jede Zeile der Matrix *ObjectiveVectors* repräsentiert den Zielfunktionsvektor einer Lösung.

#label calcConstraintViolation

Input: *pp_all, nInequal, constraintResponse*

Output: *violation*

Absicht: Die Verletzung der Nebenbedingung wird für jede Lösung ermittelt und im Vektor *violation* gespeichert.

#label calculateStrength

Input: *pp_all, ObjectiveVectors, violation*

Output: *strength_vector*

Absicht: Für jedes Individuum der Population wird die Anzahl der von ihm dominierten Lösungen bestimmt. Die strength-Werte der Individuen werden im Vektor *strength_vector* gespeichert.

#label calculateFitness

Input: *ObjectiveVectors, violation, strength_vector*

Output: *raw_fitness_vector, fitness_bucket, fitness_bucket_mod*

Absicht: Die Fitneß jedes Individuums wird aus der Summe der strength-Werte der Lösungen bestimmt, die dieses Individuum dominieren. Die Fitneßwerte werden im Vektor *raw_fitness_vector* gespeichert. Die Vektoren *fitness_bucket* und *fitness_bucket_mod* enthalten die Informationen, wie viele Individuen einen bestimmten Fitneßwert besitzen.

#label calculateDistances

Input: *Objective Vectors*
Output: *dist, copies, NN*

Absicht: Der Abstand jeder Lösung zu jeder anderen Lösung der Population wird ermittelt und in der Matrix *dist* gespeichert. Gleichzeitig wird überprüft, ob Lösungen mehrfach vorhanden sind, und die Häufigkeit des Vorkommens jeder Lösung im Vektor *copies* gespeichert. Die Matrix *NN*, die auf den *k*-ten Nachbarn einer Lösung *i* verweist, wird initialisiert.

#label dominates

Input: *a_objective, b_objective, a_violation, b_violation*
Output: *dominates*

Absicht: Das Verhältnis zweier Lösungen *a* und *b* wird bezüglich des Kriteriums der Pareto-Dominanz unter Nebenbedingungen bestimmt. Die Subroutine liefert für die Variable *dominates* den Wert 1, wenn Lösung *a* Lösung *b* dominiert. Andernfalls wird der Wert 0 zurückgegeben.

#label environmentalSelection

Input: *fitness_bucket, alpha*
Output: *old_index*

Absicht: Die Hauptroutine zur Archivselektion ruft die Funktionen zum Auffüllen oder zur Reduktion des Archivs auf und liefert den Vektor *old_index* mit den Indizes der für das Archiv der nächsten Generation selektierten Individuen.

#label truncate_nondominated

Input: *raw_fitness_vector, copies, dist*
Output: *killed*

Absicht: Aus der Menge der nicht dominierten Lösungen werden α Lösungen für das Archiv der nächsten Generation ausgewählt. Die nicht selektierten oder dominierten Individuen werden als gelöscht markiert.

#label archive_densityInput: *count, max_copies, marked*Output: *marked*

Absicht: Die Variable *count* repräsentiert die Anzahl der Lösungen, die mit der Häufigkeit *max_copies* in der Population vorkommen. Der Vektor *marked* enthält die Indizes dieser Lösungen. Die Subroutine ermittelt die Lösung, die den kleinsten Abstand zu ihrem *k*-ten Nachbarn aus der Menge der markierten Lösungen besitzt. Der Index der ermittelten Lösung wird als erste Zeile von *marked* gespeichert.

#label truncate_dominatedInput: *fitness_bucket_mod, fitness_bucket, raw_fitness_vector*Output: *killed*

Absicht: Aus der Menge der dominierten Lösungen werden die Besten ausgewählt, um das Archiv der nächsten Generation aufzufüllen. Die nicht selektierten Individuen werden als gelöscht markiert.

#label population_densityInput: *num, raw_fitness_vector*Output: *killed*

Absicht: Falls mehrere dominierte Lösungen mit identischer Fitness zur Auswahl stehen, werden aus den noch nicht selektierten Individuen diejenigen zum Auffüllen des Archivs ausgewählt, die den größten Abstand zu ihrem nächsten Nachbarn in der Population besitzen. Alle dominierten Individuen, die nicht für das Archiv ausgewählt wurden, werden als gelöscht markiert.

#label getNNInput: *index, k, dist*Output: *min_index*

Absicht: Für ein Individuum (*index*) wird der *k*-te Nachbar in der Population ermittelt und dessen Index als *min_index* zurückgegeben.

#label getNNd

Input: *index, k, dist*
Output: *NNd*

Absicht: Für ein Individuum (*index*) wird der Abstand zu seinem *k*-ten Nachbarn in der Population ermittelt und als *NNd* gespeichert. Die Subroutine gibt für *NNd* den Wert (-1) zurück, wenn der ermittelte *k*-te Nachbar bereits als gelöscht markiert wurde.

#label matingSelection

Input: *raw_fitness_vector, tournament, mu, dist, old_index*
Output: *parents*

Absicht: Zur Auswahl der Elternindividuen aus der Menge der Lösungen des Archivs wird eine binäre Turniersélection durchgeführt. Der Index des jeweiligen Gewinners wird im Vektor *parents* gespeichert, bis die Anzahl μ der zu selektierenden Eltern erreicht ist. Danach werden die Individuen des Archivs (*old_index*) als Elite markiert, um deren Mutation zu verhindern.

#label recombination

Input: *parents*
Output:

Absicht: Auf die selektierten Elternindividuen wird ein Cross-over-Operator angewendet.

#label mutation

Input: *MutStddevMat, DataRange, MutRate, MutationSet1*
Output:

Absicht: Ein Mutationsoperator wird auf die Population angewendet. Dabei sind die Individuen des Archivs von der Mutation ausgeschlossen.

Anhang B

Dokumentation des SLang-Kommandos

genetic xover

Keywords: genetic, xover, crossover, singlepoint, shuffle, chromosom

Attributes: shuffle, singlepoint/multipoint/segmented/uniform/arithmetic/sbx/copy, get_start_seed/set_start_seed, marriage_broker/no_marriage_broker

Inputs: ident: *population_id*
object (INTEGER VECTOR): *parents*
...
if **sbx**: object (REAL): *distribution_parameter*
if **set_start_seed**: object (INTEGER): *seed*

Outputs: if get_start_seed: object (INTEGER): *seed*

Example: genetic xover, sbx, 1 parents 5, /
* The selected xover operator is applied to the individuals of population 1 specified in parents. The offspring individuals are generated by performing simulated binary crossover on randomly chosen pairs of parents and are appended to the population. The sbx-operator uses the distribution_parameter 5. /

Action: If **copy** the individuals specified by parents are duplicated and appended to the population. If not copy different crossover (*xover*) methods are performed on pairs of parents out of object *parents*. The offsprings are appended to the population and marked internally as new born (see genetic extract, new borns, ...) until command genetic next generation or genetic modify, fitness, clears this flag. In case the number of parents is an odd number one random parent out of parents is used twice and the additional offspring is removed. This is necessary for the *xover* algorithm which operates on pairs of parents.

sbx performs simulated binary crossover on the chromosomes of the selected parents. Each gene of the chromosome is chosen for crossover with probability 0.5. A small value of *distribution_parameter* allows solutions far away from parents to be created and a large value restricts only near-parent solutions to be created as offspring. *distribution_parameter* can take any non-negative value but values between 0 and 5 are recommended.

Date: 25.07.2004

FurtherDocumentation: -

Erklärung

Ich erkläre, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Weimar, 26.07.2004

Thesen

Die meisten praktischen strukturmechanischen Optimierungsprobleme beinhalten mehrere, oftmals konkurrierende Zielfunktionen, die bei der Lösung des Problems gleichzeitig berücksichtigt werden müssen. Dies führt, im Gegensatz zur Einzieloptimierung, bei der nur ein Optimum gefunden werden kann, zu einer Front Pareto-optimaler Lösungen.

- Pareto-optimal ist eine Lösung dann, wenn es keinen Entscheidungsvektor gibt, der für ein Zielkriterium eine Verbesserung darstellen würde, ohne gleichzeitig mindestens ein anderes zu verschlechtern.

Eine Methode zur Mehrzieloptimierung muß deshalb folgende Anforderungen erfüllen:

- Finde eine Anzahl von Lösungen nahe den Pareto-optimalen Lösungen.
- Finde Lösungen, die verschieden genug sind, um die gesamte Ausbreitung der Pareto-Front zu repräsentieren.

Evolutionäre Algorithmen sind geeignete Verfahren zur Lösung von Mehrzieloptimierungsproblemen, da sie mehrere Pareto-optimale Lösungen in einem einzigen Suchlauf ermitteln können.

Der SPEA2-Algorithmus verbindet verschiedene Konzepte der Bewertung und Selektion von Individuen zur Mehrzieloptimierung.

- Die Fitneßzuweisung berücksichtigt für jedes Individuum die Anzahl der Individuen, die es dominiert und von denen es dominiert wird. Dadurch erfolgt eine differenzierte Rangbildung zwischen den Individuen einer Population.
- Durch Anwendung des Prinzips des Elitismus wird verhindert, daß die besten Lösungen durch Variationsoperatoren verschlechtert werden. Sie werden unverändert in die nächste Generation übernommen.
- Neben der rangbasierten Fitneß dient ein Maß zur Dichteabschätzung der Gewährleistung der Diversität der Population. Es erhöht den Selektionsdruck auf Randlösungen.

Die Berücksichtigung von Nebenbedingungen läßt sich durch die Modifikation des Dominanz-Kriteriums in den SPEA2-Algorithmus einbinden.

Die Ergebnisse von SPEA2 für verschiedene Testprobleme haben die Abhängigkeit der Konvergenz von folgenden Faktoren gezeigt:

- Größe der Population und Verhältnis zwischen Archivgröße und Anzahl der Nachkommen
- Anzahl der Generationen

- Begrenzung des Variablensuchraums
- Dimension des Zielfunktionsraums
- Suchleistung des Cross-over-Operators

Für eine hohe Populationsgröße und eine hohe Anzahl von Generationen findet SPEA2 eine hochgradig konvergierte Approximation der Pareto-Front. Dies erfordert jedoch eine große Anzahl von Zielfunktionsauswertungen, was für ein praktisches Ingenieurproblem hohe Anforderungen an die Computer-Ressourcen stellt.

Für eine begrenzte Anzahl von Zielfunktionsauswertungen ist die Qualität der Lösungen abhängig von der Suchkraft des Cross-over-Operators und von der Dimension des Zielfunktionsraums.

- Eine große Anzahl von Zielfunktionen führt zu einem schnellen Anstieg der Anzahl nicht dominierter Lösungen.
- Es kommt zu einer Stagnation der Konvergenz aufgrund des schwachen richtungsgebenden Selektionsdrucks.

Wird durch die Variation ein Individuum gefunden, das viele Lösungen des Archivs dominiert, kann wieder eine differenziertere Rangordnung hergestellt werden. Dies kann geschehen durch:

- Anwendung eines Cross-over-Operators mit hoher Suchleistung.
- Zufällige Reinitialisierung eines Teils der Population.

Eine weitere denkbare Modifikation von SPEA2 für die Anwendung mit begrenzten Zielfunktionsauswertungen ist die Variabilität der Größe des Archivs. Alle nicht dominierten Individuen würden in das Archiv für die nächste Generation übernommen. Dies setzt jedoch eine klare Begrenzung der Generationsanzahl voraus.

Enthält die Pareto-Front am Ende eines Optimierungslaufs nur wenige, aber gleichmäßig verteilte Lösungen, kann der Verlauf der Pareto-Front durch eine Antwortflächenmethode approximiert werden.