Ein genetischer Algorithmus mit Mehrfachpopulationen zur Numerischen Optimierung

Hartmut Pohlheim, Ilmenau

Dipl.-Ing. Hartmut Pohlheim ist Promotionsstudent im Fachgebiet Dynamik und Simulation ökologischer Systeme des Institutes für Automatisierungs- und Systemtechnik der Technischen Universität Ilmenau. Hauptarbeitsfeld: Entwurf und systemtechnische Anwendung genetischer Algorithmen

Adresse: Technische Universität Ilmenau, Fakultät für Informatik und Automatisierung, Institut für Automatisierungs- und Systemtechnik, Fachgebiet Dynamik und Simulation ökologischer Systeme, PSF 327, D-98684 Ilmenau,
E-Mail: pohlheim@systemtechnik.tu-ilmenau.de

Im Beitrag wird ein genetischer Algorithmus mit Mehrfachpopulationen (MPGA) vorgestellt. Die Operatoren des Algorithmus werden im einzelnen beschrieben. Insbesondere wird auf die Umsetzung des Prinzips der Mehrfachpopulationen eingegangen.

Eine neue Methode der Fitneßzuweisung durch Ranking unter Verwendung einer nichtlinearen Verteilung wird vorgestellt. Nichtlineares Ranking ermöglicht einen höheren selektiven Druck als konventionelle Ranking-Methoden.

Die Verwendung des genetischen Algorithmus zur Optimierung der Steuerung dynamischer Systeme wird an einem Beispiel gezeigt.

Der Algorithmus wurde in MATLAB implementiert und ist als Teil der Genetic Algorithm Toolbox verfügbar.

A Multiple Population Genetic Algorithm for Numerical Optimization

A genetic algorithm with multiple population (MPGA) is described. The operators of the algorithms are discussed in detail. Especially the multiple population concept is explained.

A new method for fitness assignment by ranking using a non-linear distribution is introduced. The use of non-linear ranking permits higher selective pressures than the conventional ranking methods.

The use of the genetic algorithm in dynamic and control system optimization is presented together with an example.

The algorithm was implemented in MATLAB and is available as part of the Genetic Algorithm Toolbox.

1. Einleitung

Das Interesse am Einsatz von genetischen Algorithmen hat sich in den letzten Jahren deutlich verstärkt. Verglichen mit traditionellen Such- und Optimierungsverfahren sind die genetischen Algorithmen robust, global und mit einem geringeren Aufwand einsetzbar, insbesondere wenn nur ein geringes Wissen über das System vorliegt. Da die genetischen Algorithmen keine Informationen über den Gradienten benötigen und durch das stochastische Verhalten des Suchverfahrens sind genetische Algorithmen fähig, den gesamten Lösungsraum mit einer höheren Wahrscheinlichkeit zum Finden des Optimums zu durchsuchen.

Diese Arbeit stellt den genetischen Algorithmus mit Mehrfachpopulationen (MPGA - Multiple Population Genetic Algorithm) vor. In Abschnitt 2 wird der Algorithmus in seinen einzelnen Komponenten beschrieben. Die relevanten Punkte werden diskutiert und detailliert erläutert. Abschnitt 3 zeigt die Verwendung des Algorithmus zur Optimierung dynamischer Systeme. Im Abschnitt 4 wird ein Ausblick auf die Weiterentwicklung des Algorithmus und damit der Toolbox gegeben. Der letzte Abschnitt führt die verwendete Literatur auf, die für weitere Studien in das Gebiet nützlich ist.

Der in MATLAB implementierte MPGA ist Teil der *Genetic Algorithm Toolbox for MATLAB* ([2]), die an der University of Sheffield entwickelt wurde. Mehrere high-level Funktionen bieten ein einheitliches Interface für eine größere Anzahl von low-level Funktionen, wodurch eine einfache Benutzung erreicht und die Überschaubarkeit des resultierenden MATLAB-Quelltextes erhöht wird. Diese Toolbox ist nicht nur für die Forschung und

Pohlheim, H.: Ein genetischer Algorithmus mit Mehrfachpopulationen zur Numerischen Optimierung.
at-Automatisierungstechnik 3 (1995), pp. 127-135, 1995.

Anwendung nützlich, sondern zusammen mit der Einführung und dem Referenzhandbuch kann sie zum Lehren von genetischen und evolutionären Algorithmen in den verschiedensten Bereichen verwendet werden.

2. Beschreibung des genetischen Algorithmus

Es existieren drei Schulen von evolutionären Algorithmen: genetische Algorithmen, Evolutionsstrategien und genetische Programmierung. Genetische Algorithmen wurden ursprünglich in den USA von J. H. Holland und D. E. Goldberg ([6] und [4]) entwickelt. Evolutionsstrategien, in Deutschland hauptsächlich von I. Rechenberg und H.-P. Schwefel ([12] und [14]) entwickelt, benutzen eine andere Herangehensweise. Beide sind jedoch von denselben Prinzipien der natürliche Evolution beeinflußt. Eine Diskussion der Gemeinsamkeiten und Unterschiede zwischen diesen beiden Schulen liegt nicht im Rahmen dieses Beitrages. Ein guter Vergleich kann in [5] gefunden werden.

Evolutionäre Algorithmen unterscheiden sich von gradientenbasierten Suchverfahren vor allem in der Art der Veränderung der Parameter/Variablen.. Wo Gradientenverfahren die Variablen nach deterministischen Regeln verändern, basieren evolutionäre Algorithmen auf zufällig verteilten Regeln. Evolutionäre Algorithmen modellieren natürliche Prozesse, z.B. Rekombination, Mutation, Selektion, Migration und Diffusion. Sie arbeiten an Populationen von Individuen anstelle von einzelnen Lösungen. Dadurch wird die Suche in einer parallelen Art und Weise durchgeführt.

Der genetische Algorithmus mit Mehrfachpopulationen verwendet Ideen der Evolutionsstrategien und der genetischen Algorithmen.

Im MPGA werden direkt die reellen Werte der Variablen verwendet im Gegensatz zur Umwandlung der reellen Werte in binäre und Verwendung dieser binären Werte, wie es üblicherweise bei den genetischen Algorithmen erfolgt. Bei vielen numerischen Steuerungs- und Optimierungsproblemen liegen die Variablen/Parameter als reelle Werte vor. Damit besteht kein Grund, diese Werte in eine andere Repräsentation zu überführen. Die Genauigkeit der reellen Repräsentation ist generell besser als die der binären. Zusätzlich zeigen viele Arbeiten Vorteile der reellen Repräsentation gegenüber der binären für die numerische Optimierung ([8], [9], [11], [19]).

In der reellen Repräsentation ist jedes Individuum als ein Vektor von reellen Zahlen mit derselben Länge wie die Zielfunktion kodiert. Damit besteht bei einer Zielfunktion mit drei Variablen jedes Individuum aus einem Vektor von drei reellen Zahlen. Speziell in MATLAB spart die Verwendung der reellen Zahlen eine große Menge Speicherplatz, verglichen mit den deutlich längeren Individuen in binärer Repräsentation. Daraus erwächst bei gleichem Problem eine deutlich kürzere Rechenzeit der reellen Repräsentation.

Hier ist eine Gliederung des genetischen Algorithmus für eine Population:

```
initialisiere Population
berechne Zielfunktion/Fitneß für Individuen
while not Abbruchkriterium erfüllt
        selektiere Individuen für Fortpflanzung (Eltern)
        rekombiniere Eltern und produziere Nachkommen
        mutiere Nachkommen
        berechne Zielfunktion/Fitneß für Nachkommen
        füge Nachkommen in Population ein ⇨ neue Population
end
```

Zu Beginn der Berechnungen wird eine Anzahl von Individuen (die Population) zufällig initialisiert (jeder Variablen wird ein zufälliger Wert aus dem möglichen Intervall zugewiesen). Dann wird die Zielfunktion für diese Individuen berechnet. Damit ist die erste Generation produziert.

Individuen werden entsprechend ihrer Fitneß für die Produktion von Nachkommen ausgewählt. Immer zwei Eltern rekombinieren zur Produktion von zwei Nachkommen. Alle Nachkommen werden mit einer vorgegebenen Wahrscheinlichkeit mutiert und nachfolgend wird deren Fitneß berechnet. Die Nachkommen werden in die Population eingefügt und ersetzen die Eltern, womit eine neue Generation produziert ist. Diese Schleife wird solange durchlaufen, bis ein Abbruchkriterium erreicht ist. Das beste Individuum der letzten Generation ist die Lösung, das gefundene Optimum.

Der genetische Algorithmus für eine Population ist leistungsfähig und erreicht für eine große Klasse von Problemen gute Ergebnisse.

In der natürlichen Evolution tendieren Arten dazu, sich in Unterpopulationen der Gesamtpopulation fortzupflanzen, die bis zu einem gewissen Grad voneinander isoliert sind. Fortpflanzung zwischen Individuen verschiedener Unterpopulationen ist möglich, wahrscheinlicher ist aber, daß sich die Individuen innerhalb einer Unterpopulation fortpflanzen. Das Migrations- oder Inselmodell unterteilt eine Population in viele kleine Unterpopulationen. Jede Unterpopulation entwickelt sich als abgeschlossene Einheit mit lokaler Selektion und Reproduktion, genau wie der genetische Algorithmus für eine Population. Von Zeit zu Zeit jedoch werden ein oder mehrere Individuen zwischen den Unterpopulationen ausgetauscht, so daß Individuen der einen Unterpopulation in eine andere eingeführt werden. Die Grad der Migration wird durch die Anzahl der auszutauschenden Individuen, die Zeit der Isolation der Unterpopulationen und die Austauschpfade bestimmt und entscheidet, wie stark die genetischen Unterschiede in der Gesamtpopulation sind. Der genetische Algorithmus mit Mehrfachpopulationen (MPGA) modelliert diese Unterteilung der Gesamtpopulation in Unterpopulationen und bildet damit die Evolution einer Art in einer Weise nach, die näher an der Natur ist als dies beim genetischen Algorithmus für eine Population der Fall ist.

Hier ist eine Gliederung des genetischen Algorithmus für Mehrfachpopulation:

```
initialisiere alle Unterpopulationen
berechne Zielfunktion/Fitneß für Individuen
while not Abbruchkriterium für alle Unterpopulationen erfüllt
        for einige Generationen
                for alle Unterpopulationen
                        genetischer Algorithmus für eine Population
                end
        end
        migriere Individuen zwischen den Unterpopulationen
end
```

Die folgenden Unterabschnitte beschreiben Operatoren bzw. Routinen des MPGA im Detail. Für die Benutzung der Routinen und Parameter der Funktionen sowie zur Implementierung wird auf das Referenzhandbuch der *Genetic Algorithm Toolbox* ([2]) verwiesen. Dort ist jede Funktion detailliert beschrieben, die Benutzung wird gezeigt und die verschieden Möglichkeiten, die geboten werden, sind ausführlich dargestellt.

2.1. Selektion

Der genetische Algorithmus beginnt (nach der Initialisierung) mit der Selektion der benötigten Anzahl von Eltern für die Fortpflanzung (Rekombination). Der erste Schritt ist:

- Fitneßzuweisung proportional zum Zielfunktionswert (proportional fitness assignment) oder
- Fitneßzuweisung nach der Reihenfolge der Zielfunktionswerte (rank-based fitness assignment)

Verschiedene Vergleiche (z.B. [18]) zeigten das robustere Verhalten der zweiten Methode, des Ranking. Diese Methode wird im MPGA verwendet.

Die direkte Selektion wird im nächsten Schritt durchgeführt. Eltern werden in Abhängigkeit von ihrer Fitneß durch einen der folgenden Algorithmen ausgewählt:

- roulette-wheel selection
- stochastic universal sampling

Siehe auch [1] für einen Vergleich verschiedener Selektionsverfahren.

2.1.1. Fitneßzuweisung durch Ranking

Beim Ranking werden die Individuen der Population entsprechend ihrer Zielfunktionswerte sortiert. Der jedem Individuum zugewiesene Fitneßwert hängt nur von der Position des Individuums innerhalb dieser sortierten Population ab und nicht von seinem Zielfunktionswert direkt (wie bei der poportionalen Fitneßzuweisung).

Ranking bietet eine gleichmäßige Skalierung über die Population und eine einfache Methode, den selektiven Druck zu kontrollieren. (Der selektive Druck gibt die Wahrscheinlichkeit des besten Individuums an, ausgewählt zu werden, verglichen mit der durchschnittlichen Wahrscheinlichkeit der Selektion aller Individuen).

Wenn *Nind* die Anzahl von Individuen der Population ist, *Pos* die Position eines Individuums in dieser Population (das schlechteste Individuum hat *Pos*=1, das beste Individuum *Pos*=*Nind*) und *SP* der selektive Druck, dann berechnet sich der Fitneßwert für das Individuum wie folgt:

Lineares Ranking:

$$Fitneß(Pos) = 2 - SP + 2 \cdot (SP - 1) \cdot \frac{(Pos - 1)}{(Nind - 1)}.$$

Lineares Ranking erlaubt Werte des selektiven Druckes bis 2.0. Für größere Werte muß nichtlineares Ranking verwendet werden.

Mit MPGA wird eine neue Methode des nichtlinearen Ranking vorgestellt. Diese erlaubt die direkte Kontrolle des selektiven Drucks genau wie beim linearen Ranking.

Nichtlineares Ranking:

$$Fitneß(Pos) = \frac{Nind \cdot X^{Pos-1}}{\sum_{i=1}^{Nind} X^{i-1}}.$$

X wird als die Lösung des folgenden Polynoms berechnet:

$$0 = (SP - 1) \cdot X^{Nind-1} + SP \cdot X^{Nind-2} + \ldots + SP \cdot X + SP.$$

Bild 1 vergleicht lineares und nichtlineares Ranking graphisch.

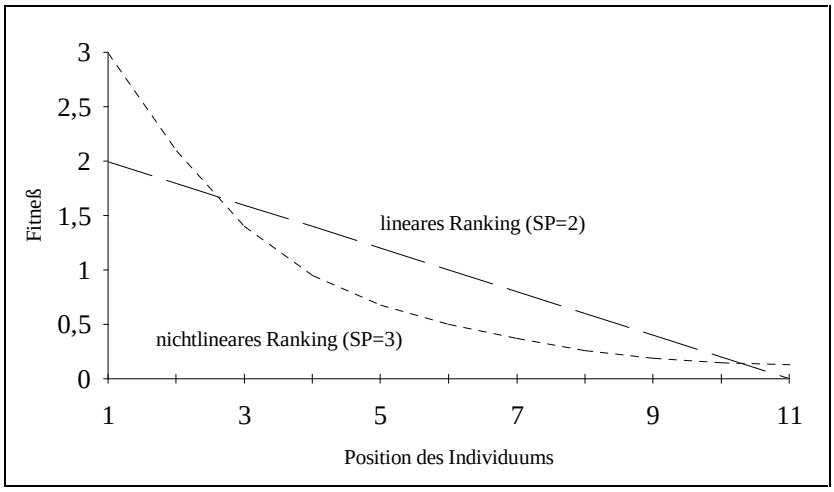

Bild 1: Fitneßzuweisung für lineares und nichtlineares Ranking

Die Wahrscheinlichkeit jedes Individuums selektiert zu werden ist seine Fitneß normalisiert mit der gesamten Fitneß der Population.

2.1.2. Roulette wheel Selektion

Das einfachste Selektionsverfahren ist Roulette-wheel-Selektion, auch stochastic sampling with replacement (siehe [1]) genannt. Dies ist ein stochastisches Verfahren und funktioniert wie folgt:

Alle Individuen werden in aufeinanderfolgenden Segmenten auf einer Linie angeordnet, wobei die Größe des Segmentes jedes Individuums proportional zu seiner Fitneß ist. Eine Zufallszahl wird erzeugt und das Individuum ausgesucht, dessen Segment diese Nummer überspannt. Der Prozeß wird so oft wiederholt, bis die notwendige Zahl von Individuen ausgewählt ist. Diese Technik entspricht einem Roulette, wobei jedes Segment proportional zur Fitneß eines Individuums ist.

Dieser Algorithmus hat keinen bias (bias: die absolute Differenz zwischen der normalisierten Fitneß eines Individuums und seiner erwarteten Fortpflanzungswahrscheinlichkeit). Er garantiert aber keinen minimalen spread (spread: Bereich der möglichen Werte für die Anzahl der Nachkommen eines Individuums).

2.1.3. Stochastic universal sampling

Beim stochastic universal sampling (siehe [1]) werden die Individuen in aufeinanderfolgenden Segmenten auf einer Linie angeordnet, wobei die Größe des Segmentes jeden Individuums proportional zu seiner Fitneß ist, genau wie bei roulette-wheel Selektion.. Bei dieser Technik werden so viele Zeiger in gleichmäßigem Abstand über dieser Linie angeordnet, wie Individuen selektiert werden sollen. Wenn *NSel* die Anzahl der zu selektierenden Individuen ist, dann ist die Distanz zwischen den Zeigern 1/*NSel* und die Position des ersten Zeigers wird durch eine zu erzeugende Zufallszahl im Bereich [0, 1/*NSel*] gegeben. Es werden die Individuen ausgewählt, in deren Segmenten die Zeiger stehen.

Stochastic universal sampling bietet keinen bias und minimalen spread und sichert eine Selektion der Nachkommen, die näher an dem ist, was erwartet wird.

2.2. Rekombination

Wie in der Natur werden durch Rekombination neue Individuen produziert, die Teile der Information von beiden Eltern in sich vereinen. Rekombination findet im allgemeinen zwischen Paaren von Individuen (den selektierten Eltern) statt. Jedes Paar produziert pro Generation zwei Nachkommen. Im folgenden werden verschiedene Rekombinationsverfahren für Individuen in der reellen Repräsentation beschrieben (siehe [9] und [11]).

Diskrete Rekombination führt einen Austausch der Variablenwerte zwischen den Individuen durch. Für jede Variable eines Nachkommen wird zufällig und mit gleicher Wahrscheinlichkeit entschieden, welcher Elter seinen Variablenwert an diesen Nachkommen weitergibt.

Diskrete Rekombination kann die Ecken eines Hyperwürfels generieren, der durch die Variablenwerte der Eltern definiert ist. Diskrete Rekombination kann für jede Art von Variablen genutzt werden, egal ab in reeller oder binärer Repräsentation.

Intermediate Rekombination ist eine Methode, die nur auf reellwertige Variablen angewandt werden kann. Hier wird der Variablenwert des Nachkommen innerhalb bzw. knapp außerhalb des Bereiches gewählt, den die entsprechenden Variablen der Eltern aufspannen. Die Nachkommen werden damit nach folgender Gleichung gebildet:

$$Nachkomme = Elter1 + Alpha \cdot (Elter2 - Elter1).$$

Alpha ist ein Skalierungsfaktor und wird zufällig gleichverteilt im Intervall [-0.25, 1.25] gewählt. Jede Variable in einem Nachkommen ist das Resultat der Kombination der Variablen der Eltern nach obenstehendem Ausdruck, wobei für jede Variable ein neues *Alpha* gewählt wird.

Intermediate Rekombination kann jeden Punkt generieren, der innerhalb eines Hyperwürfels etwas größer als von den Eltern definiert liegt.

Line Rekombination ist ähnlich der intermediate Rekombination. Doch hier wird nur ein *Alpha* pro Nachkommen erzeugt und für alle seine Variablen verwendet.

Line Rekombination kann jeden Punkt auf der Linie generieren, die durch die Variablen der Eltern definiert ist.

2.3. Mutation

Nach der Rekombination wird jeder Nachkomme mutiert. Die Variablen der Nachkommen werden durch die Addition kleiner Zufallswerte (die Mutationsschrittweite) zu den Variablen mutiert. Dieser Prozeß findet mit einer niedrigen Wahrscheinlichkeit statt, wobei die Wahrscheinlichkeit für das Mutieren einer Variablen mit zunehmender Variablenzahl pro Individuum abnimmt, die Mutationswahrscheinlichkeit ist invers proportional zur Variablenzahl.

Die Größe des Mutationsschrittes ist nicht leicht zu wählen. Die optimale Schrittweite hängt vom zu behandelnden Problem ab und kann sogar während des Optimierungsprozeßes veränderlich sein. Kleine Schritte sind oft erfolgreich, manchmal ist man aber mit größeren Schritten schneller.

In [9] und [11] wird solch ein Operator beschrieben (der Mutationsoperator des Breeder Genetic Algorithm):

$$\textit{mutierte Variable} = \textit{Variable} \pm \textit{Bereich} \cdot \textit{delta} \qquad (+\text{ oder } -\text{ mit gleicher Wahrscheinlichkeit}),$$

$$\textit{Bereich} = 0{,}5 \cdot \textit{Domain der Variable} \qquad (\text{Suchintervall}),$$

$$\textit{delta} = \sum_{i=0}^{m-1} \alpha_i 2^{-i} \quad \alpha_i = 1 \text{ mit Wahrscheinlichkeit } \frac{1}{m}, \text{ sonst } 0;\ m = 20.$$

Dieser Mutationsoperator kann jeden Punkt in einem Hyperwürfel generieren, der durch die Variablen des Individuums und den Bereich der Mutation definiert wird. Dieser Operator testet aber öfter in der Nähe einer Variablen, d.h., die Wahrscheinlichkeit von kleinen Schrittweiten ist größer als die von großen Schritten (nichtlineare Verteilung). Mit m=20 kann der Mutationsoperator das Optimum bis zu einer Genauigkeit von $\textit{Bereich} \cdot 2^{-19}$ erreichen.

2.4. Reinsertion

Wenn die Nachkommen durch Selektion, Rekombination und Mutation aus den Individuen der alten Population produziert sind, wird die Fitneß der Nachkommen bestimmt. Wenn die Nachkommen nicht die neue Population bilden sollen, so muß ein Reinsertion-Verfahren angewandt werden, um zu entscheiden, welche Nachkommen in die Population eingefügt werden.

Es existieren verschieden Verfahren für das Einfügen von Nachkommen:

- Produziere so viele Nachkommen wie Eltern und ersetze alle Eltern durch Nachkommen (jedes Individuum lebt nur eine Generation - pure reinsertion).
- Produziere weniger Nachkommen als Eltern und ersetze die Eltern gleichmäßig zufällig durch die Nachkommen (uniform reinsertion).
- Produziere weniger Nachkommen als Eltern und ersetze die schlechtesten Eltern durch die Nachkommen (elitist reinsertion).
- Produziere mehr Nachkommen als für das Einfügen benötigt werden und füge nur die besten Nachkommen ein (fitness-based reinsertion).

In MPGA stehen alle diese Verfahren zur Auswahl. Empfohlen wird die Verwendung der elitest kombiniert mit der Fitness-based-Reinsertion. In jeder Generation wird eine gegebene Anzahl der schlechtesten Eltern durch die Besten der Nachkommen ersetzt (siehe Bild 2). Wenn mehr Nachkommen als benötigt produziert wurden, führt Fitness-based-Reinsertion eine Truncation-Selektion zwischen den Nachkommen durch, d.h. die schlechtesten Nachkommen werden aussortiert, bevor sie in den Fortpflanzungsprozeß eingreifen können.. Auf der anderen Seite können die besten Individuen (Eltern) viele Generationen leben. Der Algorithmus stellt jedoch sicher, daß in jeder Generation einige Nachkommen eingefügt werden. Dabei wird nicht überprüft, ob die eingefügten Nachkommen fitter sind als die Eltern, die sie ersetzen.

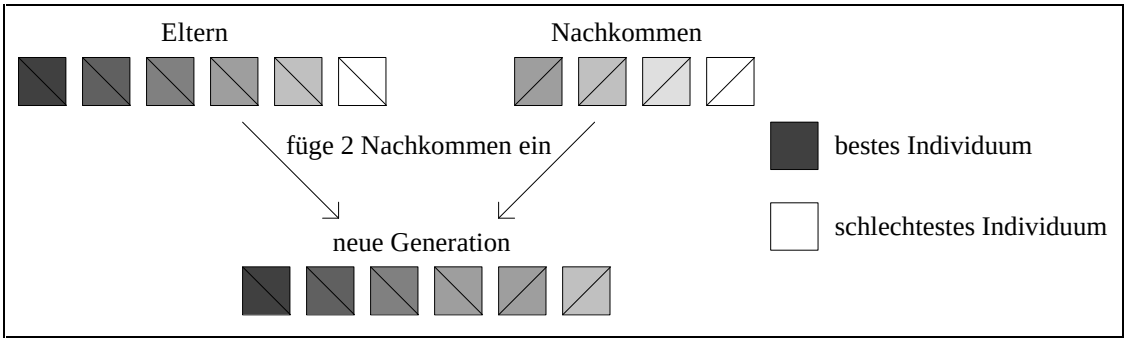

Bild 2: Schema für elitist und fitness-based Reinsertion

2.5. Mehrfachpopulationen/Parallelität

Parallele Algorithmen wurden zur Beschleunigung von Berechnungen entwickelt, wobei die Möglichkeiten paralleler Computer genutzt werden. Für die Implementierung von parallelen genetischen Algorithmen gibt es drei verschiedene Ansätze: das globale, das Diffusions- und das Migrationsmodell. Beim globalen Modell wird die in den genetischen Algorithmen liegende Parallelität ausgenutzt, die durch die Arbeit an Populationen von Individuen entsteht. Das Diffusionsmodell behandelt jedes Individuum separat, das sich seinen Fortpflanzungspartner in der Nachbarschaft sucht. Durch die lokale Selektion und Reproduktion entsteht eine Diffusion der Individuen über die Population.

Im Migrationsmodell wird die Gesamtpopulation in mehrere Unterpopulationen eingeteilt, die sich für eine festzulegende Zeit oder Anzahl von Generationen (Isolationszeit) unabhängig voneinander entwickeln. Nach dieser Zeit werden einige Individuen zwischen den Unterpopulationen ausgetauscht (Migration). Die Anzahl der auszutauschenden Individuen (die Migrationsrate), die Art der Auswahl der Individuen für die Migration und das Schema der Migration entscheiden über die genetische Vielfalt, die sich in den Unterpopulationen entwickeln kann und den Austausch an Informationen zwischen den Unterpopulationen. Mit der parallelen Implementierung des Migrationsmodell kann nicht nur eine Verringerung der Berechnungszeiten erreicht werden, sondern die Qualität der Lösung ist insgesamt größer, verglichen mit einer globalen Population oder vollständig isolierten Unterpopulationen von derselben Gesamtgröße. Damit liefert die Implementierung des Migrationsmodelles in einer seriellen Weise (pseudo-parallel) auch bessere Resultate (der Algorithmus findet das Optimum öfter oder mit weniger Zielfunktionsberechnungen). Im MPGA ist das Migrationsmodell implementiert.

Die Auswahl der Individuen für Migration kann erfolgen:

- gleichmäßig zufällig (wähle die Individuen für Migration zufällig aus) oder
- fitness-based (wähle die besten Individuen für Migration).

Viele Möglichkeiten bestehen für die Struktur der Migration der Individuen zwischen den Unterpopulationen. Migration kann stattfinden:

- zwischen allen Unterpopulationen (komplette Netzstruktur - uneingeschränkt), siehe Bild 3,
- in einer Nachbarschaftsstruktur, siehe Bild 4,
- in einer Ringstruktur.

Die allgemeinste Strategie für die Migration ist die **uneingeschränkte Migration**. Hierbei können die Individuen von jeder zu jeder Unterpopulation migrieren. Für jede Unterpopulation wird ein Pool von potentiellen Migranten aus den anderen Unterpopulationen gebildet. Aus diesem Pool wird die benötigte Anzahl von Individuen gleichmäßig zufällig ausgewählt.

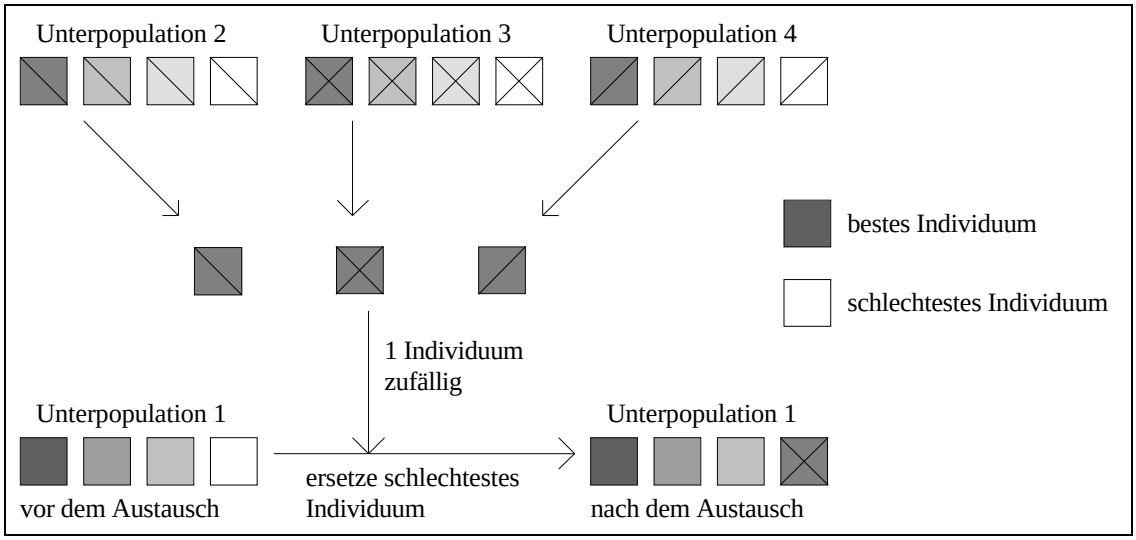

Bild 3: Schema der Migration von Individuen zwischen Unterpopulationen

Bild 3 zeigt eine graphische Beschreibung der uneingeschränkten Migration für 4 Unterpopulationen mit fitness-based Selektion. Die Unterpopulationen 2, 3 und 4 bilden einen Pool ihrer besten Individuen. Ein Individuum wird aus diesem Pool gleichmäßig zufällig ausgewählt und ersetzt das schlechteste Individuum in Unterpopulation 1. Diese Schleife wird für jede Unterpopulation durchlaufen. Damit wird gesichert, daß keine Unterpopulation Individuen von sich selbst erhält.

Bei der **Nachbarschaftsstruktur** (siehe Bild 4) findet die Migration von Individuen nur zwischen benachbarten Unterpopulationen statt. Für jede Unterpopulation findet die Wahl der möglichen Migranten nach dem gewünschten Verfahren (gleichmäßig zufällig oder fitness-based) aus den beiden benachbarten Unterpopulationen statt. Aus diesem Pool werden die einzufügenden Individuen ausgewählt (ähnlich wie im Bild 3).

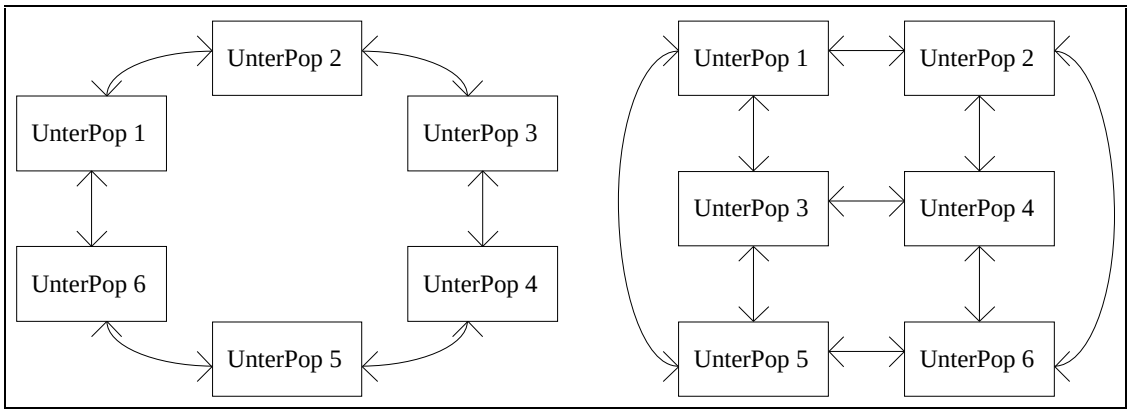

Bild 4: Nachbarschaftsmigrationsstruktur in einer und zwei Dimensionen

Bild 4 zeigt zwei verschiedene Möglichkeiten der Nachbarschaftsstruktur der Migration. Auf der linken Seite ist die eindimensionale Variante zu sehen. Die rechte Seite zeigt die Erweiterung der Nachbarschaftsstruktur auf zwei Dimensionen. An manchen Stellen wird diese Struktur torus genannt.

Die einfachste Strategie für die Migration ist die **Ringstruktur**. Diese ist ähnlich der Nachbarschaftsstruktur (siehe Bild 4). Hierbei werden Individuen nur in einer Richtung zwischen benachbarten Unterpopulationen transferiert. Zum Beispiel migrieren Individuen von Unterpopulation 6 nur zur Unterpopulation 1 und Individuen von Unterpopulation 1 nur zur Unterpopulation 2 und so weiter.

Mit dem Algorithmus mit Mehrfachpopulationen wurden für viele der getesteten Funktionen bessere Ergebnisse erzielt als mit dem Algorithmus mit einer Population. Gleichlautende Resultate wurden auch in [7], [10], [13], [15], [16] und [17] berichtet.

2.6. Beispielfunktion

Im Anschluß an die Darstellung von MPGA soll an dieser Stelle ein Beispiel mit einer oft genutzten Testfunktion gezeigt werden, Rastrigin's Funktion 6.

Die Grundlage für Rastrigin's Funktion 6 ist De Jong's Funktion 1:

$$f_1(x) = \sum_{i=1}^{n} x_i^2$$

Durch die Addition einer Kosinus-Modulation werden viele lokale Minima produziert, womit diese Testfunktion multimodal wird. Jedoch folgt die Lage der lokalen Minima einer regulären Struktur.

$$f_6(x) = 10 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos\left(2 \cdot \pi \cdot x_i\right) \right) \qquad -5{,}12 \le x_i \le 5{,}12$$

globales Minimum: $x_i = 0 \qquad f(x) = 0$

lokale Minima: $f(x) = 1, 2, 3, \ldots$

Mit dieser Testfunktion wurden verschiedene Minimierungsläufe durchgeführt, wobei das Verhalten bei unterschiedlicher Gliederung der Gesamtpopulation in Unterpopulationen untersucht wurde.

Die für alle Läufe konstanten Parameter von MPGA waren: Dimension der Testfunktion = 20, Individuen = 120, Generation gap = 0.9, Mutationsrate =0.05 (1/Dimension), Migrationszeit = 20 (alle 20 Generationen), Migrationsrate = 0.15, uneingeschränkte Migration, Selektionsdruck = 2, linear Ranking, stochastic universal sampling, diskrete Rekombination. Die Minimierung erfolgte über 500 Generationen.

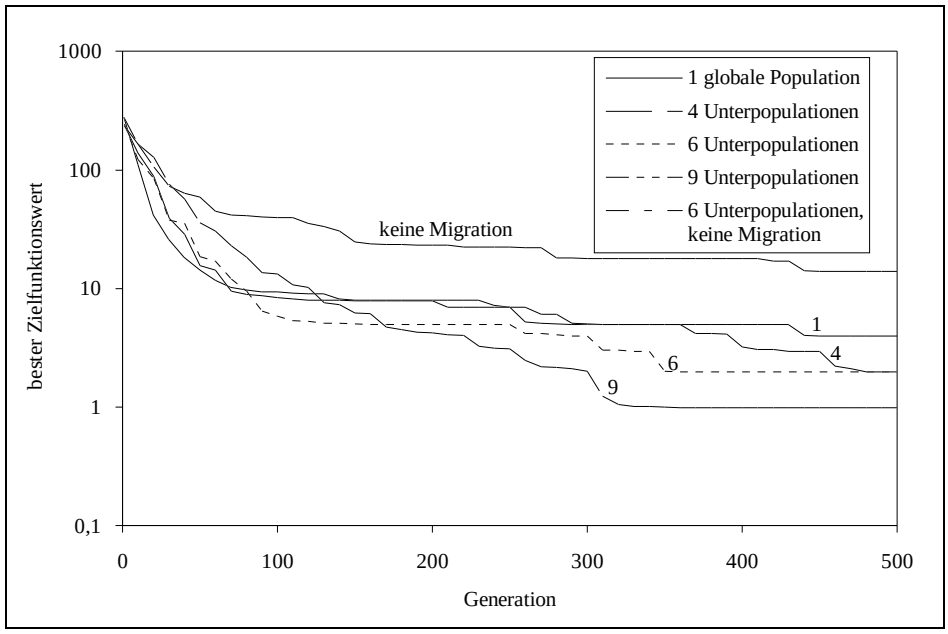

Bild 5: Bester Zielfunktionswert bei verschiedener Unterteilung der Gesamtpopulation

Bei 120 Individuen ist eine Unterteilung der Population in 1, 2, 3, 4, 6, 9 ,10 und 12 Unterpopulationen möglich. Unterpopulationen mit weniger als 10 Individuen sind nicht mehr sinnvoll. Bild 5 zeigt den Verlauf der Zielfunktionswerte des jeweils besten Individuums aller Unterpopulationen für die Läufe mit 1, 4, 6 und 9 Unterpopulationen über 500 Generationen und zum Vergleich 6 Unterpopulationen ohne Migration.

Zu Beginn der Minimierung ist eine einzige/globale Population schneller als die unterteilten Populationen. Dieses Bild ändert sich aber mit fortschreitender Minimierung. Ab der 150. Generation haben die Mehrfachpopulationen kleinere Zielfunktionswerte. In je mehr Unterpopulationen die Population geteilt ist, umso besser werden die Ergebnisse. Allerdings darf diese Unterteilung nicht übertrieben werden. Sobald die Unterpopulationen zu klein werden, verschlechtern sich die Minimierungsergebnisse (hier nicht dargestellt). Sehr deutlich ist zu sehen, daß ohne Migration zwischen den Unterpopulationen die Qualität der Lösung stark absinkt. Die einzelnen, kleinen Populationen bleiben in lokalen Minima stecken. In diesem Beispiel wurde nicht bis zum Erreichen des globalen Minimums gerechnet, sondern über eine feste Anzahl von Generationen. Dies ist bei der praktischen Anwendung das am häufigsten verwendete Abbruchskriterium.

3. Dynamische Optimierung

Dynamische Steuerungsprobleme sind komplex und nicht einfach zu lösen. Die Verwendung von spezifischen Methoden für die dynamische Optimierung, z.B. der Hamiltonfunktion, sind kompliziert und problematisch. Die Anwendung der speziellen Methoden verlangt eine große Portion an mathematischer Unterstützung selbst für Probleme von überschaubarer Größe. Nur die trivialsten Systeme können analytisch gelöst werden.

In dem folgenden Beispielproblem korrespondiert jedes Individuum im genetischen Algorithmus mit einem (diskreten) Steuervektor. Jede Variable in einem Individuum entspricht dem Steuerwert zu einem Zeitschritt des dynamischen Optimierungsproblems. In diesem Abschnitt ist x der Zustandsvektor und u die Steuergröße des Systems.

Ein **Doppelintegrator** (push-cart system: geradlinige Bewegung eines Körpers unter dem Einfluß einer Kraft) soll energieoptimal aus einem vorgegebenen Anfangszustand in einen vorgegebenen Endzustand gesteuert werden. Die Steuergröße *u* ist die auf den Körper wirkende Kraft, die Zustandsgrößen sind die Geschwindigkeit x_1 und der Weg/die Position x_2 des Körpers. Das System wird durch die folgenden Zustandsgleichungen beschrieben.

$$\dot{x}_1 = u$$
$$\dot{x}_2 = x_1$$
$$y = x_2$$

Die folgenden Kriterien sind zu erfüllen:

- Zeit: $0 \le t \le 1$
- Anfangsbedingungen: $x_1(0) = 0$ $x_2(0) = -1$
- Endbedingungen: $x_1(1) = 0$ $x_2(1) = 0$

Das zu minimierende Zielfunktional ist:

- $f(u) = \int_{t=0}^{1} u(t)^2 \, dt$.

Für diese Aufgabenstellung kann die analytische Lösung gefunden werden:

- $u = 6 - 12 \cdot t$, Minimum = 12.

Bild 6 zeigt den Verlauf von optimalem Steuervektor und der Zustandsvektoren für das kontinuierliche System.

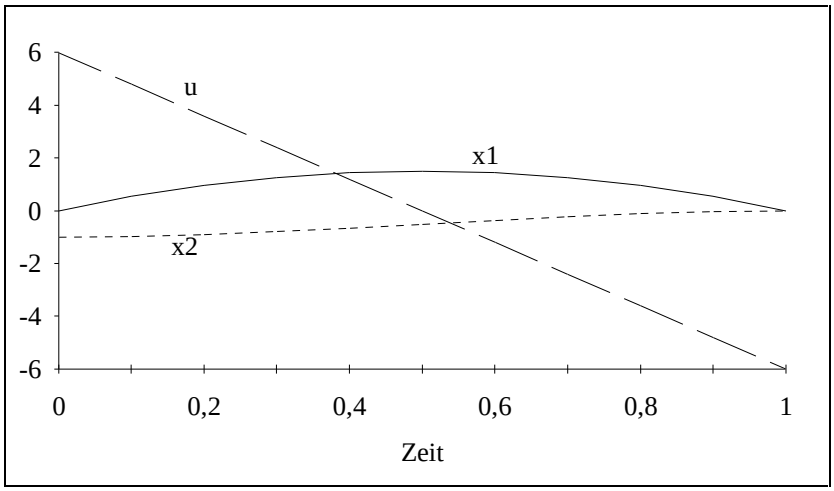

Bild 6: Steuer- und Zustandsvektoren des Doppelintegrators für die optimale Lösung

Der Doppelintegrator wurde in drei Varianten, unter Verwendung eines SIMULINK-Modells, einer S-Funktion und von *Control-System-Toolbox*-Routinen implementiert.

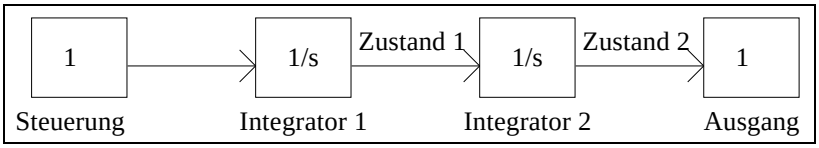

Bild 7: Blockdiagramm des Doppelintegrators

Die SIMULINK-Methode verwendet das Modell im Bild 7. Die S-Funktion löst die linearen Gleichungen, die oben gegeben sind. Die dritte Methode (*Control System Toolbox*) benutzt die Übertragungsfunktion:

$$G(s) = \frac{1}{s^2}.$$

Eine typische Technik zum Einbeziehen von Beschränkungen ist die Verwendung einer Straffunktion (penalty function). Diese Methode wurde hier angewandt. Eine bewertete Differenz der Endzustände zu den geforderten Endbedingungen wird zum Zielfunktionswert addiert. Damit wird das mehrkriterielle Problem in ein einkriterielles Problem transformiert und kann mit der jetzigen Form von MPGA bearbeitet werden.

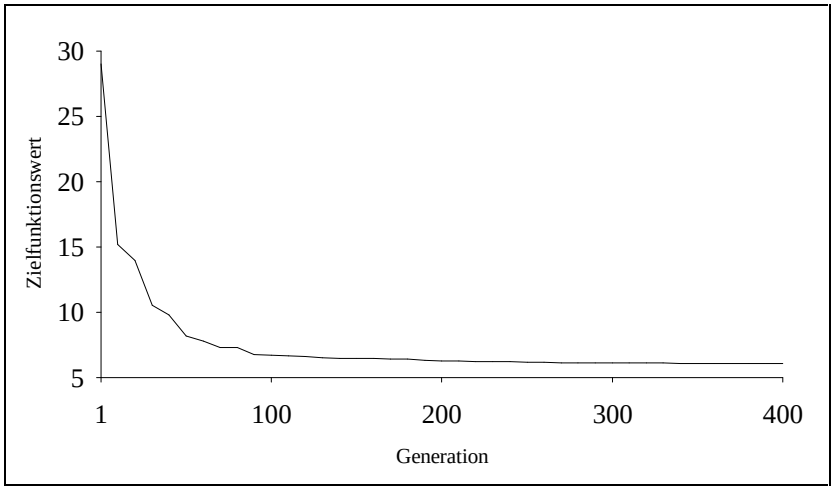

Bild 8: Zielfunktionswert des besten Individuums pro Generation für eine Optimierung des Doppelintegrators

Bild 8 zeigt den Verlauf einer Optimierung des Doppelintegrators mit MPGA. Dabei wurden folgende Parameter verwendet: Dimension des Modells = 20, Individuen = 120, Unterpopulationen = 6, Generation gap = 0.9, Mutationsrate =0.05 (1/Dimension), Migrationszeit = 20 (alle 20 Generationen), Migrationsrate = 0.15, uneingeschränkte Migration, Selektionsdruck = 2, linear Ranking, stochastic universal sampling, diskrete Rekombination. Die Minimierung erfolgte über 400 Generationen. Die Lösung dieses Optimierungslaufes ist identisch zum Graphen in Bild 6.

Obwohl das gezeigte Beispiel recht einfach ist, zeigt es doch die grundlegende Herangehensweise. Mit dem Verständnis des Doppelintegrators und von MPGA ist es im weiteren ohne größere Probleme möglich, das System des Doppelintegrators gegen andere Systeme in einer der vorgestellten Darstellungsvarianten auszutauschen. Bis auf einige Parameteranpassungen, die zum großen Teil automatisch erfolgen, verändert sich MPGA nicht. Es ist möglich, sich voll und ganz auf das zu lösende Problem zu konzentrieren.

4. Ausblick

In diesem Beitrag wurde ein Überblick über den genetischen Algorithmus mit Mehrfachpopulationen gegeben. Seine verschiedenen Bestandteile wurden ausführlich dargestellt und bei mehreren Varianten auf die Anwendungsmöglichkeiten verwiesen. Im Anschluß daran wurden an einem Beispiel die Vorteile der Unterteilung der Gesamtpopulation in Unterpopulationen und der Migration von Individuen gezeigt. Den Abschluß bildete die Darstellung eines dynamischen Optimierungsproblemes, das als Grundlage für weitere Arbeiten in dieser Richtung dienen kann.

Aus dem hier präsentierten Material dürfte deutlich geworden sein, daß MPGA ein leistungsfähiges Werkzeug zur Optimierung darstellt. MPGA ist robust, schnell und bietet zudem die Möglichkeit der einfachen Umsetzung auf parallele Plattformen.

Mit der *Genetic Algorithm Toolbox*, in der MPGA enthalten ist, steht dieser Algorithmus jedem Anwender in einer vertrauten und einheitlichen Umgebung zur Verfügung. Die Anwendung dürfte sich nicht nur für den Regelungs- und Systemtechniker als nützlich erweisen, sondern auch auf dem allgemeinen Feld der evolutionären Algorithmen, speziell wenn die große Anzahl von aufgabenspezifischen Toolboxen für MATLAB betrachtet wird.

Die aktuelle Version von MPGA ist auf einkriterielle Probleme beschränkt. An der University of Sheffield wird an einer Weiterentwicklung der Toolbox gearbeitet, die dann auch die Lösung mehrkriterieller Probleme ermöglicht (siehe [3]). Außerdem kann eine Erweiterung des Mehrfachpopulationenkonzeptes in der Richtung erfolgen, daß jede der Unterpopulationen mit einem eigenen Satz an Parametern arbeitet. Damit können in einem Rechengang verschiedene Strategien parallel verfolgt werden.

Literatur

[1] *Baker, J. E.*: Reducing Bias and Inefficiency in the Selection Algorithm. Proceedings of the Second International Conference on Genetic Algorithms and their Application, S.14-21, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, 1987.

[2] *Chipperfield, A., Fleming, P. J., Pohlheim, H. und Fonseca, C. M.*: Genetic Algorithm Toolbox for use with Matlab. Technical Report No. 512, Department of Automatic Control and Sytems Engineering, University of Sheffield, 1994.

[3] *Fonseca, C. M. und Fleming P. J.*: Genetic Algorithms for Multiple Objective Optimization: Formulation, Discussion and Generalization. Proceedings of the Fifth International Conference on Genetic Algorithms and their Application, S. 416-423, San Mateo, California, USA: Morgan Kaufmann Publishers, 1993.

[4] *Goldberg, D. E.*: Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Mass., USA: Addison-Wesley, 1989.

[5] *Hoffmeister, F. und Bäck, T.*: Genetic Algorithms and Evolutionary Strategies: Similarities and Differences. Proceedings of Parallel Problems Solving from Nature, S.455-469, Volume 496 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1991.

[6] *Holland, J. H.*: Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press, 1975.

[7] *Lohmann, R.*: Application of Evolution Strategy in Parallel Populations. Proceedings of Parallel Problems Solving from Nature I, S.198-208, Volume 496 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1991.

[8] *Michalewicz, Z.*: Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Heidelberg, New York: Springer-Verlag, 1992.

[9] *Mühlenbein, H. und Schlierkamp-Voosen, D.*: Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization. Evolutionary Computation, 1 (1), S. 25-49, 1993.

[10] *Mühlenbein, H., Schomisch, M. und Born, J.*: The parallel genetic algorithm as function optimizer. Parallel Computing, 17, S.619-632, 1991.

[11] *Mühlenbein, H.*: The Breeder Genetic Algorithm - a provable optimal search algorithm and its application. Colloquium on Applications of Genetic Algorithms, IEE 94/067, London, 1994.

[12] *Rechenberg, I.*: Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: Frommann-Holzboog, 1973.

[13] *Rudolph, G.*: Global Optimization by Means of Distributed Evolution Strategies. Proceedings of Parallel Problems Solving from Nature I, S.209-213, Volume 496 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1991.

[14] *Schwefel, H.-P.*: Numerical optimization of computer models. Chichester: J. Wiley & Sons, 1981.

[15] *Starkweather, T. Whitley, D. und Mathias, K.*: Optimization using Distributed Genetic Algorithms. Proceedings of Parallel Problems Solving from Nature I, S.176-185, Volume 496 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1991.

[16] *Voigt, H.-M., Born, J. und Santibanez-Koref, I.*: Modelling and Simulation of Distributed Evolutionary Search Processes for Function Optimization. Proceedings of Parallel Problems Solving from Nature, S.373-380, Volume 496 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1991.

[17] *Voigt, H.-M., Santibanez-Koref, I. und Born, J.*: Hierarchically Structured Distributed Genetic Algorithm. Technical Report, TU Berlin, 1993.

[18] *Whitley, D.*: The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. Proceedings of the Third International Conference on Genetic Algorithms and their Application, S.116-121, San Mateo, California, USA: Morgan Kaufmann Publishers, 1989.

[19] *Wright, A. H.*: Genetic Algorithms for Real Parameter Optimization. Foundations of Genetic Algorithms, S.205-218, San Mateo, California, USA: Morgan Kaufmann Publishers, 1991.