

3.– 6. September 2012  
in Nürnberg



# Herbstcampus

Wissenstransfer  
par excellence

## Programme züchten

Evolution als Optimierungsverfahren

Heiko Spindler

Freiberuflicher IT-Berater

# Agenda

---

- Was ist Evolution?
- Simulieren von Evolution
- Ein einfaches Beispiel
- Programme züchten
- Fazit

# Evolution der Evolutionstheorie

---

Da Vinci (1452-1519)

- Ahnung der Bedeutung von Fossilien

Georges Baron de Cuvier (1769-1832)

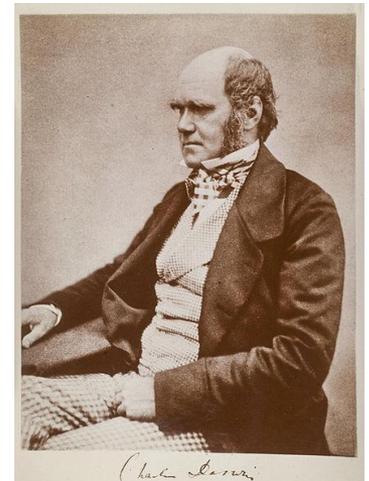
- Arten entstehen und verschwinden durch Naturkatastrophen

Jean Baptiste de Lamarck (1809)

- Erste Abstammungstheorie vererbare Veränderungen

Charles R. Darwin (1859)

“On the Origin of Species by Means of Natural Selection (Die Entstehung der Arten durch natürliche Zuchtwahl)“



# Beobachtungen

---

- Es gibt einen Überschuss an Individuen
  - Lebewesen sind nicht identisch
  - Die Populationen sind aber relative Konstanz
- Ein großer Teil der Lebewesen stirbt ohne Nachkommen
- Es gibt einen Selektionsdruck

Individuen, die mit den Lebensbedingungen am besten zurechtkommen, haben eine größere Überlebenschance und erzeugen mehr Nachkommen.

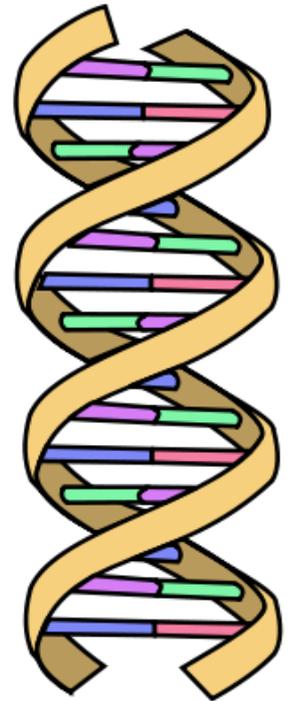
# Der genetische Code des Lebens

---

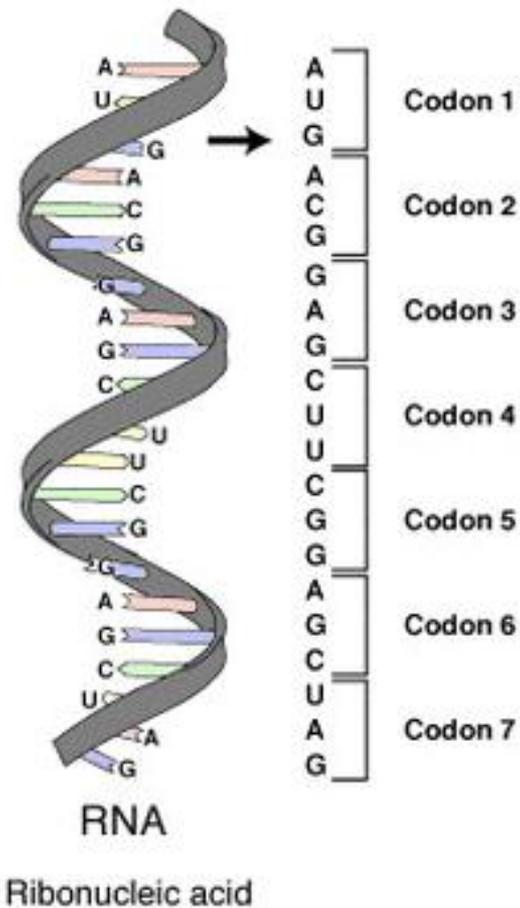
Alle Organismen übersetzen den genetischen Code in gleicher Weise!

Der genetische Code ist in Genen organisiert.

Für jedes Gen gibt es eine Reihe von möglichen Werten (Allele).



# DNA: Deoxyribonucleic acid



Helix mit Sequenzen der 4 Basen:

Adenin (A), Guanin (G),  
Cytosil C), Thymin (T)

Jeweils 3 Basen zusammen beschreiben  
ein Codon (64 Worte, alle außer 3  
beschreiben eine Aminosäure)

Aminosäuren bilden die Bausteine für  
die Proteine (Eiweiße) zusammensetzen.

# Evolutionenfaktoren

---

- Biologische Evolutionenfaktoren
  - Natürliche Selektion
  - Mutation: Austauschen von Basen (z. B. durch UV-Licht)
  - Sexuelle Rekombination: Crossover von DNS-Teilen zwischen den Eltern

## Allgemeine Evolutionenfaktoren:

- Veränderung + Selektion = Evolution



# Vorkommen von Evolution

---

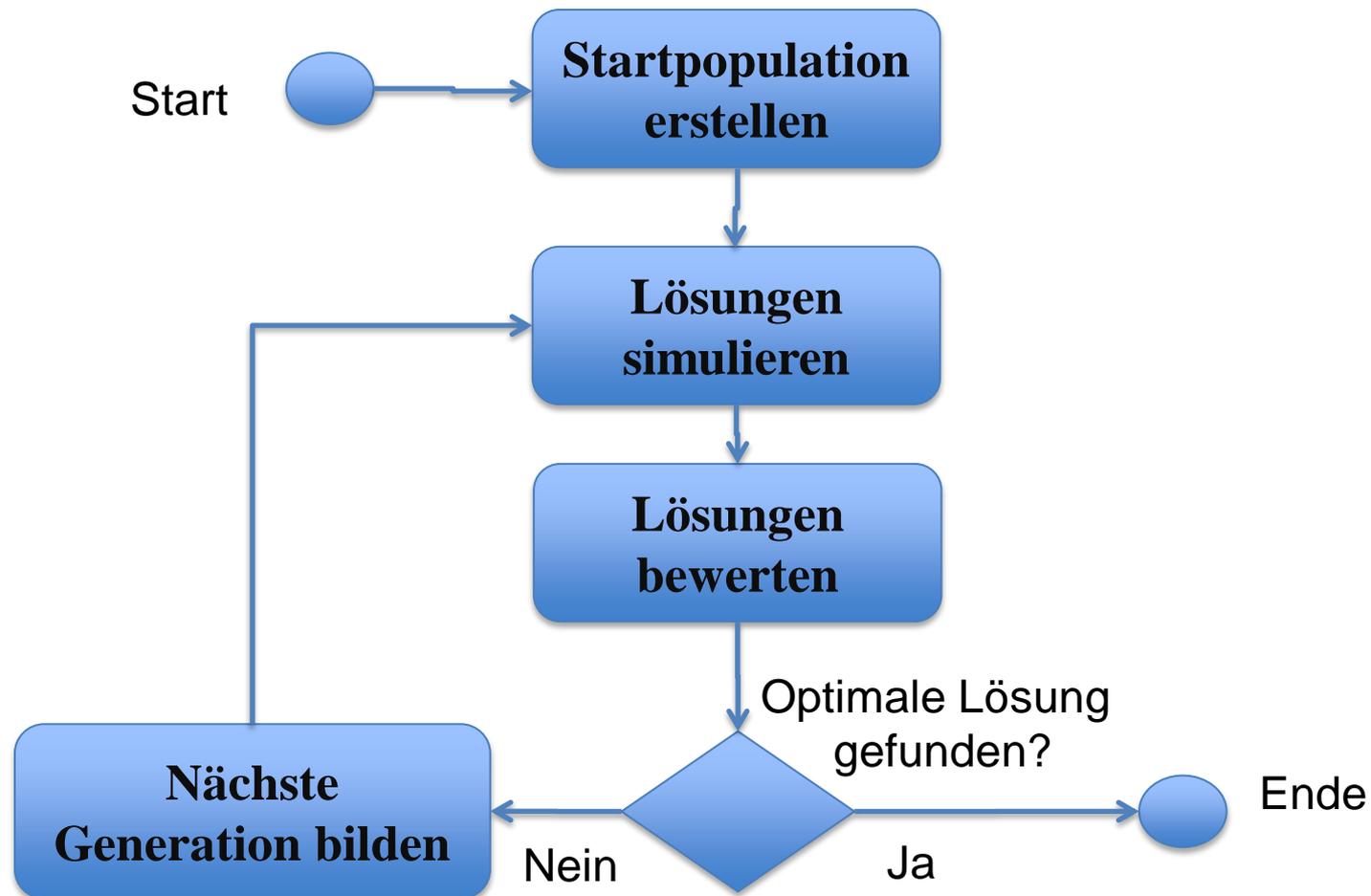
- Lebewesen und Arten
- Technische Produkte (z. B. Computer, Autos)
- Mode und Geschmäcker (z. B. Musik, Kleidung, etc.)
- Verhaltensweisen (z. B. Familie vs. Karriere)
- Standards (z. B. JAVA, USB, ...)
- Unternehmen
- ...

➔ alle Systeme, die die beiden  
Evolutionbedingungen erfüllen

# Biologische versus künstliche Evolution

<b>Biologisch</b>	<b>Künstlich</b>	<b>Beispiel TSP</b>
<b>Lebewesen</b>	<b>Lösungsversuch</b>	<b>Eine Rundreise</b>
<b>Gen/Eigenschaft</b>	<b>Teilelement der Lösung</b>	<b>Ort X als 2. Stopp besuchen</b>
<b>Leben eines Lebewesens</b>	<b>Simulation und Bewertung eines Lösungsversuchs</b>	<b>Berechnen der Gesamtlänge der Rundreise</b>
<b>Mutation oder Rekombination von Genen</b>	<b>Mutation oder Rekombination von des Reiseplans</b>	<b>Vertauschen der Reihenfolge der Orte</b>

# Prozess: Simulierte Evolution



# Veränderungen des Erbgutes

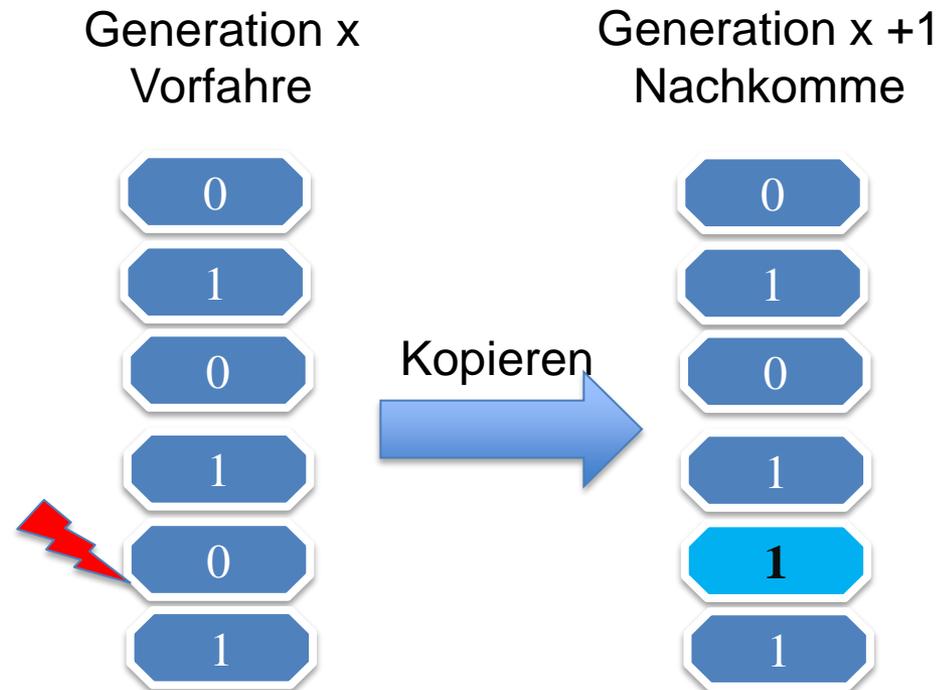
---

- Grundsätzlich zerstörerisch ...
- ... aber bring Neues in die Population ein

## Typen von genetischer Operationen

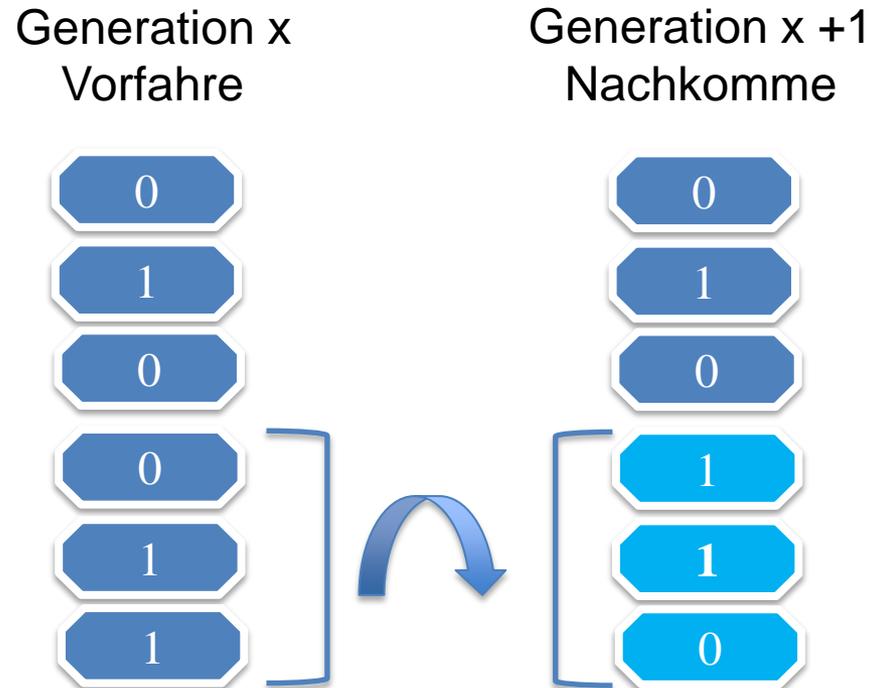
- Kopieren eines Lösungsversuchs
- Kombinieren von 2 Lösungsversuchen

# Mutation



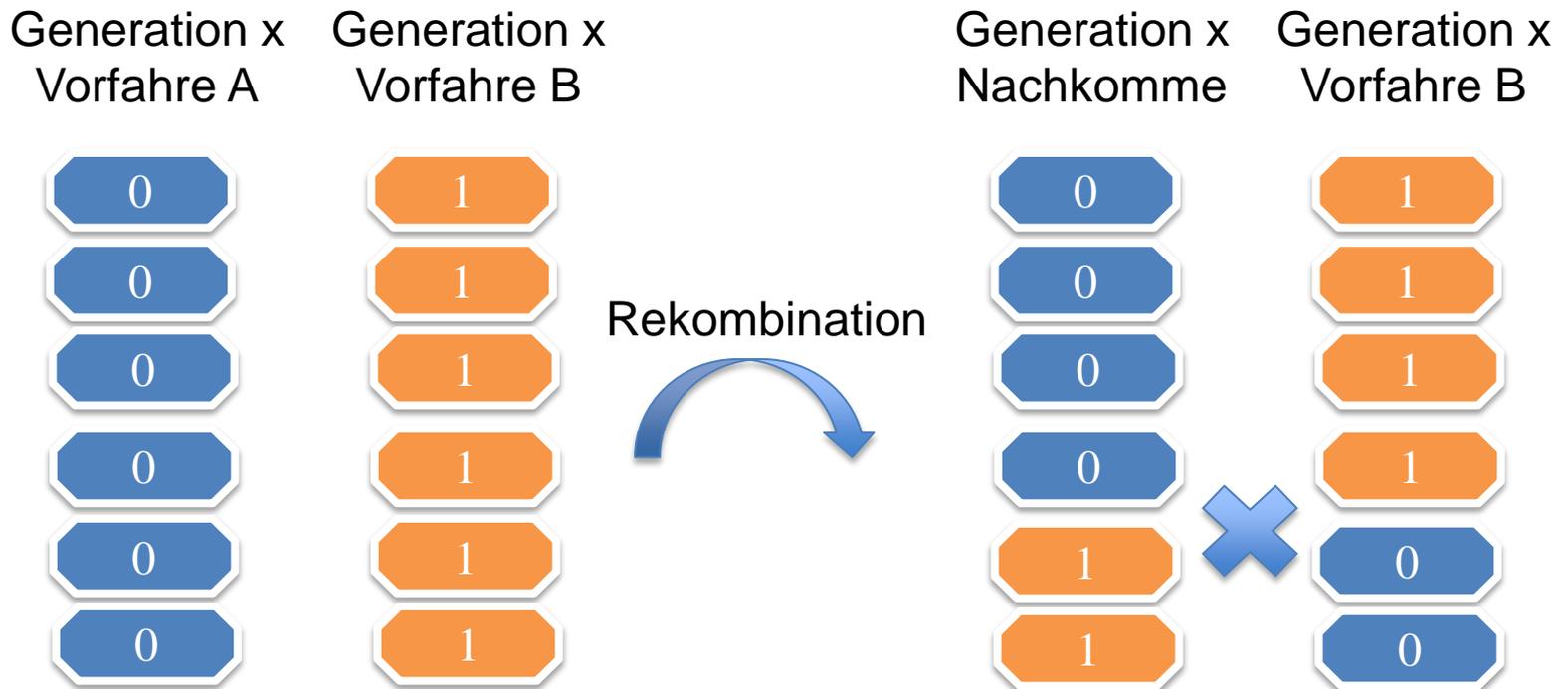
- An einer zufälligen Stelle im Erbgut wird ein Gen zufällig auf einen anderen Wert (Allel) verändert.

# Inversion



- Die Reihenfolge von Teilen des Erbgutes verändert sich.

# Rekombination und CrossOver



## Was ist zu tun?

---

1. Abbildung der Gene für die Lösungen
2. Definieren Bewertungsfunktion
3. Genetischen Operatoren (Veränderung)
4. Selektion von guten Lösungen
5. Implementieren des Ablaufs

# JGAP

---

- Framework für die Implementierung von Optimierungen mit Hilfe von simulierter Evolution
- Java und Open Source
- Link: <http://jgap.sourceforge.net/>
  
- Aufgaben
  - Bildet den grundlegenden Algorithmus ab
  - Datenstrukturen für die Abbildung von Genen mit unterschiedlichen Datentypen
  - Bildet die genetischen Operatoren ab (Mutation, Rekombination)
  - Interface für Bewertungsfunktionen

Beispiel:  
Welche Wörter sind hier versteckt?

---

**STIFTUER**  
**TISCHUND**  
**URBTHFAM**  
**NAEIAESI**  
**DDAGSLEN**  
**EIMEEDLU**  
**IOELEN\_T**  
**S\_RKATZE**

## Lösung: Alle versteckten Worte

---

FELD x:5 y:2 vertikal

REGEN x:1 y:2 diagonal

BEAMER x:2 y:2 vertikal

TISCH x:0 y:1 horizontal

MINUTE x:7 y:2 vertikal

KATZE x:3 y:7 horizontal

HASE x:4 y:2 vertikal

STALL x:2 y:1 diagonal

IGEL x:3 y:3 vertikal

TUER x:4 y:0 horizontal

HUND x:4 y:1 horizontal

STIFT x:0 y:0 horizontal

SEE x:4 y:4 vertikal

NASE x:6 y:1 vertikal

EIS x:0 y:5 vertikal

STUNDE x:0 y:0 vertikal

RADIO x:1 y:2 vertikal

# Ein Beispiel: Aufgaben-Generator

## Buchstabenmatrix

---

- Erstellen einer Buchstabenmatrix, in der Wörter versteckt sind
- Ziele
  - Möglichst viel Worte verstecken
  - Worte können senkrecht, waagrecht und diagonal verlaufen
  - Worte dürfen sich überlagern und die gleichen Buchstaben nutzen
  - Kein Wort darf über den Rand hinaus gehen

STIFTUER  
TISCHUND  
URBTHFAM  
NAEIAESI  
DDAGSLEN  
EIMEEDLU  
IOELEN\_T  
S\_RKATZE

# Abbildung der Buchstabenmatrix

## Klasse: Feld.java

---

```
public void reset()
public void putWord( String word, int
    x, int y, DirectionEnum direction )
public boolean isWordUsed( String word)
public int getWordCount()
public int getViolations()
public int getLetters()
public int getMehrfach()
```

# Buchstabenmatrix: Abbildung der Gene

---

- Insgesamt: 15 bis 25 Worte sollen versteckt werden
- In einer 8 mal 8 Buchstaben-Matrix
- Struktur der Gene:

Position 1	Position 2	Position 3	Position 4
Wortindex: (0..20)	Position X: (0..7)	Position Y: (0..7)	Richtung: Horizontal (0), Vertikal (1), Diagonal (2)

# Abbildung der Gene

---

```
Gene[] sampleGenes = new Gene[ MAX_GENES * ELEMENTS_PER_GENE ];
for (int i = 0; i < MAX_GENES * ELEMENTS_PER_GENE; i = i
+ELEMENTS_PER_GENE)
{
    sampleGenes[i] =new IntegerGene(conf, 0, WORDS.Length-1 );
    sampleGenes[i+1]=new IntegerGene(conf, 0, Field.MAX_WIDTH-1 );
    sampleGenes[i+2]=new IntegerGene(conf, 0, Field.MAX_HEIGHT-1 );
    sampleGenes[i+3]=new IntegerGene(conf, 1,DirectionEnum.values().Length);
}
IChromosome sampleChromosome = new Chromosome(conf, sampleGenes);
conf.setSampleChromosome(sampleChromosome);
```

# Wortmatrix: Fitnessfunktion – Höher, schneller, weiter

---

- Besser:
  - Mehr Worte
  - Mehr Buchstaben
  - Buchstaben werden mehrfach benutzt
  
- Fehler:
  - Wort steht über den Rand der Matrix hinaus
  - Wort überschrieben vorhanden Buchstaben

# Fitnessfunktion

---

```
protected double evaluate(ICHromosome a_subject) {
    Field field = new Field();
    for (int i=0; i<a_subject.size(); i=i+Crossword.ELEMENTS_PER_GENE) {
        Integer wordPos = (Integer)a_subject.getGene(i).getAllele();
        Integer xpos = (Integer)a_subject.getGene(i+1).getAllele();
        Integer ypos = (Integer)a_subject.getGene(i+2).getAllele();
        Integer direction = (Integer)a_subject.getGene(i+3).getAllele();
        DirectionEnum d = DirectionEnum.values()[direction-1];

        String word = Crossword.WORDS[wordPos];
        field.putWord( word, xpos, ypos, d);
    }
    int violations = field.getViolations()+1;
    int letters = field.getLetters()+1;
    int mehrfach = field.getMehrfach()+1;
    int wordCount = field.getWordCount()+1;

    return ((letters*wordCount*mehrfach) / (violations*violations)) ;
}
```

# Konfiguration

---

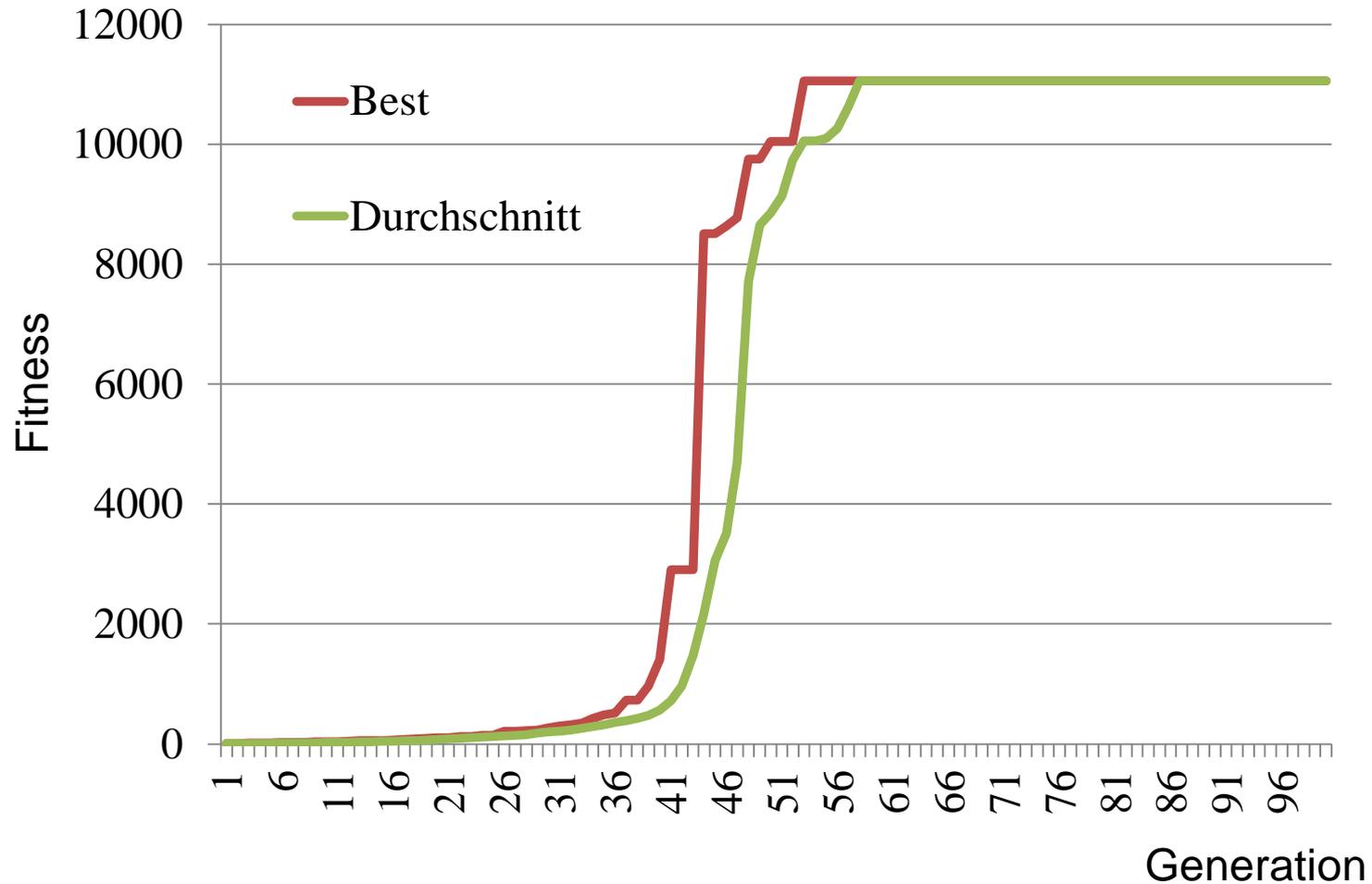
```
conf.getGeneticOperators().clear();
```

```
MutationOperator mutOp = new  
    MutationOperator(conf);  
mutOp.setMutationRate( 60 );  
    conf.addGeneticOperator(mutOp);
```

```
conf.addGeneticOperator(new  
    CrossoverOperator(conf, 2));
```

```
conf.setPopulationSize(800);  
conf.setPreservFittestIndividual(true);
```

# Wortmatrix: Beispiel eines Optimierungslaufs



# Wortmatrix: Betrachtungen des Suchraums

---

Wie viele Lösungen wurden getestet?

Beispiel:

- Populationsgröße: 200
- Generationen: 10000

Ergebnis:

- Ca. 2 Millionen Lösungen wurden untersucht

## Wortmatrix: Betrachtungen des Suchraums

---

Mögliche Anzahl der Lösungen:

= (Ausprägungen der Gene) hoch (Anzahl Gene)

Rechnung für die Wortmatrix:

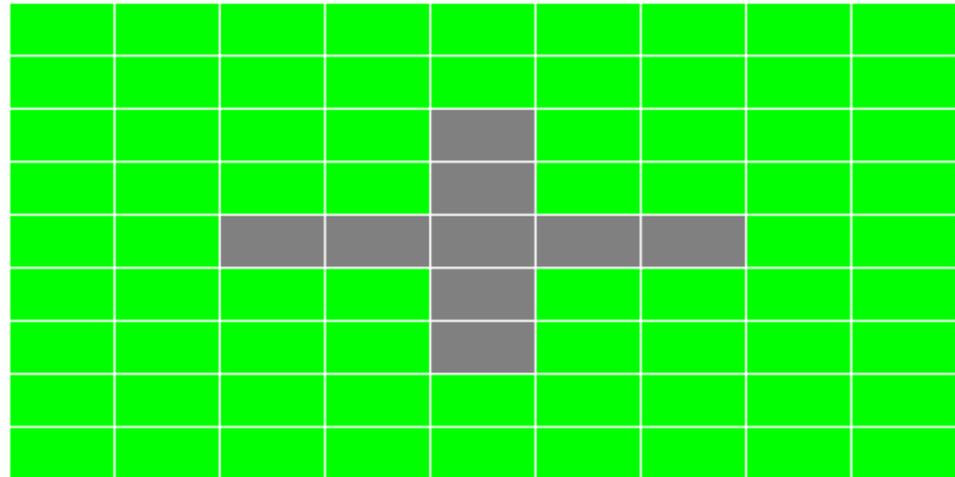
= (Anzahl Wort \* Positionen x \* Positionen y \*  
Richtung) hoch (Anzahl Gene)

$$= (20 * 8 * 8 * 3)^{20} = 3840^{20} = 4,86 * 10^{71}$$

# Beispiel: Programme züchten

---

- Aufgabe:
  - Ein Roboter soll alle Zelle eines Feldes markieren.
  - Das Feld besteht aus 9 mal 9 Zellen.
  - Manche Zellen enthalten Hindernisse.
  - Der Roboter hat einen Speicher für 30 Befehle.



# Befehle

Befehl	Bemerkung
<b>Move</b> <links, rechts, hoch, runter>	Bewegen um eine Zelle
<b>Set</b>	Markiert die aktuelle Zelle
<b>Nop</b>	„No Operation“
<b>Goto</b> <Speichersplatz>	Ausführung wird mit dem Befehl an <Speicherplatz> fortgesetzt.
<b>If</b> <links, rechts, hoch, runter> = <frei, markiert, Wand> <b>Goto</b> <Speicherplatz>	Bedingter Sprung, wenn das untersuchte Feld im abgefragten Zustand ist.

# Abbildung Gene

---

Befehle werden als 4 Integer-Werte abgelegt:



Nicht benötigte Parameter werden ignoriert.

Beispiel:

- Goto-Befehlscode + 1. Parameter mit Zieladresse

# Fitnessfunktion

---

Priorität 1: „Löse die Aufgabe“

- Alle Felder markieren (Fitness zwischen 0 und 100)  
entspricht dem Prozentsatz markierter Felder.
- Effizienz spielt keine Rolle.

...

# Fitnessfunktion

---

## Priorität 1: „Löse die Aufgabe“

- Alle Felder markieren (Fitness zwischen 0 und 100)  
entspricht dem Prozentsatz markierter Felder.
- Effizienz spielt keine Rolle.

## Priorität 2: „Lösen die Aufgabe gut“

- Jeder Befehl kostet Energie. Ein neu markiertes Feld bringt etwas Energie.
- Je mehr Energie nach dem Markieren aller Zellen übrig ist desto effizienter ist der Algorithmus.
- $\text{Fitness} = 100 + \text{überschüssige Energie nach dem markieren aller Felder}$

# Fitnessfunktion

---

```
public float calcFitness()
{
    float anzleer= feld.getNumberWithEntry(Feld.leer);
    float anzbelegt= feld.getNumberWithEntry(Feld.belegt);

    if (feld.aktFelderMarked >= (feld.MaxFelderToMark))
    {
        if (aktEnergie < 0) {
            return 100;
        } else {
            return ((100 + (aktEnergie)));
        }
    } else {
        return ((anzbelegt * 100) / (anzleer + anzbelegt));
    }
}
```

globalbest.indiv

Datei Individuum	
<pre> 0.MOVE links 1.SET 2.IF (links==frei) GOTO 0 3.MOVE rechts 4.IF (rechts==frei) GOTO 7 5.IF (hoch==frei) GOTO 16 6.MOVE runter 7.MOVE rechts 8.MOVE rechts 9.SET 10.GOTO 0 11.MOVE links 12.GOTO 26 13.IF (hoch==Wand) GOTO 22 14.SET 15.SET 16.MOVE hoch 17.GOTO 1 18.NOP 19.MOVE hoch 20.MOVE hoch 21.GOTO 3 22.MOVE hoch 23.MOVE links 24.MOVE hoch 25.MOVE links 26.IF (runter==markiert) GOTO 21 27.MOVE runter 28.NOP 29.IF (links==frei) GOTO 20                     </pre>	
	Energie : 404
	Schritte : 120

# Parameter und ihre Auswirkungen 1

---

- Populationsgröße
  - Sinnvolle Werte: 40-500 Individuen
  - In einer großen Population kann sich eine nur leichte Verbesserung evtl. nicht durchsetzen und geht in der Menge unter.
- Rekombinationsrate
  - Hohe Rate: Prozess konvergiert sehr schnell.
  - Niedrige Rate: Gute Individuen (insbesondere nur leichte Verbesserungen) setzen sich evtl. nur langsam oder gar nicht durch

## Parameter und ihre Auswirkungen 2

---

- Mutationsrate
  - Zu hoch bedeutet zu viel Zerstörung
  - Zu Niedrig bedeutet zu wenig Kreativität und Neues
  - Gute Ergebnisse: 1% bis 5% Mutationsrate pro Gen
  - Idee: Anpassen der Mutationsrate:
    - Beispiel: Lange Phasen der Stagnation führen zum Ansteigen der Mutationsrate

# Vorteile der Simulierten Evolution

---

- Grundlegende Prozess ist einfach
  - Generisch implementierbar
- Es ist kein Wissen über einen Lösungsweg notwendig
  - Es reicht aus Lösungen zu simulieren und zu bewerten
- Ist für komplexe Probleme geeignet
- Gut parallelisierbar

# Missverständnis bzgl. Evolution

---

- Langsam
- Mutationen sind extrem zerstörerisch
- Evolution kann nichts „Neues“ erzeugen, sondern nur Vorhandenes variieren
  - Wie können komplizierte Organe (z. B. menschliches Auge oder Gehirn) entstehen?
- Evolution ist nicht zielgerichtet: Versuch & Irrtum
- Evolution gibt es nur in der Natur

## Wann ist das Vorgehen nicht geeignet?

---

- Lösungen sind (leicht) analytisch zu finden
- Lösungen sind schwer zu bewerten oder zu simulieren

Beispiel:

- Direkt Java-Code zu erzeugen macht wenig sinn.

# Kritische Erfolgsfaktoren

---

- Bewertungsfunktion
  - Eindeutig
  - Sollte die gewünschte Fitness wiedergeben
  - Sollte leicht zu berechnen sein
- Gute Abbildung einer Lösung als Gene
  - Alle möglichen Kombinationen sollten gültige Lösung darstellen
  - Werte sollten unabhängig sein
  - Möglichst kompakt

# Erweiterungen

---

- Insel Modell
  - Isolierte Populationen tauschen gute Lösungen aus.
- Eliten
  - Die beste(n) Lösung(en) überleben immer.
- Räuber Beute Modell
  - Es laufen zwei Optimierungen gegeneinander: Suche nach einem Programm das alle Felder markiert und eine Optimierung, die für die erste Optimierung schlechte Startpositionen sucht und diese dadurch herausfordert.
- Interaktive Evolution
  - Auswahl und Bewertung durch Benutzer.

# Beispiele für die Anwendung

---

- Erstellen von Stunden- und Dienstplänen
- Erstellen von Produktionsplänen
- Optimieren von Chipstrukturen und Leiterplatten
- Optimieren von Vertriebsregionen (Pharma-Bereich)
- Optimieren von Routen- und Tourenplänen
- Erzeugen von optimalen Strukturparametern für neuronale Netze (Schichten, Anzahl Neuronen, ...)

# Literatur 1

---

Principia Evolvica. Simulierte Evolution mit Mathematica, von  
Christian Jacob

Gebundene Ausgabe: 706 Seiten

Verlag: Dpunkt. Verlag GmbH (Januar 1999)

ISBN: 3920993489

Evolutionäre Algorithmen: Genetische Algorithmen - Strategien und  
Optimierungsverfahren – Beispielanwendungen, von Ingrid

Gerdes, Frank Klawonn, Rudolf Kruse

Taschenbuch: 252 Seiten

Verlag: Vieweg+Teubner Verlag; Auflage: 2004 (29. Juli 2004)

ISBN: 3528055707

## Literatur 2

---

Bücher von Richard Dawkins,

- The Selfish Gene (Das egoistische Gen)
- The Blind Watchmaker (Der blinde Uhrmacher)

Bauplan für eine Seele, von Dietrich Dörner

Taschenbuch: 832 Seiten

Verlag: rororo

ISBN: 3499611937

3.– 6. September 2012  
in Nürnberg



# Herbstcampus

Wissenstransfer  
par excellence

Vielen Dank!

Heiko Spindler

Freiberuflicher IT-Berater