Optimierung mit Evolutionären Algorithmen
Optimization by Means of Evolutionary Algorithms

Frank Kursawe und Hans–Paul Schwefel, Dortmund

Universität Dortmund
Fachbereich Informatik
Lehrstuhl für Systemanalyse
44221 Dortmund
Tel.: 0231 / 9700 - 976 bzw. - 952 FAX: 0231 / 9700 - 959
http://LS11-www.cs.uni-dortmund.de/

Dipl.–Inform. Frank Kursawe ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Systemanalyse (Professor Schwefel) im Fachbereich Informatik an der Universität Dortmund; Hauptarbeitsfelder: Systemanalyse, Evolutionsstrategien

Prof. Dr.-Ing. Hans–Paul Schwefel ist seit 1985 Inhaber des Lehrstuhls Informatik XI (Systemanalyse) an der Universität Dortmund. Ferner ist er Präsident des Informatik Centrum Dortmund (ICD) und Sprecher des Sonderforschungsbereichs „Design und Management komplexer technischer Prozesse und Systeme mit Methoden der Computational Intelligence“ (SFB 531) sowie Mitherausgeber der Zeitschriften „BioSystems“, „Evolutionary Computation“ und „IEEE Transactions on Evolutionary Computation“; Hauptarbeitsfelder: Ingenieurinformatik, Systemanalyse, naturanaloge parallele Problemlösungsmethoden

Zusammenfassung

In diesem Artikel werden die beiden wichtigsten Algorithmen vorgestellt, die Prinzipien der biologischen Evolution nachahmen, um schwierige Optimieraufgaben zu lösen, für die traditionelle Algorithmen versagen. Sie werden heute unter dem Namen *Evolutionäre Algorithmen* zusammengefaßt. Neben der Erklärung der Funktionsweise werden auch Hinweise gegeben, wie man durch die Einbeziehung weiterer natürlicher Prinzipien zu noch besseren Algorithmen gelangen kann.

In this article the two most important algorithms are described that mimic principles of organic evolution in order to solve optimization problems being too difficult for traditional algorithms. Nowadays, they are called *Evolutionary Algorithms*. Besides the explanation how these algorithms work, some hints are given how one can improve these algorithms by including more natural principles from biology.

1 Einleitung

Seit etwa vier Milliarden Jahren gibt es Leben auf unserer Erde. Neben der erstaunlichen Artenvielfalt hat die Evolution auch viele Organismen und Formen hervorgebracht, die gut an ihre jeweilige Umwelt angepaßt sind. Reduziert man den Blickwinkel z. B. auf Teile wie Blutgefäßsysteme, für die es seit langer Zeit unveränderte Naturgesetze zu lernen galt, kann man sogar von optimalen Anpassungen sprechen [16], so z. B. beim Erlernen der optimalen Durchmesser an Verzweigungsstellen in Blutgefäßsystemen. Mit der Bionik gibt es seit Anfang der sechziger Jahre einen Wissenschaftszweig, der Lösungen aus der Natur in technische Produkte umzusetzen sucht.

Offenbar wohnt dem hochkomplexen natürlichen Evolutionsprozeß *auch* eine Optimierkomponente inne. Warum sollte man nicht versuchen, durch die Nachahmung von grundlegenden Evolutionsprinzipien zu neuen, eventuell besonders leistungsfähigen bzw. generell anwendbaren Optimierverfahren zu kommen?

2 Evolutionäre Algorithmen

Anfang der sechziger Jahre stellten sich verschiedene Forscher unabhängig voneinander die Frage, wie man jene Mechanismen, die zu der heute beobachtbaren Vielfalt des Lebens geführt haben, für Optimierungszwecke nutzbar machen könnte. In Deutschland führte dies zu den *Evolutionsstrategien* (ES) [14, 17], in den U.S.A. zu den *Genetischen Algorithmen* (GA) [11] und zum Konzept des *Evolutionary Programming* (EP) [8]. Evolutionsstrategien und Genetische Algorithmen werden im weiteren noch detailliert vorgestellt, für eine gute Darstellung des Evolutionary–Programming–Konzepts sei auf [4] und [5] verwiesen. Bezüglich neuerer Literatur sei auch auf [15, 18] verwiesen.

Diese Verfahren sowie das später hinzugekommene *Genetic Programming* (GP), das evolutionäre Suchprinzipien in den Suchraum von Programmiersprachen überträgt, werden heute unter dem Namen *Evolutionäre Algorithmen* (EA) zusammengefaßt, da sie alle bestrebt sind, Prinzipien der biologischen Evolution nachzuahmen, um Optimierprobleme zu lösen.

Evolutionäre Algorithmen werden dazu verwendet, Optimierprobleme folgender Form zu lösen: Eine „Black Box“ mit Variationsmöglichkeit der Eingänge — diese können diskreter Natur (Schalter) und/oder kontinuierlicher Natur (Regler) sein — sei gegeben, die als Antwort auf die Regler– bzw. Schaltereinstellungen ein Qualitätsmaß ausgibt. Dieses wird üblicherweise als reelle Zahl gemessen, im Prinzip reicht aber auch eine Antwort in der Form „Einstellung A ist besser als Einstellung B“. Eine Metapher möge dies veranschaulichen: Ein Optimierverfahren steht vor dem Problem wie ein Bergsteiger im Nebel. Von einem beliebigen Punkt ausgehend soll möglichst der Mt. Everest erklommen werden. Zumindest sollte aber die Himalaya–Region erreicht werden, auf keinen Fall möchte man

schon in einem Mittelgebirge steckenbleiben. Das Bild 1 zeigt zwei künstliche Gebirge (genauer: Zielfunktionstopologien), die jeweils in Abhängigkeit von $n = 2$ Eingangsgrößen (x und y) ein Maß für die Qualität eines Punktes – die erreichte Höhe – liefern. Einen zweidimensionalen Suchraum wie in Bild 1 kann man natürlich von einem Rechner noch sehr schnell rastern lassen und so denjenigen Punkt (x, y) mit der größten Höhe finden. Optimierprobleme aus der Praxis verfügen aber über deutlich mehr als zwei Eingangsgrößen, so daß eine Rasterung wegen des exponentiell in n steigenden Aufwandes nicht mehr anwendbar ist. Mit zunehmender Größe des Suchraumes benötigt man also Verfahren, die sich einen Weg im Gebirge suchen, der an einer möglichst hoch gelegenen Stelle endet. In Kasten 2 wird formaler in das Problem der Optimierung eingeführt.

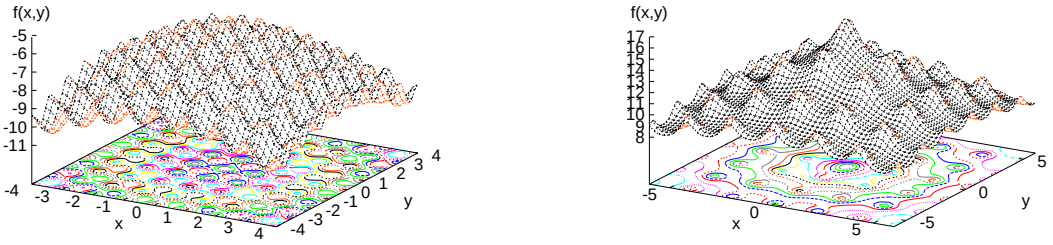

Abbildung 1: Zwei schwierige Zielfunktionstopologien oder Gebirge.

Evolutionäre Algorithmen ahmen Prinzipien der biologischen Evolution nach. Entsprechend stammen viele der Begriffe, die im folgenden verwendet werden, aus der Biologie. Für diejenigen, denen diese biologischen Begriffe nicht vertraut sind, stellt „Kasten 1: Glossar“ den Zusammenhang zwischen der biologischen und den mathematisch–technischen Begriffswelt her.

Der Einsatz evolutionärer Regeln in der Informationsverarbeitung ist nur dann sinnvoll, wenn es entweder keine spezialisierten Verfahren gibt oder aber die traditionellen Optimierverfahren Schwierigkeiten bekommen, weil die Zielfunktion nichtlinear, diskontinuierlich, „mehrgipfelig“ (multimodal) oder stochastisch gestört ist — Die Gebirgslandschaft ist stark zerklüftet, enthält senkrechte Wände, Plateaus, und der Boden bewegt sich an einigen Stellen leicht. Evolutionäre Verfahren überwinden diese Probleme dadurch, daß sie einerseits *nicht* prädiktiv anhand eines spezifischen Modells der Zielfunktion lokale Informationen sammeln und ausnutzen und andererseits üblicherweise mit einer Gruppe von kommunizierenden „Bergsteigern“ arbeiten statt nur von *einer* aktuell besten Position ausgehend weiter zu suchen.

In der Begriffswelt der Evolutionären Algorithmen heißt eine zusammengehörende Menge von zulässigen Reglereinstellungen Individuum. Üblicherweise wird einem Individuum über die Zielfunktion sein Fitneßwert zugeordnet. Auf das Bergsteigerbeispiel

übertragen, repräsentiert ein Individuum eine Position im Gebirge, die anhand der erreichten Höhe bewertet und gegebenenfalls mit anderen verglichen wird. Dies kann einerseits experimentell geschehen; dann benötigt man ein Optimierverfahren, das sich robust gegenüber Störungen durch Meßfehler (schwankendem Boden im Gebirge) verhält. Andererseits gibt es die Möglichkeit, die Fitneß durch Computersimulation zu berechnen.

Eine Menge von μ Individuen bildet die Population, die sich unter Verwendung folgender (genetischer) Operatoren verändert: Die Mutation modifiziert die genetische Information eines Individuums, nachdem die Rekombination „nur“ genetisches Material meist zweier Eltern neu zu einem Nachkommen zusammengesetzt hat. Die Umweltselektion bestimmt anhand der Fitneßbewertung, welche Individuen in der nächsten Generation zur Paarungsselektion zugelassen werden. Im Rückgriff auf die Metapher des Bergsteigens kann man also von einer Menge von Wanderern sprechen, die Informationen austauschen und von denen nur die besten weiterklettern bzw. mehr Kinder bekommen dürfen. Bild 2 stellt das allgemeine Iterationsschema eines evolutionären Algorithmus dar.

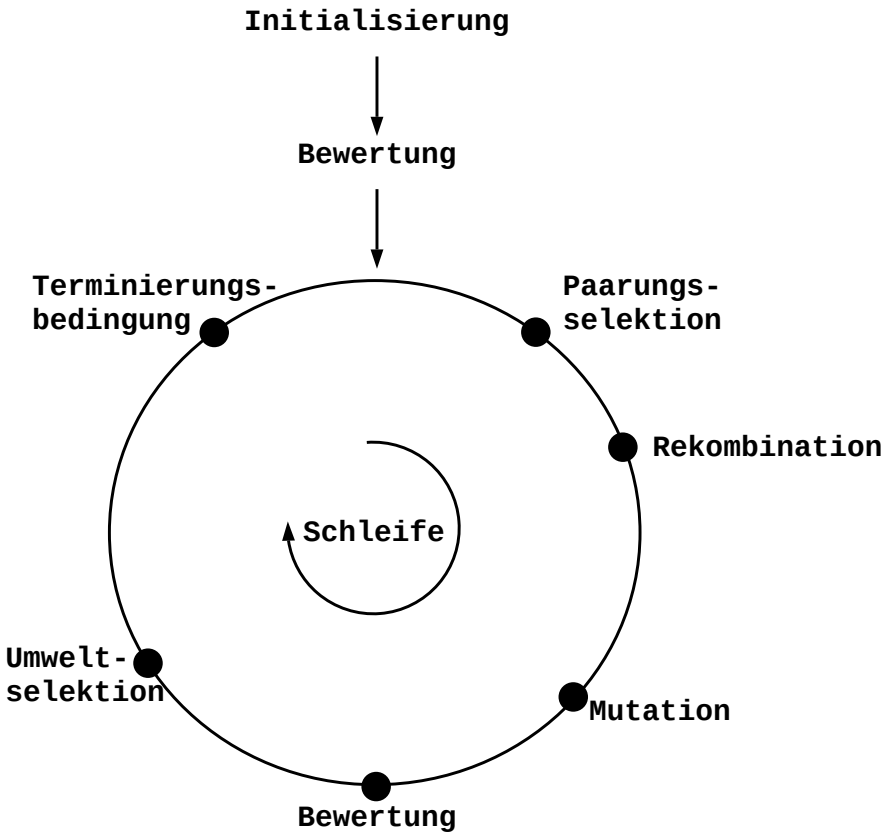

Abbildung 2: Iterationsschema eines Standard–EA.

Nachdem eine Anfangspopulation generiert wurde, wird die Schleife solange durchlaufen, bis eine Terminierungsbedingung greift. Die Paarungsselektion wählt zufällig zwei oder mehr aus jenen Individuen aus, welche die Umweltselektion des vorherigen Schleifendurchlaufes überlebt haben. Die Ausgewählten dürfen ihre Informationen (Gene) zu

einem Nachkommen rekombinieren, sich also fortpflanzen. Anschließend sorgt die Mutation für eine Modifikation der Informationen, die dann anhand der Fitneß– oder Zielfunktion bewertet werden. Dieses Maß bildet die Basis für die Umweltselektion, die nur einem Teil der Population erlaubt, in der nächsten Iteration Nachkommen zu erzeugen. Dies setzt natürlich voraus, daß die Anzahl der Nachkommen die der Eltern übersteigt (Geburtenüberschuß).

Die verschiedenen Klassen evolutionärer Algorithmen unterscheiden sich durch die Repräsentation der Individuen und demzufolge auch durch die auf ihr arbeitenden Operatoren. Für den Fall der Parameteroptimierung befindet sich in [5] eine gute Übersicht.

Ein heute wichtiger Vorteil evolutionärer Verfahren für deren praktischen Einsatz ist ihre inhärente und skalierbare Parallelität [10]. Auf diese Weise kann die Rechenleistung beliebiger Parallelrechnerarchitekturen für rechenintensive Probleme genutzt werden, z. B. Simulationsmodelle.

Die beiden wichtigsten, sich jährlich abwechselnden, Konferenzen zu allen Aspekten evolutionären Rechnens sind die *International Conference on Genetic Algorithms* (ICGA), die seit 1985 stattfindet, und die seit 1990 ausgerichtete Konferenzserie *Parallel Problem Solving from Nature* (PPSN).

2.1 Evolutionsstrategien (ES)

Obwohl ursprünglich für die experimentelle Optimierung und diskrete Suchräume entwickelt, bildet der $\mathbb{R}^n$ seit der Realisierung auf Rechnern typischerweise den Suchraum von Evolutionsstrategien. Daher besteht ein Individuum aus einem n–dimensionalen reellwertigen Objektvariablen– oder Parametervektor $\vec{x}$ und einer Menge von Strategieparametern (Schrittweiten und gegebenenfalls Kovarianzen), welche die Mutation steuern (siehe auch Kasten 2). Die Objektvariablen sind die Regler an der Black Box, die ihnen zugeordneten Strategieparameter legen fest, wie stark die Mutation an diesen Reglern drehen darf. Oder, um auf die Bergsteiger zurückzukommen: Die Strategieparameter repräsentieren die Schrittlänge eines Individuums in jede der n Richtungen, sowie gegebenenfalls auch Korrelationen zwischen ihnen.

Die Besonderheit von ES liegt darin begründet, daß sowohl Objekt– als auch Strategieparameter rekombiniert und mutiert werden, wodurch sich bei entsprechender Selektion die Möglichkeit einer ständigen Selbstadaptation der Schrittweiten ergibt. Die Bergsteiger geben also nicht nur ihre jeweiligen Positionen im Gebirge an ihre Nachkommen weiter, sondern auch Information über Größe und Vorzugsrichtungen von Veränderungen (die „Schrittlänge“). Im Gegensatz zu traditionellen Optimierverfahren, die nur dann schnell konvergieren, wenn das eingebaute interne Modell zur jeweiligen Zielfunktion paßt, ent-

wickeln ES ein internes Modell der Fitneß–Landschaft erst während der Suche. Bild 3 veranschaulicht die Trennung in Geno– und Phänotyp:

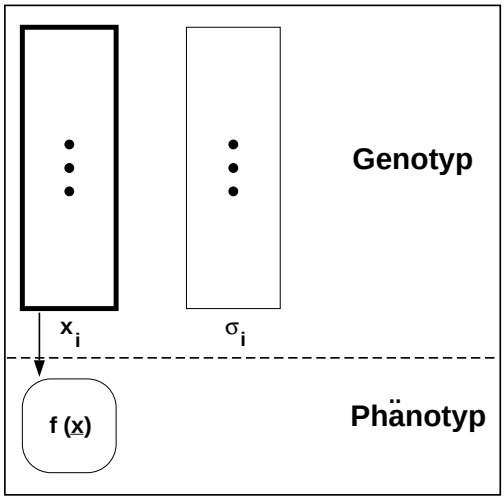

Abbildung 3: ES–Geno– und Phänotyp.

Die x_i entsprechen der Position eines Bergsteigers, die σ_i repräsentieren die Schrittlängen in jede der n Richtungen. Der Phänotyp oder die Fitneß besteht aus der Gebirgshöhe an der durch die x_i festgelegten Position. Die Schrittweiten werden also nicht direkt mitbewertet. Wenn ein Nachkomme eine gute Position vererbt bekommt, aber für den weiteren Fortschritt ungeeignete Schrittweiten, wird ihn die Selektion im nächsten Schritt verwerfen.

Für die Rekombination gibt es folgende Möglichkeiten, jeweils getrennt bei Objekt– und Strategieparametern einstellbar:

- Keine Rekombination

  Ein Nachkomme wird durch Klonen eines Elters erzeugt. Diese Variante wird nur der Vollständigkeit halber und zu Vergleichszwecken mit aufgeführt, in der Praxis sind Evolutionsstrategien mit Rekombination immer erfolgreicher.

- Diskrete Rekombination zweier Eltern

  Zwei Eltern werden „ausgewürfelt“, und für jede der n Komponenten x_i erbt der Nachkomme diese Information zufällig von Elter 1 oder Elter 2.

- Diskrete Rekombination aller μ Eltern

  Für jede der n Positionen wird ein Elternindividuum „ausgewürfelt“, das die entsprechende Komponente x_i vererbt. Die Population wird in dieser Variante als Genpool betrachtet, aus dem einzelne Gene gezogen werden.

- Intermediäre Rekombination zweier Eltern

Zwei Eltern werden „ausgewürfelt“, und für jede der n Komponenten x_i erbt der Nachkomme den Mittelwert der beiden Elternpositionen.

- Intermediäre Rekombination aller μ Eltern

  Für jede der n Komponenten x_i werden je zwei Eltern „ausgewürfelt“, deren Mittelwert dem Nachkommen zugewiesen wird.

Alle Varianten lassen sich biologisch rechtfertigen, je nachdem welche Art von Lebewesen bzw. Genen man betrachtet. Bild 4 veranschaulicht die Unterschiede zwischen intermediärer und diskreter Rekombination.

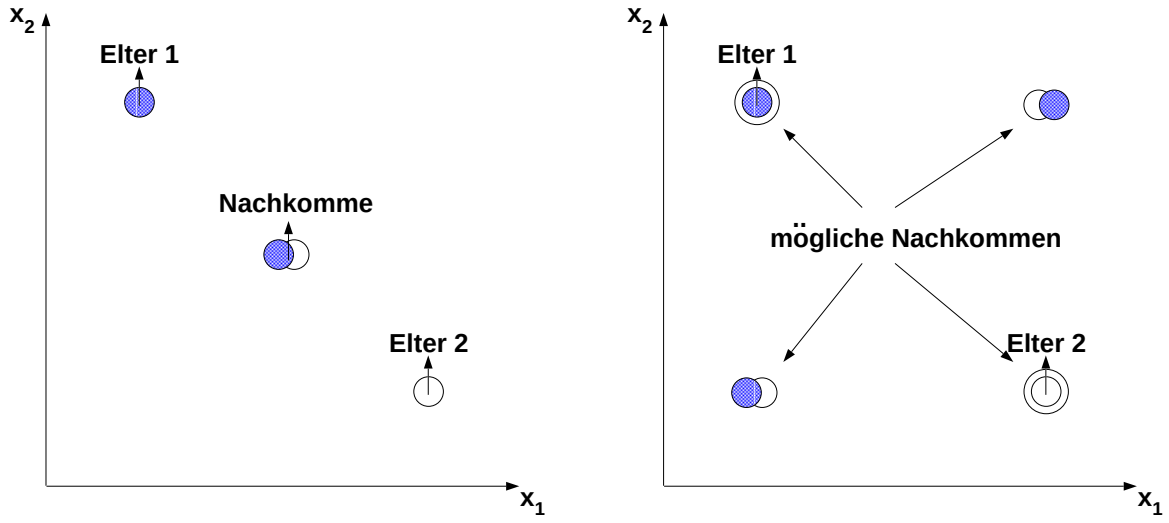

Abbildung 4: ES–Rekombinationstypen: intermediär (links) und diskret (rechts)

Mutationen werden in einer Weise vorgenommen, daß kleine Änderungen wahrscheinlicher sind als größere — Der Apfel fällt nicht weit vom Stamm. Dies wird auf einem Rechner über die Addition normalverteilter Zufallszahlen mit Erwartungswert 0 realisiert. Bei der Mutation der Strategieparameter wird multiplikativ eine logarithmische Normalverteilung verwendet, so daß die Halbierung bzw. Verdopplung einer Schrittweite gleich wahrscheinlich auftritt.

Durch Rekombination und Mutation werden aus μ Eltern pro Generation λ Nachkommen erzeugt ($\mu \leq \lambda,\ \mu/\lambda \approx 1/7$), welche die Selektion gemäß der jeweiligen Fitneß danach wieder auf μ Individuen reduzieren muß. Bild 5 veranschaulicht die Selektion in einer ES: Die μ Eltern einer Generation dürfen sich mit gleicher Wahrscheinlichkeit $p_i = 1/\mu$ fort-

pflanzen. Das Individuum mit der besten Fitneß wird also unter den Überlebenden *nicht* bevorzugt.

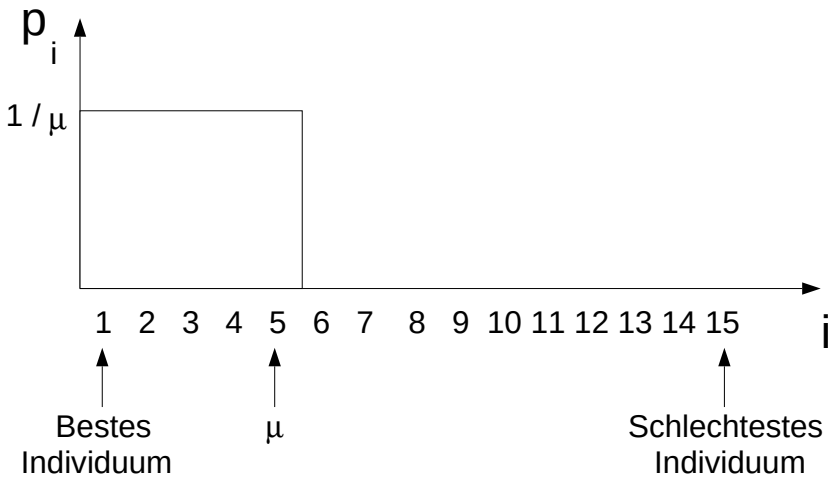

Abbildung 5: $(\mu \dot{+} \lambda)$–Selektion einer ES (hier: $(5 \dot{+} 15)$–ES).

Je nach gewählter Selektionsvariante werden die μ besten aller $(\mu + \lambda)$ Individuen zu Eltern der nächsten Generation (Plus–Selektion, gekennzeichnet durch das „+“), oder die Selektion verwirft die Eltern nach jedem Reproduktionszyklus und macht die μ besten Individuen der λ Nachkommen (μ, λ) zu Eltern der nächsten Generation (Komma–Selektion, gekennzeichnet durch das „,“ im Klammerausdruck zwischen μ und λ). Gute Ergebnisse erzielt man oft mit einer $(\mu = 15, \lambda = 100)$–Evolutionsstrategie.

Die Komma–Variante erscheint auf den ersten Blick verschwenderisch, weil sie auch Verschlechterungen von einer Generation zur nächsten in Kauf nimmt. Um auf einen höheren Berg zu gelangen, muß zunächst ein Tal durchwandert werden. Aber *nur* die Komma–Variante ermöglicht die ständige Selbstadaptation der Strategieparameter. Dies ist wichtig, weil auf dem Weg zum Gipfel schmale Täler oder steile Grate warten können. Ferner erlaubt es diese Variante auch, z. B. wandernden Optima zu folgen. Unsere Bergsteiger können sich also auch in einem Gebirge bewegen, das sich mit der Zeit verändert.

Die Plus–Variante dagegen erhält zwar Individuen mit guter Fitneß, jedoch können diese durch einen Zufallstreffer auf einen lokalen Gipfel gelangt sein und werden dann in jeder Generation für den weiteren Anstieg ungeeignete Schrittweiten an ihre Nachkommen vererben.

In Kasten 3 findet man den Algorithmus einer Standard–ES, im Gegensatz zu Bild 2 in einer Pseudo–Programmiersprache dargestellt.

Schwefel und Rudolph haben die $(\mu, \kappa, \lambda, \rho)$–ES als neue Variante vorgeschlagen, welche die Plus– und Komma–Variante zusammenführt [19]. Über den Parameter κ kann man das maximale Alter eines Individuums steuern, für $\kappa = 1$ erhält man eine Komma–,

für $\kappa = \infty$ eine Plus–Strategie. Die Anzahl der an der Rekombination eines Nachkommens beteiligten Eltern wird über ρ gesteuert, wobei gilt: $1 \leq \rho \leq \mu$. Hat man es mit mehr als einer Zielfunktion zu tun, gibt es auch bereits eine ES–Variante, die eine Teilmenge der effizienten Lösungen derart berechnet, daß man einen guten Einblick in die Struktur des Problems gewinnen kann [12].

Bis auf einen anderen Selektionsmechanismus und den Verzicht auf Rekombination ist das Konzept des *Evolutionary Programming* mit dem der ES fast identisch, so daß an dieser Stelle auf Details verzichtet wird. In [7] findet man eine genauere Beschreibung, in [5] eine vergleichende Darstellung.

2.2 Genetische Algorithmen (GA)

Angenommen, die Black Box verfügt nicht mehr über kontinuierlich regelbare Eingänge, sondern über Ein–Aus–Schalter als Inputs: In GAs werden typischerweise aus Nullen und Einsen bestehende Ketten (Bitstrings) der Länge l zur Repräsentation von Individuen verwendet, also Elemente des Raumes $\{0,1\}^l = \mathbb{B}^l$ (siehe auch Kasten 2). Dies bedeutet jedoch nicht, daß man mit GAs nur pseudo–Boole'sche Probleme angehen kann, für die diese Kodierung direkt geeignet ist.

Komplexere Datenstrukturen, wie reelle Zahlen, Listen, Bäume usw. müssen durch entsprechende Abbildungen auf Bitstrings abgebildet werden. Allerdings schwächt jede zusätzliche Abbildung zwischen der binären und der Problemrepräsentation die nötige Kausalität zwischen Geno– und Phänotyp, so daß im Einzelfall zu überlegen ist, ob man nicht eine für das Problem generische Repräsentation wählt und die genetischen Operatoren anpaßt statt einen Standard-GA zu verwenden und das Problem mehr oder weniger geschickt in einem Bitstring zu kodieren.

Eine GA-Population besteht aus μ Individuen, typischerweise $\mu = 50$. Die (Paarungs–) Selektion wählt zwei Individuen für die nächste Paarung mit einer zu ihrer Fitneß proportionalen Wahrscheinlichkeit aus. Wie in Bild 6 dargestellt, kann man sich eine Art „Glücksrad“ vorstellen, auf dem die Individuen eine Fläche gemäß ihrem Anteil an der Gesamtfitneß bekommen. Der Selektionszeiger wird also häufiger die besseren Individuen zur Fortpflanzung auswählen, was zu einer vorzeitigen Konvergenz des Algorithmus

schon bei einfachen Problemen führen kann. Bei kleinen Populationen wie in Bild 6 wird Individuum 4 sehr schnell die gesamte Population mit seinen Genen dominieren.

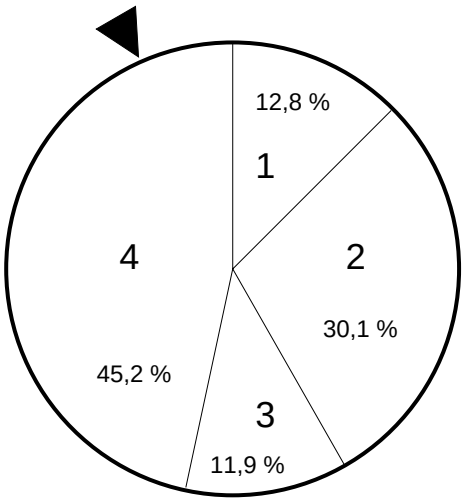

Abbildung 6: Fitneß–proportionale Selektion eines GA.

Mit Wahrscheinlichkeit $p_{crossover}$, typischerweise $0,6$ [6], werden exakt μ mal je zwei Eltern zufällig bestimmt, ferner eine ganze Zahl c aus $\{1, 2, \ldots, l-1\}$. Der Nachkomme erhält die ersten c Bits von dem einen, den Rest vom anderen Elternteil. Anschließend wird mit Wahrscheinlichkeit $p_{mutation}$, üblicherweise $0,001$ [6], entschieden, ob jedes Bit mutiert (invertiert) wird oder nicht. Dieses Vorgehen wird wiederholt, bis ein Terminierungskriterium greift. *Im Gegensatz zu Evolutionsstrategien findet also keine Umwelt–, sondern nur eine Paarungsselektion statt.*

Die Darstellung von GAs fällt verglichen mit dem Abschnitt über ES deutlich kürzer aus, weil auf Bitstrings die Manipulationsmöglichkeiten durch genetische Operatoren eingeschränkt sind. Die Rekombination kann diese Datenstruktur nur an einer oder mehreren Stellen „zerhacken“ und neu zusammensetzen, und der Mutation bleibt nur die Invertierung einzelner Positionen. Da die Mutationsrate vorgegeben ist, bleibt nur die Wahl, mit „Siebenmeilenstiefeln“ oder Trippelschritten durch das Gebirge zu laufen. Eine Selbstadaptation der Schrittweite (hier: der Mutationswahrscheinlichkeit oder –rate) ist mit einem Standard–GA nicht möglich.

In Kasten 4 findet man den Algorithmus eines Standard–GA. Um den Vergleich mit der in Kasten 3 dargestellten Standard–ES zu erleichtern, wurde dieselbe Pseudo–Programmiersprache gewählt.

Als algorithmisch orientierte Einführung eignet sich [9], für problemspezifische Anpassungen von GAs sei auf [13] verwiesen.

3 Anwendungen Evolutionärer Algorithmen

Probleme aus der industriellen Praxis sind typischerweise mit Restriktionen behaftet. Um stets über eine komplette Population gültiger Lösungen verfügen zu können, sind verschiedene Ansätze möglich:

- Unzulässige Individuen werden durch die Wahl der Kodierung prinzipiell ausgeschlossen.
- Unzulässige Individuen werden sofort verworfen und der Rekombinations– und Mutationsprozeß solange wiederholt, bis hinreichend viele Individuen im zulässigen Bereich liegen.
- Unzulässige Individuen werden mit einem Fitneßmalus bestraft.
- Unzulässige Individuen werden genetisch „repariert“.

Evolutionäre Algorithmen haben sich in vielen Anwendungsfeldern als Möglichkeit erwiesen, entweder existierende Lösungen zu verbessern oder überhaupt erst eine Optimierung zu ermöglichen. Die folgende Aufzählung soll lediglich einen Eindruck von der Vielfalt der Problemdomänen geben. Weitere Anwendungen evolutionärer Algorithmen findet man in [3], die umfangreichste Sammlung von Literaturreferenzen zum Thema EA bietet [1].

- Maschinenbelegungsprobleme,
- Packungsprobleme,
- Nährmedienoptimierung für biotechnologische Prozesse,
- Pipelineüberwachung,
- Stundenplanprobleme,
- Kraftwerkseinsatzplanung,
- Tragwerkoptimierung,
- Synthese eines Viergelenkgetriebes,
- Optimierung der externen Parameter von Simulationsmodellen,
- Optimierung optischer Linsen.

Diese Vielfalt liegt darin begründet, daß Evolutionäre Algorithmen außer dem Qualitätsmaß keine Informationen über das Innere der Black Box benötigen. Diese Universalität wird natürlich auf der anderen Seite mit einem Geschwindigkeitsverlust gegenüber jenen Verfahren bezahlt, die speziell für ein Problem entworfen wurden und problemspezifisches Wissen ausnutzen. Je mehr ein Bergsteiger über das Gebirge weiß, umso sicherer und schneller wird er den höchsten Gipfel finden.

4 Zusammenfassung

Dieser Artikel hat die wesentlichen Unterschiede der beiden Hauptrichtungen evolutionärer Algorithmen, Evolutionsstrategien und Genetische Algorithmen, dargestellt. Trotz aller Unterschiede in der jeweiligen Realisierung beweist das zunehmende Interesse an diesen vor fast 30 Jahren kreierten Verfahren doch die hohe Problemlösungsfähigkeit und Robustheit der zugrundeliegenden evolutionären Prinzipien.

Verwendet man eine der hier vorgestellten Varianten, kann man zwar auf getestete Software und eine recht gut entwickelte Theorie zurückgreifen, aber in der hier dargestellten Form dürften sie für praktische Probleme nur selten direkt anwendbar sein. Kritische Punkte sind dabei unter anderem die Problemrepräsentation sowie die Behandlung von Restriktionen. Viele Beispiele, wie GAs durch Hybridisierung leistungsfähiger gemacht werden können, findet man in [2] und [13].

Eine Hoffnung besteht allerdings darin, daß alle hier erwähnten Verfahren die biologische Evolution erst auf einer sehr einfachen Ebene modellieren. Ein besseres Verständnis von natürlichen Phänomenen wie Alterung, Geschlechterdifferenzierung, Anpassung an sich ändernde Umwelten (Erhaltung der genetischen Diversität und damit der Evolvierbarkeit), Ausnutzung der Parallelität ohne globales Wissen über alle Fitneßwerte oder adaptive Genotyp–Phänotyp–Abbildungen dürfte zu noch breiter einsetzbaren Evolutionären Algorithmen führen.

Dieser Artikel wurde durch die Deutsche Forschungsgemeinschaft im Rahmen des Sonderforschungsbereichs 531 unterstützt.

Literatur

[1] J. T. Alander. An Indexed Bibliography of Genetic Algorithms: Years 1957–1993. Art of CAD Ltd, Espoo, Finland, 1994.

[2] Th. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[3] Th. Bäck, F. Hoffmeister und H.-P. Schwefel. Applications of Evolutionary Algorithms. Report of the Systems Analysis Research Group SYS–2/92, University of Dortmund, Department of Computer Science, February 1992.

[4] Thomas Bäck, Günter Rudolph und Hans-Paul Schwefel. Evolutionary Programming and Evolution Strategies: Similarities and Differences. In D. B. Fogel und W. Atmar (Hrsg.), *Proc. 2nd Annual Conf. Evolutionary Programming*, pages 11–22, La Jolla CA, 1993. Evolutionary Programming Society, San Diego CA.

[5] Thomas Bäck und Hans-Paul Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[6] K. A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975. Diss. Abstr. Int. 36(10), 5140B, University Microfilms No. 76–9381.

[7] D. B. Fogel. *Evolving Artificial Intelligence.* PhD thesis, University of California, San Diego, CA, 1992.

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

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

[10] F. Hoffmeister. Scalable Parallelism by Evolutionary Algorithms. In M. Grauer und D. B. Pressmar (Hrsg.), *Parallel Computing and Mathematical Optimization*, volume 367 of *Lecture Notes in Economics and Mathematical Systems*, pages 177–198. Springer, Berlin, 1991.

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

[12] Frank Kursawe. Evolution Strategies for Vector Optimization. In Gwo-Hshiung Tzeng und Po Lung Yu (Hrsg.), *Proc. 10th Int'l Conf. Multiple Criteria Decision Making*, pages 187–193, Taipei, 1992. National Chiao Tung University.

[13] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Artificial Intelligence. Springer, Berlin, 1992.

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

[15] I. Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik.* Frommann–Holzboog, Stuttgart, 1994.

[16] R. Rosen. *Optimality Principles in Biology.* Butterworths, London, 1967.

[17] H.-P. Schwefel. *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*. Birkhäuser, Basel, 1977.

[18] H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[19] H.-P. Schwefel und G. Rudolph. Contemporary Evolution Strategies. In F. Morán, A. Moreno, J. J. Merelo und P. Chacón (Hrsg.), *Advances in Artificial Life. Third International Conference on Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 893–907. Springer, Berlin, 1995.

Liste der Bildunterschriften:

Bild 1: Zwei schwierige Zielfunktionstopologien oder Gebirge.

Bild 2: Iterationsschema eines Standard–EA.

Bild 3: ES–Geno– und Phänotyp.

Bild 4: ES–Rekombinationstypen: intermediär (links) und diskret (rechts).

Bild 5: $(\mu \dot{+} \lambda)$–Selektion einer ES (hier: $(5 \dot{+} 15)$–ES).

Bild 6: Fitneß–proportionale Selektion eines GA.

Kasten 1: Glossar

Die im Text verwendeten biologischen Begriffe werden mit ihren mathematisch–technischen Entsprechungen in Beziehung gesetzt.

Individuum: Repräsentation eines Punktes im Lösungsraum, ein Satz von Entscheidungsvariablen.

Population: Menge von Individuen (Lösungen).

Diversität: Verschiedenheit der Individuen einer Population.

Eltern: durch die Selektion zur Reproduktion zugelassene Lösungen.

Elter: Singular von Eltern, dessen Verwendung sinnvoll ist, wenn ein Nachkomme durch Klonen und nicht durch Rekombination erzeugt wird.

Nachkommen, Kinder: von den Eltern erzeugte Individuen.

Geburtenüberschuß: zwei Eltern erzeugen mehr als nur zwei Nachkommen.

genetische Operatoren: Zusammenfassung von Rekombination, Mutation, Selektion.

Rekombination: vermischt die genetische Information von Individuen.

Klonen: Erstellung einer genetisch identischen Kopie eines Individuums.

Mutation: verändert die genetische Information von Individuen.

Selektion: läßt Individuen gemäß ihrer Fitneß zur Reproduktion zu.

Fitneß: Qualitätsmaß für ein Individuum, oft Zielfunktionswert.

Fitneßgebirge, Qualitätsgebirge: Veranschaulichung einer Zielfunktion durch geeignete Projektion.

Genom, Genotyp: gesamtes genetisches Material eines Individuums.

Genpool: Menge aller Gene in einer Population.

Phänotyp: dekodierter Genotyp, den die Selektion bewertet.

Kasten 2: Problemformulierung

Optimierproblem: $min\{f(\vec{x}) : \vec{x} \in \mathcal{M}\}$. Die Zielfunktion f bildet im allgemeinen den zulässigen Bereich $\mathcal{M}$ in die reellen Zahlen $\mathbb{R}$ ab. Minimierungs– und Maximierungsproblem sind äquivalent wegen $maxf = -min(-f)$.

Individuum (ES): $(\vec{x}, s_1, \ldots, s_r)$, $r \geq 1$. Dabei sei $\vec{x} \in \mathcal{M}$ eine zulässige Lösung, während die $s_1, \ldots, s_r$ Strategieparameter repräsentieren. Einem Individuum wird über die Zielfunktion $f(\vec{x})$ eine Fitneß zugeordnet.

Individuum (GA): $\{0,1\}^l$. Wenn das Problem binärer Natur ist, sind keine weiteren Abbildungsschritte nötig. Anderenfalls muß man die entsprechende Abbildung in den Definitionsbereich von f angeben.

Kasten 3: Grundalgorithmus einer ES

setze $t := 0$
initialisiere Population $P_0 \in \mathbb{R}^n$
bewerte die Individuen mittels f
`while` Terminierungskriterium nicht erfüllt `do`
  rekombiniere λ Nachkommen aus μ Elternindividuen (P_t) (mit gleicher Reproduktionswahrscheinlichkeit)
  mutiere die λ Nachkommen
  bewerte die λ Nachkommen mittels f
  selektiere μ Individuen für P_{t+1}:
    `if` (μ, λ)–Selektion `then` selektiere μ aus λ Nachkommen
      `else` selektiere μ aus μ Eltern und λ Nachkommen
  setze $t := t + 1$
`od`

Kasten 4: Grundalgorithmus eines GA

setze $t := 0$
initialisiere Population $P_0 \in \mathbb{B}^l$
bewerte die Individuen mittels f
`while` Terminierungskriterium nicht erfüllt `do`
  selektiere Individuen zur Rekombination gemäß ihrem Anteil an der Gesamtfitneß
  rekombiniere Individuen aus P_t mit Wahrscheinlichkeit $p_{crossover}$ (an rein zufälliger Stelle)
  mutiere diese Individuen mit Wahrscheinlichkeit $p_{mutation}$
  bewerte diese Individuen mittels f
  setze $t := t + 1$
`od`