Genetische Algorithmen – Darstellung, Eigenschaften und eine Anwendung in der Statistik

S. Niermann
M.D. Jöhnk*

Diskussionspapier 238
ISNN 0949-9962

Zusammenfassung: In der vorliegenden Arbeit wird der Genetischer Algorithmus dargestellt, Anwendungsfelder in der Statistik skizziert und auf die Schätzung der Parameter einer Johnson-Verteilung angewandt. Hierbei wird das Konvergenzverhalten dieser Algorithmen in Abhängigkeit von der Mutationswahrscheinlichkeit, der durchschnittlichen Schrittweite bei Mutationen und der Wahl der Fitness-Funktion untersucht. Schließlich findet eine Visualisierung des Genetischen Algorithmus und seines Verhaltens unter Verwendung des Programmpaketes S-PLUS statt.

Abstract: In this paper the genetic algorithm is described, applications in the field of statistics are indicated and it is applied to the estimation of the parameters of a Johnson-type distribution. The convergence behaviour is analyzed with respect to the mutation probability, the kind of mutations and the fitness criterion chosen. Finally the algorithm and its behaviour is visualized with S-PLUS.

*Adresse der Autoren:
Institut für Quantitative Wirtschaftsforschung
Königsworther Platz 1
30167 Hannover
Fax-Nr. 0049 511 762-3923
niermann@mbox.iqw.uni-hannover.de

1 Problemstellung

Verfahren und Algorithmen, die nach dem Vorbild biologischer Prozesse und Modelle konstruiert sind, erfreuen sich in der jüngeren Vergangenheit einer großen Beliebtheit in verschiedenen wissenschaftlichen Disziplinen.
Neben den Künstlichen Neuronalen Netzen sind in diesem Zusammenhang in erster Linie solche Algorithmen zu nennen, die ihr Vorbild in der Evolution haben. Diese werden auch Evolutorische Algorithmen genannt. Ihre prominentesten Vertreter sind die Genetischen Algorithmen. Sie sind in vielen Fällen ein geeignetes Instrument zur Lösung von Optimierungsproblemen.

Einer Vielzahl solcher Optimierungsprobleme begegnet man auch in der Statistik. Daher gibt es eine Reihe von Anwendungsfeldern, bei denen der Einsatz Genetischer[1] Algorithmen potentiell nützlich sein könnte:

- Wahl einer Funktion, die eine beobachtete Zeitreihe gut approximiert,
- Auswahl einer oder mehrerer Transformationen, die vor der Anwendung eines statistischen Verfahrens (z.B. Regression) durchgeführt wird,
- Schätzung von Parametern.

Neben der Lösung von Optimierungsproblemen sind Genetische Algorithmen geeignet, um die Interaktion von Individuen in Gruppen abzubilden. Individuen sind hierbei durch eine Strategie gekennzeichnet, die angibt, wie sich das Individuum in verschiedenen Situationen verhalten wird. Diese Strategien müssen sich dann in einem evolutorischen Umfeld bewähren.

Eine der ersten Untersuchungen dieser Art ist das sogenannte *Axelrod Tournament* (Axelrod 1984), bei dem untersucht wird, welche Verhaltensformen (kooperativ oder konfliktorientiert, verläßlich oder spontan) in verschiedenen sozialen Kontexten erfolgreich sind. Das Hauptaugenmerk liegt also hier auf der Analyse der Interaktion der Mitglieder der Population und den resultierenden gesellschaftlichen Strukturen. Unter ähnlichen Aspekten wurden Genetische Algorithmen auch schon zur Simulation von Finanzmärkten eingesetzt, um zu untersuchen ob, bzw. unter welchen Bedingungen Finanzmärkte effizient sind (Palmer et al. 1994).

In dieser Arbeit wird dem gegenüber die Fähigkeit Genetischer Algorithmen zur Lösung von Optimierungsproblemen betont. Neben der Darstellung der Funktionsweise Genetischer Algorithmen stehen in dieser Arbeit zwei Aspekte im Vordergrund.

- Einerseits sollen Genetische Algorithmen als Instrument bei der Bearbeitung statistischer Probleme ins Blickfeld gerückt werden.

[1] In der Fachliteratur zur Künstlichen Intelligenz werden die auf (Holland 1975) zurückgehenden Genetischen Algorithmen von den auf (Rechenberg 1973) zurückgehenden Evolutorischen Strategien unterschieden. Bei Genetischen Algorithmen wird der Aspekt der Weitergabe der Erbinformationen von 2 Eltern auf einen oder mehrere Nachkommen (Crossover) betont, wohingegen bei Evolutorischen Strategien der Fokus auf Mutation und Auslese gerichtet ist. Da bei dem in dieser Arbeit verwendeten Algorithmus das Crossover neben der Mutation stattfindet, wird dieser Algorithmus von den Autoren als Genetischer Algorithmus bezeichnet, wohl wissend, dass die binäre Kodierung der Erbinformation von Puristen als konstituierendes Element eines Genetischen Algorithmus aufgefasst wird.

- Andererseits soll aufgezeigt werden, inwieweit Ergebnisse, die mit Genetischen Algorithmen erzielt werden von der statistisch-formalen Implementierung des Genetischen Algorithmus abhängen.

Die Vorgehensweise ist daher wie folgt. In Kapitel 2 wird die Grundstruktur eines Genetischen Algorithmus dargestellt. Anhand eines einfachen Beispiels wird die Funktionsweise exemplarisch dargestellt. Daran schließt sich die Darstellung derjenigen statistischen Problemstellung an, die unter Verwendung eines Genetischen Algorithmus gelöst werden soll.

In Kapitel 4 wird der Genetische Algorithmus in S-PLUS, Version 3.4 implementiert. Hierbei soll dokumentiert werden, an welchen Stellen bei der Implementierung eines Genetischen Algorithmus für die Funktionsweise und das Verhalten des Genetischen Algorithmus sensible Entscheidungen getroffen werden.

Abschließend wird der Genetische Algorithmus auf das zu lösende Optimierungsproblem angewendet. Um Erkenntnisse über das Verhalten von Genetischen Algorithmen zu generieren, werden Simulationen durchgeführt.

2 Der Genetische Algorithmus

Die Grundidee eines Genetischen Algorithmus besteht darin, dass eine Population sich durch Weitergabe der Erbinformationen (Crossover), spontane Veränderungen bei der Vererbung (Mutation) und eine anschließende Selektion (Survival of The Fittest) dynamisch entwickelt.

Dieses Grundprinzip erwies sich als erfolgreich bei der Entwicklung von Lebewesen. Deshalb wurde dieses biologische Prinzip auf andere Klassen von Problemen angewandt.

Die mit den beiden oben dargestellten Anwendungsbereichen korrespondierenden Fragestellungen lauten damit:

- Wie ist die individuelle Fitness eines durchschnittlichen oder auch des besten Individuums einzuschätzen? Diese Fitness ist interpretierbar als die Fähigkeit, eine bestimmte Aufgabe zu lösen – also im klassischen Sinne zu überleben.

  Soll also ein Optimierungsproblem mit einem Genetischen Algorithmus gelöst werden, wird man das fitteste Individuum der Population zum Maßstab nehmen.

- Daneben kann von Interesse sein, wie die Entwicklung der Population im Zeitablauf ist. Werden die Individuen der Population im Laufe der Zeit immer uniformer? Nähern sich alle, einige oder keines der Individuen dem Optimum? Entstehen bestimmte topologische Strukturen?

Jedes Individuum ist durch bestimme Eigenschaften gekennzeichnet. Bei Genetischen Algorithmen ist jedes Individuum durch seine Gene eindeutig spezifiziert. Die Gene eines Individuums wiederum determinieren die Fitness.

Ein genetischer Algorithmus weist die folgende Grundstruktur auf:

1) Erzeuge eine Startpopulation
2) Erzeuge die nachfolgende Generation
   2a) Bestimme ein Heiratsschema
   2b) Weitergabe der Erbinformation der Eltern an die Kinder
      2bi) Crossover
      2bii) Mutation
3) Survival of the Fittest
4) FALLS ein Abbruchkriterium erfüllt ist
   DANN ENDE
   SONST Schritt 2)

Dieser Algorithmus und die individuelle Ausgestaltung desselben sollen zunächst anhand eines einfachen Beispiels erläutert werden. Es ist zu beachten, dass die Eigenschaften eines Genetischen Algorithmus wesentlich von dieser konkreten Ausgestaltung abhängt.

Betrachten wir das Problem der Schätzung der Parameter einer normalverteilten Zufallsvariable durch einen Genetischen Algorithmus.

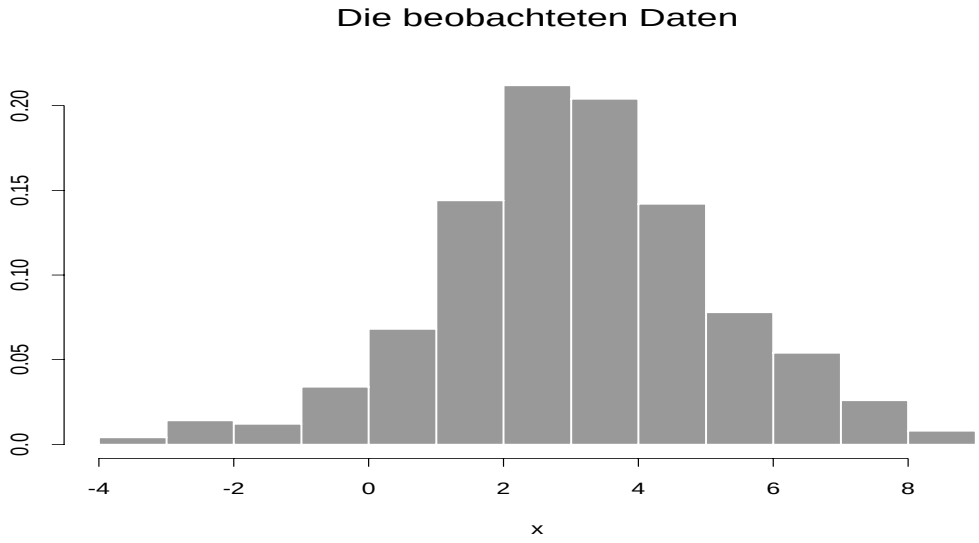


Abb. 1

Zu den hier dargestellten Daten sollen die Parameter so bestimmt werden, dass Modell und Daten gut übereinstimmen. Jedes Individuum der Population hat damit zwei *Gene* – Erwartungswert μ und Varianz σ^2, die das Erscheinungsbild determinieren.

1) Erzeuge die Startpopulation

   In dem Beispiel besteht die Population aus 4 Individuen, die jeweils 2 Gene haben. Als Alternative hierzu könnte die Erbinformation auch binär kodiert werden. Die Repräsentation der Erbinformationen durch einen zweielementigen numerischen Vek-

tor scheint uns jedoch die unmittelbarere und problemadäquate Darstellung zu sein. De Jong (1985) (zitiert nach Zbigniew 1996, S. 3) schrieb in diesem Zusammenhang:

> What should one do when Elements in the space to be searched are most naturally represented by more complex data structures such as arrays, trees, digraphs, etc. Should one attempt to 'linearize' them into a string representation or are there ways to creatively redefine crossover and mutation to work directly on such structures. I am not aware of any progress in this area.

Dieser - nicht explizit ausgesprochenen - Empfehlung folgend, werden die Erbinformationen durch einen numerischen Vektor (μ, σ^2) kodiert.

Individuum	1	2	3	4
Gene /Parameter	(0,1)	(1,3)	(3,5)	(3,1)

Natürlich erweist es sich für den Genetischen Algorithmus als günstig, wenn die Startwerte sich in der Nähe der optimalen Werte befinden.

2a) Bestimme ein Heiratsschema

Im einfachsten Fall werden Paare von Individuen zufällig ausgewählt, die dann miteinander verheiratet werden. In dem hier betrachteten Beispiel werden die Individuen 1 und 4 sowie die Individuen 2 und 3 miteinander verheiratet.

> Hinweis: Es existieren Varianten des Genetischen Algorithmus, bei denen bei der Verheiratung topologische Strukturen berücksichtigt werden. Individuen wählen dann Partner, die sich in der Nähe befinden. Es ergeben sich dann Subpopulationen – wie bei auseinanderdriftenden Kontinentem –, die gelegentlich untereinander vermischt werden. Durch diese Vorgehensweise soll vermieden werden, dass der Algorithmus in einem lokalen Minimum steckenbleibt. Diese Variante wird von Mühlenbein (1995) beschrieben.
>
> Eine andere Variante besteht darin, dass auch bei der Partnerwahl ein Fitness-Kriterium angewandt wird.

2bi) Crossover

Beim Crossover wird ein neuer Satz Gene aus den Erbanlagen der Eltern gebildet. In *The Hitch-Hiker´s Guide to Evolutionary Computation* (Heitkoetter 1993) liest sich die Beschreibung dieses Prozesses wie folgt:

> At the molecular level what occurs (wild oversimplification alert!) is that a pair of chromosomes bump into one another, exchange chunks of genetic information and drift apart. This is the recombination operation, which GA/GPers generally refer to as crossover because of the way that genetic material crosses over from one chromosome to another.

In unserem Beispiel erhält das Kind der Individuen 1 und 4 die kompletten Erbanlagen des Individuums 4, das zweite Kind erhält das *Gen* μ von Individuum 1 und das *Gen* σ^2 von Individuum 3. Damit erhalten die Kinder die Erbanlagen (3,1) und (1,5).

2bii) Mutation

In seltenen Fällen treten bei den Kindern Mutationen auf. Bei solchen Mutationen verändern sich die durch die Eltern vererbten Informationen – in diesem Fall Parameterwerte. Im Beispiel erhalte Kind 2 anstelle des Parameterwertes $\sigma^2 = 5$ den Parameterwert $\sigma^2 = 4$. Damit weisen die Kinder die Erbanlagen (3,1) und (1,4) auf.

> Hinweis: Die Wahrscheinlichkeit, mit der Mutationen auftreten, beeinflusst das Verhalten des Genetischen Algorithmus stark. Sie muss sorgfältig gewählt werden.
>
> Daneben ist zu spezifizieren, welche Parameterwerte die Kinder nach etwaigen Mutationen aufweisen. Bei numerischen Genen bietet sich an, dass sich der neue Wert als Realisation einer normalverteilte Zufallsvariable ergibt, die symmetrisch um alten Parameterwert ist. In diesem Fall hängt das Verhalten des Genetischen Algorithmus stark von der Varianz der gewählten Normalverteilung ab.
> Bei binären Genen mutiert das Gen zu der jeweils anderen möglichen Ausprägung.

3) Survival of the Fittest

Nun besteht die Population aus 6 Individuen. Für jedes dieser Individuen muss dann ein Fitness-Wert bestimmt werden. In diesem Beispiel ist ein Individuum umso fitter, je besser die Anpassung des durch die beiden Parameter determinierten Verteilungsmodells an die Daten ist.

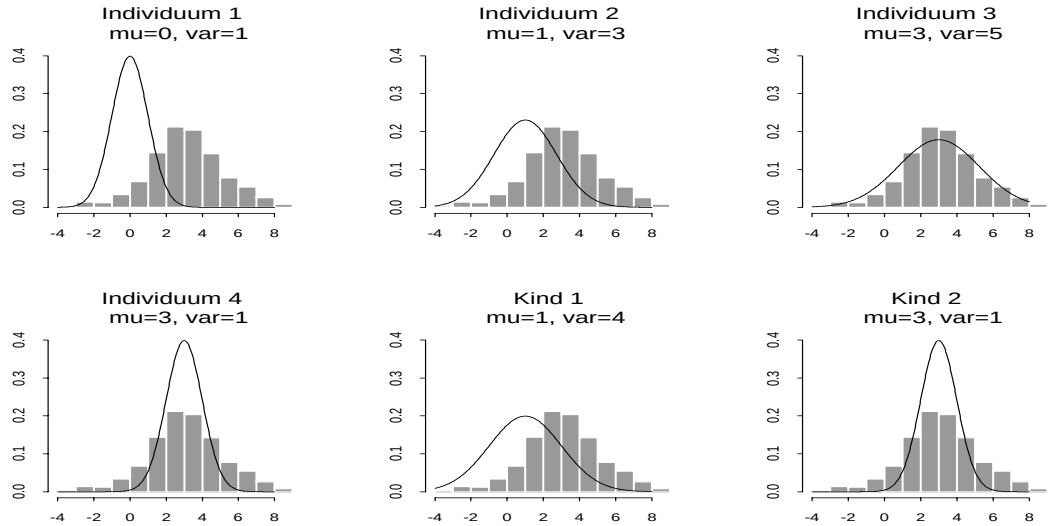


Abb. 2

In dem vorliegenden Beispiel weisen die Individuen 1 und 2 die schlechtesten Fitness-Werte auf und werden entsprechend aus dem Datensatz entfernt. Bei dieser Art der

Modellierung bleibt die Zahl der Individuen in der Bevölkerung von Generation zu Generation konstant.

4) Der Genetische Algorithmus bricht ab, falls eine vorgegebene Anzahl von Iterationsschritten erreicht wurde oder ein sonstiges Abbruchkriterium erfüllt ist. Im Allgemeinen wird ein solches Abbruchkriterium beinhalten, dass die Fitnesswerte des oder der Individuen mit der besten Fitness sich nicht mehr nennenswert ändern.

3 Das statistische Problem: Die Schätzung der Parameter einer vierparametrigen Johnson-Verteilung

Das Verteilungssystem von Johnson entsteht durch ein System von Transformationen aus einer Basisverteilung. JOHNSON (1949) wählt als Basisverteilung die Standardnormalverteilung (Kennbuchstabe S). Zunächst wird eine standardnormalverteilte Zufallsvariable Z linear transformiert: $U = \mu + \sigma \cdot Z$. Daran anschließend wird die Variable Y durch eine von drei nichtlinearen Transformationen

$$Y = \frac{exp(U)}{1 + exp(U)} \quad \Leftrightarrow \quad U = log\left(\frac{Y}{1 \Leftrightarrow Y}\right) \tag{1}$$

$$Y = exp(U) \quad \Leftrightarrow \quad U = ln(X) \tag{2}$$

$$Y = sinh(U) = \frac{exp(U) \Leftrightarrow exp(\Leftrightarrow U)}{2} \quad \Leftrightarrow \quad U = arsinh(Y) = ln(Y + \sqrt{(1 + Y^2)}) \tag{3}$$

eingeführt, die abschließend noch einmal linear transformiert wird zu $X = \xi + \lambda \cdot Y$.

Bei der erstaufgeführten Transformation wird Y auf das Intervall $(0, 1)$ beidseitig begrenzt, sie wird daher mit dem Index B gekennzeichnet. Bei der letztgenannten Transformation ist Y unbegrenzt, sie erhält deshalb den Index U. Die verbleibende Transformation führt zu einem linksseitig begrenzten Bereich für Y, sie wird daher mit dem Index L versehen.

Damit erhält man ein System von Verteilungen: die S_U, S_B und S_L-Verteilung.

Die Parameter μ und σ beeinflussen die Form der Verteilung, die Parameter ξ und λ sind konstruktionsgemäß Linearparameter.

In späteren Veröffentlichungen betrachteten JOHNSON und TADIKAMALLAH (1992) statt der Standardnormalverteilung die logistische Verteilung (Kennbuchstabe L). Damit ergeben sich die L_U-, die L_B- und die L_L-Verteilung.

In der vorliegenden Arbeit sollen die Parameter einer L_U-Verteilung unter Verwendung eines Genetischen Algorithmus geschätzt werden.

4 Die Implementierung des Genetischen Algorithmus

Die Implementierung des hier verwendeten Genetischen Algorithmus erfolgte mit dem Programmpaket S-PLUS Ver. 3.4 unter Zuhilfenahme des an der Universität Bielefeld von Prof. Dr. Peter Naeve und Dr. Peter Wolf entwickelten REVWEB-Systems. Zur Erhöhung

der Übersichtlichkeit sind einige Blöcke von S-PLUS-Befehlen in den Anhang aufgenommen worden.

Um Simulationen durchführen zu können, in denen das Verhalten des Genetischen Algorithmus analysiert werden soll, muss zunächst ein Datensatz generiert werden. In diesem Fall werden $n = 100$ Beobachtungswerte verwendet. Diese werden generiert, indem 100 standard-logistisch-verteilte Zufallszahlen ausgewählt werden, die dann den oben beschriebenen Transformationen unterzogen werden. Zuvor werden die gewünschten Parameterwerte erfragt und die gewünschte Anzahl der Generationen erfragt.

1 ⟨* 1⟩≡

```
z<-rlogis(100)
cat("Parameter mu:\n")
mu<-scan(n=1)
cat("Parameter sigma:\n")
sigma<-scan(n=1)
cat("Parameter xi:\n")
xi<-scan(n=1)
cat("Parameter lambda:\n")
lambda<-scan(n=1)
cat("Anzahl der Generationen\n")
gen.anzahl<-scan(n=1)
bester<-1

x<-xi+lambda*(sinh(mu+sigma*z))
```

This definition is continued in chunks 2 and 3.
Root chunk (not used in this document).

Als Fittness-Kriterium wird die Summe der quadrierten (vertikalen) Abstände der empirischen Verteilungsfunktion von der theoretischen Verteilungsfunktion verwendet.

Falls die empirische Verteilungsfunktion gezeichnet wird, werden als Verteilungsfunktionswerte die folgenden *Plotting Positions* verwendet: Der Wert der empirischen Verteilungsfunktion des i-ten Beobachtungswertes aus der Rangwertreihe ist gegeben durch $\hat{F}(x_{(i)}) = \frac{2 \cdot i - 1}{2 \cdot n}$.

Diese Werte werden in einem Vektor h gespeichert.

2 ⟨* 1⟩+≡

```
h<-(1:length(x))/length(x)-1/(2*length(x))
par(mfrow=c(2,1))
hist(x,prob=T)
title("Histogramm der Daten")
plot(sort(x),h,xlab="x",ylab="FDach(x)",ylim=c(0,1))
title("Empirische Verteilungsfunktion \n der Daten")
⟨erzeuge Startpopulation 4⟩
par(mfrow=c(3,1))
```

Die Startpopulation besteht aus $n_p = 100$ Individuen. Hierbei können alle Individuen dieselben Startwerte haben. Dies ist eine Frage der Implementierung bzw. der Eigenschaften des Verfahrens. Bei der Implementierung Genetischer Algorithmen wird allgemein empfohlen, dass

- die Startwerte des GA in der Nähe der optimalen Lösung liegen sollten,
- ein möglichst großer Bereich der möglichen Parameterwerte abgedeckt werden sollte.

Zu Demonstrationszwecken werden diese Empfehlungen hier nicht befolgt. Statt dessen werden allen Individuen dieselben Parameterwerte zugeordnet:

$\sigma_{start} = 1, \quad \mu_{start} = 0, \quad \xi_{start} = 1 \quad \text{und} \quad \lambda_{start} = 0.$

Da jedes Individuum aus $n_G = 4$ Genen besteht, kann die gesamte Population durch eine (4×100)-Matrix repräsentiert werden.

Bei der Erzeugung der Startpopulation werden gleichzeitig die Vektoren, in denen die Schätzwerte $(sigma.dach, mu.dach, xi.dach, lambda.dach)$, die durchschnittliche Fitness $(mean.fitness)$ und die Variabilität der Fitness $(var.fitness)$ in der Population gespeichert werden, initialisiert.

Die klassischen Schritte eines genetischen Algorithmus werden wiederholt, bis das Abbruchkriterium erf"ullt ist.

3 ⟨* 1⟩+≡

```
for(i in 1:gen.anzahl){
⟨bestimme Heiratsschema 5⟩
⟨Crossover 6⟩
⟨Mutation 7⟩
⟨Bestimme Fitness 8⟩
⟨Survival of the fittest 9⟩
⟨berechne Ergebnisse 10⟩
print(i)
}
cat("\n Weiter mit RETURN \n")
scan(n=1)
⟨zeige Ergebnisse 11⟩
```

Im nächsten Schritt wird ein Heiratsschema bestimmt. Hierzu werden 50 Individuen zufällig ausgewählt. Diese werden mit den verbeibenden 50 Individuen (in der Reihenfolge ihres Auftretens) verheiratet. Jeweils mit Wahrscheinlichkeit 0.5 erhalten die Kinder jedes der Gene dann vom Vater bzw. von der Mutter – bezogen auf jedes einzelne Gen.

Die so entstandenen Kinder werden dann an die Populationsmatrix herangehängt.

Schließlich finden bei einigen der Individuen Mutationen statt. Welche das sind, steht in der Matrix *welche* (Wert 1) und die Schrittweite ergibt sich als Realisation einer normalverteilten Zufallsvariable. Die Eigenschaften des Genetischen Algorithmus werden wesentlich von dem Anteil der Mutationen und der durchschnittlichen Schrittweite bei Mutationen abhängen. Zusätzlich wird sichergestellt, dass das fitteste Individuum keiner Mutation unterliegt und damit sicher in die nächste Generation gelangt.[2]

Meist werden Genetische Algorithmen auf der Basis binärer Vektoren implementiert. Dabei ist dann klar, wie eine Mutation wirkt: Eine 0 mutiert zu einer 1 und umgekehrt. Bei numerischen Genen ist dies nicht so eindeutig. Welchen Wert hat etwa ein Gen mit der Ausprägung 0.3 nach einer Mutation? In dieser Arbeit wurde die folgende Herangehensweise gewählt: Der neue Wert ergibt sich als Ausprägung einer normalverteilten Zufallsvariable mit dem Erwartungswert 0.3 und einer Standardabweichung s. Damit schwankt der neue Wert zufällig um den alten Wert, wobei die durchschnittliche Schrittweite durch den Parameter s festgelegt wird. Es stellt sich dann unmittelbar die Frage, wie s gewählt werden sollte und welchen Einfluß die Wahl von s auf die Eigenschaften des Genetischen Algorithmus hat.

Die Fitness wird als Summe der quadrierten Abstände der empirischen von der geschätzten Verteilungsfunktion operationalisiert. Auch an dieser Stelle sind alternative Operationalisierungen denkbar, von denen die Eigenschaften des Genetischen Algorithmus abhängen werden.

Damit haben wir nun einen Vektor mit 150 Fitness-Werten. Daraus sollen nun diejenigen 100 Individuen selektiert werden, die die besten Fitness-Werte aufweisen.

Wird dieses wiederbelebbare Papier[3] unter REVIVE aufgerufen, so lässt sich die Güte der Anpassung des fittesten Individuums Generation für Generation nachverfolgen. In der oberen Grafik (Abbildungen 3 und 4) ist die durchschnittliche Fitness aller Individuen (gemessen in Summe der quadrierten Abstände der empirischen von der theoretischen Verteilungsfunktion) dargestellt. In der mittleren Grafik ist die Varianz der Fitnesswerte aller Individuen der Population dargestellt. Sie gibt an, ob die Population eher homogen oder eher inhomogen ist. In der unteren Grafik werden schließlich die empirische und die theoretische Verteilungsfunktion in einem Diagramm dargestellt, um die Güte der Anpassung überprüfen zu können.

[2] Im Jargon der Evolutorischen Algorithmen bezeichnet man dies als *Überleben einer Elite*.

[3] Hinweise hierzu in Anhang B.

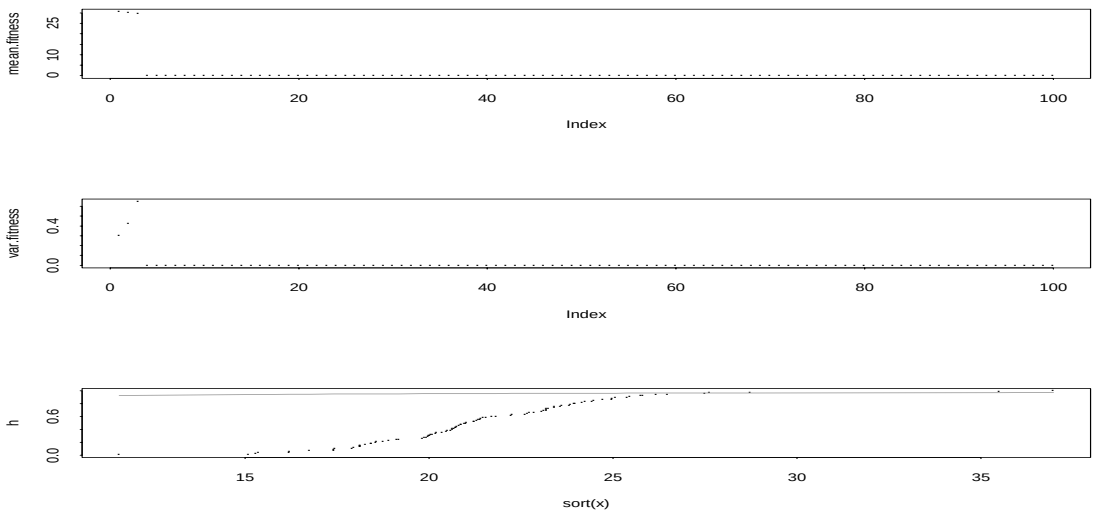


Abb. 3

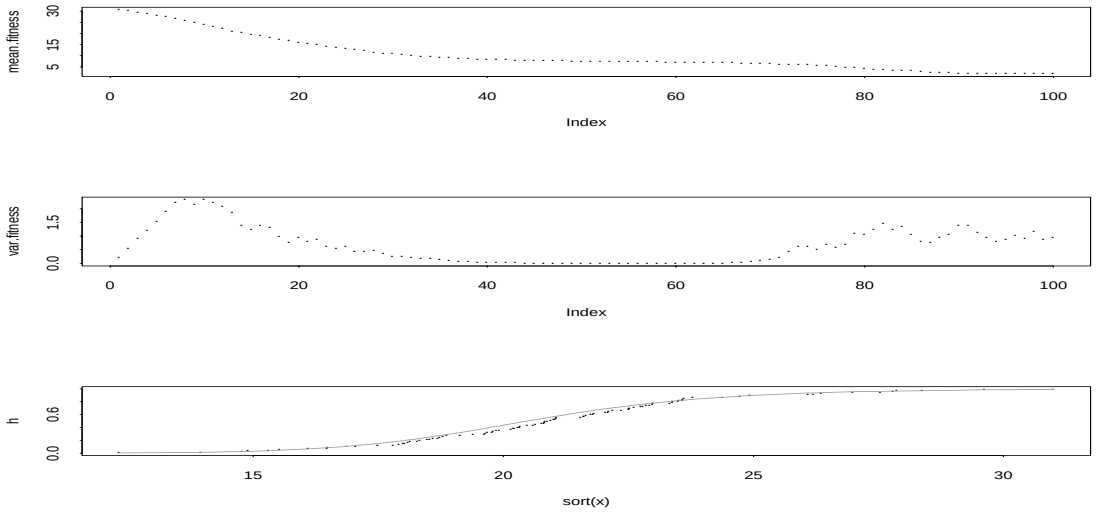


Abb. 4

Während die theoretische Verteilungsfunktion in Abbildung 3 nach 3 Generationen noch sehr weit von der empirischen Verteilungsfunktion entfernt liegt, ergibt sich nach 100 Generationen in Abbildung 4 eine gute Anpassung. In der oberen Grafik der Abbildung 4 ist zu erkennen, dass die durchschnittliche Fitness (ein kleiner Abstand entspricht einer hohen Fitness) der Individuen in der Population von Generation zu Generation zugenommen hat.

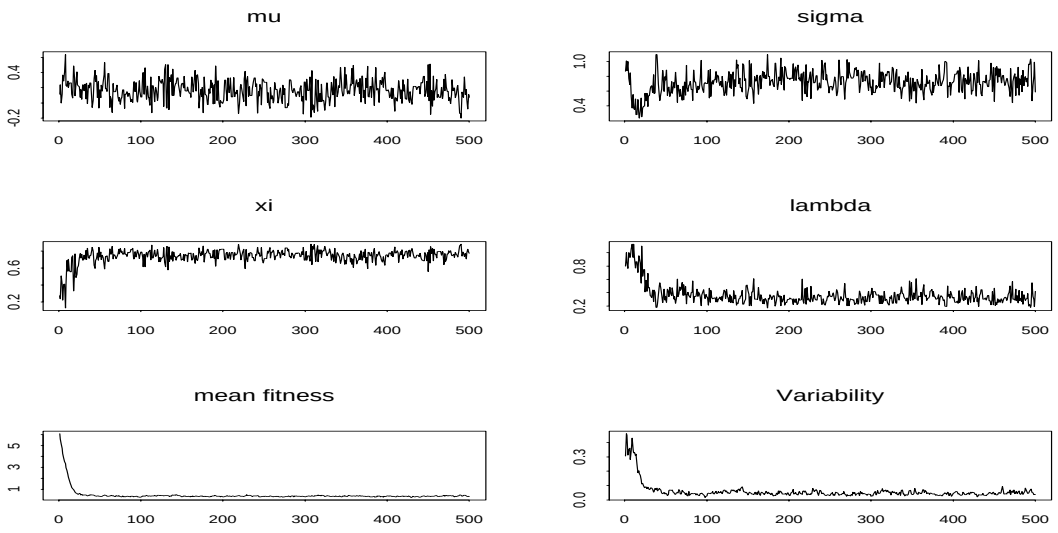
Die Varianz der Fitness-Werte steigt zunächst an – Ausgangspunkt war ja auch eine homogene Population –, nimmt dann wieder ab, um nach ca. 70 Generationen erneut deutlich anzuwachsen. Zu diesem Zeitpunkt zeigten sich auch deutliche Verbesserungen bei der Anpassung.

5 Simulationsergebnisse

Das oben dargestellte Programm wurde verwendet, um Simulationen durchzuführen, die die Eigenschaften eines Genetischen Algorithmus zu untersuchen. In den Simulationen wird nach jedem Generationswechsel das Individuum mit dem besten Fitness-Wert – also der besten Anpassung – bestimmt und dessen Parameterwerte festgehalten. Diese werden als Schätzer für die Parameter interpretiert. Auf diese Weise erhält man die Zeitpfade der Parameterschätzer. Daneben wird der durchschnittliche Fitness-Wert aller Individuen in der Population bestimmt. Da wir hier mit einem Abstandsmaß als Fitness-Kriterium arbeiten, entspricht ein geringer **M**ittlerer **Q**uadratischer **A**bstand (MQA) einer hohen Fitness.

Schließlich wird die Varianz dieses Abstandsmaßes bezogen auf die Individuen in der Population als Heterogenitätsmaß bestimmt. Da in der ersten Generation alle Individuen dieselben Startwerte aufweisen, hat diese Varianz zunächst den Wert Null.

In einer Simulation mit einer Mutationswahrscheinlichkeit $p = 0.5$ und einer mittleren Schrittweite $s = 0.1$ ergibt sich das folgende Bild. Die Anzahl der Generationen ist fest und umfaßt 500 Generationen.



In diesem Fall konvergiert der Algorithmus nicht. Es ist deutlich zu erkennen, daß nach etwa 40 Generationen keine Verbesserung mehr erzielt wird und die Parameterschätzer trotzdem recht stark schwanken.

Variiert man nun die Mutationswahrscheinlichkeit und/oder die mittlere Schrittweite, zeigt sich, daß die Schätzwerte konvergieren, wenn p und s klein gewählt werden. Dann konvergiert der Algorithmus allerdings erst recht spät. Dies kann den nachfolgenden Grafiken entnommen werden.

Im Lichte dieser Ergebnisse drängt sich die Idee auf, die Mutationswahrscheinlichkeit und die mittlere Schrittweite mit zunehmender Anzahl an Generationen zu verringern. Bezeichnen g die Generation und G die Anzahl der Generationen insgesamt, könnte die Mutationswahrscheinlichkeit p_g in der g'ten Generation entsprechend der Funktion

$$p_g = 0.5 \cdot exp(\frac{\Leftrightarrow 5 \cdot g}{G})$$

gewählt werden.

Die mittlere Schrittweite s_g könnte entsprechend operationalisiert werden:

$$s_g = 0.1 \cdot exp(\frac{\Leftrightarrow 5 \cdot g}{G}).$$

Simulationen mit diesen Setzungen und der Verwendung geeigneter Startwerte haben gezeigt, dass etwa 100 Generationen ausreichend sind, um zu guten Ergebnissen zu kommen. Bei jeder dieser 100 Generationen müssen 150 theoretische Verteilungsfunktionen mit der empirischen Verteilungsfunktion verglichen werden. Das sind 15000 Vergleiche. Würde man bei einer vierparametrigen Verteilung ein Suchgitter verwenden, würden bei nur 20 Werten pro Variable insgesamt schon $20^4 = 160000$ Vergleiche durchgeführt werden müs sen. Damit ist ein Genetischer Algorithmus auch unter Aufwandsbetrachtungen effizienter als etwa ein auf Suchgittern beruhendes Verfahren.

Obwohl Genetische Algorithmen im allgemeinen als *Black-Box*-Verfahren aufgefaßt werden, ist es also durchaus nicht unerheblich, wie die Mutationswahrscheinlichkeit und die mittlere Schrittweite gewählt werden. Diese beiden Parameter wurden hier exemplarisch ausgewählt, um zu demonstrieren, daß es nützlich ist, den *schwarzen Kasten* auszuleuchten, um bessere Ergebnisse zu erzielen.

Ein weiterer Ansatzpunkt für Untersuchungen stellt das Fitness-Kriterium selbst dar. Wenn anstelle des Mittleren Quadratischen Abstandes der empirischen von der theoretischen Verteilungsfunktion der Mittlere Absolute Abstand oder entsprechend des Vorschlags von ROUSSEEUW (1984) den Median der absoluten Abstände als Fitness-Kriterium verwendet wird, erhält man eine robustifizierte Variante eines Genetischen Algorithmus.

Literatur:

Axelrod, Robert: The Evolution of Cooperation, Basic Books, New York 1984.

Heitkoetter, Joerg, ed. (1993): The Hitch-Hiker´s Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ), Usenet: comp.ai.genetic. Available via anonymous ftp from rtfm.mit.edu in pub/usenet/news.answers/ai-faq/genetic/part?.

Holland, J.H.: Adaption in Natural and Artificial Systems, University of Michigan Press, Ann harbor, 1975.

Johnson, N.L. (1949) Systems of frequency curves generated by methods of translation, in: Biometrika, Vol. 36, S. 149-76.

Johnson, N.L. und P.R. Tadakimalla: Translated Families of Distributions, in: Balakrishnan, N. (Hrsg.): Handbook of the Logistic distribution, Decker, New York 1992.

Mühlenbein, Heinz: Genetische Algorithmen und Evolutionstheorien - Auf der Suche nach verschollenen Schätzen, GMD-Spiegel 2/95, 1995.
URL: http://set.gmd.de/AS/gmdsp/muehlen.html.

Palmer, R.G., W.B. Arthur, J.H. Holland, B. LeBaron und P. Taylor: Artificial Economic Life: A simple Model of a Stockmarket. in: Physika D, 1994 (Vol. 75), S. 264-274.

Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog Verlag, stuttgart 1973.

Rousseeuw, P.J.: Least Median of Squares Regression, in: Journal of the American Statistical Association, 1984 (Vol. 79), S. 871-880.

Zbigniew, M.: Genetic Algorithms + Data Structures = Evolution Programs, Springer Verlag, 3.Auflage, Berlin 1996.

A Der S-Plus-Code

4 ⟨*erzeuge Startpopulation* 4⟩≡
```
theta<-matrix(rep(c(0,1,0,1),100),nrow=4)
sigma.dach<-rep(0,gen.anzahl)
mu.dach<-rep(0,gen.anzahl)
xi.dach<-rep(0,gen.anzahl)
lambda.dach<-rep(0,gen.anzahl)
mean.fitness<-rep(0,gen.anzahl)
var.fitness<-rep(0,gen.anzahl)
fitness<-rep(0,150)
```

This code is used in chunk 2.

5 ⟨*bestimme Heiratsschema* 5⟩≡
```
vaeter<-sample(1:100,50)
muetter<-(1:100)[-vaeter]
```

This code is used in chunk 3.

6 ⟨*Crossover* 6⟩≡
```
auswahl.v<-matrix(rbinom(200,1,.5),nrow=4)
auswahl.m<-matrix(rep(1,200),nrow=4)-auswahl.v
theta.kinder<-theta[,vaeter]*auswahl.v+theta[,muetter]*auswahl.m
theta.gross<-cbind(theta,theta.kinder)
```

This code is used in chunk 3.

7 ⟨*Mutation* 7⟩≡

```
welche<-matrix(rbinom(600,1,.5),nrow=4)
welche[,bester]<-0
mutation<-matrix(rnorm(600,0,sd=0.1),nrow=4)
theta.gross<-theta.gross+welche*mutation
print(dim(theta.gross))
```

This code is used in chunk 3.

8 ⟨*Bestimme Fitness* 8⟩≡

```
for(j in 1:150){
hilf1<-exp((asinh((sort(x)-theta.gross[3,j])/theta.gross[4,j])-theta.gross[1,j])/
theta.gross[2,j])
F.modell<-hilf1/(1+hilf1)
fitness[j]<-sum((F.modell-h)^2)
}
```

This code is used in chunk 3.

9 ⟨*Survival of the fittest* 9⟩≡

```
auswahl<-order(fitness)[1:100]
print(length(auswahl))
theta<-theta.gross[,auswahl]
fit.neu<-fitness[auswahl]
bester<-(1:100)[order(fit.neu)[1]]
hilf1<-exp((asinh((sort(x)-theta.gross[3,bester])/theta.gross[4,bester])
-theta.gross[1,bester])/theta.gross[2,bester])
F.modell.best<-hilf1/(1+hilf1)
```

This code is used in chunk 3.

10 ⟨*berechne Ergebnisse* 10⟩≡

```
mean.fitness[i]<-mean(fit.neu)
var.fitness[i]<-var(fit.neu)
plot(mean.fitness)
plot(var.fitness)
plot(sort(x),h)
points(sort(x),F.modell.best,type="l",col=2)
mu.dach[i]<-theta[1,bester]
sigma.dach[i]<-theta[2,bester]
lambda.dach[i]<-theta[4,bester]
xi.dach[i]<-theta[3,bester]
print(c(mu.dach[i],sigma.dach[i],lambda.dach[i],xi.dach[i]))
```

This code is used in chunk 3.

11 ⟨*zeige Ergebnisse* 11⟩≡

```
cat("\n mu.dach:",round(mu.dach[gen.anzahl],6))
cat("\n sigma.dach:",round(sigma.dach[gen.anzahl],6))
cat("\n xi.dach:",round(xi.dach[gen.anzahl],6))
cat("\n lambda.dach:",round(lambda.dach[gen.anzahl],6))
cat("\n Best Fitness:",round(fit.neu[bester],6))
hilf1<-exp((asinh((sort(x)-xi)/lambda)-mu)/sigma)
F.modell<-hilf1/(1+hilf1)
mqf<-sum((F.modell-h)^2)
cat("\n MQF bei wahren Parametern:",mqf,"\n")


par(mfrow=c(3,2))
plot(mu.dach,type="l")
title("mu")
plot(sigma.dach,type="l")
title("sigma")
plot(xi.dach,type="l")
title("xi")
plot(lambda.dach,type="l")
title("lambda")
plot(mean.fitness,type="l")
title("Mean Fitness")
plot(var.fitness,type="l")
title("Var Fitness")
```

This code is used in chunk 3.

B REVWEB

Dieses Papier ist als wiederbelebbares Papier konzipiert. Bentutzer können den in dieser Arbeit dargestellten Genetischen Algorithmus unter S-PLUS zum Leben erwecken und mit selbst gewählten Parametersetzungen ablaufen lassen.

Voraussetzung hierfür ist

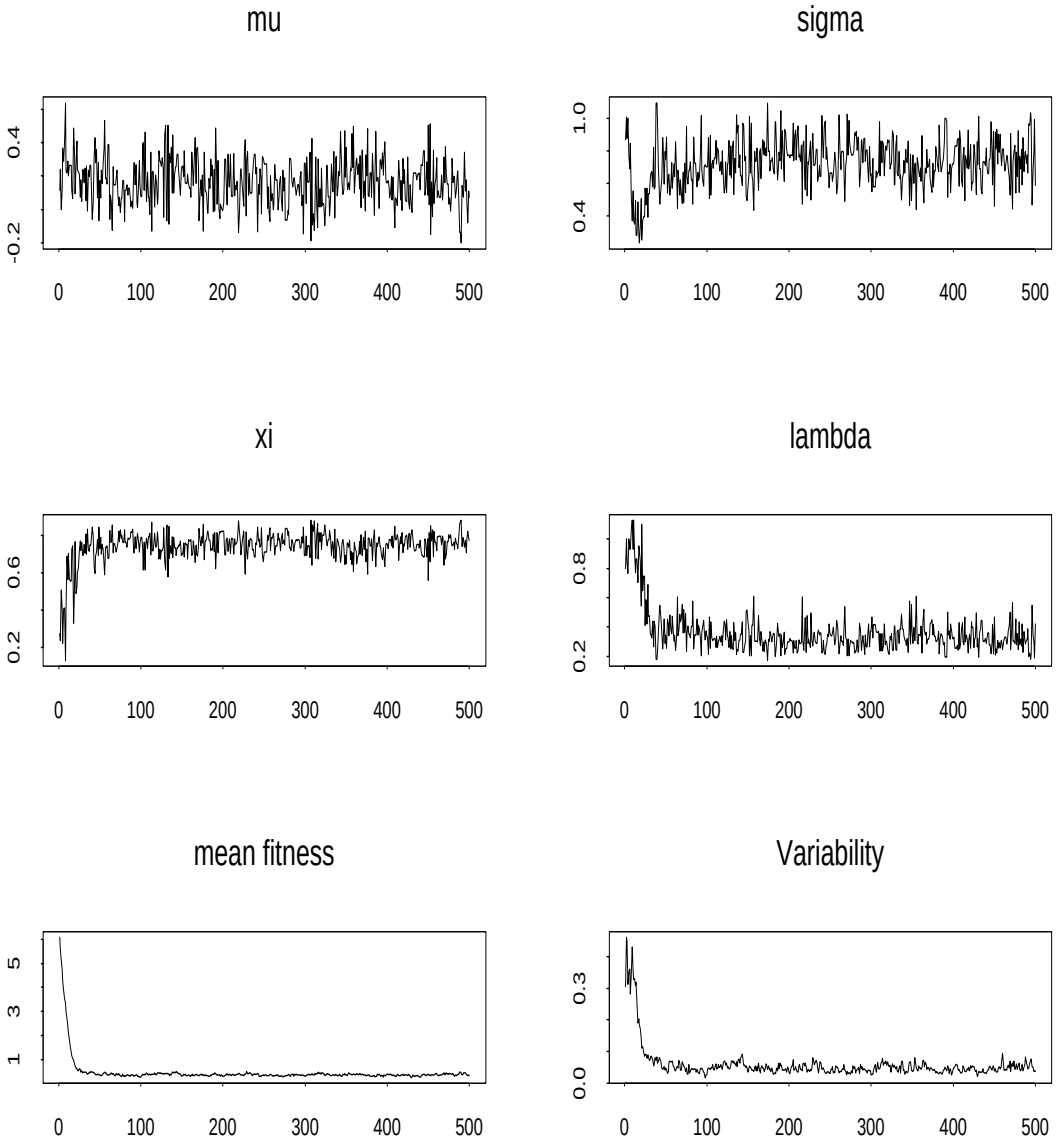
- die Implementierung des REVWEB-System unter S-PLUS,
- die Verfügbarkeit der Datei *genalg.rev*, aus der mit REVWEB u.a. die Dateien *genalg.tex* (dieses Dokument) und *genalg.sch* (ähnlich einer Batch-Datei für S-PLUS) erzeugt werden können.

Informationen über das REVWEB-System und das System selbst sind erhältlich unter: http://www.wiwi.uni-bielefeld.de/ naeve/software/revweb/revweb.html.

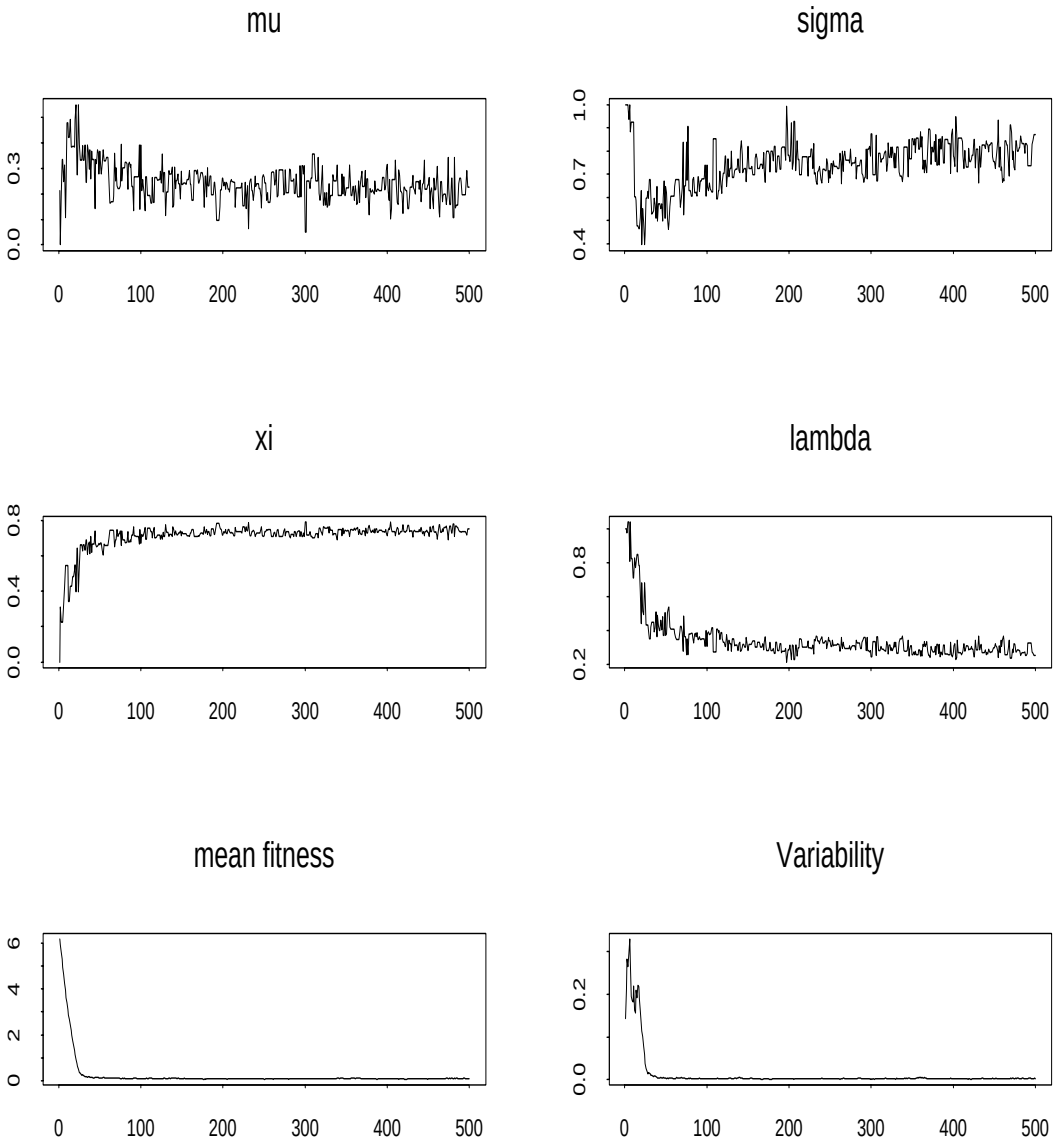
Die Datei *ganalg.rev* erhalten Sie auf Anfrage (niermann@mbox.iqw.uni-hannover.de).

C Die grafische Darstellung der Simulationsergebnisse

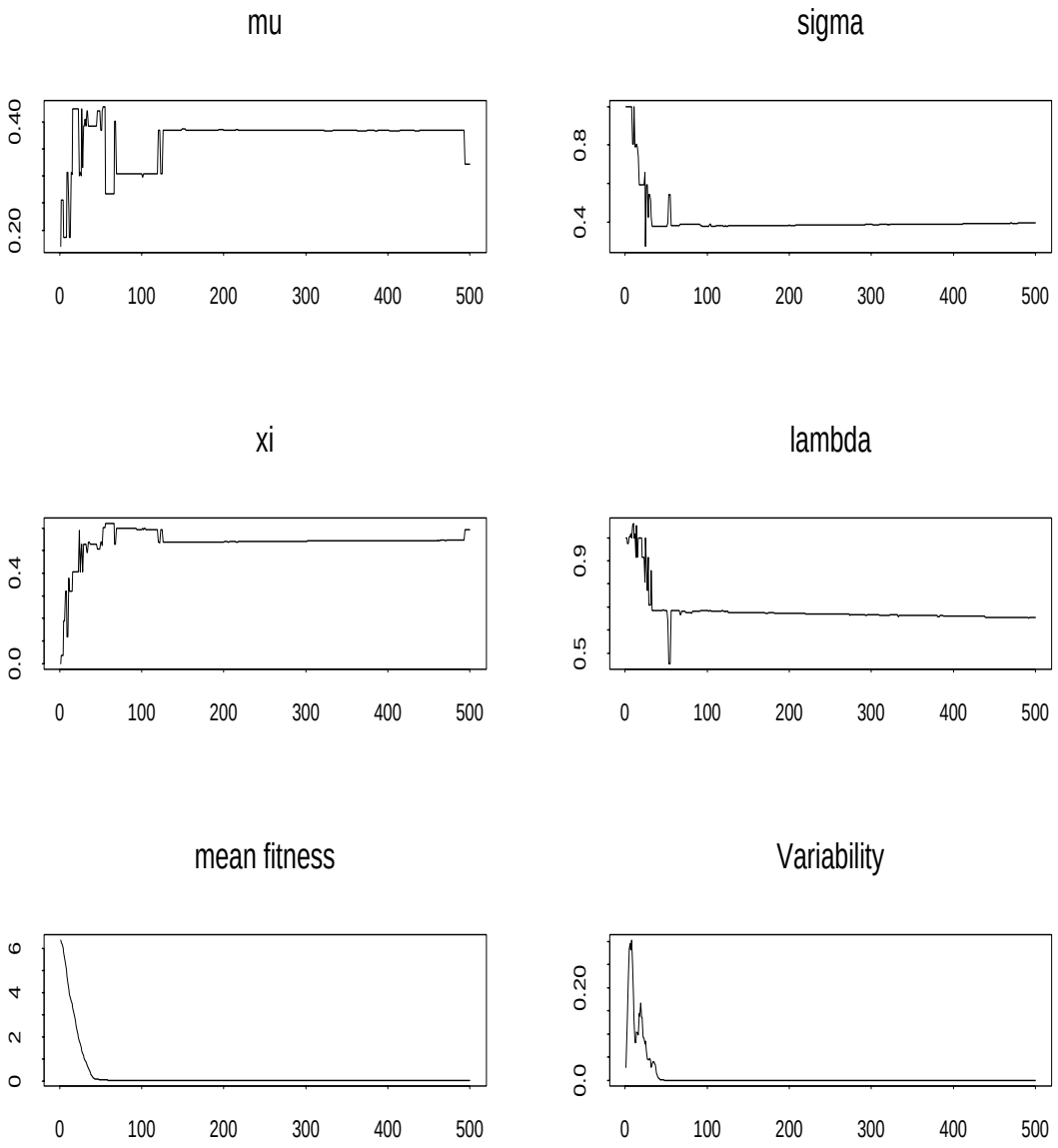
Mutationswahrscheinlichkeit $p = 0.5$,
mittlere Schrittweite $s = 0.1$



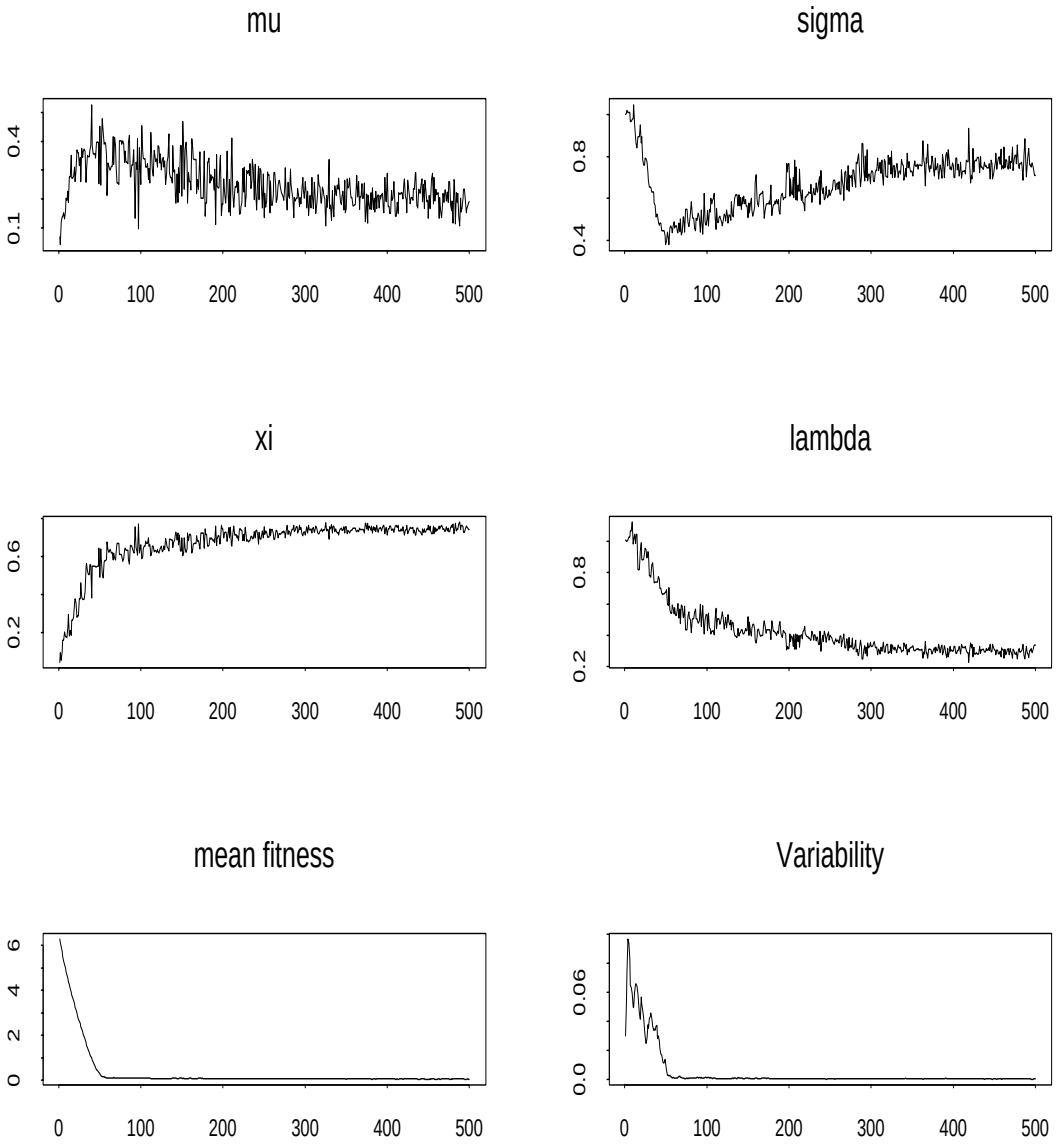
Mutationswahrscheinlichkeit $p = 0.2$,
mittlere Schrittweite $s = 0.1$



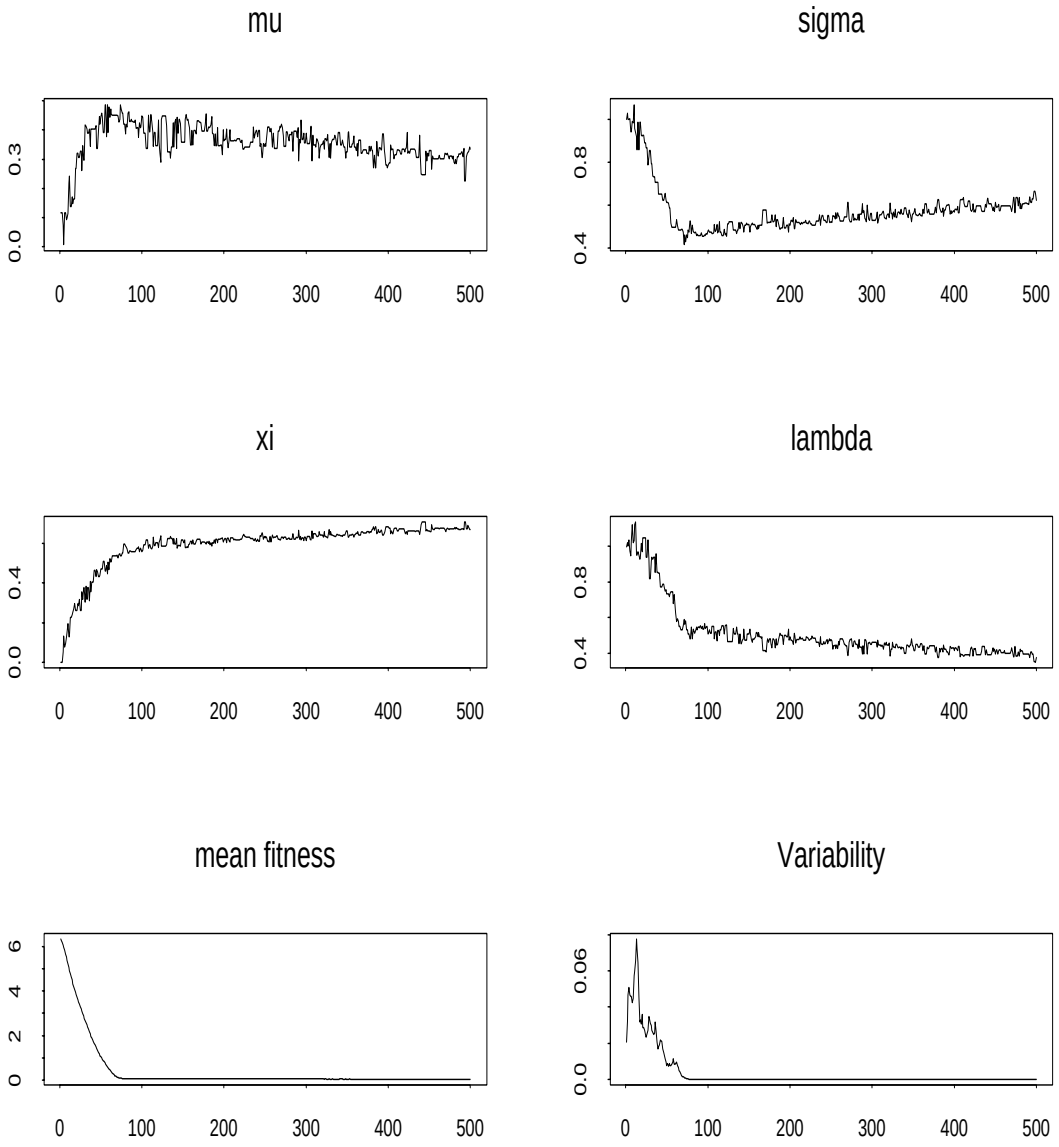
Mutationswahrscheinlichkeit $p = 0.05$,
mittlere Schrittweite $s = 0.1$



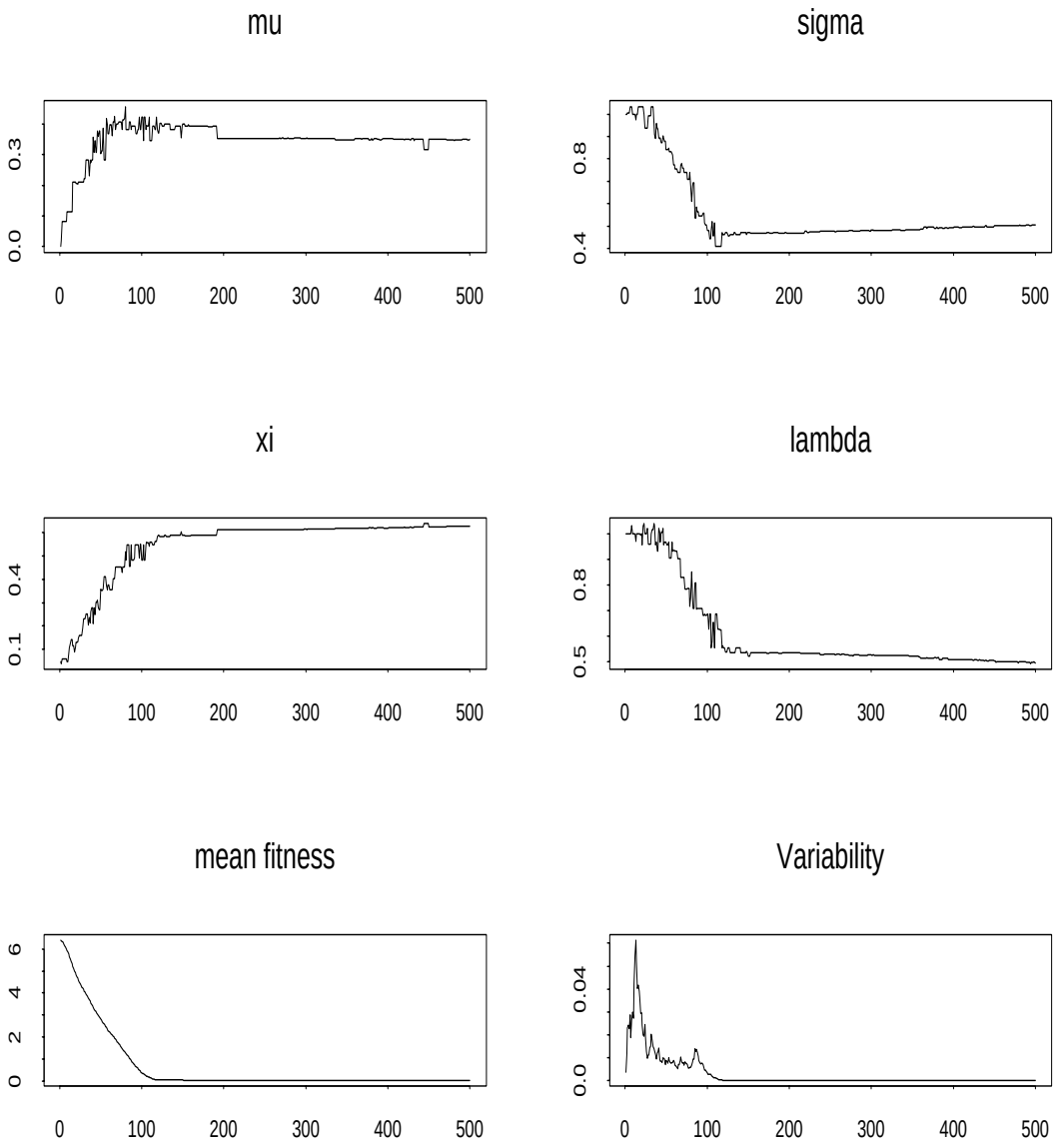
Mutationswahrscheinlichkeit $p = 0.5$,
mittlere Schrittweite $s = 0.03$



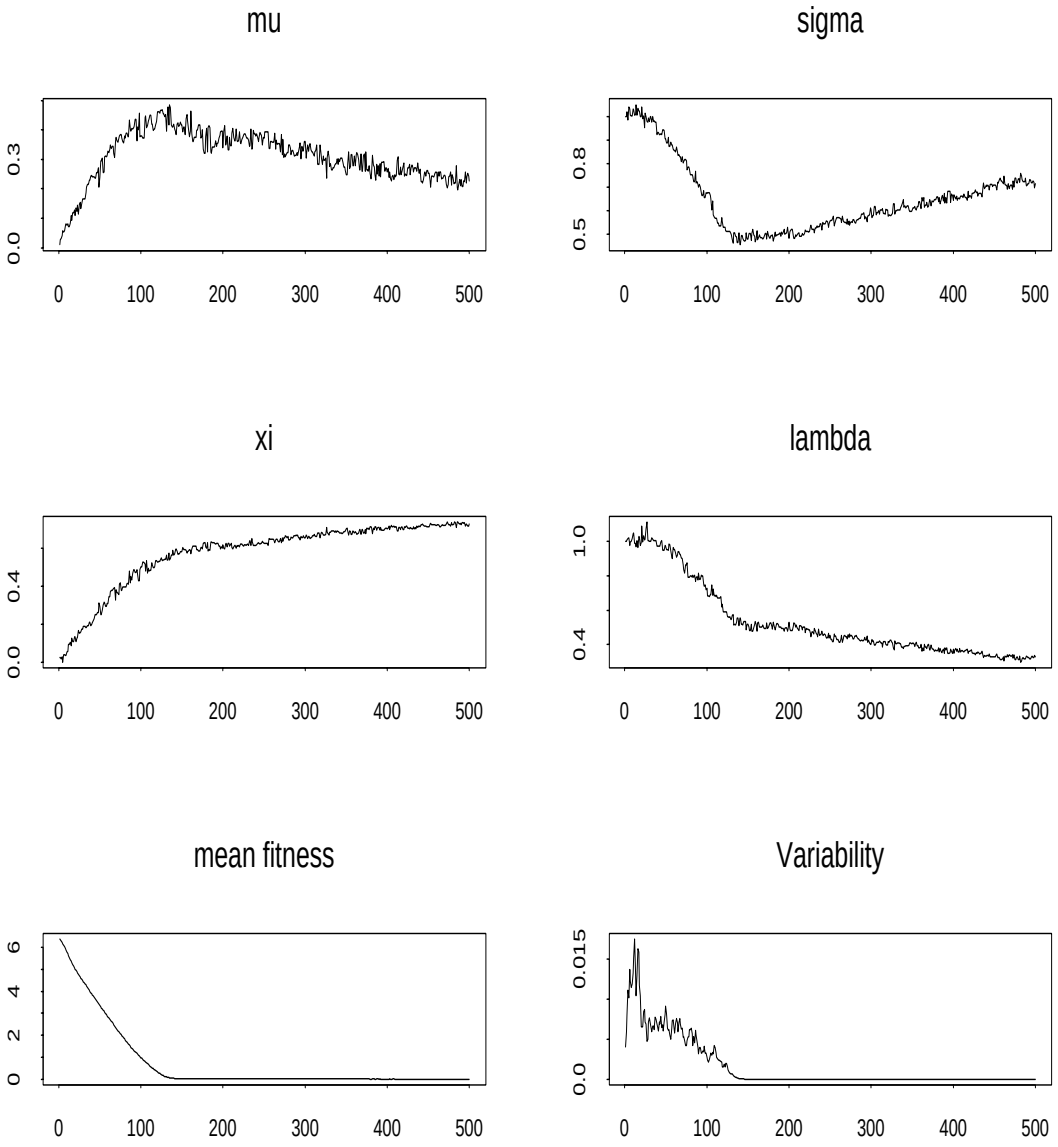
Mutationswahrscheinlichkeit $p = 0.2$,
mittlere Schrittweite $s = 0.03$



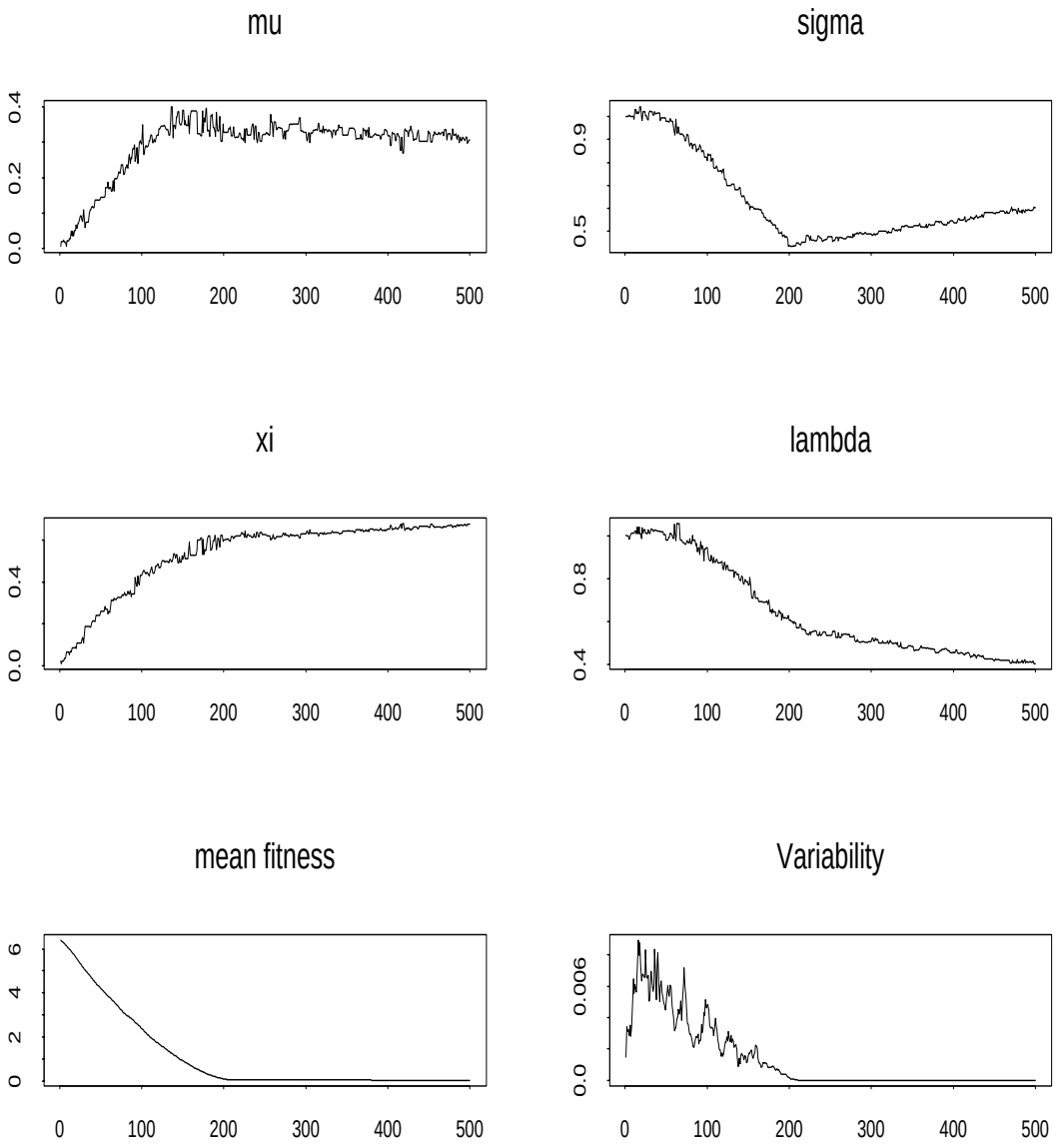
Mutationswahrscheinlichkeit $p = 0.05$,
mittlere Schrittweite $s = 0.03$



Mutationswahrscheinlichkeit $p = 0.5$,
mittlere Schrittweite $s = 0.01$



Mutationswahrscheinlichkeit $p = 0.2$,
mittlere Schrittweite $s = 0.01$



Mutationswahrscheinlichkeit $p = 0.05$,
mittlere Schrittweite $s = 0.01$