

UNIVERSITÄT DORTMUND

Wintersemester 2007/08

Praktische Optimierung
(Vorlesung)

Prof. Dr. Günter Rudolph
Fachbereich Informatik
Lehrstuhl für Algorithm Engineering

Evolutionäre Algorithmen (EA)

(1+1)-EA:

Wähle $X^{(0)} \in \mathbb{R}^n, s_0 > \varepsilon > 0, k = 0$

```

while ( $s_k > \varepsilon$ ) {
   $Y = X^{(k)} + s_k \cdot m^{(k)}$ 
  if  $f(Y) < f(X^{(k)})$  then  $X^{(k+1)} = Y ; s_{k+1} = a^+(s_k)$ 
  else  $X^{(k+1)} = X^{(k)} ; s_{k+1} = a^-(s_k)$ 
  k++
}

```

Schrittweite Zufallsvektor

einfachstes Modell der Evolution:

Mutation

Selektion

Schrittweitenanpassung: z.B.

$$\left. \begin{array}{l} a^+(s) = s / \gamma \\ a^-(s) = s \cdot \gamma \end{array} \right\} \gamma \in (0,1)$$

Wie sollte die Mutationsverteilung gewählt werden?

Rudolph: PO (WS 2007/08) • Evolutionäre Algorithmen 2

Evolutionäre Algorithmen

Forderungen an Such- / Mutationsverteilung von $m^{(k)}$

- Keine Richtung ohne Grund bevorzugen → Symmetrie um 0
- Kleine Änderungen wahrscheinlicher als große → Unimodal mit Modus 0
- Steuerbar: Größe der Umgebung, Streuung → Parametrisierbar
- Leicht erzeugbar
- ...

symmetrisch, multimodal ⇒ symmetrisch, unimodal

Rudolph: PO (WS 2007/08) • Evolutionäre Algorithmen 3

Evolutionäre Algorithmen

Gleichverteilung $f_m(x) = \frac{1}{2r} \cdot 1_{(-r,r)}(x)$

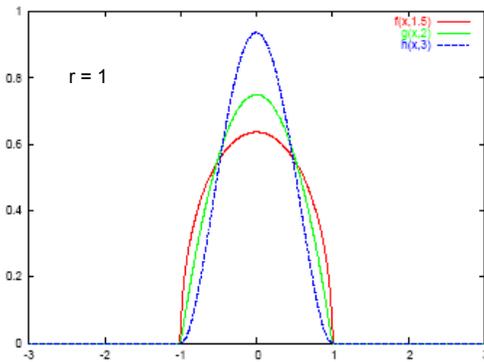
- symmetrisch
- unimodal
- steuerbar → r
- leicht erzeugbar:

$$m = r(2u - 1)$$

wobei $u \in [0,1)$ gleichverteilt (aus Bibliothek)

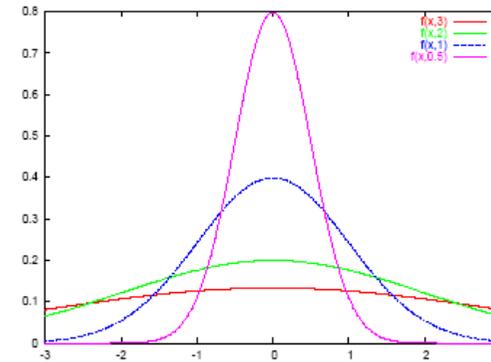
Rudolph: PO (WS 2007/08) • Evolutionäre Algorithmen 4

Beta-Verteilung $f_m(x) = \frac{r^{1-2p}}{\sqrt{\pi}} \cdot \frac{\Gamma(p + \frac{1}{2})}{\Gamma(p)} (1-x^2)^{p-1} \cdot 1_{(-r,r)}(x)$



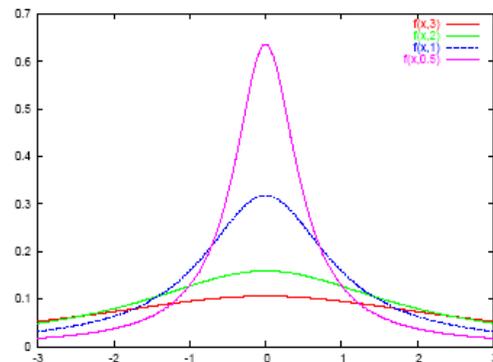
- symmetrisch
- unimodal
- steuerbar $\rightarrow r, p$
- leicht erzeugbar (Bibliothek)

Normalverteilung $f_m(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$



- symmetrisch
- unimodal
- steuerbar $\rightarrow \sigma$
- nicht ganz so leicht erzeugbar (Bibliothek)

Cauchy-Verteilung $f_m(x) = \frac{1}{c\pi} \cdot \frac{1}{1 + (\frac{x}{c})^2}$



- symmetrisch
- unimodal
- steuerbar $\rightarrow c$
- leicht erzeugbar (Bibliothek)

Besonderheit:
unendliche Varianz

Höherdimensionale Suchräume: Symmetrie? Unimodalität? Steuerbarkeit?

↓
Rotationssymmetrie

Definition:

Sei T eine $(n \times n)$ -Matrix mit $T^T T = I_n$, (I_n : n -dim. Einheitsmatrix)
 T heißt **orthogonale Matrix** oder **Rotationsmatrix**. ■

Beispiel:

$$T = \begin{pmatrix} \cos \omega & -\sin \omega \\ \sin \omega & \cos \omega \end{pmatrix}$$

$y = T^T x \Rightarrow$ Vektor x wurde um Winkel ω gedreht

Evolutionäre Algorithmen

Definition:

n-dimensionaler Zufallsvektor x heißt

sphärisch symmetrisch oder **rotationssymmetrisch**

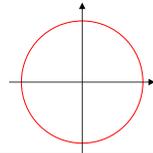
$\Leftrightarrow x \stackrel{d}{=} T \cdot x$ für jede orthogonale Matrix T . ■

$x \stackrel{d}{=} y$ bedeutet: x hat die gleiche Verteilung wie y

Beispiel: Gleichverteilung auf Kreis (Hyperkugel der Dimension $n = 2$)

u gleichverteilt in $[0, 1] \Rightarrow \omega = 2\pi u$

$$x \stackrel{d}{=} \begin{pmatrix} \cos \omega \\ \sin \omega \end{pmatrix}$$



Evolutionäre Algorithmen

Satz:

Zufallsvektor x rotationssymmetrisch $\Leftrightarrow x \stackrel{d}{=} r u^{(n)}$, wobei

r nichtnegative Zufallsvariable und

$u^{(n)}$ Zufallsvektor mit Gleichverteilung auf n -dim. Hyperkugelrand mit Radius 1. ■

Bemerkung:

r und $u^{(n)}$ sind stochastisch unabhängig, $u^{(n)} \stackrel{d}{=} \frac{x}{\|x\|}$

Erzeugung von rotationssymmetrischen Zufallsvektoren:

1. Wähle zufällige Richtung $u^{(n)}$
2. Wähle zufällige Schrittweite r
3. Multiplikation: $x = r u^{(n)}$

Evolutionäre Algorithmen

Beispiel: Multivariate Normalverteilung

Zufallsvektor m erzeugbar via

1. $m = \sigma \cdot (m_1, m_2, \dots, m_n)$,
wobei $m_i \sim N(0, 1)$ stoch. unabh., oder

2. $m = r \cdot u$, wobei $r \sim \chi_n(\sigma)$, $u \sim U(\partial S_n(1))$.

\uparrow χ -Verteilung mit n Freiheitsgraden \uparrow Gleichverteilung auf Hyperkugelrand

$$\partial S_n(r) = \{x \in \mathbb{R}^n : \|x\| = r\} \text{ Hyperkugelrand}$$

Evolutionäre Algorithmen

Beispiel: Multivariate Cauchyverteilung

Zufallsvektor m erzeugbar via

1. $m = \sigma \cdot (m_1, m_2, \dots, m_n) / m_0$,
wobei $m_i \sim N(0, 1)$ stoch. unabh., oder

2. $m = r \cdot u$, wobei $r/n \sim F_{n,1}$, $u \sim U(\partial S_n(1))$.

\nearrow F-Verteilung mit $(n, 1)$ Freiheitsgraden \uparrow Gleichverteilung auf Hyperkugelrand

Achtung:

Zufallsvektor aus n unabh. Cauchy-Zufallsvariablen nicht rotationssymmetrisch!

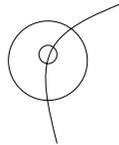
Evolutionäre Algorithmen

(1+1)-EA mit Schrittweitenanpassung (1/5-Erfolgsregel, Rechenberg 1973)

Idee:

- Wenn viele erfolgreiche Mutationen, dann Schrittweite zu klein.
- Wenn wenige erfolgreiche Mutationen, dann Schrittweite zu groß.

bei infinitesimal
kleinem Radius ist
Erfolgsrate = 1/2



Ansatz:

- Protokolliere erfolgreiche Mutationen in gewissem Zeitraum
- Wenn Anteil größer als gewisse Schranke (z. B. 1/5), dann Schrittweite erhöhen, sonst Schrittweite verringern

Evolutionäre Algorithmen

Satz:

(1+1)-EA mit 1/5-artiger Schrittweitensteuerung konvergiert für streng konvexe Probleme zum globalen Minimum mit linearer Konvergenzordnung.

Jägersküpfer 2006

lineare Konvergenzordnung:

$$E[f(X_{k+1}) - f^* | X_k] \leq c \cdot E[f(X_k) - f^*] \quad \text{mit } c \in (0,1)$$

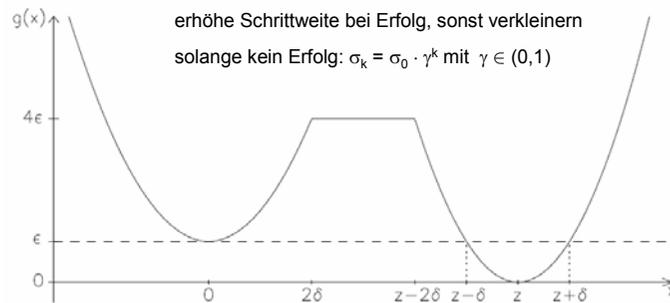
deshalb im allgemeinen, multimodalen Fall:

⇒ schnelle Konvergenz zum lokalen Optimum

Anmerkung: gleiche Konvergenzordnung wie Gradientenverfahren!

Evolutionäre Algorithmen

Konvergenzproblematik bei der Schrittweitenanpassung



Annahme: $X_0 = 0$

Frage: Wird lokales Optimum sicher verlassen (Übergang zu $[z-\delta, z+\delta]$) ?

Evolutionäre Algorithmen

Sei q_k Wahrscheinlichkeit, im Schritt k das lokale Optimum zu verlassen.

Kriterium für sicheres Verlassen:

$$1 - \prod_{k=1}^{\infty} (1 - q_k) = 1 \Leftrightarrow \prod_{k=1}^{\infty} (1 - q_k) = 0 \Leftrightarrow \sum_{k=1}^{\infty} \log \frac{1}{1 - q_k} = \infty$$

Kriterium für unsicheres Verlassen:

$$1 - \prod_{k=1}^{\infty} (1 - q_k) < 1 \Leftrightarrow \prod_{k=1}^{\infty} (1 - q_k) > 0 \Leftrightarrow \sum_{k=1}^{\infty} \log \frac{1}{1 - q_k} < \infty$$

Vereinfachung des log-Terms →

Evolutionäre Algorithmen

Lemma:

Sei $x \in (0,1)$. Dann gilt: $x < \log\left(\frac{1}{1-x}\right) < \frac{x}{1-x}$

Beweis:

Reihenentwicklung $\log\left(\frac{1}{1-x}\right) = -\log(1-x) = \sum_{i=1}^{\infty} \frac{x^i}{i}$

also: $0 < x < \sum_{i=1}^{\infty} \frac{x^i}{i} < \sum_{i=1}^{\infty} x^i = \sum_{i=0}^{\infty} x^i - 1 = \frac{x}{1-x}$

q.e.d.

Evolutionäre Algorithmen

Hinreichendes Kriterium für unsicheres Verlassen:

$$\sum_{k=1}^{\infty} \log \frac{1}{1-q_k} < \sum_{k=1}^{\infty} \frac{q_k}{1-q_k} < \frac{1}{1-q_1} \sum_{k=1}^{\infty} q_k < \infty$$

↑ Lemma
↑ weil q_k monoton fallend

$$p_k = P\{0 \rightarrow (z-\delta, z+\delta)\} = P\{z-\delta < Z < z+\delta\} = F_Z(z+\delta) - F_Z(z-\delta) = 2\delta f_Z(z-\delta + \theta \cdot 2\delta) \quad \text{mit } \theta \in (0,1)$$

Mittelwertsatz der Differentialrechnung!

Annahme: Dichte $f_Z(\cdot)$ von Z ist unimodal

dann: $2\delta f_Z(z+\delta) < p_k < 2\delta f_Z(z-\delta)$ und deshalb: $q_k = 2\delta f_Z(z-\delta)$

Evolutionäre Algorithmen

Z sei normalverteilt $f_Z(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$

$$p_k \leq q_k = \delta \sqrt{\frac{2}{\pi}} \frac{1}{\sigma_k} \exp\left(-\frac{(z-\delta)^2}{2\sigma_k^2}\right) = A \eta_k \exp(-B \eta_k^2)$$

wobei

$$A = \delta(2/\pi)^{1/2}, \quad B = (z-\delta)^2/2, \quad \eta_k = 1/\sigma_k.$$

Sei $\eta_k = \eta_0 \beta^k$ mit $\beta = 1/\gamma > 1$

$$\sum_{k=1}^{\infty} \frac{\beta^k}{\exp(B \eta_0^2 \beta^{2k})} \quad \text{konvergiert nach Wurzelkriterium!}$$

⇒ kein sicheres Entkommen von lokalen Optima!

$$\sum_{k=0}^{\infty} |a_k| < \infty \quad \text{falls} \\ \lim_{k \rightarrow \infty} |a_k|^{1/k} = \alpha < \infty$$

Evolutionäre Algorithmen

Schrittweitensteuerung nach Rechenberg:

Individuum (x, σ) $\gamma \in (0,1) \subset \mathbb{R}$

$$\sigma^{(k)} = \begin{cases} \sigma^{(k-\Delta k)} / \gamma, & \text{falls } \frac{\# \text{Verbesserungen}}{\# \text{Mutationen}} > 1/5 \text{ während } \Delta k \text{ Mutationen} \\ \sigma^{(k-\Delta k)} \cdot \gamma, & \text{sonst} \end{cases}$$

Problem: keine Konvergenz mit W'keit 1

aber: schnelle Konvergenz zum lokalen Optimum + W'keit > 0 dieses zu verlassen!

⇒ kein globales Verfahren, aber gutes nicht-lokales Verhalten!

Beobachtung: Anpassung σ sprunghaft ⇒ Anpassung kontinuierisieren!

Evolutionäre Algorithmen

Schrittweitensteuerung nach Schwefel:

Individuum (x, σ) : auch Strategieparameter wie σ werden mutiert

Mutation:

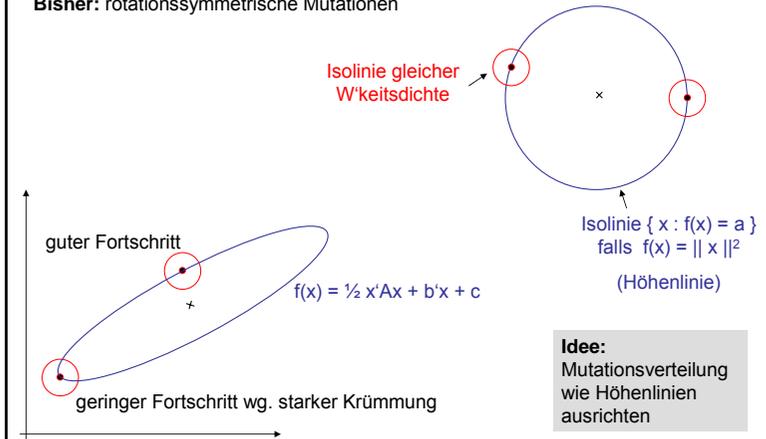
- $\sigma_{k+1} = \sigma_k \cdot \exp(N(0, \tau^2))$ $\tau = 1 / n^{1/2}$
- $X_{k+1} = X_k + \underbrace{\sigma_{k+1}} \cdot N(0, I)$

Wichtig: die bereits mutierte Schrittweite wird verwendet!

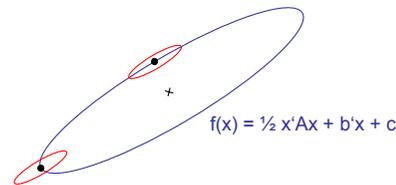
„Schrittweite“ σ wird multiplikativ verändert (logarithmisch normalverteilt), neue Schrittweite wird verwendet bei additiver Veränderung der Position

Evolutionäre Algorithmen

Bisher: rotationssymmetrische Mutationen



Evolutionäre Algorithmen



Wie erzeugt man solche Mutationsverteilungen?

- $Z \sim N(0, \sigma^2 I_n)$ \Rightarrow rotationssymmetrisch (I_n = Einheitsmatrix mit Rang n)
- $Z \sim N(0, D^2)$ \Rightarrow ellipsoid, achsenparallel ($D = \text{diag}(\sigma_1, \dots, \sigma_n)$, Diagonalmatrix)
- $Z \sim N(0, C)$ \Rightarrow ellipsoid, frei beweglich (C = Kovarianzmatrix)
 $C = C'$ (symmetrisch) und $\forall x: x'Cx > 0$ (positiv definit)

Evolutionäre Algorithmen

Wie muss Kovarianzmatrix C gewählt werden?

Ansatz: Taylor-Reihenentwicklung

$$f(x+h) = f(x) + \underbrace{h' \nabla f(x)}_{\text{linear}} + \underbrace{\frac{1}{2} h' \nabla^2 f(x) h}_{\text{quadratisch}} + \underbrace{R(x, h)}_{\text{Restterme (ignorierbar, da } h \text{ klein)}}$$

$\nabla^2 f(x) = H(x)$ Hessematrix

\rightarrow enthält Informationen über Skalierung und Orientierung der Höhenlinien

\rightarrow Es wird sich zeigen: Wähle $C = H^{-1}$!

Evolutionäre Algorithmen

Approximation: $f(x) \approx \frac{1}{2} x'Ax + b'x + c \Rightarrow$ Hessematrix $H = A$

Koordinatentransformation: $x = Qy$ $Q: (n \times n) - \text{Matrix}$

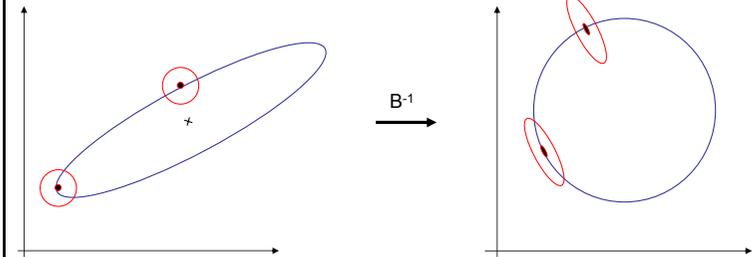
$$\begin{aligned} \Rightarrow f(Qy) &= \frac{1}{2} (Qy)' A (Qy) + b' (Qy) + c \\ &= \frac{1}{2} y' Q' A Q y + b' Q y + c \\ &= \frac{1}{2} y' Q' B' B Q y + b' Q y + c \\ &= \frac{1}{2} y' (Q' B') (B Q) y + b' Q y + c \\ &= \frac{1}{2} y' y + b' B^{-1} y + c \end{aligned}$$

mit Cholesky-Zerlegung $A = B'B$
sei jetzt $Q = B^{-1}$

rotationssymmetrische
Höhenlinien!

also: wir benötigen
Dreiecksmatrix Q bzw. B^{-1}

Evolutionäre Algorithmen



\Rightarrow durch Koordinatentransformation mit B^{-1} wird Problem kugelsymmetrisch!
 \Rightarrow also kugelsymmetrische Mutation transformieren!

Evolutionäre Algorithmen

Satz:

Sei $y \sim N(0, I_n)$ und $Q'Q$ eine positiv definite Matrix mit Rang n .
Dann $x = Q'y \sim N(0, Q'Q)$.

\Rightarrow mit $Q' = B^{-1}$ können wir Mutationsverteilungen wie gewünscht ausrichten!

aber: woher bekommen wir Matrix Q ?

\Rightarrow Selbstanpassung der Matrixelemente wie bei Schrittweite nach Schwefel

da $H = A = B'B$, ist $H^{-1} = (B'B)^{-1} = B^{-1}(B^{-1})' \stackrel{\text{def}}{=} C = Q'Q$

Q entsteht durch Cholesky-Zerlegung von C , ist also Dreiecksmatrix

\rightarrow Skalierungsfaktoren je Zeile herausziehen: in Diagonalmatrix S ablegen

\rightarrow Q zerlegbar in $Q = S \cdot T$ mit $t_{ii} = 1$ (S hat n Parameter, T hat $n(n-1)/2$ Parameter)

Evolutionäre Algorithmen

Satz:

Jede sym., pos. definite Matrix A ist zerlegbar via $A = T'DT$ und umgekehrt,
wobei T orthogonale Matrix ($T' = T^{-1}$) und D Diagonalmatrix mit $d_{ii} > 0$.

\Rightarrow also wählen wir $S = D^{1/2}$, so dass $A = (TS)'(TS)$

Satz:

Jede orthogonale Matrix T kann durch das Produkt von $n(n-1)/2$ elementaren Rotationsmatrizen $R_{ij}(\omega_k)$ dargestellt werden:

$$T = \prod_{i=1}^{n-1} \prod_{j=i+1}^n R_{ij}(\omega_k)$$

$R_{ij}(\omega) =$ wie Einheitsmatrix, jedoch mit $r_{ii} = r_{jj} = \cos \omega$, $r_{ij} = -r_{ji} = -\sin \omega$

Evolutionäre Algorithmen

Geometrische Interpretation

durch $Q'y = TSy$ wird rotationssymmetrischer Zufallsvektor y

- zunächst achsenparallel skaliert via Sy
- und dann durch $n(n-1)/2$ elementare Rotationen in gewünschte Orientierung gebracht via $T(Sy)$

Mutation der Winkel ω :

$$\omega^{(t+1)} = (\omega^{(t)} + W + \pi) \bmod (2\pi) - \pi \in (-\pi, \pi]$$

wobei $W \sim N(0, \kappa^2)$ mit $\kappa = 5^\circ\pi / 180^\circ$

→ Individuum jetzt: (x, σ, ω) mit n Schrittweiten (Skalierungen) + $n(n-1)/2$ Winkel

Praxis zeigt:

Idee gut, aber Realisierung nicht gut genug (funktioniert nur für kleines n)

Evolutionäre Algorithmen

Wie könnte man sonst noch an Matrixelemente von Q kommen?

(Rudolph 1992)

Modellannahme: $f(x) \approx \frac{1}{2} x'Ax + b'x + c$

Beobachtung: Bei (μ, λ) – Selektion werden λ Paare $(x, f(x))$ berechnet.

⇒ Falls $\lambda > n(n+1)/2 + n + 1$, dann **überbestimmtes** lineares Gleichungssystem:

$$\left. \begin{array}{l} f(x_1) = \frac{1}{2} x_1'Ax_1 + b'x_1 + c \\ \vdots \\ f(x_\lambda) = \frac{1}{2} x_\lambda'Ax_\lambda + b'x_\lambda + c \end{array} \right\} v = (A, b, c) \text{ hat } n(n-1)/2 + n + 1 \text{ zu schätzende Parameter, wobei } A = B'B$$

⇒ multiple lineare Regression für $f = Xv \rightarrow X'f = X'Xv \rightarrow (X'X)^{-1}X'f = v$

⇒ aus Schätzer $v = (A, b, c)$ bekommen wir Hessematrix $H = A$

⇒ Cholesky-Dekomposition von H und Matrixinversion liefert Q

Praxis zeigt: funktioniert sehr gut, aber zu hoher Aufwand: $(X'X)^{-1}$ kostet $\mathcal{O}(n^6)$

Evolutionäre Algorithmen

Idee: Matrix C nicht in jeder Generation schätzen, sondern iterativ nähern!

(Hansen, Ostermeier et al. 1996ff.)

→ Covariance Matrix Adaptation Evolutionary Algorithm (CMA-EA)

Setze initiale Kovarianzmatrix auf $C^{(0)} = I_n$

$$C^{(t+1)} = (1-\eta) C^{(t)} + \eta \sum_{i=1}^{\mu} w_i d_i d_i' \quad \eta : \text{„Lernrate“} \in (0,1)$$

$$m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i;\lambda} \quad \text{Mittelpunkt aller } \underline{\text{selektierten}} \text{ Eltern}$$

Aufwand:
 $\mathcal{O}(\mu n^2 + n^3)$

$$d_i = (x_{i;\lambda} - m) / \sigma \quad \text{Sortierung: } f(x_{1;\lambda}) \leq f(x_{2;\lambda}) \leq \dots \leq f(x_{\lambda;\lambda})$$

$$\text{dyadisches Produkt: } dd' = \begin{pmatrix} d_1 d_1 & d_1 d_2 & \dots & d_1 d_\mu \\ d_2 d_1 & d_2 d_2 & \dots & d_2 d_\mu \\ \vdots & \vdots & \ddots & \vdots \\ d_\mu d_1 & d_\mu d_2 & \dots & d_\mu d_\mu \end{pmatrix} \text{ ist positiv semidefinite Streuungsmatrix}$$

Evolutionäre Algorithmen

Variante:

$$m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i;\lambda} \quad \text{Mittelpunkt aller } \underline{\text{selektierten}} \text{ Eltern}$$

$$p^{(t+1)} = (1 - \chi) p^{(t)} + (\chi (2 - \chi) \mu_{\text{eff}})^{1/2} (m^{(t)} - m^{(t-1)}) / \sigma^{(t)} \quad \text{„Evolutionsspfad“}$$

$$p^{(0)} = 0 \quad \chi \in (0,1)$$

$$\left. \begin{array}{l} C^{(0)} = I_n \\ C^{(t+1)} = (1 - \eta) C^{(t)} + \eta p^{(t)} (p^{(t)})' \end{array} \right\} \text{Aufwand: } \mathcal{O}(n^2)$$

→ Cholesky-Zerlegung: $\mathcal{O}(n^3)$ für $C^{(t)}$



State-of-the-art: CMA-EA

- erfolgreiche Anwendungen in der Praxis
- insbesondere wenn Zielfunktionsauswertung zeitaufwändig
(z.B. Zielfunktionsauswertung durch Simulationsprogramm)

Implementierungen im WWW verfügbar