Vergleich Stochastischer Verfahren zur Globalen Optimierung

Diplomarbeit zur Erlangung des akademischen Grades

Magister der Naturwissenschaften

an der Formal- und Naturwissenschaftlichen Fakultät

der Universität Wien

eingereicht von

Erich Janka

Wien, im April 1999

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Worum es nicht geht

In der Mathematik treten immer wieder Fragestellungen auf, bei denen möglichst kleine, einfache oder allgemein irgendwie „optimale“ mathematische Objekte gesucht werden. So werden in der Algebra beispielsweise maximale Ideale, Primkörper oder der größte gemeinsame Teiler untersucht. In der Topologie betrachtet man die kleinste Menge, welche alle Grenzwerte einer gegebenen Menge enthält, also den Abschluß. Man kann fragen, welche geschlossene, stetige Kurve mit gegebener Bogenlänge das Gebiet mit maximaler Fläche umschließt. Diese Aufzählung ließe sich vermutlich beinahe endlos fortsetzen. Optimieren kann man deshalb wohl zu recht als grundlegende Idee bezeichnen. Da unter „optimal“ in den oben angeführten Beispielen oft grundsätzlich Verschiedenes verstanden wird, scheint es mir so gut wie unmöglich zu sein sie alle mit einer einheitlichen Methode zu behandeln. Der hier angedeutete Begriff von „Optimierung“ ist für die Zielsetzung dieser Arbeit viel zu weit gefaßt.

1.2 Worum es geht

Sei es in Physik, Technik, Wirtschaft oder in einem von vielen anderen Gebieten, oft tauchen Fragestellungen auf, die nach „optimalen“ Lösungen verlangen. Ohne genauer auf die Probleme einzugehen will ich hier stichwortartig nur kurz ein paar Beispiele geben:

- Motoren sollen minimalen Verbrauch haben.
- Die Leiterbahnen von Microchips sollen möglichst geringen Raum beanspruchen.
- Lieferwege sollen minimiert werden.
- Gewinne sollen maximiert werden.
- Licht nimmt immer den „kürzesten“ Weg.
- Raumsonden sollen möglichst wenig Zeit zu ihrem Ziel benötigen.
- Materialverbrauch soll möglichst klein sein.

Diese Liste ließe sich noch beliebig verlängern.

Solche Probleme lassen sich oft mathematisch modellieren, sich also in die Sprache der Mathematik übersetzen, wo man sie dann einheitlich behandeln kann. Das geschieht meist mit Programmen, welche die zur Verfügung stehenden Daten dazu verwenden, das Problem so gut es ihnen möglich ist zu lösen.

Diese Arbeit versucht die verschiedenen Eigenschaften von Algorithmen zur globalen Optimierung aufzuzeigen. Viele Fragen können nur beantwortet werden, wenn man konkrete Ausformungen der Algorithmen in Gestalt von Software-Programmen betrachtet. Dabei sollte man sich aber im klaren sein, daß Aussagen über ein Programm nicht unbedingt auch Aussagen über den Algorithmus sein müssen. (Wenn man einen guten Algorithmus schlecht implementiert, kann er wohl keine guten Ergebnisse liefern.) Aus obigem Grund wird im folgenden auch oft von konkreten Software-Programmen die Rede sein.

Ein Ziel dieser Arbeit war es, einen Überlick über Verfahren zur globalen Optimierung im allgemeinen und speziell jenen, die stochastische Elemente beinhalten, zu geben. Dazu gehörte auch die Berücksichtigung und Behandlung von Nebenbedingungen. Ein weiteres bestand darin, verschiedene, frei verfügbare Softwarepakete zu implementieren, an Hand von konkreten Testproblemen zu vergleichen. Weiters soll für jedes getestete Programm an Hand der Funktionsauswertungen ein Abbruchkriterium bestimmt werden. Letztendlich sollen die Ergebnisse dieser Arbeit im `HTML`-Format aufbereitet und auf den Internet-Seiten von Neumaier [33] und von mir [22] verfügbar gemacht werden.

Kapitel 2

Das Problem

2.1 Die Aufgabenstellung

Das Optimierungsproblem stellt sich in der Mathematik als die Aufgabe, ein Minimum einer reellwertigen Funktion zu finden. Präziser formuliert heißt das:
Zu einer gegebenen reellwertigen Funktion

$$f : D \to \mathbb{R}, \quad D \subseteq \mathbb{R}^n$$

wird ein x mit folgenden Eigenschaften gesucht:

$$x \in D : \quad \text{es gibt ein} \quad U(x) \subseteq D \quad \text{mit} \quad f(x) \leq f(z) \quad \text{für alle} \quad z \in U(x)$$

D ist dabei der Definitionsbereich der Funktion f und $U(x)$ steht für eine offene Umgebung des Punktes x. Man sucht also einen Punkt im Definitionsbereich, für den es eine (vielleicht extrem kleine) Umgebung gibt, sodaß in dieser Umgebung kein Punkt mit größerem Funktionswert als $f(x)$ existiert. Wenn man so ein x gefunden hat, dann nennt man das Paar ein $(x, f(x))$ ein *lokales Minimum.*

In vielen Fällen will man aber mehr. Oft reicht es nicht aus, ein lokales Minimum zu finden, sondern man will wirklich einen Punkt x mit der Eigenschaft finden, daß es im ganzen Definitionsbereich keinen Punkt mit einem geringeren

Funktionswert gibt. Das läßt sich erreichen, indem man in obiger Definition fordert, daß $U(x) = D$ ist, oder anders formuliert:
Sei eine reellwertige Funktion f gegeben:

$$f : D \to \mathbb{R}, \quad D \subseteq \mathbb{R}^n,$$

dann wird ein x mit folgenden Eigenschaften gesucht:

$$x \in D : \quad f(x) \leq f(z) \quad \text{für alle} \quad z \in D$$

Der Wert $f(x)$ heißt dann *globales Minimum*, und x wird (globaler) *Minimalpunkt* genannt. Die Funktion f nennt man auch *Zielfunktion*. Im folgenden werde ich einen Minimalpunkt auch kurz mit x^* und das globale Minimum $f(x^*)$ mit f^* bezeichnen.

Es stellt übrigens keine Beschränkung der Allgemeinheit dar, wenn man sich nur mit Minimierung und nicht mit Maximierung befaßt, da man letztere stets auf erstere zurückführen kann:

$$\max_{x \in D}(f(x)) \quad = \quad -\min_{x \in D}(-f(x))$$

Deshalb werde ich ab jetzt die Begriffe *Minimum* und *Optimum* synonym verwenden. *Globale Optimierung* befaßt sich nun damit, wie man derartige Aufgabenstellungen „zufriedenstellend“ löst, wie man also (globale) Minima f^* tatsächlich berechnen kann. Das läuft letzten Endes darauf hinaus, Computerprogramme zu schreiben, die in der Lage sind das Minimum einer gegebenen Funktion „ausreichend“ anzunähern.

Jedenfalls ist zu klären, was die Begriffe „zufriedenstellend“ beziehungsweise „ausreichend“ überhaupt bedeuten sollen. Das hängt aber weniger von der verwendeten Mathematik ab, als von Computern und den zugrundeliegenden Problemen und diese kommen meist aus Anwendungsgebieten.

2.2 Einschränkungen und Schwierigkeiten

Die Aufgabenstellung enthält noch gewisse Einschränkungen und Schwierigkeiten, die dazu führen, daß man gewisse Abstriche bezüglich der Güte der Lösung hinnehmen muß:

- In der Wirtschaft ist etwa Zeit (die ja bekanntlich gleich Geld ist) ein sehr wichtiger Faktor, und wenn die Berechnung einer optimalen Lösung zuviel Zeit in Anspruch nimmt, so gibt es eben keine andere Alternative, als sich mit einer ungenauen (aber möglichst guten) Lösung zufriedenzugeben.
- Enthält ein Problem ungenaue Meßdaten, so ergeben diese zwangsläufig auch eine ungenaue Lösung.
- Es ist sinnlos 100 Stellen einer Größe zu berechenen, wenn nur zwei für das Problem relevant sind.
- Computer können Zahlen nur mit einer gewissen Genauigkeit darstellen und deswegen kommt es unweigerlich zu Rundungsfehlern.
- Es gibt zwar Computer-Algebraprogramme, wie etwa `Mathematica` oder `Maple`, die in der Lage sind exakt zu optimieren. Diese sind aber für globale Optimierung ungeeignet, da sie viel zu langsam sind.

2.3 Was ist eine Lösung

Es ist eine Aufgabe des Problemstellers, zu beschreiben, was eine Lösung überhaupt sein soll. Hier sind mehrere Möglichkeiten denkbar:

- Finde eine möglichst gute Annäherung an ein globales Optimum (unter vorgebegenen Rahmenbedingungen).
- Finde *ein* globales Optimum.

- Finde möglichst viele (alle) globale Optima.
- Finde einen Punkt $\bar{x}$, dessen Funktionswert nur um eine vorher definierte Toleranz ε vom Optimum abweichen darf:

  $$\bar{x} \in D : \quad \|f(\bar{x}) - f^*\| \leq \varepsilon$$

  $f(\bar{x})$ nennt man auch ε-Minimum.

Die Aufgabenstellung an die Mathematik ist es nun, Algorithmen und infolge Programme zu entwickeln, um den oben gestellten Forderungen zu entsprechen.

Kapitel 3

Die Zielfunktion

Da an die Funktion f noch keine weiteren Anforderungen gestellt wurden, kann sie sich noch beliebig „bösartig“ verhalten. Da die Aufgaben aber meist nicht aus der Mathematik selbst kommen, sondern von konkreten (außermathematischen) Problemstellungen hergeleitet sind, kann man annehmen, daß sich die Funktionen nicht allzu pathologisch verhalten. Selbst „mathematische“ Funktionen verhalten sich meist nicht schlecht, zumindest wenn man von konstruierten Beispielen absieht. Meistens kann man wohl voraussetzen, daß die zu optimierende Funktion (n-mal) stetig differenzierbar oder stetig ist oder zumindest stückweise „glatt“ oder stückweise stetig. Auch vom Definitionsbereich D kann man aus den selben Gründen annehmen, daß er sich „vernünftig“ verhält. Um mathematische Aussagen treffen zu können, wird vielfach vorausgesetzt, daß D kompakt, also beschränkt und abgeschlossen ist.

Trotzdem bleibt es noch sehr schwer, allgemeine theoretische Aussagen herzuleiten, die auch praktische Relevanz besitzen. Ist etwa die zu optimierende Funktion und der Definitionsbereich konvex, so ist jedes lokale zugleich globales Optimum und man kann sich auf Verfahren der lokalen Optimierung beschränken. Leider ist diese Bedingung nur allzu selten erfüllt.

Aber selbst wenn eine Funktion unendlich oft stetig differenzierbar ist, kann sie zwischen zwei „Meßpunkten“ sehr starke Schwankungen aufweisen. Hier wird eine weitere Einschränkung angedeutet, die aus der Verwendung von Computern resultiert. Computer rechnen numerisch, können Funktionen also nur an endlich vielen Punkten auswerten und zwischen zwei solchen benachbarten Punkten entzieht sich die Funktion der Handhabung. Kennt man eine untere Schranke S_U und eine obere Schranke S_O für die Ableitung ∇f auf einer Teilmenge $M \subset D$, welche zwei Punkte x_1 und x_2 enthält, so bleiben die Funktionswerte zwischen x_1 und x_2 jedenfalls durch

$$f(x_1) - K\|x_1 - x_2\|_2 \qquad \text{und} \qquad f(x_1) + K\|x_1 - x_2\|_2 \tag{3.1}$$

beschränkt. Dabei ist $K = \max\{\|S_O\|_1, \|S_U\|_1\}$. Nun können diese Schranken aber riesengroß sein, beziehungsweise brauchen sie gar nicht zu existieren. Derartige Schranken existieren nämlich nur für auf M Lipschitz-stetige Funktionen (siehe Definition 4.9, S. 23).

Man sieht, daß man nur für wirklich „gutartige“ Funktionen darauf hoffen kann, sie mit Computern zu optimieren, beziehungsweise Aussagen darüber treffen zu können, wie groß die Chancen eines gegebenen Computerprogramms sind, das Problem zu lösen. Wie die „wirklich“ vorkommenden Funktionen sich meist nicht sehr schlecht verhalten, so sind aber auch wenige so einfach (etwa konvex) gebaut, daß das Optimum ohne viel Aufwand zu bestimmen wäre. Programme zur Globalen Optimierung versuchen also wenigstens die „mittelschweren“ Probleme zufriedenstellend zu lösen.

Die Information, welche man von der Zielfunktion besitzt, kann je nach Problem sehr stark variieren. Dementsprechend sind auch eine Vielzahl von Algorithmen und infolge von Programmen möglich, die verschiedene Arten von Information zu nutzen verstehen. Jedenfalls muß es möglich sein, den Funktionswert für einen Punkt x im Definitionsbereich zu bestimmen. Aber das allein kann schon sehr langwierig und umständlich sein. So ist es etwa möglich, daß die Auswertung einer Funktion sehr kompliziert und rechenintensiv ist oder sogar irgendeine

Messung dafür notwendig ist. In diesem Fall ist es offensichtlich wichtig, daß das gewählte Verfahren möglichst sparsam mit Funktionsauswertungen umgeht. Weiters ist es möglich, daß Ableitungen der Zielfunktion verfügbar sind oder sogar Krümmungsinformation in Form der Hessematrix. Es kann aber auch vorkommen, daß man nicht einmal einen Funktionswert exakt bestimmen kann. Das tritt etwa dann ein, wenn die Funktionswerte aus Meßergebnissen bestehen, also verrauscht sind. Ist das der Fall, so weiß man darüber aber meist Bescheid und kann sich dementsprechend verhalten.

So wie verschiedene Informationen über die Zielfunktion vorhanden sind, so verwendet ein bestimmtes Software-Programm dementsprechend diverse Informationen, um ein gegebenes Problem zu lösen. Bei der Auswahl eines Programms muß man also erst einmal darauf achten, daß das Programm nicht mehr Daten benötigt, als man überhaupt zu bieten hat. Programme, die allein mit Funktionswerten auskommen, sind wohl am universellsten einsetzbar, lassen aber andererseits oft vorhandene Information unberücksichtigt und können dadurch stark an Effizienz verlieren. Auch hängt die Gewißheit, daß ein Programm ein Optimum gefunden hat von Ableitungsinformationen ab.

Kapitel 4

Der Definitionsbereich

Wie bereits erwähnt, handelt es sich bei Zielfunktionen prinzipiell um Funktionen der Art:

$$f: \quad D \to \mathbb{R}, \quad D \subseteq \mathbb{R}^n$$

Im folgenden will ich nicht auf die Gestalt der Funktion selbst, sondern vielmehr auf das Wesen des Definitonsbereiches eingehen. Wenn man das Problem auf die oben dargestellte Weise formuliert, dann ist das für die rechnerische Behandlung umständlich. Der Definitionsbereich D besteht aus einer Teilmenge des $\mathbb{R}^n$ und enthält meist unendlich viele Punkte, die man nicht mehr aufzählend angeben kann. Für mathematische Aussagen über das Verhalten von verschiedenen Algorithmen wird oft auch gefordert, daß D kompakt ist. Nun schreibt man aber die Zielfunktion zuerst mit ganz $\mathbb{R}^n$ als Definitionsbereich. Da man damit aber viel mehr Punkte zuläßt, muß man die überzähligen Punkte wieder beseitigen. Das geschieht so, daß man noch *Nebenbedingungen* hinzufügt, welche die Punkte erfüllen müssen. Die Nebenbedingungen können aus Gleichungen oder Ungleichungen bestehen, sie können aber auch eine Menge M aus zugelassenen Punkten enthalten. M muß dabei eine dem Computer verständliche Gestalt besitzen und sollte möglichst einfach darstellbar sein. Zum Beispiel eine Aufzählung von endlich vielen Punkten oder ein der Programmiersprache bekannter Zahlenbereich (natürliche Zahlen werden etwa in der Programmiersprache `C` als Variablentyp „`int`" repräsentiert). Damit läßt sich das Optimierungsproblem auf folgende Art und Weise schreiben:

$$\begin{array}{lll} \min f(x): & \mathbb{R}^n \to \mathbb{R} & \\ \text{s.t.} & A(x) \geq 0; & A: \mathbb{R}^n \to \mathbb{R}^{m_1} \\ & B(x) = 0; & B: \mathbb{R}^n \to \mathbb{R}^{m_2} \\ & (x \in M; & M \subset \mathbb{R}^n) \end{array} \tag{A}$$

Dabei steht die Abkürzung s.t. für (engl.) „subject to" und gibt an, daß es sich beim folgenden um Nebenbedingungen handelt. $A(x) \geq 0$ steht darin für ein Reihe von Ungleichungen, die erfüllt sein müssen und $B(x) = 0$ steht analog für ein System von Gleichungen, und M ist eine Teilmenge des $\mathbb{R}^n$, in welcher x liegen muß. Gleichungen der Form $B(x) = 0$ könnte man überhaupt weglassen, da man jede auch als Paar von Ungleichungen schreiben kann:

$$B(x) = 0 \qquad \Leftrightarrow \qquad B(x) \geq 0 \quad \text{und} \quad -B(x) \geq 0$$

Dabei nimmt man aber oft in Kauf, daß man es der durch diese Nebenbedingungen bestimmten Teilmenge des $\mathbb{R}^n$ nicht mehr ansieht, daß sie keine inneren Punkte besitzt.

Der Betrag $|\cdot|$, sowie die Vergleichsoperatoren $=$, $\leq$, $<$, $\geq$ und $>$ sind in dieser Arbeit immer dann komponentenweise zu verstehen, wenn mindestens eines der beiden Argumente ein Vektor ist. So gilt zum Beispiel:

$$\left| \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix} \right| \qquad \Leftrightarrow \qquad \begin{pmatrix} |x_1| \\ \vdots \\ |x_k| \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix} \leq 0 \quad \Leftrightarrow \quad x_i \leq 0 \quad \text{für} \quad i = 1, \ldots, k \qquad \text{oder}$$

$$\begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix} \leq \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} \quad \Leftrightarrow \quad x_i \leq y_i \quad \text{für} \quad i = 1, \ldots, k$$

4.1 Nebenbedingungen in fertigen Programmen

Wie Nebenbedingungen berücksichtigt werden, hängt von den einzelnen Programmen ab, die jeweils bestimmte Arten von Nebenbedingungen zulassen.

Die einfachste Art von Nebenbedingungen sind die, welche nur die einzelnen Koordinaten des $\mathbb{R}^n$ betreffen. Nebenbedingungen dieser Art schneiden gewissermaßen einen (Hyper-)Quader aus dem $\mathbb{R}^n$ heraus. Diese sind ein Spezialfall von *linearen* Ungleichungs-Nebenbedingungen, welche ein Gebiet bestimmen, das durch ein Polyeder begrenzt wird und das gegebenfalls auch leer oder auch unbeschränkt sein kann. Obwohl die Beschränkung der Koordinaten ein Spezialfall ist, muß sie von Programmen nicht unbedingt als solcher behandelt werden und wird oft eigenständig verarbeitet.

Welche Arten von Nebenbedingungen zugelassen sind, hängt wieder von dem Programm ab, das man benutzen möchte. Auch für die Nebenbedingungen gilt, daß sie sich einigermaßen „vernünftig“ verhalten sollten. Es scheint klar zu sein, daß konkrete Programme bevorzugt einfachere Nebenbedingungen erlauben. So sind koordinatenweise Nebenbedingungen häufiger anzutreffen als lineare und diese wiederum häufiger als noch kompliziertere Nebenbedingungen.

Manche Programme unterstützen überhaupt keine Nebenbedingungen. In diesem Fall kann es passieren, daß das Programm die Funktion an Punkten auswerten will, die zwar im $\mathbb{R}^n$ liegen, aber für den Computers nicht mehr darstellbar sind. Das Ergebnis ist eine Fehlermeldung und ein Abbruch des Programms.

Wie man trotzdem verschiedene Nebenbedingungen verwenden kann, wird in Abschnitt 4.3 beschrieben.

4.2 Nebenbedingungen selbst programmieren

Hier möchte ich kurz behandeln, wie man Nebenbedingungen selbst in ein Programm implementieren kann. Die hier vorgestellte Idee ist es, ein Optimierungsproblem mit Nebenbedingungen durch ein anderes, leichter zu lösendes, zu approximieren. Selbstverständlich kann man dabei nur eine Annäherung an das *echte* Optimum erwarten. Deshalb konstruiert man eine Folge von Problemen (P_i), die gegen das ursprüngliche „konvergieren“ (in einer noch festzulegenden Weise) und hofft, daß die jeweiligen Optimalpunkte x_i^* gegen x^*, einen Optimalpunkt des ursprünglichen Problems, streben. Dabei definiert man sich neue Zielfunktionen, deren Minima, welche im interessanten Bereich liegen, auch ohne Verwendung von Nebenbedingungen gefunden werden können.

Wie in FIACCO & MCCORMICK [16] beschrieben, sucht man also ein x^*, welches folgendes Problem minimiert:

$$\begin{aligned} \min f(x): \quad & \mathbb{R}^n \to \mathbb{R} \\ \text{s.t.} \quad & g_i(x) \geq 0, \quad i = 1, 2, 3, \ldots, m \end{aligned} \tag{B}$$

Die verwendeten Ersatzfunktionen haben etwa folgende Gestalt:

$$\varphi(x, \lambda(t)) = f(x) + \sum_{i=1}^{m} \lambda_i(t) G(g_i(x))$$

Dabei ist t ein Parameter, $\lambda_i(t)$ sind positive Gewichtsfaktoren und $G(y)$ ist eine monotone Funktion in $y \in \mathbb{R}$, welche sich bei $y = 0$ geeignet verhält, etwa:

$$G(y) > 0 \quad \text{falls} \quad y < 0 \qquad \text{und} \qquad G(y) = 0 \quad \text{für} \quad y \geq 0 \tag{4.1}$$

(siehe Abschnitt 4.2.2) oder

$$G(y) \to \infty \quad \text{für} \quad y \downarrow 0 \qquad \text{und} \qquad G(y) = \infty \quad \text{für} \quad y \leq 0 \tag{4.2}$$

(siehe Abschnitt 4.2.1).

Indem man nun t schrittweise vergrößert, nähern sich die Ersatzprobleme P_i immer mehr dem ursprünglichen Problem an. Im Fall (4.1) sind dabei die Gewichtsfunktionen $\lambda_i(t)$ streng monoton wachsend und die $\lambda_i(t)G\left(g_i(x)\right)$ werden *Strafterme* genannt, während bei (4.2) die $\lambda_i(t)$ streng monoton fallend sind und die entsprechenden Korrekturterme *Barriereterme* genannt werden.

Bei den P_i handelt es sich nun jeweils um Optimierungsprobleme *ohne* Nebenbedingungen und solche lassen sich leichter behandeln, da man die Nebenbedingungen ja nicht mehr berücksichtigen muß. Eigentlich müßte man eine ganze Folge von Problemen lösen, aber für eine genügend genaue Annäherung an die Lösung sind in der Praxis nur wenige Probleme P_i zu lösen. Weiters reicht es aus, die P_i nur approximativ und nicht exakt zu lösen, da man an dieser Lösung eigentlich gar nicht interessiert ist. Im Laufe des Programms sollte man aber auch die Zwischenlösungen immer genauer bestimmen, um ein gutes Ergebnis zu erhalten. Zusätzlich ist es möglich, Information, die bei der Lösung von P_k gewonnen wurde, gewinnbringend bei der Lösung von P_{k+1} zu verwenden (Bertsekas [7]). Außerdem wird man vermutlich „bald“ in einem vielversprechenden Teil des Definitionsbereichs landen, von wo auch lokale Optimierung zum Optimum führt.

Stellt ein Programm Anforderungen an die Zielfunktion, wie etwa Stetigkeit oder Differenzierbarkeit, so muß man selbstverständlich darauf achten, daß auch die Ersatzfunktionen diesen Anforderungen gerecht werden.

4.2.1 Innere-Punkte-Verfahren

Definition 4.1 *Ein Punkt $x \in \mathbb{R}^n$ heißt* **zulässig***, wenn er im Definitionsbereich liegt, also alle Nebenbedingungen erfüllt.*

An dieser Stelle werde ich einige Bezeichnungen einführen:

Bemerkung 4.2 *Sei M eine Menge, dann ist M° das* **Innere**, *$\overline{M}$ der* **Abschluß** *und ∂M der* **Rand** *von M.*

Definition 4.3 *Eine Menge M heißt* **robust**, *wenn $M = \overline{M^\circ}$ gilt.*

Bei einem Innere-Punkte-Verfahren wählt man die Barriereterme so, daß der Algorithmus den zulässigen Bereich nicht verlassen kann, indem man am Rand der Nebenbedingungen „Barrieren" errichtet (daher der Name). Dazu setzt man voraus, daß der Definitionsbereich robust ist und $D^\circ = \{x|\, g_i(x) > 0,\, i = 1, \ldots, m\}$ gilt, x also keine Gleichungen als Nebenbedingungen zu erfüllen braucht.
Sei die Funktion

$$S_I : D^\circ \to \mathbb{R}$$

stetig auf D° und erfülle folgende Eigenschaft:

$$g_i\left(\lim_{k\to\infty}(x_k)\right) = 0 \quad \Rightarrow \quad \lim_{k\to\infty} S_I(x_k) = +\infty$$

Sei weiters

$$s(r) : \mathbb{R}^+ \to \mathbb{R}$$

eine stetige, streng monoton wachsende Funktion, die an der Stelle 0 stetig ergänzt werden kann, indem man $s(0) = 0$ setzt.
Mögliche Wahlen für die Ersatzfunktionen sind zum Beispiel:
Für $S_I :\ x \mapsto -\sum\limits_{i=1}^{m} \ln(g_i(x))$ und $s(r) = r$ ergibt sich:

$$U(x, r) = f(x) - r \sum_{i=1}^{m} \ln\left(g_i(x)\right) \tag{4.3}$$

oder für $S_I : \mapsto \sum\limits_{i=1}^{m} 1/g(x)$ und $s(r) = r$:

$$U(x, r) = f(x) + r \sum_{i=1}^{m} \frac{1}{g_i(x)} \tag{4.4}$$

Beide Beispiele erfüllen alle Anforderungen, wie man leicht erkennen kann.

Definition 4.4 *Ein* **Innere-Punkte-Verfahren** *zur Lösung von unbeschränkten Optimierungsproblemen ist eine Methode, welche folgendermaßen vorgeht:*

Schritt 1: *Zuerst wird eine Funktion* $U(x, r_0) := f(x) + s(r_0)S_I(x)$ *definiert, wobei* $s(r)$ *und* $S_I(x)$ *wie oben sind. Dabei ist* r_0 *eine positive, reelle Zahl. Weiters sei ein* $x_0 \in D^\circ$ *als Startpunkt verfügbar;* $k = 0$.

Schritt 2: *Von* x_k $(k \in \mathbb{N})$ *ausgehend wird ein Minimalpunkt* $x_{k+1} = x(r_k)$ *von* $U(x, r_k)$ *in* D *berechnet.*

Schritt 3: *Nun wird ein* r_{k+1} *mit* $r_k > r_{k+1} > 0$ *so bestimmt, daß* $\lim_{k\to\infty} r_k = 0$ *ist und mit* $k = k + 1$ *bei Schritt 2 fortgesetzt.*

Definition 4.5 *Eine Menge* $N \subset M$ *heißt* **isolierte** *Menge (von* M*), wenn* $\overline{N} \cap \overline{M \setminus N} = \emptyset$ *gilt.*

Folgender Satz zeigt, daß Innere-Punkte-Verfahren tatsächlich, unter gewissen Umständen, zu (lokalen) Minima konvergieren:

Satz 4.6 *Seien die Funktionen* $f, g_1, \ldots, g_m$ *stetig und* $U(x, r) = f(x) + s(r)S_I(x)$ *wie in Definition 4.4. Sei* $A^* := \{x |\ f(x) = v^*\ (v^* \text{ist lokales Minimum})\}$ *eine isolierte, kompakte Menge mit* $A^* \cap D^\circ \neq \emptyset$ *(daraus folgt* $A^* \neq \emptyset$*) und sei weiters* r_k *eine streng monoton fallende Nullfolge, dann gilt:*

(i) Es gibt eine kompakte Menge S *mit* $A^* \subset S^\circ$ *und für jeden Punkt* $y \in D \cap S$ *gilt:* $y \notin A^* \Rightarrow f(y) > v^*$. *Für kleine* r_k *existieren Minima* x_k *von* $U(x, r_k)$ *in* $D^\circ \cap S^\circ$ *und jeder Grenzwert einer Teilfolge von* (x_k) *von Minimalpunkten liegt in* A^*.

(ii) $\lim_{k\to\infty} s(r_k)S_I\left(x(r_k)\right) = 0$

(iii) $\lim_{k\to\infty} f\left(x(r_k)\right) = v^*$

(iv) $\lim_{k\to\infty} U(x(r_k), r_k) = v^*$

(v) *Die Folge* $(f(x(r_k)))$ *ist monoton fallend.*

(vi) *Die Folge* $(S_I(x(r_k)))$ *ist monoton wachsend.*

Der Beweis läßt sich in FIACCO et. al [16] nachlesen.

Ein Vorteil dieser Methode liegt darin, daß man den zulässigen Bereich nie verläßt. Das ist besonders dann von Bedeutung, wenn die Zielfunktion außerhalb des Definitionsbereiches gar nicht definiert ist. Weiters verändern die Barriereterme (Gleichungen (4.3) und (4.4)) nichts an der Differenzierbarkeit der ursprünglichen Funktion, wie man durch Anwendung der Quotienten- oder Kettenregel leicht einsehen kann, da sowohl $\ln(x)$, als auch $1/x$ auf $\mathbb{R}^+$ aus der Klasse C^∞ sind.

Es gibt allerdings auch Nachteile: Erstens ist es oft nicht einfach, überhaupt einen zulässigen Punkt zu finden, zweitens braucht das Verfahren nicht unbedingt gegen ein *globales* Minimum von $f(x)$ zu konvergieren, und drittens werden die Strafterme von allen Nebenbedingungen *immer* berücksichtigt, also auch dann, wenn sie gar nicht verletzt werden.

Für die Implementation empfiehlt es sich, die r_k nicht zu rasch abnehmen zu lassen, da das zu schlechter Konditionierung des Problems führt.

4.2.2 Äußere-Punkte-Verfahren

Bei einem Äußere-Punkte-Verfahren wählt man die Strafterme so, daß die Funktion auf D unverändert bleibt, Verletzungen der Nebenbedingungen aber bestraft werden, und zwar umso stärker, je größer die Verletzung ausfällt. Dadurch versucht man zu verhindern, daß das Programm Punkte behandelt, die weit vom zulässigen Bereich liegen.

Gleichungs-Nebenbedingungen führen zu Straftermen wie etwa:

$$S_O(x) = \sum_{i=1}^{m} \frac{1}{2} g_i(x)^2 \tag{4.5}$$

$$S_O(x) = \cosh\left(g_i(x)\right) - 1 \tag{4.6}$$

$$S_O(x) = \rho^{-1} \sum_{i=1}^{m} |g_i(x)|^\rho, \ \rho > 1 \tag{4.7}$$

$$S_O(x) = \rho^{-1} \sum_{i=1}^{m} |g_i(x)|^\rho + \frac{1}{2} g_i(x)^2, \ \rho > 1 \tag{4.8}$$

und Ungleichungs-Nebenbedingungen lassen sich zum Beispiel mit

$$S_O(x) = \sum_{i=1}^{m} \frac{1}{2} \left(\min\{0,\ g_i(x)\}\right)^2 \tag{4.9}$$

berücksichtigen.

Alle diese Strafterme erfüllen folgende Eigenschaften (Bertsekas [7]):

1. $S_O(x)$ ist stetig differenzierbar und streng konvex auf $\mathbb{R}$.
2. $S_O(x) = 0$ und $S'_O(x) = 0$ für $g_i(x) = 0$
3. $\lim S'_O(x) = \infty$ für $g_i(x) \to \infty$

Sei allgemein $S_O : \mathbb{R} \to \mathbb{R}$ eine stetige Funktion, welche folgende Eigenschaft erfüllt:

$$S_O(x) = 0 \quad \text{für} \quad g_i \geq 0,\ i = 1, \ldots, m \quad \text{und} \quad S_O(x) > 0 \quad \text{sonst}$$

Sei weiters $p(t)$ eine stetige, reellwertige Funktion mit den Eigenschaften:

- $p(t)$ ist streng monoton wachsend auf $\mathbb{R}^+$
- Ist (t_k) eine monoton wachsende Folge positiver Zahlen mit folgender Eigenschaft:

$$\lim_{k\to\infty} t_k = +\infty \quad \Rightarrow \quad \lim_{k\to\infty} p(t_k) = +\infty$$

Definition 4.7 *Ein* **Äußere-Punkte-Verfahren** *zur Lösung von unbeschränkten Optimierungsproblemen ist ein Verfahren, welches folgendermaßen vorgeht:*

Schritt 1: *Zuerst wird eine Funktion $T(x,t) := f(x) + p(t)S_O(x)$ definiert und ein $t_0 > 0$ bestimmt. Dabei seien $p(t)$ und $S_O(x)$ wie oben definiert. Weiters wird ein (lokales) Minimum $x_0 = x(r_0)$ von $T(x,t_0)$ berechnet; $k = 0$.*

Schritt 2: *Es wird ein $t_{k+1} > t_k$ bestimmt und mit x_k als Startwert ein Minimalpunkt x_{k+1} von $T(x,t_{k+1})$ berechnet.*

Schritt 3: *Nun wird mit $k = k+1$ bei Schritt 2 forgesetzt.*

Dabei ist darauf zu achten, daß $\lim\limits_{k\to\infty} t_k = +\infty$ gilt.

Analog zu Satz 4.6 gibt es auch für Äußere-Punkte-Verfahren einen Satz, der die Konvergenz zu lokalen Minima behandelt:

Satz 4.8 *Seien die Funktionen $f, g_1, \ldots, g_m$ stetig und $T(x,t) = f(x)+p(t)S_O(t)$ wie in Definition 4.7. Sei $A^* := \{x \,|\, f(x) = v^* \ (v^* \text{ist lokales Minimum})\}$ eine nichtleere, isolierte, kompakte Menge und sei weiters (t_k) eine unbeschränkte, streng monoton wachsende Folge reeller Zahlen, dann gilt:*

(i) *Es gibt eine kompakte Menge S mit $A^* \subset S^\circ$ und für jeden Punkt $y \in D \cap S$ gilt: $y \notin A^* \Rightarrow f(y) > v^*$. Für ausreichend große t_k existieren Minima von $T(x,t_k)$ in S° und jeder Grenzwert einer Teilfolge von (x_k) solcher Minimalpunkte liegt in A^*.*

(ii) $\lim\limits_{k\to\infty} p(t_k)S_O(x(t_k)) = 0$

(iii) $\lim\limits_{k\to\infty} f(x(t_k)) = v^*$

(iv) $\lim\limits_{k\to\infty} T(x(t_k),t_k) = v^*, \quad T[x(t_k),t_k] \le v^*, \quad \text{für alle} \quad k$

(v) *Die Folge $(f(x(t_k)))$ ist monoton wachsend.*

(vi) *Die Folge $(S_O(x(t_k)))$ ist monoton fallend.*

Der Beweis läßt sich wieder in Fiacco et. al [16] nachlesen.

Als Vorteile von Äußere-Punkte-Verfahren sind zu erwähnen, daß man keinen zulässigen Punkt benötigt um sie anwenden zu können, daß für den Definitionsbereich auch nicht-robuste Mengen möglich sind und daß ein Strafterm nur berücksichtigt werden muß, wenn die zugrunde liegende Nebenbedigung verletzt ist. Ein weiterer Vorteil gegenüber Innere-Punkte-Verfahren ist weiters, daß wenn ein globales Minimum einer Ersatzfunktion in D liegt, es bereits globales Minimum der ursprünglichen Funktion $f(x)$ ist.

Ein Nachteil ist es, daß die Strafterme (4.5)–(4.8) nur für Gleichungs-Nebenbedingungen verwendbar sind, und die Strafterme (4.7)–(4.9) für $\rho \neq 2k,\ k \in \mathbb{N}$ nur endlich oft differenzierbare Ersatzfunktionen liefern. Auch ist es möglich, daß $f(x)$ stärker fällt, als die Strafterme wachsen und daher jede Ersatzfunktion der Folge nach unten unbeschränkt ist.

4.2.3 Mischverfahren

Nun ist es selbstverständlich möglich, beide Verfahren zu kombinieren.
Am sinnvollsten scheint es mir zu sein, wenn man für die Ungleichungs-Nebenbedingungen, die am Startpunkt strikt erfüllt sind, Barriereterme wählt, wie sie für Innere-Punkte-Verfahren verwendet werden und die Strafterme der Gleichungs-Nebenbedingungen den Äußere-Punkte-Verfahren entnimmt, denn dadurch bleibt die Differenzierbarkeit erhalten (sofern man sich für (4.5) oder (4.6) als Strafterme bei Gleichungs-Nebenbedingungen entscheidet). Man will also folgendes Problem lösen:

$$\begin{aligned} \min f(x):\quad & \mathbb{R}^n \to \mathbb{R} \\ \text{s.t.}\quad & g_i(x) \geq 0, \quad i = 1, \ldots, m_1 \\ & g_i(x) = 0, \quad i = m_1 + 1, \ldots, m \end{aligned} \tag{C}$$

Dazu wählt man positive Folgen (r_k^i), $i = 1, \ldots, m_1$ und (t_k^i), $i = m_1 + 1, \ldots, m$

und definiert die Ersatzfunktion

$$V(x, r_k^i, t_k^i) := f(x) + \sum_{i=1}^{m_1} r_k^i S_{I\,i}(x) + \sum_{i=m_1+1}^{m} t_k^i S_{O\,i}(x)$$

Die r_k^i und t_k^i entsprechen den r_k und t_k aus den Abschnitten 4.2.1 und 4.2.2, nur statt einem einzelnen Wert für alle Nebenbedingungen einer Klasse, bekommt nun jede Nebenbedingung ihren eigenen Gewichtsfaktor. Dadurch ist es unter Umständen möglich, den Algorithmus zu beschleunigen, indem man die Gewichtsfaktoren geeignet wählt. Details dazu findet man in Fiacco et. al [16], S.114ff.

4.3 Nicht unterstütze Nebenbedingungen

Hat man ein Problem mit einer Nebenbedingung, welche von einem Programm nicht unterstützt wird, dann kann man sich stets dadurch helfen, daß man statt der ursprünglichen Zielfunktion eine abgeänderte verwendet, welche die Nebenbedingungen berücksichtigt. Dies kann etwa durch Straf- oder Barriereterme geschehen, welche bei Verletzung einer Nebenbedingung hinzugefügt werden, wie in den Abschnitten 4.2.1 und 4.2.2 beschrieben. Der Unterschied zu den vorhergehenden Abschnitten liegt darin, daß man dem Programm nicht die Zielfunktion und die Nebenbedingungen übergibt, sondern zuerst die Ersatzfunktion bildet und diese dann dem Programm überläßt. Wie zuvor wird die Zielfunktion so verändert, daß sie sich im Inneren des zulässigen Bereiches D° nur geringfügig bis gar nicht ändert und höchstens am Rand, jedenfalls aber dort, wo die Nebenbedingungen verletzt sind, die Funktionswerte stark vergrößern. Auch hier muß man gegebenfalls darauf achten, daß die modifizierte Zielfunktion die vom Programm geforderten Eigenschaften (Stetigkeit, Differenzierbarkeit, ...) beibehält. Beim anschließenden Optimieren hofft man dann, daß das gefundene Optimum die Nebenbedingungen erfüllt und dem gesuchten Optimum (der unmodifizierten Zielfunktion) entspricht oder zumindest in dessen Nähe liegt.

In den folgenden Abschnitten 4.3.1–4.3.3 soll folgendes Optimierungsproblem behandelt werden:

$$\begin{aligned} \min f(x): \quad & \mathbb{R}^n \to \mathbb{R} \\ \text{s.t.} \quad & x \in D\ (\subset \mathbb{R}^n), \quad D \text{ beschränkt} \end{aligned} \tag{D}$$

Gewisse Klassen dieses Problems sollen für Programme, welche nur komponentenweise Beschränkungen oder gar keine Nebenbedingungen zulassen, zugänglich gemacht werden (PINTER [34], S.120ff). Dazu bettet man D in ein n-dimensionales Intervall $I = [a, b] \subseteq \mathbb{R}^n$ $(D \subseteq I)$ ein und erweitert f auf geeignete Weise auf I. Oft sind geeignete Vektoren a und b $(a_i \leq x_i \leq b_i,\ i = 1, \ldots, n)$ anfänglich bereits bekannt. Es empfiehlt sich stets, I möglichst klein zu halten.

4.3.1 Konvexer Definitionsbereich

Definition 4.9 *Eine Funktion f heißt* **Lipschitz-stetig** *auf $X \subset \mathbb{R}^n$, wenn es eine Konstante K gibt, sodaß gilt:*

$$\|f(x) - f(y)\| \leq K\|x - y\| \quad \textit{für alle} \quad x,\ y \in X$$

K heißt dann **Lipschitz-Konstante** *von f auf D.*

In diesem Abschnitt soll die Menge D robust (siehe Definition 4.3) und konvex sein, und f sei Lipschitz-stetig auf D. Weiters wird ein innerer Punkt x_c von D und ein (Über)schätzer der Lipschitz-Konstanten L der Funktion f auf D benötigt. Die Existenz eines solchen x_c ist durch die Robustheit von D gewährleistet.

Sei $x_h(x)$ jener Punkt auf der Strecke $\overline{x_c\, x}$, welcher von x_c den größten Abstand besitzt, wobei gilt:

$$x_h := \quad x_h(x) = \quad \lambda_h x + (1 - \lambda_h) x_c \tag{4.10}$$

$$\lambda_h := \quad \lambda_h(x) = \quad \max\{0 \leq \lambda \leq 1 :\ \lambda x + (1 - \lambda) x_c \in D\} \tag{4.11}$$

Auf Grund der Eindeutigkeit von λ_h ist auch x_h eindeutig. Die Existenz des Maximums in (4.11) ist dadurch gewährleistet, daß D robust und somit abgeschlossen ist.

Der Punkt x_h könnte man auch als Projektion (mit Projektionszentrum x_c) auf den Rand von D definieren. Auch so ist x_h sicher eindeutig, da D konvex und $x_c \in D^\circ$ ist.
In jedem Fall gilt:

$$x_h = x, \quad \text{für} \quad x \in D \qquad \text{und} \qquad x \in \partial D, \quad \text{falls} \quad x \notin D$$

da D abgeschlossen ist.
Nun definieren wir die Ersatzfunktion:

$$f_e(x) := f(x_h) + L \, \|x - x_h\| \quad \text{für} \quad x \in I \tag{4.12}$$

Dabei ist I eine Obermenge von D, welche wir wieder als (uneigentliches) n-dimensionales Intervall annehmen wollen. Das führt zu dem Problem:

$$\begin{aligned} \min f(x) : \quad & \mathbb{R}^n \to \mathbb{R} \\ \text{s.t.} \qquad & x \in I \end{aligned} \tag{E}$$

Es gilt:

$$f_e(x) = f(x) \quad \text{für} \quad x \in D \qquad \text{und} \qquad f_e(x) \geq f_e(x_h) \quad \text{für} \quad x \notin D$$

und daher sind die globalen Optimierungsprobleme (D) und (E) äquivalent.

Einerseits hängt die Funktion $f_e(x)$ wegen den Gleichungen (4.10) und (4.11) auch vom gewählten Punkt x_c ab, ohne daß dies hervorgehoben wäre. Der Punkt x_c muß aber fix gewählt sein.

Andererseits ist $f_e(x)$ vom gewählten Intervall I unabhängig. Daher könnte man auf die Bedingung $x \in I$ verzichten und erhielte ein unbeschränktes Problem. Das ist aber nur dann von Bedeutung, wenn diese Methode bei Programmen eingesetzt wird, welche keine Schranken als Nebenbedingungen erlauben.

Die Funktion f_e aus Gleichung (4.12) läßt sich noch verallgemeinern:

$$f_e(x) = f(x_h) + \varphi(|x - x_h|) \quad \text{für} \quad x \in I$$

Dabei ist der Betrag wie üblich komponentenweise zu verstehen und $\varphi(z)$ ist eine Lipschitz-stetige, streng monoton wachsende reellwertige Funktion mit $\varphi(0) = 0$.

Definition 4.10 *Der* **Durchmesser** *einer Menge M ist definiert als:*

$$\Delta(M) = \sup_{x,\, y \in M} \|x - y\|$$

und die **Distanz** *eines Punktes x zu einer Menge M ist definiert als:*

$$d(x,\, M) = \min_{z \in M} \|x - z\|$$

Das folgende Lemma ist eine Verallgemeinerung des Lemmas aus PINTER [34], S. 122. Da mir der ursprüngliche Beweis fragwürdig erscheint, werde ich einen eigenen angeben.

Lemma 4.11 *Der Operator x_h ist Lipschitz-stetig. Es gilt:*

$$\|x_h(x_1) - x_h(x_2)\| \leq C\, \|x_1 - x_2\| \qquad \textit{für alle} \qquad x_1, x_2 \in I$$

mit

$$C = \frac{\Delta(D)}{d(x_c, \partial D)}$$

Beweis. Es ist klar, daß das Lemma für den Fall $x_1 = x_2$ jedenfalls erfüllt ist; sei daher $x_1 \neq x_2$.

Wegen der Beschränktheit von D und der Konstruktion von x_h gilt:

$$\|x_h(x_1) - x_h(x_2)\| \quad \leq \quad \Delta(D) \quad < \quad \infty, \qquad \text{für alle} \qquad x_1,\, x_2 \in I$$

Für alle x_1 und x_2 mit $\|x_1 - x_2\| \geq d(x_c, \partial D)$ gilt somit folgende Ungleichungskette:

$$\|x_h(x_1) - x_h(x_2)\| \quad \leq \quad \Delta(D)\, \frac{\|x_1 - x_2\|}{\|x_1 - x_2\|} \quad \leq \quad \Delta(D)\, \frac{\|x_1 - x_2\|}{d(x_c, \partial D)}$$

und das Lemma ist für diesen Fall bewiesen.

Es bleibt noch zu zeigen, daß das Lemma auch für x_1 und x_2 mit

$$\|x_1 - x_2\| \quad < \quad d(x_c, \partial D) \tag{4.13}$$

gilt. Wir setzen dazu $z = x_h(x_1)$ und definieren eine Parallelprojektion

$$p_{\|} : \quad I \rightarrow \partial D$$

welche einen Punkt x parallel zur Strecke $\overline{x_c z}$ auf den Rand von D projiziert. Vergleicht man $p_{\|}$ mit dem Operator x_h, so erhält man folgende Abschätzung:

$$\|x_h(x_1) - x_h(x_2)\| \quad \leq \quad \|p_{\|}(x_1) - p_{\|}(x_2)\| \tag{4.14}$$

Wegen (4.13) ist $p_{\|}$ immer definiert, da die Projektion den Punkt x_1 direkt nach der Richtung von x_c abbildet. Deshalb muß auch die zu $\overline{x_c z}$ parallele Gerade, welche durch x_2 geht und einem Abstand zu x_c hat, der laut Voraussetzung kleiner als $d(x_c, \partial D)$ ist, jedenfalls den Rand von D schneiden.

Bei der Parallelprojektion kommt es nicht auf die Distanz der Punkte x_1 und x_2 von x_c an, solange sie auf den Geraden liegen, entlang derer sie projiziert werden, und ihre relative Position zueinander beibehalten. Daher kann man ohne Beschränkung der Allgemeinheit annehmen, daß für den Punkt x_1 gilt:

$$x_1 \;=\; z \in \partial D$$

Wegen der Konvexität von D ist die Norm $\|p_{\|}(x_1) - p_{\|}(x_2)\|$ dann maximal, wenn $\overline{x_c z}$ orthogonal auf $\overline{z x_2}$ steht, weshalb dies ab nun vorausgesetzt wird.

Ich werde nun zeigen, daß

$$\|p_{\|}(x_1) - p_{\|}(x_2)\| \quad \leq \quad C\,\|x_1 - x_2\| \tag{4.15}$$

erfüllt ist.

Sei $X \in \partial D$, dann ist der Abstand von X zu x_c jedenfalls mindestens gleich $d(x_c, \partial D)$. Das rechtwinklige Dreieck $x_c X z$ mit rechtem Winkel bei X in Abbildung 4.1 ist so gewählt, daß der ungünstigste Fall noch übertroffen wird, da

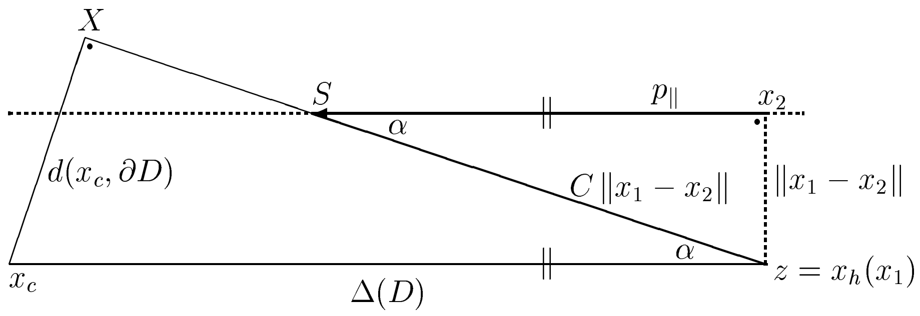

Abbildung 4.1: Skizze zu Beweis von Lemma 4.11

$\|x_c - X\| = d(x_c, \partial D)$ gesetzt ist und immer gilt:

$$\|x_c - z\| \quad \leq \quad \Delta(D) - d(x_c, \partial D)$$

Der Punkt S ist der Schnittpunkt zwischen der Strecke $\overline{Xz}$ mit der Parallelen zu $\overline{x_c z}$ durch x_2.

Es ist nicht möglich, daß das Dreieck mit den Eckpunkten x_2, z und $p_{||}(x_2)$ bei dem Punkt $p_{||}(x_2)$ einen kleineren Winkel hat als der Winkel α bei dem Punkt z des Dreiecks $x_c X z$. Denn gäbe es einen Punkt x_2 mit

$$\|p_{||}(x_1) - p_{||}(x_2)\| \quad > \quad C\,\|x_1 - x_2\|$$

dann müßte $p_{||}(x_2)$ von x_2 aus hinter dem Punkt S liegen. Das ist aber nicht möglich, da D konvex und $d(x_c, \partial D)$ der Minimalabstand von x_c zum Rand von D ist. Somit ist Gleichung (4.15) bewiesen.

Kombiniert man nun die Gleichungen (4.14) und (4.15), so erhält man:

$$\|x_h(x_1) - x_h(x_2)\| \quad \leq \quad \frac{\Delta(D)}{d(x_c, \partial D)}\,\|x_1 - x_2\|$$

und somit ist die Behauptung bewiesen. ■

Satz 4.12 (Pinter [34]) *Sei die Funktion f Lipschitz-stetig auf D mit einer (überschätzten) Lipschitz-Konstanten L. Seien f_e und C so definiert wie in Lemma 4.11, dann ist f_e Lipschitz-stetig auf dem n-dimensionalen Intervall I mit einer Lipschitz-Konstanten $L_e = L(2C + 1)$.*

Beweis. Seien x_1 und x_2 zwei beliebige Punkte aus I, dann kann man wegen Gleichung (4.12), S. 24

$$f_e(x_1) - f_e(x_2) \;=\; f(x_{1_h}) + L\|x_1 - x_{1_h}\| - f(x_{2_h}) - L\|x_2 - x_{2_h}\|$$

schreiben, und daher gilt:

$$|f_e(x_1) - f_e(x_2)| \;\leq\; |f(x_{1_h}) - f(x_{2_h})| + L|\,\|x_1 - x_{1_h}\| - \|x_2 - x_{2_h}\|\,|$$

Wenn man nun Lemma 4.11 anwendet, so erhält man:

$$|f(x_{1_h}) - f(x_{2_h})| \quad\leq\quad L\|x_{1_h} - x_{2_h}\| \quad\leq\quad LC\|x_1 - x_2\|$$

und

$$\begin{aligned} |\,\|x_1 - x_{1_h}\| - \|x_2 - x_{2_h}\|\,| \;&\leq\; \|x_1 - x_{1_h} - x_2 + x_{2_h}\| \\ \leq\; \|x_1 - x_2\| + \|x_{1_h} - x_{2_h}\| \;&\leq\; (1+C)\|x_1 - x_2\| \end{aligned}$$

und daher:

$$|f_e(x_1) - f_e(x_2)| \quad\leq\quad L(C + 1 + C)\|x_1 - x_2\|$$

Daraus folgt die Behauptung. ∎

Um diese Methode umsetzen zu können, ist es noch nötig, die λ_h zu bestimmen. Dafür lassen sich Bisektions-Algorithmen einsetzen. Es ist aber auch möglich, spezielle Strukturen, wie etwa Linearität, auszunutzen (Pinter [34], S.125ff).

Bemerkung 4.13 *Ist es möglich, den Definitionsbereich D in $n \in \mathbb{N}$ (nicht unbedingt disjunkte) konvexe Teilmengen D_i aufzuteilen, dann kann man diese Methode immer noch anwenden.*

Dazu bestimmt man nun jeweils das globale Minimum f_i^* in jedem der einzelnen D_i. Für das globale Minimum f^* des Problems mit unzerlegtem Definitionsbereich gilt dann:

$$f^* = \min_{1 \leq i \leq n} \{f_i^*\}$$

4.3.2 Sternförmiger Definitionsbereich

In PINTER [34] ist auch eine Verallgemeinerung von konvexen (Abschnitt 4.3.1) auf sternförmige Definitionsbereiche angegeben. Die dort angegebene Definition weicht aber von der üblichen ab (etwa in HEUSER [18], S.245) und schließt Robustheit von vornherein ein.
Sei $S^n(x_c, r)$ die n-dimensionale Kugel mit Mittelpunkt x_c und Radius r:

$$S^n(x_c, r) := \{z \mid \|x_c - z\| \leq r \}$$

Wie in Abschnitt 4.3.1 sei $x_h(x)$ definiert als:

$$x_h(x) := \lambda_h x + (1 - \lambda_h)x_c$$

mit

$$\lambda_h := \lambda_h(x) = \max \{0 \leq \lambda \leq 1 : \lambda x + (1 - \lambda)x_c \in D\}.$$

Definition 4.14 *Eine* **Sterntransformation** *von $S^n(x_c, r)$ ist durch eine stetige Funktion x_h gegeben, welche die Strecke*

$$\{x_c + \alpha x_S \mid 0 \leq \alpha \leq 1\} \tag{4.16}$$

auf die Strecke

$$\{x_c + \alpha x_h(x_S) \mid 0 \leq \alpha \leq 1\} \tag{4.17}$$

auf folgende Weise abbildet:

$$x_c + \alpha x_S \quad \mapsto \quad x_c + \alpha x_h(x_S)$$

Dabei ist x_S ein Randpunkt von $S^n(x_c, r)$.

Die Definition von x_h bedingt, daß die Trägergeraden der Strecken (4.16) und (4.17) übereinstimmen.

Definition 4.15 *Eine Menge M heißt* **sternförmig**, *wenn sie durch eine Sterntransformation entstanden ist. Der Punkt x_c heißt* **Zentrum** *von M.*

Meist gibt es viele Punkte, die als Zentrum einer sternförmigen Menge in Frage kommen.

Da bei einem Problem mit sternförmigem Definitionsbereich D jeder von dessen Zentrum $x_c \in D^\circ$ ausgehende Strahl genau einen Schnittpunkt mit ∂D hat, lassen sich die Ergebnisse von Abschnitt 4.3.1 auch auf sternförmiges D übertragen. Die Eigenschaft $x_c \in D^\circ$ ist deshalb gesichert, da auf Grund der Stetigkeit einer Sterntransformation das Zentrum einer sternförmigen Menge M immer in deren Inneren M° liegt.

Insbesondere ist es wieder oft möglich, ein Problem P mit nicht-sternförmigem Definitonsbereich D in mehrere kleinere Probleme P_i mit jeweils sternförmigem Defintionsbereich D_i aufzuspalten. Selbstverständlich werden die Anwendungsmöglichkeiten dieser Methode damit wesentlich erweitert.

Wie im vorangehenden Abschnitt kann man in einfachen Fällen die Projektionspunke anlytisch finden oder mittels Bisektion annähern.

4.3.3 Ein n-dimensionales Intervall als Definitionsbereich

In diesem Abschnitt setzte ich voraus, daß die Funktion f Lipschitz-stetig auf einem n-dimensionalen Intervall $D = [a, b]$ ist. Nun soll Problem (E), S. 24, auch für Programme zugänglich gemacht werden, welche überhaupt keine Nebenbedingungen zulassen.

Sei $x_h(x)$ jener Punkt aus D, welcher von x minimalen Abstand besitzt:

$$x_h(x) \;:=\; \begin{pmatrix} \mathrm{proj}(x_1,\, [a_1, b_1]) \\ \vdots \\ \mathrm{proj}(x_n,\, [a_n, b_n]) \end{pmatrix} \tag{4.18}$$

wobei gilt:

$$\mathrm{proj}(\xi,\, [\alpha,\, \beta]) := \begin{cases} \alpha, & \text{falls} \quad \xi < \alpha \\ \xi, & \text{falls} \quad \alpha \le \xi \le \beta \\ \beta, & \text{falls} \quad \beta < \xi \end{cases} \tag{4.19}$$

Wie in Abschnitt 4.3.1 wird auch hier eine Ersatzfunktion f_e definiert:

$$f_e(x) := f(x_h) + \varphi(|x - x_h(x)|), \qquad x \in \mathbb{R}^n \tag{4.20}$$

Der Betrag ist in dieser Gleichung wie üblich komponentenweise zu verstehen und $\varphi(z)$ ist eine Lipschitz-stetige, streng monoton wachsende Funktion mit $\varphi(0) = 0$.

Lemma 4.16 *Der Operator $x_h(x)$ ist Lipschitz-stetig auf $\mathbb{R}^n$ und es gilt:*

$$\|x_h(x) - x_h(y)\| \;\leq\; K\,\|x - y\|$$

Beweis. Da auf $\mathbb{R}^n$ alle Normen äquivalent sind, reicht es also zu zeigen:

$$\|x_h(x) - x_h(y)\|_\infty \;\leq\; K\,\|x - y\|_\infty \tag{4.21}$$

für alle $x, y \in \mathbb{R}^n$ (siehe Definition 4.9).

Es gilt für jede einzelne Komponente, wie man an Hand der Definition von $\text{proj}(\xi,\, [\alpha,\, \beta])$ in Gleichung (4.19) leicht einsehen kann, folgende Bedingung:

$$|x_{h_i}(x) - x_{h_i}(y)| \;\leq\; |x_i - y_i|$$

und daher auch:

$$\max_i |x_{h_i}(x) - x_{h_i}(y)| \;\leq\; \max_i |x_i - y_i|$$

Somit gilt Gleichung (4.21) und das Lemma ist bewiesen. ∎

Satz 4.17 *Die in den Gleichungen* (4.20) *bis* (4.19) *definierte Funktion $f_e(x)$ ist Lipschitz-stetig auf $\mathbb{R}$.*

Beweis. Als Summe von zwei Lipschitz-stetigen Funktionen ist $f_e(x)$ ebenfalls Lipschitz-stetig, wie man durch einfaches Anwenden der Dreiecks-Ungleichung erkennen kann. ∎

Diese Methode läßt sich auch bei entarteten Intervallen ($a_i = b_i$ für mindestens ein i), also nicht-robustem Definitionsbereich, anwenden.

Aus Gründen der Einfachkeit, läßt sich $\varphi(z)$ in der Praxis als

$$\varphi(z) = \|z\|_1, \quad \varphi(z) = \|z\|_2 \quad \text{oder} \quad \varphi(z) = \|z\|_\infty$$

wählen.

Im Unterschied zu den vorangehenden Abschnitten (4.3.1 und 4.3.2) lassen sich in diesem Fall die Projektionspunkte leicht berechnen. Ein Nachteil besteht aber darin, daß man dabei auf (Hyper-)Quader als Definitionsbereich eingeschränkt bleibt.

Kapitel 5

Lösungverfahren

Ein globales Minimum ist durch zwei Bedingungen charakterisiert. Die eine, daß es sich um ein Minimum handelt, ist eine lokale Bedingung und die, daß es das *globale* Minimum sein soll, ist eine globale Bedingung. Daher ist es wohl sinnvoll, wenn ein Algorithmus, also letzten Endes ein Programm, sowohl globale als auch lokale Suche vereint. Wie das geschieht, ist aber von Algorithmus zu Algorithmus verschieden.

Algorithmen und infolge auch Programme zur globalen Optimierung lassen sich in mehrere Klassen einteilen, wie es in nachfolgender Tabelle 5.1 geschehen ist:

		deterministisch	stochastisch	
			quasi - MC	MC
rigoros	mit Garantie	**B&B**		
	ohne Garantie		**SA**, **Cluster**	**SA**, **Cluster**
heuristisch			**B&PB**	**EA**, **B&PB**

Tabelle 5.1: Klassifikation von Verfahren

B&B steht dabei für Branch und Bound, **B&PB** für Branch und Probability Bound, **EA** für evolutive oder genetische Algorithmen und **SA** für Simulated

Annealing. Manche Varianten dieser Algorithmen können von dieser Einteilung abweichen.

Deterministisch sei dabei ein Verfahren, welches Schritt für Schritt immer genau vorherbestimmt ist, also keine zufällige Komponente enthält.

Stochastisch heißt ein Verfahren dann, wenn es Zufallszahlen oder Vergleichbares verwendet. Dabei wird noch zwischen zwei Spielarten unterschieden:

Monte Carlo-Methoden (MC), also solchen, die problemunabhängige pseudo-Zufallszahlen verwenden und

Quasi-Monte Carlo-Methoden, die problemunabhängige deterministische Zahlenfolgen einsetzen, welche den Defininitionsbereich möglichst gleichmäßig abdecken.

Häufig konvergiert die von einem Algorithmus bestimmte Folge y_n mit Wahrscheinlichkeit 1 gegen das globale Minimum:

$$P\left(\lim_{n\to\infty} y_n = f^*\right) = 1 \tag{5.1}$$

Dabei sei (y_n) die monoton fallende Folge der so bestimmten Punkte: y_n soll der beste von den ersten n berechneten Punkten $x \in D$ sein. Diese Bedingung ist nicht sehr streng, da ja nirgends eine Aussage darüber gemacht wird, wie viele Funktionsauswertungen für eine bestimmte Genauigkeit nötig sind. Sie wird sogar von reiner Zufallssuche, wie etwa vom in Abschnitt 6.2.1 beschriebenen Programm, erfüllt. Der Rechenaufwand für ein bestimmtes Problem kann enorm sein, aber obiges Kriterium liefert im vornhinein keine Information darüber.

Rigoros heißt ein Verfahren dann, wenn es Bedingung (5.1) erfüllt.

Mit Garantie bedeutet, daß das Verfahren sicher ein globales Minimum findet, und

Ohne Garantie bedeutet eben, daß das nicht unbedingt eintreten muß.

Heuristisch sollen jene Methoden heißen, die irgendwelche Heuristiken verwenden, ohne daß Bedingung (5.1) gezeigt werden konnte.

Der Unterschied zwischen einem Monte Carlo-Verfahren und einem Quasi-Monte Carlo-Verfahren besteht eigentlich nur in der Wahl der Zahlenfolge. Daher lassen sie sich ohne größeren Aufwand ineinander umwandeln. Die (Quasi-)Zufallszahlen sind meiner Ansicht nach meist für die globale Suche verantwortlich, indem sie einen gegebenen Wertebereich möglichst gut abdecken. Der Algorithmus sorgt dann für die Einschränkung des Suchbereichs, also für die lokale Suche. Quasi-Zufallszahlen scheinen mir deshalb geeigneter, da sie weder großen „Löcher" im Suchbereich offen lassen, noch andere Bereiche besonders gründlich abdecken, wie dies bei „echten" Zufallszahlen vorkommt. Quasi-Zufallszahlen sollten jedoch nicht dazu verwendet werden, Entscheidungen zu treffen, wie dies etwa bei evolutiven Algorithmen (Abschnitt 5.3) oder Simulated Annealing (Abschnitt 5.4) vorkommt. In diesem Fall kommt es nämlich nicht darauf an, daß ein mehrdimensionaler Bereich möglichst gut von einer Folge von Zufallszahlen abgedeckt wird, sondern daß aufeinanderfolgende Zufallszahlen voneinander möglichst unabhängig sind.

Ab nun werde ich keine Unterscheidung mehr machen und nur mehr von *Zufallszahlen* beziehungsweise *Zufallsfolgen* sprechen.

Der Rest dieses Kapitels wird sich nun mit verschiedenen Methoden zur Lösung von globalen Optimierungsproblemen beschäftigen. Da es in diesem Rahmen nicht möglich ist, alle detailliert zu beschreiben, ist es nötig, gewisse Einschränkungen zu treffen. Ich werde mich daher mit stochastischen Verfahren eingehender beschäftigen als mit anderen.

5.1 Branch und Bound

Ein wesentlicher Vorteil dieser Klasse von deterministischen Methoden ist es, daß es möglich ist, zu wissen, ob man ein Optimum gefunden hat oder nicht. Ein weiterer Vorteil ist es auch, daß es nicht möglich ist, ein globales Optimum zu „übersehen“. Dazu reichen aber lokale Informationen über die Zielfunktion, wie etwa die Kenntnis der Ableitungen an allen Punkten im Definitionsbereich, nicht aus, sondern man benötigt auch globale Informationen, wie etwa Lipschitz-Konstanten.

Die Vorgehensweise bei Branch und Bound (engl.: *verzweigen* und *beschränken*) ist so, daß man bei einem großen Problem den Definitionsbereich D_0 immer wieder in kleinere Bereiche D_i $(i = 1, \ldots, s)$, die ich im folgenden auch *Boxen* nennen werde, aufspaltet und die so entstandenen kleineren Probleme weiter untersucht. Auf diese Weise bekommt man eine immer größere Anzahl von Boxen. Weiß man von einer Box, daß sich in ihr sicher kein globales Minimum befindet, so kann man diese Box „verwerfen“ und braucht sie in Folge nicht mehr zu berücksichtigen.

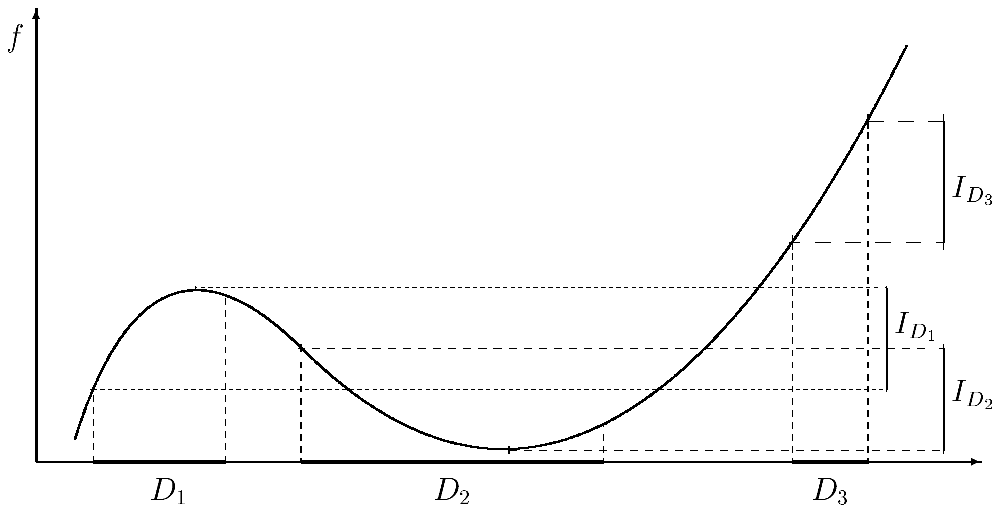

Abbildung 5.1: Verwerfen einer Box

Abbildung 5.1 veranschaulicht, wann es möglich ist, eine Box zu verwerfen.

Dazu schließt man die Funktionswerte des verkleinerten Definitionsbereichs in ein reelles Intervall I_{Box} ein. Liegt nun die Untergrenze eines solchen Intervalls über der Obergrenze eines anderen Intervalls, so kann man erstere Box ruhig wegwerfen, da sie für ein globales Minimum nicht mehr in Frage kommt.

In Abbildung 5.1 kann man sich also nur der Box D_3 entledigen, da die Untergrenze von I_{D_3} sowohl über der Obergrenze von I_{D_1} als auch der von I_{D_2} liegt. Die Box D_1 enthält zwar das globale Optimum nicht, läßt sich aber dennoch nicht verwerfen, da das Kriterium nicht erfüllt ist.

Die Intervalle der Funktionswerte, I_{D_i}, sind meist nicht exakt bekannt und man muß daher oft mit überschätzten Intervallen arbeiten.

Auf Grund dieses Prozesses kann man hoffen und unter bestimmten Voraussetzungen auch beweisen, daß man nicht immer nur teilt und somit immer mehr (exponentiell viele) Boxen bekommt, sondern auch wirklich einmal ein Ende erreicht, da man fast alle Boxen verwerfen kann.

Ein Nachteil besteht darin, daß die Gefahr besteht, immer wieder teilen zu müssen, sodaß die Anzahl der zu behandelnden Teilbereiche exponentiell steigt, wenn man keine oder nur wenige der Boxen wegwerfen kann. Ein weiterer Nachteil ist es wohl auch, daß der Aufwand für ein funktionsfähiges und praxistaugliches Programm sowohl vom mathematischen Aufwand (Finden eines guten Algorithmus) als auch vom Programmieraufwand sehr groß ist.

Oft ist es möglich, bekannte Information dazu zu nutzen, die Boxen zu verkleinern ohne sie aufzuteilen. Weiß man etwa, daß die Funktion in der Box I_D konvex ist und im Punkt x_D ihr eindeutig bestimmtes Minimum annimmt, so läßt sich I_D auf den einzigen Punkt x_D reduzieren.

5.2 Branch und Probability Bound

Dieser Abschnitt folgt im großen und ganzen Zhigljavsky [41], S. 147ff.

Branch und Probability Bound unterscheidet sich vom Prinzip her nur wenig von

Branch und Bound. Beide Verfahren bestehen im wesentlichen aus zwei Schritten, die sich immer wiederholen:

Branch: Aus den bereits vorhandenen Boxen werden welche ausgewählt, die geteilt und durch die entstandenen Boxen ersetzt werden.

Bound: Die Funktionswerte in den Boxen werden geschätzt und danach werden Boxen auf Grund irgendeines Kriterium aus der Liste der zu bearbeitenden Boxen gelöscht.

Der Unterschied ist, daß bei Branch und Bound deterministisch festgelegt ist, wann man eine Box wegwirft – nämlich so, daß kein globales Minimum verlorengehen kann, wie in Abschnitt 5.1 beschrieben – während bei Branch und Probablity Bound dafür Wahrscheinlichkeiten zu Rate gezogen werden. Mit Hilfe der Funktion φ schließt man dann solange Boxen aus, bis man die Minima in so kleinen Boxen eingesperrt hat, wie man sich anfangs vorgegeben hat oder ein anderes Abbruchskriterium erfüllt ist. Da man immer wieder Boxen wegwirft, in denen ein Minimum liegen könnte (wenn auch mit geringer Wahrscheinlichkeit), erfüllt Branch und Probability Bound nicht die Bedingung (5.1).

5.2.1 Bewertung von Boxen

Anders als bei Branch und Bound verwendet man hier eine reellwertige Funktion

$$\varphi : \quad \mathcal{B} \to \mathbb{R}$$

welche die „Güte“ einer Box repräsentieren soll. Je kleiner φ, desto besser die Box. Der Definitionsbereich $\mathcal{B}$ sei dabei die Menge der möglichen Boxen.

Die Funktion φ sollte nun die Eigenschaft besitzen, daß für je zwei Mengen B_1 und B_2 die Ungleichung $\varphi(B_1) \geq \varphi(B_2)$ immer dann gilt, wenn die geschätzte Wahrscheinlichkeit dafür, daß das globale Optimum in B_1 liegt mindestens so groß ist als die, daß es in B_2 liegt. Man kann nun verschiedene Kriterien für diese geschätzte Wahrscheinlichkeit annehmen. Von den folgenden Kriterien besitzt nur Kriterium a) obige Eigenschaft und sollte daher auch verwendet werden.

Sei im Folgenden M_D der minimale Funktionswert in B und f_k^* der beste bisher gefundene Funktionswert. Unter der Voraussetzung, daß $B \in \mathcal{B}$ gilt, kann man etwa ein φ mit einer der folgenden Eigenschaften verwenden:

a) $\varphi(B)$ ist ein Schätzer für die Wahrscheinlichkeit $P(M_B \geq f_k^*)$.

b) $\varphi(B)$ ist die untere Grenze eines Konfidenzintervalls für M_B mit fix vorgegebenem Konfidenzniveau.

c) $\varphi(B)$ ist ein Schätzer für den minimalen Funktionswert in D, also für

$$M_B = \min_{x \in B} f(x)$$

d) $\varphi(B)$ ist ein Schätzer des Mittelwertes

$$\int_B f(x)\,\mathrm{d}x$$

Die empfohlene Reihenfolge entspricht der angegebenen, und wie schon erwähnt sollte a) der Vorzug gegeben werden. In schwierigen Fällen, etwa bei einer verrauschten Zielfunktion, wenn a)–c) nicht anwendbar sind, kann man meist noch auf d) zurückgreifen.

Ein φ, welches a) genügt, kann folgendermaßen konstruiert werden:
In der Menge aller Boxen werden im Laufe des Verfahrens zufällig Punkte bestimmt. Die Punkte, welche in B liegen, seien $\xi_l,\ l = 1, \ldots, m$. Ohne Beschränkung der Allgemeinheit seien sie so sortiert, daß gilt:

$$f(\xi_1) \geq f(\xi_2) \geq \ldots \geq f(\xi_m)$$

Dann ist die Gütefunktion definiert als:

$$\varphi(f^*, B) \ := \begin{cases} 1 & f(\xi_m) = f(\xi_{m-i}) \neq f^* \\ \left(1 - \left(\frac{f(\xi_m) - f^*}{f(\xi_{m-i}) - f^*}\right)^{\frac{n}{2}}\right)^i & \text{sonst} \end{cases} \tag{5.2}$$

Dabei ist n wie üblich die Dimension des Problems und f^* der beste bisher gefundene Funktionswert. Ein Wert für i wird in Abschnitt 5.2.5 vorgeschlagen. $\varphi(f^*, B)$ kann als Wahrscheinlichkeit interpretiert werden, da nur Werte im Intervall $[0,1]$ angenommen werden können. $\varphi(f^*, B)$ ist groß, wenn der kleinste bekannte Funktionswert in B, $f(\xi_m)$, nahe bei f^*, dem kleinsten bisher berechneten Funktionswert, liegt. Der Wert $\varphi(f^*, B)$ ist auch dann groß, wenn der Quotient $(f(\xi_m) - f^*)/(f(\xi_{m-i}) - f^*)$ klein wird. Das passiert dann, wenn sich $f(\xi_{m-i})$ und $f(\xi_m)$ stark unterscheiden. Zusammenfassend läßt sich sagen, daß $\varphi(f^*, B)$ daher als Gütefunktion geeignet ist.

Zhiglajavsky [41] gibt für $\varphi(f^*, B)$ zwei Interpretationsmöglichkeiten:
Im asymptotisch schlechtesten Fall (i ist konstant und $m \to \infty$) ist

$$\varphi(f^*, B) \quad \geq \quad P\left(\sup_{x \in B} f(x) \geq f^*\right)$$

oder $\varphi(f^*, B)$ ist mindestens so groß wie die Wahrscheinlichkeit, daß die Hypothese

$$H_0: \quad \sup_{x \in B} f(x) \;\geq\; f^*$$

bei der Alternative

$$H_1: \quad \sup_{x \in B} f(x) \;<\; f^*$$

angenommen wird.

Ein φ, welches d) erfüllt ist noch einfacher zu berechnen:
Seien $\xi_1, \ldots, \xi_m$ wie oben die in B liegende Punkte, dann kann man φ folgendermaßen wählen:

$$\varphi(B) = \frac{1}{n+1} \sum_{l=1}^{m} f(\xi_l)$$

Es ist klar, daß sich $\varphi(f^*, B)$ beziehungsweise $\varphi(B)$ im Laufe des Algorithmus ständig verändert, da neue Punkte hinzukommen und auch f^* nicht gleich bleibt.

5.2.2 Boxen verwerfen

Bei Branch und Probability Bound wird eine Box dann weggeworfen, wenn die geschätzte Wahrscheinlichkeit dafür, daß ein globales Optimum in ihr liegt, „klein“ ist, während bei Branch und Bound eine Box nur dann weggeworfen wird, wenn sicher kein globales Optimum in der Box liegt.

Die Bedingung, wann eine Box D verworfen ist, lautet:

$$\varphi(f^*, D) \quad \leq \quad \varepsilon$$

In diesem Fall ist nämlich die geschätzte Wahrscheinlichkeit, daß die Box ein Minimum enthält, so klein, daß man wagt, sie wegzuwerfen. Es ist klar, daß bei kleinem ε mehr Boxen behalten werden und somit der Algorithmus langsamer wird, während bei größerem ε dafür leichter ein Minimum „verloren geht“.

Am Ende jeder Iteration kann man die Gleichung (5.2) auf die Vereinigung aller zu verwerfender Boxen anwenden und kann somit die Wahrscheinlichkeit abschätzen, mit der ein globales Minimum verworfen wird.

5.2.3 Zufallspunkte wählen

In jedem Iterationsschritt wird eine Anzahl von neuen Zufallspunkten bestimmt. Es ist nicht immer ratsam, diese neuen Zufallspunkte gleichverteilt in der Menge der Boxen zu wählen. Besser ist es, in besseren Boxen mehr und in schlechteren Boxen weniger Punkte auszuwählen. Man kann etwa die Anzahl der Punkte aus den Boxen B_i direkt proportional zu den $\varphi(f^*, B_i)$ wählen.

Da $\varphi(f^*, B)$ nicht nur von den Punkten in B abhängt, sondern durch f^* auch von denen in den anderen Boxen, kann es vorkommen, daß nach einigen Iterationen die Box B so schlecht wird, daß $\varphi(f^*, B) \leq \varepsilon$ wird, und sie verworfen wird. Deshalb empfiehlt es sich, in Boxen, für welche mit einem geeigneten ε^* gilt:

$$\varepsilon \ < \ \varphi(f^*, B) \ \leq \ \varepsilon^*$$

einige Iterationen lang keine Punkte mehr zu wählen und abzuwarten, ob B nicht verworfen wird.

Wie bereits erwähnt, wird die Güte einer Box auf Grund der Zufallspunkte ermittelt. Um bessere Ergebnisse zu erhalten, verwendet man am besten die neuen, also die in dieser Iteration bestimmten, und auch alte Punkte. Um nicht zu viele Punkte verwalten zu müssen, kann man sich auf alte Punkte beschränken, deren Funktionswerte ausreichend nahe bei f^* liegen.

5.2.4 Boxen teilen

Dieser Teil des Algorithmus hat die Aufgabe, große Boxen in mehrere kleinere umzuwandeln. Die 0-te Box sei dabei der ursprüngliche Definitionsbereich, den wir mit D_0 bezeichnen wollen. Dabei muß man sich überlegen, wie man die vorhandenen Boxen geeignet verwaltet. Es empfiehlt sich, eine Liste der Boxen anzulegen, aus der man gegebenenfalls Boxen löschen oder neue hinzufügen kann. Das wird nötig, wenn man Boxen wegwirft beziehungsweise durch mehrere neue ersetzt.

Die Form der Boxen unterliegt dabei prinzipiell wenigen Einschränkungen. Die einfachste Variante ist es, wenn man (hyper-)quaderförmige Boxen verwendet, da man diese sehr leicht teilen kann, zum Beispiel, wenn man sie ganz einfach in der n-ten Koordinate halbiert. Es ist aber auch möglich, ellipsoide Boxen zu verwenden. Es wird übrigens nicht verlangt, daß die Boxen paarweise disjunkt sein müssen. Mehr Details dazu findet man etwa in Zhigljavsky [41], S. 148f.

Prinzipiell kann man anfänglich festlegen, wie die Boxen geteilt werden oder man teilt die Boxen adaptiv, also abhängig von den Funktionswerten. Letzteres ist aufwendiger aber auch flexibler und daher wohl die bessere Wahl.

In jedem Iterationsschritt wird die Menge der Boxen aktualisiert. Eine Möglichkeit ist es, wieder ein ε^* zu verwenden und Boxen mit

$$\varphi(f^*, B) \;\geq\; \varepsilon^*$$

zu teilen.

Ein anderer Weg ist folgender:

Die Punkte ξ_l seien die m zufällig bestimmten Punkte. Bei jeder Iteration k werden im aktuellen Bereich D_k solange Boxen $B_{k,j}$ ausgewählt, welche jeweils den Punkt ξ_j im Zentrum haben, bis D_k vollständig überdeckt ist oder die Hypothese, daß das globale Minimum in der von den $B_{k,j},\ j = 1, \ldots, r$ noch nicht überdeckten Menge

$$D_k \setminus \bigcup_{j=1}^{r} B_{k,j}$$

liegt, verworfen wird. Dabei gilt für die ξ_j:

$$\xi_j \in D_k \setminus \bigcup_{s=1}^{j-1} B_{k,s} \qquad \text{mit} \qquad f(\xi_j) \geq f(\xi_l), \qquad \text{für alle} \qquad \xi_l \in D_k \setminus \bigcup_{s=1}^{j-1} B_{k,s}$$

Nun wird ein neuer Suchbereich D_{k+1} bestimmt. Das geschieht, indem man D_{k+1} gleich

$$B_k := \bigcup_{j=1}^{r} B_{k,j}$$

wählt. Will man eine bestimmte Gestalt von D_{k+1} erreichen, dann kann man D_{k+1} auch als (Hyper-)Rechteck wählen, welches B_k überdeckt oder mehrere Rechtecke oder Ellipsoide mit derselben Eigenschaft.

5.2.5 Algorithmus

Hier will ich noch kurz einen Algorithmus skizzieren, der durch die oben dargestellten Sachverhalte motiviert ist (Zhigljavsky [41], S. 154f):

Algorithmus 5.1

Schritt 1: *Sei* $k = 0$, $D_0 = D$ *und* $f^*_{-1} = \infty$

Schritt 2: *Wähle zufällig* N *Punkte* $\xi_{k,j}$ $(j = 1, \ldots, N)$ *im Suchbereich* D_k.

Schritt 3: *Bestimme*

$$f^*_k := \max\{f_{k-1}, f(\xi_{k,1}), \ldots, f(\xi_{k,N})\}$$

Schritt 4: *Setze* $Y_{k,0} = D_k$ *und* $j = 1$.

Schritt 5: *Setze* $Y_{k,j} = Y_{k,j-1} \setminus B_{k,j}$, *wobei* $B_{k,j}$ *ein (Hyper-)Würfel mit vorgegebenem Inhalt sein soll, welcher um den Punkt* $\xi_j \in Y_{k,j-1}$ *zentriert ist, der den kleinsten Funktionswert aufweist.*

Schritt 6: *Ist die Anzahl* m *der berechneten Punkte in* $Y_{k,j}$ *zu klein, um vernünftige statistische Aussagen erhalten zu können (etwa* $m \leq m_0$*), dann setze man mit* $D_{k+1} = D_k$ *bei Schritt 5.1 fort. Ansonsten wird* $\varphi(Y_{k,j})$ *an Hand von Gleichung* (5.2) *bestimmt, wobei die Werte* η_m *und* η_{m-i} *den sortierten Funktionswerten in*

$$Y_{k,j} \cap \bigcup_{l=1}^{k} \{\xi_{k,1}, \ldots, \xi_{k,N}\}$$

entsprechen.

Ist $\varphi(Y_{k,j}) \leq \varepsilon$, *dann gehe zu Schritt 5.1.*

Schritt 7: *Erhöhe* j *um* 1 *und gehe zu Schritt 5.1.*

Schritt 8: *Wähle* D_{k+1} *als Vereinigung von disjunkten Hyper-Rechtecken, welche die Vereinigung* $B_{k,1} \cup \cdots \cup B_{k,j}$ *überdecken.*

Schritt 9: *Ersetze* k *durch* $k+1$ *und gehe zu Schritt 5.1.*

Dabei kann man etwa $N = 100$, $m_0 = 15$, $\rho = 0.1$ *und* $i = \min\{5, \lfloor m/10 \rfloor\}$ *wählen.* $\lfloor x \rfloor$ *ist die größte ganze Zahl, für die* $\lfloor x \rfloor \leq x$ *gilt.*

Wie in Abschnitt 5.2.3 erwähnt, kann man in Schritt 5.1 die Anzahl der Punkte in den jeweiligen Boxen adaptiv wählen. Der Vollständigkeit wegen kann man in diesen Algorithmus noch ein Abbruchkriterium einbauen. Das kann etwa durch eine untere Schranke für die Boxengröße geschehen oder auch durch Analyse der Streuung aller in den Boxen liegenden Punkte. Um den Algorithmus endgültig abzuschließen, kann man am Ende noch lokal minimieren.

5.3 Evolutive Algorithmen

Das Wesen eines evolutiven (beziehungsweise genetischen) Algorithmus besteht in der wiederholten Ausführung folgender Schritte:

1. Aus einer Menge $\mathcal{M}_t$ von Punkten im Definitionsbereich, der sogenannten *Population*, werden Punkte, die *Individuen*, bestimmt.

2. Diese Individuen werden durch verschiedene Operationen verändert und bilden dann die nächste *Generation* $\mathcal{M}_{t+1}$.

Wesentlich dabei ist, daß neue Individuen durch Kombination von mehreren alten gebildet werden können. Bei Abbruch des Programms sollte das bisher beste Chromosom eine gute Näherung für das globale Optimum liefern.

Da es eine fast unüberschaubare Anzahl von Möglichkeiten zur Auswahl und Modifikation von Punkten gibt, ist es unmöglich von *dem* evolutiven Algorithmus zu sprechen, und man kann vermutlich zu jedem Problem einen Algorithmus konstruieren, der schnell eine Lösung dieses Problems findet. Andererseits ist es vermutlich sehr schwer, ein Programm zu schreiben, welches generell gute Ergebnisse liefert und ohne viel herumzuprobieren einfach eingesetzt werden kann.

Manche Autoren verstehen unter „genetischen Algorithmen“ nur solche, welche auf eine „diskrete“ Darstellung der Werte aufbauen, also statt der Darstellung einer Dezimalzahlen als Dezimalzahl dazu mehrere ganze Zahlen verwendet (siehe folgender Abschnitt). Diese Arbeit wird sich auch an diese Konvention halten. Im folgenden wird nicht mehr explizit darauf hingewiesen, wann von genetischen beziehungsweise evolutiven Algorithmen die Rede ist, da dies aus dem Zusammenhang klar sein sollte.

Die folgendenden Abschnitte befassen sich im wesentlichen mit Algorithmen, welche aus Genen aufgebaute Chromosomen verwenden, wie dies bei den Programmen `Genocop` (Abschnitt 6.2.2) und `PGAPack` (Abschnitt 6.2.4) der Fall ist. Ein anderer Zugang wird im Programm `DE` verwendet und in Abschnitt 6.2.3 beschrieben.

5.3.1 Darstellung der Individuen

Im Gegensatz zu den meisten anderen Optimierungsverfahren, verwenden evolutive Algorithmen oft nicht einen n-dimensionalen, reellwertigen Vektor für die Darstellung der Punkte im Definitionsbereich. Statt dessen werden die Punkte auch als Binär- oder Integer-Vektoren kodiert, welche *Chromosomen* genannt werden. Die einzelnen Eintragungen heißen dann die *Gene* des Choromosoms.

Will man bei einer Kodierung in der Binärdarstellung den Definitionsbereich mit einem Gitter einer vorgegebenen Genauigkeit ε erreichen, dann benötigt man für jede Koordinate k Stellen:

$$k = \left\lceil \frac{\log(l) - \log(\varepsilon)}{\log 2} \right\rceil$$

Dabei ist l die größte Ausdehnung des Definitionsbereiches entlang den Koordinatenachsen und $\lceil x \rceil$ ist die kleinste ganze Zahl, für die $\lceil x \rceil \geq x$ gilt. Ein Chromosom hat dann folgende Gestalt:

$$(b_{nk-1} \cdots b_{(n-1)k} \cdots b_{k-1} \cdots b_1 b_0), \qquad b_i \in \{0, 1\}$$

Es ist natürlich auch möglich, die Genauigkeit koordinatenweise festzulegen. Die Bestimmung der Chromosomen würde sich nur unwesentlich verändern.

Wie man zwischen einer reellen Zahl x, welche im Intervall $[x_u, x_o]$ liegt, und deren Binärdarstellung umrechnet, ergibt sich aus folgenden Formeln:

$$\begin{aligned} x' &= (b_{k-1} \cdots b_0)_2 = \left(\sum_{i=0}^{k-1} b_i 2^i \right)_{10} \\ x &= x_u + \frac{x_o - x_u}{2^k - 1} x' \end{aligned}$$

Diese Methode, ein Chromosom zu kodieren, kann man im 1-dimensionalen Fall einfach verwenden. Im höher-dimensionalen Fall ist es aus Gründen, die in Abschnitt 5.3.4 klar werden, nicht gut, obige Kodierungsmethode koordinatenweise anzuwenden und dann die jeweiligen Ergebnisse aneinanderzureihen um ein Chromosom zu erhalten. Das liegt daran, daß Änderungen an einem weiter vorne stehenden Gen größere Auswirkungen auf die Position des Punktes im Definitionsbereich haben, als die Änderung eines späteren Gens. Deshalb scheint es sinnvoll, die Gene geeignet zu mischen. Sei $b^i_{k-1} \cdots b^j_0$ $(j = 1, \ldots, n)$ die Kodierung der i-ten Koordinate, dann sei das Chromosom folgendermaßen aufgebaut:

$$(b_{nk-1} \cdots b_{k-1} \cdots b_1 b_0) = (b^1_{k-1} b^2_{k-1} \cdots b^n_{k-1} b^1_{k-2} \cdots b^{n-1}_0 b^n_0) \qquad b_i, b^j_i \in \{0,1\}$$

Aber auch so werden die einzelnen Koordinaten nicht ganz gleich behandelt. Es ist aber möglich, diese Schwierigkeiten zu umgehen (siehe Abschnitt 5.3.4).

Will man die Gene mit Werten aus den natürlichen Zahlen haben, so wählt man die größte zu verwendende Zahl und kodiert analog zu oben im entsprechenden Zahlensystem. Auch hier sollte man auf die Reihenfolge achten. Auch reelle Gene sind natürlich möglich.

Verwendet man ein Programm, welches mit reellen Vektoren umgehen kann, wie etwa `Differential Evolution` (Abschnitt 6.2.3), so läßt sich ein Punkt wie üblich als reellwertiger Vektor darstellen. In diesem Fall ist es nicht möglich, eine Misch-Strategie zu verwenden.

Im folgenden werde ich nicht mehr zwischen Punkten x und Chromosomen unterscheiden und bei Bedarf auf Schwierigkeiten, welche sich aus der Darstellung ergeben, speziell hinweisen.

5.3.2 Initialisierung

Es gibt prinzipiell zwei Möglichkeiten, wie die erste Generation in diversen Programmen bestimmt wird. Entweder wird von einem vorgebebenen Punkt aus gestartet, also alle Chromosomen der ersten Generation sind identisch oder die Startpopulation wird zufällig gewählt. Hier könnte man auch bei genetischen

Algorithmen Quasi-Zufallszahlen verwenden da es hier auf die Abdeckung des Definitionsbereichs ankommt. Da die Algorithmen aber von der Startpopulation weitgehend unabhängig sein sollten, scheint das von untergeordneter Bedeutung zu sein.

Sollte man bereits a priori Kenntnis über die mögliche Lage der (globalen) Minima haben, so kann man das in der Wahl der ersten Generation berücksichtigen.

Eine wichtiger Parmameter bei einem genetischen Algorithmus ist die Anzahl S der Punkte, welche eine Population enthält. Ist sie zu klein, konvergiert der Algorithmus zu schnell und somit vermutlich gegen ein lokales Optimum. Wählt man n wiederum zu groß, so gleicht der Algorithmus zu sehr einer Zufallssuche (siehe Abschnitt 6.2.1) und der Fortschritt wird zu klein. Die tatsächliche Größe von S hängt sehr von der Implementation ab.

Prinzipiell wäre es möglich, S je nach Fortschritt des Programms variabel zu gestalten (siehe Michalewicz [28], S. 72ff). In Phasen, wo der Definitionsbereich abgesucht werden soll empfehlen sich größere S, wohingegen für die Konvergenz gegen vielversprechende Gebiete kleinere S geeigneter sind. Im folgenden möchte ich mich aber auf eine konstante Populationsgröße S festlegen.

5.3.3 Auswahl

Der Sinn dieses Teils des Algorithmus besteht darin, verstärkt „bessere“ Chromosomen als Basis für die nächste Generation $\mathcal{M}_{t+1}$ auszuwählen. Dabei muß aber darauf geachtet werden, daß eine nötige Vielfalt erhalten bleibt, damit das Verfahren nicht zu rasch gegen ein lokales Minimum konvergiert. Dazu kann man entweder die Funktionswerte selbst oder eine daraus abgeleitete Rangordnung heranziehen (Michalewicz [28], S. 58ff). Eine einfache Möglichkeit, Punkte auszuwählen besteht darin, jedem Punkt $x_i \in \mathcal{M}_k$ eine Wahrscheinlichkeit

$$p_i = \frac{f(x_{max})}{f(x_i)\sum_{j=1}^{S}\frac{f(x_{max})}{f(x_j)}}, \qquad i = 1, \ldots, S$$

zuzuordnen und entsprechend der Wahrscheinlichkeiten S mal einen Punkt auszuwählen. Dabei sei x_{max} jener Punkt von $\mathcal{M}_t$ mit dem größten Funktionswert. Sucht man ein globales Maximum, so kann man die Wahrscheinlichkeiten auch einfacher wählen:

$$p_i = \frac{f(x_i)}{\sum_{j=1}^{S} f(x_j)}, \qquad i = 1, \ldots, S$$

Den Chromosomen entsprechen so disjunkte, rechts offene Teilintervalle von $[0, 1)$, und wenn eine Zufallszahl in einem solchen Teilintervall liegt, gilt der zugehörende Punkt als ausgewählt. Diese Methode hat einige Nachteile:

- $f(x)$ darf nicht negativ sein. Sollte $f^* \leq 0$ sein, so muß man $f(x)$ durch $f(x) + C$ mit $C > -f^*$ ersetzen. Dazu muß aber eine untere Schranke $-C$ für f^* bekannt sein.

- $f(x)$ und $f(x) + C$ mit $C \in \mathbb{R}$ unterscheiden sich als Optimierungsproblem nur unwesentlich, führen aber zu unterschiedlichen Wahrscheinlichkeiten p_i. Je größer C ist, umso ähnlicher sind sich die Wahrscheinlichkeiten und das Auswahlkriterum gleicht immer mehr einer rein zufälligen Auswahl.

- Kommt es einmal zu einem „Super“-Punkt, dessen Funktionswert um sehr viel kleiner ist der aller anderen, so wird dieser sehr oft ausgewählt und die Vielfalt in der Population geht verloren. Der Definitionsbereich wird nicht mehr so gründlich abgesucht. Da der „Super“-Punkt möglicherweise nur ein lokales Minimum repräsentiert, bleibt der Algorithmus oft in diesem hängen.

Eine mögliche Modifikation ist es, nicht S-mal zufällig ein Chromosom zu bestimmen, sondern nur einmal eine Zufallszahl zu berechnen. Die fehlenden $S - 1$

Punkte werden bestimmt, indem man zu der einen Zufallszahl den Wert $1/S$ entsprechend oft addiert beziehungsweise subtrahiert (das führt zu genau S Punkten im Intervall $[0, 1)$).

Beim unmodifizierten Auswahlkriterium könnte man den letzten Nachteil dadurch vermeiden, daß man eine Obergrenze für die Auswahl jedes einzelnen Punktes einführt.

Eine andere Möglichkeit, die oben angesprochenen Nachteile zu vermeiden, ist es, die Punkte entsprechend ihren Funktionswerten zu ordnen und die Wahrscheinlichkeit, daß ein Punkt ausgewählt wird nun von seinem Rang R abzuleiten. Der Punkt mit dem niedrigsten Funktionswert erhält Rang $R = 1$ und der mit dem größten Funktionswert den Rang $R = S$.

Zwei Varianten, die Wahrscheinlichkeit für die Auswahl des Punktes mit dem Rang R sind:

- Die Wahrscheinlichkeiten nehmen linear ab:
$$p_R = q - (R-1)\left(q - \frac{1}{S}\right)\frac{2}{S-1}$$
- Die Wahrscheinlickeiten nehmen nicht-linear ab:
$$p_R = \frac{q\,(1-q)^{R-1}}{1-(1-q)^S}$$

In beiden Fällen ist q ein Parameter, der das Verhältnis zwischen der Vielfalt in der kommenden Generation und der Konvergenz bestimmt und also in etwa der oben eingeführten Konstanten C entspricht. Im ersten Fall muß $1/S \leq q \leq 2/S$ und im zweiten $0 \leq q \leq 1$ gelten. Größere q führen jeweils zu einer stärkeren Auswahl von niederen Rängen.

Eine weitere interessante Möglichkeit bietet *Turnier-Auswahl.* Hier wählt man S-mal zufällig (gleichverteilt) κ Punkte $x_1, \ldots, x_\kappa$ aus der aktuellen Generation und verwendet jeweils den besten Punkt. Je größer κ ist, umso häufiger werden Punkte mit kleineren Funktionswerten ausgewählt, und die Gefahr, in einem lokalen Minimum hängenzubleiben, steigt. Oft wird $\kappa = 2$ gewählt.

Es wäre auch denkbar, das Auwahlkriterium von Generation zu Generation variieren zu lassen, denn so könnte man ähnlich wie bei Simulated Annealing (Abschnitt 5.4) eine zeitlich variable „Temperatur“ $T \geq 0$ einführen und den besseren Punkt mit Wahrscheinlichkeit

$$p_1 = \frac{1}{1 + e^{\frac{f(x_1) - f(x_2)}{T}}}$$

auswählen. Der Punkt x_1 sei dabei jener mit dem kleineren Funktionswert. Anders als bei Simulated Annealing ist es aber nicht so klar, wie man den Verlauf von T bei evolutiven Algorithmen wählen soll.

Es sind noch viele andere Varianten denkbar, etwa ob das beste Chromosom immer ausgewählt wird oder nicht, deren Diskussion aber den Rahmen dieser Arbeit sprengen würde.

5.3.4 „Evolutive“ Operationen

Hat man nun S Chromosomen augewählt, so werden diese noch auf verschiedene Weisen modifiziert, um neue Punkte zu erhalten und so den Definitionsbereich zu durchsuchen. Auch dazu gibt es wieder viele verschiedene Möglichkeiten. Die folgenden Operationen (oder ähnliche) werden jeweils mit einer bestimmte Wahrscheinlichkeit oder an einer bestimmten Anzahl von Chromosomen angewandt. Dabei lassen sich die Wahrscheinlichkeiten der Operatoren von Generation zu Generation natürlich wieder variieren.

Für die konkrete Programmierung muß man sich noch entscheiden, ob man auf jedes Chromosom mehrere, genau einen oder höchstens einen der folgenden Operatoren wirken lassen will. Alle Chromosomen sind hier binär kodiert. Es sollte keine Schwierigkeit bereiten, die Operatoren auch für reelle oder natürliche Darstellungsformen umzuschreiben.

Crossover

Crossover wirkt jeweils auf zwei Chromosomen und wird mit einer Wahrscheinlichkeit p_c auf einen Punkt angewandt. p_c liegt dabei typischerweise nahe bei 1. Die Idee dahinter ist es, die Vorteile von mehreren Chromosomen in einem neuen zu vereinen und somit einen Fortschritt bei der Optimierung zu erzielen.

Die einfachste Form des Crossover ist 1-Punkt-Crossover, wie in folgender Gleichung schematisch dargestellt. Dabei wird zufällig ein Wert α aus dem Bereich $2, \ldots, k$ ausgewählt und ab diesem Index die Eintragungen der beiden Chromosomen vertauscht. k ist dabei die Länge eines Chromosoms.

$$\begin{array}{c} (0\,1\,0\,0\,1 \mid 1\,1\,0\,1) \\ (0\,0\,1\,1\,1 \mid 0\,1\,0\,0) \end{array} \longrightarrow \begin{array}{c} (0\,1\,0\,0\,1 \mid \mathbf{0\,1\,0\,0}) \\ (0\,0\,1\,1\,1 \mid \mathbf{1\,1\,0\,1}) \end{array}$$

Häufig wird auch 2-Punkt-Crossover verwendet. Hier werden alle Eintragungen zwischen zwei Werten $1 \leq \alpha_1 \leq \alpha_2 \leq k$ ausgetauscht. Zur Veranschaulichung dient folgende Gleichung, in welcher $\alpha_1 = 4$ und $\alpha_2 = 7$ gewählt ist.

$$\begin{array}{c} (0\,1\,0 \mid 0\,1\,1\,1 \mid 0\,1) \\ (0\,0\,1 \mid 1\,1\,0\,1 \mid 0\,0) \end{array} \longrightarrow \begin{array}{c} (0\,1\,0 \mid \mathbf{1\,1\,0\,1} \mid 0\,1) \\ (0\,0\,1 \mid \mathbf{0\,1\,1\,1} \mid 0\,0) \end{array}$$

Selbstverständlich könnte man analog noch andere Varianten des Crossover bilden.

Hier wird klar, wie die Bemerkung aus Abschnitt 5.3.1 gemeint war. Die Wahrscheinlichkeit, daß die ersten Eintragungen eines Chromosoms bei einem 1-Punkt-Crossover verändert wird, ist sehr gering, während die späteren Werte mit hoher Wahrscheinlichkeit vertauscht werden.

Will man alle Gene gleichberechtigt behandeln, so kann man das erreichen, indem man sich eine zufällige „Maske“ bestimmt und an den Stellen, wo bei der Maske eine 1 steht, die Werte der beiden Chromosomen vertauscht. In folgender Gleichung ist diese Methode mit der Maske (1 0 0 1 1 0 0 1 1) veranschaulicht:

$$\begin{array}{c} (0\,1\,0\,0\,1\,1\,1\,0\,1) \\ (0\,0\,1\,1\,1\,0\,0\,1\,0) \end{array} \longrightarrow \begin{array}{c} (\mathbf{0}\,1\,0\,\mathbf{1\,1}\,1\,1\,\mathbf{1\,0}) \\ (\mathbf{0}\,0\,1\,\mathbf{0\,1}\,0\,0\,\mathbf{0\,1}) \end{array}$$

Dieser Operatortyp beinhaltet gewissermaßen globale und lokale Suche in einem. Je größer etwa α bei 1-Punkt-Crossover ist, desto lokaler sind die Veränderungen an den Punkten, und kleine α erlauben große Sprünge im Definitionsbereich.

Crossover kann nur die Information nutzen, die bereits in der aktuellen Generation vorhanden ist. Hat etwa jedes Chromosom eine **0** an der dritten Stelle, so kann dies durch Crossover nicht verändert werden. Daher ist es Vorteilhaft „Mutationen“ einzuführen, um den Defintionsbereich besser durchsuchen zu können.

Mutationen

Dieser Operator wird mit einer kleinen Wahrscheinlichkeit p_m auf ein Gen angewandt. p_m bewegt sich typischerweise im Bereich von wenigen Prozent beziehungsweise Promille. Für jedes Gen wird dabei eine Zufallszahl im Intervall $[0, 1]$ bestimmt und falls diese kleiner als p_m ist, wird der Wert des Gens geändert, also statt einer **0** eine **1** beziehungsweise statt einer **1** eine **0** geschrieben. Die in Davis [12] S. 15f eingeführte Methode bei einer Mutation einem Gen einen *zufälligen* Wert zuzuordnen, halte ich nur für wenig zielführend, da in etwa der Hälfte der Fälle eine **1** mit einer **1** oder eine **0** mit einer **0** überschrieben wird. Es würde also effektiv nichts geschehen, aber trotzdem Rechenzeit verbraucht werden.

Auch dieser Operator wirkt global oder lokal, je nachdem ob ein geändertes Gen eher vorne oder hinten in einem Chromosom steht.

Inversionen

Eine weitere Möglichkeit, ein Chromosom zu verändern besteht darin, einen Teil seiner Gene in umgekehrter Reihenfolge anzuordnen. Dazu werden wieder zwei Stellen $1 \leq \alpha_1 < \alpha_2 \leq k$ gewählt und die Reihenfolge der Gene von α_1 bis α_2 umgedreht. In folgender Gleichung ist dies mit $\alpha_1 = 4$ und $\alpha_2 = 8$ exemplarisch dargestellt:

$$(0\,1\,1 \mid 1\,1\,0\,1\,0 \mid 1) \quad \longrightarrow \quad (0\,1\,1 \mid \mathbf{0\,1\,0\,1\,1} \mid 1)$$

Natürlich ist auch hier eine Unzahl von weiteren Operatoren denkbar, auf die aber hier nicht eingegangen wird.

5.3.5 Theoretische Grundlagen

In Michalewicz [28] wird für genetische Algorithmen erläutert, warum diese Methode zumindest bei binärer Codierung zielführend ist. Dazu werden Schemata verwendet. Ein Schema ist wie ein Chromosom aufgebaut, nur daß zusätzlich zu 0 und 1 auch das Symbol $*$ verwendet wird. Ein Schema repräsentiert dabei alle Chromosomen, welche an allen Stellen, die kein $*$ als Eintrag haben, mit dem Schema übereinstimmen. So entspricht also das Schema $S = (0 * 1\,1\,1\,0\,1\,0\,1)$ den beiden Chromosomen $(0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1)$ und $(0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1)$. Es ist leicht einzusehen, daß ein Schema, welches r $*$-Symbole enthält, genau 2^r Chromosomen repräsentiert. Zwei wichtige Charakteristiken sind die *Ordnung* $o(S)$ und die *definierende Länge* $\delta(S)$ eines Schemas. $o(S)$ ist die Länge des Schemas abzüglich der Anzahl der $*$-Symbole und $\delta(S)$ ist die Differenz zwischen der ersten und letzten fixen Stelle des Schemas.

Wie einzelne Chromosomen, kann man auch Schemata bewerten. Dazu kann man etwa das Mittel der Funktionswerte aller Chromosomen einer Population, welche vom Schema S repräsentiert werden, verwenden. Folgender Satz besagt im wesentlichen, warum dies möglich ist:

Satz 5.2 (Schema Theorem) *Sei die Populationsgröße extrem groß. Schemata, die eine kleine definierende Länge haben, eine kleine Ordnung aufweisen und über einige Generationen überdurchschnittlich gut bleiben, werden in aufeinanderfolgenden Generationen eine exponentionell wachsende Anzahl von Chromosomen der Population repräsentieren.*

Der Beweis besteht eigentlich nur darin, auszurechnen, mit welcher Wahrscheinlichkeit ein Schema gewisse genetische Operationen unbeschadet übersteht und dies mit der zu erwartenden Nachkommenschaft in Relation zu setzen. So gilt etwa für die Wahrscheinlichkeit p_1, daß ein Schema S von 1-Punkt-Crossover zerstört wird:

$$p_1(S) \approx 1 - p_c \frac{\delta(S)}{m-1}$$

Dabei ist p_c die Wahrscheinlichkeit für ein Crossover und m ist die Anzahl der Stellen des Chromosoms. Gleichheit gilt deshalb nicht, da bei Crossover ein Schama nicht zerstört werden *muß*. Ähnlich gilt für die Wahrschenlichkeit p_2, daß ein Schema S den Mutations-Operator nicht überlebt:

$$p_2(S) = (1 - p_m)^{o(S)} \approx 1 - o(S)p_m$$

Dabei ist p_m die Wahrscheinlichkeit, daß eine Stelle durch Mutation verändert wird. Da p_m in der Regel klein ist, ist die Approximation gerechtfertigt. Die Anzahl $\xi(S, t+1)$ der Chromosomen in der Generation $t+1$, welche vom Schema S repräsentiert werden, weisen etwa folgendes Wachstumsverhalten auf:

$$\xi(S, t+1) \geq \xi(S, t)F(S, t)(1 - p_1 - p_2)$$

Dabei ist $F(S, t)$ eine Bewertungsfunktion, die dann groß wird, wenn das Schema S in der Generation t Chromosomen mit kleinen Funktionswerten repräsentiert.

Abgesehen von der Forderung nach einer extrem großen Populationsgröße, zeigt Satz 5.2 bereits gewisse Schwierigkeiten von genetischen Algorithmen auf. Hat man nämlich ein überdurchschnittlich gutes Schema in einer Generation, das keinen globalen Minimalpunkt repräsentiert, so besteht die Gefahr, daß dieses Schema immer mehr Chromosomen repräsentiert und der Algorithmus zu rasch gegen ein lokales Minimum konvergiert. Oft ist man also dazu bestrebt, genau dieses starke Wachstum hintanzuhalten beziehungsweise in geordnetere Bahnen zu lenken. Weiters wächst die Wahrscheinlichkeit p_1 wenn man statt 1-Punkt-Crossover 2-Punkt-Crossover oder komplexere Crossover-Varianten verwendet.

5.4 Simulated Annealing

Simulated Annealing versucht, Zielfunktionen zu minimieren, indem eine Folge von Zufallspunkten (x_s) berechnet wird, welche im Idealfall gegen einen globalen Minimalpunkt konvergiert. Um zu vermeiden, in einem lokalen Minimum hängenzubleiben, wird explizit vorausgesetzt, daß die Folge nicht monoton fallend sein muß. Es werden also auch Punkte akzeptiert, die einen größeren Funktionswert als ihre Vorgänger haben. Veranschaulichen läßt sich das Prinzip durch einen Gummiball, welcher innerhalb des Definitionsbereiches auf der zu optimierenden Funktion herumhüpft. Anfänglich besitzt er noch viel Energie – eine hohe Temperatur – und kann sowohl weiter als auch höher springen. Im Laufe des Prozesses sinkt die Temperatur, und er konvergiert im besten Falle gegen ein globales Minimum. Es gibt also keine klare Unterscheidung zwischen lokaler und globaler Suche, da dies von einer variablen Temperatur abhängt.

Die Idee wurde erstmals von Metropolis et al. [27] aufgegriffen und besteht im Wesentlichen darin, vom jeweils letzten akzeptierten Punkt ausgehend, einen neuen Zufallspunkt zu bestimmen und diesen entsprechend einem vorgegebenem Kriterium als neuen Punkt der Folge (x_s) zu akzeptieren oder dies bleiben zu lassen. Um das alles zu erreichen, verwendet man drei funktionale Abängigkeiten, ähnlich den in Ingber et. al [19] beziehungsweise in Ingber [20] beschriebenen:

$g_T(x_k)$: Wahrscheinlichkeitsdichte über dem Definitionsbereich, gemäß der ein Punkt x_{neu} bestimmt wird, der dann als Punkt für die Zufallsfolge (x_s) in Frage kommt. $g_T(x_k)$ ist für die theoretische Betrachtung wichtig, während für die konkrete Implementation ein Generator für entsprechende Zufallspunkte von Bedeutung ist.

$h_T(\Delta f_k)$: Funktion, welche die Wahrscheinlichkeit angibt, mit welcher der Punkt x_{neu} als neuer Punkt der Folge akzeptiert wird, also ob $x_{k+1} = x_{\text{neu}}$ oder $x_{k+1} = x_k$ gesetzt wird.

$T(k)$: Funktion, welche den „Temperaturabfall" abhängig von der bereits verstrichenen Zeit k repräsentiert.

Sowohl $g_T(x_k)$ als auch $h_T(\Delta f_k)$ sind von $T(k)$ und somit von k abhängig. Δf_k ist dabei die Differenz der Funktionswerte $f(x_{\text{neu}}) - f(x_k)$. Es wäre natürlich möglich, für g_T und h_T verschiedene Temperaturfunktionen zu verwenden, wovon aber in Ingber et. al [19] abgeraten wird.
Es ist im Allgemeinen sehr schwierig, zu einer gegebenen Dichte $g_T(x_k)$ ein Verfahren anzugeben, welches entsprechend verteilte Punkte zu produzieren vermag. Daher wird statt dessen oft auch eine Markov-Kette, etwa ein n-dimensionaler Wiener Prozeß, verwendet. (Boender et. al [9].)
Boltzmann Annealing ist eine Variante, welche folgendermaßen beschrieben werden kann:

$$g_T(\Delta x_k) = (2\pi T(k))^{-\frac{n}{2}} e^{-\frac{(\Delta x)^2}{2T(k)}} \tag{5.3}$$

$$h_T(\Delta f_k) = \begin{cases} 1 & \text{falls} \quad \Delta f_k \leq 0 \\ \frac{1}{1+e^{\frac{\Delta f_k}{T(k)}}} & \text{falls} \quad \Delta f_k > 0 \end{cases} \tag{5.4}$$

$$T(k) = \frac{T_0}{\log(k)} \tag{5.5}$$

n ist dabei wie üblich die Dimension des Problems. Die Definition von $h_T(\Delta f_k)$ ist an Boender et. al [9] angelehnt, da weder in [19] noch in [20] eine Fallunterscheidung wie bei Gleichung (5.4) angegeben ist und somit auch Punkte, die eine Verbesserung darstellen, abgelehnt werden könnten. Da die Dichte $g_T(\Delta x)$ in diesem Fall nicht mehrdimensional ist, muß im Anschluß an die Bestimmung des Δx noch ein Punkt auf der gefundenen n-dimensionalen Sphäre bestimmt werden.

Bei Boltzmann Annealing ist die Bedingung (5.1) erfüllt. Diese – hier unbewiesene – Behauptung, wird im folgenden ähnlich wie in Ingber et. al [19] veranschaulicht:
Laut Boender et. al [9], S. 836, ist (5.1) erfüllt, wenn folgende Gleichung für

alle x im Definitonsbereich gilt:

$$\prod_{k=0}^{\infty}(1-g_T(x)) = 0 \tag{5.6}$$

Lemma 5.3 *Sei* $0 < z_k < 1$ *für alle* k*, dann gilt:*

$$\prod_{k=1}^{\infty}(1-z_k) = 0 \quad \Longleftrightarrow \quad \sum_{k=1}^{\infty} z_k = \infty$$

Beweis: Falls die Folge (z_k) keine Nullfolge ist, ist der Satz klar. Sei also (z_k) eine Nullfolge, dann gilt:

$$\begin{aligned} -\frac{\log(1-z)}{z} = -\frac{\log(1-z)-\log(1)}{z} &\overset{z\to 0}{\longrightarrow} -\log'(y)\Big|_{y=1} = -1 \\ \Longrightarrow \quad \frac{z}{2} < -\log(1-z) < \frac{3z}{2} &\qquad \text{für genügend kleine } z. \end{aligned}$$

Daraus folgt:

$$\begin{aligned} & \lim_{K\to\infty} \sum_{k=0}^{K} z_k = \infty \\ \Longleftrightarrow \quad & \lim_{K\to\infty} -\sum_{k=0}^{K} \log(1-z_k) = \infty \\ \Longleftrightarrow \quad & \lim_{K\to\infty} \log\left(\prod_{k=0}^{K}(1-z_k)\right) = -\infty \end{aligned}$$

Durch Anwendung der Exponentialfunktion läßt sich der Beweis vollenden. ■

Auf Grund der Wahl der Wahrscheinlichkeitsdichte (5.3) und der Temperaturfunktion (5.5) erhält man folgende Ungleichungskette, wenn der Definitionsbereich D beschränkt und T_0 im Verhältnis zu $\Delta(D)$ (siehe Definition 4.10) „groß genug“ ist:

$$\sum_{k=1}^{\infty} g_T(\Delta x) \;\geq\; \sum_{k=1}^{\infty} e^{-\log(k)} \;=\; \sum_{k=1}^{\infty}\frac{1}{k} \;=\; \infty$$

und wegen des Lemmas gilt somit die Gleichung (5.6) für alle Δx. Wenn man mit g die Punkte auf der Sphäre geeignet verteilt (möglichst gleichmäßig) wählt, so gilt Gleichung (5.6) für alle x im Definitionsbereich.

Die Funktion $T(k)$ bei Boltzmann Annealing fällt sehr langsam wenn k wächst, weshalb auch der Algorithmus nicht schnell konvergiert. Auch ist es nur schwer möglich, koordinatenbezogene Unterschiede der Testfunktion zu berücksichtigen.

Bei **Very Fast Annealing** (Ingber et. al [19]) ist $g_T(x)$ koordinatenweise definiert, wodurch es auf einfache Art möglich ist, koordinatenweise Beschränkungen des Definitionsbereiches einzuführen als auch jeder Koordinate eine eigene Temperaturfunktion $T_i(k)$ zuzuordnen. Ein weiterer Vorteil ist es, daß die angegebene Temperaturfunktion sehr schnell fällt und die Eigenschaft (5.1) trotzdem erhalten bleibt:

$$\begin{aligned} g_T(y) &= \prod_{i=1}^{n} \left(2(|y_i| + T_i) \log\left(1 + \frac{1}{T_i}\right)\right)^{-1} && (5.7) \\ T_i(k) &= T_{0,i}\, e^{-c_i k^{\frac{1}{n}}} && (5.8) \end{aligned}$$

Dabei ist $T_{0,i}$ eine fest zu wählende Anfangstemperatur, und c_i ist ein Steuerparameter. h_T bleibt wie in Gleichung (5.4). Gilt für den Definitionsbereich:

$$x_i \in [A_i, B_i], \qquad \text{für alle} \quad i \tag{5.9}$$

dann wird x_{neu} folgendermaßen berechnet:

$$x_{\text{neu}} = x_{k,i} + y_i(B_i - A_i)$$

Da x_{neu} ebenfalls im Definitionsbereich liegen soll, werden solange y_i berechnet, bis Bedingung (5.9) erfüllt ist.

Die y_i erhält man mit einer, über dem Intervall $[0,1]$ gleichverteilten Zufallsvariable u_i folgendermaßen:

$$y_i = \operatorname{sgn}\left(u_i - \frac{1}{2}\right)\left(\left(1 + \frac{1}{T_i}\right)^{|2u_i - 1|} - 1\right) T_i$$

und sie sollen nach [19] gemäß Gleichung (5.7) verteilt sein.

Da verschiedene Koordinaten auch verschieden schnell konvergieren können, kann es sinnvoll sein, die T_i von Zeit zu Zeit anzupassen um nicht zu lange bei einem lokalen Minimum hängenzubleiben. Sollte das etwa an den Zeitpunkten t_j passieren, so geschieht das in INGBER et. al [19] auf die Art, daß der Zeitparameter k aktualisiert wird:

$$k_i \;=\; c_i^{-n} \left(\log \left(\frac{T_{0,i}}{T_i \left(\frac{s_{\max}}{s_i} \right)} \right) \right)^n$$

Dabei gilt:

$$s_i = \left| \frac{f^*_{t_{j-1}} - f^*_{t_j}}{x_{t_{j-1},i} - x_{t_j,i}} \right| \qquad \text{und} \qquad s_{\max} = \max(s_i)\,, \qquad i = 1, \ldots, n$$

und f^*_t ist der bis zum Zeitpunkt t beste gefundene Funktionswert.

ALUFFI-PENTINI et al. [4], [5] berechnen Δx_k, indem sie eine geeignete Diskretisierung der stochastischen Differentialgleichung wählen:

$$\mathrm{d}\xi \;=\; -\nabla f(x)\;\mathrm{d}\,T + \varepsilon(T)\,\mathrm{d}\omega$$

Es wird also ein Schritt der Methode des steilsten Abstiegs mit einem Wiener Prozess $\mathrm{d}\omega$ überlagert. Für die Implementierung wird der Gradient $\nabla f(x)$ allerdings durch einen gut gewählten Differenzenquotionen ersetzt. Ein Schritt wird dann nicht akzeptiert, wenn sich der Funktionswert zu stark verschlechtert. Mehr Details sind in Abschnitt 6.2.5 beziehungsweise in den oben zitierten Arbeiten zu finden.

In BOENDER et. al [9] sind noch weitere Abarten von Simulated Annealing angegeben, welche die Bedingung (5.1) aber meist nicht erfüllen.

In Zusammenhang mit der begrenzten Rechengenauigkeit von Computern, bietet es sich an, als Abbruchkriterium die Temperaturfunktion $T(k)$ heranzuziehen (INGBER [21]). Ist $\|T(k)\|$ kleiner als die Maschinengenauigkeit oder ein anderer, vom Benutzer festgelegter Wert, so hält das Programm an. Für die Norm kann man etwa die Maximumsnorm wählen.

Getestete Software, die auf diese Weise zu minimieren verucht, ist in den Abschnitten 6.2.6 und 6.2.5 beschrieben.

5.5 Clustering

Die primitivste Methode, um eine Funktion zu minimieren ist es, einfach nur die Funktionswerte an zufällig ausgewählten Punkten zu bestimmen. Es ist auch nicht viel besser, wenn man an jedem der Punkte eine lokale Suche startet, da so ein und dasselbe Minimum immer wieder gefunden und somit viel Rechenzeit verschwendet wird. Die in diesem Abschnitt vorgestellten Verfahren stellen einen Kompromiß zwischen den beiden Vorgangsweisen dar. Um nicht zu oft mit einer lokalen Suche in einem bestimmten Minimum zu landen, setzt man den lokalen Suchalgorithmus nur selten ein. Jene Punkte, von denen keine lokale Suche gestartet wurde, organisiert man in Cluster (engl.: *Haufen*) und hofft, daß alle Punkte eines Clusters zum Einzugsbereich ein und desselben Minimums (eigentlich Minimalpunktes) gehören. Das Ziel ist es natürlich, so alle und damit auch die globalen Minima zu finden.

Bei derartigen Verfahren kommt es im Wesentlichen auf drei Punkte an, welche wiederholt durchlaufen werden:

1. Eine geeignete Menge von Zufallspunkten.
2. Ein Kriterium, um die Startpunkte für die lokale Suche zu bestimmen.
3. Eine Methode, um die Punkte den Clustern zuzuordnen.

Zuerst werden gleichverteilte Zufallspunkte im Definitionsbereich bestimmt. In Boender et al. [9] werden zwei Methoden angegeben, um von dieser Stichprobe ausgehend eine Punkt 1 entsprechende Menge zu erhalten, und der zweiten Methode der Vorzug gegeben:

Konzentration: Die Zufallspunkte werden transformiert, indem auf jeden einige wenige Schritte der Methode des steilsten Abstiegs angewandt werden. (Bei Boender et al. [8] werden nur die besseren $(1-\gamma)$ Prozent mit einem einzigen Schritt transformiert.)

Reduktion: Nur die besten Punkte der Stichprobe werden weiter verwendet.

Alle im folgenden vorgestellte Varianten wählen derart wiederholt eine feste Anzahl N von Zufallspunkten aus.

Um den Punkten 2 und 3 gerecht zu werden, wurden einige Methoden entwickelt:

5.5.1 Density Clustering

Die Idee hierbei ist es, daß sich die Funktion in der Nähe eines lokalen Minimums x^* gut durch die Hessematrix $H(x^*)$ beschreiben läßt. Entsprechend versucht man die Cluster als Hyper-Ellipsoide um die lokalen Minima herum aufzubauen. Dabei ist X^* die Menge der bisher gefundenen lokalen Minima:

Algorithmus 5.4 (Boender et al. [9])

Schritt 0: *Setze $k = 0$ und $X^* = \emptyset$.*

Schritt 1: *Bestimme die N Zufallspunkte $x_{(k-1)N+1}, \ldots, x_{kN}$ durch die oben erläuterte Methode der Reduktion und setze $i = 1$, $j = 1$.*

Schritt 2: *Sollten alle Punkte bereits einem Cluster zugeordnet sein, so ist mit Schritt 4 fortzufahren. Ist $j \leq |X^*|$, so wählt man das j-te Minimum x_j^* von X^* als Ausgangspunkt und fährt mit Schritt 3 fort. Ist $j < |X^*|$, so wird von dem noch keinem Cluster angehörenden Punkt mit dem kleinsten Funktionswert $\overline{x}$ eine lokale Suche gestartet. Ist das Ergebnis der Suche x^* bereits in X^* enthalten, wird $\overline{x}$ dem entsprechenden Cluster zugeordnet und bei Schritt 2 fortgesetzt. Gilt $x^* \notin X^*$, dann wird dieses Minimum als x_j^* X^* hinzugefügt, $\overline{x}$ wird im zu x_j^* gehörenden Cluster aufgenommen und mit x_j^* als Ausgangspunkt wird bei Schritt 3 fortgesetzt.*

Schritt 3: *Alle Punkte, welche bislang noch keinem Cluster zugewiesen wurden und die weniger als $r_i(x_j^*)$ von x_j^* entfernt sind, werden dem entsprechenden*

Cluster zugeordnet. Sollte kein Punkt zugeordnet worden sein, so wird j um eins erhöht und bei Schritt 2 forgesetzt. Ansonsten wird i um eins erhöht und Schritt 3 wiederholt.

Schritt 4: *Es wird k um eins erhöht und bei Schritt 1 fortgesetzt.*

r ist gegeben durch:

$$r_i(x) = \frac{1}{\sqrt{\pi}} \left(i\,\Gamma\left(1 + \frac{n}{2}\right) m(D)\, \frac{\xi \ln(kN)}{kN} \sqrt{-\det(H(x))} \right)^{\frac{1}{n}}$$

Dabei ist $m(D)$ das Lebesgue-Maß des Definitionsbereichs, ξ eine wählbare, positive Konstante und n wie üblich die Dimension des Problems.

Diese Methode funktioniert erwartungsgemäß dann gut, wenn die Approximation durch quadratische Funktionen auch bei großem Maßstab gut ist. Sollte diese Annahme nicht zutreffen, so sollte man die Verfahren aus den folgenden Abschnitten (5.5.2, 5.5.3 beziehungsweise 5.5.4) verwenden. Eine Schwachstelle des angegebenen Algorithmus scheint mir zu sein, daß in Schritt 2 ein Punkt x einem Cluster zugeordnet werden kann, obwohl x einen kleineren Funktionswert besitzt als das zum Cluster gehörende Minimum. In diesem Fall sollte x jedenfalls Ausgangspunkt einer lokalen Suche sein.

5.5.2 Single Linkage

Diese Variante geht von der Annahme aus, daß ein Punkt, welcher nahe einem Cluster liegt (das heißt, nahe einem zu diesem Cluster gehörenden Punkt), ebenfalls diesem Cluster angehören soll. Dadurch können die Cluster sich der Gestalt der Funktion besser anpassen.

Algorithmus 5.5 (Boender et al. [9])

Schritt 0, 1, 2 und 4: *Wie in Algorithmus 5.4.*

Schritt 3: *In diesem Schritt wird wieder der zum Minimum x_j^* gehörende Cluster mit bisher noch keinem Cluster zugeordneten Zufallspunkten erweitert, indem alle derartigen Punkte, die einen Abstand von weniger als r_k von einem Punkt des Clusters haben, hinzugefügt werden. Danach wird j um eins erhöht und bei Schritt 2 fortgesetzt.*

Dabei gilt für die im zu betrachtenden Bereich liegende Funktion r:

$$r_k = \frac{1}{\sqrt{\pi}} \left(\Gamma \left(1 + \frac{n}{2} \right) m(D) \frac{\xi \ln(kN)}{kN} \right)^{\frac{1}{n}} \tag{5.10}$$

Schritt 2 scheint in [9] nicht ganz korrekt zu sein, weil zu stark verkürzt wiedergegeben.

Wählt man $\xi > 2$, so geht die Anzahl der lokalen Suchen pro Iterationsschritt mit wachsendem k gegen 0. Ist $\xi > 4$, dann ist die Anzahl aller lokalen Suchen fast sicher endlich ([9], S. 834).

Wie in Abschnitt 5.5.1 wird auch in diesem Algorithmus nicht überprüft, ob ein Punkt einen kleineren Funktionswert als das Minimum hat, bevor er dem entsprechenden Cluster beigefügt wird. Bei dieser Methode wird für jeden Punkt nur einmal überprüft, ob er als Startpunkt für eine lokale Suche dienen soll. Wird ein Punkt abgelehnt, so ist diese Entscheidung endgültig. Erwähnenswert erscheint mir noch, daß die Bedingung 5.1 nicht erfüllt ist. Abhilfe schafft das im folgenden Abschnitt vorgestelle Verfahren.

Eine Variante von **Single Linkage** wurde in dem in Abschnitt 6.2.7 beschriebenen Programm implementiert.

5.5.3 Multi Level Single Linkage

In diesem Fall werden keine Cluster gebildet, sondern es wird einfach von jedem Punkt eine lokale Suche gestartet, außer innerhalb eines bestimmten Abstandes liegt ein weiterer Punkt mit einem geringeren Funktionswert. Im Gegensatz zu den vorangegangenen Algorithmen werden hier die Zufallspunkte nicht modifiziert.

Algorithmus 5.6 (Boender **et al.** [**9**])

Schritt 0: *Setzte* $k = 1$ *und* $X^* = \emptyset$.

Schritt 1: N *über* D *gleichverteilte Zufallspunkte* $x_{(k-1)N+1}, \ldots, x_{kN}$ *werden bestimmt und* $i = 1$ *gesetzt.*

Schritt 2: *Falls es einen Index* j *gibt, sodaß* $f(x_j) < f(x_i)$ *und* $\|x_j - x_i\| < r_k$ *gilt, so wird mit Schritt 3 fortgesetzt. Ansonsten wird bei* x_i *eine lokale Suche gestartet, das gefundene Minimum, so nicht bereits früher geschehen, der Menge* X^* *hinzugefügt und ebenfalls bei Schritt 3 fortgesetzt.*

Schritt 3: *Der Wert von* i *wird um eins erhöht. Gilt nun* $i \leq kN$*, so ist bei Schritt 2 fortzufahren und sonst mit um eins erhöhtem* k *bei Schritt 1.*

Die kritische Distanz r_k *ist wie in Gleichung* (5.10) *definiert.*

Auch dieser Algorithmus scheint mir nicht ganz korrekt in [9] wiedergegeben zu sein, denn eine lokale Suche kann so bei einem einzigen Punkt mehrmals beginnen. Um das zu vermeiden kann man sich die entprechenden Punkte merken und fernhin als Kandidaten für Startpunkte ausschließen. Eine andere Möglichkeit mit weitreichenderen Konsequenzen wäre es, die Anweisung $i = 1$ in Schritt 0 und nicht in Schritt 1 durchzuführen. Dadurch bekommt jeder Punkt nur einmal die Möglichkeit als Startpunkt ausgewählt zu werden (da die r_k fallen, bekommt ein Punkt in obigem Algorithmus bei jeder Erhöhung von k erneut eine Chance), ähnlich wie bei dem in Abschnitt 5.5.2 beschriebenen Methode.

Wie in Abschnitt 5.5.2 strebt auch hier für $\xi < 2$ die Wahrscheinlichkeit, daß in der k-ten Iteration eine lokale Suche gestartet wird gegen 0, und auch für $\xi < 4$ ist die Anzahl aller lokalen Suchen fast sicher endlich. Weiters gilt für diesen Algorithmus, daß fast sicher jedes lokale Minimum gefunden wird, was die Bedingung (5.1) einschließt.

5.5.4 Simple Linkage

Dieses in Locatelli & Schoen [26] vorgestellte Verfahren soll ähnlich gute Eigenschaften wie **Multi Level Single Linkage** (Abschnitt 5.5.3) aufweisen. Die Handhabung der Zufallspunkte ist aber einfacher, da immer nur ein Zufallspunkt (Gleichverteilung über dem Definitionsbereich) bestimmt wird und dann sofort – ähnlich wie in Abschnitt 5.5.3 – entschieden wird, ob von ihm eine lokale Suche startet oder nicht. Auch kommt der Punkt nie mehr als Startpunkt in Frage, wodurch keine Schleife über alle Punkte mehr notwendig ist:

Algorithmus 5.7

Schritt 0: *Setze* $k = 1$ *und* $X^* = \emptyset$.

Schritt 1: *Ein Zufallspunkt* x_k *(gleichverteilt über* D*) wird bestimmt. In* x_k *wird genau dann eine lokale Suche gestartet, wenn es einen Index* j *gibt, sodaß* $f(x_j) < f(x_k)$ *und* $\|x_j - x_k\| < r_k$ *gilt. Gegegenfalls wird das gefundene Minimum der Menge* X^* *hinzugefügt, der Wert von* k *um eins erhöht und bei Schritt 1 fortgesetzt.*

r_k *ist dabei ähnlich wie* (5.10) *definiert:*

$$r_k = \frac{1}{\sqrt{\pi}} \left(\Gamma\left(1 + \frac{n}{2}\right) \frac{\xi \ln(k)}{k} \right)^{\frac{1}{n}}$$

Ist $\xi > 0$, so strebt die Wahrscheinlichkeit für eine lokale Suche mit wachsendem k gegen 0. Gilt $\xi > 1$ und läßt man keine Punkte, die einen gewissen Abstand zum Rand unterschreiten, als Startpunkte für die lokale Suche zu, dann ist die Anzahl aller Suchvorgänge fast sicher endlich. Der Algorithmus findet ein globales Minimum x^* (genauer: $x^* - \varepsilon$, $\varepsilon > 0$, beliebig) fast sicher nach endlich vielen Schritten, was wieder die Bedingung (5.1) impliziert. Weiters strebt die Startwahrscheinlichkeit für eine lokale Suche genau denn gegen 0, wenn

$$\lim_{k \to \infty} r_k k^{1/n} = \infty$$

gilt. Für Details siehe Locatelli & Schoen [26], S. 97.

5.5.5 Allgemeine Bemerkungen

Aus Punkt 2, Seite 61, folgt, daß so nur Funktionen behandelt werden können, bei denen eine lokale Suche möglich ist. Es sollte also der Einzugsbereich jedes lokalen Optimums ein Maß größer 0 besitzen.

Im Unterschied zu **evolutiven Algorithmen** oder **Simulated Annealing** wächst der Speicherbedarf bei auf Clustern basierenden Programmen kontinuierlich im Laufe der Zeit. Das liegt natürlich daran, daß alle Zufallspunkte für spätere Vergleiche gespeichert werden. Gerade bei hochdimensionalen Problemen, wo entsprechend viele Funktionsauswertungen erforderlich sind, kann das zu erheblichem Speicherbedarf führen. Auch im Falle von Problemen mit sehr vielen Minima sind eher andere Optimierungsverfahren zu empfehlen (Boender et al. [9]). Das ist auch leicht einzusehen, da bei derartigen Programmen ja versucht wird, zu jedem (lokalen) Minimum einen Cluster zu bilden, beziehungsweise jedes solche gefunden werden soll. Die Konsequenzen für den Speicherbedarf sind offensichtlich.

Obige Beschreibungen lassen offen, wie überprüft wird, ob ein mit einer lokalen Suche gefundenes Minimum x^* bereits in X^* enthalten ist. Will man *alle* globalen Minima finden, so müssen die Minimalpunkte verglichen werden und im anderen Fall deren Funktionswerte. Diese Entscheidung liegt aber wieder beim Benutzter, kann also hier nicht getroffen werden. Gegebenenfalls kann sich so die Anzahl der Elemente von X^* stark reduzieren, wenn man die zweite Variante wählen kann.

Eine Variante, die sich am ehesten den Cluster-Verfahren (Abschnitt 5.5.4) zuorden läßt, ist in Mockus [30] beschrieben. Es werden zwar keine lokalen Suchen gestartet, dafür werden aber auch keine wirklichen Zufallspunkte berechnet, sondern für diesen Zweck eine a priori Verteilung eingesetzt, die dem Problem angepaßt werden kann.

5.6 Hybrid-Verfahren

Es scheint nicht unbedingt sinnvoll, die hier vorgestellten Methoden so klar zu trennen, wie es bisher geschehen ist. Selbstverständlich ist es möglich, die Verfahren nach belieben zu vermischen und zu rekombinieren. Davis [12] empfiehlt, für jedes reale Problem, den bereits verwendeten Algorithmus mit einem evolutiven Algorithmus zu verbinden. Dabei sollen wenn möglich die bereits verwendete Codierungsart und die Vorteile des Algorithmus erhalten bleiben. So kann man zum Beispiel eine oder mehrere Lösungen des alten Algorithmus in die erste Population eines evolutiven Algorithmus aufnehmen oder alte Transformationsarten übernehmen. Allerdings muß man auch die evolutiven Operatoren den aktuellen Gegebenheiten anpassen. Details und Beispiele sind in [12] zu finden.

Yuret [40] beschreibt eine Variante von evolutiven Algorithmen. Dabei wird zwischen *alten* und *jungen* Populationen unterschieden. Die junge Population ist gewissermaßen die aktive, die sich ständig verändert und eine fixe Populationsgröße besitzt, während die alte nur erweitert wird, indem ein Punkt, der sich lange genug in der jungen Population gehalten hat, aufgenommen wird. Die Idee ist es, daß in der alten Population Punkte gespeichert werden, die Minima entsprechen und dann verhindert wird, daß Punkte der jungen Population zu nahe an diesen Minima zu liegen kommen, um den Suchraum besser abzudecken. Selbstverständlich stehen die alten Punkte auch für Crossover zur Verfügung.

Weiters wird versucht, neue Punkte in möglichst noch unerforschten Regionen zu erzeugen und gute Suchrichtungen zu finden.

Kapitel 6

Tests

Die Tests, die im Rahmen dieser Arbeit durchgeführt wurden, beschränken sich auf Funktionen, welche entweder keine oder nur koordinatenweise Nebenbedingungen beinhalten, da diese von allen Programmen leicht zu behandeln sind.

Um verschiedenste Programme einheitlich testen zu können, schrieb ich das C-Programm `ej_eval` (siehe Abschnitt A), welches alle wesentlichen Informationen, wie Definitonsbereich, Problemdimension oder einen Funktionswert an einer bestimmten Stelle zur Verfügung stellt. Für die Funktionswerte übergibt ein Programm den Vektor x, an dem ausgewertet werden soll und `ej_eval` liefert den dazugehörigen Funktionswert zurück. Um auch Programme, welche überhaupt keine Nebenbedingungen erlauben, testen zu können, ist die in Abschnitt 4.3.3 beschriebene Methode bereits in `ej_eval` integriert.

An den einzelnen Programmen wurden so wenig Änderungen wie möglich vorgenommen. Benötigte ein Programm eine Datei, welche Parameter enthielt, so wurden diese entweder direkt im Programm gesetzt oder von `ej_eval` bereitgestellt. Parameter wurden entsprechend den Empfehlungen der zugehörigen Literatur gewählt oder auf ihrem voreingestellten Wert belassen. Das führt natürlich oft zu schlechteren Ergebnissen als in der Literatur angegeben, da für die Veröffentlichungen meist viel Arbeit in die Auswahl guter Parameter investiert wurde. Auf

Grund all desssen mußte nur mehr `ej_eval` den verschiedenen Testproblemen angepaßt werden.

Als Grundlage für die Tests der Programme dienten Dateien mit je 20 000 Funktionsauswertungen, soweit dies möglich war. Die Dateien wurden erzeugt indem entsprechend viele Funktionswerte je Programm und Testfunktion der Reihe nach abgespeichert wurden. Bei einigen Programmen wurden dazu interne Abbruchkriterien umgangen und mehrere Versionen getestet (mit und ohne Abbruchkriterium).

Compiliert wurden alle verwendeten Programme auf einer Digital AlphaStation 200 4/233 MHz mit 64 MB Memory unter Digital Unix 4.0B mit folgenden Versionen der Compiler:

Gnu C: 2.7.2.2

f77: Digital Fortran 77

f90: Digital Fortran 90 V4.0-1

6.1 Testfunktionen

Manche Angaben wie Definitionsbereich oder die Werte mancher Parameter sind in der Literatur nicht einheitlich. Auch die Funktionsdefinitionen selbst weichen in einige Punkten voneinander ab. Für die folgenden Abbildungen gilt, daß Punkte mit kleineren Funktionswerten heller dargestellt sind.

F1. Sphäre

$$f(x) = \sum_{i=1}^{n} x_i^2$$

$$-5.12 \le x_i \le 5.12, \qquad i = 1, \ldots, n$$

$$f^* = f(0, \ldots, 0) = 0$$

De Jong – Testfunktion 1: De Jong [15], Testreihe 1-1: Storn & Price [37], Desai & Patil [13], Michalewicz [28], Pohlheim [35], Bäck [6]. Diese Funktion darf für ein ernstzunehmendes Programm keine Schwierigkeiten darstellen. Sie ist stetig, unimodal, konvex und symmetrisch und gehört zu den einfachsten Testfunktionen, welche höchstens ein Kriterium für die allgemeinen Konvergenzeigenschaften von Optimierungsprogrammen darstellen.

Für die Tests wurde die Funktion mit $n = 3$ verwendet.

F2. Rosenbrock

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{n-1} \left(100(x_{i+1}^2 - x_i)^2 + (1 - x_i)^2\right) \\
-2.048 \;\leq\; x_i &\;\leq\; 2.048 \qquad i = 1, \ldots, n
\end{aligned}
$$

$$
f^* \;=\; f(1, \ldots, 1) \;=\; 0
$$

Abbildung 6.1: Rosenbrock mit $n = 2$

De Jong – Funktion 2: De Jong [15], Testreihe 1-2: Storn et. al [37], 2nd ICEO [2], Desai & Patil [13], Michalewicz [28], Pohlheim [35], Moré et. al [31], Bäck [6]. Die Rosenbrock-Funktion gilt im allgemeinen als schwieriges Optimierungsproblem, obwohl sie im 2-dimensionalen Fall nur zwei Parameter benötigt und nur ein Minimum aufweist. Das läßt sich mit dem engen, parabelförmig gebogenen Tal mit den steilen Wänden und einem scharfen Ende erklären. Viele Programme haben so die Schwierigkeit, eine geeignete Suchrichtung zu finden.

Für die Tests wurde die Funktion mit $n = 2$, $n = 5$ und $n = 10$ verwendet.

F3. Step

$$f(x) \;=\; 6n + \sum_{i=1}^{n} \lfloor x_i \rfloor$$

$$-5.12 \;\leq\; x_i \;\leq\; 5.12 \qquad i = 1, \ldots, n$$

$$f^* \;=\; f(\,[-5.12, -5), \ldots, [-5.12, -5)\,) \;=\; 0$$

De Jong – Funktion 3 (modifiziert): De Jong [15], Testreihe 1-3: Storn & Price [37], Desai & Patil [13], Michalewicz [28], Pohlheim [35]. Diese Funktion ist unstetig und enthält viele Plateaus, welche Schwierigkeiten beim Optimieren bereiten können, da dort ein Programm keine gute Suchrichtung herausfinden kann.

Für die Tests wurde die Funktion mit $n = 5$ verwendet.

F4. Hyper-Ellipsoid

$$f(x) = \sum_{i=1}^{n} i^2 x_i^2$$

$$-1 \leq x_i \leq 1 \qquad i = 1, \ldots, n$$

$$f^*(0, \ldots, 0) = 0$$

Abbildung 6.2: 2-dimensionaler Schnitt durch Hyper-Ellipsoid mit $n = 30$

Testreihe 2-1: Storn & Price [37], Pohlheim [35]. Diese Funktion ist eine Variante der Sphäre. Auch sie ist unimodal und konvex, besitzt aber sehr langgezogene, flache Täler entlang der Kooridinatenachsen. Dort ist es wieder schwer, eine gute Suchrichtung zu finden.

Für die Tests wurde $n = 30$ gesetzt.

F5. Neumaier Funktion Nr. 3

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \\
-n^2 \; \leq & \; x_i \; \leq \; n^2 \qquad i = 1, \ldots, n \\
f^*(x^*) &= -\frac{n(n+4)(n-1)}{6}, \qquad x_i^* = i(n+1-i)
\end{aligned}
$$

Abbildung 6.3: Neumaier Funktion Nr. 3 mit $n = 2$

NEUMAIER [33]. Diese Funktion ist konvex und besitzt somit nur einen Minimalpunkt. Die Funktion sieht etwa wie ein gedrehter Hyper-Ellipsoid aus. In höheren Dimensionen werden die Täler zunehmend langgezogener und schmäler, ähnlich dem in Abbildung 6.2 dargestellten. Durch eine zusätzliche Drehung wird die Funktion schwerer zu minimieren, da sie nicht mehr separabel ist. Dieser Umstand dürfte sich vor allem auf das Verhalten von evolutiven Algorithmen auswirken da sie diese Eigenschaft, wegen ihrer Darstellungsweise der Punkte, besonders häufig ausnutzen.

Für die Tests wurden folgende Parameter verwendet:

n	10	15	20	25	30
f^*	-210	-665	-1520	-2900	-4930

F6. Schaffer Funktion Nr. 2

$$f(x) \;=\; \left(x_1^2 + x_2^2\right)^{0.25} \left(\left(50\left(x_1^2 + x_2^2\right)^{0.1}\right)^2 + 1\right)$$

$$-100 \;\leq\; x_i \;\leq\; 100 \qquad i = 1, 2$$

$$f^*(0,0) \;=\; 0$$

Abbildung 6.4: Schaffer Funktion Nr. 2

Michalewicz [28]. Auch diese Funktion besitzt nur ein einziges Minimum.

F7. Shekel-N

$$f(x) = -\sum_{j=1}^{N} \frac{1}{\sum_{i=1}^{4}(x_i - a_{ij})^2 + c_j}$$

$$0 \leq x_i \leq 10 \qquad i = 1, \ldots, 4$$

j	a_{1j}	a_{2j}	a_{3j}	a_{4j}	c_j
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.3
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

$$f^*(4,4,4,4) = \begin{cases} -10.15320 & \text{für} \quad N = 5 \\ -10.40294 & \text{für} \quad N = 7 \\ -10.53641 & \text{für} \quad N = 10 \end{cases}$$

Dixon & Szegö [14], Törn [39], Michalewicz [28], Bäck [6]. Für die Tests wurde die Funktion mit $N = 5$, $N = 7$ und $N = 10$ verwendet.

F8. Hartman-Funktion

$$f(x) = -\sum_{j=1}^{4} c_i \exp\left(-\sum_{i=1}^{n} a_{ij}(x_i - p_{ij})^2\right)$$

$$0 \leq x_i \leq 1 \qquad i = 1, \ldots, n$$

$$c = (1.0 \quad 1.2 \quad 3.0 \quad 3.2)$$

$n = 3$:

$$(a_{ij}) = \begin{pmatrix} 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \\ 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \end{pmatrix} \qquad (p_{ij}) = \begin{pmatrix} 0.36890 & 0.11700 & 0.26730 \\ 0.46990 & 0.43870 & 0.74700 \\ 0.10910 & 0.87320 & 0.55470 \\ 0.03815 & 0.57430 & 0.88280 \end{pmatrix}$$

$$f^*(0.1, 0.55592, 0.85218) = -3.86278215$$

$n = 6$:

$$(a_{ij}) = \begin{pmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{pmatrix}$$

$$(p_{ij}) = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

$$f^*(0.20169, 0.15001, 0.47687, 0.2753, 0.31165, 0.65730) = -3.32236801$$

Dixon & Szegö [14]. p_i ist dabei die ungefähre Position des i-ten lokalen Minimums, a_i ist proportional zu den Werten der Hessematrix am i-ten lokalen Minimum und $c_i > 0$ entspricht dem Funktionswert dieses lokalen Minimums.

Für die Tests wurde die Funktion mit $n = 3$ und $n = 6$ verwendet.

F9. Goldstein-Price-Funktion

$$\begin{aligned} f(x) &= \left(1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right) \cdot \\ & \quad \left(30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right) \end{aligned}$$

$$-2 \ \leq \ x_i \ \leq \ 2 \qquad i = 1, 2$$

$$f^*(0, -1) \ = \ 3$$

Abbildung 6.5: Goldstein-Price-Funktion

Dixon & Szegö [14], Desai & Patil [13], Michalewicz [28], Pohlheim [35]. Diese multimodale Funktion weist in der Nähe des globalen Minimums sehr große Unterschiede in den Funktionswerten auf.

F10. Branin-Funktion

$$f(x) \ = \ a\left(x_2 - bx_1^2 + cx_1 - d\right)^2 + e(1 - f)\cos(x_1) + e$$

$$a = 1\,, \quad b = \frac{5.1}{4\pi^2}\,, \quad c = \frac{5}{\pi}\,, \quad d = 6\,, \quad e = 10\,, \quad f = \frac{1}{8\pi}$$

$$(x_1,\, x_2) \;\in\; ([-5, 10]\,, [0, 15])$$

$$f^*(-\pi, 12.275) \;=\; f^*(\pi, 2.275) \;=\; f^*(9.42478, 2.475) \;=\; 0.3978873$$

Abbildung 6.6: Branin-Funktion

Dixon & Szegö [14], Michalewicz [28], Pohlheim [35].

F11. Six-hump camel back

$$f(x) \;=\; \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + 4\left(x_2^2 - 1\right) x_2^2$$

$$(x_1,\, x_2) \;\in\; \left([-3, 3]\,, [-2, 2]\right)$$

$$f^* \;=\; f(x) = f(-x) \;=\; -1.0316285\,, \qquad x = (0.089842, -0.712656)$$

Abbildung 6.7: Six-hump camel back Fläche und Konturen

Dixon & Szegö [14], Storn & Price [37], Michalewicz [28], Pohlheim [35]. Diese Funktion besitzt im Definitonsbereich sechs lokale Minimalpunkte, von denen zwei global sind.

F12. Shubert-Funktion

$$
\begin{aligned}
f(x) &= \prod_{i=1}^{n} \left(\sum_{j=1}^{5} j \cos((j+1)x_i + j) \right) \\
-10 \ \leq \ x_i &\leq \ 10 \qquad i = 1, \ldots, n \\
f^*(x^*) &= -186.7309088 \qquad (n = 2)
\end{aligned}
$$

$$
\begin{aligned}
x^* \quad = \quad & \begin{pmatrix} -7.08351 \\ 4.85806 \end{pmatrix}, \begin{pmatrix} 5.48286 \\ 4.85806 \end{pmatrix}, \begin{pmatrix} 4.85806 \\ -7.08351 \end{pmatrix}, \begin{pmatrix} 4.85806 \\ 5.48286 \end{pmatrix}, \\
& \begin{pmatrix} -7.08351 \\ -7.70831 \end{pmatrix}, \begin{pmatrix} -7.70831 \\ -7.08351 \end{pmatrix}, \begin{pmatrix} -1.42512 \\ -0.80032 \end{pmatrix}, \begin{pmatrix} -0.80032 \\ -1.42512 \end{pmatrix}, \\
& \begin{pmatrix} -1.42512 \\ -7.08351 \end{pmatrix}, \begin{pmatrix} -7.08351 \\ -1.42512 \end{pmatrix}, \begin{pmatrix} -7.70831 \\ 5.48286 \end{pmatrix}, \begin{pmatrix} 5.48286 \\ -7.70831 \end{pmatrix}, \\
& \begin{pmatrix} 5.48286 \\ 4.85806 \end{pmatrix}, \begin{pmatrix} 4.85806 \\ 5.48286 \end{pmatrix}, \begin{pmatrix} -7.70831 \\ -0.80032 \end{pmatrix}, \begin{pmatrix} -0.80032 \\ -7.70831 \end{pmatrix}, \\
& \begin{pmatrix} -1.42512 \\ -0.80032 \end{pmatrix}, \begin{pmatrix} -0.80032 \\ -1.42512 \end{pmatrix}
\end{aligned}
$$

Abbildung 6.8: Shubert - Funktion mit $n = 2$

Dixon & Szegö [14], Storn [37], Michalewicz [28]. Im 2-dimensionalen Fall besitzt diese Funktion im Definitionsbereich 760 lokale Minima, von denen 18 global sind.

Für die Tests wurde die Funktion mit $n = 2$ verwendet.

F13. Shekels Foxholes

$$f(x) = \frac{1}{0.002 + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{2} (x_j - a_{ij})^6}}$$

$$-65.536 \leq x_i \leq 65.536 \qquad i = 1,2$$

$$(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \ldots & 32 & 32 & 32 \end{pmatrix}$$

$$f^* = f(-32, -32) = 0.9980038$$

Abbildung 6.9: Shekels Foxholes

De Jong - Funktion 5: De Jong [15], Testreihe 1-5: Storn & Price [37], Michalewicz [28], Pohlheim [35].

F14. Modifizierte Shekels Foxholes

Diese Funktion war Testfunktion beim Second International Contest on Evolutionary Optimization [2] und lautet auf der angegebenen Web-Seite:

$$f(x) = -\sum_{i=1}^{m} \frac{1}{\|x - A(i)\|^2 + c(i)}$$
$$0 \leq x_i \leq 10 \qquad i = 1, \ldots n$$

Diese Funktion entspricht der dort verfügbaren `C`-Datei. Auf der Web-Seite selbst stand kein $c(i)$ im Nenner, was aber auf einen Druckfehler zurückzuführen sein dürfte. Die Werte der Vektoren $A(i)$ und c sind in der Datei `shekel.h`, welche in [2] zu finden ist, angegeben.

Für die Tests wurde $m = 30$, $n = 5$ und $n = 10$ verwendet.

F15. Coranas Parabel

$$f(x) = \sum_{i=1}^{4} \begin{cases} 0.15\,(z_i - 0.05 \cdot \mathrm{sgn}(z_i))^2\, d_i & \text{falls} \quad |x_i - z_i| < 0.05 \\ d_i x_i^2 & \text{sonst} \end{cases}$$
$$-1000 \leq x_i \leq 1000 \qquad i = 1, \ldots, 4$$
$$z_i = \left\lfloor \left| \frac{x_i}{0.2} \right| + 0.49999 \right\rfloor \cdot \mathrm{sgn}(x_i) \cdot 0.2$$
$$d = (1, 1000, 10, 100)$$
$$f^*(\,[-0.05, 0.05], \ldots, [-0.05, 0.05]\,) = 0$$

Corana [10], Testreihe 1-6: Storn & Price [37]. Diese Funktion besteht aus einer Parabel, welche Löcher enthält, die mit geringerer Distanz zum Ursprung immer tiefer werden. Die Schwierigkeit für ein Programm besteht darin, zu vermeiden in einem dieser Löcher hängenzubleiben.

F16. Griewanks Funktion

$$f(x) = \frac{1}{d} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$-k \;\leq\; x_i \;\leq\; k \qquad i = 1, \ldots, n$$

$$f^*(0, \ldots, 0) \;=\; 0$$

GRIENWANK [17], Testreihe 1-7, Testreihe 2-4: STORN & PRICE [37], DESAI & PATIL [13], BÄCK [6]. Diese Funktion ist wieder eine Modifikation der Sphäre durch eine periodische Funktion.

Für die Tests wurden die Parameter der Funktion folgendermaßen belegt:

$$n \;=\; 10, \qquad d \;=\; 4000, \qquad k \;=\; 400.$$

F17. Zimmermanns Problem

$$\begin{aligned} f(x) \;&=\; \max\{\;\; h_1(x), \quad p(h_2(x))\, \mathrm{sgn}(h_2(x)), \quad p(h_3(x))\, \mathrm{sgn}(h_3(x)), \\ &\qquad p(-x_1)\, \mathrm{sgn}(x_1), \quad p(-x_2)\, \mathrm{sgn}(x_2) \quad \} \end{aligned}$$

$$\begin{aligned} h_1(x) &= 9 - x_1 - x_2 \\ h_2(x) &= (x_1 - 3)^2 + (x_2 - 2)^2 - 16 \\ h_3(x) &= x_1 x_2 - 14 \\ p(\delta) &= 100(1 + \delta) \end{aligned}$$

$$0 \;\leq\; x_i \;\leq\; 100 \qquad i = 1, 2$$

$$f^*(7, 2) \;=\; 0$$

ZIMMERMANN [42], Testreihe 1-8; STORN & PRICE [37] – in dieser Arbeit hat sich ein Druckfehler eingeschlichen, den ich hoffentlich nicht wiederholt habe. Eine Besonderheit dieser Testfunktion ist es, daß sich das Minimum in einer Ecke des von den Nebenbedingungen h_1–h_3 aufgespannten Bereichs befindet.

F18. Katsuuras Funktion

Abbildung 6.10: Zimmermanns Problem

$$f(x) = \prod_{i=1}^{n} \left(1 + i \sum_{k=1}^{d} \text{abstoint} \left(2^k x_i \right) 2^{-k} \right)$$
$$-1000 \leq x_i \leq 1000 \qquad i = 1, \ldots, n$$
$$f^*(0, \ldots, 0) = 1$$

Dabei ist abstoint(x) der Abstand von x zur nächstgelegenen ganzen Zahl, wie in Abbildung 6.12 dargestellt ist. Die Funktion ist also stetig aber nicht differenzierbar.

Testreihe 2-2: Storn & Price [37].

Für die Tests wurde $n = 10$ und $d = 32$ verwendet.

Abbildung 6.11: Katsuuras Funktion mit $n = 2$

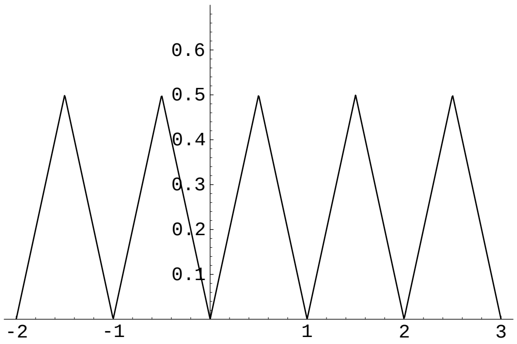

Abbildung 6.12: Darstellung der Funktion `abstoint`

F19. Rastrigins Funktion

$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - A\cos(2\pi x_i) \right)$$
$$-600 \leq x_i \leq 600 \qquad i = 1, \ldots, n$$
$$f^*(0, \ldots, 0) = 0$$

Testreihe 2-3: Storn & Price [37], Pohlheim [35], Desai & Patil, [13]. Diese Funktion besitzt im Definitionsbereich sehr viele lokale Minima und ist dement-

Abbildung 6.13: Globale und lokale Darstellung der Rastrigins Funktion mit Dimension $n = 2$

sprechend schwer zu minimieren. Es handelt sich hierbei wieder um eine Modifikation der Sphäre. Die Überlagerung mit dem Cosinus wirkt sich in der Nähe des globalen Minimums stärker aus als in entlegeneren Gegenden.

Für die Tests wurde $n = 20$ und $A = 10$ verwendet.

F20. Ackleys Funktion

$$f(x) = -20 \exp\left(-0.02\sqrt{n^{-1}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(n^{-1}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

$$-30 \;\leq\; x_i \;\leq\; 30 \qquad i = 1, \ldots, n$$

$$f^*(0, \ldots, 0) \;=\; 0$$

Abbildung 6.14: Ackleys Funktion mit $n = 2$

Testreihe 2-5: Storn & Price [37], Pohlheim [35]. Ackleys Funktion weist viele Minima im Definitionsbereich auf.

Für die Tests wurde $n = 30$ verwendet.

F21. Schaffer Funktion Nr. 1

$$f(x) \;=\; 0.5 + \frac{\left(\sin\sqrt{x_1^2+x_2^2}\right)^2 - 0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2}$$

$$-100 \;\leq\; x_i \;\leq\; 100 \qquad i=1,2$$

$$f^*(0,0) \;=\; 0$$

Abbildung 6.15: Schaffer Funktion Nr. 1

Michalewicz [28]. Die Minima dieser Funktion sind konzentrische Kreise um den Usprung. Die Schwankung der Funktionswerte nimmt mit abnehmendem Abstand zum Nullpunkt zu.

F22. Easom Funktion

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$$

$$-100 \le x_i \le 100 \qquad i=1,2$$

$$f^*(\pi,\pi) = -1$$

Abbildung 6.16: Lokale Darstellung der Easom Funktion

Michalewicz [28]. Diese Funktion ist eine periodische Funktion, mit einer Exponentialfunktion, welche mit der Entfernung zum globalen Minimum rasch abfällt. Die Schwankungen der Funktionswerte sind daher beim globalen Minimum besonders groß.

F23. Bohachevsky Funktion Nr. 1

$$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$

$$-50 \le x_i \le 50 \qquad i = 1, 2$$

$$f^*(0,0) = 0$$

Abbildung 6.17: Bohachevsky Funktion Nr. 1

MICHALEWICZ [28]. Auch diese Funktion ist eine Variante der Sphäre und besitzt viele Minima im Definitionsbereich.

F24. Bohachevsky Funktion Nr. 2

$$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$$

$$-50 \le x_i \le 50 \qquad i = 1, 2$$

$$f^*(0,0) = 0$$

Michalewicz [28]. Ähnlich wie die Funktion Bohachevsky Nr. 1, handelt es sich hier um eine mit einer periodischen Funktion überlagerte Sphäre.

F25. Colville Funktion

$$f(x) = 100\left(x_2 - x_1^2\right)^2 + (1-x_1)^2 + 90\left(x_4 - x_3^2\right)^2 + (1-x_3)^2 + 10.1\left((x_2-1)^2 + (x_4-1)^2\right) + 19.8(x_2-1)(x_4-1)$$

$$-10 \le x_i \le 10 \qquad i = 1, \ldots, 4$$

$$f^*(1, \ldots, 1) = 0$$

Michalewicz [28].

F26. Neumaier Funktion Nr. 2

$$f(x) = \sum_{k=1}^{n}\left(b_k - \sum_{i=1}^{n} x_i^k\right)^2$$

$$0 \le x_i \le n \qquad i = 1, \ldots, n$$

Neumaier [33]. Das globale Minimum dieser Funktion liegt immer bei 0, wenn man bei vorher bestimmtem Minimalpunkt x^* den Vektor b wie folgt wählt:

$$b_k = \sum_{i=1}^{n} (x_i^*)^k .$$

Abbildung 6.18: Neumaier Funktion Nr. 2 mit $n = 2$ und $x^* = (1, 2)$

Die Lösungen bestehen genau aus den Permutationen der Komponenten einer Lösung. Das Problem wird besonders schwierig, wenn mehrere Komponenten einer Lösung identisch sind, da dann der Rang der Hessematrix größer wird und die Funktion in einigen Richtungen sehr flach wird.

Für die Tests wurde $n = 4$ mit $b = (8, 18, 44, 114)$ und somit

$$f^*(1, 2, 2, 3) \;=\; 0$$

verwendet.

F27. Modifizierte Langerman Funktion

$$\begin{aligned} f(x) \;&=\; -\sum_{i=1}^{m} c(i) \left(e^{-\frac{1}{\pi}\|x-A(i)\|^2} \cos\left(\pi \|x - A(i)\|^2\right)\right) \\ 0 \;&\leq\; x_i \;\leq\; 10 \qquad i = 1, \ldots, n \end{aligned}$$

2^{nd} ICEO [2].

Die dort beschriebene Funktion gibt den Wert $m = 15$ an. Der dort ebenfalls erhältliche `C`-Code verwendet $m = 5$, ebenso die im Rahmen dieser Arbeit durchgeführten Tests.

Für die Tests wurde $n = 5$ und $n = 10$ verwendet.

F28. Epistatic Michalewicz Funktion

$$f(x) = -\sum_{i=1}^{n} \sin(y_i) \left(\sin\left(\frac{iy_i^2}{\pi}\right)\right)^{2m}$$

$$y_i = \begin{cases} x_i \cos\left(\frac{\pi}{6}\right) - x_{i+1} \sin\left(\frac{\pi}{6}\right) & \text{falls} \quad i \equiv 1 \bmod 2 \\ x_{i-1} \sin\left(\frac{\pi}{6}\right) + x_i \cos\left(\frac{\pi}{6}\right) & \text{falls} \quad i \equiv 0 \bmod 2 \quad \text{und} \quad i \neq n \\ y_n = x_n & \text{falls} \quad i \equiv 0 \bmod 2 \quad \text{und} \quad i = n \end{cases}$$

$$0 \leq x_i \leq \pi \qquad i = 1, \ldots, n$$

2nd ICEO [2]. Die dort gegebene Definition für y_n weicht von der für diese Arbeit verwendeteten ebendort zu findenden `C`-Datei ein wenig ab. Auch möchte ich darauf hinweisen, daß der `C`-Code bei einer `if`-Anweisung fehlerhaft sein dürfte.

Für die Tests wurde $m = 10$, $n = 5$ und $n = 10$ verwendet.

F29. Odd Square Funktion

$$f(x) = -e^{-\frac{\|x-a\|_\infty^2}{2\pi}} \cos\left(\pi \|x-a\|_\infty^2\right) \left(1 + c\,\frac{\|x-a\|_2^2}{\|x-a\|_\infty^2 + 0.01}\right)$$

$$a = (1, 1.3, 0.8, -0.4, -1.3, 1.6, -0.2, -0.6, 0.5, 1.4)$$

$$-5\pi \leq x_i \leq 5\pi \qquad i = 1, \ldots, n$$

2nd ICEO [2]. Die dort gegebene Definition der Zielfunktion und die ebenfalls dort zu findende – und für diese Arbeit verwendetete – `C`-Datei unterscheiden sich durch ein vermutlich falsches Vorzeichen. Vom Vektor a sind natürlich nur die jeweils ersten n Einträge relevant.

Für die Tests wurde $c = 0.2$, $n = 5$ und $n = 10$ verwendet.

F30. Chebychev-Polynom

Dieses Problem versucht die Koeffizienten a_i eines Polynoms $p_a(x)$ zu finden, sodaß folgende Bedingungen erfüllt sind:

$$\begin{aligned} p_a(\pm 1.2) &\geq d \\ |p_a(-1 + k\,0.02)| &\leq 1 \qquad k = 0, \ldots, 100 \end{aligned}$$

Um das zu erreichen wird in [2] eine Funktion $f(a)$ definiert, welche mit entsprechenden Straftermen versehen ist (vergleiche Gleichung (4.5) in Abschnitt 4.2.2).

$$\begin{aligned} -512 \leq\ & x_i \leq 512 \qquad i = 1, \ldots, n \quad \text{für} \quad n = 9 \\ -32768 \leq\ & x_i \leq 32768 \qquad i = 1, \ldots, n \quad \text{für} \quad n = 17 \end{aligned}$$

2nd ICEO [2].

Für die Tests wurde $n = 9$ mit $d = 72.66066$ und $n = 17$ mit $d = 10558.145$ verwendet.

6.2 Optimierungs-Software

In diesem Abschnitt werde ich kurz die getesteten Optimierungsprogramme beschreiben. Alle folgenden Programme sind Versionen, welche für das Betriebssystem `UNIX` geschrieben wurden. Es handelt sich dabei aber nur um jene, bei denen es mir möglich war eine funktionsfähige Version zu erzeugen. Viele, hier nicht erwähnte, erzeugten Fehler beim compilieren oder das compilierte Programm funktionierte nicht fehlerfrei. Als Ausnahmen möchte ich hier noch die Programme `GAGA` [36] von Poole und `MCS` [33], [32] von Huyer & Neumaier erwähnen. `GAGA` wurde nicht in die Testreihe aufgenommen, weil es nur ganzzahlige Werte zuläßt, und von `MCS` war keine vollständige `C`-Version verfügbar (wohl aber eine für `MATLAB`). Da aber für die in Abschnitt 6.3.1 beschriebenen Tests Ergebnisse dieses Programms verfügbar waren, ist es dort selbstverständlich berücksichtigt.

Auf der Internet-Seite von Neumaier [33] sind Links zu den hier erwähnten Programmen zu finden.

6.2.1 Random

Dieses von mir in `C` geschriebene Programm versucht zu optimieren, indem es im Definitionsbereich einfach nur gleichverteilte Zufallspunkte bestimmt und die Funktion an diesen auswertet.

Es ist dafür konzipiert, als untere Grenze für die Qualität anderer Optimierungssoftware zu dienen und nicht als vollwertiges Optimierungsprogramm, da die Konvergenzeigenschaften viel zu schlecht sind. Trotzdem ist Bedingung (5.1) erfüllt, was noch einmal deren Schwäche herausstreicht.

6.2.2 Genocop

Dieses in `C` geschriebene Programm ist in Michalewicz [28] und [29] beschrieben und gehört zur Klasse der genetischen Algorithmen. Zum Download bietet sich die Internetseite von Michalewicz [29] an, wo auch frühere Versionen zur Verfügung stehen.

Für diese Arbeit wurden zwei dort verfügbare Varianten dieses Software-Paketes getestet, nämlich `Genocop 3.0` und `Genocop III` (Version 1.0).
`Genocop 3.0` kann nur mit linearen Nebenbedingungen umgehen. Dabei werden lineare Gleichungs- und Ungleichungs-Nebenbedingungen sowie koordinatenweise Nebenbedingungen getrennt eingegeben. Weiters stehen dem Benutzer noch einige Parameter zur Verfügung. Die in Klammern stehenden Zahlen entsprechen von den Autoren empfohlenen Werten oder den Voreinstellungen der Softwarepakete, die für die Tests unverändert übernommen wurden. So kann unter anderem etwa die Populationsgröße (70), die maximale Generationenzahl (500), ein Koeffizient, der den selektiven Druck steuert (0.1) gewählt werden. Weiters kann man entscheiden, ob man maximieren oder minimieren will und ob am Anfang von einem

einzigen Punkt oder einer zufälligen Population (voreingestellt) gestartet werden soll. Alle anderen Parameter sollten laut Beschreibung nicht verändert werden. `Genocop III` basiert auf `Genocop 3.0` und enthält zusätzlich noch die Möglichkeit nicht-linearer Ungleichungs-Nebenbedingungen und infolgedessen einige zusätzliche Parameter. Im Unterschied zu `Genocop 3.0` werden hier Gleichungs-Nebenbedingungen nicht extra berücksichtigt. Auf Grund der hier berücksichtigten Testfunktionen waren diese Unterschiede aber nicht relevant.

Der Code compilierte problemlos, es stellte sich allerdings heraus, daß die Datei mit den Parameterwerten nicht korrekt gelesen wurde. Da derartige Dateien, im Rahmen der hier durchgeführten Tests, ohnehin umgangen wurden, stellte dies aber keine Schwierigkeit dar.

Dieses Programm ist sehr einfach zu handhaben und man benötigt nur eine sehr kurze Dokumentation, um es verwenden zu können. Außerdem kommt es mit einer geringen Zahl von Parametern aus.

Obwohl das Programm bei der ersten Begutachtung seine Aufgabe ganz gut zu erfüllen schien, lieferte es aber bei den ersten Tests mit dem von mir geschriebenen Treiberprogramm `ej_eval` Ergebnisse, die nur wenig besser als die vom Programm `Random` waren. Die Ursache war, daß `Genocop` Vektoren nicht mit dem Index 0, wie es bei `C`-Programmen allgemein üblich ist, sondern mit Index 1 beginnen läßt, was unweigerlich zu einem Kommunikationsproblem mit der Funktion `ej_eval` führte. Die entsprechenden Tests wurden nach Behebung dieses Problems selbstverständlich wiederholt.

6.2.3 DE

DE ist ein von Storn et. al [37], [38] in `C` geschriebenes Programm, welches zur Klasse der evolutiven Algorithmen zu zählen ist. Dieses Programm machte in keinerlei Hinsicht Schwierigkeiten. Auf der Internet-Seite von Storn [38] sind alle benötigten Dateien für die hier getestete `C`-Version zu finden. Es sind dort

aber auch Versionen in `Matlab` und `C++`, sowie eine Beschreibung des Verfahrens vorhanden. Eine Kurzanleitung für die Handhabung des Programms fehlt, ist aber auch nicht wirklich erforderlich, da die wenigen benötigten Dateien gut organisiert sind und die angeführten Beispieldateien zum Verständnis vollkommen ausreichen. Die Datei, welche Werte für vom Benutzer änderbare Paramter enthält, wurde hier wie üblich umgangen und die Werte im Hauptprogramm gesetzt oder von `ej_eval` bereitgestellt.
Beim First International Contest on Evolutionary Computation (1^{st} ICEO) [1] erreichte dieses Programm den dritten Rang und wurde somit der Sieger unter den Teilnehmern, die evolutive Strategien anwandten.

Punkte werden hier als Vektoren im $\mathbb{R}^n$ aufgefaßt und in Generationen zusammengefaßt. Für jeden Vektor x_i der aktuellen Generation wird ein Vektor y_i als Kandidat für die folgende Generation erzeugt, indem ein Vielfaches der Differenz zweier Vektoren x_2 und x_3 zu einem dritten x_1 addiert wird:

$$y_i = x_1 + F \cdot (x_2 - x_3), \quad \text{für alle} \quad i$$

Die Vektoren x_1, x_2 und x_3 sind dabei zufällig ausgewählte Punkte der alten Generation und der konstante Faktor F liegt im Intervall $[0, 2]$. Das entspricht in etwa den Mutationen in herkömmlichen genetischen Algorithmen (Abschnitt 5.3.4, S. 53). Das Prinzip ist in Abbildung 6.19 schematisch dargestellt. Die geschlossenen Kurven entsprechen dabei Höhenschichtlinien und Sterne repräsentieren Punkten der aktuellen Generation.

Um die Vielfalt der Vektoren zu erhöhen, wird Crossover (siehe Abschnitt 5.3.4, S. 52) komponentenweise auf die durch Mutation erzeugten Vektoren angewandt. Jede Komponente des Vektors y_i wird mit der entsprechenden Komponente von x_i vertauscht, wenn eine gleichverteilte $[0, 1]$-Zufallszahl größer als eine vorgegebene Konstante CR ist. Um auszuschließen, daß y_i so nicht verändert wird, findet jedenfalls eine Vertauschung an einer zufälligen Stelle statt. Analog könnte man auch sicherstellen, daß nicht alle Komponenten vertauscht werden,

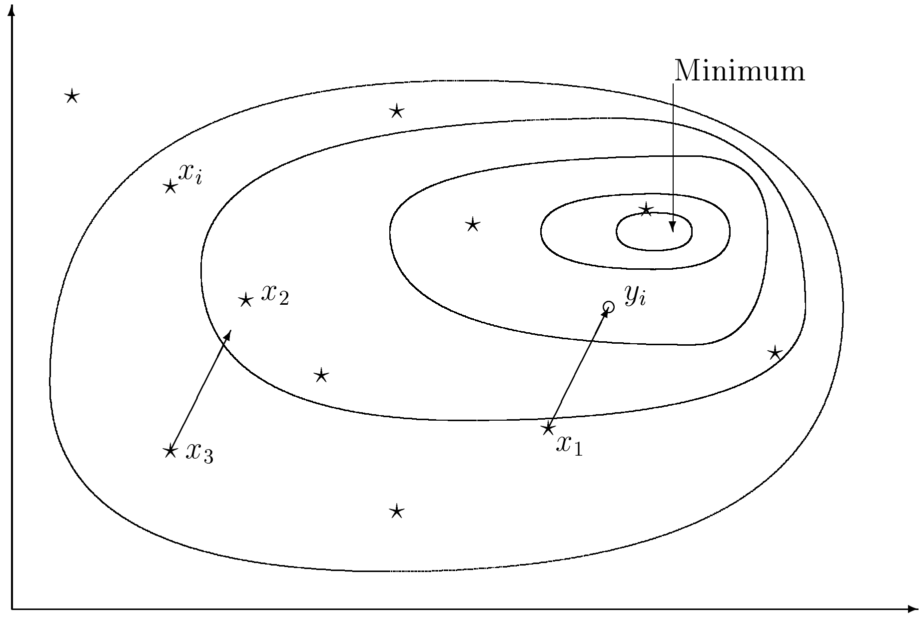

Abbildung 6.19: Mutation bei Differential Evolution.

da sonst y_i mit x_i identisch wird. Das ist aber im Programm nicht vorgesehen.

Besitzt nun der Vektor y_i einen kleineren Funktionswert als x_i, so wird y_i in die folgende Generation aufgenommen, sonst x_i.

Insgesamt stehen zehn Abarten des Programms zur Wahl. So kann man etwa statt des Vektors x_1 immer den Punkt der Generation mit dem kleinsten Funktionswert wählen oder für die Mutation mehr Punkte verwenden:

$$y_i = x_1 + F \cdot (x_2 + x_3 - x_4 - x_5)$$

Für die hier dargestellten Tests wurde aber die oben ausführlich beschriebene Variante verwendet. Die Parameter wurden dabei wie folgt gewählt: Der Faktor $F = 0.9$, die Wahrscheinlichkeit für Crossover $CR = 1$ und die Größe einer Generation wurde auf 60 festgelegt. Größere Werte der Konstante CR führen zu einer schnelleren Konvergenz, während kleinere Werte den Algorithmus robuster machen. Empfohlen ist, daß CR nicht außerhalb des Intervalls $[0.5, 1]$ liegen soll.

Dieses Programm unterstützt keine Nebenbedingungen, und die anfangs übergebenen Intervallgrenzen dienen nur dazu, den Bereich für die erste Generation einzuschränken. Dieser Umstand war mit ein Grund dafür, die Methode von Abschnitt 4.3.3 in `ej_eval` zu implementieren.

6.2.4 PGAPack

Dieses Programm gehört zu den evolutiven Algorithmen. Es erscheint anfangs etwas unübersichtlich, da dem Benutzer eine Unzahl von Parametern zur Verfügung stehen, von denen die meisten aber Standardeinstellungen besitzen und somit nicht explizit angegeben werden müssen. Die mitgelieferte, umfangreiche Anleitung würde durch eine Seite Kurzanleitung für den Einstieg gut ergänzt. Es steht dem Anwender frei, ob er die Zielfunktion maximieren oder minimieren möchte. Weiters kann man für die Darstellung der Gene unter anderem reelle, ganzzahlige oder binäre Werte wählen. PGAPack besitzt eine Vielzahl an Parametern, die der Anwender bestimmen kann. Standardmäßig ist etwa Tournier-Auswahl aktiviert, wobei das bessere Chromosom mit einer Wahrscheinlichkeit von 0.6 auch der Gewinner ist. Es ist auch erlaubt, daß Chromosomen in einer Population, die eine voreingestellte Größe von 200 Punkten besitzt, mehrfach enthalten sind. Mutationen werden standardmäßig nur bei Chromosomen angewandt, bei denen *kein* Crossover durchgeführt wurde und werden mit einer Wahrscheinlichkeit, die so groß ist, wie der Kehrwert von der Anzahl der Gene (im reellen Fall also $1/n$), auf ein Gen angewandt. Das geschieht, wenn man die Voreinstellung unverändert läßt, indem zum Wert eine $N(0, 0.1)$-verteilte Zufallsvariable addiert wird.

Bei der Festlegung der Art des Crossover für die hier durchgeführten Tests wurde ausnahmsweise nicht die Standardeinstellung, welche 2-Punkt-Crossover war, verwendet, da dies bei Testfunktionen mit 2-dimensionalen Definitionsbereich zu einer Fehlermeldung führte. Statt dessen wurde eine Variante gewählt, bei der jede Koordinate mit Wahrscheinlichkeit 0.5 ausgetauscht wird.

6.2.5 SIGMA

Dieses Programm ist ein von Aluffi-Pentini et al [4], [5] geschriebenes Fortran-Programm. Es ist eine Abart von **Simulated Annealing** und auf den Internetseiten von Neumaier [33] und der ACM [3] als Algorithmus 667 verfügbar. Im Paket waren Teile des Codes mehrfach vorhanden, wodurch sich Schwierigkeiten beim Compilieren ergaben. Nach dem Löschen der entsprechenden Teile funktionierte das Programm aber fehlerfrei. Mit der Beschreibung des Programms in [5] gab es keine Schwierigkeiten. Zusätzlich wäre es wünschenswert, wenn es eine Kurzbeschreibung im Internet verfügbar gäbe.

Um dem Hauptprogramm wichtige Parameter und Voreinstellungen mitzuteilen, ist eine Treiber-Routine inkludiert, welche von mir für die Tests entsprechend modifiziert wurde. Ein zu wählender Startpunkt war anfänglich in der Mitte des Definitionsbereiches $D = [U, O]$ gewählt. Das ist aber für einige Testprobleme eine schlechte Wahl, da genau dort das globale Minimum liegt und der Algorithmus dieses bereits bei der ersten Funktionsauswertung findet. Um das zu vermeiden, wurden später alle Komponenten des Anfangspunktes auf den Wert $(O_i + U_i)/3$ gesetzt. Alle anderen Parameter wurden wie üblich auf den voreingestellten Werten belassen.

Der Aufbau der Software sieht mehrere Versuche vor, ein globales Minimum zu finden. Jeder Versuch besteht darin, mehrere Folgen von Punkten, auch Trajektorien genannt, parallel zu berechnen. Nach einer vorgegebenen Anzahl von erfolgreichen oder einer maximalen Anzahl von Versuchen endet das Programm. Ein Versuch gilt dann als erfolgreich, wenn der kleinste, in den zu diesem Versuch gehörenden Trajektorien, gefundene Funktionswert höchstens so groß, wie der kleinste bisherige Funktionswert ist. Neue Punkte einer Trajektorie werden dabei wie folgt gebildet:

$$x_{k+1} = x_k - h_k \gamma(x_k) + \varepsilon(t_k) u_k \sqrt{h_k}, \qquad k \in \mathbb{N}$$

wobei x_0 ein am Anfang zu wählender Startpunkt und $\gamma(x_k)/n$ eine Approximation einer Ableitung in einer zufällig gewählten Richtung ist. Weiters ist h_k eine adaptive Schrittweite, $\varepsilon(t)$ ein Wert, der die Stärke der Störung beschreibt, und u_k eine Zufallsvariable, der eine n-dimensionale Gaußverteilung zugrunde liegt. Die Zeit, die das Berechnen der Trajektorien beansprucht ist in Abschnitte eingeteilt. Nach jedem dieser Abschnitte wird eine der Trajektorien verworfen und dafür eine andere aufgespalten.

Es ist auch möglich, ein n-dimensionales Intervall als Definitionsbereich zu übergeben. In diesem Fall wird die Zielfunktion außerhalb dieses Intervalls durch eine exponentiell wachsende Funktion überlagert.

6.2.6 VFSR und ASA

Der Name `VFSR` steht für **Very Fast Simulated Reannealing**, während `ASA` **Adaptive Simulated Annealing** bedeutet. Die Programme stammen von Ingber et al. [19], [20] und sind über das Internet erhältlich [21].

Beide Programmpaket enthalten leider keine Kurzanleitung. Um Einstellungen zu ändern muß man daher im `C`-Code die richtige Stellen suchen. Das Programm `VFSR` terminierte, wenn ein Temperaturwert zu klein wurde. Um auf jeden Fall die angestrebten 20 000 Funktionsauswertungen zu erhalten, mußten eingebaute Abbruchkriterien umgangen werden. Dies geschah, indem der Wert der Temperatur vergrößert wurde. Auf Grund der fehlenden Dokumentation war das entsprechend schwierig. Von diesem Programm wurde zusätzlich zur Originalversion eine modifizierte Version getestet.

`ASA` ist eine Weiterentwicklung von `VFSR`. Bei diesem Programm ist es möglich, den temperaturbedingten Abbruch per Option auszuschalten. In diesem Falle wird laut Beschreibung einfach die Abfrage nach zu kleiner Temperatur ausgesetzt. Da es also eine implementierte Variante gab, 20 000 Funktionsauswertungen zu erreichen, wurde diese verwendet, und es kommt zu keiner „aufgezwungenen“ Temperaturerhöhung wie bei `VFSR`.

Daß sich die Abbruchkriterien A1 und A2 (siehe Abschnitt 6.3.3) auf die unmodifizierte Version von `ASA` (bei den Tests als `ASAorig` bezeichnet) und die modifizierte Variante (bei den Tests `ASA` genannt) nicht anwenden ließen, liegt daran, daß diese nur sehr wenige Funktionsauswertungen durchführen konnten, bevor sie mit einem Fehler terminierten. Die Ursache dafür lag möglicherweise an einem verwendeten Parameter, der eine Art von Ableitung der Funktion darstellen soll und der, wegen der stark oszillierenden Funktionswerte der Easom Funktion (F22), einen zu großen Wert hätte annehmen müssen und infolge einen Absturz des Programms verursachte.

6.2.7 CS

Dieses namenlose Programm (hier `CS` genannt) von Tibor Csendes [11] ist eine Implementation des Algorithmus, welcher in Boender et. al [8] vorgestellt und hier in Abschnitt 5.5.2 besprochen wird. Für die lokale Suche wird ein Quasi-Newton-Verfahren verwendet und statt der Euklidischen Norm wird die schneller zu berechnende Maximums-Norm angewandt.

Zuerst ließ sich das Programm nicht compilieren, was sich aber durch Löschen einiger im Paket vorhandener, aber unnötiger Dateien beheben ließ. Obwohl in der `readme`-Datei volle Kompatibilität zu Fortran77 angegeben wird, verursachte dies mit einigen in- und output Befehlen Schwierigkeiten, die aber bei Verwendung des Fortran90-Compilers nicht mehr auftraten.

Das Paket enthält nur eine kurze Beschreibung in Form einer `readme`-Datei, welche sowohl die theoretischen wie auch die praktischen Aspekte des Programms behandelt. Als Voreinstellung sind Probleme mit einer maximalen Dimension von 15 zugelassen, was sich aber mit Hilfe der Beschreibung ändern läßt. Die Voreinstellung für die Anzahl der in jeder Iteration zu bestimmenden Zufallspunkte ist 100 und wurde wie alle Parameter so belassen.

Mir war es bei diesem Programm nicht möglich, durch Studium des Codes herauszufinden, wie und wo Abbruchskriterien verwendet werden. Daher wurden mit diesem Programm oftmals auch keine 20 000 Funktionsauswertungen erreicht. Auch kam es hin und wieder vor, daß das Programm in einer Endlosschleife gefangen war, ohne Funktionsauswertungen durchzuführen.

Durch Herumprobieren hatte ich letzten Endes drei Varianten dieser Software, die dann auch alle getestet wurden:

CSorig: Die oben beschriebene, unmodifizierte Variante.

CSrand: Eine, welche für die lokale Suche eine Zufallssuche einsetzt und ebenfalls im Programmpaket enthalten war.

CS: Eine, welche von mir so modifiziert wurde, daß doch 20 000 Funktionsauswertungen durchgeführt werden, die jedoch meist abstürzt oder ebenfalls in einer Endlosschleife landet. Bevor dies geschieht konnte diese Variante aber meist noch gute Funktionswerte liefern.

Auch dieses Programm kennt keine Nebenbedingungen. Die anfänglich eingegebenen Intervallgrenzen dienen nur dazu, den Bereich für die Zufallspunkte einzugrenzen, und werden bei der lokalen Minimierung ignoriert. Dieser Umstand wird aber bekanntlich von `ej_eval` abgefangen.

6.3 Testergebnisse

6.3.1 Dixon-Szegö Testset

Tabelle 6.2 bildet eine Erweiterung jener in Neumaier & Huyer [32], S. 20, welche wieder Ergebnisse aus Jones et al. [23] verwendet. Unter dem horizontalen Strich sind die für diese Arbeit getesteten Verfahren angegeben. Der Testset stammt aus Dixon & Szegö [14].

Als Abbruchkriterium für die getesteten Programme wurde das erstmalige Erreichen eines relativen Fehlers von weniger als 0.01% vom globalen Optimum, also $(f-f^*)/f^* < 10^{-4}$, verwendet. Dieses Kriterium gilt jedenfalls für die von mir getesteten Programme, `MCS` und die Programme von Jones et al., Perttunen und Perttunen & Stuckman. Für das unter Storn-Price geführte Programm `DE` gilt statt dessen ein relativer Fehler von weniger als 1%. Für die Algorithmen, die in Jones et al. [23] zitiert sind, gelten die von den Autoren angegebenen Abbruchkriterien.

6.3.2 Testset aus Abschnitt 6.1

Die Grundlage für die Tests bilden die 45 Funktionen, die in Abschnitt 6.1 vorgestellt werden. Abbildung 6.20, S. 112 stellt eine Art Rangliste der Testfunktionen dar, die ein Maß dafür sein soll, wie schwierig eine Funktion zu minimieren ist. Dargestellt ist, wie oft jedes Programm den besten überhaupt gefundenen relativen Funktionswert bis auf eine Genauigkeit von 10^{-6}, 10^{-4} beziehungsweise 10^{-2} erreicht hat. Die Rangliste basiert in erster Linie darauf, wie oft ein Programm bei einer Genauigkeit von 10^{-4} das beste war. Zur weiteren Abstufung wurden auch die Werte bei den Toleranzen von 10^{-6} und 10^{-2} herangezogen. Für die folgenden Abbildungen 6.21–6.24 wurden die Testfunktionen in drei Klassen eingeteilt, die auf dieser Rangliste beruhen: 24 *schwere*, 13 *mittelschwere* und 8 *leichte* Funktionen.

Die Testergebnisse werden hier in Form von mehreren Tabellen und Abbildungen präsentiert (siehe Abschnitt 6.3.5). Die Zeilen der Tabellen 6.3–6.7 entsprechen den in Abschnitt 6.1 vorgestellen Testfunktionen. Die Zeilen lassen sich den in Abschnitt 6.2 besprochenen Programmpaketen folgendermaßen zuweisen: ASA und ASAorig (Abschnitt 6.2.6), CS, CSorig und CSrand (Abschnitt 6.2.7), DE (Abschnitt 6.2.3), Geno3 und GenoIII (Abschnitt 6.2.2), PGA (Abschnitt 6.2.4), Random (Abschnitt 6.2.1), SIGMA (Abschnitt 6.2.5) und VFSR und VFSRorig (Abschnitt 6.2.6).
Die jeweils besten gefundenen Werte einer Testfunktion sind dabei unterstrichen.

Tabelle 6.3 enthält dabei die besten ermittelten Funktionswerte nach maximal 20 000 Funktionsauswertungen. Die Darstellung ist auf drei Stellen genau und wurde durch Abschneiden der hinteren Stellen erzeugt.
Tabelle 6.4 enthält ebenfalls die besten Funktionswerte nach maximal 20 000 Auswertungen. In diesem Fall wurden sie aber modifiziert, um nicht nur die Programme an Hand einer Testfunktion vergleichen zu können, sondern auch die Testfunktionen in bezug auf einzelne Programme. Um das zu erreichen, stehen in der Tabelle die folgendermaßen veränderten Funktionswerte $\bar{f}$, die ab nun *relative* Funktionswerte heißen sollen:

$$\bar{f} = \frac{f - f^*}{f_m - f^*} \tag{6.1}$$

Dabei ist f der ursprüngliche Funktionswert, ab jetzt auch *absoluter* Funktionswert genannt. Der Wert f^* ist der kleinste (von allen Programmen) gefundene Funktionswert des aktuellen Testproblems, und f_m wird gebildet, indem von jedem Programm der Median aller Funktionsauswertungen des aktuellen Testproblems gebildet wird und danach der Median dieser Mediane bestimmt wird. Wie man leicht erkennt, ist somit der kleinste relative Funktionswert immer gleich 0. Wegen der Modifikation kann es vorkommen, daß die unterstrichenen Stellen nicht vollkommen mit denen aus Tabelle 6.3 übereinstimmen. Auch wurden die dreistellige Darstellung in diesem Fall durch optimale Rundung erhalten.
Tabelle 6.7 enthält die jeweils besten relativen Funktioswerte $\bar{f}$ nach 1 000, 2 500, 5 000, 10 000 beziehungsweise 20 000 Funktionsauswertungen. Falls ein Programm vor Erreichen einer dieser Marken terminiert hat, so ist die entsprechende Stelle in der Tabelle ein * vor dem Funktionswert eingefügt. Auch hier wurden die Zahlen auf drei Stellen gerundet.
Tabelle 6.1 gibt an, wie oft ein spezielles Programm den bis zu den einzelnen Marken besten Funktionswert (bei größtmöglicher Genauigkeit) gefunden hat.

Die Werte der Spalten A1 und A2 geben an, wie oft ein Programm bei Anwendung des entsprechenden Abbruchkriteriums aus Abschnitt 6.3.3 den kleinsten Funktionswert (aller Programme *mit* Abbruchkriterium) gefunden hat. Die in Klammern gesetzten Werte geben an, wie oft das Programm bei Anwendung der Abbruchkriterien den überhaupt besten gefundenen Funktionswert (aller Programme *ohne* Abbruchkriterium) erreicht hat, und kursive Zahlen geben darüber Auskunft, bei wie vielen Testfunktionen ein Programm mehr als 20 000 Funktionsauswertungen benötigt hätte, um das Abbruchkriterium zu erfüllen.

Die Daten für die Spalten A1 und A2 wurden dabei den Tabellen (6.5) und (6.6) entnommen.

Abbildungen 6.21–6.24, S. 113ff zeigen, wie oft ein Programm – nach 1 000, 2 500, 5 000, 10 000 und 20 000 Funktionsauswertungen beziehungsweise bei Anwendung der Abbruchkriterien A1 und A2 (siehe Abschnitt 6.3.3) – den kleinsten von allen Programmen gefundenen Funktionswert erreicht hat. Die Funktionen sind dabei zusätzlich in die drei Klassen *schwer*, *mittelschwer* und *leicht* aus Abbildung 6.20 aufgeschlüsselt. Während Abbildung 6.21 größtmögliche Genauigkeit zu Grunde gelegt wurde, sind bei den Abbildungen 6.22–6.24 noch Toleranzen von 10^{-6}, 10^{-4} beziehungsweise 10^{-2} bei den relativen Funktionswerten erlaubt. Die erlaubten Toleranzen beziehen sich dabei auf den überhaupt besten relativen Funktionswert, der von allen Programmen nach 20 000 Funktionsauswertungen gefunden wurde.

6.3.3 Abbruchkriterium

In dieser Arbeit soll an den Testproblemen aus Abschnitt 6.1 ein Abbruchkriterium ausprobiert werden. Mit Hilfe eines Parameters p, wird aus dem Quotienten Q_j der Differenzen $f^*_{k_{j-2}} - f^*_{k_{j-1}}$ und $f^*_{k_{j-1}} - f^*_{k_j}$ der Quotient $\overline{Q_{j+1}}$ von $f^*_{k_{j-1}} - f^*_{k_j}$ und $f^*_{k_j} - f^*_{k_{j+1}}$ geschätzt, wobei f^*_s der bis zur s-ten Funktionsauswertung beste Funktionswert ist und p mit Hilfe der Methode der kleinsten Quadrate berechnet wurde. Die k_j sind dabei Marken aus einer vorgegebenen Menge. Startet man nun das Optimierungsprogramm, so berechnet man zuerst einmal k_2 Funktionswerte und bestimmt Q_2. Nun schätzt man den folgenden Quotienten Q_3, wie in Gleichung (6.2) angegeben:

$$\overline{Q_{j+1}} \quad = \quad p\,Q_j = p\,\frac{f^*_{k_{j-2}} - f^*_{k_{j-1}}}{f^*_{k_{j-1}} - f^*_{k_j}} \tag{6.2}$$

Ist nun der geschätzte Quotient kleiner als 1, so wird bis zur k_{j+1}-ten Funktionsauswertung weitergerechnet, und obiger Vorgang wiederholt, bis der der geschätzte Quotient eimal größer als 1 ist oder ein anderes Abbruchkriterium zum Zuge kommt.

Für diese Arbeit wurden die Marken auf die Werte $20\,000/2^s$ (abgerundet) gesetzt. Tabellen 6.5 und 6.6 enthalten die besten relativen Funktionswerte bei Anwendung des Abbruchkriteriums. Die besten gefundenen Werte eines Testproblems (von allen Programmen) sind unterstrichen und ein * bedeutet, daß das Programm nach 20 000 Funktionsauswertungen noch weiter minimimiert hätte. Fehlende Eintragungen bedeuten, daß nicht ausreichend viele Funktionsauswertungen produziert wurden. Da die Modifikation immer mittels der besten überhaupt gefundenen Funktionswerte durchgeführt wurde, erkennt man die Programme, die auch mit der Abbruchbedingung diese kleinsten Funktionswerte gefunden haben, am Eintrag 0.

Für Tabelle 6.5, S.120 wurden mindestens 156 Funktionsauswertungen durchgeführt und die erste Marke für die Lösung des kleinsten Quadrate-Problems auf $20\,000/2^{11}$ gesetzt (Abbruchkriterium A1), während für Tabelle 6.6 mindestens 312 Funktionsauswertungen und als kleinste Marke $20\,000/2^9$ verwendet wurden (Abbruchkriterium A2). Vermutlich ist A2 die bessere Wahl, da die Parameterschätzung bei A1 auch sehr frühe Funktionswerte verwendet werden und am Anfang möglicherweise noch überdurchschnittlich große Fortschritte erziehlt werden, die sich negativ auf das Ergebnis auswirken könnten.

6.3.4 Kommentar zu den Testergebnissen

An dieser Stelle soll kein Urteil über die einzelnen Programme beziehungsweise Algorithmen abgegeben werden, wohl aber auf den, in Abschnitt 6.3.5 dargestellten, Ergebnissen beruhende Empfehlungen.

Bei der Betrachtung der Abbildungen 6.21–6.24 zeigt sich, daß die „Güte" eines Programms, im Vergleich zu den anderen, nicht sehr von der zu Grunde gelegten Genauigkeit abzuhängen scheint, da kein Programm wirklich auffallende Zuwächse verzeichnen konnte.

Die erziehlten Ergebnisse bei den *leichten* Testfunktionen – besonders bei größeren Toleranzen – werden hier nicht weiter analysiert, können aber als negatives Beurteilungskriterium herangezogen werden. Weiters legen die Daten den Schluß nahe, daß Algorithmen, welche auf Bildung von Clustern beruhen, trotz ihrer Nachteile sich besonders für schwere Optimierungsprobleme eignen und bei den mittelschweren Problemen eher unterlegen sind. Es scheint mir sehr wahrscheinlich, daß dies auf die Verwendung von lokalen Optimierern beruht. Dies nicht zuletzt deshalb, weil die Variante, welche Zufallszahlen zur lokalen Optimierung einsetzte (`CSrand`), zu den erfolglosesten Programmen gehörte. Vermutlich ist es daher einer Überlegung wert, wie man einen bestimmten Algorithmus geeignet mit lokalen Suchen verbinden kann, um bessere Ergebnisse zu erhalten.

Es zeichnet sich auch die Tendenz ab, daß gut programmierte genetische Algorithmen anfangs sehr schwach sind aber überdurchschnittlich stark von großen Laufzeiten profitieren, während auf Simulated Annealing basierende Verfahren die größten Fortschritte am Anfang machen und später eher zurückbleiben.

Es verwundert nicht, daß die von mir modifizierten Programme besser abschneiden, als die ursprüglichen, da ja durch Umgehung integrierter Abbruchbedingungen mehr Funktionsauswertungen durchgeführt wurden.

Die in Abschnitt 6.3.3 vorgestellten Abbruchkriterien scheinen nicht für alle Programme gleichermaßen geeignet. So sind genetische Algorithmen bei Anwendung der Abbruchkriterien in der Rangliste weiter hinten zu finden als sonst. Besonders das Programm `SIGMA` und die genetischen Algorithmen schnitten mit den Abbruchkriterien schlechter ab als ohne. Auch in Hinblick auf die Anzahl der Funktionen, bei denen mit Abbruchkriterium mehr als 20 000 Funktionsauswertungen durchgeführt worden wären, unterscheiden sich die Programme erheblich, wie man an Hand von Tabelle 6.1 (kursive Einträge) leicht überprüfen kann.

6.3.5 Testergebnisse

Dieser Abschnitt enthält die Abbildungen und Tabellen, welche die Ergebnisse der für diese Arbeit durchgeführten Tests darstellen und veranschaulichen sollen.

Programm	1 000	2 500	5 000	10 000	20 000	A1			A2		
ASA	12	14	14	16	14	13	(11)	*16*	11	(9)	*17*
ASAorig	13	11	11	13	12	12	(10)	*16*	10	(8)	*16*
CS	23	23	24	23	22	24	(15)	*0*	23	(15)	*0*
CSorig	21	21	21	19	18	19	(11)	*0*	19	(12)	*0*
CSrand	1	1	1	1	1	5	(3)	*0*	13	(9)	*0*
DE	0	0	5	12	19	2	(2)	*0*	3	(3)	*1*
Geno3	0	3	7	11	19	4	(3)	*16*	6	(6)	*23*
GenoIII	2	0	1	4	7	1	(1)	*4*	3	(2)	*8*
PGA	4	5	4	4	3	3	(1)	*16*	4	(1)	*30*
Random	0	0	0	0	0	0	(0)	*0*	0	(0)	*0*
SIGMA	3	4	4	5	6	1	(1)	*0*	1	(1)	*0*
VFSR	13	12	16	15	13	18	(11)	*19*	16	(11)	*21*
VFSRorig	13	10	12	11	10	14	(9)	*17*	12	(9)	*18*

Tabelle 6.1: So oft lieferte ein Programm den besten Wert

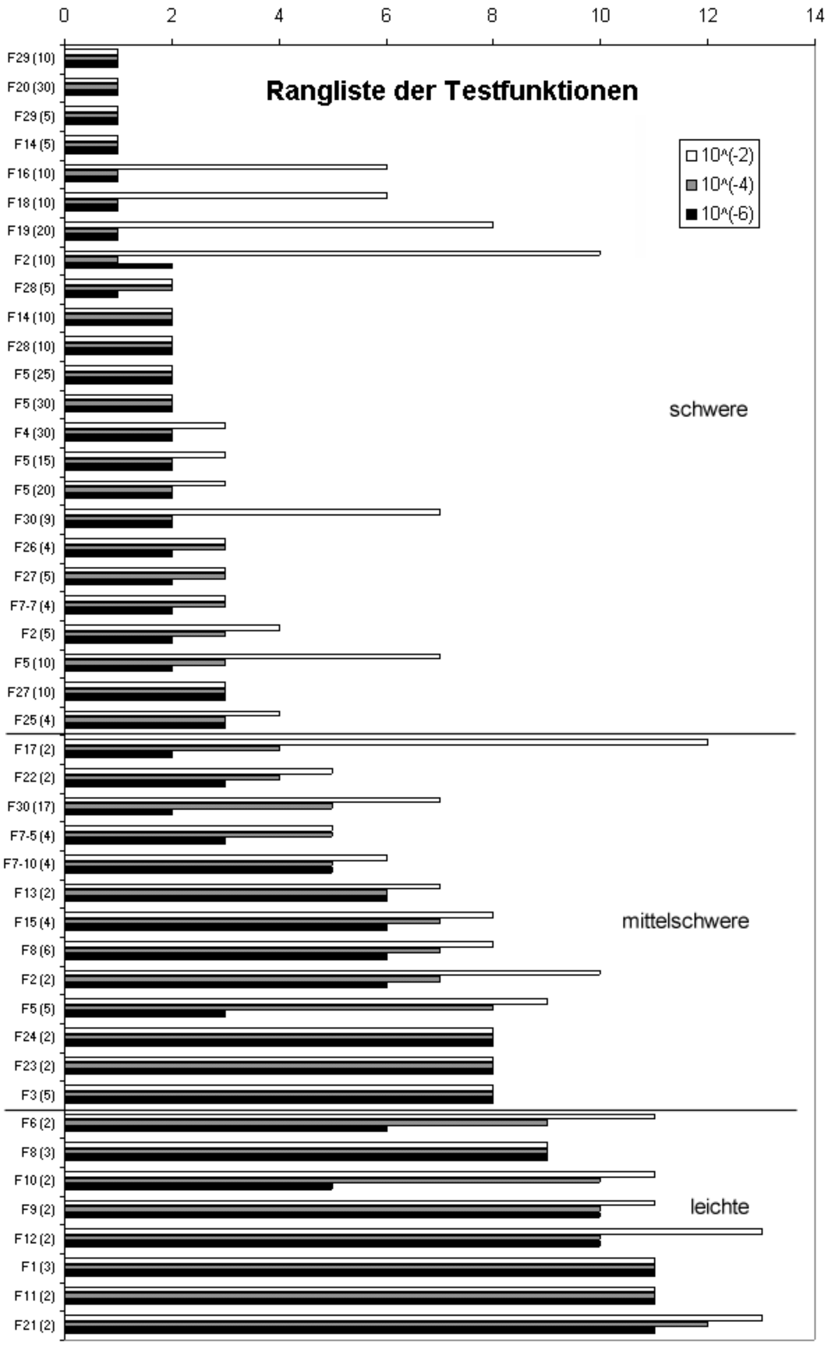

Abbildung 6.20: Rangliste der Testfunktionen. Aufgetragen ist die Zahl der Programme, die den besten relativen Funktionswert bis auf eine Genauigkeit von 10^2, 10^4 beziehungsweise 10^6 erreichten.

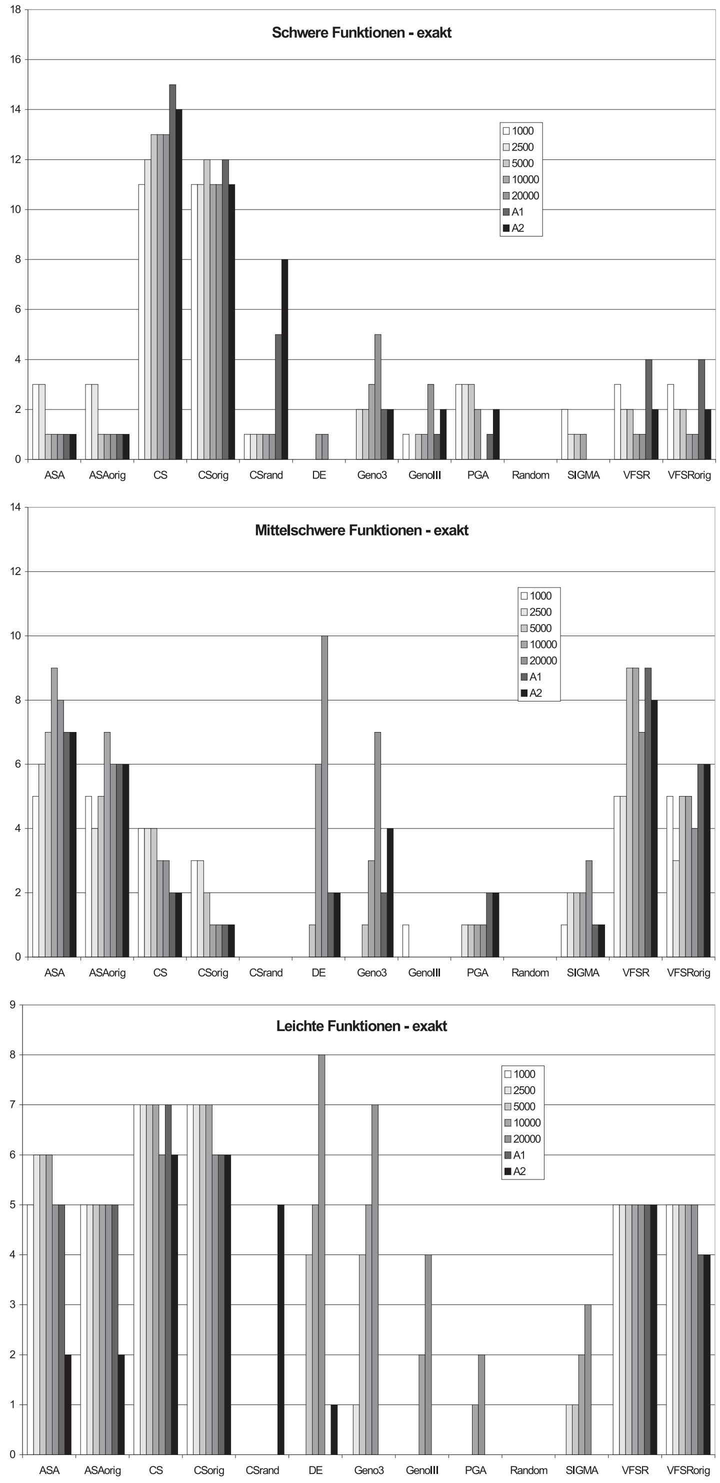

Abbildung 6.21: Performance der Funktionen – Größtmögliche Genauigkeit

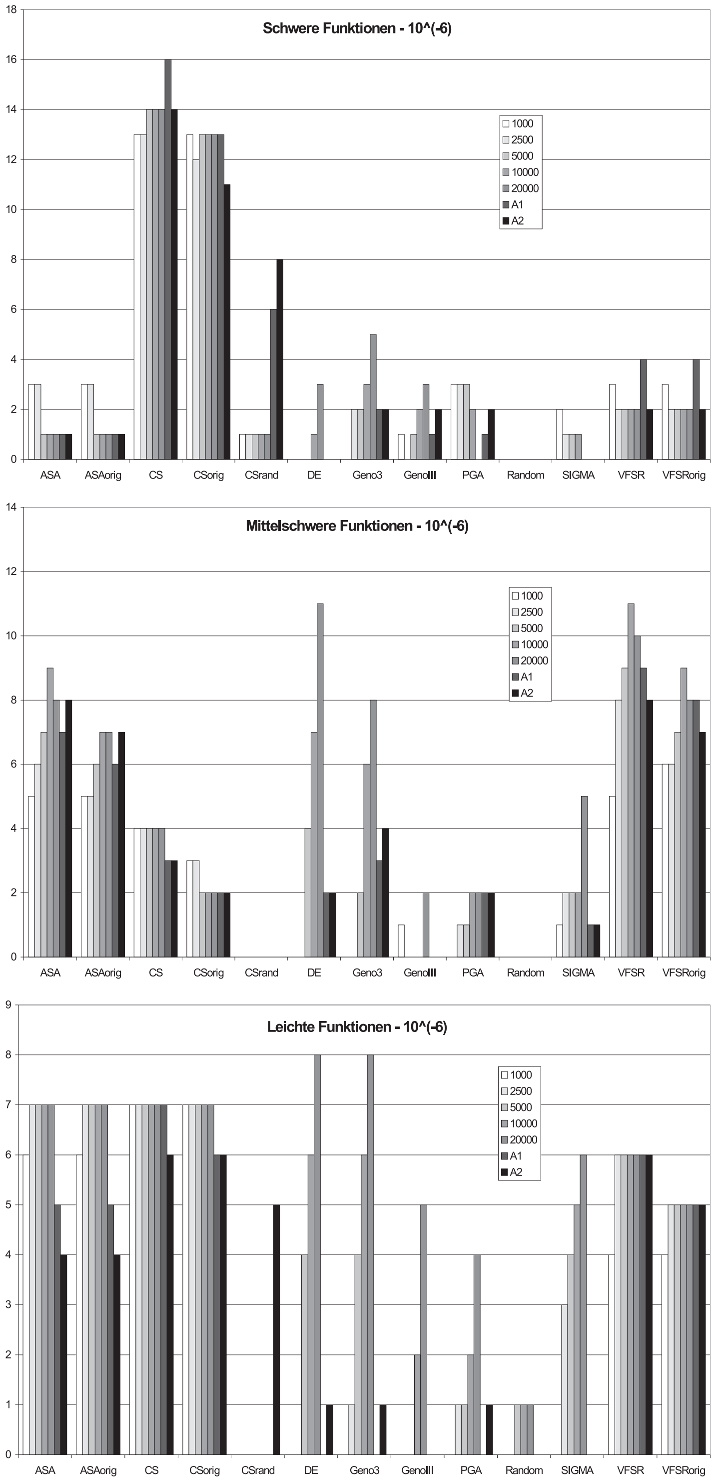

Abbildung 6.22: Performance der Funktionen – Toleranz: 10^{-6}

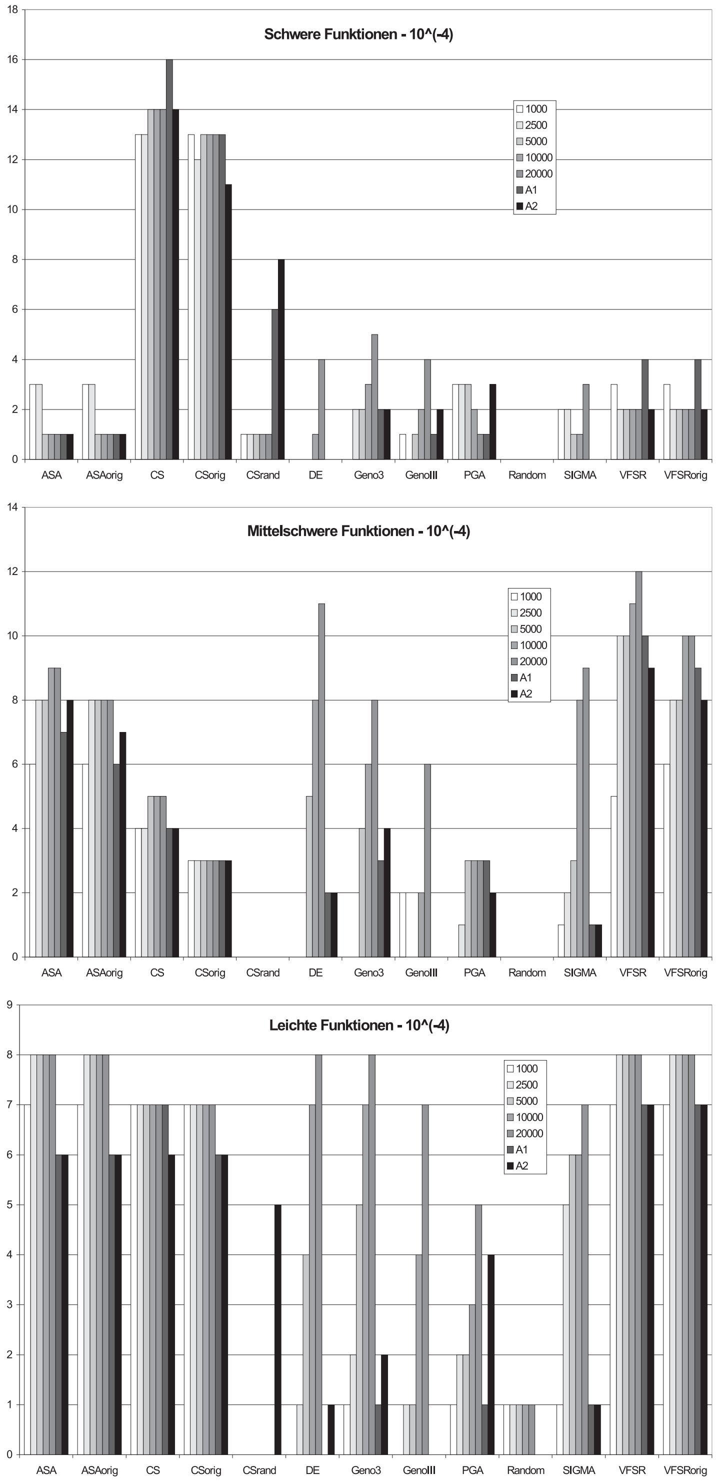

Abbildung 6.23: Performance der Funktionen – Toleranz: 10^{-4}

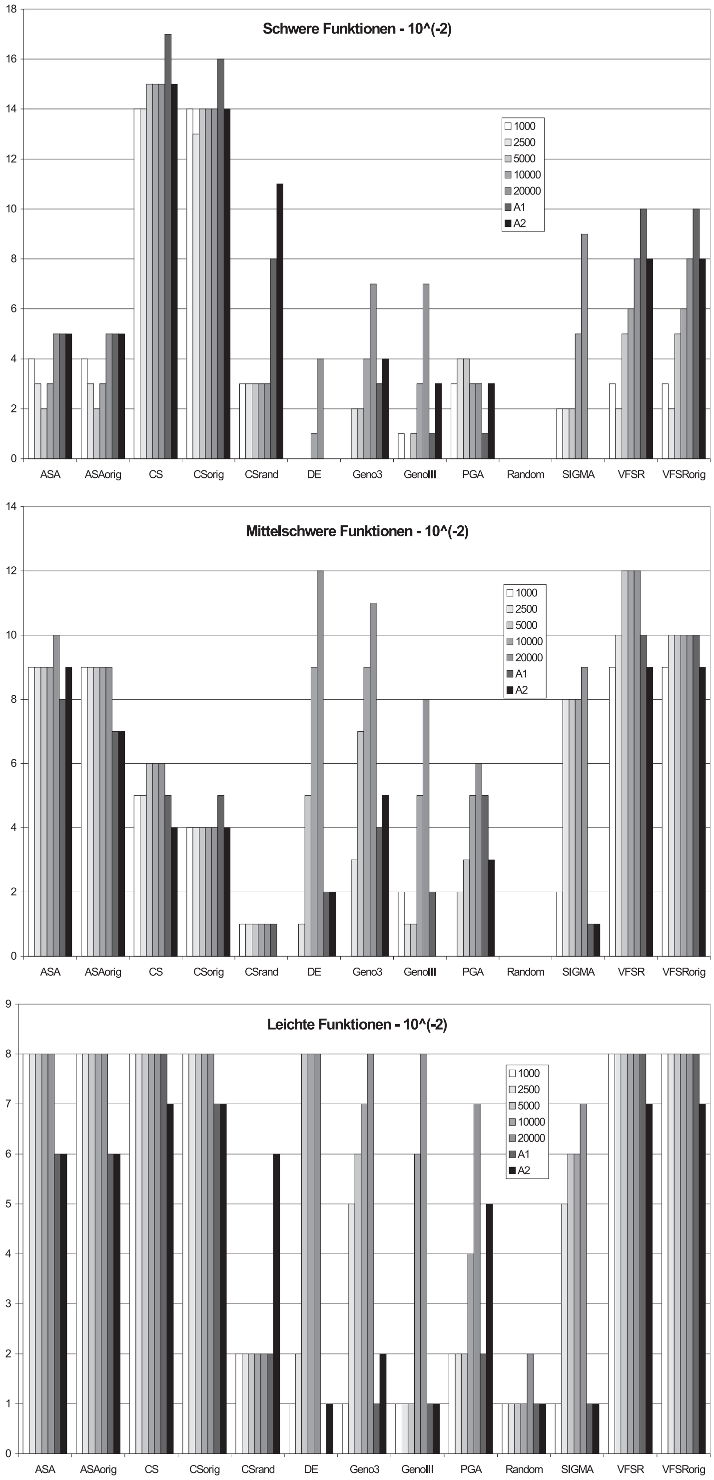

Abbildung 6.24: Performance der Funktionen – Toleranz: 10^{-2}

Programm	S5	S7	S10	H3	H6	GP	BR	C6	SHU
Bremmerman	(a)	(a)	(a)	(a)	(a)	(a)	250		
Mod. Bremmerman	(a)	(a)	(a)	(a)	515	300	160		
Zilinskas	(a)	(a)	(a)	8641			5129		
Gomulka–Branin	5500	5020	4860						
Törn	3679	3606	3874	2584	3447	2499	1558		
Gomulka–Törn	6654	6084	6144						
Gomulka–V.M.	7085	6684	7352	6766	11125	1495	1318		
Price	3800	4900	4400	2400	7600	2500	1800		
Fagiuoli	2514	2519	2518	513	2916	158	1600		
De Biase–Frontini	620	788	1160	732	807	378	587		
Mockus	1174	1279	1209	513	1232	362	189		
Bélisle et al. (b)				339	302	4728	1846		
Boender et al.	567	624	755	235	462	398	235		
Snyman–Fatti	845	799	920	365	517	474		178	
Kostrowicki–Piela	-	-	-	200	200	120		120	
Yao							1132	< 6000	
Perttunen	516	371	250	264		82	97	54	197
Perttunen–Stuckman	109	109	109	140	175	113	109	96	(a)
Jonet et al.	155	145	145	199	571	191	195	285	2967
Storn–Price (c)	6400	6194	6251	476	7220	1018	1190	416	1371
MCS(e)	84	129	103	79	111	81	41	42	69
ASA(d)	-	-	604	74	599	197	332	269	343
ASA	-	-	604	74	599	197	332	269	343
VFSR (d)	-	-	-	133	854	406	364	264	287
VFSR	-	-	-	133	802	406	364	264	287
Toms667	-	-	1557	4461	3109	3243	1509	743	-
Csendes(d)	779	1082	2515	123	316	133	115	121	171
Csendes	-	-	-	123	316	133	115	121	171
Csendes (e)	-	-	-	-	-	-	264	129	-
Genocop 3.0	-	-	-	2621	-	1327	1952	1535	2792
Genoocpp III	-	-	-	4229	-	5130	9659	5190	-
DE	8378	7661	8374	1926	-	1819	4534	2598	11589
PGA	-	-	-	1898	-	10836	-	1313	2426
random	-	-	-	-	-	-	-	-	-

(a) Verfahren konvergiert gegen lokales Minimum.
(b) Mittelwert der Testläufe. Bei H6 konvergierten 70 % der Testläufe.
(c) Mittelwert von 25 Testläufen. Für H6 wurden nur 24 Testläufe verwendet;
In einem Fall konvergierte das Verfahren nicht innerhalb von 12 000 Funktionsauswertungen.
(d) In diesem Fall wurden programminterne Abbruchkriterien umgangen.
(e) Die lokale Suche wird hier mittels Zufallszahlen durchgeführt.
- Erforderliche Genauigkeit nach 12 000 Funktionsauswertungen noch nicht erreicht

Tabelle 6.2: Anzahl der Funktionsauswertungen beim Dixon-Szegö-Testset

Tabelle 6.3: Die besten gefundenen absoluten Funktionswerte

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
F1	3	3.96e−20	5.83e−17	8.50e−23	8.50e−23	2.76e−05	6.78e−26	3.72e−11	1.29e−10	1.30e−09	1.51e−02	1.56e−11	9.92e−20	9.91e−20
F2	2	1.94e−05	5.93e−03	2.19e−11	2.19e−11	2.91e−02	0.00e+00	0.00e+00	3.16e−06	5.07e−05	2.36e−03	1.12e−12	1.64e−11	2.11e−05
F2	5	5.15e−01	3.11e−01	8.82e−11	8.82e−11	4.12e−03	8.34e−07	2.80e+00	3.59e+00	1.69e+00	1.06e+01	9.03e−01	3.38e−01	3.38e−01
F2	10	2.97e+00	2.97e+00	3.65e−10	3.65e−10	8.99e+00	4.29e+02	8.18e+00	8.85e+00	5.23e+01	5.14e+03	5.08e+00	1.47e−01	1.47e−01
F3	5	0.00e+00	0.00e+00	1.10e+01	1.10e+01	1.10e+01	0.00e+00	0.00e+00	2.00e+00	0.00e+00	5.00e+00	0.00e+00	0.00e+00	0.00e+00
F4	30	1.47e+01	1.47e+01	9.84e−21	9.84e−21	8.34e+03	1.93e+02	9.71e+00	5.02e+00	4.86e+00	1.06e+03	9.65e−01	1.50e+01	1.50e+01
F5	5	-3.00e+01	-3.00e+01	-2.99e+01	-2.99e+01	-2.86e+01	-3.00e+01	-2.99e+01	-2.93e+01	-1.49e+01	-1.63e+01	-2.99e+01	-3.00e+01	-3.00e+01
F5	10	-2.05e+02	-2.05e+02	-2.09e+02	-2.09e+02	6.66e+03	2.14e+03	-3.91e+01	-5.07e+01	-4.64e+01	2.89e+03	-2.09e+02	-2.08e+02	-2.08e+02
F5	15	1.28e+03	1.28e+03	-6.64e+02	-6.64e+02	8.14e+04	2.38e+04	2.40e+02	-1.78e+02	9.63e+02	4.83e+04	-5.26e+02	5.20e+02	5.20e+02
F5	20	9.60e+03	9.60e+03	-1.51e+03	-1.51e+03	3.48e+05	2.67e+05	-6.09e+02	8.92e+02	2.35e+02	2.47e+05	2.73e+02	2.49e+03	2.49e+03
F5	25	2.17e+04	2.17e+04	-2.89e+03	-2.89e+03	8.51e+05	2.35e+05	3.01e+04	2.50e+03	6.20e+03	8.83e+05	5.53e+03	2.86e+04	2.86e+04
F5	30	1.02e+05	1.02e+05	-4.92e+03	-4.92e+03	1.76e+06	2.32e+06	3.70e+04	2.12e+04	3.94e+04	2.04e+06	1.45e+04	1.15e+05	1.15e+05
F6	2	1.47e−08	2.91e−05	5.96e−01	5.96e−01	1.70e+02	0.00e+00	0.00e+00	1.04e−02	1.78e−02	1.59e+03	8.96e−08	6.72e−08	9.97e−04
F7-5	4	-2.68e+00	-2.68e+00	-1.01e+01	-5.10e+00	-5.08e+00	-1.01e+01	-1.01e+01	-1.01e+01	-2.68e+00	-2.65e+00	-1.01e+01	-5.10e+00	-5.10e+00
F7-7	4	-2.75e+00	-2.75e+00	-1.04e+01	-5.12e+00	-5.11e+00	-1.04e+01	-5.08e+00	-1.04e+01	-2.76e+00	-2.83e+00	-5.12e+00	-5.12e+00	-5.12e+00
F7-10	4	-1.05e+01	-1.05e+01	-1.05e+01	-5.17e+00	-5.15e+00	-1.05e+01	-2.80e+00	-5.01e+00	-1.05e+01	-2.95e+00	-1.05e+01	-5.17e+00	-5.17e+00
F8	3	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.85e+00	-3.86e+00	-3.86e+00	-3.86e+00
F8	6	-3.32e+00	-3.32e+00	-3.32e+00	-3.32e+00	-3.27e+00	-3.20e+00	-3.20e+00	-3.32e+00	-3.20e+00	-3.07e+00	-3.32e+00	-3.32e+00	-3.32e+00
F9	2	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.02e+00	3.00e+00	3.00e+00	3.00e+00
F10	2	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.98e−01	3.97e−01	3.97e−01	3.97e−01
F11	2	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00
F12	2	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02	-1.86e+02
F13	2	9.98e−01	9.98e−01	2.98e+00	6.90e+00	6.90e+00	9.98e−01	9.98e−01	9.98e−01	1.99e+00	1.02e+00	2.98e+00	9.98e−01	9.98e−01
F14	5	-2.67e+00	-2.67e+00	-3.47e+00	-3.47e+00	-3.46e+00	-3.18e+00	-1.04e+01	-2.67e+00	-3.52e+00	-2.14e+00	-9.78e+00	-3.29e+00	-3.29e+00
F14	10	-1.47e+00	-1.47e+00	-1.02e+01	-1.02e+01	-3.60e−01	-8.59e−01	-1.42e+00	-1.41e+00	-1.47e+00	-3.37e−01	-2.37e+00	-1.47e+00	-1.47e+00
F15	4	0.00e+00	0.00e+00	7.76e+05	7.76e+05	1.57e+06	1.83e−02	0.00e+00	2.19e+00	4.82e+01	1.77e+05	4.44e+05	0.00e+00	0.00e+00
F16	10	1.76e−01	1.76e−01	6.64e−02	6.64e−02	7.09e−04	1.39e+00	1.78e−01	6.36e−01	2.52e−01	2.06e+01	4.80e−02	6.71e−02	6.71e−02
F17	2	6.98e−01	6.99e−01	7.37e−01	7.37e−01	7.52e−01	0.00e+00	6.98e−01	6.19e−03	1.18e−03	1.62e+02	1.64e−02	2.79e−10	1.59e−02
F18	10	1.05e+01	1.05e+01	6.85e+01	3.39e+02	8.32e+02	4.17e+02	1.00e+00	3.16e+02	3.05e+00	1.99e+02	2.16e+01	9.70e+00	9.70e+00
F19	20	2.09e+03	2.09e+03	3.66e+05	3.66e+05	2.72e+06	9.63e+04	8.03e+02	2.49e+03	9.12e+02	6.86e+05	1.69e+03	1.91e+03	1.91e+03
F20	30	2.77e+00	2.77e+00	6.47e+00	7.48e+00	8.15e+00	6.46e+00	2.09e+00	1.40e+00	1.49e+00	5.88e+00	3.62e+00	2.78e+00	2.78e+00
F21	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.25e−06	0.00e+00	0.00e+00	2.16e−09	1.26e−11	1.62e−16	1.79e−19	0.00e+00	0.00e+00
F22	2	-1.08e−47	0.00e+00	0.00e+00	0.00e+00	0.00e+00	-1.00e+00	-1.00e+00	-9.91e−01	-9.99e−01	-8.02e−01	-4.66e−01	-1.00e+00	-9.00e−49
F23	2	0.00e+00	5.36e−14	4.12e−01	4.12e−01	4.85e−01	0.00e+00	0.00e+00	3.07e−12	8.82e−01	6.25e−01	0.00e+00	0.00e+00	2.06e−12
F24	2	0.00e+00	1.11e−16	2.18e−01	2.18e−01	2.28e−01	0.00e+00	1.45e−13	1.39e−10	2.18e−01	2.94e−01	3.11e−14	0.00e+00	8.67e−14
F25	4	1.21e−01	1.21e−01	1.91e−11	1.91e−11	7.04e+00	4.15e−07	3.85e−01	3.29e+00	9.56e−02	2.50e+01	2.65e−02	1.98e−01	1.98e−01
F26	4	2.47e−01	2.48e−01	2.17e−01	2.17e−01	1.35e+00	2.27e−01	2.67e−01	2.39e−01	2.67e−01	2.82e−01	2.17e−01	2.38e−01	2.60e−01
F27	5	-9.64e−01	-9.64e−01	-4.82e−01	-4.69e−01	-7.60e−02	-9.64e−01	-5.01e−01	-5.13e−01	-5.13e−01	-5.10e−01	-4.97e−01	-9.08e−01	-9.08e−01
F27	10	-2.74e−01	-2.74e−01	-2.63e−03	-2.63e−03	-2.63e−03	-4.00e−02	-5.13e−01	-1.12e−02	-1.45e−01	-4.92e−03	-2.26e−04	-5.13e−01	-5.13e−01
F28	5	-3.72e+00	-3.72e+00	-2.79e+00	-2.79e+00	-2.94e+00	-4.12e+00	-4.64e+00	-3.82e+00	-4.64e+00	-3.24e+00	-4.49e+00	-3.84e+00	-3.84e+00
F28	10	-7.66e+00	-7.66e+00	-7.38e+00	-5.93e+00	-3.32e+00	-4.11e+00	-7.55e+00	-6.65e+00	-6.47e+00	-5.29e+00	-5.87e+00	-9.03e+00	-9.03e+00
F29	5	-4.09e−01	-4.09e−01	-5.00e−01	-2.86e−01	-2.55e−01	-8.47e−01	-7.27e−01	-8.98e−01	-8.53e−01	-5.77e−01	-4.05e−01	-1.27e−01	-1.27e−01
F29	10	-1.54e−02	-1.54e−02	-3.83e−02	-3.83e−02	-1.43e−02	-1.37e−01	-4.07e−01	-7.72e−01	-1.26e−01	-8.92e−02	-1.22e−01	-2.66e−02	-2.66e−02
F30	9	3.40e+03	3.40e+03	3.94e+00	3.94e+00	1.04e+04	4.95e+05	8.66e+03	7.22e+02	4.55e+03	7.08e+04	1.56e+04	2.22e+03	2.22e+03
F30	17	2.68e+05	2.68e+05	5.27e+11	5.27e+11	1.53e+12	1.14e+08	6.05e+06	8.50e+10	3.76e+08	5.35e+11	1.47e+06	4.17e+05	4.17e+05

Tabelle 6.4: Die besten gefundenen relativen Funktionswerte

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
F1	3	1.89e-17	2.79e-14	4.06e-20	4.06e-20	1.32e-02	0.00e+00	1.78e-08	6.20e-08	6.22e-07	7.22e+00	7.49e-09	4.74e-17	4.74e-17
F2	2	1.41e-03	4.28e-01	1.58e-09	1.58e-09	2.10e+00	0.00e+00	0.00e+00	2.28e-04	3.67e-03	1.71e-01	8.15e-11	1.19e-09	1.52e-03
F2	5	1.15e-01	6.98e-02	0.00e+00	0.00e+00	9.23e-04	1.87e-07	6.28e-01	8.05e-01	3.80e-01	2.39e+00	2.02e-01	7.57e-02	7.57e-02
F2	10	2.86e-03	2.86e-03	0.00e+00	0.00e+00	8.67e-03	4.14e-01	7.89e-03	8.53e-03	5.05e-02	4.96e+00	4.90e-03	1.43e-04	1.43e-04
F3	5	0.00e+00	0.00e+00	2.75e+00	2.75e+00	2.75e+00	0.00e+00	0.00e+00	5.00e-01	0.00e+00	1.25e+00	0.00e+00	0.00e+00	0.00e+00
F4	30	4.50e-02	4.50e-02	0.00e+00	0.00e+00	2.54e+01	5.90e-01	2.96e-02	1.53e-02	1.48e-02	3.24e+00	2.94e-03	4.58e-02	4.58e-02
F5	5	4.55e-06	4.55e-06	6.82e-06	6.82e-06	3.13e-01	0.00e+00	9.69e-04	1.42e-01	3.41e+00	3.11e+00	6.37e-05	4.55e-06	4.55e-06
F5	10	3.20e-03	3.20e-03	0.00e+00	0.00e+00	5.24e+00	1.79e+00	1.30e-01	1.21e-01	1.25e-01	2.36e+00	5.38e-05	7.77e-04	7.77e-04
F5	15	9.06e-02	9.06e-02	0.00e+00	0.00e+00	3.82e+00	1.14e+00	4.21e-02	2.26e-02	7.57e-02	2.28e+00	6.45e-03	5.51e-02	5.51e-02
F5	20	1.05e-01	1.05e-01	0.00e+00	0.00e+00	3.30e+00	2.54e+00	8.59e-03	2.28e-02	1.66e-02	2.35e+00	1.69e-02	3.79e-02	3.79e-02
F5	25	6.68e-02	6.68e-02	0.00e+00	0.00e+00	2.32e+00	6.46e-01	8.95e-02	1.47e-02	2.47e-02	2.40e+00	2.29e-02	8.54e-02	8.54e-02
F5	30	1.03e-01	1.03e-01	0.00e+00	0.00e+00	1.70e+00	2.23e+00	4.03e-02	2.52e-02	4.26e-02	1.97e+00	1.87e-02	1.16e-01	1.16e-01
F6	2	3.94e-11	7.77e-08	1.59e-03	1.59e-03	4.53e-01	0.00e+00	0.00e+00	2.77e-05	4.75e-05	4.26e+00	2.39e-10	1.79e-10	2.65e-06
F7-5	4	9.99e-01	9.99e-01	0.00e+00	6.76e-01	6.78e-01	0.00e+00	0.00e+00	4.01e-06	9.99e-01	1.00e+00	1.34e-06	6.76e-01	6.76e-01
F7-7	4	9.99e-01	9.99e-01	0.00e+00	6.89e-01	6.91e-01	0.00e+00	6.94e-01	4.70e-05	9.98e-01	9.88e-01	6.89e-01	6.89e-01	6.89e-01
F7-10	4	0.00e+00	0.00e+00	0.00e+00	9.49e-01	9.52e-01	0.00e+00	1.37e+00	9.77e-01	3.35e-04	1.34e+00	0.00e+00	9.49e-01	9.49e-01
F8	3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	0.00e+00	0.00e+00	0.00e+00	6.93e-02	9.32e+01	2.57e-01	0.00e+00	0.00e+00
F8	6	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.98e-01	8.38e-01	8.38e-01	5.06e-03	8.38e-01	1.75e+00	9.85e-05	0.00e+00	0.00e+00
F9	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.27e-01	0.00e+00	0.00e+00	0.00e+00	1.50e-03	1.20e+00	0.00e+00	0.00e+00	0.00e+00
F10	2	6.10e-05	6.10e-05	0.00e+00	0.00e+00	1.93e-02	0.00e+00	0.00e+00	0.00e+00	5.19e-03	2.54e-01	6.10e-05	6.10e-05	6.10e-05
F11	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.93e-02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	4.72e+00	0.00e+00	0.00e+00	0.00e+00
F12	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.12e-04	0.00e+00	0.00e+00	2.31e-03	0.00e+00	5.47e-04	0.00e+00	0.00e+00	0.00e+00
F13	2	0.00e+00	0.00e+00	2.00e+00	5.94e+00	5.94e+00	0.00e+00	0.00e+00	4.70e-04	1.00e+00	2.31e-02	2.00e+00	0.00e+00	0.00e+00
F14	5	9.82e-01	9.82e-01	8.80e-01	8.80e-01	8.82e-01	9.17e-01	0.00e+00	9.82e-01	8.74e-01	1.05e+00	7.84e-02	9.04e-01	9.04e-01
F14	10	9.01e-01	9.01e-01	0.00e+00	0.00e+00	1.02e+00	9.64e-01	9.06e-01	9.07e-01	9.01e-01	1.02e+00	8.08e-01	9.01e-01	9.01e-01
F15	4	0.00e+00	0.00e+00	1.31e+01	1.31e+01	2.66e+01	3.11e-07	0.00e+00	3.72e-05	8.16e-04	3.00e+00	7.51e+00	0.00e+00	0.00e+00
F16	10	2.61e-02	2.61e-02	9.78e-03	9.78e-03	0.00e+00	2.07e-01	2.64e-02	9.46e-02	3.74e-02	3.08e+00	7.05e-03	9.88e-03	9.88e-03
F17	2	6.87e-03	6.88e-03	7.25e-03	7.25e-03	7.40e-03	0.00e+00	6.87e-03	6.09e-05	1.16e-05	1.59e+00	1.62e-04	2.75e-12	1.57e-04
F18	10	4.85e-03	4.85e-03	3.44e-02	1.72e-01	4.23e-01	2.12e-01	0.00e+00	1.61e-01	1.05e-03	1.01e-01	1.05e-02	4.43e-03	4.43e-03
F19	20	5.38e-03	5.38e-03	1.52e+00	1.52e+00	1.13e+01	3.97e-01	0.00e+00	7.03e-03	4.55e-04	2.85e+00	3.71e-03	4.60e-03	4.60e-03
F20	30	4.92e-01	4.92e-01	1.82e+00	2.18e+00	2.42e+00	1.82e+00	2.47e-01	0.00e+00	3.39e-02	1.61e+00	7.97e-01	4.95e-01	4.95e-01
F21	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.07e-03	0.00e+00	0.00e+00	1.99e-06	1.16e-08	1.49e-13	1.64e-16	0.00e+00	0.00e+00
F22	2	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	0.00e+00	0.00e+00	8.40e-03	2.00e-07	1.97e-01	5.34e-01	0.00e+00	1.00e+00
F23	2	0.00e+00	9.88e-14	7.61e-01	7.61e-01	8.94e-01	0.00e+00	0.00e+00	5.66e-12	1.63e+00	1.15e+00	0.00e+00	0.00e+00	3.80e-12
F24	2	0.00e+00	3.91e-16	7.68e-01	7.68e-01	8.04e-01	0.00e+00	5.11e-13	4.91e-10	7.68e-01	1.04e+00	1.10e-13	0.00e+00	3.05e-13
F25	4	2.56e-02	2.56e-02	0.00e+00	0.00e+00	1.49e+00	8.79e-08	8.17e-02	6.97e-01	2.02e-02	5.29e+00	5.62e-03	4.19e-02	4.19e-02
F26	4	7.36e-02	7.58e-02	0.00e+00	9.74e-07	2.76e+00	2.30e-02	1.22e-01	5.35e-02	1.21e-01	1.57e-01	9.74e-05	4.93e-02	1.05e-01
F27	5	0.00e+00	0.00e+00	5.00e-01	5.14e-01	9.22e-01	5.37e-05	4.80e-01	4.68e-01	4.68e-01	4.72e-01	4.84e-01	5.91e-02	5.91e-02
F27	10	4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	9.22e-01	0.00e+00	9.78e-01	7.17e-01	9.90e-01	1.00e+00	1.95e-07	1.95e-07
F28	5	4.77e-01	4.77e-01	9.60e-01	9.60e-01	8.77e-01	2.67e-01	0.00e+00	4.22e-01	4.24e-05	7.23e-01	7.81e-02	4.14e-01	4.14e-01
F28	10	3.55e-01	3.55e-01	4.26e-01	7.99e-01	1.47e+00	1.27e+00	3.83e-01	6.14e-01	6.61e-01	9.65e-01	8.15e-01	0.00e+00	0.00e+00
F29	5	6.26e-01	6.26e-01	5.09e-01	7.83e-01	8.22e-01	6.59e-02	2.19e-01	0.00e+00	5.79e-02	4.11e-01	6.31e-01	9.87e-01	9.87e-01
F29	10	9.86e-01	9.86e-01	9.56e-01	9.56e-01	9.87e-01	8.26e-01	4.75e-01	0.00e+00	8.41e-01	8.89e-01	8.46e-01	9.71e-01	9.71e-01
F30	9	8.58e-03	8.58e-03	0.00e+00	0.00e+00	2.64e-02	1.25e+00	2.19e-02	1.81e-03	1.15e-02	1.79e-01	3.95e-02	5.61e-03	5.61e-03
F30	17	0.00e+00	0.00e+00	2.69e+01	2.69e+01	7.83e+01	5.85e-03	2.95e-04	4.33e+00	1.92e-02	2.73e+01	6.14e-05	7.61e-06	7.61e-06

Tabelle 6.5: Die besten relativen Funktionswerte bei Anwendung des Abbruchkriteriums A1 (mindestens 156 Funktionsauswertungen)

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
F1	3	1.59e+01	1.59e+01	4.06e-20	4.06e-20	3.85e-01	3.04e+02	2.36e+02	2.87e+02	2.05e+02	9.41e+02	1.03e+02	4.74e-17	4.46e-12
F2	2	4.24e+00	4.24e+00	1.58e-09	1.58e-09	2.10e+00	3.16e+00	*0.00e+00	1.12e+01	1.50e+01	1.93e-01	1.57e+01	2.07e+01	2.07e+01
F2	5	*1.15e-01	*6.98e-02	0.00e+00	0.00e+00	9.23e-04	2.84e+01	9.79e+00	2.33e+00	1.29e+01	9.33e+00	2.72e+01	*7.57e-02	*7.57e-02
F2	10	*2.86e-03	*2.86e-03	0.00e+00	0.00e+00	1.56e-02	1.39e+01	*7.89e-03	1.01e+01	2.20e+01	6.10e+00	4.43e+00	*1.43e-04	*1.43e-04
F3	5	0.00e+00	2.50e-01	2.75e+00	2.75e+00	2.75e+00	0.00e+00	2.75e+00	2.75e+00	0.00e+00	2.25e+00	0.00e+00	0.00e+00	0.00e+00
F4	30	*4.50e-02	*4.50e-02	0.00e+00	0.00e+00	0.00e+00	5.19e+00	*2.96e-02	6.34e-01	*1.48e-02	5.56e+00	2.86e+00	*4.58e-02	*4.58e-02
F5	5	4.55e-06	4.55e-06	6.82e-06	9.10e-06	3.13e-01	4.18e+01	1.48e+01	2.54e+01	*3.41e+00	4.14e+01	1.12e+01	4.55e-06	4.55e-06
F5	10	*3.20e-03	*3.20e-03	1.29e-06	1.29e-06	5.24e+00	7.07e+00	*1.30e-01	4.94e+00	*1.25e-01	3.19e+00	5.49e-01	*7.77e-04	*7.77e-04
F5	15	*9.06e-02	*9.06e-02	0.00e+00	0.00e+00	0.00e+00	4.03e+00	*4.21e-02	*2.26e-02	*7.57e-02	3.95e+00	1.85e-01	*5.51e-02	*5.51e-02
F5	20	1.44e-01	1.44e-01	2.17e-07	2.17e-07	2.17e-07	4.85e+00	3.98e+00	4.31e+00	*1.66e-02	3.99e+00	1.08e-01	*3.79e-02	*3.79e-02
F5	25	*6.68e-02	*6.68e-02	4.76e-05	4.76e-05	4.76e-05	4.15e+00	*8.95e-02	*1.47e-02	3.83e+00	4.29e+00	7.37e-02	7.79e-01	7.79e-01
F5	30	*1.03e-01	*1.03e-01	0.00e+00	6.27e-04	6.27e-04	4.37e+00	*4.03e-02	4.66e-01	4.00e+00	3.68e+00	7.08e-02	*1.16e-01	*1.16e-01
F6	2	2.85e+00	2.85e+00	1.59e-03	1.59e-03	1.46e+00	2.19e+01	3.94e+01	7.93e+01	6.32e+01	3.74e+01	2.81e+00	*1.79e-10	3.04e-06
F7-5	4	9.99e-01	9.99e-01	6.76e-01	6.76e-01	6.78e-01	1.27e+00	1.28e+00	1.20e+00	1.25e+00	1.27e+00	1.24e+00	6.76e-01	6.76e-01
F7-7	4	9.99e-01	9.99e-01	6.89e-01	6.89e-01	6.91e-01	1.26e+00	1.27e+00	1.20e+00	1.25e+00	1.25e+00	1.24e+00	6.89e-01	6.89e-01
F7-10	4	0.00e+00	0.00e+00	9.49e-01	9.49e-01	9.52e-01	1.70e+00	*1.37e+00	1.44e+00	*3.35e-04	1.68e+00	1.67e+00	9.49e-01	9.49e-01
F8	3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	1.72e+03	1.23e+03	8.66e+03	3.27e-01	1.32e+03	2.01e+02	9.90e-03	9.90e-03
F8	6	0.00e+00	0.00e+00	0.00e+00	4.31e-03	2.98e-01	9.36e+00	1.01e+01	9.70e+00	*8.38e-01	4.10e+00	7.15e+00	0.00e+00	0.00e+00
F9	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.27e-01	6.94e+01	2.95e+02	4.26e+02	*1.50e-03	1.57e+02	1.51e+03	0.00e+00	0.00e+00
F10	2	6.10e-05	6.10e-05	0.00e+00	0.00e+00	1.93e-02	1.70e+02	1.20e+02	2.77e+01	1.36e+02	4.66e+01	1.89e+00	6.10e-05	6.10e-05
F11	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.93e-02	1.94e+03	4.83e+00	5.90e+00	3.14e+02	8.56e+02	2.04e+01	0.00e+00	0.00e+00
F12	2	0.00e+00	0.00e+00	0.00e+00	1.20e-02	2.16e-03	1.05e-01	1.09e-01	4.04e-01	3.27e-01	4.96e-01	8.75e-01	0.00e+00	0.00e+00
F13	2	0.00e+00	0.00e+00	2.99e+00	5.94e+00	1.08e+01	0.00e+00	9.23e+01	1.89e-03	1.00e+00	6.86e-02	1.17e+01	0.00e+00	0.00e+00
F14	5	9.82e-01	9.82e-01	8.80e-01	8.80e-01	1.03e+00	1.22e+00	1.20e+00	9.82e-01	1.18e+00	1.22e+00	1.14e+00	9.04e-01	9.04e-01
F14	10	*9.01e-01	*9.01e-01	0.00e+00	0.00e+00	1.02e+00	1.03e+00	*9.06e-01	1.02e+00	*9.01e-01	1.02e+00	1.02e+00	*9.01e-01	*9.01e-01
F15	4	0.00e+00	0.00e+00	1.31e+01	1.31e+01	2.66e+01	3.33e+01	*0.00e+00	2.14e+01	*8.16e-04	3.01e+01	7.41e+01	0.00e+00	0.00e+00
F16	10	*2.61e-02	*2.61e-02	9.78e-03	9.78e-03	0.00e+00	8.36e+00	*2.64e-02	*9.46e-02	*3.74e-02	4.49e+00	6.86e-01	*9.88e-03	*9.88e-03
F17	2	6.89e-03	6.89e-03	7.25e-03	7.25e-03	7.40e-03	1.28e+01	*6.87e-03	1.37e+01	5.88e-05	9.04e+00	1.06e-01	*2.75e-12	1.57e-04
F18	10	1.34e-01	1.34e-01	3.44e-02	1.72e-01	4.23e-01	3.46e-01	7.44e-01	5.19e-01	*1.05e-03	6.43e-01	3.18e+00	*4.43e-03	*4.43e-03
F19	20	*5.38e-03	*5.38e-03	1.52e+00	1.52e+00	1.52e+00	5.22e+00	*0.00e+00	3.75e-01	5.43e+00	4.27e+00	1.47e+00	*4.60e-03	*4.60e-03
F20	30	5.91e-01	5.91e-01	2.18e+00	2.18e+00	2.18e+00	1.82e+00	*2.47e-01	1.70e+00	1.79e+00	1.81e+00	7.97e-01	1.34e+00	1.34e+00
F21	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.07e-03	1.21e-02	8.78e-05	2.29e-03	8.95e-05	1.34e-03	1.02e-06	0.00e+00	0.00e+00
F22	2			1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	9.81e-01	*2.00e-07	1.00e+00	1.00e+00	1.00e+00	1.00e+00
F23	2	1.93e+00	1.93e+00	7.61e-01	7.61e-01	8.94e-01	8.28e+00	9.73e+01	5.33e+01	1.63e+00	1.26e+01	8.66e-01	0.00e+00	3.87e-12
F24	2	1.87e+00	1.87e+00	7.68e-01	7.68e-01	8.04e-01	1.39e+01	*5.11e-13	1.34e+02	*7.68e-01	2.03e+01	9.05e-01	0.00e+00	3.06e-13
F25	4	*2.56e-02	*2.56e-02	0.00e+00	0.00e+00	2.66e+00	2.05e+02	1.13e+02	4.03e+02	6.85e+02	9.17e+01	2.51e+00	*4.20e-02	*4.20e-02
F26	4	6.03e-01	6.03e-01	2.74e-04	2.74e-04	2.76e+00	1.05e+01	1.80e+01	5.35e-02	2.61e+01	4.51e+00	1.61e+02	5.43e+00	5.43e+00
F27	5	1.04e-07	1.04e-07	5.14e-01	5.14e-01	9.22e-01	8.69e-01	9.89e-01	8.59e-01	4.69e-01	9.22e-01	9.23e-01	5.91e-02	5.91e-02
F27	10	4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	1.00e+00	1.00e+00	1.00e+00	1.00e+00	9.90e-01	1.00e+00	1.95e-07	1.95e-07
F28	5	4.77e-01	4.77e-01	9.60e-01	9.60e-01	9.62e-01	1.31e+00	1.28e+00	1.49e+00	1.25e+00	1.06e+00	9.89e-01	4.14e-01	4.14e-01
F28	10	*3.55e-01	*3.55e-01	6.89e-01	8.78e-01	1.47e+00	1.69e+00	1.33e+00	8.84e-01	*6.61e-01	1.36e+00	1.67e+00	*0.00e+00	*0.00e+00
F29	5	*6.26e-01	*6.26e-01	5.09e-01	7.83e-01	8.46e-01	8.34e-01	9.87e-01	7.84e-01	5.75e-01	6.93e-01	6.87e-01	*9.87e-01	*9.87e-01
F29	10	*9.86e-01	*9.86e-01	9.56e-01	9.56e-01	9.87e-01	9.99e-01	9.86e-01	*0.00e+00	9.39e-01	9.67e-01	8.54e-01	*9.71e-01	*9.71e-01
F30	9	*8.58e-03	*8.58e-03	0.00e+00	5.64e-03	2.64e-02	2.06e+00	1.21e+00	3.10e+00	2.98e+00	2.61e+00	1.33e+01	*5.61e-03	*5.61e-03
F30	17	*0.00e+00	*0.00e+00	2.69e+01	2.69e+01	2.69e+01	1.21e-02	*2.95e-04	5.56e+01	*1.92e-02	5.48e+01	4.47e+01	*7.61e-06	*7.61e-06

Tabelle 6.6: Die besten relativen Funktionswerte bei Anwendung des Abbruchkriteriums A2 (mindestens 312 Funktionsauswertungen)

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
F1	3	3.33e-06	3.33e-06	4.06e-20	4.06e-20	4.06e-20	3.04e+02	*1.78e-08	9.40e+00	*6.22e-07	3.11e+02	8.90e+02	4.74e-17	4.46e-12
F2	2	*1.41e-03	4.51e-01	1.58e-09	1.58e-09	2.10e+00	3.16e+00	*0.00e+00	1.12e+01	1.50e+01	1.93e-01	3.01e+01	*1.19e-09	1.54e-03
F2	5	*1.15e-01	*6.98e-02	0.00e+00	0.00e+00	9.23e-04	2.84e+01	*6.28e-01	2.33e+00	*3.80e-01	9.33e+00	2.72e+01	*7.57e-02	*7.57e-02
F2	10	*2.86e-03	*2.86e-03	0.00e+00	0.00e+00	0.00e+00	1.39e+01	*7.89e-03	*8.53e-03	*5.05e-02	6.10e+00	4.43e+00	*1.43e-04	*1.43e-04
F3	5	0.00e+00	2.50e-01	2.75e+00	2.75e+00	2.75e+00	0.00e+00	2.00e+00	2.75e+00	0.00e+00	2.25e+00	0.00e+00	0.00e+00	0.00e+00
F4	30	*4.50e-02	*4.50e-02	0.00e+00	0.00e+00	0.00e+00	3.42e+00	*2.96e-02	*1.53e-02	*1.48e-02	3.24e+00	2.86e+00	*4.58e-02	*4.58e-02
F5	5	4.55e-06	4.55e-06	6.82e-06	6.82e-06	3.13e-01	4.18e+01	1.48e+01	3.75e+00	*3.41e+00	1.92e+01	1.12e+01	4.55e-06	4.55e-06
F5	10	*3.20e-03	*3.20e-03	1.29e-06	1.29e-06	1.29e-06	2.88e+00	*1.30e-01	5.58e-01	*1.25e-01	3.19e+00	1.07e+00	*7.77e-04	*7.77e-04
F5	15	*9.06e-02	*9.06e-02	0.00e+00	0.00e+00	0.00e+00	4.03e+00	*4.21e-02	*2.26e-02	*7.57e-02	3.95e+00	2.75e-01	*5.51e-02	*5.51e-02
F5	20	1.44e-01	1.44e-01	2.17e-07	2.17e-07	2.17e-07	4.85e+00	*8.59e-03	3.18e-01	*1.66e-02	3.99e+00	1.56e-01	*3.79e-02	*3.79e-02
F5	25	*6.68e-02	*6.68e-02	4.76e-05	4.76e-05	4.76e-05	4.15e+00	*8.95e-02	*1.47e-02	*2.47e-02	2.40e+00	1.20e-01	7.79e-01	7.79e-01
F5	30	*1.03e-01	*1.03e-01	0.00e+00	6.27e-04	6.27e-04	4.24e+00	*4.03e-02	*2.52e-02	*4.26e-02	2.98e+00	9.20e-02	*1.16e-01	*1.16e-01
F6	2	1.60e-09	6.44e-07	1.59e-03	1.59e-03	1.59e-03	2.19e+01	3.94e+01	1.23e+00	*4.75e-05	3.74e+01	2.81e+00	*1.79e-10	3.04e-06
F7-5	4	9.99e-01	9.99e-01	6.76e-01	6.76e-01	6.78e-01	1.27e+00	1.23e+00	1.20e+00	*9.99e-01	1.27e+00	1.24e+00	6.76e-01	6.76e-01
F7-7	4	9.99e-01	9.99e-01	6.89e-01	6.89e-01	6.91e-01	1.26e+00	*6.94e-01	6.83e-01	*9.98e-01	1.25e+00	1.24e+00	6.89e-01	6.89e-01
F7-10	4	0.00e+00	0.00e+00	9.49e-01	9.49e-01	9.52e-01	1.70e+00	*1.37e+00	1.44e+00	*3.35e-04	1.68e+00	1.67e+00	9.49e-01	9.49e-01
F8	3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	1.72e+03	1.23e+03	0.00e+00	3.27e-01	1.32e+03	1.61e+03	9.90e-03	9.90e-03
F8	6	0.00e+00	0.00e+00	0.00e+00	4.31e-03	2.98e-01	9.36e+00	*8.38e-01	8.91e+00	*8.38e-01	4.10e+00	7.15e+00	0.00e+00	0.00e+00
F9	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.94e+01	2.95e+02	4.26e+02	*1.50e-03	1.57e+02	1.51e+03	0.00e+00	0.00e+00
F10	2	6.10e-05	6.10e-05	0.00e+00	0.00e+00	1.93e-02	1.70e+02	1.20e+02	2.77e+01	1.36e+02	4.66e+01	1.89e+00	6.10e-05	6.10e-05
F11	2	1.04e-01	1.04e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	4.83e+00	5.90e+00	5.45e-02	8.56e+02	2.04e+01	0.00e+00	0.00e+00
F12	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.05e-01	1.09e-01	4.04e-01	1.77e-06	5.71e-02	9.15e-01	0.00e+00	0.00e+00
F13	2	0.00e+00	0.00e+00	2.99e+00	5.94e+00	5.94e+00	0.00e+00	0.00e+00	1.89e-03	1.00e+00	6.86e-02	1.17e+01	0.00e+00	0.00e+00
F14	5	9.82e-01	9.82e-01	8.80e-01	8.80e-01	1.03e+00	1.17e+00	1.20e+00	*9.82e-01	1.18e+00	1.05e+00	1.14e+00	9.04e-01	9.04e-01
F14	10	*9.01e-01	*9.01e-01	0.00e+00	0.00e+00	0.00e+00	1.03e+00	*9.06e-01	1.02e+00	*9.01e-01	1.02e+00	1.02e+00	*9.01e-01	*9.01e-01
F15	4	0.00e+00	0.00e+00	1.31e+01	1.31e+01	1.31e+01	*3.11e-07	*0.00e+00	2.14e+01	*8.16e-04	3.01e+01	7.41e+01	0.00e+00	0.00e+00
F16	10	*2.61e-02	*2.61e-02	9.78e-03	9.78e-03	0.00e+00	8.36e+00	*2.64e-02	*9.46e-02	*3.74e-02	4.49e+00	3.05e+00	*9.88e-03	*9.88e-03
F17	2	6.89e-03	6.89e-03	7.25e-03	7.25e-03	7.25e-03	1.28e+01	*6.87e-03	1.37e+01	5.88e-05	9.04e+00	4.70e+00	*2.75e-12	1.57e-04
F18	10	1.34e-01	1.34e-01	3.44e-02	1.72e-01	1.72e-01	3.46e-01	0.00e+00	5.19e-01	*1.05e-03	6.43e-01	1.08e+00	*4.43e-03	*4.43e-03
F19	20	*5.38e-03	*5.38e-03	1.52e+00	1.52e+00	1.52e+00	5.22e+00	*0.00e+00	3.75e-01	*4.55e-04	4.27e+00	1.47e+00	*4.60e-03	*4.60e-03
F20	30	5.91e-01	5.91e-01	2.18e+00	2.18e+00	2.18e+00	1.82e+00	*2.47e-01	6.49e-01	*3.39e-02	1.81e+00	7.97e-01	*4.95e-01	*4.95e-01
F21	2	5.26e-10	5.26e-10	0.00e+00	0.00e+00	0.00e+00	1.21e-02	8.78e-05	2.29e-03	8.95e-05	1.34e-03	1.02e-06	0.00e+00	0.00e+00
F22	2			1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	9.81e-01	*2.00e-07	1.00e+00	1.00e+00	1.00e+00	1.00e+00
F23	2	4.07e-13	4.07e-13	7.61e-01	7.61e-01	7.61e-01	8.28e+00	*0.00e+00	1.29e+00	1.63e+00	1.26e+01	8.66e-01	0.00e+00	3.87e-12
F24	2	3.09e-14	3.09e-14	7.68e-01	7.68e-01	7.68e-01	1.39e+01	*5.11e-13	1.69e+01	*7.68e-01	2.03e+01	9.05e-01	0.00e+00	3.06e-13
F25	4	*2.56e-02	*2.56e-02	0.00e+00	0.00e+00	2.66e+00	1.04e+02	5.29e+01	1.54e+01	*2.02e-02	9.17e+01	3.35e+00	*4.20e-02	*4.20e-02
F26	4	6.03e-01	6.03e-01	2.74e-04	2.74e-04	2.76e+00	1.05e+01	1.80e+01	5.35e-02	4.09e-01	4.51e+00	1.61e+02	5.43e+00	5.43e+00
F27	5	1.04e-07	1.04e-07	5.14e-01	5.14e-01	5.14e-01	8.69e-01	4.80e-01	7.69e-01	4.68e-01	9.22e-01	9.23e-01	5.91e-02	5.91e-02
F27	10	4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	1.00e+00	1.00e+00	1.00e+00	1.00e+00	9.90e-01	1.00e+00	1.95e-07	1.95e-07
F28	5	4.77e-01	4.77e-01	9.60e-01	9.60e-01	9.60e-01	1.29e+00	1.28e+00	9.93e-01	*4.24e-05	1.06e+00	1.01e+00	4.14e-01	4.14e-01
F28	10	*3.55e-01	*3.55e-01	6.89e-01	8.78e-01	8.78e-01	1.69e+00	1.33e+00	8.84e-01	*6.61e-01	1.36e+00	1.67e+00	*0.00e+00	*0.00e+00
F29	5	*6.26e-01	*6.26e-01	5.09e-01	7.83e-01	7.83e-01	8.34e-01	9.87e-01	7.84e-01	5.75e-01	6.93e-01	6.87e-01	*9.87e-01	*9.87e-01
F29	10	*9.86e-01	*9.86e-01	9.56e-01	9.56e-01	9.56e-01	9.99e-01	*4.75e-01	*0.00e+00	9.39e-01	9.67e-01	8.54e-01	*9.71e-01	*9.71e-01
F30	9	*8.58e-03	*8.58e-03	0.00e+00	5.64e-03	5.64e-03	1.92e+00	1.21e+00	3.03e-01	*1.15e-02	2.61e+00	1.33e+01	*5.61e-03	*5.61e-03
F30	17	*0.00e+00	*0.00e+00	2.69e+01	2.69e+01	2.69e+01	1.21e-02	*2.95e-04	5.56e+01	*1.92e-02	5.48e+01	4.47e+01	*7.61e-06	*7.61e-06

Tabelle 6.7: Die besten relativen Funktionswerte nach 1 000, 2 500, 5 000, 10 000 beziehungsweise 20 000 Auswertungen

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
		6.67e-05	6.67e-05	4.06e-20	*4.06e-20	*1.32e-02	7.10e+01	1.55e+01	9.40e+00	3.40e+01	2.00e+02	2.44e+00	2.01e-06	2.01e-06
		2.84e-13	2.84e-13	4.06e-20	*4.06e-20	*1.32e-02	9.48e-01	2.72e-01	8.45e+00	7.88e+00	1.62e+02	1.02e-05	4.46e-12	4.46e-12
F1	3	1.36e-15	*2.79e-14	4.06e-20	*4.06e-20	*1.32e-02	4.61e-04	8.78e-02	6.84e-01	6.69e-01	7.22e+00	6.77e-08	4.74e-17	*4.74e-17
		1.11e-16	*2.79e-14	*4.06e-20	*4.06e-20	*1.32e-02	1.04e-10	1.98e-05	7.42e-04	2.96e-03	7.22e+00	7.49e-09	4.74e-17	*4.74e-17
		1.89e-17	*2.79e-14	*4.06e-20	*4.06e-20	*1.32e-02	0.00e+00	1.78e-08	6.20e-08	6.22e-07	7.22e+00	7.49e-09	4.74e-17	*4.74e-17
		4.61e-01	4.61e-01	1.58e-09	*1.58e-09	*2.10e+00	2.53e+00	1.43e+00	9.10e+00	4.41e-01	1.93e-01	1.07e+01	2.43e-03	2.43e-03
		3.75e-01	*4.28e-01	1.58e-09	*1.58e-09	*2.10e+00	1.84e-02	8.06e-03	2.66e-01	1.91e-01	1.93e-01	7.98e-03	6.12e-04	*1.52e-03
F2	2	1.65e-01	*4.28e-01	1.58e-09	*1.58e-09	*2.10e+00	8.06e-09	2.84e-06	2.66e-01	1.91e-01	1.93e-01	7.98e-03	2.03e-04	*1.52e-03
		2.21e-02	*4.28e-01	*1.58e-09	*1.58e-09	*2.10e+00	0.00e+00	1.53e-10	1.01e-01	3.67e-03	1.93e-01	5.19e-09	4.99e-08	*1.52e-03
		1.41e-03	*4.28e-01	*1.58e-09	*1.58e-09	*2.10e+00	0.00e+00	0.00e+00	2.28e-04	3.67e-03	1.71e-01	8.15e-11	1.19e-09	*1.52e-03
		6.77e-01	3.78e-01	0.00e+00	*0.00e+00	*9.23e-04	8.33e+00	2.60e+00	2.33e+00	1.81e+00	9.33e+00	8.54e-01	6.03e-01	6.03e-01
		6.26e-01	3.30e-01	0.00e+00	*0.00e+00	*9.23e-04	1.86e+00	7.90e-01	1.84e+00	5.95e-01	3.17e+00	7.43e-01	4.93e-01	4.93e-01
F2	5	5.36e-01	2.85e-01	0.00e+00	*0.00e+00	*9.23e-04	9.52e-01	7.26e-01	8.76e-01	4.74e-01	3.17e+00	5.99e-01	3.57e-01	3.57e-01
		3.39e-01	1.67e-01	*0.00e+00	*0.00e+00	*9.23e-04	9.88e-02	6.76e-01	8.61e-01	4.32e-01	2.39e+00	3.01e-01	1.96e-01	1.96e-01
		1.15e-01	6.98e-02	*0.00e+00	*0.00e+00	*9.23e-04	1.87e-07	6.28e-01	8.05e-01	3.80e-01	2.39e+00	2.02e-01	7.57e-02	7.57e-02
		1.79e-01	1.79e-01	8.84e-10	8.84e-10	*8.67e-03	1.16e+01	3.54e-01	8.14e-01	5.54e-01	6.10e+00	2.47e-01	7.25e-02	7.25e-02
		1.76e-02	1.76e-02	0.00e+00	*0.00e+00	*8.67e-03	1.16e+01	1.13e-01	6.62e-02	3.39e-01	4.96e+00	1.06e-02	1.78e-02	1.78e-02
F2	10	3.80e-03	3.80e-03	0.00e+00	*0.00e+00	*8.67e-03	2.93e+00	6.63e-02	3.12e-02	1.78e-01	4.96e+00	6.32e-03	4.93e-03	4.93e-03
		3.09e-03	3.09e-03	*0.00e+00	*0.00e+00	*8.67e-03	1.13e+00	8.56e-03	1.07e-02	1.21e-01	4.96e+00	5.68e-03	2.32e-04	2.32e-04
		2.86e-03	2.86e-03	*0.00e+00	*0.00e+00	*8.67e-03	4.14e-01	7.89e-03	8.53e-03	5.05e-02	4.96e+00	4.90e-03	1.43e-04	1.43e-04
		0.00e+00	*0.00e+00	2.75e+00	*2.75e+00	*2.75e+00	1.60e+00	1.25e+00	1.25e+00	2.50e-01	2.25e+00	0.00e+00	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	2.75e+00	*2.75e+00	*2.75e+00	6.34e-01	5.00e-01	1.25e+00	0.00e+00	2.00e+00	0.00e+00	0.00e+00	0.00e+00
F3	5	0.00e+00	*0.00e+00	2.75e+00	*2.75e+00	*2.75e+00	1.46e-01	0.00e+00	7.50e-01	0.00e+00	2.00e+00	0.00e+00	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	*2.75e+00	*2.75e+00	*2.75e+00	0.00e+00	0.00e+00	7.50e-01	0.00e+00	1.50e+00	0.00e+00	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	*2.75e+00	*2.75e+00	*2.75e+00	0.00e+00	0.00e+00	5.00e-01	0.00e+00	1.25e+00	0.00e+00	0.00e+00	0.00e+00
		3.05e-01	3.05e-01	1.48e+01	1.48e+01	*2.54e+01	3.42e+00	6.81e-01	5.78e-01	6.39e-01	3.79e+00	1.27e+00	3.22e-01	3.22e-01
		1.12e-01	1.12e-01	5.36e+00	5.36e+00	*2.54e+01	3.42e+00	4.52e-01	2.49e-01	3.85e-01	3.78e+00	3.97e-01	1.70e-01	1.70e-01
F4	30	8.26e-02	8.26e-02	6.33e-07	6.33e-07	*2.54e+01	3.42e+00	2.38e-01	1.46e-01	2.26e-01	3.46e+00	4.28e-02	8.19e-02	8.19e-02
		5.47e-02	5.47e-02	0.00e+00	0.00e+00	*2.54e+01	9.92e-01	5.50e-02	5.66e-02	8.62e-02	3.24e+00	7.34e-03	6.00e-02	6.00e-02
		4.50e-02	4.50e-02	*0.00e+00	*0.00e+00	*2.54e+01	5.90e-01	2.96e-02	1.53e-02	1.48e-02	3.24e+00	2.94e-03	4.58e-02	4.58e-02
		3.11e-03	3.11e-03	6.82e-06	6.82e-06	3.13e-01	4.37e+00	8.66e+00	1.39e+00	4.68e+00	1.92e+01	1.43e+00	5.50e-03	5.50e-03
		6.60e-05	6.60e-05	6.82e-06	6.82e-06	3.13e-01	4.37e+00	7.35e-01	1.39e+00	3.61e+00	1.37e+01	1.21e-03	2.05e-05	2.05e-05
F5	5	4.55e-06	4.55e-06	6.82e-06	6.82e-06	3.13e-01	2.07e-01	8.92e-02	7.29e-01	3.49e+00	7.08e+00	6.37e-05	9.10e-06	9.10e-06
		4.55e-06	4.55e-06	6.82e-06	6.82e-06	3.13e-01	2.32e-04	6.67e-03	4.95e-01	3.42e+00	6.22e+00	6.37e-05	4.55e-06	4.55e-06
		4.55e-06	4.55e-06	6.82e-06	6.82e-06	3.13e-01	0.00e+00	9.69e-04	1.42e-01	3.41e+00	3.11e+00	6.37e-05	4.55e-06	4.55e-06
		3.45e-01	3.45e-01	0.00e+00	0.00e+00	*5.24e+00	2.88e+00	2.59e+00	5.58e-01	7.42e-01	3.19e+00	7.17e-01	5.78e-02	5.78e-02
		1.81e-01	1.81e-01	0.00e+00	0.00e+00	*5.24e+00	2.88e+00	6.90e-01	3.65e-01	4.22e-01	3.19e+00	2.27e-01	1.75e-02	1.75e-02
F5	10	4.55e-02	4.55e-02	0.00e+00	0.00e+00	*5.24e+00	2.88e+00	3.89e-01	2.46e-01	2.00e-01	3.19e+00	5.41e-02	5.72e-03	5.72e-03
		9.63e-03	9.63e-03	0.00e+00	0.00e+00	*5.24e+00	2.76e+00	1.66e-01	1.75e-01	1.33e-01	3.18e+00	2.40e-03	2.10e-03	2.10e-03
		3.20e-03	3.20e-03	0.00e+00	0.00e+00	*5.24e+00	1.79e+00	1.30e-01	1.21e-01	1.25e-01	2.36e+00	5.38e-05	7.77e-04	7.77e-04
		3.65e-01	3.65e-01	0.00e+00	0.00e+00	*3.82e+00	1.61e+00	6.21e-01	7.40e-01	6.52e-01	2.43e+00	2.47e-01	2.61e-01	2.61e-01
		2.63e-01	2.63e-01	0.00e+00	0.00e+00	*3.82e+00	1.61e+00	3.41e-01	2.94e-01	3.40e-01	2.30e+00	1.00e-01	1.54e-01	1.54e-01
F5	15	2.11e-01	2.11e-01	0.00e+00	0.00e+00	*3.82e+00	1.61e+00	1.36e-01	9.12e-02	1.95e-01	2.30e+00	5.61e-02	1.14e-01	1.14e-01
		1.71e-01	1.71e-01	0.00e+00	0.00e+00	*3.82e+00	1.50e+00	6.63e-02	4.28e-02	1.26e-01	2.30e+00	2.25e-02	8.03e-02	8.03e-02
		9.06e-02	9.06e-02	0.00e+00	0.00e+00	*3.82e+00	1.14e+00	4.21e-02	2.26e-02	7.57e-02	2.28e+00	6.45e-03	5.51e-02	5.51e-02
		3.80e-01	3.80e-01	5.51e-06	5.51e-06	*3.30e+00	3.22e+00	5.49e-01	3.18e-01	4.11e-01	2.46e+00	1.21e-01	2.44e-01	2.44e-01
		2.33e-01	2.33e-01	2.17e-07	2.17e-07	*3.30e+00	2.90e+00	3.18e-01	1.35e-01	2.77e-01	2.46e+00	7.56e-02	1.67e-01	1.67e-01
F5	20	1.44e-01	1.44e-01	0.00e+00	0.00e+00	*3.30e+00	2.90e+00	1.11e-01	5.79e-02	1.34e-01	2.46e+00	5.72e-02	9.12e-02	9.12e-02
		1.44e-01	1.44e-01	0.00e+00	0.00e+00	*3.30e+00	2.60e+00	1.52e-02	3.83e-02	4.93e-02	2.35e+00	3.31e-02	6.14e-02	6.14e-02
		1.05e-01	1.05e-01	0.00e+00	0.00e+00	*3.30e+00	2.54e+00	8.59e-03	2.28e-02	1.66e-02	2.35e+00	1.69e-02	3.79e-02	3.79e-02

Tabelle 6.7: Die besten relativen Funktionswerte nach 1 000, 2 500, 5 000, 10 000 beziehungsweise 20 000 Auswertungen

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
		5.10e-01	5.10e-01	1.09e-04	1.09e-04	*2.32e+00	8.19e-01	1.21e+00	5.44e-01	6.38e-01	3.16e+00	8.68e-02	2.95e-01	2.95e-01
		2.06e-01	2.06e-01	4.76e-05	4.76e-05	*2.32e+00	8.19e-01	7.33e-01	1.97e-01	3.12e-01	2.40e+00	5.79e-02	2.08e-01	2.08e-01
F5	25	1.29e-01	1.29e-01	0.00e+00	0.00e+00	*2.32e+00	8.19e-01	3.93e-01	8.06e-02	1.60e-01	2.40e+00	4.57e-02	1.41e-01	1.41e-01
		7.78e-02	7.78e-02	0.00e+00	0.00e+00	*2.32e+00	6.92e-01	1.18e-01	4.30e-02	5.17e-02	2.40e+00	3.23e-02	9.86e-02	9.86e-02
		6.68e-02	6.68e-02	0.00e+00	0.00e+00	*2.32e+00	6.46e-01	8.95e-02	1.47e-02	2.47e-02	2.40e+00	2.29e-02	8.54e-02	8.54e-02
		7.30e-01	7.30e-01	6.08e-01	6.08e-01	*1.70e+00	3.09e+00	9.95e-01	4.66e-01	8.95e-01	2.98e+00	8.06e-02	4.30e-01	4.30e-01
		3.70e-01	3.70e-01	4.88e-02	4.88e-02	*1.70e+00	2.64e+00	6.05e-01	2.13e-01	4.99e-01	2.95e+00	4.94e-02	2.86e-01	2.86e-01
F5	30	2.89e-01	2.89e-01	6.27e-04	6.27e-04	*1.70e+00	2.61e+00	1.81e-01	1.02e-01	3.09e-01	1.97e+00	3.59e-02	2.03e-01	2.03e-01
		1.14e-01	1.14e-01	0.00e+00	0.00e+00	*1.70e+00	2.53e+00	5.24e-02	5.28e-02	1.36e-01	1.97e+00	2.56e-02	1.28e-01	1.28e-01
		1.03e-01	1.03e-01	0.00e+00	0.00e+00	*1.70e+00	2.23e+00	4.03e-02	2.52e-02	4.26e-02	1.97e+00	1.87e-02	1.16e-01	1.16e-01
		8.78e-06	8.78e-06	1.59e-03	*1.59e-03	*4.53e-01	6.36e+00	8.78e+00	1.23e+00	4.04e+00	2.68e+01	1.12e-01	1.53e-04	1.53e-04
		1.60e-09	*7.77e-08	1.59e-03	*1.59e-03	*4.53e-01	1.46e-01	6.39e-01	1.23e+00	1.48e+00	2.53e+01	9.93e-07	2.74e-08	*2.65e-06
F6	2	3.18e-10	*7.77e-08	1.59e-03	*1.59e-03	*4.53e-01	4.95e-04	1.84e-03	1.18e+00	3.71e-01	2.03e+01	1.56e-08	8.21e-09	*2.65e-06
		7.44e-11	*7.77e-08	*1.59e-03	*1.59e-03	*4.53e-01	1.25e-08	1.59e-08	1.49e-02	2.11e-02	1.23e+01	3.23e-10	7.24e-10	*2.65e-06
		3.94e-11	*7.77e-08	*1.59e-03	*1.59e-03	*4.53e-01	0.00e+00	0.00e+00	2.77e-05	4.75e-05	4.26e+00	2.39e-10	1.79e-10	*2.65e-06
		9.99e-01	9.99e-01	0.00e+00	6.76e-01	6.78e-01	1.07e+00	1.18e+00	1.16e+00	1.00e+00	1.07e+00	6.42e-01	6.76e-01	6.76e-01
		9.99e-01	9.99e-01	0.00e+00	6.76e-01	6.78e-01	7.70e-01	5.44e-01	6.48e-01	1.00e+00	1.07e+00	6.42e-01	6.76e-01	6.76e-01
F7-5	4	9.99e-01	9.99e-01	0.00e+00	6.76e-01	6.78e-01	1.01e-01	5.44e-01	2.92e-02	1.00e+00	1.07e+00	6.42e-01	6.76e-01	6.76e-01
		9.99e-01	9.99e-01	0.00e+00	6.76e-01	6.78e-01	4.01e-06	9.26e-03	7.93e-04	9.99e-01	1.07e+00	7.93e-04	6.76e-01	6.76e-01
		9.99e-01	9.99e-01	0.00e+00	6.76e-01	6.78e-01	0.00e+00	0.00e+00	4.01e-06	9.99e-01	1.00e+00	1.34e-06	6.76e-01	6.76e-01
		9.99e-01	9.99e-01	6.89e-01	6.89e-01	6.91e-01	1.10e+00	1.02e+00	1.04e+00	1.01e+00	1.07e+00	1.24e+00	6.89e-01	6.89e-01
		9.99e-01	9.99e-01	0.00e+00	6.89e-01	6.91e-01	6.16e-01	7.46e-01	6.83e-01	9.98e-01	1.07e+00	6.94e-01	6.89e-01	6.89e-01
F7-7	4	9.99e-01	9.99e-01	0.00e+00	6.89e-01	6.91e-01	2.01e-02	7.05e-01	1.56e-01	9.98e-01	1.07e+00	6.89e-01	6.89e-01	6.89e-01
		9.99e-01	9.99e-01	0.00e+00	6.89e-01	6.91e-01	0.00e+00	6.94e-01	2.13e-02	9.98e-01	1.06e+00	6.89e-01	6.89e-01	6.89e-01
		9.99e-01	9.99e-01	0.00e+00	6.89e-01	6.91e-01	0.00e+00	6.94e-01	4.70e-05	9.98e-01	9.88e-01	6.89e-01	6.89e-01	6.89e-01
		0.00e+00	0.00e+00	9.49e-01	9.49e-01	9.52e-01	1.38e+00	1.44e+00	1.41e+00	5.67e-01	1.47e+00	3.20e-01	9.49e-01	9.49e-01
		0.00e+00	0.00e+00	1.14e-02	9.49e-01	9.52e-01	8.04e-01	1.40e+00	1.41e+00	5.13e-02	1.47e+00	0.00e+00	9.49e-01	9.49e-01
F7-10	4	0.00e+00	0.00e+00	0.00e+00	9.49e-01	9.52e-01	7.53e-02	1.37e+00	1.39e+00	1.13e-02	1.47e+00	0.00e+00	9.49e-01	9.49e-01
		0.00e+00	0.00e+00	0.00e+00	9.49e-01	9.52e-01	5.31e-06	1.37e+00	9.90e-01	1.25e-03	1.44e+00	0.00e+00	9.49e-01	9.49e-01
		0.00e+00	0.00e+00	0.00e+00	9.49e-01	9.52e-01	0.00e+00	1.37e+00	9.77e-01	3.35e-04	1.34e+00	0.00e+00	9.49e-01	9.49e-01
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	7.98e+02	1.25e+02	8.84e+02	4.77e+01	1.31e+03	1.28e+02	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	1.13e+00	9.95e+00	5.26e+01	8.61e-01	9.32e+01	2.32e+01	0.00e+00	0.00e+00
F8	3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	0.00e+00	5.94e-01	8.32e-01	3.27e-01	9.32e+01	2.57e-01	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	0.00e+00	1.29e-01	0.00e+00	3.27e-01	9.32e+01	2.57e-01	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.70e+01	0.00e+00	0.00e+00	0.00e+00	6.93e-02	9.32e+01	2.57e-01	0.00e+00	0.00e+00
		2.81e-04	2.81e-04	0.00e+00	0.00e+00	2.98e-01	9.26e+00	1.84e+00	8.91e+00	9.37e-01	1.75e+00	3.12e+00	6.47e-04	2.95e-04
		2.11e-05	2.11e-05	0.00e+00	0.00e+00	2.98e-01	2.80e+00	1.10e+00	2.08e+00	8.46e-01	1.75e+00	6.40e-03	1.41e-05	1.41e-05
F8	6	7.03e-06	7.03e-06	0.00e+00	0.00e+00	2.98e-01	1.52e+00	8.74e-01	1.10e+00	8.40e-01	1.75e+00	1.52e-03	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.98e-01	8.47e-01	8.39e-01	1.30e-01	8.39e-01	1.75e+00	9.85e-05	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.98e-01	8.38e-01	8.38e-01	5.06e-03	8.38e-01	1.75e+00	9.85e-05	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	*0.00e+00	*3.27e-01	1.55e+01	3.24e+00	1.38e+02	1.05e+01	1.57e+02	8.42e+00	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	0.00e+00	*0.00e+00	*3.27e-01	8.29e-04	0.00e+00	2.65e+00	2.08e+00	1.69e+01	1.66e-02	0.00e+00	*0.00e+00
F9	2	0.00e+00	*0.00e+00	0.00e+00	*0.00e+00	*3.27e-01	0.00e+00	0.00e+00	3.33e-02	1.50e-01	1.69e+01	5.18e-05	0.00e+00	*0.00e+00
		0.00e+00	*0.00e+00	*0.00e+00	*0.00e+00	*3.27e-01	0.00e+00	0.00e+00	1.55e-04	7.21e-02	1.20e+00	0.00e+00	0.00e+00	*0.00e+00
		0.00e+00	*0.00e+00	*0.00e+00	*0.00e+00	*3.27e-01	0.00e+00	0.00e+00	0.00e+00	1.50e-03	1.20e+00	0.00e+00	0.00e+00	*0.00e+00
		6.10e-05	6.10e-05	0.00e+00	*0.00e+00	*1.93e-02	2.60e+01	2.21e+00	2.77e+01	5.02e+01	4.66e+01	1.27e-01	6.10e-05	6.10e-05
		6.10e-05	*6.10e-05	0.00e+00	*0.00e+00	*1.93e-02	2.00e+00	1.22e-03	8.07e+00	1.18e+01	2.02e+01	6.10e-05	6.10e-05	*6.10e-05
F10	2	6.10e-05	*6.10e-05	0.00e+00	*0.00e+00	*1.93e-02	1.83e-04	0.00e+00	1.79e+00	1.62e+00	2.77e+00	6.10e-05	6.10e-05	*6.10e-05
		6.10e-05	*6.10e-05	*0.00e+00	*0.00e+00	*1.93e-02	0.00e+00	0.00e+00	6.10e-05	8.81e-01	2.77e+00	6.10e-05	6.10e-05	*6.10e-05
		6.10e-05	*6.10e-05	*0.00e+00	*0.00e+00	*1.93e-02	0.00e+00	0.00e+00	0.00e+00	5.19e-03	2.54e-01	6.10e-05	6.10e-05	*6.10e-05

Tabelle 6.7: Die besten relativen Funktionswerte nach 1 000, 2 500, 5 000, 10 000 beziehungsweise 20 000 Auswertungen

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.93e−02	5.48e+01	4.83e+00	5.90e+00	1.24e+01	4.16e+02	1.49e−02	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.93e−02	5.25e−01	4.95e−03	3.37e+00	5.45e−02	2.75e+01	0.00e+00	0.00e+00	0.00e+00
F11	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.93e−02	0.00e+00	0.00e+00	6.98e−01	5.45e−02	2.73e+01	0.00e+00	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.93e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.35e+01	0.00e+00	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.93e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	4.72e+00	0.00e+00	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.12e−04	1.05e−01	8.59e−02	4.04e−01	2.67e−03	5.71e−02	8.71e−01	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.12e−04	2.70e−02	1.10e−03	4.04e−01	1.77e−06	5.71e−02	8.70e−01	0.00e+00	0.00e+00
F12	2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.12e−04	5.52e−03	0.00e+00	1.49e−01	1.77e−06	1.40e−02	8.70e−01	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.12e−04	6.74e−04	0.00e+00	2.03e−02	1.77e−06	1.40e−02	1.71e−01	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.12e−04	0.00e+00	0.00e+00	2.31e−03	0.00e+00	5.47e−04	0.00e+00	0.00e+00	0.00e+00
		0.00e+00	0.00e+00	2.99e+00	*5.94e+00	*5.94e+00	2.00e+00	9.83e+00	4.70e−04	1.00e+00	6.86e−02	1.17e+01	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	2.99e+00	*5.94e+00	*5.94e+00	1.02e+00	2.00e+00	4.70e−04	1.00e+00	6.86e−02	1.17e+01	0.00e+00	*0.00e+00
F13	2	0.00e+00	*0.00e+00	*2.00e+00	*5.94e+00	*5.94e+00	0.00e+00	9.34e−04	4.70e−04	1.00e+00	6.86e−02	7.96e+00	0.00e+00	*0.00e+00
		0.00e+00	*0.00e+00	*2.00e+00	*5.94e+00	*5.94e+00	0.00e+00	0.00e+00	4.70e−04	1.00e+00	6.86e−02	2.99e+00	0.00e+00	*0.00e+00
		0.00e+00	*0.00e+00	*2.00e+00	*5.94e+00	*5.94e+00	0.00e+00	0.00e+00	4.70e−04	1.00e+00	2.31e−02	2.00e+00	0.00e+00	*0.00e+00
		9.82e−01	9.82e−01	8.80e−01	8.80e−01	8.82e−01	1.17e+00	1.01e+00	1.18e+00	9.75e−01	1.17e+00	1.14e+00	9.04e−01	9.04e−01
		9.82e−01	9.82e−01	8.80e−01	8.80e−01	8.82e−01	1.13e+00	6.97e−01	1.16e+00	8.83e−01	1.17e+00	1.13e+00	9.04e−01	9.04e−01
F14	5	9.82e−01	9.82e−01	8.80e−01	8.80e−01	8.82e−01	1.06e+00	6.45e−02	1.11e+00	8.77e−01	1.05e+00	9.43e−01	9.04e−01	9.04e−01
		9.82e−01	9.82e−01	8.80e−01	8.80e−01	8.82e−01	9.18e−01	7.66e−04	9.84e−01	8.74e−01	1.05e+00	9.43e−01	9.04e−01	9.04e−01
		9.82e−01	9.82e−01	8.80e−01	8.80e−01	8.82e−01	9.17e−01	0.00e+00	9.82e−01	8.74e−01	1.05e+00	7.84e−02	9.04e−01	9.04e−01
		9.12e−01	9.12e−01	9.01e−01	9.01e−01	1.02e+00	1.02e+00	1.02e+00	1.02e+00	1.01e+00	1.02e+00	1.02e+00	9.03e−01	9.03e−01
		9.01e−01	9.01e−01	0.00e+00	0.00e+00	1.02e+00	1.02e+00	1.02e+00	1.00e+00	9.94e−01	1.02e+00	9.53e−01	9.01e−01	9.01e−01
F14	10	9.01e−01	9.01e−01	0.00e+00	0.00e+00	1.02e+00	1.02e+00	1.01e+00	9.94e−01	9.09e−01	1.02e+00	9.01e−01	9.01e−01	9.01e−01
		9.01e−01	9.01e−01	0.00e+00	0.00e+00	1.02e+00	1.01e+00	9.26e−01	9.44e−01	9.03e−01	1.02e+00	9.01e−01	9.01e−01	9.01e−01
		9.01e−01	9.01e−01	0.00e+00	0.00e+00	1.02e+00	9.64e−01	9.06e−01	9.07e−01	9.01e−01	1.02e+00	8.08e−01	9.01e−01	9.01e−01
		1.56e−04	1.56e−04	1.31e+01	*1.31e+01	*2.66e+01	2.89e+01	1.42e+00	8.45e+00	6.02e−01	1.80e+01	2.65e+01	2.25e−04	2.25e−04
		0.00e+00	0.00e+00	1.31e+01	*1.31e+01	*2.66e+01	4.77e+00	9.00e−04	8.51e−01	3.20e−01	5.37e+00	2.42e+01	5.71e−07	5.71e−07
F15	4	0.00e+00	*0.00e+00	1.31e+01	*1.31e+01	*2.66e+01	2.88e+00	3.24e−04	3.64e−02	1.01e−01	3.00e+00	2.41e+01	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	*1.31e+01	*1.31e+01	*2.66e+01	5.12e−02	3.16e−04	1.00e−02	1.03e−02	3.00e+00	1.55e+01	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	*1.31e+01	*1.31e+01	*2.66e+01	3.11e−07	0.00e+00	3.72e−05	8.16e−04	3.00e+00	7.51e+00	0.00e+00	*0.00e+00
		1.61e−01	1.61e−01	9.78e−03	9.78e−03	*0.00e+00	6.43e+00	2.34e+00	4.87e−01	6.32e−01	4.49e+00	6.86e−01	1.56e−01	1.56e−01
		8.16e−02	8.16e−02	9.78e−03	*9.78e−03	*0.00e+00	6.43e+00	1.36e+00	3.46e−01	3.83e−01	3.08e+00	1.57e−01	8.69e−02	8.69e−02
F16	10	4.18e−02	4.18e−02	9.78e−03	*9.78e−03	*0.00e+00	3.13e+00	3.20e−01	1.71e−01	2.04e−01	3.08e+00	1.78e−02	4.51e−02	4.51e−02
		2.69e−02	2.69e−02	*9.78e−03	*9.78e−03	*0.00e+00	8.56e−01	1.37e−01	1.11e−01	1.56e−01	3.08e+00	1.49e−02	1.13e−02	1.13e−02
		2.61e−02	2.61e−02	*9.78e−03	*9.78e−03	*0.00e+00	2.07e−01	2.64e−02	9.46e−02	3.74e−02	3.08e+00	7.05e−03	9.88e−03	9.88e−03
		6.88e−03	6.88e−03	7.25e−03	*7.25e−03	*7.40e−03	8.88e+00	8.57e−02	8.15e+00	2.24e+00	9.04e+00	7.58e−03	1.58e−04	1.58e−04
		6.88e−03	*6.88e−03	7.25e−03	*7.25e−03	*7.40e−03	4.32e−01	7.00e−03	1.45e+00	2.19e−04	8.01e+00	6.92e−03	2.42e−09	*1.57e−04
F17	2	6.88e−03	*6.88e−03	7.25e−03	*7.25e−03	*7.40e−03	1.14e−05	6.91e−03	3.73e−01	5.88e−05	3.91e+00	6.88e−03	8.21e−10	*1.57e−04
		6.87e−03	*6.88e−03	*7.25e−03	*7.25e−03	*7.40e−03	7.03e−09	6.87e−03	6.34e−04	5.88e−05	1.83e+00	6.88e−03	2.07e−11	*1.57e−04
		6.87e−03	*6.88e−03	*7.25e−03	*7.25e−03	*7.40e−03	0.00e+00	6.87e−03	6.09e−05	1.16e−05	1.59e+00	1.62e−04	2.75e−12	*1.57e−04
		1.34e−01	1.34e−01	3.44e−02	*1.72e−01	*4.23e−01	3.46e−01	1.00e−01	5.19e−01	7.73e−03	6.43e−01	1.92e−01	8.58e−02	8.58e−02
		4.91e−02	4.91e−02	3.44e−02	*1.72e−01	*4.23e−01	2.84e−01	1.71e−02	2.41e−01	5.74e−03	3.75e−01	1.79e−01	5.27e−02	5.27e−02
F18	10	2.43e−02	2.43e−02	3.44e−02	*1.72e−01	*4.23e−01	2.84e−01	7.54e−04	2.41e−01	2.85e−03	3.75e−01	1.78e−01	2.43e−02	2.43e−02
		1.38e−02	1.38e−02	*3.44e−02	*1.72e−01	*4.23e−01	2.84e−01	0.00e+00	1.61e−01	2.11e−03	3.75e−01	1.15e−01	8.18e−03	8.18e−03
		4.85e−03	4.85e−03	*3.44e−02	*1.72e−01	*4.23e−01	2.12e−01	0.00e+00	1.61e−01	1.05e−03	1.01e−01	1.05e−02	4.43e−03	4.43e−03
		1.64e−01	1.64e−01	1.52e+00	*1.52e+00	*1.13e+01	5.22e+00	1.86e+00	3.75e−01	4.46e−01	4.27e+00	7.80e−01	2.03e−01	2.03e−01
		6.14e−02	6.14e−02	1.52e+00	*1.52e+00	*1.13e+01	5.10e+00	8.37e−01	1.89e−01	2.41e−01	4.27e+00	5.65e−02	7.30e−02	7.30e−02
F19	20	4.05e−02	4.05e−02	1.52e+00	*1.52e+00	*1.13e+01	3.34e+00	3.64e−01	1.04e−01	9.59e−02	4.09e+00	1.35e−02	4.12e−02	4.12e−02
		1.18e−02	1.18e−02	*1.52e+00	*1.52e+00	*1.13e+01	1.09e+00	1.72e−02	3.45e−02	2.54e−02	3.56e+00	4.97e−03	1.38e−02	1.38e−02
		5.38e−03	5.38e−03	*1.52e+00	*1.52e+00	*1.13e+01	3.97e−01	0.00e+00	7.03e−03	4.55e−04	2.85e+00	3.71e−03	4.60e−03	4.60e−03

Tabelle 6.7: Die besten relativen Funktionswerte nach 1 000, 2 500, 5 000, 10 000 beziehungsweise 20 000 Auswertungen

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
		6.99e-01	6.99e-01	2.22e+00	2.22e+00	*2.42e+00	1.82e+00	1.01e+00	6.49e-01	7.57e-01	1.73e+00	7.97e-01	8.00e-01	8.00e-01
		5.95e-01	5.95e-01	2.18e+00	*2.18e+00	*2.42e+00	1.82e+00	7.99e-01	5.06e-01	4.57e-01	1.64e+00	7.97e-01	6.49e-01	6.49e-01
F20	30	5.91e-01	5.91e-01	2.18e+00	*2.18e+00	*2.42e+00	1.82e+00	5.90e-01	2.80e-01	2.55e-01	1.64e+00	7.97e-01	5.29e-01	5.29e-01
		5.91e-01	5.91e-01	2.18e+00	*2.18e+00	*2.42e+00	1.82e+00	4.19e-01	1.94e-01	1.13e-01	1.64e+00	7.97e-01	4.97e-01	4.97e-01
		4.92e-01	4.92e-01	*1.82e+00	*2.18e+00	*2.42e+00	1.82e+00	2.47e-01	0.00e+00	3.39e-02	1.61e+00	7.97e-01	4.95e-01	4.95e-01
		3.20e-13	3.20e-13	0.00e+00	*0.00e+00	*2.07e-03	7.65e-04	8.78e-05	1.51e-03	3.10e-05	4.43e-06	1.02e-06	0.00e+00	0.00e+00
		0.00e+00	*0.00e+00	0.00e+00	*0.00e+00	*2.07e-03	1.84e-06	8.78e-05	4.13e-05	2.20e-07	4.43e-06	5.39e-07	0.00e+00	*0.00e+00
F21	2	0.00e+00	*0.00e+00	0.00e+00	*0.00e+00	*2.07e-03	0.00e+00	2.79e-05	4.13e-05	2.20e-07	2.23e-07	1.51e-15	0.00e+00	*0.00e+00
		0.00e+00	*0.00e+00	*0.00e+00	*0.00e+00	*2.07e-03	0.00e+00	0.00e+00	1.99e-06	1.16e-08	2.43e-08	7.57e-16	0.00e+00	*0.00e+00
		0.00e+00	*0.00e+00	*0.00e+00	*0.00e+00	*2.07e-03	0.00e+00	0.00e+00	1.99e-06	1.16e-08	1.49e-13	1.64e-16	0.00e+00	*0.00e+00
		1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	9.81e-01	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00
		1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	9.81e-01	1.00e+00	1.00e+00	1.00e+00	0.00e+00	1.00e+00
F22	2	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	9.98e-01	2.72e-03	9.81e-01	9.98e-05	5.05e-01	1.00e+00	0.00e+00	1.00e+00
		1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	0.00e+00	0.00e+00	8.32e-02	5.00e-07	5.05e-01	1.00e+00	0.00e+00	1.00e+00
		1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	0.00e+00	0.00e+00	8.40e-03	2.00e-07	1.97e-01	5.34e-01	0.00e+00	1.00e+00
		5.98e-10	5.98e-10	7.61e-01	*7.61e-01	*8.94e-01	1.79e+00	6.00e-01	1.29e+00	1.63e+00	1.26e+01	7.73e-01	2.77e-09	2.77e-09
		0.00e+00	*9.88e-14	7.61e-01	*7.61e-01	*8.94e-01	7.33e-03	2.34e-02	1.29e+00	1.63e+00	1.26e+01	1.24e-03	6.14e-16	*3.80e-12
F23	2	0.00e+00	*9.88e-14	7.61e-01	*7.61e-01	*8.94e-01	1.78e-08	4.74e-05	2.45e-02	1.63e+00	1.03e+01	1.24e-03	0.00e+00	*3.80e-12
		0.00e+00	*9.88e-14	*7.61e-01	*7.61e-01	*8.94e-01	0.00e+00	4.09e-16	2.45e-05	1.63e+00	3.00e+00	2.07e-05	0.00e+00	*3.80e-12
		0.00e+00	*9.88e-14	*7.61e-01	*7.61e-01	*8.94e-01	0.00e+00	0.00e+00	5.66e-12	1.63e+00	1.15e+00	0.00e+00	0.00e+00	*3.80e-12
		3.20e-12	3.20e-12	7.68e-01	*7.68e-01	*8.04e-01	8.71e-01	3.10e+00	1.69e+01	5.73e+00	2.03e+01	2.10e-01	3.06e-13	3.06e-13
		0.00e+00	*3.91e-16	7.68e-01	*7.68e-01	*8.04e-01	3.57e-02	2.99e-03	4.17e+00	7.69e-01	2.03e+01	1.04e-01	7.81e-16	*3.05e-13
F24	2	0.00e+00	*3.91e-16	7.68e-01	*7.68e-01	*8.04e-01	1.72e-08	2.02e-08	1.02e+00	7.68e-01	1.96e+01	3.95e-04	0.00e+00	*3.05e-13
		0.00e+00	*3.91e-16	*7.68e-01	*7.68e-01	*8.04e-01	0.00e+00	3.15e-12	1.59e-03	7.68e-01	5.62e+00	3.16e-05	0.00e+00	*3.05e-13
		0.00e+00	*3.91e-16	*7.68e-01	*7.68e-01	*8.04e-01	0.00e+00	5.11e-13	4.91e-10	7.68e-01	1.04e+00	1.10e-13	0.00e+00	*3.05e-13
		5.35e-01	5.35e-01	0.00e+00	*0.00e+00	*1.49e+00	1.04e+02	3.44e+01	9.07e+00	6.54e-02	9.17e+01	1.62e+00	9.18e-01	9.18e-01
		1.14e-01	1.14e-01	0.00e+00	*0.00e+00	*1.49e+00	1.12e+01	2.36e+01	2.88e+00	2.17e-02	4.49e+01	4.50e-01	2.54e-01	2.54e-01
F25	4	8.78e-02	8.78e-02	0.00e+00	*0.00e+00	*1.49e+00	4.89e+00	3.94e+00	9.10e-01	2.06e-02	4.49e+01	3.61e-01	1.45e-01	1.45e-01
		5.36e-02	5.36e-02	*0.00e+00	*0.00e+00	*1.49e+00	7.47e-01	2.79e-01	7.50e-01	2.04e-02	5.29e+00	1.25e-01	8.92e-02	8.92e-02
		2.56e-02	2.56e-02	*0.00e+00	*0.00e+00	*1.49e+00	8.79e-08	8.17e-02	6.97e-01	2.02e-02	5.29e+00	5.62e-03	4.20e-02	4.20e-02
		2.93e-01	2.93e-01	9.74e-07	9.74e-07	*2.76e+00	1.05e+01	8.56e+00	8.67e+00	4.09e-01	4.51e+00	1.47e+00	1.77e-01	1.77e-01
		2.93e-01	2.93e-01	9.74e-07	*9.74e-07	*2.76e+00	1.05e+01	7.96e+00	1.79e+00	4.09e-01	2.92e+00	1.29e+00	1.32e-01	1.32e-01
F26	4	8.98e-02	8.98e-02	9.74e-07	*9.74e-07	*2.76e+00	9.02e+00	7.10e-01	5.35e-02	4.09e-01	2.92e+00	7.72e-01	1.31e-01	1.31e-01
		7.68e-02	7.68e-02	*0.00e+00	*9.74e-07	*2.76e+00	1.63e-01	1.50e-01	5.35e-02	4.09e-01	2.18e-01	1.05e-04	1.26e-01	1.26e-01
		7.36e-02	*7.58e-02	*0.00e+00	*9.74e-07	*2.76e+00	2.30e-02	1.22e-01	5.35e-02	1.21e-01	1.57e-01	9.74e-05	4.93e-02	*1.05e-01
		1.56e-03	1.56e-03	5.00e-01	5.14e-01	9.22e-01	7.46e-01	5.94e-01	7.69e-01	4.68e-01	4.72e-01	9.23e-01	5.00e-01	5.00e-01
		1.04e-07	1.04e-07	5.00e-01	5.14e-01	9.22e-01	6.16e-01	5.22e-01	7.69e-01	4.68e-01	4.72e-01	5.68e-01	5.91e-02	5.91e-02
F27	5	0.00e+00	0.00e+00	5.00e-01	5.14e-01	9.22e-01	5.57e-01	4.81e-01	7.69e-01	4.68e-01	4.72e-01	5.68e-01	5.91e-02	5.91e-02
		0.00e+00	0.00e+00	5.00e-01	5.14e-01	9.22e-01	1.02e-01	4.80e-01	4.72e-01	4.68e-01	4.72e-01	4.84e-01	5.91e-02	5.91e-02
		0.00e+00	0.00e+00	5.00e-01	5.14e-01	9.22e-01	5.37e-05	4.80e-01	4.68e-01	4.68e-01	4.72e-01	4.84e-01	5.91e-02	5.91e-02
		4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	1.00e+00	9.27e-01	1.00e+00	9.36e-01	9.90e-01	1.00e+00	6.24e-06	6.24e-06
		4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	1.00e+00	8.52e-01	1.00e+00	8.50e-01	9.90e-01	1.00e+00	2.73e-06	2.73e-06
F27	10	4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	1.00e+00	7.08e-02	1.00e+00	7.17e-01	9.90e-01	1.00e+00	1.95e-07	1.95e-07
		4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	9.99e-01	0.00e+00	9.97e-01	7.17e-01	9.90e-01	1.00e+00	1.95e-07	1.95e-07
		4.64e-01	4.64e-01	9.95e-01	9.95e-01	9.95e-01	9.22e-01	0.00e+00	9.78e-01	7.17e-01	9.90e-01	1.00e+00	1.95e-07	1.95e-07
		4.77e-01	4.77e-01	9.60e-01	9.60e-01	8.77e-01	1.28e+00	6.84e-01	9.93e-01	5.38e-02	1.06e+00	9.88e-01	4.14e-01	4.14e-01
		4.77e-01	4.77e-01	9.60e-01	9.60e-01	8.77e-01	1.25e+00	2.08e-01	9.93e-01	6.21e-03	7.23e-01	9.88e-01	4.14e-01	4.14e-01
F28	5	4.77e-01	4.77e-01	9.60e-01	9.60e-01	8.77e-01	1.03e+00	2.24e-02	6.48e-01	2.83e-03	7.23e-01	9.88e-01	4.14e-01	4.14e-01
		4.77e-01	4.77e-01	9.60e-01	9.60e-01	8.77e-01	5.35e-01	6.11e-04	5.50e-01	1.04e-04	7.23e-01	6.50e-01	4.14e-01	4.14e-01
		4.77e-01	4.77e-01	9.60e-01	9.60e-01	8.77e-01	2.67e-01	0.00e+00	4.22e-01	4.24e-05	7.23e-01	7.81e-02	4.14e-01	4.14e-01

Tabelle 6.7: Die besten relativen Funktionswerte nach 1 000, 2 500, 5 000, 10 000 beziehungsweise 20 000 Auswertungen

Prob.	dim	ASA	ASAorig	CS	CSorig	CSrand	DE	Geno3	GenoIII	PGA	Random	SIGMA	VFSR	VFSRorig
		8.20e−01	8.20e−01	7.99e−01	7.99e−01	1.47e+00	1.60e+00	9.13e−01	8.84e−01	8.55e−01	1.36e+00	1.31e+00	2.19e−01	2.19e−01
		8.05e−01	8.05e−01	6.89e−01	7.99e−01	1.47e+00	1.47e+00	6.99e−01	8.84e−01	7.11e−01	1.36e+00	1.07e+00	1.85e−02	1.85e−02
F28	10	3.58e−01	3.58e−01	4.26e−01	7.99e−01	1.47e+00	1.47e+00	5.86e−01	8.84e−01	6.77e−01	1.34e+00	8.39e−01	3.92e−03	3.92e−03
		3.55e−01	3.55e−01	4.26e−01	7.99e−01	1.47e+00	1.32e+00	4.04e−01	6.14e−01	6.64e−01	1.30e+00	8.15e−01	1.76e−04	1.76e−04
		3.55e−01	3.55e−01	4.26e−01	7.99e−01	1.47e+00	1.27e+00	3.83e−01	6.14e−01	6.61e−01	9.65e−01	8.15e−01	0.00e+00	0.00e+00
		6.27e−01	6.27e−01	5.09e−01	7.83e−01	8.22e−01	8.34e−01	8.14e−01	5.79e−01	2.24e−01	6.93e−01	6.87e−01	9.87e−01	9.87e−01
		6.26e−01	6.26e−01	5.09e−01	7.83e−01	8.22e−01	7.80e−01	4.17e−01	3.75e−01	1.47e−01	6.61e−01	6.87e−01	9.87e−01	9.87e−01
F29	5	6.26e−01	6.26e−01	5.09e−01	7.83e−01	8.22e−01	4.53e−01	2.55e−01	2.09e−01	9.24e−02	6.61e−01	6.87e−01	9.87e−01	9.87e−01
		6.26e−01	6.26e−01	5.09e−01	7.83e−01	8.22e−01	2.50e−01	2.21e−01	4.63e−02	6.40e−02	4.11e−01	6.75e−01	9.87e−01	9.87e−01
		6.26e−01	6.26e−01	5.09e−01	7.83e−01	8.22e−01	6.59e−02	2.19e−01	0.00e+00	5.79e−02	4.11e−01	6.31e−01	9.87e−01	9.87e−01
		9.87e−01	9.87e−01	9.56e−01	9.56e−01	9.87e−01	9.95e−01	8.81e−01	8.78e−01	9.18e−01	9.67e−01	8.54e−01	9.73e−01	9.73e−01
		9.86e−01	9.86e−01	9.56e−01	9.56e−01	9.87e−01	9.76e−01	6.18e−01	7.23e−01	8.78e−01	9.62e−01	8.54e−01	9.71e−01	9.71e−01
F29	10	9.86e−01	9.86e−01	9.56e−01	9.56e−01	9.87e−01	9.34e−01	5.60e−01	5.19e−01	8.55e−01	9.59e−01	8.54e−01	9.71e−01	9.71e−01
		9.86e−01	9.86e−01	9.56e−01	9.56e−01	9.87e−01	9.34e−01	4.83e−01	2.55e−01	8.42e−01	9.15e−01	8.54e−01	9.71e−01	9.71e−01
		9.86e−01	9.86e−01	9.56e−01	9.56e−01	9.87e−01	8.26e−01	4.75e−01	0.00e+00	8.41e−01	8.89e−01	8.46e−01	9.71e−01	9.71e−01
		4.67e−02	4.67e−02	9.95e−09	9.95e−09	*2.64e−02	1.92e+00	7.05e−01	3.03e−01	2.63e−01	6.03e−01	5.77e−01	7.85e−02	7.85e−02
		2.62e−02	2.62e−02	0.00e+00	*0.00e+00	*2.64e−02	1.92e+00	1.72e−01	1.06e−01	9.17e−02	6.03e−01	2.22e−01	4.08e−02	4.08e−02
F30	9	2.13e−02	2.13e−02	0.00e+00	*0.00e+00	*2.64e−02	1.92e+00	9.48e−02	3.94e−02	3.71e−02	4.59e−01	1.11e−01	2.92e−02	2.92e−02
		1.31e−02	1.31e−02	*0.00e+00	*0.00e+00	*2.64e−02	1.48e+00	3.40e−02	7.51e−03	1.82e−02	1.79e−01	6.04e−02	9.51e−03	9.51e−03
		8.58e−03	8.58e−03	*0.00e+00	*0.00e+00	*2.64e−02	1.25e+00	2.19e−02	1.81e−03	1.15e−02	1.79e−01	3.95e−02	5.61e−03	5.61e−03
		1.21e−01	1.21e−01	2.72e+01	2.72e+01	*7.83e+01	1.76e+00	3.78e+01	5.56e+01	4.32e+00	2.88e+01	1.41e+01	4.37e−02	4.37e−02
		1.28e−03	1.28e−03	2.69e+01	2.69e+01	*7.83e+01	1.21e−02	1.17e+01	4.29e+01	2.77e+00	2.88e+01	2.26e+00	4.43e−04	4.43e−04
F30	17	1.54e−04	1.54e−04	2.69e+01	*2.69e+01	*7.83e+01	1.21e−02	1.80e+00	2.17e+01	1.34e+00	2.88e+01	1.40e−03	8.68e−05	8.68e−05
		2.87e−05	2.87e−05	*2.69e+01	*2.69e+01	*7.83e+01	1.17e−02	1.35e−02	1.14e+01	3.18e−01	2.88e+01	6.14e−05	4.36e−05	4.36e−05
		0.00e+00	0.00e+00	*2.69e+01	*2.69e+01	*7.83e+01	5.85e−03	2.95e−04	4.33e+00	1.92e−02	2.73e+01	6.14e−05	7.61e−06	7.61e−06

Anhang A

Durchführung der Tests

A.1 Management der Tests

In diesem Abschnitt wird beschrieben, wie die oben behandelten Tests durchgeführt wurden und wie sie in Zukunft für ein beliebiges Optimierungsprogramm durchgeführt werden können.

Im folgenden ist darauf zu achten, daß `ejeval.c`, `ej_eval`, `ejeval`, `ejevai` und `ejeval_f.` unterschiedliche Bedeutungen haben. Die beiden Buchstaben *ej* dienten dazu, um von mir geschriebenen Routinen von fremden unterscheiden zu können und somit ein späteres Auffinden zu erleichtern.

Um ein Optimierungsprogramm zu testen, kopiert man den Quellcode in ein beliebiges Verzeichnis. Dorthin legt man auch die Dateien `ejeval.c`, `ejeval_f.c`, `test-all` und die Verzeichnisse `include` und `results`. Dabei ist `ejeval.c` jene Datei, die die Routine `ej_eval` und die Routinen zur Berechnung der Funktionswerte beinhaltet. Die Datei `ejeval_f.c` wird zusätzlich zum Test von Fortran-Programmen benötigt, da sie die beiden Routinen `ej_eval` und `ej_evai` enthält (siehe unten). Das Unterverzeichnis `include` enthält für jede Testfunktion ein Header-File, in dem der Name und die Dimension des Testproblems, sowie die Vektoren für die untere und obere Grenze des Definitionsbereiches enthalten ist.

In das Untervererzeichnis `results` werden alle Ergebnisfiles gespeichert.

Um das Programm mit allen Tests laufen zu lassen, ruft man einfach das Shell-Script `test-all` auf. Dieses kopiert der Reihe nach jede in `include` enthaltene Datei nach `ej_include.h`, compiliert das Programm, führt es aus und kopiert das entstandene Datenfile in das Verzeichnis `results`. Jedenfalls muß man in `test-all` den auszuführenden Programmnamen (`PROG`) und den Aufruf zum Compilieren (`MAKE`) den Umständen anpassen.

Beim Aufruf des Compilers ist eine der beiden folgenden Optionen hinzuzufügen, je nachdem ob das Programm für die Variablendarstellung `real` oder `double` verwendet:

`-DCONFIG_real=DOUBLE` oder `-DCONFIG_real=REAL`.

Natürlich muß auch die Datei `ejeval.c` und für Fortran-Programme (siehe unten) auch `ejeval_f.c` eingebunden werden.

Um die benötigten Dateien aus den 20 000 Zeilen langen Dateien zu extrahieren, weiterzuverarbeiten und die formatieren LaTeX- beziehungsweise `HTML`-Tabellen zu erzeugen, wurden mehrere `Perl`- und Shell-Scripts eingesetzt.

A.1.1 `C`-Programmepakete

Die Handhabung bei `C`-Programmen ist dann sehr einfach. Im Quellcode wird einfach an jeder Stelle, an der ein Funktionswert berechnet werden soll oder an der die Dimension des Testproblems oder die Grenzen des Definitionsbereichs benötigt werden, die Funktion `ej_eval` entsprechend Abschnitt A.2, aufgerufen. Wenn das Programm irgendwelche Abbruchsbedingungen beinhaltet, dann sollten diese ausgeschaltet sein, beziehungsweise erst nach der geforderten Anzahl von Funktionsauswertungen aktiv werden.

A.1.2 Fortran-Programmpakete

Da es bei Fortran-Programmen nicht wie in `C` möglich ist, auch `int`-Variablen als `double` zurückzugeben und danach umzuwandeln, gibt es für den Aufruf von `ej_eval` zusätzlich zwei Funktionen, welche in der Datei `ejeval_f.c` enthalten sind. Hier wird also, entweder `ejevai`, wenn `int`-Werte benötigt werden, oder sonst `ejeval` aufgerufen. Die Handhabung der Übergabeparameter ist in Abschnitt A.2 beschrieben. Ansonsten ist die Vorgangsweise gleich wie bei `C`-Programmen.

A.2 Die Funktion ej_eval

Diese in `C` geschriebene Routine ist der zentrale Baustein, um die getesteten Softwarepakete mit den erforderlichen Daten zu versorgen und die Dateien mit den jeweils 20 000 Funktionsauswertungen zu erstellen. Sie ist folgendermaßen definiert:

```
real ej_eval(real *tmp, real *retvec, int ret)
```

Der Variablentyp `real` ist dabei entweder `double` oder `float` und wird beim Compilieren festgelegt, wie in Abschnitt A.1 beschrieben. Das ist deshalb notwendig, weil verschiedene Programmpakete unterschiedliche Darstellungsformen verwenden und der Aufwand für zwei Versionen von `ej_eval` unnötig groß gewesen wäre. Je nach dem Wert, der bei dem Funktionsaufruf `ret` zugewiesen wird, bestimmt die Routine den entsprechenden Rückgabewert. Der Pointer `*tmp` und die Variable `ret` enthalten dabei (bei Bedarf) benötigte Information, während der Pointer `*retvec` und die Variable `returnwert` zur Rückgabe der angeforderten Daten dienen. Folgende Tabelle stellt, in Abhängigkeit von dem `ret` zugewiesenen Wert, die Belegung der einzelnen Variablen dar, wobei ein leerer Eintrag bedeutet, daß `ej_eval` diese Variable nicht benötigt.

`ret`	`*tmp`	`*retvec`	`returnwert`
-1	x		$f(x)$
0			n
1		U	
2		O	
5			FMAX
6			U_{min}
7			O_{max}

Tabelle A.1: Aufruf- und Rückgabeparameter der Funktion `ej_eval`

Der Punkt x ist dabei jener, dessen Funktionswert $f(x)$ berechnet werden soll. Der Wert n ist die Dimension des aktuellen Testproblems und FMAX steht für die maximale Anzahl von erlaubten Funktionswerten. U und O sind die vektorwertige Unter- beziehungsweise Obergrenze des Definitionsbereichs und $U_{\text{min}} = \min\limits_{1 \le i \le n} \{U_i\}$ und $O_{\text{max}} = \max\limits_{1 \le i \le n} \{ O_i \}$ sind deren kleinster beziehungsweise größter Wert.

Bei jedem Aufruf von `ej_eval` mit dem Wert -1 für `ret`, wird die Routine des aktuellen Testproblems zur Berechnung des Funktionswertes aufgerufen, der Funktionswert in das Datenfile geschrieben, ein Zähler um eins erhöht und der Funktionswert an das aufrufende Programm übergeben. Wenn der Zähler den Wert 20 000 erreicht hat, wird das Programm terminiert, da das Datenfile die gewünschte Größe erreicht hat.

A.2.1 Header-Dateien

Um die Funktion `ej_eval` an jedes Testproblem anzupassen, dient ein Header-File `ej_include.h`, welches von `test-all` aus dem Verzeichnis `includes` in das Arbeitsverzeichnis kopiert wird.

In dieser Datei ist der Name des aktuellen Testproblems, dessen Dimension und die Grenzen des Definitionsbereichs gespeichert. Der Name muß dabei gleich dem Namen einer in `ejeval.c` enthaltenen `C`-Funktion sein, die die Funktionswerte des entsprechenden Problems berechnet.

Literaturverzeichnis

[1] First International Contest on Evolutionary Optimization: `http://iridia.ulb.ac.be/langerman/ICEO.html`

[2] Second International Contest on Evolutionary Optimization: `http://homepages.ulb.ac.be/~gseront/ICEO/2ndICEO.html`

[3] Association for Computing Machinery—Collected Algorithms `http://www.acm.org/calgo/contents/`

[4] Aluffi-Pentini F., Parisi V., Zirilli F.: A Global Optimization Algorithm Using Stochastic Differential Equations. ACM Trans. Math. Softw., 14, 345–365, December 1988.

[5] Aluffi-Pentini F., Parisi V., Zirilli F.: Algorithm 667: SIGMA – A Stochastic-Integration Global Minimization Algorithm. ACM Trans. Math. Softw., 14, 366–380, December 1988.

[6] Bäck T.: A User's Guide to GENEsYs 1.0. University of Dortmund, 1992.

[7] Bertsekas D.: Constrained Optimization and Lagrange Multiplier Methods. New York: Academic Press, 1982.

[8] Boender C., Rinnooy K., Timmer G., Stougie L.: A Stochastic Method for Global Optimization. Mathematical Programming 22, 125–140. Amsterdam: North-Holland, 1982.

[9] Boender C., Romeijn H.: Stochastic Methods. In: Horst R., Pardalos P.: Handbook of Global Optimization. Dordrecht: Kluwer, 829ff, 1995.

[10] Corana A., Marchesi M., Martini C., Ridella S.: Minimizing Mulitmodal Functions of Continuous Variables with the "Simulated Annealing Algorithm". ACM Trans. Math. Softw., 2, 272–280, March 1987.

[11] Csendes T.:
`http://www.inf.u-szeged.hu/~csendes/`

[12] Davis L.: Handbook of Genetic Algorithms. London, New York: International Thomson Computer Press, 1996.

[13] Desai R., Patil R.: SALO: Combining Simulated Annealing and Local Optimization for Efficient Global Optimization. 9^{th} Florida AI Research Symposium (FLAIRS), Juni 1996.

[14] Dixon L., Szegö G.: The Global Optimization Problem: an Introduction. In: Dixon L., Szegö G.: Towards Global Optimization 2, 1–15. Amsterdam: North-Holland, 1978.

[15] De Jong K.: An analysis of the behaviour of a class of genetic adaptive systems. PhD Thesis, University of Michigan, 1975.

[16] Fiacco A., McCormick G.: Nonlinear Programming. Philadelphia: Siam, 1990.

[17] Grienwank A.: Generalized Descent for global Optimization. JOTA 34, 11–39, 1981.

[18] Heuser H.: Lehrbuch der Analysis, Teil 2. Stuttgart: B. G. Teubner, 1986.

[19] Ingber L., Rosen B.: Genetic Algorithms and Very Fast Simulated Reannealing: A comparison. J. of Mathematical and Computer Modelling 16, 87–100, 1992.

[20] Ingber L.: Simulated Annealing: Practice versus theory. J. of Mathematical and Computer Modelling 18, 29–57, 1993.

[21] Ingber L.:
`http://www.ingber.com`

[22] Janka E.:
`http://solon.mat.univie.ac.at/~vpk/math/`

[23] Jones D., Perttunen C., Stuckman B.: Lipschitzian Optimization without the Lipschitz Constant. J. of Optimization Theory and Applications 79, 269–284, 1993.

[24] Knuth D.: The Art of computer Programming. Massachusetts: Addison-Wesley, 1969.

[25] Levine D.: PGAPack: Parallel Genetic Algorithm Library
`http://www-unix.mcs.anl.gov/~levine/`

[26] Locatelli M., Schoen F.: Simple Linkage: Analysis of a Threshold-accepting Global Optimization Method. J. of Global Optimization 9, 95–111, 1996.

[27] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E.: Equation of state calculations by fast computing machines. J. of Chemical Physics 21, 1087–1092, 1953.

[28] Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed. Berlin: Springer, 1996.

[29] Michalewicz Z.:
`http://www.coe.uncc.edu/~zbyszek/`

[30] Mockus J.: Bayesian Approach to Global Optimization. Dordrecht: Kluwer, 1989.

[31] Moré J., Garbow B., Hillstrom K.: Testing Unconstrained Optimization Software. ACM Trans. Math. Softw., 7, March 1981.

[32] Neumaier A., Huyer W.: Global Optimization by Multilevel Coordinate Search. Dordrecht: Kluwer, 1998.

[33] Neumaier A.: Global (and Local) Optimization
http://solon.cma.univie.ac.at/~neum/glopt.html

[34] Pintér J.: Global Optimization in Action. Berlin, Heidelberg, New York: Springer, 1992.

[35] Pohlheim H.: GEA Toolbox—Examples of Objective Functions.
http://
www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/fcnindex.html

[36] Poole I.: Genetic Algorithm for General Application (GAGA)
ftp://cs.ucl.ac.uk/darpa/gaga.shar

[37] Storn R., Price K.: Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continous Spaces. J. of Global Optimization 11: 1997. Dordrecht: Kluwer 1997.

[38] Storn R.: Differential Evolution (DE) for Continuous Function Optimization
http://http.icsi.berkeley.edu/~storn/code.html

[39] Törn A., Žilinskas A.: Global Optimization. Lecture Notes in Computer Science, 350. Berlin: Springer 1989.

[40] Yuret D.: From Genetic Algorithms to Efficient Optimization. Master thesis, M.I.T., 1994.

[41] Zhigljavsky A.: Theory of Global Random Search. Dordrecht: Kluwer, 1991.

[42] Zimmermann W.: Operations Research. Oldenbourg, 1990.