Genetische Algorithmen für die Numerische Optimierung

Torsten Hehl

Physikalisches Institut Tübingen

18.12.2023 / 8.1.2024

Klassische Verfahren

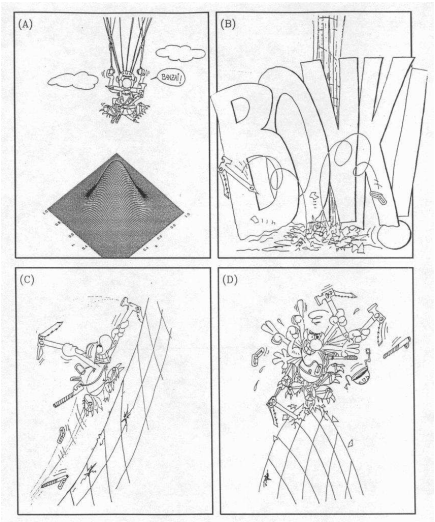
Ziel: Optimierung (Maximierung/Minimierung) einer Funktion (z.B. χ^2) mit möglichst wenig Funktionsaufrufen
Verfahren: Kletterverfahren entlang des steilsten Abfalls/Anstiegs
(ab jetzt: immer Suche nach Maximum)

Suche

(A) Landung in der Nähe des globalen Maximums,

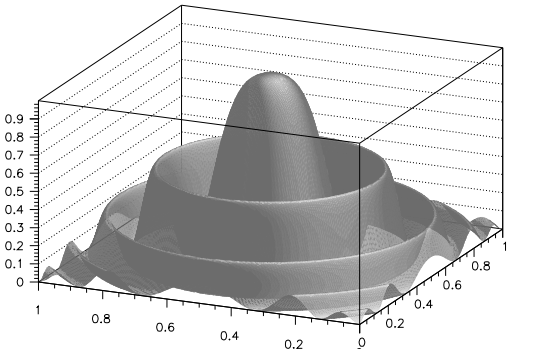
(C) Klettern entlang des steilsten Anstiegs,

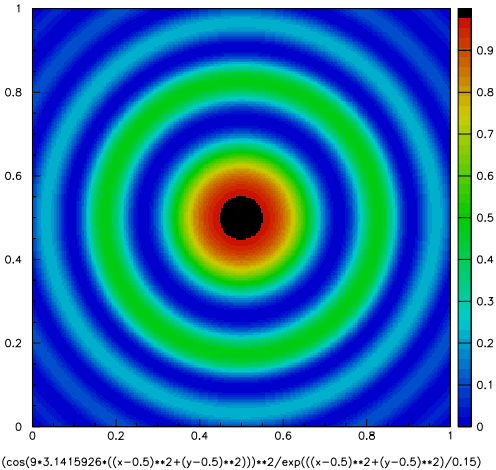
(D) Maximum erreicht



Probleme der klassischen Verfahren

Klettern kann man nur auf *lokale* Maxima: $\rightarrow$ globales Maximum ist hier schwer zu finden.





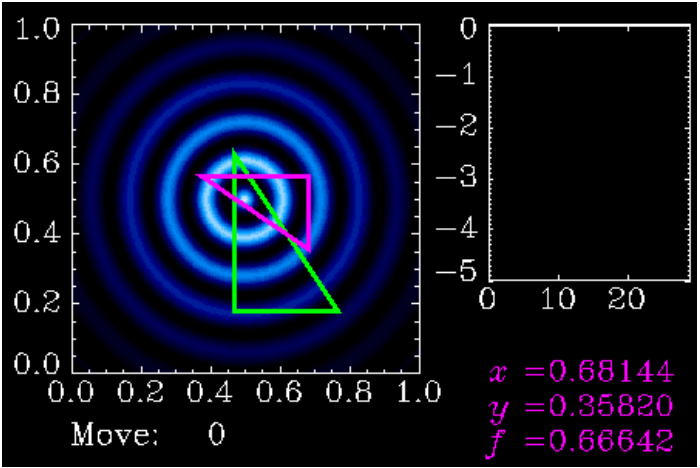
Ausweg mit klassischen Verfahren

Iteration (wiederholter Start von verschiedenen Punkten)

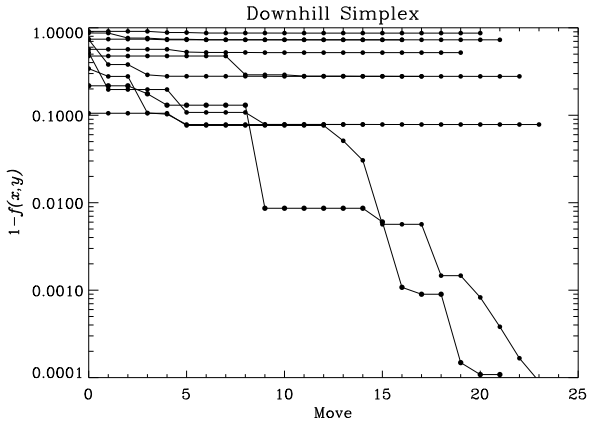
Drei Leistungskriterien:

- Absolut: Numerische Präzision der Lösung
- Global: Ist das gefundene Maximum wirklich das globale Maximum?
- Relativ: Wie schnell konvergiert das Verfahren?

Simplex-Vefahren ist relativ robust, kommt durch enge, lange Täler und über Sattel:

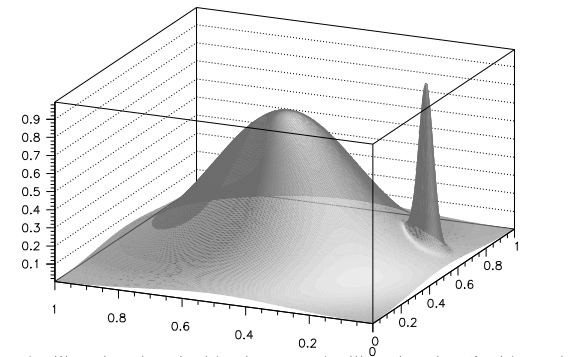


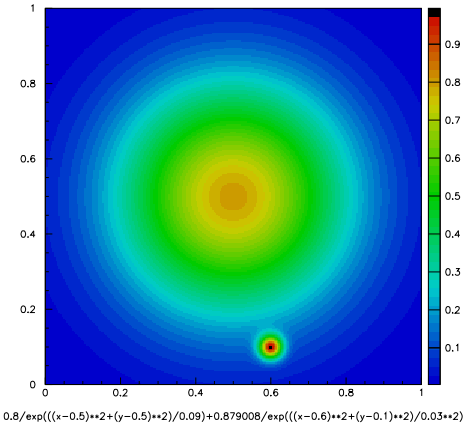
Nur ein Bruchteil der Startversuche landet im globalen Maximum:



Ein Neustart nach vermeintlichem Fund des globalen Maximums empfiehlt sich ...

Ein weiteres Problem





Nur in ca. 1% aller Startwerte wird die Spitze der schmalen Verteilung gefunden.
Problem verschärft sich drastisch mit höherer Dimension!

Natürliche Auslese: nur die Fittesten überleben

Vererbung: nächste Generation erhält zumindest einen Teil der überdurchschnittlichen Eigenschaften

Variabilität: Individuen unterschiedlicher Fitness müssen koexistieren, sonst keine Selektion möglich

Kodierung in den Genen: Weitergabe und Veränderung (Mutation) an nächste Generation, Anpassung sorgt für Individuen knapp über dem Durchschnitt (Optimum ist nicht nötig)
Kumulative Auswahl sorgt für schnellere Anpassung als zufälliges Suchen im Parameterraum

JEG SNAKKER BARE LITT NORSK

Dieser norwegische Satz soll durch ein genetisches Verfahren gefunden werden.
Wahrscheinlichkeit, diesen 27-Buchstabensatz mit den 30 norwegischen Buchstaben spontan zu finden:
$30^{-27} \approx 10^{-40}$

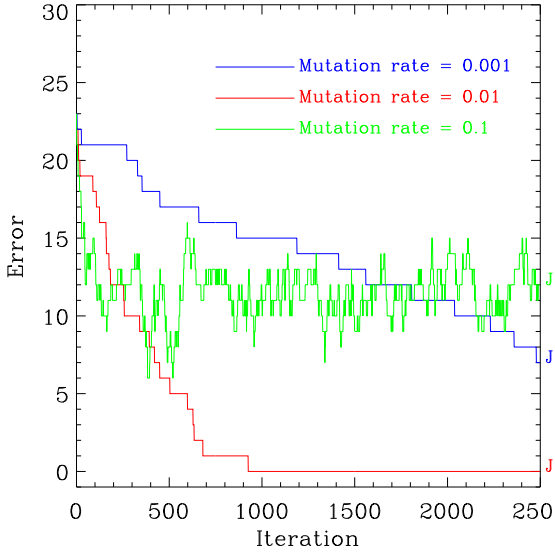
Genetisches Verfahren:

1. Bilde 10 zufällige Sätze mit je 27 Buchstaben
2. Satz mit den meisten richtigen Buchstaben — Auslese
3. Dupliziere diesen besten Satz zehnmal — Vererbung
4. Ändere in jedem dieser Sätze zufällig einige Buchstaben — Mutation
5. Wiederhole Schritte 2 – 4, bis der Zielsatz gefunden wird

Ich spreche nur ein bisschen norwegisch

Target	J E G	S N A K K E R	B A R E	L I T T	N O R S K	
1	Z E B Y E N Æ T U V P	Q Å O D E M I	F V G H D O O			23
50	V E G Æ E N Æ R O E O	Q Å B D E M I	F V N Å D O K			19
100	V E G Æ E N Æ K C E O	O P H Z E M I	F V N Å Ø O K			18
150	V E G W X N Æ K C E O	N A H A D M I	C F N N E R O K			16
200	J E G W X N P K K E O	B A H A	R I C E Å N E R O K			12
250	J E G W V N R K K E	B A E A	R I Å E Å N R T K			12
300	J E G R N R K K E	B A E T	U I Ø Ø N Q R K K			10
350	J E G K N K K K E	B A R	U I Ø Ø N Q R M K			9
400	J E G K N V K K E	B A R	P I H Ø N Q R S K			8
450	J E G K N V K K E R	B A R	L I D Ø N Å R S K			6
500	J E G Ø N V K K E R	B A R	L I S Ø N K R S K			6
550	J E G Ø N F K K E R	B A R E	L I S I N B R S K			5
600	J E G S N F K K E R	B A R E	L I A W N B R S K			4
650	J E G S N A K K E R	B A R E	L I A W N O R S K			2
700	J E G S N A K K E R	B A R E	L I A T N O R S K			1
750	J E G S N A K K E R	B A R E	L I A T N O R S K			1
800	J E G S N A K K E R	B A R E	L I A T N O R S K			1
850	J E G S N A K K E R	B A R E	L I A T N O R S K			1
900	J E G S N A K K E R	B A R E	L I Y T N O R S K			1
950	J E G S N A K K E R	B A R E	L I T T N O R S K			0

Abhängigkeit der Konvergenz von der Mutationsrate



Konvergenz
hängt stark von der Mutationsrate ab!

JXO NNAKKKRMBARERLI LHANPLBK

JEGÆSNNKKI R BARE LI TÆ ONHSK

JEG SNAKKER BARE LI TT NORSK

Ein einfacher genetischer Algorithmus

1. Generiere eine zufällige Population und messe ihre Fitness (z.B. χ^2)
2. Die fittesten Individuen werden zur Fortpflanzung ausgewählt
3. Ersetze Eltern durch Kinder
4. Bestimme die Fitness der Kinder
5. Wiederhole Schritte 2 – 4, bis fittestes Individuum fit genug ist

Kodierung

```
P(P1)  x=0.14429628  y=0.72317247
P(P2)  x=0.71281369  y=0.83459991
```

Fortpflanzung

```
S(P1)      14429628 72317247
S(P2)      71281369 83459991
```

Kreuzung

```
S(O1)      14481369 83459991
S(O2)      71229628 72317247
```

Mutation

```
S'(O2)     71229628 78317247
```

Dekodierung

```
P(O2)  x=0.71229628  y=0.78317247
P(O1)  x=0.14481369  y=0.83459991
```

PIKAIA – eine genetischer Algorithmus für numerische Optimierungen

`PIKAIA`: FORTRAN-77 Routine

(Charbonneau & Knapp, 1995-2002)

`www.hao.ucar.edu/modeling/pikaia/pikaia.php`

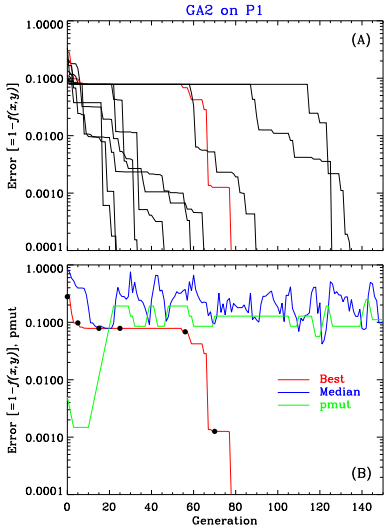
Nutzerdefinierte Funktion $f(\mathrm{x})$ in begrenztem n-dimensionalen Gebiet wird *maximiert*,

$$\mathrm{x} \equiv (x_1, x_2, \ldots x_n), \qquad x_k \in [0, 1]$$

- starte mit Anfangspopulation N_p, Populationsgröße fix
- Zahl der Generationen N_g ist Abbruchkriterium
- Auswahl: proportional zu Fitness-Rang (nicht Fitness, sonst droht Degenerierung)
- Kreuzung: wie in Schema vorgeführt
- Fortpflanzung: Kreuzungsrate (0.85), Mutationsrate (0.005 je Stelle)
- Elitismus: Fittestes Individuum mutiert nicht
- Variable Mutationsrate: Fitness-Differenz $\Delta f \sim 1/mut.r.$

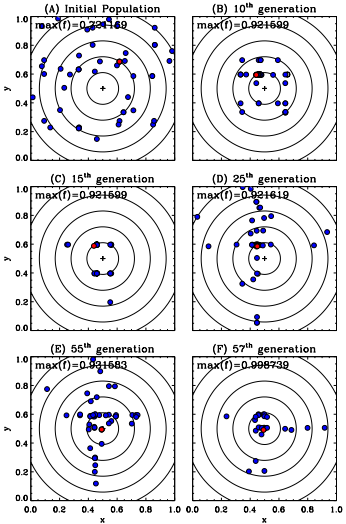
Lösung von Problem 1 (konzentrische Wellen) mit PIKAIA:

Entwicklung des fittesten Individuums für verschiedene Anfangspopulationen



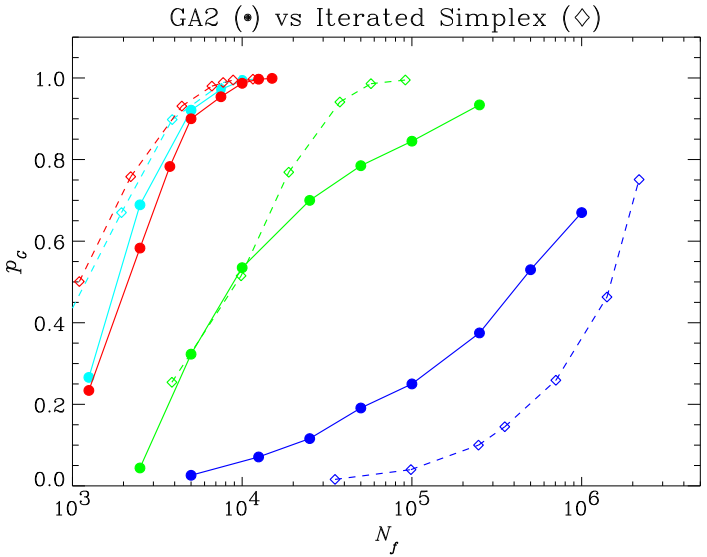
Entwicklung der Generationen

Erst in der 55. Generation ist ein Mutant in der Nähe des Maximums der Fitteste!



Weitere Verfeinerungen von PIKAIA

- Hamming-Wände: Das Kodierschema kennt keinen Übertrag, der Übergang von z.B. ..19.. $\rightarrow$ ..20.. oder umgekehrt durch synchrone Mutationen ist extrem unwahrscheinlich
- Binäre Kodierung: Binärstellen stellen kein unüberwindliches Hindernis dar
- Schleichende Mutation: Übertrag wird berücksichtigt


GA2 (•) vs Iterated Simplex (◇)
1.0
0.8
0.6
0.4
0.2
0.0
p_c
10^3
10^4
10^5
10^6
N_f

Alternativen zu PIKAIA

- Python: **geneticalgorithm** (`https://pypi.org/project/geneticalgorithm`) Empfehlung!
- weitere Python-Algorithmen `https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6`
- F90-Version mit Optimierungen:
  `http://jacobwilliams.github.io/pikaia/`
- VBA-Version für Excel:
  `http://www.ecy.wa.gov/programs/eap/models/pikaia.zip`
- Excel/LibreOffice: Solver (Evolutionary Method)
- GA in MATLAB:
  >> help ga
- GPAPACK (in C), siehe GitHub