Evolutionäre Algorithmen: Ein robustes Optimierkonzept

Günter Rudolph Hans–Paul Schwefel

Universität Dortmund
Fachbereich Informatik
Lehrstuhl für Systemanalyse
D–44221 Dortmund

1 Einführung

Bei der Betrachtung des seit etwa 4 Milliarden Jahren andauernden Evolutionsprozesses auf der Erde stellt man fest, daß viele der hervorgebrachten Formen gut, vielleicht sogar optimal an ihre Umwelt angepaßt sind. Folglich gibt es heute mit der Bionik eine Wissenschaft, die Lösungen aus der Natur für die Technik zu nutzen versucht und damit sehr erfolgreich ist. Warum also sollte eine Nachahmung von Prinzipien der biologischen Evolution nicht auch zu besseren Optimierverfahren führen ?

Diese Frage wurde Anfang der sechziger Jahre von verschiedenen Forschergruppen aufgeworfen. So wurden in Deutschland von Rechenberg [1] und Schwefel [2] die *Evolutionsstrategien* (ES), in den USA von Holland [3] die *Genetischen Algorithmen* (GA) sowie von Fogel u.a. [4] *Evolutionary Programming* (EP) entwickelt.

Diese unabhängig voneinander entstandenen Entwicklungen, die heute unter dem Sammelbegriff *Evolutionäre Algorithmen* (EA) zusammengefaßt werden, haben die gemeinsame Eigenschaft, daß sie bewußt Prinzipien der biologischen Evolution nachahmen, um sie im Sinne von Optimierregeln einzusetzen.

Die Hauptanwendungsgebiete von EA sind Optimierungsprobleme, für die keine Spezialverfahren bekannt sind oder bei denen traditionelle Optimierverfahren aufgrund von Nichtlinearitäten, Diskontinuitäten und Multimodalität versagen. Die Eigenschaft der Robustheit von EA liegt darin begründet, daß zum einen keinerlei Annahmen über das gestellte Problem getroffen werden und daß zum anderen stets mit einer Menge von zulässigen Lösungen (*Population* von Lösungen) gearbeitet wird.

Dadurch werden – im Gegensatz zu herkömmlichen Verfahren – zur gleichen Zeit mehrere Wege zum Optimum ausprobiert, wobei auch noch Informationen über die verschieden Wege (durch *Vererbung*) ausgetauscht werden. Auf diese Weise ist das Wissen über das Problem in der Population verteilt, wodurch eine frühzeitige Stagnation während der Optimierung verhindert wird.

Die Verwendung des Populationskonzeptes eröffnet natürlich auch zahlreiche Möglichkeiten zur Verwendung von Parallelrechnern: EA sind auf allen Parallelrechnerarchitekturen in nahezu beliebiger Skalierung implementierbar. Auf diese Weise können nun Probleme angegangen werden, die zuvor aus Zeitgründen nicht behandelt worden sind.

Solche praktische wie auch theoretische Fragestellungen werden auf den *International Conferences on Genetic Algorithms* (seit 1985), den *Parallel Problem Solving from Nature*–Konferenzen (seit 1990) und der *Annual Conference on Evolutionary Programming* (seit 1992) behandelt, wobei die ersten beiden Konferenzen im zweijährigen Turnus abgehalten werden.

2 Evolutionäre Algorithmen

Evolutionäre Algorithmen werden dazu eingesetzt, mathematische Optimierungsprobleme der Form

$$\min\{f(x) \,:\, x \in \mathcal{M}\} \tag{1}$$

zumindest approximativ zu lösen. Dabei bezeichnet $f : \mathcal{M} \to \mathbb{R}$ die Zielfunktion und $\mathcal{M}$ den zulässigen Bereich. Obwohl die drei Entwicklungen GA, ES und EP eine Reihe von Unterschieden in ihrer expliziten Ausprägung besitzen, so ist ihr konzeptuelles doch Schema gleich: Allgemein kann ein *Individuum* $a \in \mathcal{A}$ durch das Tupel $a = (x, s_1, \ldots, s_r) \in \mathcal{M} \times \mathcal{S}_1 \times \ldots \times \mathcal{S}_r$ mit $r \geq 0$ beschrieben werden, wobei $x \in \mathcal{M}$ eine zulässige Lösung und $s_1, \ldots, s_r$ Strategieparameter repräsentieren. Jedem Individuum $a \in \mathcal{A}$ wird über eine Funktion $F : \mathcal{A} \to \mathbb{R}$ ein *Fitnesswert* zugeordnet. Die Zielfunktion ist stets in die Fitnessfunktion eingearbeitet, und in vielen Fällen ist sie sogar mit ihr identisch.

Eine *Population* besteht aus einem Tupel von μ Individuen und wird während jeder Iteration unter Verwendung von sogenannten genetischen Operatoren verändert: Während bei der *Mutation* jedes Individuum zufällig verändert wird, wird bei der *Rekombination* aus zwei zufällig gewählten Individuen ein neues Indivuum erzeugt. Die *Selektion* schließlich entscheidet anhand der Fitness der

BEISPIEL:

Die zu minimierende Zielfunktion sei

$$f(x) = \sum_{i=1}^{2} \left\{ x_i^2 + 10\,[1 - \cos(2\pi x_i)] \right\}$$

mit dem zulässigen Bereich $\mathcal{M} = [-5, 5]^2$.
Ein Individuum könnte wie folgt kodiert sein:

$$a = (x, \sigma) \in \mathbb{R}^2 \times \mathbb{R}_+ = \mathcal{A} \; .$$

Als Fitnessfunktion verwendet man

$$F(a) = F(x, \sigma) = f(x) \; .$$

Der Objektvariablenvektor des Nachommen wird durch die Vorschrift

$$x_{t+1} = x_t + z_t$$

erzeugt, wobei z_t ein normalverteilter Zufallsvektor mit Erwartungswert $(0, 0)^T$ und Kovarianzmatrix $\sigma^2\, I$ ist.

Abbildung 1: Beispiel

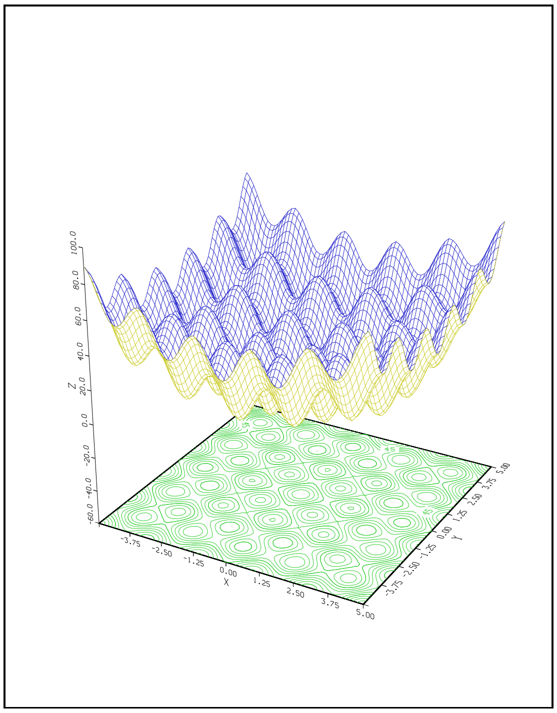

Abbildung 2: 3–D Plot des Beispielproblems

neu erzeugten Individuen, welche von diesen in die nächste Iteration (Generation) übernommen werden. Abbildung 3 skizziert das allgemeine Schema eines EA.

Die verschiedenen Ausprägungen der EA unterscheiden sich in der Repräsentation der Individuen und in den genetischen Operatoren. Diese Unterschiede werden im folgenden kurz umrissen. Für eine ausführlichere Darstellung im Fall der Parameteroptimierung sei auf [5] verwiesen.

2.1 Genetische Algorithmen

Der sogenannte kanonische GA (vgl. etwa [6]) benutzt zur Repräsentation von Individuen Binärstrings der Länge l, also Elemente des $\mathbb{B}^l$. Diese Repräsentation ist für Probleme der pseudobooleschen Optimierung direkt geeignet. Möchte man andere Problemklassen bearbeiten, so ist eine Reformulierung des Problems auf den Binärraum notwendig. Grundsätzlich ist man jedoch nicht auf die Binärrepräsentation beschränkt: Ein Individuum kann durchaus durch komplexere Datenstrukturen wie Listen, Bäume etc. dargestellt werden. In diesem Fall müssen die genetischen Operatoren auf die jeweilige Repräsentation angepaßt werden. Wir werden uns hier jedoch auf den Fall $\mathcal{A} = \mathcal{M} = \mathbb{B}^l$ beschränken.

Eine Population besteht aus μ Individuen. Für jedes Bit im Binärstring eines Individuums wird durch ein Zufallsexperiment entschieden, ob das Bit invertiert wird. Dazu wird eine auf dem Intervall $[0, 1]$ gleichmäßig verteilte Zufallszahl gezogen. Falls die Realisation kleiner gleich einer fest vorgegebenen Mutationswahrscheinlichkeit $p_m \in (0, 1)$ ist, dann wird das Bit invertiert. Die Rekombination (auch: *crossover*) wird wie folgt durchgeführt: Man ziehe zufällig zwei Individuen aus der Population sowie eine Zufallszahl $c \in \{1, 2, \ldots, l-1\}$. Ein neues Individuum erhält die ersten c Positionen seines Bitstrings von einem, die restlichen vom anderen Elternteil. Aus den so neu erzeugten und anschließend mutierten Individuen wird dann mit dem Selektionsoperator die neue Population ausgewählt. Dabei werden die Individuen mit einer zur ihrer Fitness proportionalen Häufigkeit selektiert. Der gesamte Zyklus wird wiederholt, bis ein frei wählbares Terminierungskriterium erfüllt ist.

Initialisiere Population $P_0 \in \mathcal{A}^\mu$
Bewerte die Individuen anhand $F(.)$
Setze $t := 0$
`repeat`
    Rekombiniere neue Individuen aus P_t
    Mutiere die neuen Individuen
    Bewerte die Individuen anhand $F(.)$
    Selektiere die neue Population P_{t+1}
    Setze $t := t + 1$
`until` Terminierungskriterium erfüllt

Abbildung 3: Skizze eines Evolutionären Algorithmus

2.2 Evolutionsstraegien

Evolutionsstrategien wurden ursprünglich zur experimentellen Optimierung eingesetzt, d.h., daß ein reales Objekt tatsächlich verändert wurde, bevor seine Güte bzw. Fitness vermessen wurde. Die Mutationen waren natürlich diskreter Natur. Erst bei der Realisierung des Algorithmus als Computerprogramm wurde zur Mutation eine Normalverteilung benutzt.

Der typische Suchraum für Evolutionsstrategien ist somit der n–dimensionale reelle Raum: $\mathcal{M} = \mathbb{R}^n$. Ein Individuum $a = (x, \sigma, \omega) \in \mathbb{R}^n \times \mathbb{R}_+^n \times (-\pi, \pi]^{n(n-1)/2}$ besteht aus dem Objektvariablenvektor x und aus einem Satz von Strategieparametern, die die Mutationsverteilung steuern. Bei der Mutationsverteilung handelt es sich zumeist um eine n–dimensionale Normalverteilung mit dem Nullvektor als Erwartungswert und einer Kovarianzmatrix, die durch eine geeignete Transformation aus den Strategieparamtern gebildet wird. Die Strategieparameter werden ebenfalls mutiert und rekombiniert. Eine detaillierte Beschreibung findet amn in [5].

Eine Möglichkeit zur Rekombination besteht darin, zwei Individuen zufällig auszuwählen, um dann etwa aus deren Objektvariablenvektoren x_a und x_b durch $x_a + (x_b - x_a)\,\xi$ den Objektvariablenvektor des neuen Individuums zu erzeugen. Dabei bezeichnet ξ einen auf $[0, 1]^n$ oder $\{0, 1\}^n$ gleichverteilten Zufallsvektor. Auf ähnliche Weise werden die Strategieparameter rekombiniert.

Durch Rekombination und Mutation werden aus μ Individuen $\lambda > \mu$ neue Individuen erzeugt ($\mu \leq \lambda$). Bei der Selektion werden entweder nur aus den λ Nachkommen oder aus allen $\mu{+}\lambda$ Individuen die μ besten für die nächste Population ausgewählt, bevor der gesamte Zyklus wiederholt wird.

2.3 Evolutionary Programming

Mitte der sechziger Jahre schlugen L.J. Fogel u.a. [4] eine Methode zur Evolution von endlichen Automaten vor, die sie *Evolutionary Programming* nannten. Diese Automaten wurden zur Lösung von Vorhersageproblemen eingesetzt. Ein Individuum bzw. Automat wurde durch seine Zustandsübergangstabelle repräsentiert, die der Mutation unterworfen wurde. Als Fitnessmaß diente die Anzahl der richtig vorhergesagten Symbole des Automaten. Diese Methode wurde von D.B. Fogel [7] für die Parameteroptimierung im $\mathbb{R}^n$ angepaßt. Bis auf einen etwas anderen Selektionsmechanismus ist der EP–Algorithmus mit Evolutionsstrategien nahezu identisch, weshalb wir auf eine nähere Beschreibung verzichten möchten. Eine detaillierte Darstellung findet sich in [5].

3 Abschließende Bemerkungen

Durch die Erschwinglichkeit von Parallelrechnern verschiedenster Architektur sind die vorgestellten historischen Versionen von EAs mehrfach modifiziert worden. Zumeist ist der Selektionsmechanismus verändert worden, während die genetischen Operatoren Mutation und Rekombination unverändert blieben. Diese parallelen Varianten haben jedoch nichts von der Robustheit ihrer Vorgänger verloren. Im Gegenteil: Die Empirie zeigt, daß gerade parallele Versionen eine verbesserte Lösungsqualität aufweisen.

Literatur

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

[2] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie.* Interdisciplinary systems research; 26. Birkhäuser, Basel, 1977.

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

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

[5] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation,* 1(1):1–23, 1993.

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

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