Einkriterielle Evolutionäre Algorithmen

Einführung

Dr. Carsten Franke

Züricher Hochschule für Angewandte Wissenschaften

21. Februar 2012

Lernziele der gesamten Vorlesungsreihe

Erwerb der Beherrschung von Werkzeugen und Methoden aus der Computational Intelligence, Befähigung zur Lösung praktischer Probleme, in den Bereichen

- Optimierung
- Maschinelles Lernen

Vorlesungsplanung

Prüfungen

Prüfungen

- 4 kurze, schriftliche Tests (ca. 30 min)
- mündliche Prüfung (ca. 20 min)

Gesamtnote

- Vorschlag Franke / Poland (Gewichtung: 0,5)
  - Noten der schriftlichen Tests
  - kleine Variationen entsprechend des persönlichen Eindrucks
- Note der mündlichen Prüfung (Gewichtung: 0,5)

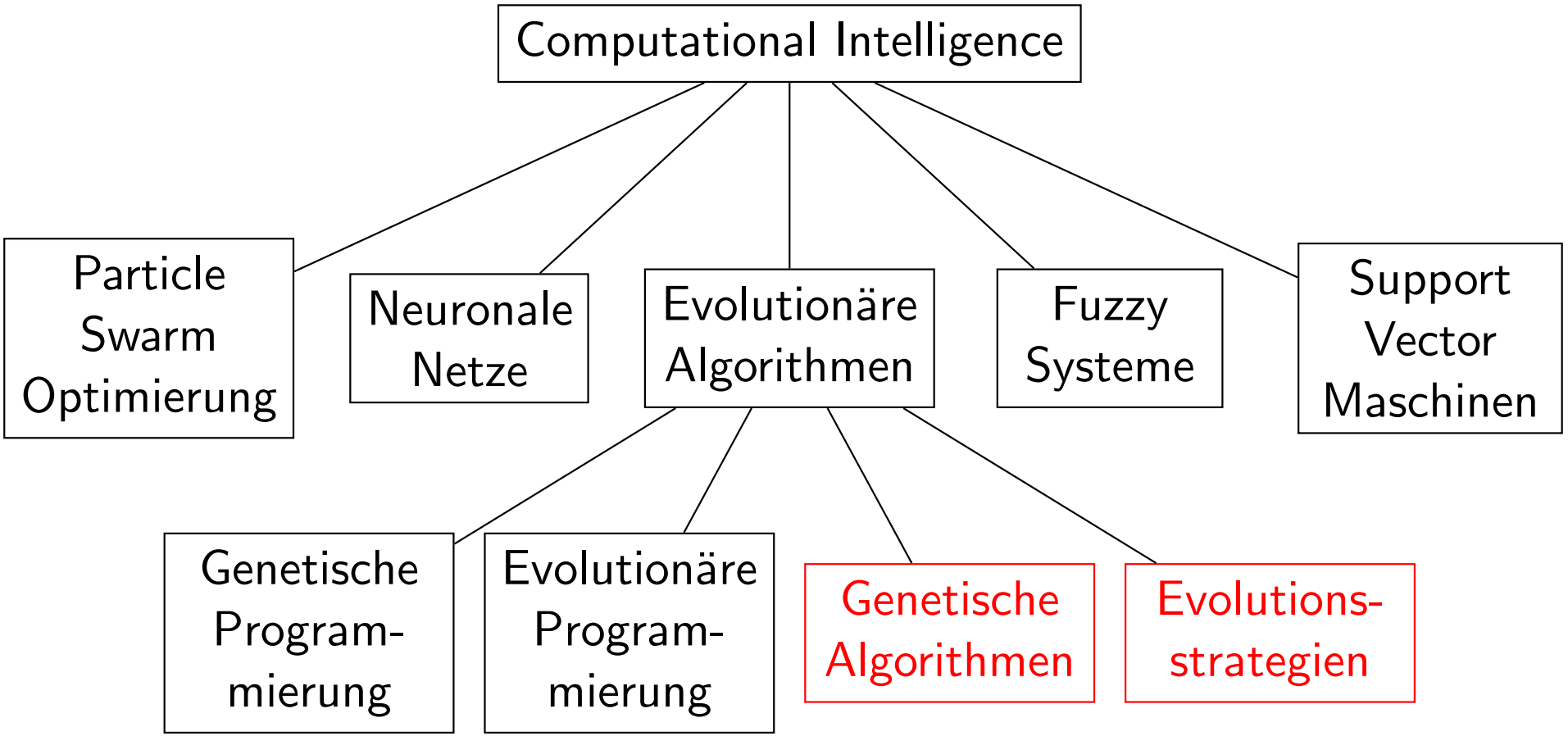
Vorlesungsplanung - Termine (Änderungen vorbehalten)

1. 21.02.2012: Einkriterielle Evolutionäre Optimierung I (CF)
2. 28.02.2012: Einkriterielle Evolutionäre Optimierung II (CF)
3. 06.03.2012: Test (1+2), Mehrkriterielle Evolutionäre Optimierung I (CF)
4. 13.03.2012: Statistische Lerntheorie I (JP)
5. 20.03.2012: Statistische Lerntheorie II (JP)
6. 27.03.2012: Test (4+5), Neuronale Netze (JP)
7. 10.04.2012: Support Vector Maschinen I (JP)
8. 02.05.2012: Mehrkriterielle Evolutionäre Optimierung II (CF)
9. 08.05.2012: Genetische Fuzzy Systeme (CF)
10. 15.05.2012: Test (3+8+9), Meta-Heuristiken (ACO, PSO) (CF)
11. 22.05.2012: Simulated Annealing und andere Suchmethoden (CF)
12. 29.05.2012: Support Vector Maschinen II (JP)
13. 05.06.2012: Test (6+7+12), Clustering (JP)
14. 12.06.2012: Lernen und Spieltheorie (JP)
15. 26.06.2012: 1. Termin mündliche Prüfungen
16. 03.07.2012: 2. Termin mündliche Prüfungen

Lernziele der heutigen Vorlesung

- Einordnung der Einkriteriellen Evolutionären Algorithmen im Bereich Computational Intelligence
- Einführung des Evolutionären Zyklus
  - Erlernen des Ablaufs
  - Erlernen der Kodierungsvarianten
  - Erlernen der evolutionären Operatoren
- Implementierung eines einfachen Genetischen Algorithmus

Klassifikation (unvollständig)



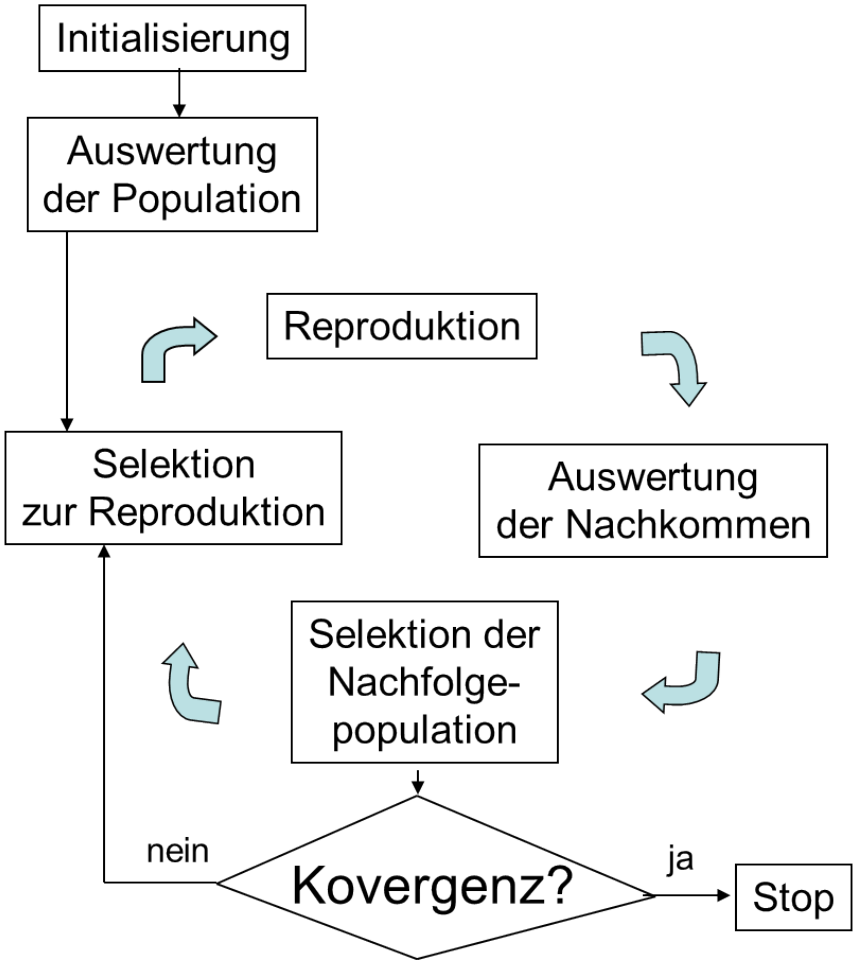
Evolutionäre Algorithmen (EA)

- Genetische Programmierung (Koza, Banzhaf)
  - Ist ein genetischer Algorithmus angewandt auf Computer Programme um effiziente und sinnvolle Programme zu entwickeln (Programme sind meist Entscheidungsbäume)
- Evolutionäre Programmierung (Fogel)
  - Mutationsbasierter EA angewandt auf diskrete Suchräume (hat Ähnlichkeit zu Evolutionären Strategien)
- Genetische Algorithmen (Goldberg, Holland)
  - In den meisten Fällen für diskrete Suchräume
- Evolutionsstrategien (Schwefel, Rechenberg)
  - In den meisten Fällen für kontinuierliche Suchräume

Allgemeine Vor- und Nachteile

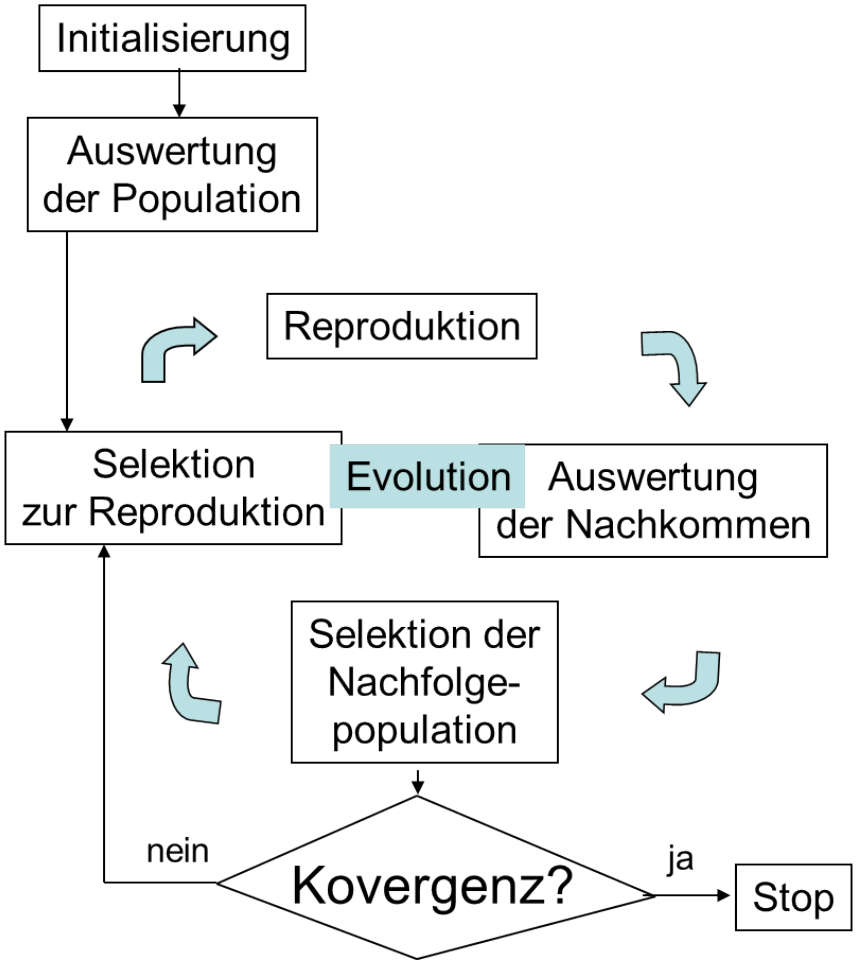
Vorteile	**Nachteile**
• Allgemein anwendbar, insbesondere wenn kaum Problemwissen vorhanden ist • Leicht an andere Probleme anpassbar • Kann interaktiv genutzt werden • Manche Methoden bieten Selbstadaption	• Keine Garantie dass eine optimale Lösung in gegebener Zeit gefunden wird • Kein vollständiger theoretischer Hintergrund • Parameteranpassung mitunter durch ,,Trial and Error"

Allgemeiner Ablauf



- **Initialisierung**
  - Kodierung
- **Evaluation**
  - Fitnesswert
- **Reproduktion**
  - Rekombination, Mutation
- **Konvergenz**
  - z.B. fixe Generationsanzahl

Allgemeiner Ablauf



- **Initialisierung**
  - Kodierung
- **Evaluation**
  - Fitnesswert
- **Reproduktion**
  - Rekombination, Mutation
- **Konvergenz**
  - z.B. fixe Generationsanzahl

Rahmenparameter

- Objektparameter (Problemkodierung): $\vec{o_k}$
- Strategieparameter (Mutation/ Rekombination/ Selektion): $\vec{s_k}$
- Fitness: $F(\vec{o_k})$
- Populationsgrösse zum Zeitpunkt t: $\mu = |P_{t,\mu}|$
- Anzahl Nachkommen: λ
- Anzahl Eltern bei der Rekombination: ρ
- Maximale Anzahl von Generationen pro Individuum: κ

Vorteile	**Nachteile**
	Genetische Algorithmen
Individuen	$a_k = (\vec{o_k}, F(\vec{o_k}))$
Populationsgrösse / Nachkommen	$\mu = \lambda$
max Anzahl Generationen pro Individuum	$\kappa = 1$
Strategieparameter	fix

Griechische Buchstaben

A	α	Alpha
B	β	Beta
Γ	γ	Gamma
Δ	δ	Delta
E	ϵ, ε	Epsilon
Z	ζ	Zeta
H	η	Eta
Θ	θ, ϑ	Theta
I	ι	Iota
K	κ, $\varkappa$	Kappa
Λ	λ	Lambda
M	μ	My

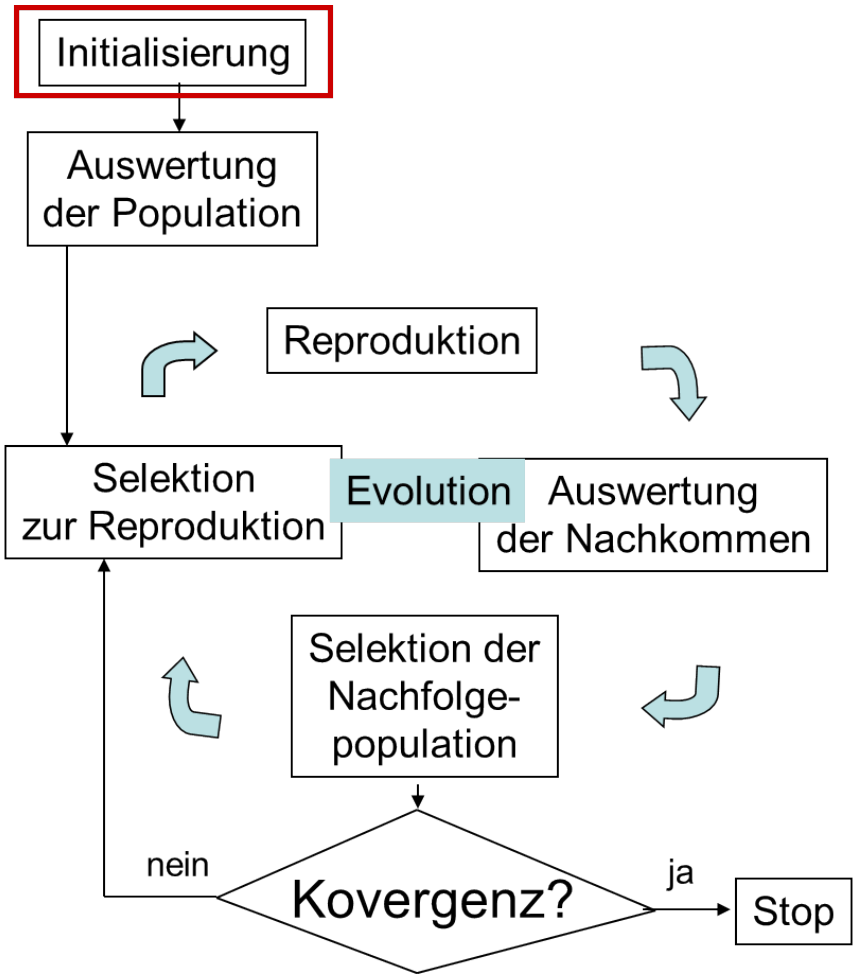
N	ν	Ny
Ξ	ξ	Xi
O	o	Omikron
Π	π, ϖ	Pi
P	ρ, ϱ	Rho
Σ	σ, ς	Sigma
T	τ	Tau
Y	υ	Ypsilon
Φ	ϕ, φ	Phi
X	χ	Chi
Ψ	ψ	Psi
Ω	ω	Omega

Übung

Übung

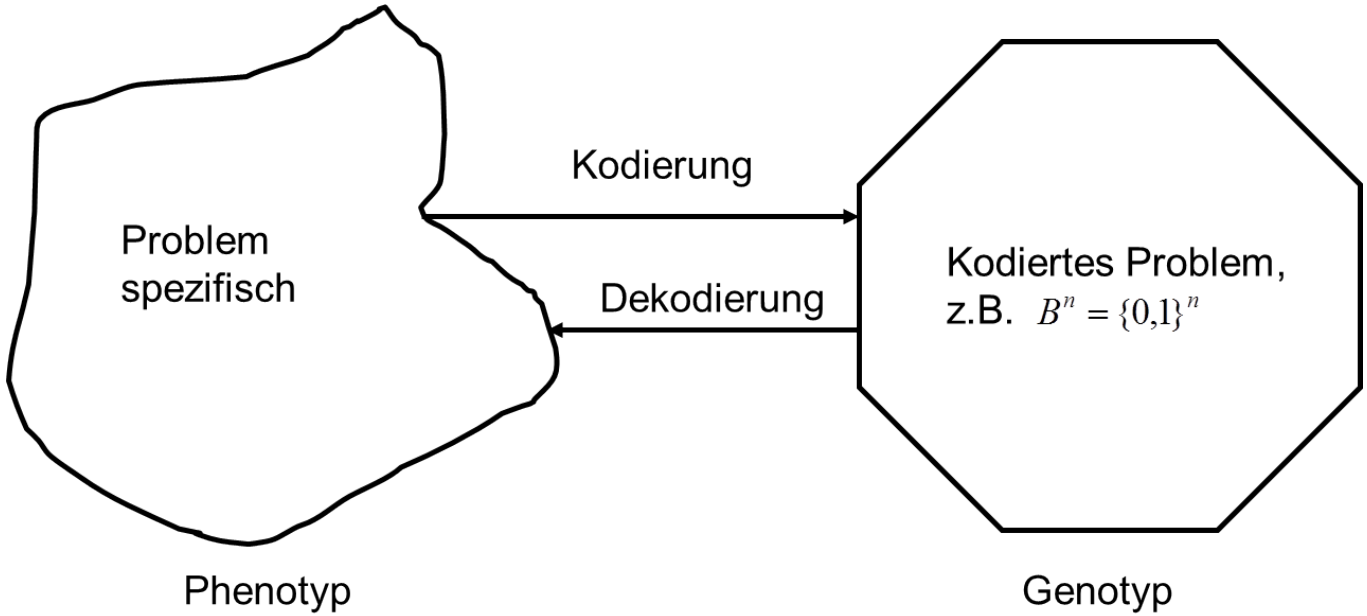
- Die Anzahl der Individuen in der letzten Population eines Genetischen Algorithmuses war 32. Bestimmen Sie λ!
- Wie gross ist das grösste k?

Allgemeiner Ablauf - Initialisierung



- **Initialisierung**
  - Kodierung
- **Evaluation**
  - Fitnesswert
- **Reproduktion**
  - Rekombination, Mutation
- **Konvergenz**
  - z.B. fixe Generationsanzahl

Phenotyp - Genotyp



Typische Kodierungen:

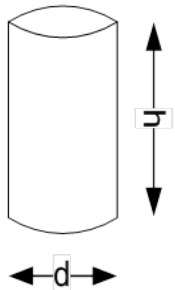
- Binär: $B^n = \{0,1\}^n$
- Reell: $\mathbb{R}^n$
- Sequenzen: $S_n = \{\pi | \pi \text{ ist Permuation auf } \{1,2,\cdots,n\}\}$

Beispiel Binäre Kodierung

Das Problem

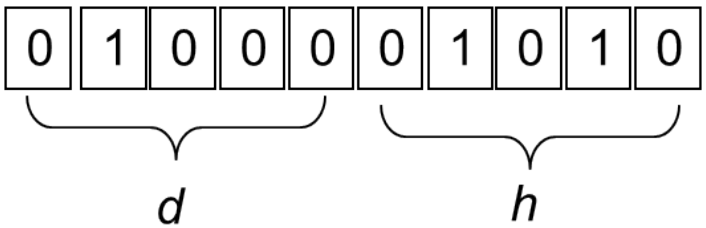
- Minimiere: $f(d,h) = \left(\frac{\pi \cdot d^2}{2} + \pi \cdot d \cdot h\right)$
- Nebenbedingung: $g(d,h) = \frac{\pi \cdot d^2 \cdot h}{4} \geq 300$
- Variablen:

$$\begin{aligned} d_{min} &\leq d \leq d_{max} \\ h_{min} &\leq h \leq h_{max} \end{aligned}$$



- Binäre Kodierung: $d(0-31)$, $h(0-31)$

Beispiel: $(d,h) = (8,10)cm$



Übung

Übung

Wie viele Bits werden benötigt, wenn d und h nicht in cm, sondern in mm angegeben werden?

Übung

Übung

- Wie viele Bits werden benötigt, wenn die Intervalle (in cm) verändert werden zu:
  - $d \in [0, 15]$
  - $h \in [1, 32]$?
- Spezifizieren Sie eine geeignete reelwertige Kodierung!

Binäre Kodierung

Annahmen:

- Phenotyp - Raum der n Parameter: $\prod_{j=1}^{n} [u_j, o_j] \subseteq \mathbb{R}^n$
- Genotyp - Raum: $\{0,1\}^l$; l - Gesamtlänge der binären Kodierung
- Dekodierungsfunktion:
  $h(\vec{a}) = (h_1(a_1, a_2, \cdots, a_k), \cdots, h_n(a_{(l-m)}, a_{(l-m+1)}, \cdots, a_l))$

Dekodierung:

- $h_i(a_{i_1}, a_{i_2}, \cdots, a_{i_{l_x}}) = u_i + \frac{o_i - u_i}{2^{l_x}-1} \cdot \left(\sum_{j=0}^{l_x-1} a_{i_{(j+1)}} \cdot 2^j \right)$

Übung

Übung

Gegeben ist folgendes Problem:

- $-5 \leq x \leq 10$
- $0,001 \leq y \leq 0,005$
- $100 \leq z \leq 1000$

Wie viele Bits sind notwendig, wenn:

1. $\Delta x = 0,01$, $\Delta y = 0,00001$, $\Delta z = 10$
2. $\Delta x = 0,1$, $\Delta y = 0,000001$, $\Delta z = 1$

Übung

Übung

Implementieren Sie (allein oder mit einem anderen Studierenden) in einer Sprache Ihrer Wahl:

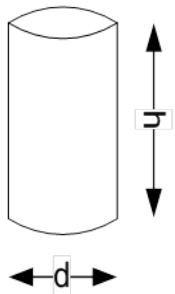
- eine Liste / einen Vektor von Individuen
- Initialisierung der Listen / Vektoren mit 30 Individuen (binär) mit zufälligen Werten (random), wobei die Struktur (aus Zeitgründen) hart kodiert sein kann
- eine Methode, die die binäre Darstellung eines Individuums in die reelwertige Darstellung übersetzt

Reelwertige Kodierung

Das Problem

- Minimiere: $f(d,h) = \left(\frac{\pi \cdot d^2}{2} + \pi \cdot d \cdot h\right)$
- Nebenbedingung: $g(d,h) = \frac{\pi \cdot d^2 \cdot h}{4} \geq 300$
- Variablen:

$$\begin{aligned} d_{min} &\leq d \leq d_{max} \\ h_{min} &\leq h \leq h_{max} \end{aligned}$$

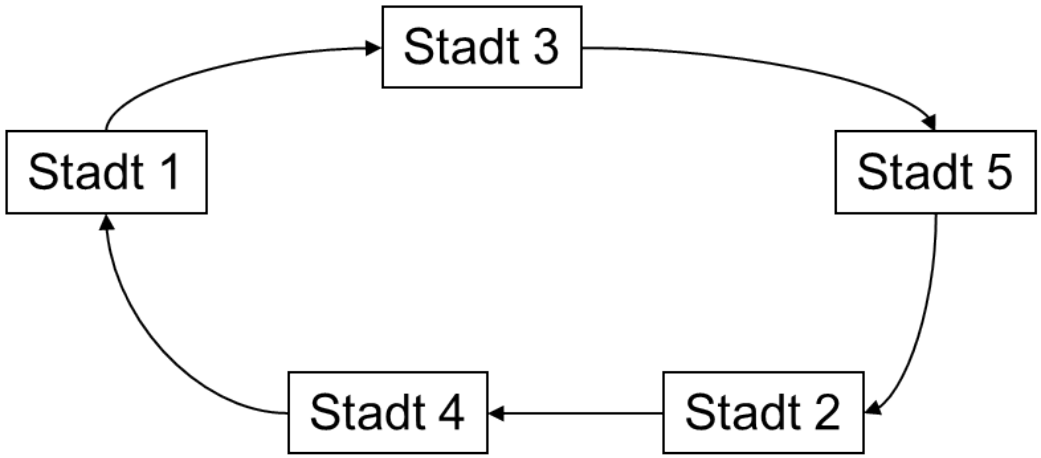


- Reelwertige Kodierung: (d,h), $d \in \mathbb{R}$, $h \in \mathbb{R}$

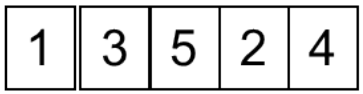
Beispiel: $(d,h) = (4,06; 10,05)cm$

Sequenz - Kodierungen

Traveling Salesperson Problem



Sequenz: $S_1 = (1, 3, 5, 2, 4)$ oft auch:

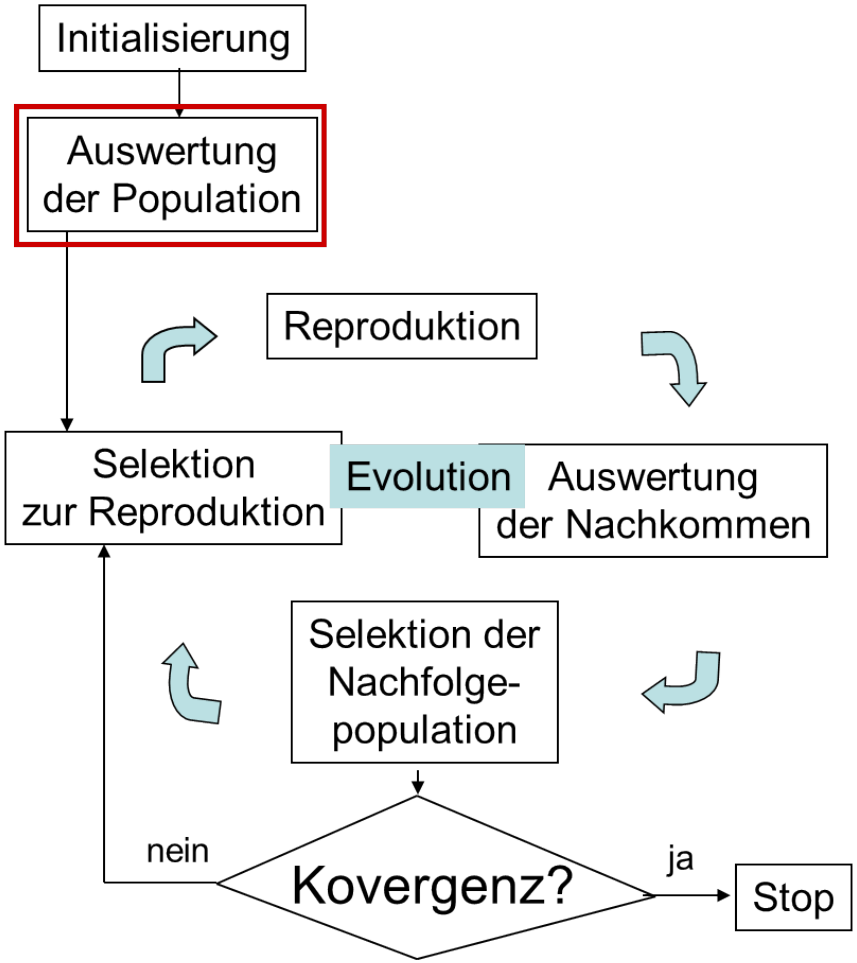


Zusammenfassung Intitialisierung

Sie sollten jetzt wissen:

- Was versteht man unter Initialisierung eines Evolutionären Algorithmuses?
- Den Unterschied zwischen Phenotyp und Genotyp erläutern können.
- Welche Kodierungsmöglichkeiten gibt es?
- Beispiele für diese Kodierungsmöglichkteiten nennen können und dies auch implementieren können.

Auswertung der Population



- **Initialisierung**
  - Kodierung
- **Evaluation**
  - Fitnesswert
- **Reproduktion**
  - Rekombination, Mutation
- **Konvergenz**
  - z.B. fixe Generationsanzahl

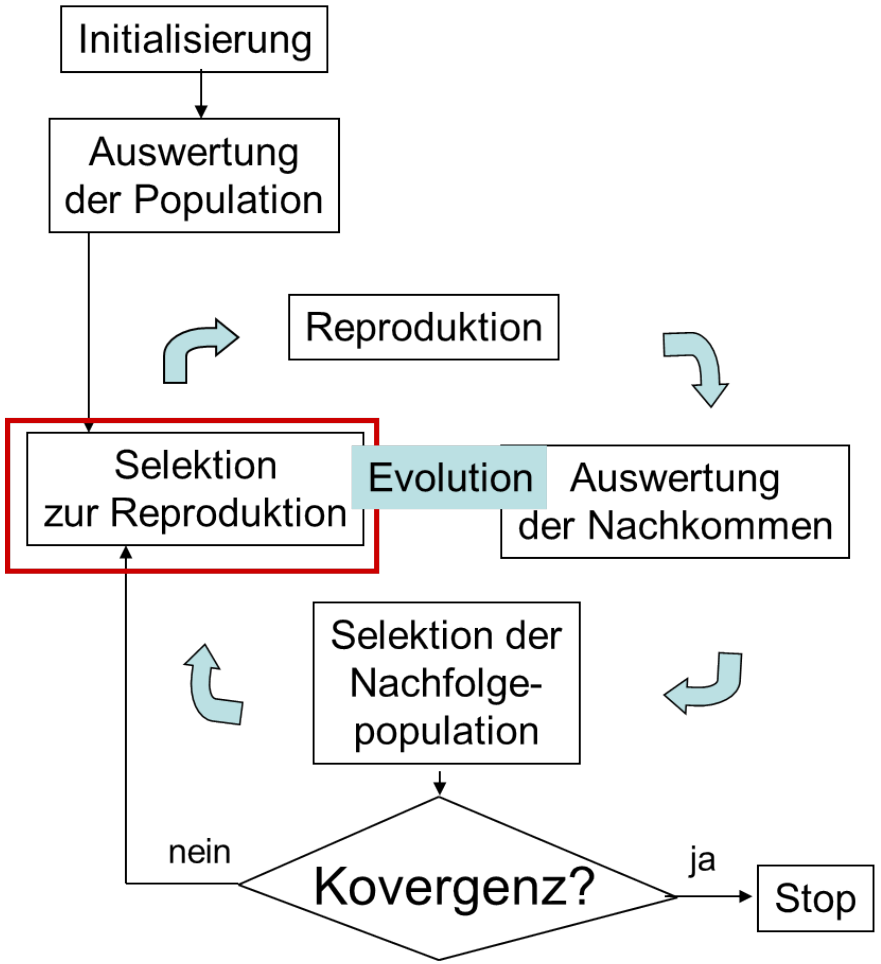
Übung

Übung

Implementieren Sie:

- 1 Methode, die jedes Individuum auswertet **und** die Nebenbedingung überprüft.
- Diese Methode sollte zwei Rückgabewerte haben:
  - Funktionswert
  - Boolean-Wert, der angibt, ob die Nebenbedingung erfüllt oder verletzt ist.

Selektion zur Reproduktion



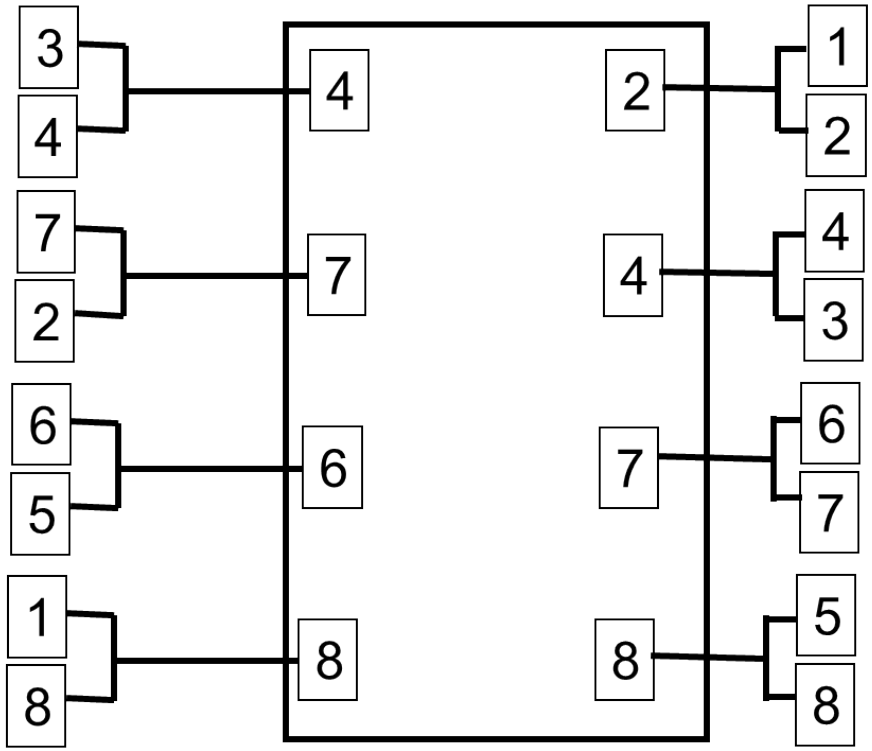
- **Initialisierung**
  - Kodierung
- **Evaluation**
  - Fitnesswert
- **Reproduktion**
  - Rekombination, Mutation
- **Konvergenz**
  - z.B. fixe Generationsanzahl

Selektionsoperatoren

- Ziele
  - Erzeugung von Kopien guter Lösungen und Eliminierung schlechter Lösungen
  - Erhaltung einer konstanten Populationsgröße (bei GAs) und eines konstanten Eltern/Nachkommen-Verhältnisses bei ES
  - Erhaltung der ,,genetischen Streuung"
- häuige Operatoren
  - Turnier-Selektion
  - Fitness-proportionale Selektion (Roulette-Wheel-Selection)
  - Rang-basierte Selektion

Turnier-Selektion

Annahme: die 8 Individuen haben die folgende Reihenfolge gemäss ihrer Fitnesswerte (niedrig zu hoch): 1,2,3,4,5,6,7,8.



Resultierender Pool: 2,4,4,6,7,7,8,8

Übung

Übung

- Nehmen Sie die Fitness der folgenden Lösungen a_i an: $F(a_1) = 23$; $F(a_2) = 45$; $F(a_3) = -1$; $F(a_4) = 9$; $F(a_5) = 9$; $F(a_6) = 78$; $F(a_7) = 0$; $F(a_8) = 234$
- Nehmen Sie die Turnierselektionen der vorangegangen Folie (die Paarungen) und eine Maximierung an. Welche Individuen a_i befinden sich im resultierenden Pool?

Fitness-proportionale Selektion

- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0

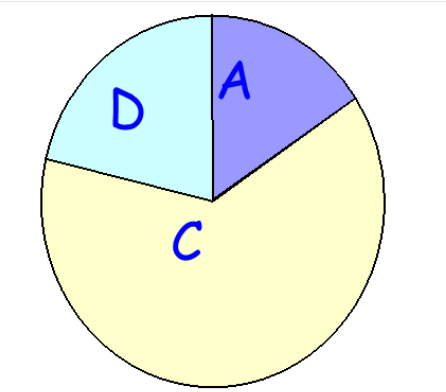
- Evaluierung gemäss Fitness (anderes Beispiel)

	a	b	Fitness	% of Total
A	23	28	0.00071	0.1588
B	1	11	0.1632(0)	0
C	8	11	0.0028	0.6264
D	8	10	0.00096	0.2148

Fitness-proportionale Selektion

- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0



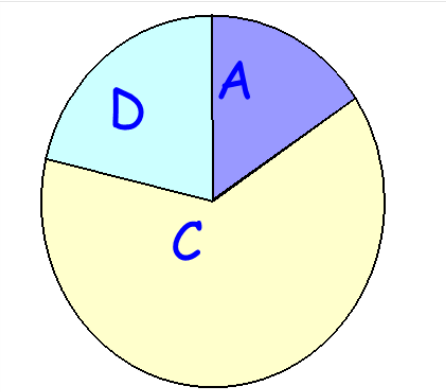
- Evaluierung gemäss Fitness (anderes Beispiel)

	a	b	Fitness	% of Total
A	23	28	0.00071	0.1588
B	1	11	0.1632(0)	0
C	8	11	0.0028	0.6264
D	8	10	0.00096	0.2148

Fitness-proportionale Selektion

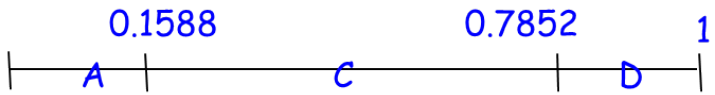
- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0



- Evaluierung gemäss Fitness (anderes Beispiel)

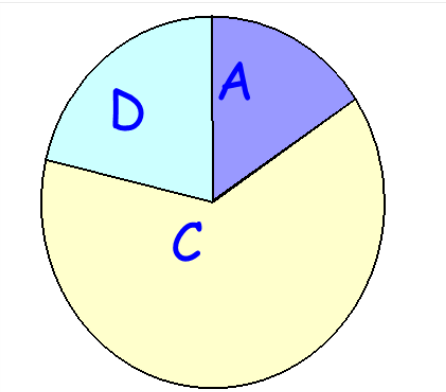
	a	b	Fitness	% of Total
A	23	28	0.00071	0.1588
B	1	11	0.1632(0)	0
C	8	11	0.0028	0.6264
D	8	10	0.00096	0.2148



Fitness-proportionale Selektion

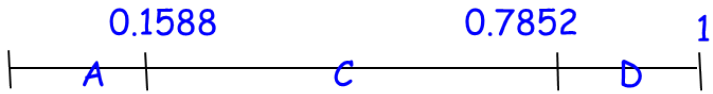
- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0



- Evaluierung gemäss Fitness (anderes Beispiel)

	a	b	Fitness	% of Total
A	23	28	0.00071	0.1588
B	1	11	0.1632(0)	0
C	8	11	0.0028	0.6264
D	8	10	0.00096	0.2148

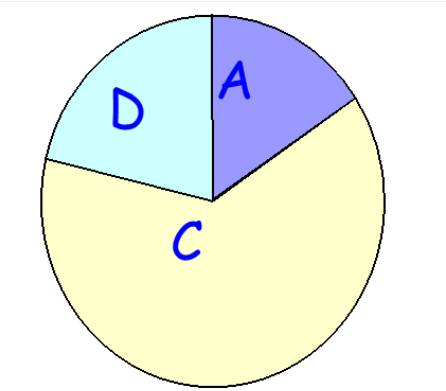


rand	
0.4186	C
0.0196	A
0.5252	C
0.8462	D

Fitness-proportionale Selektion

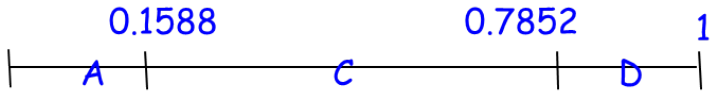
- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0

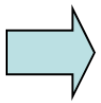


- Evaluierung gemäss Fitness (anderes Beispiel)

	a	b	Fitness	% of Total
A	23	28	0.00071	0.1588
B	1	11	0.1632(0)	0
C	8	11	0.0028	0.6264
D	8	10	0.00096	0.2148



rand	
0.4186	C
0.0196	A
0.5252	C
0.8462	D



{A,C,C,D}

Übung

Übung

Ausgehend von den Lösungen A bis D auf der letzten Folie:

- Welche Elemente werden in den nachfolgenden Pool übernommen, wenn ein Zufallsgenerator folgende Zahlen liefert: 0,1; 0,567; 0,11; 0,87; 0,34?

Rang-basierte Selektion

- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0

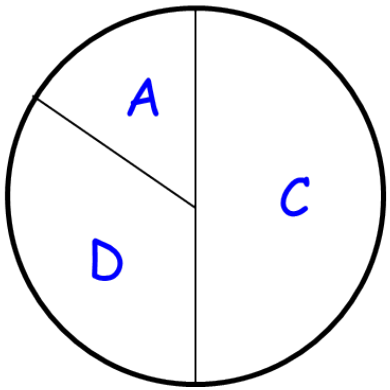
- Evaluierung gemäss Fitness (anderes Beispiel)

	a	**b**	**Fitness**	**Rang**
A	23	28	0.00071	1
B	1	11	0.1632(0)	0
C	8	11	0.0028	3
D	8	10	0.00096	2

Rang-basierte Selektion

- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0



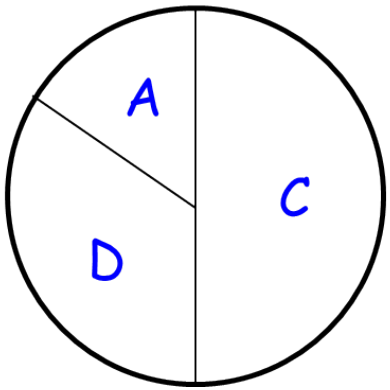
- Evaluierung gemäss Fitness (anderes Beispiel)

	a	b	Fitness	Rang
A	23	28	0.00071	1
B	1	11	0.1632(0)	0
C	8	11	0.0028	3
D	8	10	0.00096	2

Rang-basierte Selektion

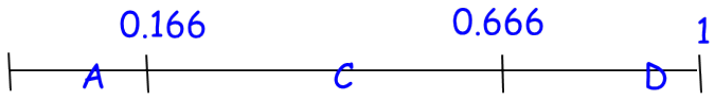
- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0



- Evaluierung gemäss Fitness (anderes Beispiel)

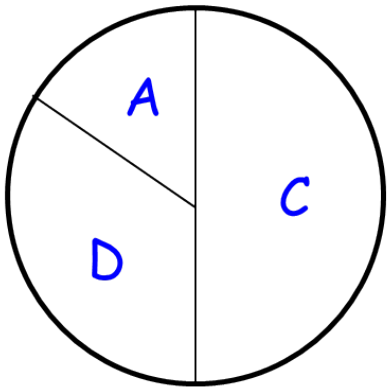
	a	b	Fitness	Rang
A	23	28	0.00071	1
B	1	11	0.1632(0)	0
C	8	11	0.0028	3
D	8	10	0.00096	2



Rang-basierte Selektion

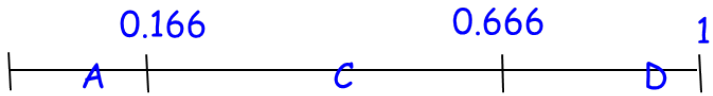
- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0



- Evaluierung gemäss Fitness (anderes Beispiel)

	a	b	Fitness	Rang
A	23	28	0.00071	1
B	1	11	0.1632(0)	0
C	8	11	0.0028	3
D	8	10	0.00096	2

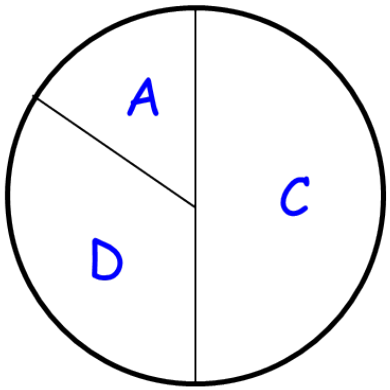


rand	
0.4186	C
0.0196	A
0.5252	C
0.8462	D

Rang-basierte Selektion

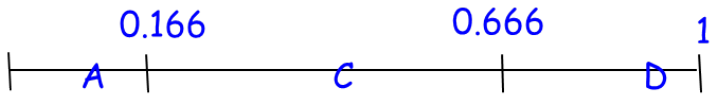
- Initiale Population

	a					b				
A	1	0	1	1	1	1	1	1	0	0
B	0	0	0	0	1	0	1	0	1	1
C	0	1	0	0	0	0	1	0	1	1
D	0	1	0	0	0	0	1	0	1	0

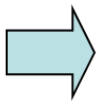


- Evaluierung gemäss Fitness (anderes Beispiel)

	a	b	Fitness	Rang
A	23	28	0.00071	1
B	1	11	0.1632(0)	0
C	8	11	0.0028	3
D	8	10	0.00096	2



rand	
0.4186	C
0.0196	A
0.5252	C
0.8462	D



{A,C,C,D}

Übung

Übung

- Bei welcher der drei vorgestellten Selektionsoperatoren bleibt die beste Lösung auf jeden Fall erhalten?
- Bei welcher der drei vorgestellten Selektionsoperatoren geht die schlechteste Lösung auf jeden Fall verloren?
- Wie viele Kopien der besten Lösung bleiben bei der Turnierselektion am Ende übrig?
- Garantiert die Fitness-proportionale Selektion die Wahl des besten Individuums?
- Garantiert die Rang-basierte Selektion die Wahl des besten Individuums?

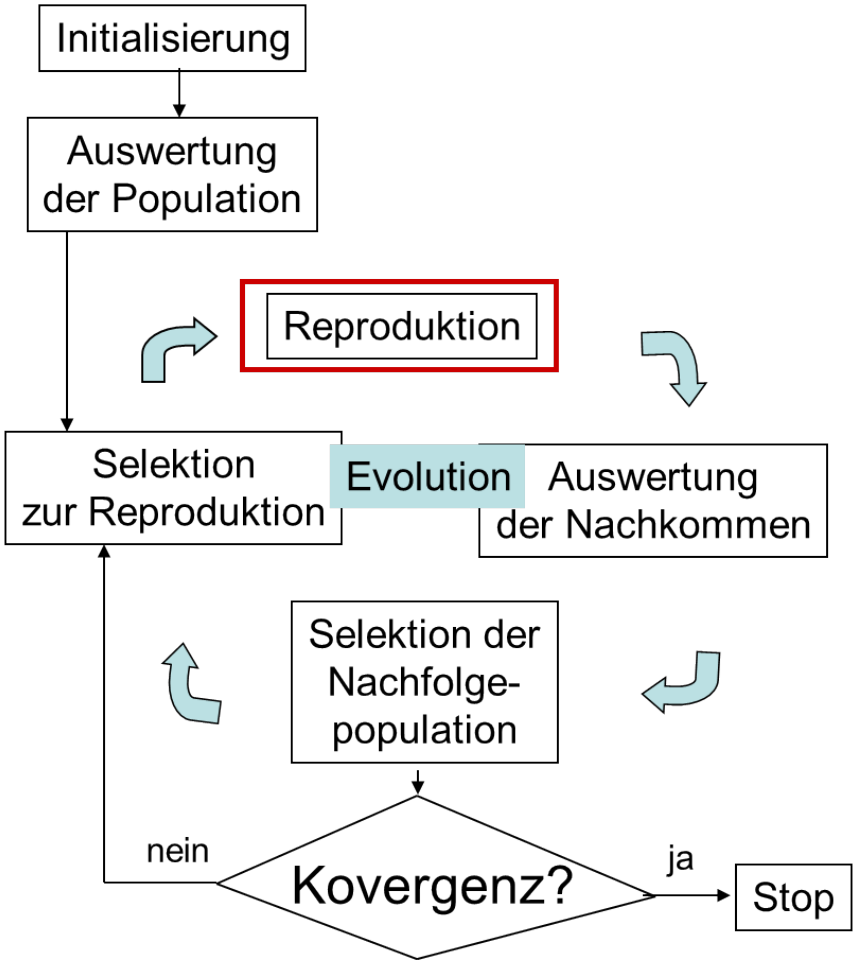
Übung

Übung

Implementieren Sie innerhalb Ihres bisherigen Programms:

- 1 Rang-basierte Selektion, wobei Sie annehmen, dass Individuen, die die Nebenbedingung verletzen, nicht selektiert werden können.

Allgemeiner Ablauf - Reproduktion



- **Initialisierung**
  - Kodierung
- **Evaluation**
  - Fitnesswert
- **Reproduktion**
  - Rekombination, Mutation
- **Konvergenz**
  - z.B. fixe Generationsanzahl

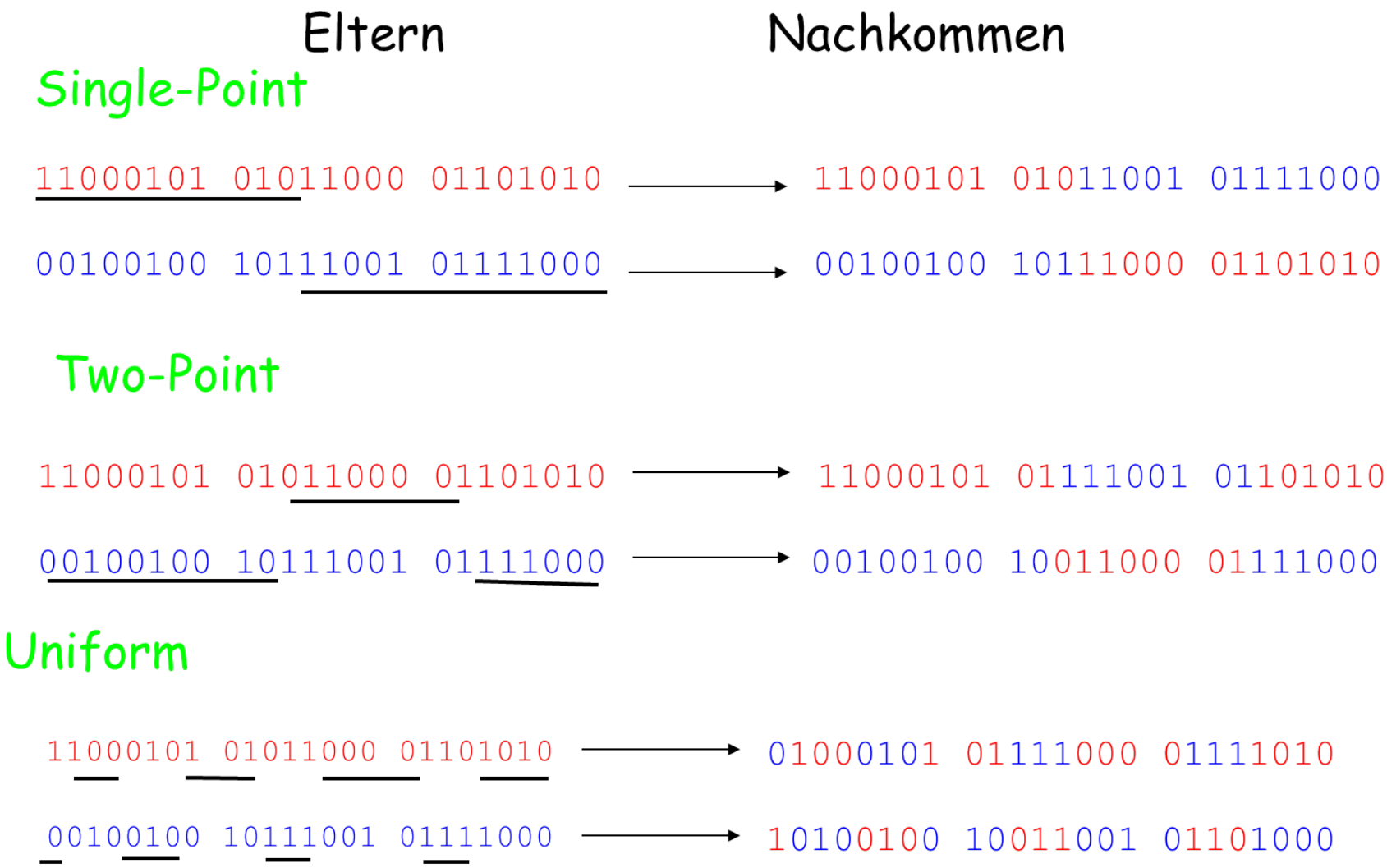
Reproduktion

- Im Allgemeinen werden dabei 2 wesentliche Operatoren genutzt
  - Rekombination
  - Mutation
- Diese sind für die 3 allgemeinen Kodierungen unterschiedlich definiert

Rekombination

- Dient der Kombination ,,guter Eigenschaften" existierender Lösungen
- Jedes zuvor für die Rekombination selektierte Individuum nimmt mit Wahrscheinlichkeit p_c an der Rekombination teil
- Jede Rekombination nutzt ρ Eltern um Nachkommen zu erzeugen
  - Genetische Algorithmen nutzen in der Regel $\rho = 2$ Eltern und generieren **2** Nachkommen
  - Evolutionsstrategien nutzen häufig mehr als $\rho > 2$ Eltern zur Erzeugung von **1** Nachkommen

Rekombination - binäre Kodierung



Übung

Übung

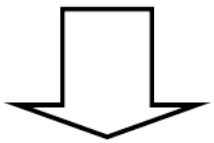
Überlegen Sie sich ein Beispiel für eine uniforme Rekombination mit $\rho = 3$!

Reelwertige Rekombination

a11	a12	a13	a14	a15	a16	a17	a18	a19
a21	a22	a23	a24	a25	a26	a27	a28	a29

...

a91	a92	a93	a94	a95	a96	a97	a98	a99



Zufällig $\rho = 3$ Lösungen ausgewählt

a11	a12	a13	a14	a15	a16	a17	a18	a19
a21	a22	a23	a24	a25	a26	a27	a28	a29
a41	a42	a43	a44	a45	a46	a47	a48	a49

Durchschnittliche Rekombination

$$ak = \frac{a1k + a2k + a4k}{3}$$

Diskrete Rekombination

a11	a42	a13	a24	a45	a46	a17	a28	a29

Reelwertige Rekombination

a11	a12	a13	a14	a15	a16	a17	a18	a19
a21	a22	a23	a24	a25	a26	a27	a28	a29

...

a91	a92	a93	a94	a95	a96	a97	a98	a99

Zufällig $\rho = 3$ Lösungen ausgewählt

a11	a12	a13	a14	a15	a16	a17	a18	a19
a21	a22	a23	a24	a25	a26	a27	a28	a29
a41	a42	a43	a44	a45	a46	a47	a48	a49

Durchschnittliche Rekombination

$$ak = \frac{a1k + a2k + a4k}{3}$$

Diskrete Rekombination

a11	a42	a13	a24	a45	a46	a17	a28	a29

Simulated Binary Crossover (SBX) - reelwertig

Wähle zufällig: $u_i \in [0,1[$ und ein $\eta_c \geq 0$: {Rechteckverteilung}

Berechne $\beta = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}}, & \text{, wenn } u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}}, & \text{sonst.} \end{cases}$

$\tilde{a}_{1i} = 0.5\left[(1+\beta)a_{1i} + (1-\beta)a_{2i}\right]$

$\tilde{a}_{2i} = 0.5\left[(1-\beta)a_{1i} + (1+\beta)a_{2i}\right]$

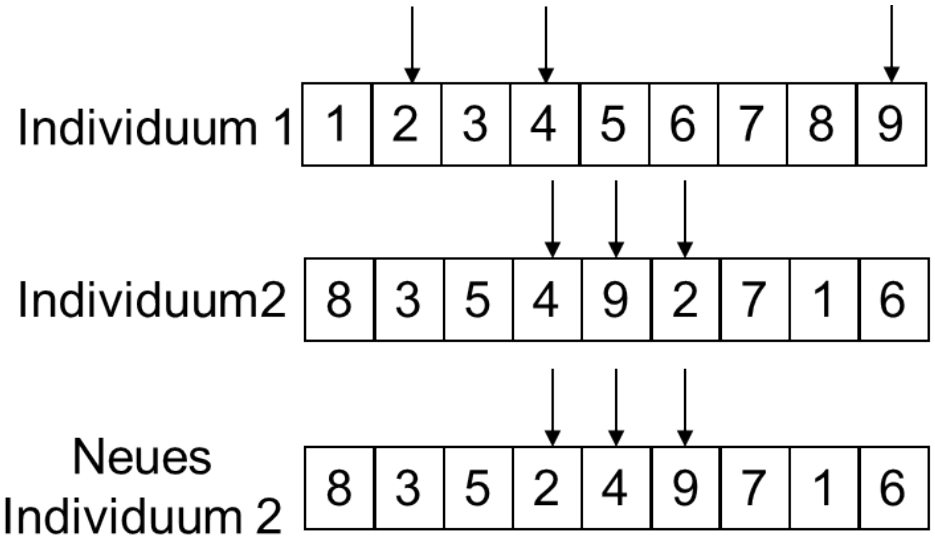
Übung

Übung

- Warum muss $u_i < 1$ sein?
- Welche Werte haben $\tilde{a}_{1i}$ und $\tilde{a}_{2i}$ falls $u_i = 0$?

Rekombination von Sequenzen

1. Zufällige Auswahl mehrerer Sequenzelemente der 1. Sequenz
2. Bestimmung der gleichen Elemente in der 2. Sequenz
3. Erzeugung eines neuen Individuums durch Umsortierung der Elemente der 2. Sequenz gemäss der Reihenfolge der 1. Sequenz



Übung

Übung

Implementieren Sie innerhalb des entstehenden Programms:

- 1 Funktion, die an 2 gegebenen binären Strings eine ,,Single-Point" Rekombination vornimmt, wobei der Single-Point zufällig gewählt wird.

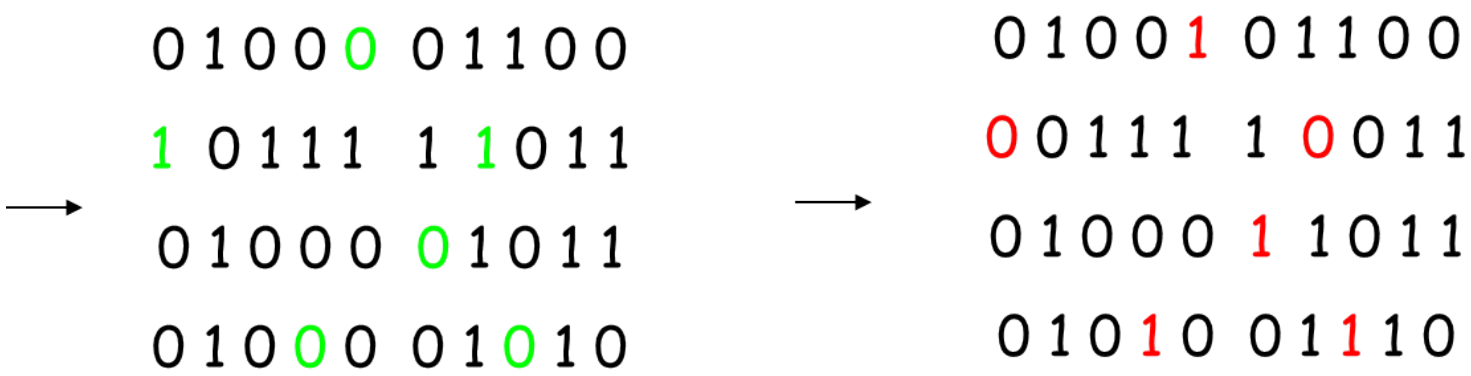
Reproduktion

- Im Allgemeinen werden dabei 2 wesentliche Operatoren genutzt
  - Rekombination
  - Mutation
- Diese sind für die 3 allgemeinen Kodierungen unterschiedlich definiert

Binäre Mutation

Wähle mit Wahrscheinlichkeit p_m ein Bit aus und invertiere es. Dabei gibt es mehrere Implementierungsvarianten:

- Bestimmung von p_m für jedes Bit einzeln
- Bestimmung des nächsten zu invertierenden Bits (dabei werden alle Bits ,,virtuell" hintereinandergelegt)



Übung

Übung

- Nehmen Sie an, dass folgender binärer String gegeben ist: 01011010110
- Ein Zufallsgenerator liefert folgende Wert: 0,1; 0,4; 0,9; 0,04; 0,56; 0,33; 0,23; 0,87; 0,56; 0,83; 0,99
- Geben Sie den resultierenden binären String nach Mutation für $p_m = 0,1$ an!

Übung

Übung

Implementieren Sie innerhalb des bereits entwickelten Codes:

- 1 Funktion, die an einem gegebenen, binären String eine Mutation mit Wahrscheinlichkeit p_m vornimmt.
- Wenden Sie diese Funktion auf alle lokalen Individuen an.

Hausaufgaben

Hausaufgaben

- Beenden Sie **alle** Beispiele, die Sie während der Vorlesung nicht fertig gestellt haben!
- Dies ist sehr wichtig, da die nachfolgende Vorlesung darauf aufbauen wird.
- Überprüfung zu Beginn der nächsten Vorlesung.