Optimierung mit Genetischen Algorithmen und eine Anwendung zur Modellreduktion

Optimization with Genetic Algorithms and an Application for Model Reduction

Maik Buttelmann und Boris Lohmann

Genetische Algorithmen sind unter gewissen Voraussetzungen in der Lage, auch komplexe Optimierungsprobleme zu behandeln und aus einer sehr großen Zahl von möglichen Lösungen die beste oder zumindest sehr gute im Sinne eines Gütemaßes zu ermitteln. Nach dem Vorbild natürlicher Auslese und Fortpflanzung werden dabei Lösungskandidaten hoher Güte bevorzugt zur Erzeugung neuer Kandidaten verwendet, in der Erwartung, schrittweise zu noch besseren Lösungen zu gelangen. Der Beitrag gibt eine Einführung in die Grundlagen Genetischer Algorithmen und illustriert eine ingenieurtechnische Anwendung bei der Reduktion und Strukturfindung nichtlinearer Systemmodelle.

This paper gives an overview of Genetic Algorithms. If some preconditions are fulfilled, Genetic Algorithms can be used to solve complex optimisation problems with a huge number of possible solutions. With the help of a fitness function and the genetic operators selection, recombination, and mutation better and better solutions can be generated to find the best one. The basics of Genetic Algorithms are discussed and an example is given.

Schlagwörter: Genetische Algorithmen, Optimierung, Suchverfahren, Modellreduktion, nichtlineare Systeme

Keywords: Genetic algorithms, optimization, search algorithms, model reduction, non-linear systems

1 Einleitung

Bei Optimierungsproblemen soll aus einer großen Menge möglicher Lösungen diejenige gefunden werden, die ein Zielkriterium (oder auch mehrere) bestmöglich erfüllt. Zumeist wird das Zielkriterium in Gestalt eines Gütemaßes formuliert, durch das jedem Lösungskandidaten eine reelle Gütemaßzahl zugewiesen werden kann. Wird die Menge der möglichen Lösungen speziell durch reellwertige Objektparameter $x_1, \ldots, x_M$ beschrieben, so kann das Optimierungsproblem formuliert werden durch:

$$\min \left\{ F(\mathbf{x}),\ \mathbf{x} = [x_1 \ldots x_M]^T \in \mathbf{X}^M,\ \mathbf{X}^M \subseteq \mathbb{R}^M \right\} \qquad (1)$$
$$\text{im Suchraum } \mathbf{X}^M = \left\{ \mathbf{x} \mid u_j \leq x_j \leq o_j \quad (1 \leq j \leq M) \right\}$$

wobei $F : \mathbf{X}^M \rightarrow \mathbb{R}$ die Zielfunktion, auch Güte- oder Qualitätsfunktion genannt wird sowie u_j und o_j Beschränkungen der Parameter beschreiben. Neben solchen Beschränkungen können auch anders geartete Nebenbedingungen zur Zielfunktion hinzutreten. Ziel ist es, denjenigen Satz von Objektparametern zu finden, der die Zielfunktion unter Beachtung der Nebenbedingungen minimiert. Die Lösung dieser Aufgabe wird erleichtert, wenn die Zielfunktion stetig und differenzierbar ist. Gradientenverfahren mit ihren vielfältigen Ausprägungen [1; 2] bieten sich in diesem Falle an, können aber das Auffinden des globalen Optimums nicht garantieren.

Ist die Zielfunktion unstetig und/oder nicht differenzierbar, sind andere Verfahren überlegen, die den Suchraum über lokale Extrema „hinausblickend“ absuchen. Zu diesen Verfahren gehören die Heuristiken: *Unter einer Heuristik soll hier eine nichtwillkürliche und häufig iterative Methode verstanden werden, die darauf abzielt, für eine gegebene Problemstellung in begrenzter Zeit eine oder mehrere möglichst gute Lösungen zu finden, ohne dass garantiert werden kann, eine global optimale Lösung zu finden.* ([7], Seite 18). Eine Gruppe solcher Verfahren sind die Evolutionären Algorithmen (EA). Zu ihnen gehören die in den 1960er Jahren von Ingo Rechenberg entwickelten Evolutionären Strategien und die zur gleichen Zeit unabhängig davon entwickelten Genetischen Algorithmen, die erstmals

von John Holland [3] vorgestellt und vor allem durch die Arbeiten des Holland-Schülers David Goldberg [4] verbreitet wurden. Die Entwicklung dieser beiden Verfahren fand lange Zeit parallel statt, erst in den 1990er Jahren erfolgte eine Annäherung durch gemeinsame Konferenzen. In diesem Artikel wird allerdings nur auf die Genetischen Algorithmen eingegangen. Weiterführende Literatur für die Evolutionären Strategien findet man in [18; 19].

Die Genetischen Algorithmen basieren auf dem Evolutionsgedanken der Natur. Basis ist eine Population von Individuen, die mithilfe der genetischen Operationen *Selektion* und *Variation* Nachkommen erzeugen, die hinsichtlich des zu lösenden Optimierungsproblems eine bessere Lösung darstellen, als ihre Eltern. Die Selektion gibt dabei dem Genetischen Algorithmus die Richtung im Suchraum vor, indem gute Individuum für die Fortpflanzung bevorzugt und schlechte gemieden werden. Dem dadurch auftretenden Diversitätsverlust, dem Absinken der Vielfalt in der Population, wirken die Variationsoperatoren *Rekombination* und *Mutation* entgegen, indem sie neue Lösungen (Varianten) aus den vorhandenen erzeugen.

Der folgende Abschnitt soll dem Leser einen Einblick in die Thematik der Genetischen Algorithmen geben, wobei das Hauptaugenmerk auf die praktische Umsetzung gerichtet ist. Ein kompletter theoretischer Überblick ist nicht möglich, moderne Übersichtswerke sind [7–11]. Als Anwendungsfeld wird in Abschnitt 3 dann die Ordnungsreduktion und Strukturfindung nichtlinearer Modelle dynamischer Systeme vorgestellt. Genetische Algorithmen in enger Verzahnung mit einem Ordnungsreduktionsverfahren führen hierbei zu kleinen Modellen einfacher innerer Struktur. Anders als bei herkömmlichen Reduktionsverfahren bleibt das reduzierte Modell physikalisch interpretierbar und das Blockschaltbild übersichtlich.

2 Genetische Algorithmen

Die Stärke der Genetischen Algorithmen (GA) ist ihre Fähigkeit, Lösungen selbst für sehr komplexe Optimierungsprobleme zu finden, bei denen andere Suchverfahren scheitern. Sie zeichnen sich durch ihre Variabilität aus, sodass sie an viele Probleme angepasst werden können. Zugleich ist diese Anpassbarkeit aber auch eine Schwäche, da es keinen allgemein gültigen „wirksamen“ Algorithmus gibt. So haben sich im Laufe der Jahre viele Varianten der GA gebildet, denen aber allen der von John Holland entwickelte Basis-Algorithmus [3] zugrunde liegt. Genetische Algorithmen sind probabilistische Algorithmen, die mithilfe einer Population von Individuen $P(t) = \{\mathbf{a}_1(t), \ldots, \mathbf{a}_n(t)\}$ (t: Iterationsschritt) eine parallele Suche im Suchraum des Optimierungsproblems durchführen. Dabei repräsentiert jedes Individuum $\mathbf{a}_i(t)$ eine mögliche Lösung des Optimierungsproblems, deren Güte, im GA meistens *Fitness* genannt, mithilfe der Zielfunktion (1) bestimmt wird.

Für diese Suche werden zwei Arten von Operatoren verwendet: *Variation* und *Selektion*. Durch die Selektion werden abhängig von der Fitness Individuen für die Reproduktion ausgewählt. So wird der Evolution eine Richtung gegeben, indem gute Zustände konserviert und schlechte Zustände eliminiert werden. Gleichzeitig wird dadurch allerdings die *Diversität* der Population, d. h. der Grad der Verschiedenartigkeit der Individuen, reduziert. Diesem Diversitätsverlust wirkt die *Variation* entgegen. Die Variationsoperatoren für den Genetischen Algorithmus sind *Rekombination* und *Mutation*, die beide die genetische Information der Individuen verändern und somit neue, verschiedenartige Individuen erzeugen.

Um eine neue Generation von Individuen zu generieren, werden jeweils Individuen aus der vorhandenen Population abhängig von ihrer Fitness selektiert, aus ihnen durch Variation zwei neue Individuen erzeugt und in die neue Population eingefügt. Man erhält so den folgenden simplen Basis-Algorithmus, auch als kanonischer GA bezeichnet:

Prozedur Genetischer Algorithmus
Beginn
  $t \leftarrow 0$
  Generiere $P(t)$ zufällig
  Berechne Fitnesswerte für $P(t)$
  Solange (**nicht** Abbruchkriterium erfüllt)
  Beginn
    $t \leftarrow t + 1$
    Bilden der neuen Population $P(t)$:
      Selektion
      Rekombination
      Mutation
    Berechne Fitnesswerte für $P(t)$
  Ende
Ende

Als Abbruchkriterium wird oft das Erreichen von t_{max}, die maximale Rechenzeit oder die Konvergenz im Suchraum verwendet. Der Algorithmus bietet eine breite Anwendbarkeit und eignet sich deshalb auch für komplexe Suchräume. Es gibt außerdem keine einschränkenden Anforderungen an die Zielfunktion, die weder stetig noch differenzierbar sein muss. Die Grundfunktionen sind einfach zu verstehen und auch bei geringen Kenntnissen der Problemstruktur anwendbar. Darüber hinaus bestehen Möglichkeiten, den Algorithmus mit anderen Verfahren zu kombinieren oder parallele Berechnungen auf mehreren Rechnern durchzuführen.

Den genannten Vorteilen stehen auch einige Nachteile gegenüber. So gibt es keine Garantie der Konvergenz des Algorithmus'. Unter Konvergenz versteht man dabei den Prozess der Annäherung an einen stationären Zustand, wobei zwei Arten von Konvergenz zu unterscheiden sind, die Gen-Konvergenz und die Konvergenz gegen Optimalzustände. Bei letzterer strebt der Algorithmus gegen ein lokales Optimum, oder im Idealfall gegen das globale Optimum. Bei der Gen-Konvergenz hingegen handelt es sich um eine vorzeitige Konvergenz, die durch einen Diversitätsverlust während der Suche hervorgerufen wird. Somit konvergiert der Algorithmus gegen Zustände, die nicht optimal sein können.

Bild 1 zeigt das Schema eines Genetischen Algorithmus'.

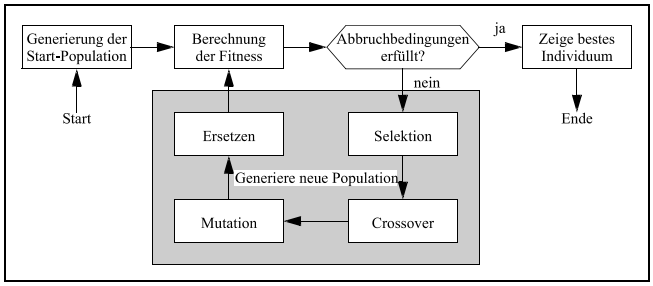

Bild 1: Ablaufplan eines Genetischen Algorithmus'.

Die einzelnen Komponenten werden im Folgenden beschrieben und es werden einige Hinweise gegeben, um den Basis-GA einem gegebenen Optimierungsproblem anzupassen.

2.1 Individuum

Um einen Genetischen Algorithmus einsetzen zu können, ist es erforderlich, mögliche Lösungen des Optimierungsproblems zu codieren. Dies geschieht in der Regel auf Zeichenketten, insbesondere Bitketten konstanter Länge[1]. In Anlehnung an den biologischen Evolutionsgedanken, der den GAs zugrunde liegt, heißen diese Ketten *Chromosomen* oder *Genome* und die einzelnen Werte der Ketten *Gene*. Ein Individuum ist der Träger dieser genetischen Informationen und wird charakterisiert durch seinen Zustand im Suchraum und seine Fitness. Der Fitnesswert ist dabei abhängig von den Genen, aus denen die Objektparameter und somit durch die Fitnessfunktion (1) die Fitness bestimmt wird.

Das Genom $\mathbf{a}_i$ eines Individuums besteht aus L Genen, deren Anzahl abhängig vom zu lösenden Problem ist. Es ist in m Segmente unterteilt, die jeweils die Informationen für einen Objektparameter x_j des Optimierungsproblems enthalten. Die Länge der einzelnen Objektparameter kann sich dabei unterscheiden, ist aber für jeden Parameter von Individuum zu Individuum gleich. Jedes Gen enthält einen Wert a, der als *Allel* bezeichnet wird und der bei binärer Codierung entsprechend die Werte 0 oder 1 annehmen kann. Das Genom eines Individuums kann demnach das folgende Aussehen haben:

$$\mathbf{a}_i = [\cdots 10\,|\,\underbrace{11001101}_{\text{Objekt-parameter } x_j}\,|\,\underbrace{011010101101}_{\text{Objekt-parameter } x_{j+1}}\,|\,01\cdots]$$

Der Objektparameter x_j enthält in diesem Beispiel 8 Bits, während x_{j+1} aus 12 Bits besteht.

Anmerkung: In der Literatur über GA werden auch häufig die Begriffe *Genotyp*[2] und *Phänotyp*[3] verwendet. Der Genotyp ist dabei die Repräsentation, auf die die Variationsoperatoren angewendet werden, also das Genom. Der Phänotyp ist dann die Ausprägung der Eigenschaften des Individuums, die durch seinen Genotyp codiert sind. Bei der Parameteroptimierung ist der Phänotyp mit den Objektparametern identisch, bei der Strukturoptimierung z. B. von Neuronalen Netzen stellt der Phänotyp hingegen eine bestimmte Struktur des Netzes dar.

Die Bitketten-Längen der einzelnen Objektparameter hängen nicht nur von dem betrachteten Problem ab, sondern auch von der gewählten *Codierung*. Soll z. B. der Objektparameter x_j einen kontinuierlichen Wert $-10 \le x_j \le 10$ enthalten mit einer Genauigkeit von mindestens 10^{-1}, so werden $20 \cdot 10 = 200$ Intervalle benötigt, die mit 8 Bits codiert werden ($128 = 2^7 < 200 < 2^8 = 256$). Die untere Intervallgrenze -10 wird durch die Bitkette 00000000 und die obere Grenze $+10$ durch 11111111 beschrieben. Alle übrigen Ketten werden linear auf den Definitionsbereich zwischen diesen Grenzen abgebildet. Bei der Decodierung wandelt man jede binäre Kette $a_1 a_2 \cdots a_7 a_8$ zuerst in eine Dezimalzahl um. Der decodierte x_j-Wert ergibt sich für obiges Beispiel dann zu:

$$x_j = -10 + \frac{10-(-10)}{2^8-1} \cdot \sum_{z=1}^{8} (a_{8-z+1}) \cdot 2^{z-1} \,.$$

Der Objektparameter $x_j = 11001101$ aus dem obigen Individuum ergibt demnach den Wert

$$x_j = -10 + \frac{20}{255} \cdot 205 \approx 6{,}1 \,.$$

Bei diesem Vorgehen ist allerdings zu beachten, dass die kontinuierlichen Werte nur angenähert werden, sodass auch das gesuchte Optimum nur angenähert werden kann. Ein Nachteil ist weiterhin, dass bei kleinen Änderungen der Bitkette (z. B. dem Kippen eines Bits) eine große Änderung des codierten Wertes auftreten kann. Um dieses zu verhindern, werden meist einschrittige Codes verwendet [27]. Sie haben die Eigenschaft, dass sich beim Übergang von einem Wert zum nächstfolgenden immer nur *ein* Bit ändert, also die Hamming-Distanz[4] benachbarter Zahlenwerte stets eins ist. Somit wird der Grundsatz *kleine Ursache, kleine Wirkung* erfüllt und die Suche nicht durch große Wertsprünge bei kleinen Bit-Änderungen erschwert. Eine geläufige Variante für einschrittige Codes ist der *Gray-Code*.

Soll keine Zahl in einem Objektparameter codiert werden, bieten sich auch andere Codierungsmöglichkeiten an. Für das bekannte Travelling Salesman Problem (TSP) z. B. werden die Stationen, die der Handlungsreisende besuchen möchte, mit Nummern versehen und während des Optimierungsprozesses lediglich deren Reihenfolge dahingehend geändert, dass sich die kürzeste Reiseroute ergibt. Die einzelnen Objektparameter x_j haben hier demnach nur ein Gen, deren Allel eine Zahl aus der Menge der natürlichen

[1] Reellwertige GAs verwenden im Gegensatz zum kanonischen GA reelle Objektparameter und spezielle Rekombinationsoperatoren. Auf diese Variante des GA wird in diesem Artikel aber nicht weiter eingegangen.

[2] **Genotyp** [griech.] (Genotypus), die Summe der genetischen Informationen eines Organismus. [28]

[3] **Phänotyp** [griech.] (Phänotypus, Erscheinungsbild), in der *Genetik* die Gesamtheit aller äußeren und inneren Strukturen und Funktionen, d. h. aller Merkmale eines Lebewesens, als das Ergebnis aus dem Zusammenwirken von Genotyp und Umwelt. [28]

[4] Die Hamming-Distanz ist ein Maß dafür, in wie vielen Stellen sich zwei Bitketten unterscheiden.

Zahlen $\mathbb{N}$ ist. Bei dem in Abschnitt 3 vorgestellten Optimierungsproblem hingegen sind in den einzelnen Objektparamtern Ja-/Nein-Entscheidungen binär codiert, wobei eine 1 ein *Ja* bedeutet und eine 0 ein *Nein*. Jeder Objektparameter besteht somit aus nur einem binären Gen.

2.2 Selektion

Der GA erzeugt aus einer Population $P(t)$ mit n Individuen eine neue Population $P(t+1)$. Hierfür werden aus jeweils zwei Individuen, den *Eltern*, zwei neue Individuen, die *Kinder*, erzeugt. Damit der GA zu einem Optimum konvergiert, werden fittere Individuen bevorzugt, ganz nach dem Grundsatz der biologischen Evolution *Survival of the Fittest*[5]. Für die Erzeugung einer neuen Generation werden zuerst mithilfe der Selektion zwei Individuen abhängig von ihrer Fitness ausgewählt und dann reproduziert. Diese Schritte werden $n/2$-mal durchgeführt, um n neue Individuen für die neue Population zu erzeugen. Dabei gibt die Selektion dem GA eine Richtung im Suchraum vor, indem gute Lösungen konserviert werden. Die dadurch abnehmende Diversität der Population wird durch die Reproduktion (Varianz) der Individuen kompensiert.

Die Selektion wird in zwei Schritten durchgeführt. Im ersten Schritt wird der fitnessabhängige Erwartungswert $E(\mathbf{a}_i)$ festgelegt, also die erwartete Anzahl der Nachkommen eines Individuums in der neuen Population. Erst im zweiten Schritt erfolgt dann die konkrete Auswahl der Individuen für die Reproduktion. So kann ein Individuum auch mehrfach ausgewählt werden. Ein wichtiger Parameter bei der Selektion ist der *Selektionsdruck*. Er beschreibt das Verhältnis der Wahrscheinlichkeit der Auswahl des besten Individuums einer Population zur durchschnittlichen Selektionswahrscheinlichkeit aller Individuen. Ist dieser Druck klein, so hat die Suche tendenziell globalen Charakter, da auch schlechte Individuen eine Chance haben, ausgewählt zu werden. Bei steigendem Druck steigt die Wahrscheinlichkeit, ein besseres Individuum auszuwählen und somit wird diejenige Region bei der Suche bevorzugt, in der sich die guten Individuen befinden; die Suche wird somit lokal.

Die richtige Auswahl der Selektionsmethode spielt für die Konvergenz der Suche mit einem GA eine entscheidende Rolle, die aber oftmals unterschätzt und vernachlässigt wird. Aus diesem Grund wird die Selektion hier ausführlich behandelt und die drei gebräuchlichsten Arten von Auswahlverfahren vorgestellt: fitnessproportionale Selektion, rangbasierte Selektion und Wettkampfselektion.

2.2.1 Fitnessproportionale Selektion

Bei den fitnessproportionalen Selektionsmethoden ist die Selektionswahrscheinlichkeit proportional zur Fitness eines Individuums. Die Fitness muss daher nichtnegativ sein und als Evolutionsziel ist nur Maximierung möglich. Die populärste und in jedem Standardwerk über GA enthaltene Realisierung ist die *Roulette-Selektion*. Bei diesem Verfahren wird im ersten Schritt des Selektionsprozesses jedem Individuum ein Segment auf dem Roulette-Rad zugewiesen. Die Größe des Segmentes ist dabei proportional zur Fitness des Individuums. Im zweiten Schritt der Selektion wird dieses Rad gedreht und dadurch ein Segment zufällig bestimmt, dessen entsprechendes Individuum für die Reproduktion kopiert wird. Das Drehen am Roulette-Rad wird für die Erzeugung einer neuen Population insgesamt n-mal durchgeführt. Je besser die Fitness eines Individuums und damit je größer das Segment auf dem Rad ist, umso höher ist die Wahrscheinlichkeit, dass mehr Nachkommen dieses Individuums sich in der neuen Population befinden. Der Nachteil dieses Verfahrens sowie der anderen fitnessbasierten Selektionsmethoden ist, dass die Anzahl von Nachkommen eines Individuums sich oftmals sehr stark von dem Erwartungswert $E(\mathbf{a}_i) = n \cdot p_s(\mathbf{a}_i)$ unterscheidet. Bei sehr ungünstigen Bedingungen kann sogar der Extremfall eintreten, dass sich n Nachkommen des *schlechtesten* Individuums in der neuen Population befinden. Ein generelleres Problem ist weiterhin, dass ein Individuum mit einer sehr guten Fitness die Selektion beherrschen kann und somit die Suche in diesem lokalen Optimum hängen bleibt.

Eine verbesserte Variante, die einige dieser Probleme umgeht, ist das *stochastic universal sampling* (SUS). Den Individuen werden die gleichen Segmente wie bei der *Roulette-Selektion* zugewiesen, aber anstatt das Rad n-mal zu drehen, wird es nur einmal mit stochastischer Intensität gedreht. Durch n gleichmäßig um das Rad angeordneten Zeiger werden dann n Individuen in einem Schritt gezogen, wodurch der *spread* deutlich verringert werden kann.

Allen fitnessproportionalen Selektionsmethoden gemein ist, dass sie am Anfang der Suche sehr schnell konvergieren und nach einigen Generationen die Population sehr viele Individuen mit einer guten Fitness enthält. Die Suche wird dadurch verlangsamt und läuft Gefahr, durch die anfangs

[5] Es ist allerdings umstritten, ob die Natur wirklich so vorgeht. Der Evolutionsforscher Charles Darwin, dem man diesen Ausspruch nachsagt, hat dieses so nicht formuliert. Die technische Umsetzung des natürlichen Evolutions-Prozesses in den Evolutionären Algorithmen folgt jedenfalls diesem Grundsatz. Dieses liegt an der wesentlich vereinfachten Vorgehensweise der Umsetzung. Ziel ist es auch nicht, die Evolution nachzubilden, sondern deren Formalismen für die Suche nach dem Optimum einzusetzen.

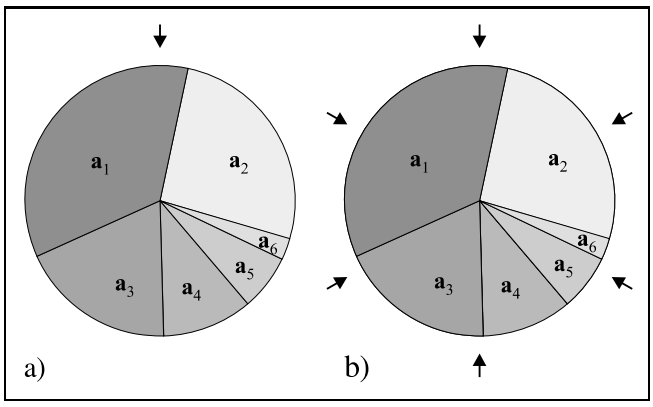

Bild 2: a) Roulette wheel selection b) Stochastic universal sampling.

schnelle Konvergenz in einem lokalen Optimum zu stagnieren. Aufgrund dieser Nachteile und dem Umstand, dass die fitnessproportionale Selektion häufig einige Anpassungen an das Optimierungsproblem erfordert, haben sich in letzter Zeit die nachfolgenden Selektionsmethoden immer mehr durchgesetzt.

2.2.2 Rangbasierte Selektion

Um die schnelle Konvergenz am Anfang der Suche zu verhindern, wurde die rangbasierte Selektion eingeführt. Anstelle der absoluten Fitnesswerte werden jetzt die Individuen abhängig von ihrer Fitness sortiert und nur noch aufgrund ihres resultierenden Ranges bewertet. So wird verhindert, dass die Suche von einigen wenigen Individuen mit hoher Fitness dominiert wird und die Suche zu schnell (in einem *lokalen* Optimum) konvergiert. Die globale Suche wird somit länger ausgeführt und dadurch die Wahrscheinlichkeit erhöht, in die Nähe des *globalen* Optimums zu gelangen. Sind nach einer gewissen Anzahl an Generationen wiederum viele Individuen mit guter Fitness in der Population enthalten, so hält diese Selektionsmethode den Selektionsdruck trotzdem noch auf einem hohen Niveau, um ein Stagnieren der Suche zu verhindern, da es keinen Unterschied macht, ob die beiden Individuen zweier benachbarter Ränge eine große oder eine kleine Differenz in ihren Fitnesswerten aufweisen. Um nun die Anzahl der Nachkommen eines jeden Individuums in der neuen Population zu bestimmen, werden die Individuen einer Population ihrer Fitness nach sortiert von Rang 1 mit dem schlechtesten Individuum bis Rang n mit der besten Lösung. Individuen hohen Rangs werden nun nach einer Zufallsauswahl mit hoher Wahrscheinlichkeit ausgewählt, schlechtere Individuen mit niedrigem Rang mit niedrigerer Wahrscheinlichkeit. Zur Festlegung der Wahrscheinlichkeit schlägt Baker in [12] folgendes Vorgehen vor: Der ranghöchsten Lösung wird der Erwartungswert E_{max} zugeordnet, wobei $1 \leq E_{max} \leq 2$ sein muss. Der Erwartungswert E_{min} für das schlechteste Individuum ergibt sich dann aus $E_{min} = 2 - E_{max}$ und der Erwartungswert eines beliebigem Individuums $\mathbf{a}_i$ berechnet sich zu

$$E(\mathbf{a}_i) = E_{min} + (E_{max} - E_{min}) \cdot \frac{r(\mathbf{a}_i) - 1}{n - 1},$$

wobei $r(\mathbf{a}_i)$ der Rang des Individuums $\mathbf{a}_i$ ist. Die Selektionswahrscheinlichkeit ergibt sich dann aus $p_s(\mathbf{a}_i) = 1/n \cdot E(\mathbf{a}_i)$. Baker hat den Wert 1,1 für E_{max} vorgeschlagen und festgestellt, dass die rangbasierte Selektion bei den vom ihm untersuchten Testproblemen vorteilhaft ist. Der eigentliche Auswahlprozess erfolgt dann mithilfe von *stochastic universal sampling*, nur dass jetzt für die Bestimmung der Segmentgrößen anstatt der absoluten Fitnesswerte die berechneten Selektionswahrscheinlichkeiten $p_s(\mathbf{a}_i)$ verwendet werden. Es bleibt aber festzuhalten, dass der reduzierte Selektionsdruck am Anfang den Suchprozess verlängern kann. Oftmals führt dies aber durch die Erhaltung der Vielfalt in der Population zu einem besseren Suchergebnis als bei der fitnessproportionalen Selektion.

2.2.3 Wettkampfselektion

In Bezug auf den Selektionsdruck ist die Wettkampfselektion vergleichbar mit der rangbasierten Selektion. Hier kann aber auf eine Berechnung der Erwartungswerte für jedes einzelne Individuum wie bei der rangbasierten Selektion verzichtet werden und es ist auch keine Sortierung der Individuen wie bei der fitnessproportionalen Selektion notwendig, sodass Rechenzeit eingespart werden kann.

In der Literatur findet man meist zwei Varianten. Bei der ersten werden zwei Individuen aus der Population zufällig mit gleicher Wahrscheinlichkeit ausgewählt. Eine Zufallszahl $r \in [0, 1]$ wird bestimmt und wenn $r < k$ gilt (k ist ein fester Parameter, z. B. 0,75) wird das fittere Individuum ausgewählt, andernfalls das schlechtere. Eine zweite, ähnliche Vorgehensweise verzichtet auf die Generierung der Zufallszahl r, sodass grundsätzlich das fittere Individuum bevorzugt wird. Der Selektionsdruck kann bei dieser Variante verändert werden, indem m Individuen ($2 \leq m < n$) ausgewählt werden. Ist $m = 2$ spricht man auch von einer *binären Wettkampfselektion*. Durch eine Erhöhung der Anzahl m bei konstanter Populationsgröße n kann, auch während des Suchprozesses, der Selektionsdruck erhöht und somit von einer globalen zu einer lokalen Suche gewechselt werden. Die Wettkampfselektion eignet sich, wie auch die rangbasierte Selektion, bei Auftreten negativer Fitnesswerte oder für Minimierungsaufgaben, ohne weitere Anpassungen vornehmen zu müssen. Ein Nachteil dieses Verfahrens ist, dass sich, gerade bei hohem Selektionsdruck, nach der Selektion n Kopien ein und desselben Individuums in der neuen Population befinden können. Trotzdem setzt sich die Wettkampfselektion aufgrund ihrer Vorteile gegenüber anderen Verfahren immer mehr durch [7].

2.2.4 Elitismus

Die oben aufgeführten Selektionsverfahren haben alle die Eigenschaft, die komplette Elternpopulation durch die neu erzeugten Nachkommen zu ersetzen (*generational replacement*). Dadurch kann es passieren, dass das bis dahin beste gefundene Individuum verloren geht. Beim Elitismus werden daher die schlechtesten Nachkommen durch die besten Eltern ersetzt. Jedes Selektionsverfahren kann in dieser Weise erweitert werden. Wird auf Elitismus verzichtet, so empfiehlt es sich, die beste Lösung außerhalb des GAs zu speichern und sie erst beim Finden eines besseren Individuums mit diesem zu überschreiben.

2.3 Variation

Um aus den selektierten Elternindividuen Nachkommen zu erzeugen, werden Variationsoperatoren, oftmals auch als Suchoperatoren bezeichnet, verwendet. Diese Operatoren erhalten die Vielfalt der Population und wirken somit dem durch die Selektion erzeugten Diversitätsverlust entgegen. Als Variationsoperatoren werden *Rekombination* und *Mutation* verwendet.

2.3.1 Rekombination

Die Rekombination ist der wichtigste Suchoperator. Er erzeugt durch Mischen der Erbinformationen zweier Elternindividuen zwei neue Nachkommen. In einem GA wird das *Crossover* als Form der Rekombination verwendet. Dessen Varianten sind das *n*-Punkt-Crossover, uniformes Crossover und problemabhängiges Crossover. Als Beispiel sei hier das 1-Punkt-Crossover [3] gezeigt:

$\mathbf{elter}_1 = [\mathbf{1\,1\,1\,1\,1\,1} \,|\, \mathbf{1\,1\,1\,1}]$

$\mathbf{elter}_2 = [0\,0\,0\,0\,0\,0 \,|\, 0\,0\,0\,0]$

$\mathbf{kind}_1 = [\mathbf{1\,1\,1\,1\,1\,1} \,|\, 0\,0\,0\,0]$

$\mathbf{kind}_2 = [0\,0\,0\,0\,0\,0 \,|\, \mathbf{1\,1\,1\,1}]$

Beim Crossover treten verschiedene Effekte auf, die die Bildung neuer Individuen beeinflussen. Bei dem oben gezeigten 1-Punkt-Crossover z. B. steigt die Wahrscheinlichkeit, dass ein Gen ausgetauscht wird, mit seiner Position auf dem Genom an. Dieser Effekt wird *positional bias* genannt. Er verhindert das gemeinsame Vererben von Genen, die weit entfernt auf dem Genom liegen. Zusammen mit der Eigenheit, dass immer der Endpunkt eines Individuums mit getauscht wird, können einige Regionen im Suchraum nicht oder nur schwer erreicht werden, während andere Regionen bevorzugt werden. Die dennoch hohe Effektivität des Crossover ist darauf zurückzuführen, dass Teile von Individuen kombiniert werden und somit immer bessere Nachkommen erzeugt werden. Das Crossover kann als eine *Makromutation* verstanden werden und ermöglicht große Sprünge im Suchraum.

Ob es aber überhaupt zum Crossover zwischen zwei Elternindividuen kommt, entscheidet sich durch die *Crossover-Rate* p_C, die typischerweise größer als 0,6 gewählt wird [7]. Wird kein Crossover durchgeführt, werden die selektierten Elternindividuen unverändert in die neue Population übernommen.

n-Punkt-Crossover

Um die Problematiken des *positional bias* und des „Endpunkt-Effektes" zu umgehen, werden *n* Schnittstellen beim *n-Punkt-Crossover* verwendet, bei dem die Elternindividuen nicht nur an einer, sondern an *n* Stellen zerschnitten und die Sektoren mit einer geraden Nummer ausgetauscht werden [7]. Als Beispiel ist hier das oftmals verwendete 2-Punkt-Crossover gezeigt:

$\mathbf{elter}_1 = [\mathbf{1\,1\,1\,1} \,|\, \mathbf{1\,1\,1} \,|\, \mathbf{1\,1\,1}]$

$\mathbf{elter}_2 = [0\,0\,0\,0 \,|\, 0\,0\,0 \,|\, 0\,0\,0]$

$\mathbf{kind}_1 = [\mathbf{1\,1\,1\,1} \,|\, 0\,0\,0 \,|\, \mathbf{1\,1\,1}]$

$\mathbf{kind}_2 = [0\,0\,0\,0 \,|\, \mathbf{1\,1\,1} \,|\, 0\,0\,0]$

Die Anzahl *n* der Crossover-Stellen ist meist eine gerade Zahl, wobei der *positional bias* mit steigendem *n* sinkt.

Uniform Crossover

Eine weitere Form des Crossover ist das *Uniform Crossover*, bei dem für jedes einzelne Bit entschieden wird, ob es zwischen den Eltern ausgetauscht wird. Die Wahrscheinlichkeit p_{UX} für den Austausch liegt dabei meist zwischen 0,5 und 0,8 [9]. Der *positional bias* ist hier nicht mehr vorhanden, sodass bei binärer Codierung jede mögliche Kombination aus beliebigen Elternindividuen erzeugt werden kann.

$\mathbf{elter}_1 = [\mathbf{1\,1\,1\,1\,1\,1\,1\,1\,1\,1}]$

$\mathbf{elter}_2 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0]$

Austausch $[n\,n\,j\,n\,j\,j\,n\,j\,n\,n]$

$\mathbf{kind}_1 = [\mathbf{1\,1}\,0\,\mathbf{1}\,0\,0\,\mathbf{1}\,0\,\mathbf{1\,1}]$

$\mathbf{kind}_2 = [0\,0\,\mathbf{1}\,0\,\mathbf{1\,1}\,0\,\mathbf{1}\,0\,0]$

Darüber hinaus gibt es noch eine Reihe weiterer Crossover-Varianten, die oftmals bei der Lösung speziell codierter Probleme entstanden sind, und es kommen immer wieder neue hinzu. Die Frage nach einem allgemein am besten geeigneten Verfahren kann deshalb nicht beantwortet werden. So bleibt es dem Anwender überlassen, auch diesen Teil seines GA dem zu lösenden Problem anzupassen, falls der Basis-GA nicht zum Ziel führt.

2.3.2 Mutation

Wenn das Crossover der Suchoperator ist, um große Sprünge im Suchraum zu machen, so ist die Mutation eine Veränderung im Kleinen. Darüber, ob die Mutation unwichtiger oder wichtiger als das Crossover ist, herrscht allerdings keine Einigkeit. M. Mitchell drückt es in [9] aber sehr treffend aus, indem sie sagt, dass es keine *Wahl* zwischen den beiden Funktionen ist, sondern darauf ankommt, die richtige *Balance* zwischen den beiden zu finden.

Bei der Mutation wird jedes Gen eines Genoms mit einer geringen Wahrscheinlichkeit p_M negiert:

Vor der Mutation:

$\mathbf{kind}_1 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0]$

Nach der Mutation von Gen 3 und Gen 8:

$\mathbf{kind}_1 = [0\,0\,\mathbf{1}\,0\,0\,0\,0\,\mathbf{1}\,0\,0]$

Für die damit *genbezogene* Mutationswahrscheinlichkeit p_M werden meist Werte von 0,01 oder 0,001 gewählt, da die Wichtigkeit der Mutation für den Erfolg der Suche oftmals als gering angesehen wird. Es gibt aber auch Stimmen, die der Mutation eine wichtigere Rolle zubilligen. So schlägt Mühlenbein in [16] eine von der Individuumlänge L abhängige *optimale Mutationswahrscheinlichkeit* von $p_{M_{opt}} = 1/L$ vor.

2.4 Wann sollen Genetische Algorithmen eingesetzt werden?

Wie eingangs schon angedeutet, haben heuristische Verfahren Stärken bei nicht-glatten Fitnessfunktionen. Speziell die

Genetischen Algorithmen vereinen gute lokale Sucheigenschaften mit der Möglichkeit größerer Sprünge im Suchraum; sie arbeiten daher häufig noch in großen Suchräumen erfolgreich, selbst wenn deren genaue Eigenschaften unbekannt sind. Auch bei Suchprozessen, bei denen mit verrauschten Messwerten gearbeitet werden muss, sind GAs Optimierungsverfahren überlegen, die nicht parallel mehrere Lösungen verarbeiten, wie z. B. *Tabu Search* oder *Simulated Annealing* [15]. Zwar gibt es Ansätze, Optimierungsprobleme hinsichtlich ihrer Lösbarkeit mittels GA formal zu klassifizieren, diese sind aber weder eindeutig noch unumstritten.

Erfolgreiche Implementierungen Genetischer Algorithmen liegen heute in großer Zahl vor, und eine zunehmende Zahl von Veröffentlichungen und Konferenzen zeigt den Einsatz in fachübergreifenden Anwendungen: So werden GAs für die Optimierung Neuronaler Netze eingesetzt [15], aber auch in der Automatisierungstechnik, wie z. B. bei der Optimierung von Fuzzy-Reglern [15], von H_2-Reglern [14] oder von prädiktiven Reglern [17]. Daneben werden Benchmark-Probleme bearbeitet, wie das Travelling-Salesman-Problem (TSP), das in fast jedem Grundlagenbuch erwähnt wird.

3 Anwendung: Modellreduktion mittels GA

Modelle komplexer dynamischer Systeme umfassen häufig große Zahlen nichtlinearer Differentialgleichungen, was Simulation, Analyse und Systementwurf erschwert. Ordnungsreduktionsverfahren können zwar die Zahl systembeschreibender Gleichungen verringern, liefern in der Regel aber reduzierte Modelle mit gesteigerter Zahl innerer Kopplungen, d. h. strukturell komplexere Modelle. Abhilfe schaffen hier Genetische Algorithmen, mit deren Hilfe geeignete einfache Modellstrukturen gefunden werden können.

3.1 Ordnungsreduktion

Betrachtet werden nichtlineare, zeitinvariante Systeme in der Zustandsdarstellung

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right), \tag{2}$$

wobei $\mathbf{x}(t)$ den n-dimensionalen Zustandsvektor und $\mathbf{u}(t)$ den p-dimensionalen Stellgrößenvektor bezeichnen; n ist die Systemordnung. Einem Vorschlag von B. Lohmann [20; 21] folgend kann die rechte Seite der Modellgleichung ohne Einschränkung der Allgemeinheit in zwei lineare Summanden und einen verbleibenden nichtlinearen Anteil aufgespalten werden gemäß

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{F}\mathbf{g}(\mathbf{x}, \mathbf{u}), \tag{3}$$

wobei jedes Element des Vektors $\mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right)$ aufgebrochen wird in die Zeitfunktionen $\mathbf{x}(t)$, $\mathbf{u}(t)$ und $\mathbf{g}(\mathbf{x}, \mathbf{u})$ und in die zugehörigen konstanten Koeffizienten, zusammengefasst in den Matrizen $\mathbf{A}$, $\mathbf{B}$ und $\mathbf{F}$. Ziel der Ordnungsreduktion ist die Gewinnung eines Modells niedrigerer Ordnung $\tilde{n}$

$$\dot{\tilde{\mathbf{x}}}(t) = \tilde{\mathbf{A}}\tilde{\mathbf{x}}(t) + \tilde{\mathbf{B}}\mathbf{u}(t) + \tilde{\mathbf{F}}\mathbf{g}(\mathbf{W}\tilde{\mathbf{x}}, \mathbf{u}), \tag{4}$$

das das Verhalten des Originals approximiert, d. h. die wesentlichen oder dominanten Zustandsgrößen gut nachbildet. Diese dominanten Zustände werden vom Nutzer vorgegeben und hängen über eine Reduktionsmatrix $\mathbf{R}$ mit dem Originalzustandsvektor zusammen gemäß

$$\mathbf{x}_{do} = \mathbf{R} \cdot \mathbf{x}(t). \tag{5}$$

Liegt das reduzierte Modell (4) vor, so kann außerdem eine Näherung $\hat{\mathbf{x}}(t)$ des Zustandsvektors $\mathbf{x}(t)$ aus dem Vektor $\tilde{\mathbf{x}}(t)$ ermittelt werden:

$$\hat{\mathbf{x}} = \mathbf{W} \cdot \tilde{\mathbf{x}}. \tag{6}$$

Die Berechnung der Matrizen $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{F}}$ und $\mathbf{W}$ kann gemäß dem in [20; 21] ausführlich dargestellten Reduktionsverfahren auf Basis von Simulationsdaten des Originalmodells nach quadratischen Optimalitätskriterien über geschlossene Formeln erfolgen.

3.2 Nachteil der Ordnungsreduktion

Es hat sich gezeigt, dass die berechneten Systemmatrizen des ordnungsreduzierten Systems $\mathbf{E} = [\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{F}}]$ und die Matrix $\mathbf{W}$ in der Regel voll besetzt sind. Dies führt zu einer hohen inneren Systemkomplexität. Um die Ordnung *und* die Komplexität gleichzeitig zu reduzieren, kann die Tatsache genutzt werden, dass das genannte Verfahren die Berücksichtigung von Nebenbedingungen des Typs

$$\mathbf{0}^T = \mathbf{e}_i^T \mathbf{H}_{Ei}, \quad i = 1 \ldots \tilde{n}$$
$$\mathbf{0}^T = \mathbf{w}_k^T \mathbf{H}_{Wk}, \quad k = 1 \ldots n$$

an die Zeilen $\mathbf{e}_i^T$ von $\mathbf{E}$ und $\mathbf{w}_k^T$ von $\mathbf{W}$ erlaubt. Wählt man dabei die Matrizen $\mathbf{H}$ so, dass in jeder Spalte genau eine Eins und ansonsten Nullen auftreten, so können gezielt einzelne Elemente von $\mathbf{E}$ bzw. $\mathbf{W}$ zu Null gemacht werden; wählt man beispielsweise $\mathbf{H}_{E3} = [0, 1, 0, \ldots, 0]^T$, so wird das zweite Element der dritten Zeile von $\mathbf{E}$ zu Null erzwungen, während alle verbleibenden Elemente der dritten Zeile im Sinne des Reduktionsverfahrens optimiert werden. Sollen mehrere Elemente der dritten Zeile zu Null gemacht werden, so weist $\mathbf{H}_{E3}$ entsprechend viele Einsen auf.

Aufgrund der hohen Anzahl an Möglichkeiten, die Nebenbedingungen aufzustellen, wird ein Genetischer Algorithmus eingesetzt, um ein Optimum bezüglich einer einfachen Modellstruktur und einer guten Nachbildung des Originalsystems zu finden [22]. Die Berechnung des reduzierten Systems erfolgt somit in zwei Schritten:

1. Aufstellen der Nebenbedingungen mithilfe des GA.
2. Berechnung der Matrizen $\mathbf{E} = [\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{F}}]$ und $\mathbf{W}$ mit der herkömmlichen Methode der Ordnungsreduktion unter Berücksichtigung der Nebenbedingungen aus 1.

3.3 Individuum

Um eine mögliche Nebenbedingung in einem Individuum zu codieren, benötigt man für jedes Element der Matrix **E** die Information, ob dieses zu Null gesetzt werden soll, oder nicht. Somit hat ein Individuum die gleiche Anzahl an Objektparameter, wie **E** Elemente, wobei die Objektparameter alle die Länge 1 haben und den Wert 0 oder 1 annehmen können. Eine 0 bedeutet dabei *nein*, es wird keine Null in **E** eingefügt und eine 1 *ja*, es wird eine Null eingefügt. Ein Beispiel: Die Matrix **E** soll folgendes Aussehen haben:

$$\mathbf{E} = \begin{bmatrix} 0 & x & 0 \\ x & 0 & x \end{bmatrix} \quad \text{mit } x: \text{ Nicht-Nullelement}$$

Das Genom des entsprechenden Individuums lautet dann:

$$\mathbf{a}_E = [\underbrace{1\,0\,1}_{\text{1. Zeile von } \mathbf{E}}\ \underbrace{0\,1\,0}_{\text{2. Zeile von } \mathbf{E}}]$$

Um die Fitness eines Individuums zu ermitteln, wird unter Berücksichtigung der entsprechenden Nebenbedingungen das dazugehörige ordnungsreduzierte System berechnet und dessen Zustandsverläufe mit denen des Originalsystems für verschiedene Eingangsanregungen verglichen:

$$F_{E1} = \underbrace{k_1 \cdot \sum_{i=1}^{n} \frac{\int_0^\infty (x_i(t) - \mathbf{w}_i\,\tilde{x}_i(t))^2 dt}{\int_0^\infty x_i^2(t)}}_{\text{Modellnachbildung}} - \underbrace{k_2 \cdot sum(\mathbf{a}_E)}_{\substack{\text{Modell-}\\ \text{komplexität}}} \stackrel{!}{=} min\,, \tag{7}$$

worin $\mathbf{w}_i$ die i-te Zeile von **W** und $sum(\mathbf{a}_E)$ die Zahl der Einsen im Vektor $\mathbf{a}_E$ bezeichnet und damit ein Maß für die Modellkomplexität ist. Ergebnis der Fitnessberechnung ist eine reelle Zahl als Fitnesswert, wobei ein kleiner Wert eine gute Nachbildung und eine geringe Systemkomplexität bedeuten. Es handelt sich demnach um ein Minimierungsproblem.

Ein Nachteil dieser Fitnessberechnung ist, dass sie die rechen- und damit zeitintensive Lösung der Differentialgleichungen erfordert. Daher wird hier eine alternative Fitnessberechnung vorgeschlagen, die auf die Simulation des reduzierten Modells verzichtet und stattdessen auf die bereits berechneten Zustandsverläufe der dominanten Zustände $\mathbf{x}_{do}$ des Originalsystems zurückgreift:

$$F_{E2} = \underbrace{k_1 \cdot \sum_{i=1}^{n} \frac{\int_0^\infty \left[\dot{x}_{i,do}(t) - \mathbf{e}_i^T \begin{bmatrix} \mathbf{x}_{do}(t) \\ \mathbf{u}(t) \\ \mathbf{g}(\mathbf{W}\mathbf{x}_{do}(t), \mathbf{u}(t)) \end{bmatrix} \right]^2 dt}{\int_0^\infty \dot{\mathbf{x}}_{do}^2(t)\,dt}}_{\text{Modellnachbildung}} - \underbrace{k_2 \cdot sum(\mathbf{a}_E)}_{\substack{\text{Modell-}\\ \text{komplexität}}} \stackrel{!}{=} min\,, \tag{8}$$

worin $\mathbf{e}_i^T$ die i-te Zeile von **E** bezeichnet. Im Unterschied zu (7) bewertet diese Fitnessfunktion nicht die tatsächlich im reduzierten Modell sich einstellenden Zeitverläufe, sondern lediglich die erwarteten Fehler in den Zeitableitungen. Dennoch wird man das Gütemaß (8) aufgrund der vielfach kleineren Rechenzeit in der Regel vorziehen.

Zu erwähnen ist, dass es sich hier um eine Mehrzieloptimierung mit zwei Zielen, exakter Modellnachbildung und niedriger Modellkomplexität, handelt. Um diese Ziele in einer einzigen Fitnessfunktion zusammenzufassen, wird der Aggregationsansatz verwendet: Die einzelnen Zielfunktionen werden mit k_1 und k_2 gewichtet und dann subtrahiert. Somit kann dieses Problem wie ein Einzelproblem behandelt werden, und der GA berechnet Lösungen, die einen Kompromiss hinsichtlich der Einzelziele darstellen.

3.4 Beispiel mit technischem Hintergrund

Bild 3 zeigt schematisch den Aufbau einer aktiven hydropneumatischen Kraftfahrzeugfederung. Das Originalmodell hat die Ordnung $n = 10$, die Systemmatrizen **A**, **B** und **F** lauten:

$$\mathbf{A} = \begin{bmatrix} 0 & 1{,}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4127 & 0 & 127{,}5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1{,}0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12{,}75 & 0 & -1613 & 0 & 1600 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1{,}0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 533\,333 & 0 & -533\,333 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1{,}244 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -162 & 1{,}0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8100 & 0 \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 4000 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 8100 \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -0{,}0148 & -0{,}0148 & -0{,}0148 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0{,}000518 & -0{,}000518 & -0{,}000518 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0{,}667 & 0{,}667 & 0{,}667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 17{,}5 & -71{,}43 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Wählt man aus technischen Erwägungen [20] sieben dominante Zustandsvariablen und setzt $\tilde{n} = 7$, so führt eine Ordnungsreduktion *ohne* Strukturvereinfachung auf folgende Matrizen:

$$\tilde{\mathbf{A}} = \begin{bmatrix} 5{,}9\cdot10^{-13} & 1{,}0\cdot10^{0} & -3{,}8\cdot10^{-13} & 5{,}2\cdot10^{-14} & 6{,}7\cdot10^{-13} & 7{,}2\cdot10^{-14} & 2{,}5\cdot10^{-14} \\ -4{,}1\cdot10^{3} & -2{,}4\cdot10^{-11} & 1{,}3\cdot10^{2} & 1{,}2\cdot10^{-11} & -1{,}4\cdot10^{-9} & -7{,}9\cdot10^{-11} & 1{,}0\cdot10^{-11} \\ 3{,}1\cdot10^{-13} & -1{,}5\cdot10^{-13} & -8{,}5\cdot10^{-14} & 1{,}0\cdot10^{0} & 3{,}2\cdot10^{-13} & 6{,}9\cdot10^{-14} & 1{,}9\cdot10^{-14} \\ 1{,}3\cdot10^{1} & 7{,}6\cdot10^{-3} & -1{,}1\cdot10^{1} & -3{,}6\cdot10^{-2} & -1{,}3\cdot10^{0} & 8{,}8\cdot10^{-2} & -8{,}6\cdot10^{-2} \\ -4{,}9\cdot10^{-14} & 7{,}1\cdot10^{-15} & 2{,}9\cdot10^{-14} & 5{,}0\cdot10^{-15} & 4{,}9\cdot10^{-14} & 1{,}2\cdot10^{0} & -2{,}3\cdot10^{-15} \\ -2{,}3\cdot10^{-12} & -1{,}1\cdot10^{-14} & 1{,}9\cdot10^{-12} & 7{,}3\cdot10^{-14} & -2{,}5\cdot10^{-12} & 3{,}5\cdot10^{-13} & -7{,}8\cdot10^{-14} \\ -1{,}4\cdot10^{1} & 1{,}2\cdot10^{0} & 1{,}5\cdot10^{1} & -3{,}0\cdot10^{0} & -2{,}1\cdot10^{1} & -1{,}4\cdot10^{1} & -1{,}9\cdot10^{1} \end{bmatrix},$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} -2{,}3\cdot10^{-14} & 4{,}7\cdot10^{-15} \\ 4{,}0\cdot10^{3} & -4{,}7\cdot10^{-13} \\ 6{,}1\cdot10^{-15} & 4{,}7\cdot10^{-15} \\ -1{,}3\cdot10^{0} & 1{,}8\cdot10^{-2} \\ 9{,}7\cdot10^{-15} & 9{,}4\cdot10^{-15} \\ 3{,}2\cdot10^{-13} & -3{,}5\cdot10^{-4} \\ 8{,}0\cdot10^{-1} & 2{,}8\cdot10^{1} \end{bmatrix},$$

$$\tilde{\mathbf{F}} = \begin{bmatrix} 5{,}0\cdot10^{-17} & 4{,}1\cdot10^{-16} & 4{,}2\cdot10^{-17} & -8{,}4\cdot10^{-14} & -3{,}5\cdot10^{-14} \\ -1{,}5\cdot10^{-2} & -1{,}5\cdot10^{-2} & -1{,}5\cdot10^{-2} & -1{,}3\cdot10^{-11} & -8{,}4\cdot10^{-12} \\ 4{,}5\cdot10^{-17} & 3{,}6\cdot10^{-16} & 4{,}3\cdot10^{-17} & -7{,}3\cdot10^{-14} & -5{,}1\cdot10^{-14} \\ 1{,}6\cdot10^{-3} & 1{,}4\cdot10^{-3} & 1{,}5\cdot10^{-3} & 3{,}0\cdot10^{-2} & 2{,}0\cdot10^{-1} \\ 2{,}6\cdot10^{-18} & 5{,}3\cdot10^{-18} & -2{,}8\cdot10^{-18} & 6{,}7\cdot10^{-15} & 2{,}0\cdot10^{-14} \\ 8{,}8\cdot10^{-17} & -2{,}5\cdot10^{-16} & 1{,}8\cdot10^{-17} & 1{,}8\cdot10^{1} & -7{,}1\cdot10^{1} \\ 2{,}3\cdot10^{-4} & -2{,}1\cdot10^{-3} & -3{,}3\cdot10^{-4} & 3{,}2\cdot10^{0} & -6{,}9\cdot10^{1} \end{bmatrix}.$$

Alle Elemente der berechneten Matrizen sind ungleich null, und somit liegt eine sehr hohe Systemkomplexität vor[6].

Um diesen Nachteil zu beheben, werden mithilfe des Genetischen Algorithmus' geeignete Nebenbedingungen generiert, die bei der Berechnung der ordnungsreduzierten Systemmatrizen berücksichtigt werden. Der Suchraum dieses Optimierungsproblems umfasst eine Anzahl von $2^{\tilde{n}\cdot(\tilde{n}+p+q)} = 2^{7\cdot(7+2+5)} = 2^{98} \approx 3\cdot10^{29}$ mögliche Lösungen (Ordnung des reduzierten Systems: $\tilde{n} = 7$, Anzahl der Eingänge $p = 2$ und Anzahl der Nichtlinearitäten $q = 2$). Die Zahl 98 ist auch gleichzeitig die Anzahl der Elemente der Matrix $\mathbf{E} = [\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{F}}]$ und somit die Chromosomenlänge L, da jedem Element von $\mathbf{E}$ ein Gen eines Individuums zugeordnet wird.

[6] Durch nachträgliches Nullsetzen der kleinsten Elemente in diesen Matrizen vereinfacht sich die innere Struktur zwar, erreicht aber bei weitem nicht die Einfachheit und Nachbildungsgüte des mittels GA gefundenen Modells.

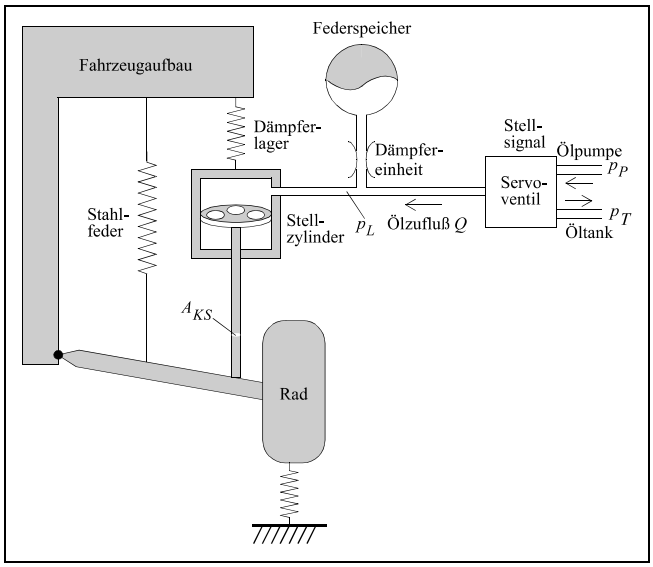

Bild 3: Aufbau der KFZ-Federung.

3.5 Entwurf des Genetischen Algorithmus'

Bei der Implementierung des Basis-GA nach Abschnitt 2 hat man sich zwischen verschiedenen Varianten der Selektion und Variation zu entscheiden. Eine allgemeingültige Empfehlung kann dabei kaum gegeben werden, einen guten Ausgangspunkt mag aber die rangbasierte Selektion mit $E_{max} = 1{,}1$ und Elitismus mit $q = 1$ zusammen mit dem 2-Punkt-Crossover mit einer Wahrscheinlichkeit von $p_C = 0{,}6$ und der bitbezogenen Mutation mit einer Wahrscheinlichkeit von $p_M = 1/L$ darstellen. Eine weitere wichtige Konstante ist die Populationsgröße n, die meist zwischen 20 und 500 (hier als Standard: $n = 40$) gewählt wird [7].

Variiert man diese Methoden und Werte innerhalb sinnvoller Intervalle, ergeben sich für das hier vorgestellte Optimierungsproblem z. B. die folgenden Varianten:

- *Populationsgröße:* $n = [20; 40; 50; 70; 100; 200; 400]$
- *Selektion:* Rangbasierte Selektion mit $E_{max} = [1{,}1; 1{,}3; 1{,}5; 1{,}7; 1{,}9]$, Wettkampfselektion[7] mit $m = [2; 4; 6; \ldots; n/2]$
- *Rekombination:* N-Punkt-Crossover mit $N = [1; 2; 3; 4]$, Uniform-Crossover, mit unterschiedlichen Wahrscheinlichkeiten $p_C = [0{,}2; 0{,}4; 0{,}6; 0{,}8]$
- *Mutation*: Bitbezogene Mutation mit $p_M = [0{,}001; 0{,}01; 1/L]$
- *Elitismus:* Übernahme der besten q Eltern: $q = [0; 1; 2; 3]$

Nachfolgend werden einige ausgewählte Ergebnisse der Suche mit Genetischen Algorithmen vorgestellt. Ausgehend von der oben beschriebenen Grundkonfiguration werden die Variablen einzeln verändert. Die Bilder 4 und 5 zeigen jeweils unten den Verlauf der Fitness des besten Individuums in der aktuellen Population über 100 Generationen und oben den Mittelwert der Fitnesswerte aller n Individuen einer Population.

Bei unterschiedlichen Populationsgrößen ist in Bild 4 zu erkennen, dass der Algorithmus für jede Populationsgröße n konvergiert und in die Region mit dem globalen Optimum vorstößt. Je größer n ist, umso höher ist auch die Wahrscheinlichkeit, ein besseres Individuum bei der Berechnung der Startpopulation zu finden, was in der unteren Abbildung zu erkennen ist. Die gerichtete Suche des GA, die durch den Selektions-Operator gewährleistet wird, ist auch in den Fitness-Durchschnittswerten der gesamten Population zu erkennen.

[7] Es wird die zweite vorgestellte Variante mit variablem Selektionsdruck verwendet.

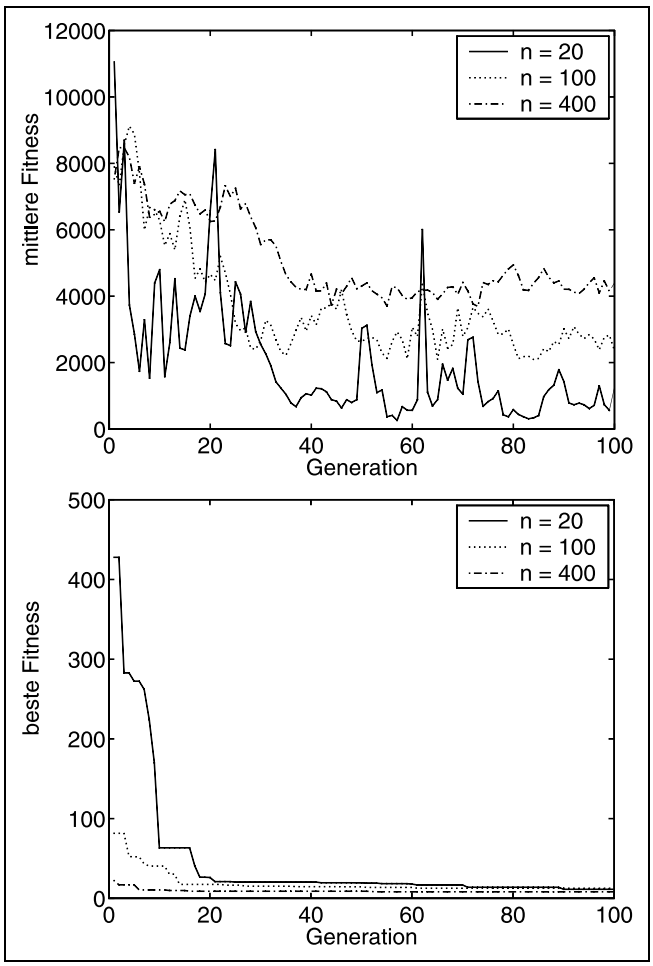

Bild 4: Auswirkungen der Populationsgröße *n*.

Bild 5 zeigt die Auswirkungen des Selektionsdrucks bei der rangbasierten Selektion durch Variation des Erwartungswertes E_{max} für das beste Individuum. Es ist deutlich zu erkennen, dass ein hoher Selektionsdruck bei großem E_{max} zu einer schnelleren Konvergenz des GAs führt. Bei $E_{max} = 1{,}1$ wird der gute Fitnesswert für das beste Individuum nicht ganz erreicht, anders bei größeren Werten. Auch die durchschnittliche Fitness aller Individuen ist bei kleinem E_{max} deutlich schlechter, da durch den geringen Selektionsdruck auch immer wieder schlechte Individuen ausgewählt werden.

Ein noch höherer Selektionsdruck als bei der rangbasierten Selektion kann bei der Wettkampfselektion erzielt werden. Bei dem geringstmöglichen Selektionsdruck im Wettkampf mit $m = 2$ ähnelt die Auswahl der Individuen bereits dem der rangbasierten Selektion mit einem hohen $E_{max} = 1{,}9$:

$$\text{selektierte Ind.} = \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 4 & 5 & 5 & 5 & 6 \\ 6 & 7 & 7 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 14 & 14 & 14 & 15 & 17 & 18 & 18 & 18 & 19 & 21 \\ 21 & 24 & 25 & 25 & 25 & 25 & 27 & 20 & 31 & 38 \end{bmatrix}.$$

Ein sehr hoher Selektionsdruck mit $m = 20$ bei einer Populationsgröße von $n = 40$ hingegen führt dazu, dass sich nur noch die besten 10 Individuen mit entsprechend vielen Nachkommen in der neuen Population durchsetzen konnten:

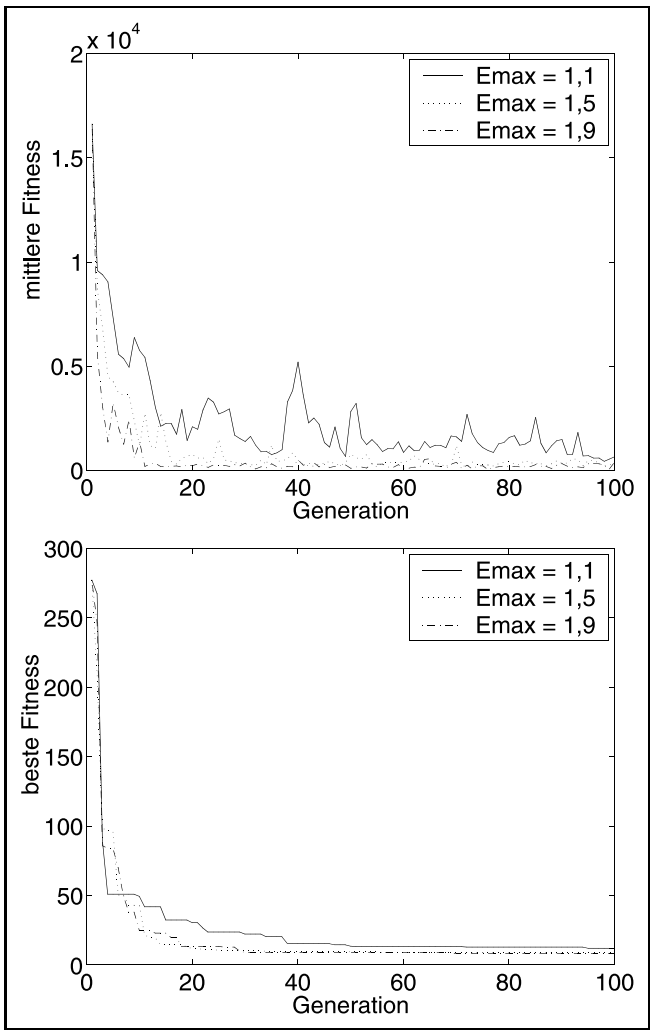

Bild 5: Auswirkungen des Erwartungswertes E_{max} bei rangbasierter Selektion.

$$\text{selektierte Ind.} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 4 & 4 & 5 & 8 & 9 & 10 \end{bmatrix}.$$

Änderungen der Variationsoperatoren *Rekombination* und *Mutation* beeinflussten die Suche *für dieses Optimierungsproblem* nicht signifikant, sodass auf eine Darstellung der Suchverläufe verzichtet wurde.

3.6 Ergebnis der Suche mit dem GA

Dem Genetischen Algorithmus gelingt es, Lösungen zu finden, die bei einer geringen Anzahl an Verkopplungen eine gute Approximation des Originalsystems bieten. Eine dieser Lösungen ist hier dargestellt:

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & 1{,}0 & 0 & 0 & 0 & 0 & 0 \\ -4127{,}5 & 0 & 127{,}5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1{,}0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -16{,}8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1{,}2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -31{,}6 \end{bmatrix},$$

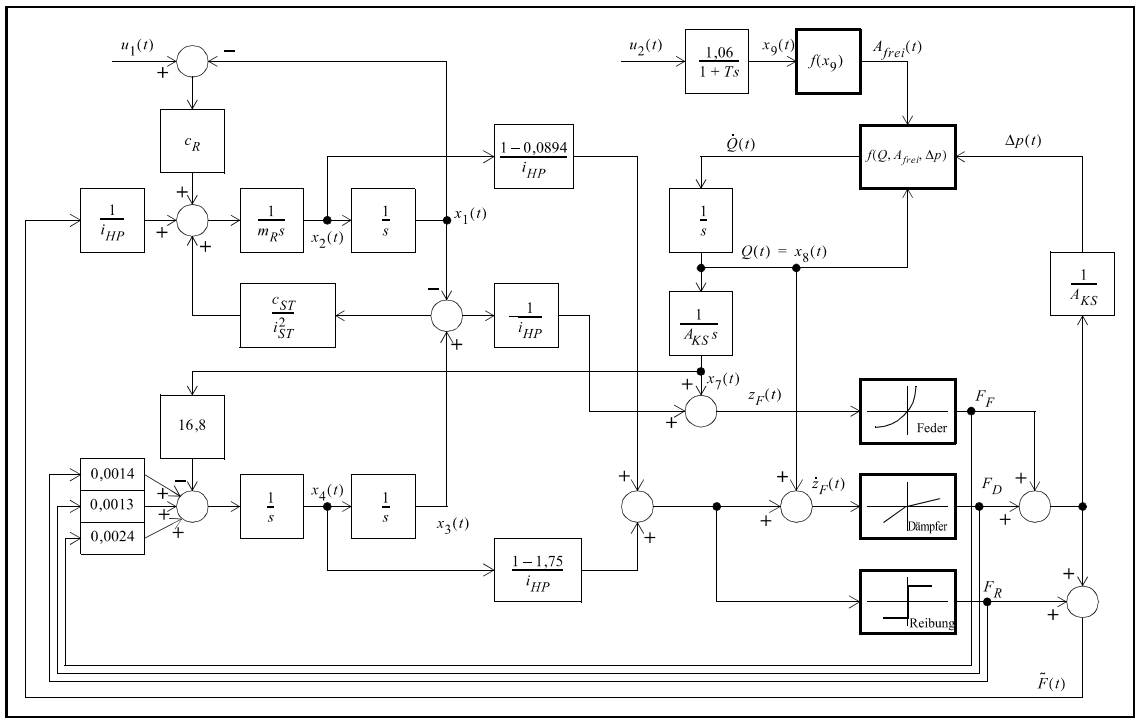

Bild 6: Blockschaltbild des reduzierten Systems.

$$\tilde{\mathbf{B}} = \begin{bmatrix} 0 & 0 \\ 4000 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 33{,}5 \end{bmatrix},$$

$$\tilde{\mathbf{F}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -0{,}015 & -0{,}015 & -0{,}015 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0{,}0024 & 0{,}0013 & 0{,}0015 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 17{,}5 & -71{,}4 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Zusammen mit der Matrix

$$\mathbf{W} = \begin{bmatrix} 1{,}0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1{,}0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1{,}0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1{,}0 & 0 & 0 & 0 \\ 0 & 0 & 1{,}0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}09 & 0 & 0{,}8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1{,}0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1{,}0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1{,}0 \\ 0 & 0 & 0 & 0 & 0 & -62{,}1 & 178{,}9 \end{bmatrix}$$

mit deren Hilfe auch die drei *nichtdominanten* Zustände des Originalsystems über die Beziehung $\hat{\mathbf{x}} = \mathbf{W} \cdot \tilde{\mathbf{x}}$ nachgebildet werden können, kann das Blockschaltbild des reduzierten Systems angegeben werden (Bild 6). In der Matrix **W** sind, da die physikalische Bedeutung der Zustände des Originalsystems und des reduzierten Systems gleich sind, in den Zeilen für die *dominanten* Zustände jeweils eine Eins und sonst Nullen und für die *nichtdominanten* Zustände Elemente ungleich null. Also muss auch hier wie für **E** eine Strukturvereinfachung vorgenommen werden, deren Ergebnis in der obigen Gleichung für **W** angegeben ist. Der Suchraum dieses Optimierungsproblems ist allerdings wesentlich kleiner mit $2^{(n-\tilde{n})\cdot\tilde{n}} = 2^{(10-7)\cdot 7} = 2^{21} \approx 2 \cdot 10^6$ Individuen (Ordnung des Originalsystems: $n = 10$, Ordnung des reduzierten Systems: $\tilde{n} = 7$). Die Berechnung und Optimierung von **W** und **E** erfolgte dabei nacheinander, und zwar zunächst **W** unter Verwendung der Fitnessfunktion

$$F_{W1} = \underbrace{k_1 \cdot \sum_{i=1}^{n} \frac{\int_0^\infty [x_i(t) - \mathbf{w}_i^T \mathbf{x}_{do}(t)]^2 dt}{\int_0^\infty x_i^2(t) dt}}_{\text{Modellnachbildung}} \underbrace{- k_2 \cdot sum(\mathbf{a}_W)}_{\substack{\text{Modell-} \\ \text{komplexität}}} \stackrel{!}{=} min,$$

worin $sum(\mathbf{a}_W)$ die Zahl der Nullelemente der Matrix **W** ist, dann **E** unter Verwendung von (8)[8].

3.7 Anmerkungen zum Optimierungsproblem

Für den Erfolg des GA beim vorliegenden Optimierungsproblem ist von entscheidender Bedeutung, dass es nicht

[8] Dieses Vorgehen führt zu suboptimalen Lösungen. Für das strenge Optimum sind **W** und **E** gleichzeitig unter Verwendung von (7) zu suchen.

nur eine einzige gute Lösung gibt, die aus der Menge von schlechten Lösungen hervorsticht, sondern dass es mindestens eine Region im Suchraum mit vielen guten Lösungen gibt. Die Individuen in einer Region unterscheiden sich dabei in ihrem Genom nur geringfügig und weisen auch eine ähnlich gute Fitness auf. Wäre dies nicht der Fall, hätten Heuristiken Schwierigkeiten, eine gute Lösung zu finden. Die Umgebung des besten gefundenen Individuums ist in Bild 7 dargestellt. Es zeigt, dass es tatsächlich nicht nur ein einziges gutes Individuum gibt, sondern mehrere Individuen mit guter Fitness und ähnlichem Genotyp [23; 24].

Um das gezeigte ordnungsreduzierte Modell mit vereinfachter Struktur zu finden, benötigt der Genetische Algorithmus für die Berechnung der optimierten Matrix **E** ca. 80 Generationen bei einer Populationsgröße von 40 Individuen. Da aber sowohl die Rekombination als auch die Mutation nur mit einer bestimmten Wahrscheinlichkeit ausgeführt werden, müssen nicht pro Generation 40 neue Fitnesswerte berechnet werden. Die Gesamtzahl von berechneten Fitnesswerten liegt mit knapp 2800 unter der theoretisch maximalen Anzahl von $80 \cdot 40 = 3200$. Es ist aber zu berücksichtigen, dass ein GA selten beim ersten Versuch eine gute Lösung findet, sondern die Parameter vielmehr dem Problem angepasst werden müssen, sodass mehrere Durchläufe erforderlich sind. Bei der großen Anzahl der Individuen von $3 \cdot 10^{29}$ im Suchraum ist der GA trotzdem ein geeignetes Werkzeug, um dieses Optimierungsproblem zu lösen.

Darüber hinaus muss darauf hingewiesen werden, dass der GA lediglich in der Lage ist, die *Region* mit der oben gezeigten Lösung zu finden. Es bietet sich dann an, diese Region mit einem lokal ausgerichteten Suchverfahren nach besseren Lösungen zu durchsuchen. Dies kann ebenfalls wieder ein GA sein, dessen Parameter entsprechend geändert wurden, oder aber ein alternatives Verfahren, wie *Tabu Search* [25] oder *Simulated Annealing* [15].

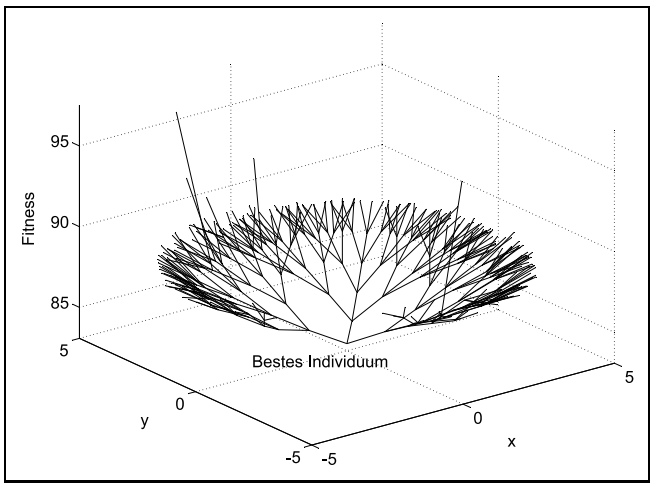

Bild 7: Ausschnitt des Suchraums in der Umgebung des besten Individuums. Ausgehend vom besten gefundenen Individuum sind jeweils nur die drei besten Nachbarn (mit Hamming-Distanz eins) dargestellt; durchaus existierende, drastisch schlechtere Nachbarn sind weggelassen.

4 Literaturhinweise

Weite Verbreitung fanden die GAs vor allem durch das Buch von David E. Goldberg [4], einem Holland-Schüler. Dieses Werk ist zwar sehr bekannt und wird immer wieder zitiert, aber in einigen Teilen gilt es als überholt. Einen aktuelleren Einstieg in die Thematik bekommt man daher durch die Werke von V. Nissen [7], T. Bäck [10], K. Weicker [11] oder M. Mitchell [9]. Ist es aufgrund der Codierung des Optimierungsproblems nicht oder nur schwer möglich, auf eine binäre Darstellung zurückzugreifen, so bietet Z. Michalewicz in [8] eine gute Hilfestellung. Im Internet sind darüber hinaus weitere Hilfestellungen zu finden. So ist z. B. unter `www.aic.nrl.navy.mil/galist/` ein Archiv zum Thema GA zu finden, wo u. a. Verknüpfungen zu diverser Software aufgeführt sind. Eine MATLAB-Toolbox mit entsprechender Beschreibung kann z. B. unter `www.ie.ncsu.edu/mirage/GAToolBox/gaot/` heruntergeladen werden.

Literatur

[1] D. A. Wimmer, R. Chattergy. Introduction to Non-linear Optimization: A Problem Solving Approach. New York, Amsterdam, Oxford: North-Holland, 1978.

[2] H. Th. Jongen, P. Jonker, F. Twilt. Nonlinear Optimization in $\mathbb{R}^n$ – II. Transversality, Flows, Parametric Aspects. Frankfurt am Main, Bern, New York: Lang, 1986.

[3] J. H. Holland. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. 1. Aufl.: Ann Arbor: Univ. of Michigan Press, 1975, 2. Aufl.: Cambridge, Mass. [u. a.]: MIT Press, 1993.

[4] D. E. Goldberg. Genetic algorithms in search, optimization, and machine learning. Reading, Mass.: Addison-Wesley, 1989.

[5] D. E. Goldberg, K. A. Deb. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In [6], S. 69–93.

[6] G. J. E. Rawlings (Hrsg.). Foundations of Genetic Algorithms. San Mateo, CA: Morgan Kaufmann Publishers, 1991.

[7] V. Nissen. Einführung in Evolutionäre Algorithmen – Optimierung nach dem Vorbild der Evolution. Braunschweig, Wiesbaden: Vieweg, 1997.

[8] Z. Michalewicz. Genetic Algortihms + Data Structures = Evolution Programs. 2nd edition, Berlin [u. a.]: Springer-Verlag, 1994.

[9] M. Mitchell. An Introduction to Genetic Algorithms. Cambridge, Massachusetts, London, England: The MIT Press, 1996.

[10] T. Bäck. Evolutionary Algorithms in Theory and Practice. New York (NY): Oxford University Press, 1996.

[11] K. Weicker. Evolutionäre Algorithmen. Stuttgart: B.G. Teubner GmbH, 2002.

[12] J. E. Baker. Adaptive Selection Methods for Genetic Algorithms. In [13].

[13] J. J. Grefenstette (Hrsg.). Proceedings of an International Conference on Genetic Algoritms and Their Applications. Hillsdale (NJ): Lawrence Erlbaum, 1985.

[14] C. L. Karr, L. M. Freeman. Industrial Applications of Genetic Algorithms. Boca Raton: The CRC Press, 1999.

[15] D. T. Pham, D. Karaboga. Intelligent Optimization Techniques – Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks. London: Springer-Verlag, 2000.

[16] H. Mühlenbein. How Genetic Algorithm Really Work I. Mutation and Hillclimbing. Proceedings of the Second Conference on Parallel Problem solving from Nature. Amsterdam: North-Holland, 1992.

[17] M. Leš, R. Svečko. Verbesserung der nichtparametrischen prädiktiven Regelung von nichtminimalphasigen Regelstrecken. **at** – Automatisierungstechnik 49(3), S. 132 ff., Oldenbourg Verlag, 2001.

[18] I. Rechenberg. Evolutionsstrategie '94. Stuttgart: Frommann-Holzboog 1994.

[19] H.-G. Beyer, H.-P. Schwefel. Evolution Strategies: A Comprehensive Introduction. Natural Computing, 1(1), S. 3–52, 2002.

[20] B. Lohmann. Ordnungsreduktion und Dominanzanalyse nichtlinearer Systeme. VDI-Fortschrittsberichte, Reihe 8, VDI-Verlag, Düsseldorf 1994.

[21] B. Lohmann. Ordnungsreduktion und Dominanzanalyse nichtlinearer Systeme. **at** – Automatisierungstechnik 42(10), S. 466 ff., Oldenbourg Verlag 1994.

[22] M. Buttelmann, B. Lohmann. Model Simplification And Order Reduction of Non-linear Systems With Genetic Algorithms. Proceedings of the IMACS Symposium on Mathematical Modelling, 3rd MATHMOD, Vienna 2000, S. 777–781.

[23] M. Buttelmann, B. Lohmann. Genetische Algorithmen für die Strukturvereinfachung ordnungsreduzierter, nichtlinearer Systeme. Proceedings 10. Workshop Fuzzy Control, Dortmund, 2000, S. 140–149.

[24] M. Buttelmann, B. Lohmann. Non-linear Model Reduction by Genetic Algorithms with Using a System Structure Related Fitness Function. European Control Conference (ECC) 2001, Porto, Portugal, S. 1870–1875.

[25] M. Buttelmann, B. Lohmann. Two Optimisation Methods for Solving the Problem of Structure Simplification of Nonlinear Systems. Methods and Models in Automation and Robotics (MMAR) 2002, Szczecin, Poland.

[26] VDI/VDE-Richtlinie 3550, Blatt 3. Computational Intelligence: Evolutionäre Algorithmen, Begriffe und Definitionen. VDI/VDE-Handbuch Regelungstechnik, Berlin: Beuth Verlag, 2003.

[27] B. Morgenstern. Elektronik III. Digitale Schaltungen und Systeme. Braunschweig/Wiesbaden: Vieweg Verlag, 1992.

[28] Meyers großes Taschenlexikon. Mannheim: BI-Taschenbuchverlag, 1992.

Manuskripteingang: 13. Dezember 2002.

Dipl.-Ing. Maik Buttelmann ist wissenschaftlicher Mitarbeiter des Institutes für Automatisierungstechnik an der Universität Bremen. Hauptarbeitsgebiete: Genetische Algorithmen, Ordnungsreduktion, Strukturvereinfachung.

Adresse: Universität Bremen, Fachbereich 01 Elektrotechnik und Informationstechnik, Institut für Automatisierungstechnik, Postfach 33 04 40, 28334 Bremen, DEUTSCHLAND, Fax: +49-(0)421-218-4707,
E-Mail: maik.buttelmann@iat.uni-bremen.de

Prof. Dr.-Ing. habil. Boris Lohmann ist kollegialer Leiter des Instituts für Automatisierungstechnik der Universität Bremen und vertritt im Studiengang Elektrotechnik das Fachgebiet Systemdynamik und Regelungstechnik. Neben der Modellbildung und Modellvereinfachung liegen Arbeitsschwerpunkte der Gruppe bei der kameragestützten Navigation autonomer Fahrzeuge und der aktiven Geräuschdämpfung mit Anwendung im KFZ.

Adresse: wie oben, E-Mail: BL@iat.uni-bremen.de