Seminar „Artificial Life und Multiangentensysteme“
Professor Dr. W. Kurth
Sommersemester 2003

Thema:

Genetische Algorithmen

Andre Eisengarten, IMT

Übersicht

- Vorwort
- Geschichte
- Grundbegriffe
- Grundaufbau eines Genetischen Algorithmus
- Codierung
- Selektion
- Rekombination

Vorwort

- viele Probleme in der Wissenschaft (speziell Informatik) sind enorm komplex und ohne Abstraktion und Optimierung kaum zu lösen
- Bsp.1: Suchen nach Lösungen in einer Vielzahl von Lösungsmöglichkeiten, z.B. Aminosäuresequenzen für ein Protein mit bestimmten Eigenschaften
- Bsp.2: Adaptivität (und Lernfähigkeit) , Anpassung eines Computerprogramms an seine „Umwelt“
- Bsp.3: Künstliche Intelligenz
- die Natur, insbesondere die Evolution bietet einfache und effiziente Methoden um solche Probleme zu lösen:

➢ Evolution sucht „selbstständig“ unter einer enormen Anzahl von Möglichkeiten nach Lösungen

➢ Evolution testet und verändert Millionen von Spezies parallel

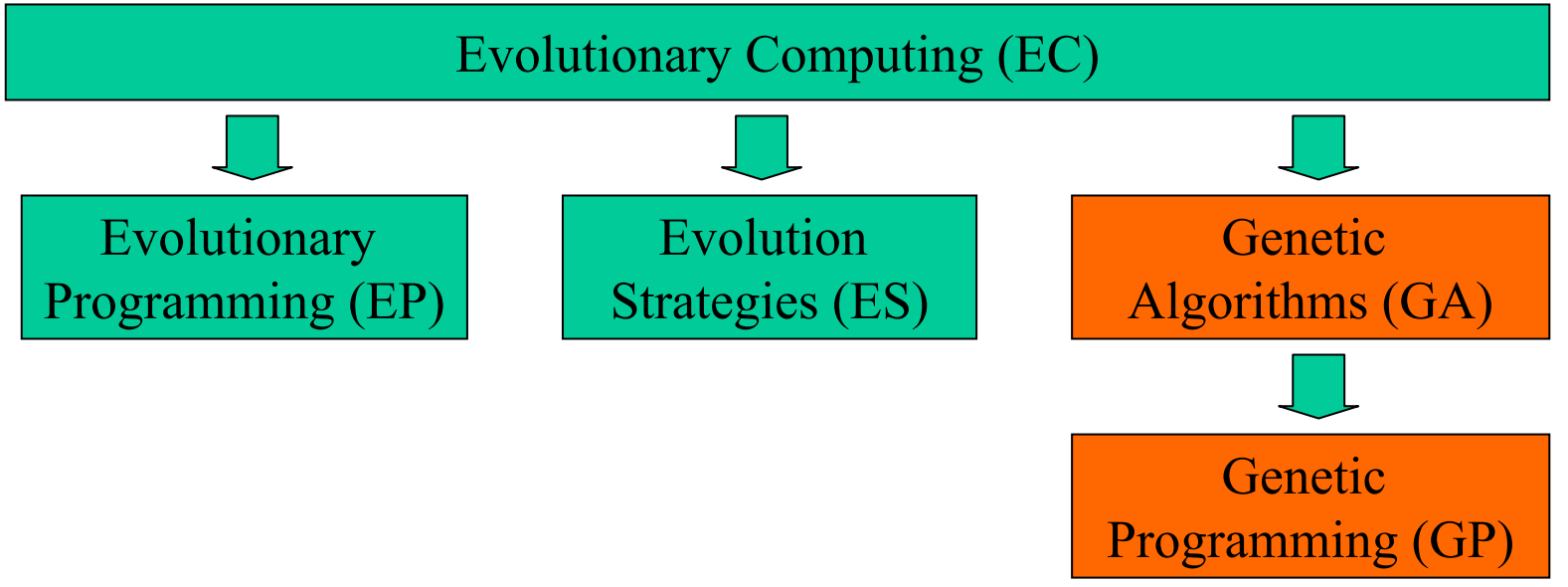
➢ Evolution bietet einfache Regeln (Mutation, Rekombination,...)

Geschichte:

- 1950 – 1960: verschiedene (Computer-) Wissenschaftler erforschen unabhängig voneinander evolutionäre Systeme
  Ziel: Evolution als Werkzeug zur Optimierung technischer Probleme
- 1960‘er Jahre: Rechenberg entwickelt die „Evolutionsstrategien“ um die Aerodynamik bei Tragflächen zu optimieren
- 1966: Fogel, Owens und Walsh bringen die „Evolutionäre Programmierung“ hervor: Erzeugung von Final-State-Maschinen durch Mutation und Selektion von Zustandsübergangsdiagrammen
- 1960‘er Jarhre: John H. Holland erfindet und entwickelt zusammen mit Kollegen und Studenten an der University of Michigan die „Genetischen Algorithmen“
  Ziel: kein Algorithmus zur Lösung von spezifischen Problemen sondern formales Studium des Phänomens der Adaption (in der Natur) und Wege, solche Mechanismen auf Computer zu portieren

Geschichte:

- 1975 John H. Holland: „Adaption in Natural and Artificial Systems“, Abhandlung über die Abstraktion der biologischen Evolution und Adaption unter GA.
- Im Gegensatz zu den anderen evolutionären Ansätzen führt Holland den Faktors Rekombination ein, eine große Innovation auf dem Gebiet der evolutionären Systeme.



Grundbegriffe

- es sei die Lösung für ein Problem durch n Parameter charakterisiert, die Werte aus einem Alphabet M annehmen können

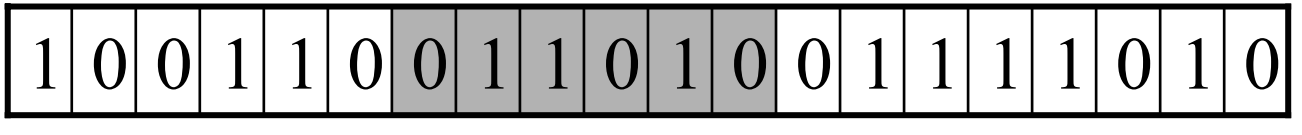
Individuum/Chromosom:

-Repräsentation einer Problemlösung durch Parameterkonfiguration

$c=(a_1\ a_2\ a_3\ ...\ a_n)$ mit $a_i \in M; \forall\ i \in \{1...n\}$

-String (bzw. Vektor) aus m endlichen Genen

Bsp: binär codierte Lösung mit M={0,1}



Grundbegriffe

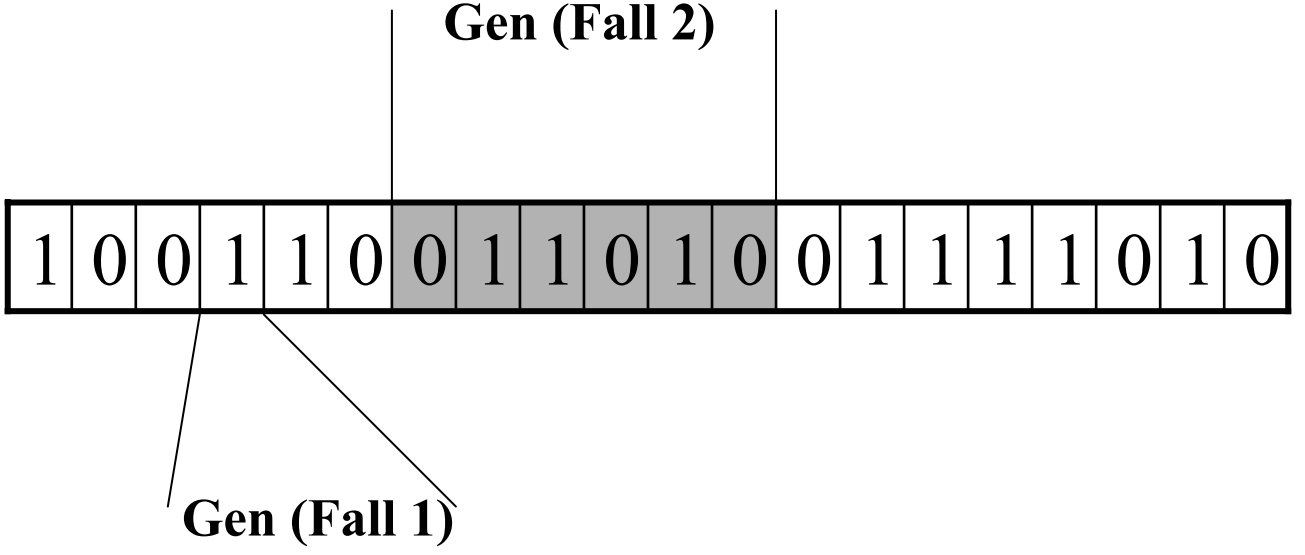
Gen:

-Fall 1: ein Parameter a_i aus der Parameterkonfiguration C oder

-Fall 2: eine kurze Parametersequenz

$g=(a_x\ a_{x+1}\ ...\ a_{x+y})$ mit $a_i\ \varepsilon\ M$; $x\ \varepsilon\ \{1...n\}\ \wedge (y+x)\leq n$

-Repräsentation eines Teils der Problemlösung (z.B. eine Variable)



Grundbegriffe

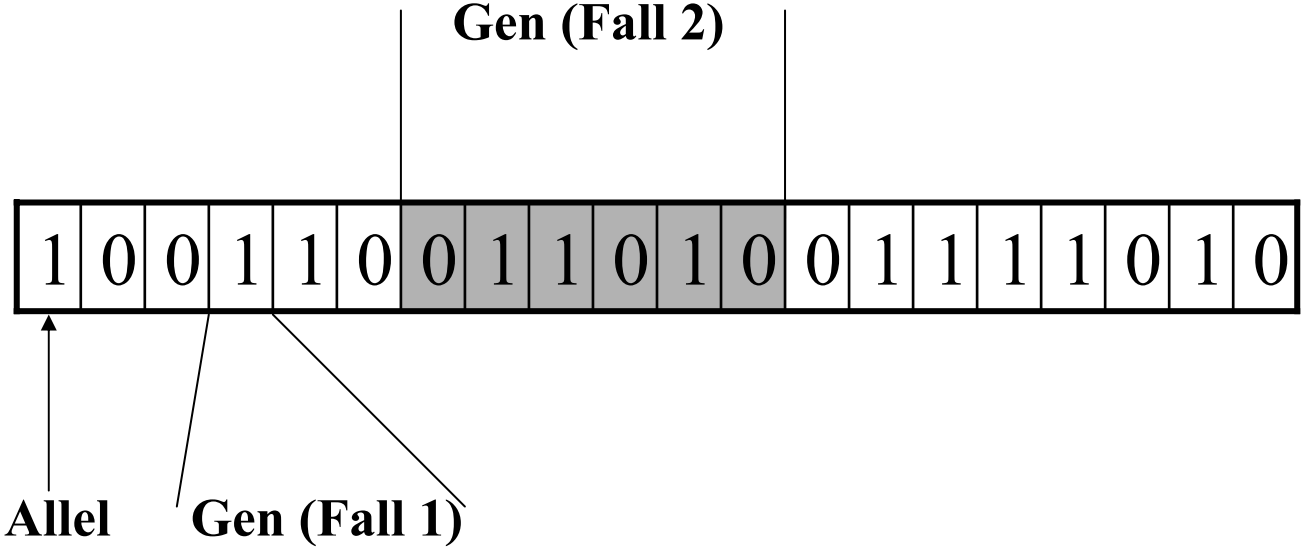
Allel:

-Wert eines Parameters a_i bzw. eines Gens, also ein Wert aus der Menge M

Länge eines Chromosoms:

-Anzahl der Parameter n

-Länge des Strings



Grundbegriffe

Genotyp:

-codierte Problemlösung (abhängig von der Codierungsmethode)

Phänotyp:

-decodierte Problemlösung (in der Biologie die konkrete Ausprägung eines Gens, z.B. die Augenfarbe)

Population:

-eine Menge gleichartiger Individuen (die Werte der Parameter sind alle Elemente des selben Alphabets M, die Länge der Chromosomen ist in den meisten GA ebenfalls gleich)

Fitness/Fitnessfunktion:

-die Fitnessfunktion ermittelt die Güte (Fitness) eines Individuums in Bezug auf die Lösung des gestellten Problems

Grundbegriffe

Selektion:

-Auswahl von Eltern (-paaren) in Bezug auf ihre Fitness (evtl. auch nach stochastischen Verfahren)

Rekombination:

-Erzeugung neuer Nachkommen mit Hilfe einer Crossover-Funktion

Mutation:

-stochastische Veränderung einzelner Gene im Individuum

Schema:

-ist ein String s mit der Länge n, der aus Elementen aus M und zusätzlichen „don‘t care“ Werten besteht

-ein Individuum c gehört zu einem Schema, falls alle Allele in s, die nicht „don‘t care“ sind, in c die selbe Ausprägung haben

Grundbegriffe

Schema:

$c=(a_1\ a_2\ a_3\ ...\ a_n)$ mit $a_i \in M; \forall\ i \in \{1...n\}$

$s=(b_1\ b_2\ b_3\ ...\ b_n)$ mit $b_i \in M \cup \{*\}; \forall\ i \in \{1...n\}$

$c_j \in s: a_i=b_i \vee b_i=* \ \forall\ i \in \{1...n\}$

Beispiel:

Schema 10*00* steht für die Chromosomen

10**0**00**0,** 10**0**00**1**, 10**1**00**0** und 10**1**00**1**

Ein Schema fasst somit Individuen („Instanzen“ des Schemas) mit n gleichen Eigenschaften zusammen. Die Fitness eines Schemas wird aus dem Durchschnit der Fitness der enthaltenden Individuen gebildet. Liegt diese über dem Durchschnitt der gesamten Population, so sind mit hoher Wahrscheinlichkeit gerade die Allele für die hohe Fitness verantwortlich, die im Schema nicht don‘t care sind. Schemata können somit Gruppen von „fitten“ Individuen zusammenfassen.

Grundaufbau eines GA

1. **Codierung** des zu optimierenden Problems als Individuum
2. Erzeugung einer (zufällig generierten) **Grundpopulation** von Individuen
3. **Selektion** der zu rekombinierenden Eltern in Abhängigkeit ihrer **Fitness**
4. **Rekombination** der selektierten Eltern (Erzeugung von Nachkommen)
5. **Mutation** der Nachkommen
6. Wiederholen von Punkt 3 bis 5, bis ein **Abbruchkriterium** erreicht ist

Codierung

- Die Codierung, also Abbildung der Lösungskandidaten auf ein Individuum ist ein zentraler Faktor für den Erfolg eines GA.

Codierung in Bezug auf die Repräsentation

- **Binäre Codierung:**

  -Ursprüngliche, durch Holland verwendete Form der Codierung mit fester Reihenfolge und fester Anordnung
  -Die Parameter für die Lösung werden auf Dualzahlen abgebildet
  -Für größere Probleme meist ungeeignet („unnatürlich“)
  -Anzahl der Schemata zu einem Individuum im Vergleich zu höherwertiger Codierung, also größerem Alphabet M jedoch höher (binärer 8-bittiger Wert (8 Gene) benötigt dezimal 3 Gene, Anzahl der Schemata 2^8 vs. 2^3
  -Problem der Hamming-Distanz zwischen Variablen:

Codierung

Codierung in Bezug auf die Repräsentation

- **Binäre Codierung:**

•Rechenbergs Prinzip der strengen Kausalität: „kleine Ursachen sollten nur eine kleine Wirkung nach sich ziehen“
•Die Hamming-Distanz zwischen aufeinander folgenden Bit-Werten ist jedoch unterschiedlich, ein Wert kann somit nicht in jedem Fall durch Mutation einer Bitstelle auf x+1 bzw. x-1 überführt werden
➢Daher wird oft der Graycode zur Codierung benutzt, die Hamming-Distanz zwischen aufeinander folgenden Werten ist immer 1.

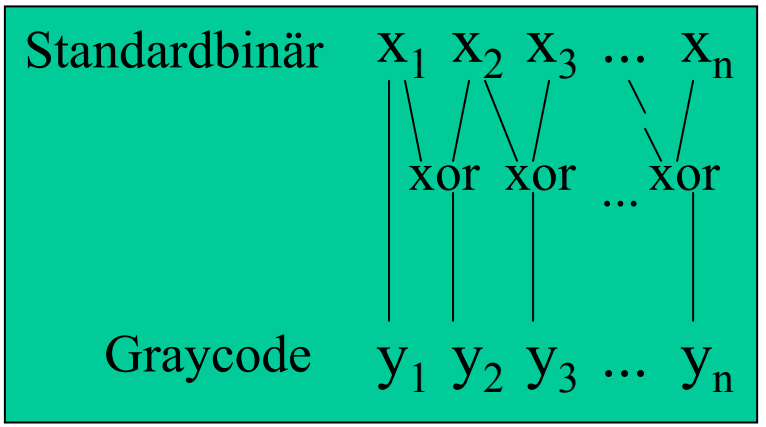
Dezimal	Standard-Binär	Distanz	Graycode	Distanz
0	000		000	
1	001	1	001	1
2	010	2	011	1
3	011	1	010	1
4	100	3	110	1

Codierung

Codierung in Bezug auf die Repräsentation

- **Binäre Codierung:**

  •Bildung des Graycode:



Codierung

Codierung in Bezug auf die Repräsentation

- **Vielzeichen/Realwert-Codierung**
  -Anzahl der Zeichen des Alphabets M ist größer als 2
  -für die Mehrzahl der Probleme geeigneter als Binäre Codierung, da eine geringere Abstraktion des Problems notwendig ist
  -zudem lässt sich ein Gen direkt auf eine Entscheidungsvariable des Problems abbilden, die Mutation eines Allels steht also in direktem Zusammenhang mit der Mutation des Wertes der Entscheidungsvariable (der Genotyp steht im engeren Verhältnis zum Phänotyp)

- **Baum-Codierung**

  -Ein Individuum wird nicht durch einen String sondern durch einen Baum repräsentiert
  -besonders für die genetische Programmierung geeignet

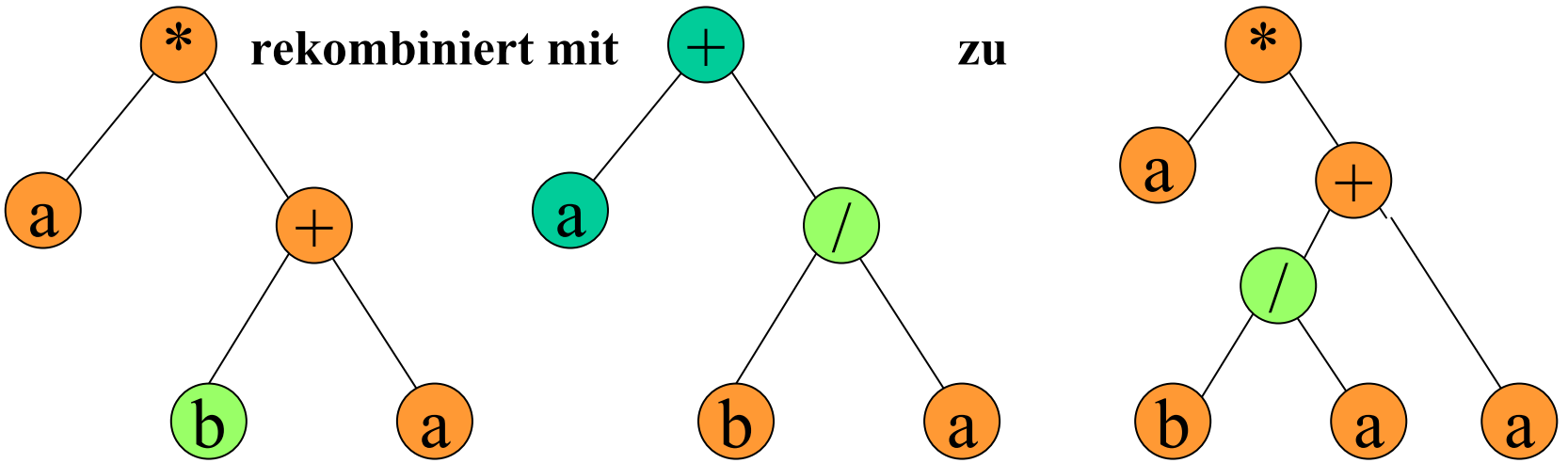
Codierung

Codierung in Bezug auf die Repräsentation

- **Baum-Codierung**

  -Mutation und Rekombination wird auf Teilbäume angewendet, daher ist die Größe des Suchraums offen, ein Individuum hat keine feste Länge

Beispiel: ein Individuum soll die Korrelation zwischen einer Anzahl von Inputvariablen zu einem Output finden, dies lässt sich durch einfache Gleichungen repräsentieren



Codierung

Codierung in Bezug auf die Repräsentation

- **Baum-Codierung**

  -Vorteilhaft ist die offene Größe (Länge) der Chromosomen, da in vielen Fällen die erforderliche Komplexität einer Lösung nicht bekannt ist.

  -Gleichzeitig kann die offene Länge ein Nachteil sein: Die Bäume können unkontrolliert ziemlich stark wachsen und so Overhead erzeugen. Zusätzlich ist eine spätere Analyse solcher unstrukturierten Bäume ein erheblicher Aufwand.

Codierung

Codierung in Bezug auf die Anordnung

-Die Anordnung, also die Konstellation der Parameter einer (Teil-) Lösung im String, stellt die Optimierung eines GA vor ein weiteres Problem:

- Die Parameter, die in hohem Maße ausschlaggebend für eine gute Fitness eines Individuums sind, sind am Anfang meist nicht bekannt
- sie werden oft als **Building Blocks** bezeichnet
- wichtig ist, daß gerade diese **funktional** abhängigen Parameter bevorzugt behandelt werden sollten, um die Evolution zu beschleunigen

➢ **Linkage-Problem**

Lösungsansätze für das Linkage-Problem

1.Schemata:

-keine explizite Lösung für das Linkage-Problem, aber gute Methode um die Building Blocks zu erfassen

Codierung

Codierung in Bezug auf die Anordnung

Lösungsansätze für das Linkage-Problem

2.Inversion:

-Umordnungsoperator

-die einzelnen Positionen im String werden mit Indices versehen

-bei einer Inversion wird ein Substring innerhalb des Individuums gespiegelt

{(1,1) **(2,1)** (3,0) (4,0) **(5,1)** (6,0) (7,1) (8,0)}

{(1,1) **(2,1)** **(5,1)** (4,0) (3,0) (6,0) (7,1) (8,0)}

-die Allele 2 und 5 seien ein Building Block, sie sind im inversen Chromosom näher beieinander, die Fitness des Individuums bleibt gleich, da die Indices mit in die Fitnessfunktion einbezogen werden

-das Schema ändert sich somit von *1**1*** zu *11*****

Codierung

Codierung in Bezug auf die Anordnung

Lösungsansätze für das Linkage-Problem

2.Inversion:

-bei einem Crossover (Single-Point-Crossover) sind die Building Blocks dadurch besser geschützt

-bei der Rekombination muss allerdings darauf geachtet werden, daß die Nachkommen alle Gene enthalten

Elter 1: {(1,0) (2,0) (3,0)	(4,1) (5,0) (6,1) (7,1) (8,1)}
Elter 2: {(1,1) (2,1) (5,1)	(4,0) (3,0) (6,0) (7,1) (8,0)}
Nachkomme 1: {(1,0) (2,0) **(3,0)**	(4,0) **(3,0)** (6,0) (7,1) (8,0)}
Nachkomme 2: {(1,1) (2,1) **(5,1)**	(4,1) **(5,0)** (6,1) (7,1) (8,0)}

-Die Gene 3 und 5 sind in den beiden Nachkommen jeweils doppelt vorhanden !!!

Codierung

Codierung in Bezug auf die Anordnung

Lösungsansätze für das Linkage-Problem

2.Inversion:

➢2 Lösungsstrategien:

(1) Crossover nur an Positionen mit gleichen Indices

(2) Master/Slave-Strategie: ein Elternteil (Slave) wird für die Rekombination kurzzeitig nach der Reihenfolge der Indices des zweiten Elternteils (Master) umgeordnet.

3.Hot Spots

-Masken, die angeben, an welchen Stellen eine Rekombination stattfinden darf

-jedes Individuum besitzt einen Hot Spot in Form eines zweiten Strings aus Elementen {0,1}. Nach einer ‚1' ist jeweils ein Crossover erlaubt

Elter 1: 11110110:00100100 Nachkomme 1: 11100100:00110110

Elter 2: 00001100:00010010 Nachkomme 2: 00011110:00000000

Codierung

Codierung in Bezug auf die Anordnung

Lösungsansätze für das Linkage-Problem

3.Hot Spots

-Vorteil: Die Crossovermasken werden koevolviert, mit der Hoffnung, daß sich diese auch verbessern

Selektion

Wie wähle ich die Individuen einer Population aus, die die Nachkommen für die nächste Generation erschaffen?

-wichtig ist, daß der Selektionsdruck beim gewählten Selektionsoperator nicht zu groß wird -> fittere Individuum könnten zu schnell die Überhand gewinnen -> lokale Optima
-ein zu kleiner Selektionsdruck verlangsamt die Evolution

Fitness-proportionale Selektion im Roulette-Prinzip

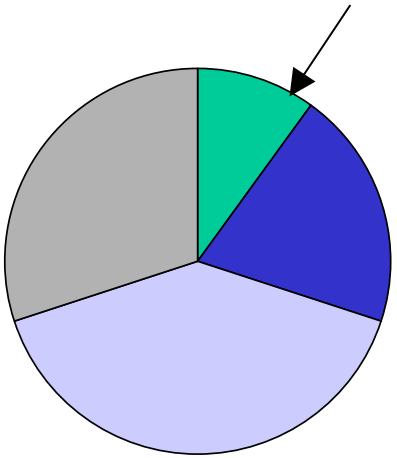
$$p_s(c_i)=\frac{\Phi(c_i)}{\sum_{j=1}^{\mu}\Phi(c_j)}$$

p_s.....Selektionswahrscheinlichkeit
μ.....Populationsgröße
c_i.....Individuum i
$\Phi(c_i)$...Fitnessfunktion

-zunächst Bestimmung der Selektionswahrscheinlichkeit der Individuen und proportionales Abtragen aller p_s auf ein Roulette-Rad

Selektion

Fitness-proportionale Selektion im Roulette-Prinzip



c_i	$p_s(c_i)$
1	.1
2	.2
3	.4
4	.3

-im zweiten Schritt wird das Roulettrad x mal gedreht und die Eltern für die nächste Generation anhand des Zeigers bestimmt (zb. Paare jeweils nacheinander)

-Nachteil: Im Extremfall x gleiche Individuen als Eltern

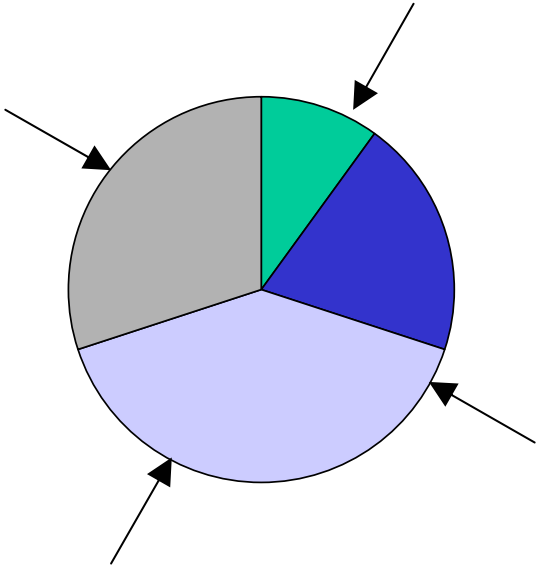
-besseres Verfahren:

Selektion

Stochastic Universal Sampling (SUS)

-ähnlich dem Roulette-Prinzip

-es gibt jedoch x Zeiger und das Rad wird nur einmal gedreht, jedes Individuum wird sooft in den „Pool“ der Eltern kopiert, wie Zeiger darauf weisen



c_i	$p_s(c_i)$	Anzahl der Kopien
1	.1	1
2	.2	0
3	.4	2
4	.3	1

Selektion

Wettkampfselektion

-jeweils z Individuen (mind. 2) werden zufällig aus der Population gewählt
-danach wird deren Fitness verglichen und das beste Chromosom kommt in den Elternpool
-das Verfahren wird ebenfalls x Mal wiederholt
-Vorteil: über z kann der Selektionsdruck eingestellt werden
-Nachteil: Im Extremfall x gleiche Individuen als Eltern

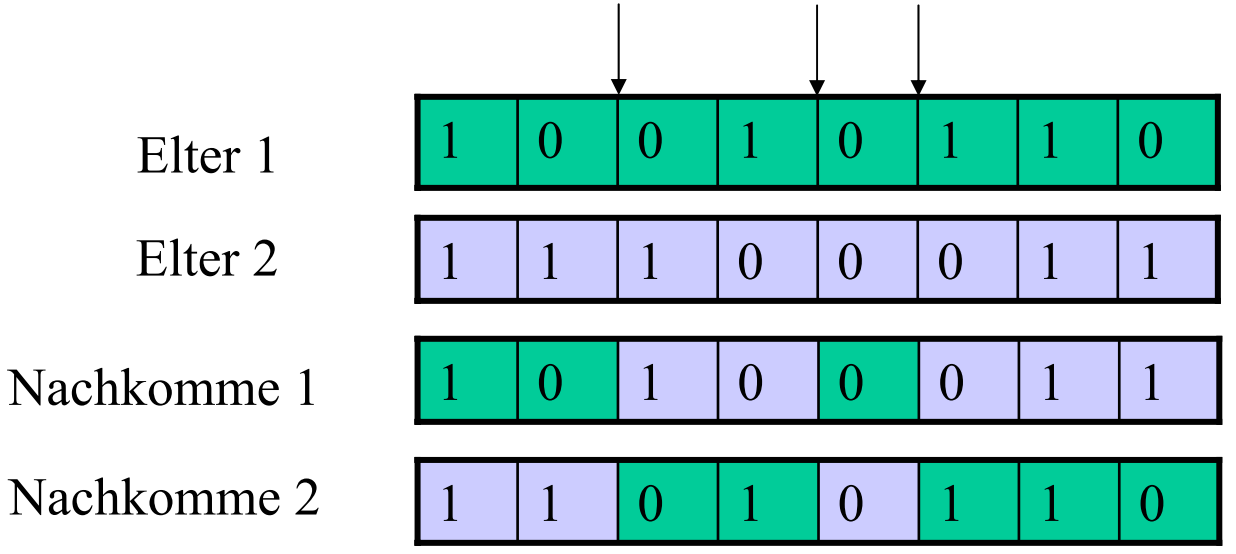
Rekombination

-beschreibt die Art und Weise der Vererbung der Merkmale auf die Nachkommen und ist der wichtigste Operator der Genetischen Algorithmen
-die gebräuchlichsten Verfahren rekombinieren nur zwei Eltern, Mehrelternbeziehungen sind jedoch möglich.
-je nach Selektionsalgorithmus werden die zu rekombinierenden Elternpaare stochastisch aus dem Elternpool gezogen oder direkt als Paare selektiert
-mit einer Crossover-Wahrscheinlichkeit p_c wird festgelegt, in wie vielen Fällen überhaupt eine Rekombination stattfindet
➢in der Praxis wird beispielsweise einfach eine Zufallszahl a gezogen und mit einer Menge gültiger Zahlen B verglichen, ist a Element von B findet eine Rekombination statt

Rekombination

Single und N-Point-Crossover

-Die Elternstrings werden an einem (N) Punkt(-en) für ein Crossover markiert.
-Die Nachkommen erhalten abwechselnd bis zur nächsten Markierung die Allele des jeweils anderen Elternteils.



Rekombination

Uniform-Crossover

-zunächst wird ein binärer Zufallsstring der Länge des Individuums generiert, diese Schablone wird über die Eltern gelegt und Position für Position verglichen
-eine ‚1' im String bedeutet für einen der beiden Nachkommen eine Übernahme des Allels des 1. Elternteils an dieser Position, eine ‚0' überträgt das entsprechende Allel von Elternteil 2
-beim zweiten Nachkommen wird umgekehrt verfahren

Elter 1	1	0	0	1	0	1	1	0
Schablone	1	1	0	1	1	0	0	1
Elter 2	1	1	1	0	0	0	1	1
Nachkomme 1	1	0	1	1	0	0	1	0
Nachkomme 2	1	1	0	0	0	1	1	1

Fazit

Das Gebiet der genetischen Algorithmen ist sehr weit abgesteckt.
Die einzelnen Strategien zur Realisierung, die Codierung und weitere Parameter sind sehr problemspezifisch.
GA bieten aber der wissenschaftlichen Problemlösung und der Forschung einen interessanten Ansatzpunkt.
Gerade das Suchen von Lösungen in einem extrem großen Suchraum ist durch reines Probieren aller Lösungen in vielen Fällen schwer möglich.
Auch die Erforschung der Vorgänge in biologischen Evolution finden in den GA einen vielversprechenden Ansatz.

Quellen

Melanie Mitchell; An Introduction to Genetic Algorithm;
The MIT Press, 1996

Frank Förster; Genetische Algorithmen – Grundkonzept und genetische Operatoren; Universität Salzburg, 2002

Sasa Hasan; Genetische Algorithmen; Universität Koblenz-Landau, 2002

Links

Beispielapplets für GA:

Biomorphe:
http://www.rennard.org/alife/english/gavintrgb.html
Maximumsuche in einer Landschaft:
http://homepage.sunrise.ch/homepage/pglaus/gentord.htm
Travelling Salesman–Problem und einfache Fittnesfunktion:
http://www.em.uni-karlsruhe.de/diplomstudien/info/gas.php

Linklisten/Archive:

The Genetic Algorithms Archive:
http://www.aic.nrl.navy.mil/galist/
Genetic Algorithms and Artificial Life Resources:
http://www.scs.carleton.ca/~csgs/resources/gaal.html