UNIVERSITÄT FRANKFURT AM MAIN

www.uni-frankfurt.de

Evolutionäre Algorithmen AS1-7

UNIVERSITÄT FRANKFURT AM MAIN

Evolutionäre Algorithmen

Genetische Algorithmen

Evolution neuronaler Netze

UNIVERSITÄT FRANKFURT AM MAIN

Lernen durch Evolutionäre Algorithmen

Lernziel: Maximieren einer Zielfunktion $R(g_1, ..., g_n)$ durch Wahl von *n* Parametern $g_1, ..., g_n$

Evolutionsschema *creeping random search*

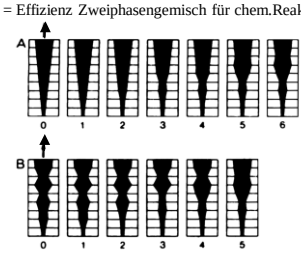
1. Wähle initial zufällige Werte **g** = ($g_1, ..., g_n$)
2. Evaluiere die Lösung, bilde R(**g**) z.B. empirisch
3. Wähle zufällige Werte g_1', ..., g_n', etwa leicht abweichend mit einer Normalverteilung N(**g**, **s**)
4. Evaluiere die Lösung, bilde R(**g'**)
5. R(**g'**) > R(**g**) ? JA: **g'**←**g** Nein: -
6. Lösung ausreichend gut ? JA: STOP.
   NEIN

Evolutionäre Algorithmen: Beispiel

Entwicklung einer Düsenform (Rechenberg 1973)

Parameter **g** = $(g_1,...,g_n)$ mit g_i = Lochdurchmesser, n = 9

Ziel = Effizienz Zweiphasengemisch für chem.Reaktion maximieren



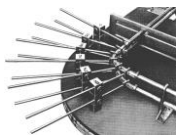
Ergebnis: optimaler Durchfluss, unabh. von Startkonfigurationen

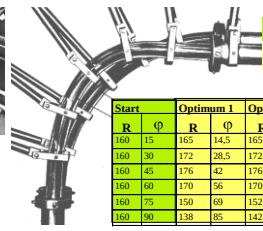
Evolutionäre Algorithmen: Beispiel

Versuchsaufbau Rohrkrümmung (Rechenberg 1973)

Parameter **g** = $(g_1,...,g_n)$ mit $g_i = (R_i,\varphi_i)$ Polarkoordinaten der Stützstellen

Ziel = Durchflußmenge/Zeit, *Änderungen* = normalvert. Abweichungen





Start		Optimum 1		Optimum 2	
R	φ	R	φ	R	φ
160	15	165	14,5	165	14,5
160	30	172	28,5	172	28,5
160	45	176	42	176	42
160	60	170	56	170	56
160	75	150	69	152	69
160	90	138	85	142	85

Ergebnis: Durchfluss **verbessert** um ca. 10%

Evolutionäre Algorithmen

Genetische Algorithmen

Evolution neuronaler Netze

Lernen durch Genetische Algorithmen

Lebewesen

Bauplan = *Genotyp* **g** = $(g_1,...,g_n)$ mit g = *Chromosom*, g_i = *Gen*
Erscheinungsbild = *Phänotyp* (**g**)
Ziel-Bewertung = *Fitness* (**g**)
Mehrere Lebewesen g_i : *Population* G = {$\mathbf{g}_i$}

Lernziel: Maximieren der Zielfunktion (Fitness)

Reproduktionsplan

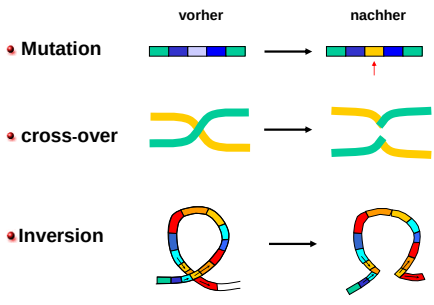
1. Bewerte die Population G	**GetFitness(G)**
2. Selektiere die Besten von G	**SelectBest(G)**
3. Vermehre sie mit Änderungen	**GenOperation(G)**

Bewerten und Selektieren

```
PROCEDURE GetFitness (G: ARRAY OF TUPEL)
BEGIN
  FOR i:=1 TO M DO          (* Bewerte die Population *)
    fitness[i] := R( G[i] )
  ENDFOR;
                            (* Normiere Fitness auf 0..1 *)
Rmin := MIN(fitness); Rmax:= MAX(fitness);
  FOR i:=1 TO M DO fitness[i] := (fitness[i]-Rmin)/(Rmax-Rmin) ;
  ENDFOR
END GetFitness;

PROCEDURE SelectBest (G: ARRAY OF TUPEL);
BEGIN                       (* Lösche die Schlechtesten *)
  FOR i:=1 TO M DO
     IF Random(0,1) >fitness[i] THEN delete(i,G) END
  ENDFOR
END SelectBest;
```


Genetische Operatoren



Vermehren mit Veränderungen

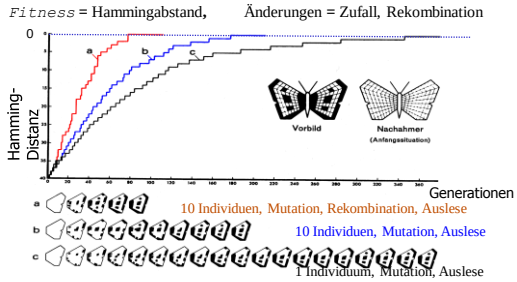
```
PROCEDURE GenOperation (G:ARRAY OF TUPEL);
BEGIN
  FOR k:=1 TO S DO
        i := Random(1,M);  j := Random(1,M);
        mutate (G,i,j);            (* Mutation bei G(i) an Stelle j *)
  END
  FOR k:=1 TO S DO
        i := Random(1,M);   j := Random(1,M);
        cross_over (G,i,j);        (* Überkreuzung von G(i) und G(j) *)
  END
END GenOperation;
```


Beispiel Schmetterling

Evolution eines binären Schmetterlings (Rechenberg 1973)

Genotyp **g** = $(g_1,...,g_n)$ mit g_i aus {1,0}= Färbung ja/nein

Fitness = Hammingabstand, Änderungen = Zufall, Rekombination



a 10 Individuen, Mutation, Rekombination, Auslese

b 10 Individuen, Mutation, Auslese

c 1 Individuum, Mutation, Auslese

Wirkung von Gen-Austausch

Schema Holland 1975

Tupel mit gleichen Lösungselementen *Hyperfläche*

$\mathbf{g}_i = (*, *, 1, 0, *, *, 5, *, ..., *)$ * = *don't care*

building block

Schemalänge L(g): Zahl der Stellen zur pos.unabh. Charakt.

Schematheorem

m(h,t) = Zahl der Individuen mit Schema h bei Generation t

f(h) = Fitness von h, a_t *= mittlere Fitness*

p = Wahrsch., dass Schema h zerstört wird

$$m(h,t+1) \geq \frac{m(h,t)f(h)}{a_t}[1-p]$$

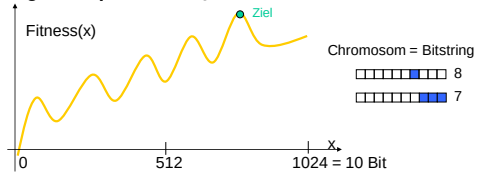
rel. Fitness

⇒ h überlebt, wenn es kurz ist (p klein) und höh. Fitness als a_t hat

Wirkung genetischer Operatoren

Idee: Kombination „guter“ Gene verbessert eine Lösung

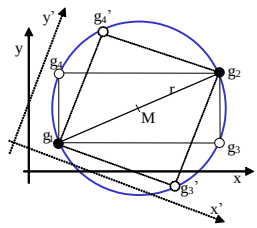
- **Gegenbeispiel:** Ansteigende Gütefunktion mit lokalen Minima



Gen-Rekombination ist nur sinnvoll
bei geeigneten Gen-Definitionen !

Evolutionäre Rekombination

Rekombination = Austausch korresp. Parameter



Beispiel

$\mathbf{g}_1 = (x_1,y_1)$ und $\mathbf{g}_2 = (x_2,y_2)$
ergibt Kinder
$\mathbf{g}_3 = (x_2,y_1)$ und $\mathbf{g}_4 = (x_1,y_2)$
oder in and. Koord.system
$\mathbf{g}_3' = (x_2',y_1')$ und $\mathbf{g}_4' = (x_1',y_2')$

$\Rightarrow$ koord.Syst. unabh.:
Thales-Kreis um *M* mit *r* !

n Parameter $\rightarrow$ *n*-dim Hyperkugel

kontin. Rekombination:
$\Rightarrow$ Normalverteilung N(M,σ)
um Mittelpunkt *M* und Abstand *r*

Rekombination ≈ Evolut. Mutation !

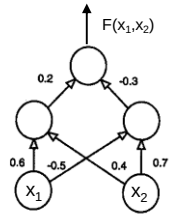
Evolutionäre Algorithmen

Genetische Algorithmen

Evolution neuronaler Netze

Kodierung der Verbindungsgewichte

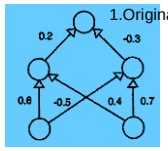
Kodierung eines Netzes **fester Struktur** durch seine Parameter



Individuum **g** = `(0.2, -0.3, 0.6, -0.5, 0.4, 0.7)`

Einfaches Crossover

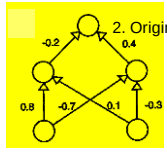
Beispiel

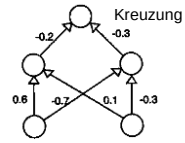


$\mathbf{g}_1$ = (**0.2, -0.3, 0.6, -0.5, 0.4, 0.7**)

$\mathbf{g}_2$ = (**-0.2, 0.4, 0.6, -0.7, 0.1, -0.3**)

$\mathbf{g}_r$ = (**-0.2, -0.3, 0.6, -0.7, 0.1, -0.3**)





Evolution der Verbindungsgewichte

Generation ← 0

Initialisierung der $\mathbf{g}_i$: Weise den Verbindungen in der Anfangspopulation des Netzwerks zufällige Gewichte zu

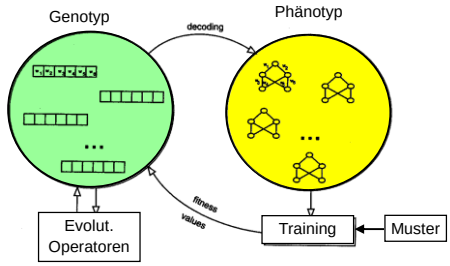
while not *Abbruchbedingung* **do**

- Generation ← Generation +1
- Konstruiere zu jedem Genotyp $\mathbf{g}_i$ das zugehörige Netzwerk N_i
- Trainiere jedes Netzwerk N_i mit den Trainingsdaten,
- Berechne die Fitness(N_i) anhand der Testdaten { (x_1,x_2) },
- Wähle und reproduziere Netzwerke bzw. $\mathbf{g}_i$ gemäß ihrer Fitness,
- Rekombiniere und mutiere gewählte Individuen $\mathbf{g}_i$.

end while

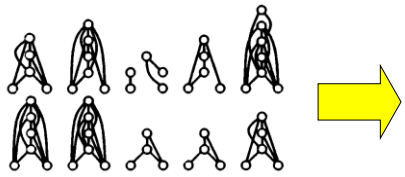
Evolution der Verbindungsgewichte

Überblick Evolutionszyklus



Evolution der Netze

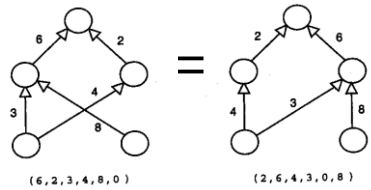
Beispiel: XOR-Netzwerk Evolution



Variable Netzstruktur: 10 Startnetze

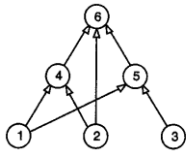
Das Permutations-Problem

Problem unterschiedlicher Genotyp, gleicher Phänotyp



Indirekte Kodierung

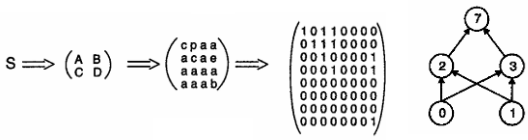
Kodierung über Adjazenzmatrix



	1	2	3	4	5	6
1	0	0	0	1	1	0
2	0	0	0	1	0	1
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

$A_{ik} = 1 \Leftrightarrow$ Knoten *i* hat Verbindung zu Knoten *k*

Indirekte Kodierung



$$S \rightarrow \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$A \rightarrow \begin{pmatrix} c & p \\ a & c \end{pmatrix} \quad B \rightarrow \begin{pmatrix} a & a \\ a & e \end{pmatrix} \quad C \rightarrow \begin{pmatrix} a & a \\ a & a \end{pmatrix} \quad D \rightarrow \begin{pmatrix} a & a \\ a & b \end{pmatrix}$$

$$a \rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad b \rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad c \rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad d \rightarrow \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad e \rightarrow \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Indirektes Kodieren

Allgemein 3-stufiges Kodierungs-Schema

Startsymbol

$$S \rightarrow \begin{pmatrix} W_1 & W_2 \\ W_3 & W_4 \end{pmatrix} \qquad W_i \in \{ A,B,...,Z \}$$

$$W_i \rightarrow \begin{pmatrix} v_1 & v_2 \\ v_3 & v_4 \end{pmatrix} \qquad v_i \in \{ a,b,...,p \}$$

$$v_i \rightarrow \begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix} \qquad z_i \in \{ 0,1 \}$$

Terminale

Grammatik

Eine **Grammatik G** hat i.A. folgende Komponenten:

- ein endliches Alphabet Σ (Terminalzeichen)
- eine endliche Menge V von Variablen mit $\Sigma \cap V = \varnothing$
- das Startsymbol $S \in V$
- eine endliche Menge P von Produktionen

Evolution von Neuronalen Netzen

Vorteile

- **Gewichte** können durch genet. Operationen verbessert werden, ohne in lokalen Optima stecken zu bleiben
- **Neue Strukturen** können entstehen, so dass lokale Optima überwunden werden können

Nachteile

- Die Simulationen sind **sehr aufwendig** und rechenintensiv
- Der Aufwand wächst **exponentiell** mit der Größe der Netze
  (Fluch der Dimensionen = Zahl der Gewichte)