Optimierung technischer Systeme mittels Evolutionsstrategien - ein Standardverfahren in SIMUL_R

R. Ruzicka
SIMUTECH
Hadikgasse 150, A-1140 Wien

Kurzfassung

Die Methode der Evolutionsstrategien wurde erst in den letzten Jahren als Optimierungsverfahren allgemein angewendet und anerkannt. Für den Bereich der Optimierung mit Simulationsmodellen als Gütefunktion lassen sich spezielle, aber doch allgemein verwendbare Algorithmen angeben. Die Simulationssprache SIMUL_R stellt als eines ihrer Optimierungsverfahren den Befehl GENOPT zur Verfügung, der diese Algorithmen einfach verwendbar implementiert. Anhand zweier Beispiele wird GENOPT erläutert und die Optimierung mittels Evolutionsstrategien vorgeführt. Den Abschluß bilden eine Abwegung von Vor- und Nachteilen der Evolutionsstrategien und einige Tips zu ihrer erfolgreichen Verwendung.

1. Einleitung

Die sogenannten *Evolutionsstrategien* (ES) - oft auch als Teilbereich der *Genetischen Verfahren* angesehen - sind eine der Natur abgeschaute Methode, die in den 60er und 70er Jahren der Mathematik und Optimierung zugänglich gemacht, jedoch erst in jüngster Zeit in mehreren Arbeiten für die Simulationstechnik entdeckt wurde ([RECH73], [DEJO75], [SCHW81], [GOLD89]).

In vielen Fällen werden ES bei der Optimierung von Zeit-diskreten (Warteschlangen-) Modellen verwendet; insbesondere dann, wenn die zu optimierenden Parameter nur einzelne, diskrete Werte annehmen können.

Die folgenden Algorithmen zeigen einen der Wege auf, ES auch für kontinuierliche Modelle zu verwenden, bei denen die Parameter aus beliebigen reellen Intervallen stammen können.

ES - angewandt zur Optimierung (Minimierung) einer reellen Funktion **f** mehrerer Parameter $\mathbf{p_i}$ - wird im folgenden, auf die Problemstellungen in der Simulation technischer Systeme zugeschnitten, vorgestellt. Für eine tiefergehende, formale Definition sei hier etwa auf [SCHW93] verwiesen.

2. Das Verfahren

Die Nomenklatur ist der Biologie entlehnt: *Individuen* entstammen einer *Population* und werden durch *Chromosomen* beschrieben (die Parametersätze), die Chromosomen bestehen aus *Allelen* (den einzelnen Parametern). Die *Fitneß* **F** eines Individuums wird als Funktion der Gütefunktion gemessen.

F ist hierbei als Dichtefunktion über den möglichen Werten von **f** anzusehen. Es wird im folgenden nicht ganz einfach ein Parametersatz *sicher* gewählt, wenn er "gut" ist, sondern nur mit einer bestimmten Wahrscheinlichkeit. Das bedeutet, daß auch "schlechtere" Parametersätze noch

zum Zug kommen können. Sinnvollerweise wird man aber doch danach trachten, daß "bessere" mit größerer Wahrscheinlichkeit gewählt werden.

Die Methode selbst läßt sich in 6 Schritten darstellen (siehe Abbildung 1).

1. *Initialisierung*	wähle eine Anfangspopulation mit **v** Individuen $\mathbf{I_1}$ bis $\mathbf{I_v}$ und den Parametersätzen ($\mathbf{p_{11}}$,..,$\mathbf{p_{1n}}$) bis ($\mathbf{p_{v1}}$,..,$\mathbf{p_{vn}}$)
2. *Bewertung*	ermittle die Fitneß der Individuen $\mathbf{F}(\mathbf{f}(\mathbf{p_{i1}},..,\mathbf{p_{in}}))$
3. *Selektion*	wähle **s** (<**v**) Individuen aus, wobei **I**i mit Wahrscheinlichkeit $\mathbf{F_i}$ selektiert wird
4. *Rekombination*	wähle Paare von selektierten Individuen und generiere durch zufällige Auswahl der Allelen neue Individuen; das neue Individuum enthält also pro Allele einen Wert des ersten oder des zweiten Eltern-Individuums
5. *Mutation*	wähle einzelne selektierte Individuen und generiere durch zufällige Auswahl und Änderung einer Allele jeweils ein neues Individuum
6. *Generierung*	Die selektierten, rekombinierten und mutierten (zusammen **v** Stück) bilden die neue Population, weiter mit Schritt 2, falls die Terminierungsbedingung noch nicht erfüllt ist.

Abbildung 1: Ein Algorithmus zur Optimierung mittels Evolutionsstrategien.

Bezogen auf die Simulation bedeutet dies, daß pro Optimierungsschritt **v** Simulationsläufe zur Ermittlung der Gütefunktion **f** durchgeführt werden und das Modell von **n** Parametern abhängt.

3. Optimierung in SIMUL_R - der Befehl GENOPT

Die Simulationssprache SIMUL_R ([RUZI88], [RUZI89], [RUZI93]) besitzt den Befehl *GENOPT* (eine sogenannte *userdefined function*), der die Methode der Evolutionsstrategien implementiert.

Pro Auswertung von **f** wird i.a. ein Simulationslauf durchgeführt. Dieser Simulationslauf kann - ohne Modell - ganz einfach auch zu einer einzelnen Funktion degenerieren, oder umgekehrt auch eine komplexe Auswertung umfassen (z.B. mehrere Simulationsläufe oder Vergleiche einer Funktion mit Meßwerten).

GENOPT erwartet folgende Argumente (einige beschreiben direkt auswertbare Ausdrücke; andere Ausdrücke, die dynamisch zur Laufzeit ausgewertet werden):

- die Anzahl der Parameter **n**
- die Größe der Population **v**
- die Gütefunktion **f** (beliebiger Ausdruck, der z.B. auf Modelldaten zugreift)
- Parameternamen $\mathbf{p_1}$ bis $\mathbf{p_n}$
- ein Feld [**v**,**n**] der Populationsparameter (zu Beginn: Anfangswerte; am Ende: Resultat)
- die Anzahl der Selektionen, Rekombinationen und Mutationen (feste Werte oder diskrete Verteilungen)
- die Fitneßfunktion **F** (als beliebiger Ausdruck)
- eine Mutationsfunktion (abhängig vom jeweiligen Parameter und dessen Vorwert)
- eine Terminierungsfunktion (beliebiger Ausdruck)

Optional werden die Populationsnummer, der Mittelwert und der beste Wert von **f** über der Population gesamplet.

GENOPT stellt ein sehr allgemeines Instrumentarium zur Variierung des Algorithmus zur Verfügung. So kann dynamisch bestimmt werden, wie viele Selektionen, Rekombinationen und Mutationen durchgeführt werden sollen, und wie mutiert wird.

Im allgemeinsten Fall kann man unter Verwendung der Runtime Funktion *unif_dis* (liefert eine gleichverteilte Zufallszahl) zufällige Anzahlen für Selektionen, Rekombinationen und Mutationen, bzw. eine zufällige Mutation beschreiben.

4. Ein Beispiel - Laufkatze

Anhand eines modellmäßig einfachen Beispiels, das typische Fragestellungen bei der Anwendung der ES zur Optimierung technischer Systeme aufzeigt, wird im folgenden der Befehl GENOPT und dessen Parameter erläutert.

Gegeben sei die Laufkatze eines Verladekrans [GRÄF86]. Diese soll sich entlang einer horizontalen Strecke von einem Punkt *A* zu einem Punkt *B* bewegen. An der Laufkatze ist an einer Stange eine Last befestigt (siehe Abbildung 2.).

Gesucht ist der Verlauf der Steuerfunktion (Spannung *u* des Antriebsmotors), der die Laufkatze in Richtung *x* von *A* nach *B* bewegt.

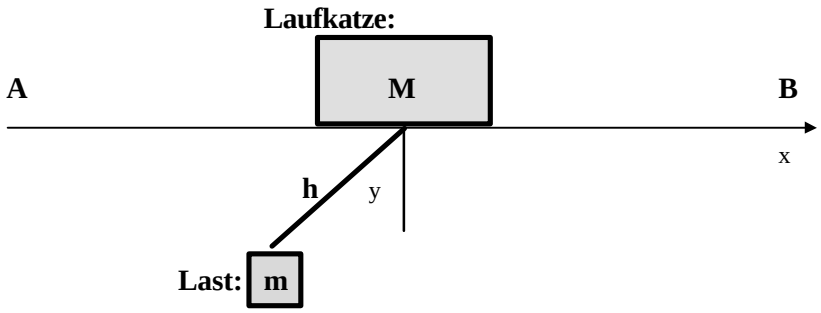

Abbildung 2. Laufkatze mit Last.

Die Laufkatze und die Last müssen zu Beginn (*A*) und am Ende (*B*) der Bewegung in Ruhe sein (Geschwindigkeit in *x*-Richtung sowie Winkelgeschwindigkeit in *y*-Richtung gleich 0).

Nach - für die Optimierungsaufgabe nicht bedeutenden - Vereinfachungen, wie z.B. der Linearisierung der Winkelfunktionen und Normierungen ergibt sich das System:

$$\mathbf{x'' = -\,c * y + u * b_u} \quad \text{mit} \quad \mathbf{x'(0) = x(0) = 0}$$
$$\mathbf{y'' = -\,a * y + u * b_u} \quad \text{mit} \quad \mathbf{y'(0) = y(0) = 0}$$

mit den zusätzlichen Randbedingungen

$$\mathbf{x(tend) = xend} \quad \text{(xend in Punkt } B \text{ bekannt)}$$
$$\mathbf{x'(tend) = y(tend) = y'(tend) = 0}$$

und der Nebenbedingung

$$\mathbf{-1 \leq u \leq 1}$$

Die Randwerte und die Nebenbedingung werden als Straffunktionen angesetzt.

Im folgenden werden zwei Ansätze für die Steuerfunktion untersucht: als Streckenzug mit gesuchter Auslenkung und als Bang-Bang Steuerung mit gesuchten Schaltintervallen. In beiden Fällen wird die Steuerfunktion durch eine SIMUL_R Tabellenfunktion approximiert.

4.1 Die Steuerfunktion als Streckenzug

Im ersten Fall wird die Steuerfunktion ganz allgemein durch eine stückweise lineare Funktion über der Zeit approximiert, deren Stützstellenwerte die Parameter der Optimierungsaufgabe sind.

Bei GENOPT werden in diesem Fall 12 Parameter (die Endzeit *tend* und die Stützstellenwerte *u0* bis *u10*) und 16 Individuen (jeweils 4 Selektionen, 2 Rekombination, 10 Mutationen) gewählt. **f** ist *tend*+Strafterme (die Strafterme für die Nebenbedingung erhöhen **f** um einen großen Wert, falls *ui* nicht im Intervall [-1,1] liegt). Die Fitneßfunktion ist $\mathbf{F}(\mathbf{f})=1/\mathbf{f}^2$ (wird von SIMUL_R auf eine Dichte normiert). Die Anfangswerte der Parameter sind Zufallszahlen. Die Mutation verändert die Parameter *ui* durch Multiplikation mit einer Zufallszahl aus [-2,-0.5] bzw. [0.5,2].

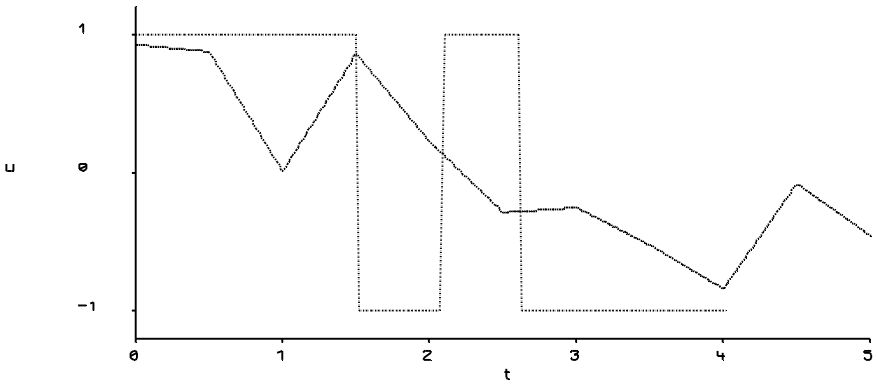

Abbildung 3. Linearinterpolierte Tabellenfunktion und "optimale" Lösung aus 4.2

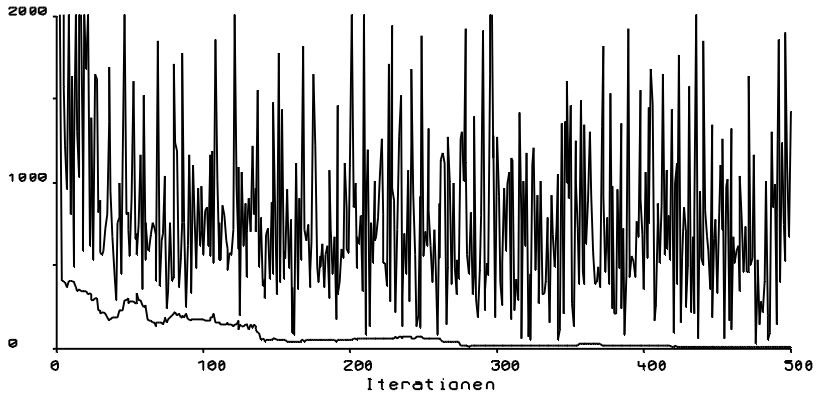

Abbildung 4. Iterationsverlauf bei linearinterpolierter Tabellenfunktion

Nach 500 Populationen erhält man eine Steuerfunktion, die nicht allzuweit vom Optimum entfernt ist (Zeit ca. 5 in Relation zum Optimum *tend*=4). Die Form der Steuerkurve läßt die Tendenz einer fallenden Steuerfunktion erkennen, jedoch noch nicht die optimale Bang-Bang Lösung (siehe Abbildung 3.). Insgesamt liefert dieser Ansatz ein nicht sehr zufriedenstellendes Resultat. Eine Verbesserung ist z.B. durch eine stückweise konstante Funktion oder mehr Stützstellen erreichbar. Der Iterationsverlauf und die Werte der Gütefunktion (die untere Kurve ist der beste Wert jeder Generation, die obere der Mittelwert) sind in Abbildung 4. dargestellt.

4.2 Die Steuerfunktion als Bang-Bang-Funktion

Beim zweiten Ansatz wird das Wissen, daß es sich bei der optimalen Funktion um eine Bang-Bang-Funktion handelt, ausgenützt: die Umschaltzeitpunkte zwischen voll Beschleunigen und voll Bremsen (genauer: die Intervalle dazwischen) werden als Parameter gewählt. Die Steuerfunktion wird als SIMUL_R Treppentabellenfunktion angesetzt.

GENOPT werden 4 Optimierungsparameter (die 4 Umschaltzeitpunkte) und ansonsten dem ersten Ansatz entsprechende Werte übergeben.

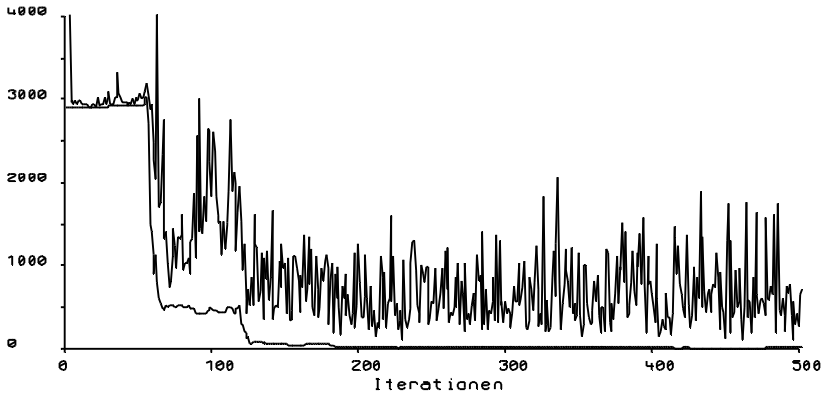

Abbildung 5. Iterationsverlauf bei Bang-Bang-Steuerung

Hier gelangt man nach 500 Iterationen schon bis auf 0.25% Genauigkeit zum Optimum. Die Kurve des Iterationsverlaufs zeigt ein interessantes Verhalten: der Algorithmus gelangt recht rasch zu einem "guten" Ergebnis, obwohl er einige Zeit in einem lokalen Minimum verharrt (siehe Abbildung 5.). Der Steuerungsverlauf ist in Abbildung 3. (Bang-Bang) gezeichnet.

5. Zusammenfassung

Zusammenfassend läßt sich feststellen, daß GENOPT eine sehr allgemeine Variante der genetischen Algorithmen implementiert und deshalb ein sehr allgemeines Optimierungswerkzeug darstellt. Auf Parallelrechnern werden die Modelle von GENOPT parallel abgearbeitet.

Allgemeine Vorteile der ES bei Anwendung auf technische Systeme sind:

- durch das Prinzip der Zufälligkeit bleibt man meistens nicht in lokalen Minima "hängen"
- an die Modelle müssen keine Stetigkeits- und Glattheitsforderungen gestellt werden

- kontinuierliche und diskrete Parameter sind mischbar
- man gelangt sehr "rasch" an "gute" Parameter, d.h. die genetischen Algorithmen sind relativ unempfindlich gegenüber der Wahl der Anfangswerte

Nachteile sind:

- keine speziellen algorithmischen Vorkehrungen für Rand- und Nebenbedingungen (eventuell Straffunktionen verwenden)
- Wann beendet man die Iteration ?

Als Praxistips seien hier angeführt:

- Verwende bei kontinuierlichen Modellen häufiger Mutationen als etwa bei diskreten (Warteschlangen-) Modellen.
- Ist man einmal nahe genug dem Optimum, so benötigen ES oft viel Rechenzeit um noch wesentliche Verbesserungen zu erreichen. Deshalb ist es in solchen Fällen vorteilhaft, wenn möglich, mit den bisher berechneten Parametern in ein anderes Optimierungsverfahren einzusteigen (z.B. Gradientenverfahren).
- Gebräuchliche Terminierungsbedingungen sind: maximale Anzahl der Iterationen vorgeben; Abbruch, wenn sich die Mittelwerte der Gütefunktionen nicht mehr wesentlich ändern; Abbruch, wenn die Varianzen innerhalb einer Population wieder zu steigen beginnen.

6. Literatur

[DEJO75] K.A.De Jong: *An analysis of the behaviour of a class of genetic adaptive systems*, Dissertation, University of Michigan, 1975

[GOLD89] D.E.Goldberg: *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, Reading, MA, 1989

[GRÄF86] M.Gräff, I.Troch: *Die Berechnung von optimalen Steuerungen für dynamische Prozesse durch Parameteroptimierung*, Bericht Nr. 2/1986, Abt. Regelungsmathematik, Hybridrechen- und Simulationstechnik des Instituts E114, Technische Universität Wien, 1986

[RECH73] I.Rechenberg: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart 1973

[RUZI88] R.Ruzicka: *SIMUL_R - eine Simulationssprache mit speziellen Befehlern zur Modelldarstellung und -analyse*, Informatik Fachberichte 179, Proceedings des 5. Symposiums Simulationstechnik, Springer, Aachen, 1988

[RUZI89] R.Ruzicka: *Environments for SIMUL_R*, 3rd European Simulation Congress, Proceedings, Edinburgh, 1989

[RUZI93] R.Ruzicka: *SIMUL_R PARALLEL - Hardware-in-the-loop Simulation mit Transputern unter Windows*, Fortschritte in der Simulationstechnik, Band 6, Proceedings des 8. Symposiums Simulationstechnik, Vieweg, Berlin, 1993

[SCHW81] H.P.Schwefel: *Numerical Optimization of Computer Models*, Wiley, Chichester, 1981

[SCHW93] T.Bäck, U.Hammel, H.P.Schwefel: *Modelloptimierung mit evolutionären Algorithmen*, Fortschritte in der Simulationstechnik, Band 6, Proceedings des 8. Symposiums Simulationstechnik, Vieweg, Berlin, 1993