

Intelligente Systeme

Evolutionäre Algorithmen

Prof. Dr. R. Kruse C. Moewes G. Ruß

{kruse,cmoewes,russ}@iws.cs.uni-magdeburg.de

Institut für Wissens- und Sprachverarbeitung

Fakultät für Informatik

Otto-von-Guericke Universität Magdeburg

Übersicht

- 1. Biologische Grundlagen**
2. Grundlagen evolutionärer Algorithmen
3. Genetische Programmierung
4. Reale Anwendungsbeispiele

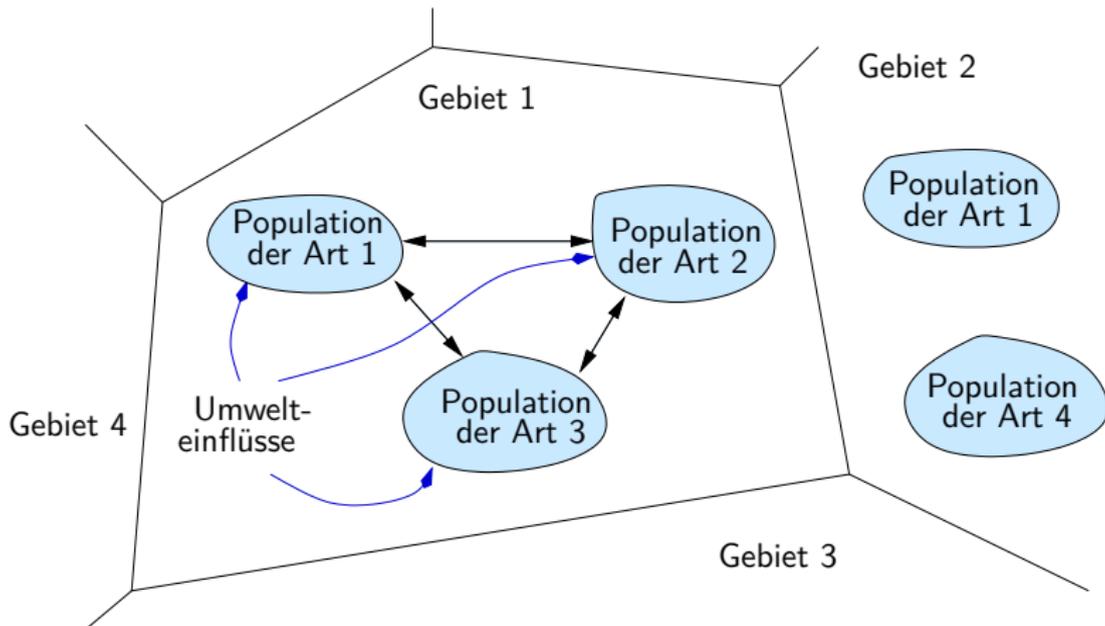
Motivation

- EAs basieren auf **biologischer Evolutionstheorie**
- grundsätzliches Prinzip:
 - **Auswahl von zufällig entstehenden, vorteilhaften Eigenschaften durch natürliche Auslese**
 - Individuen mit vorteilhaften Eigenschaften: bessere Fortpflanzungs- und Vermehrungschancen – „*differentielle Reproduktion*“
- Evolutionstheorie erklärt Vielfalt und Komplexität der Lebewesen
- erlaubt Vereinigung aller Disziplinen der Biologie

Biologischer Evolutionstheorie

- allmähliche Entwicklung der Lebewesen \Rightarrow sehr komplexen Entwicklungen
- reine Beobachtung und Hypothesenbildung durch Lamarck (1809) und Darwin (1859)
- wissenschaftliche Erklärungen: Populationsgenetik (1908)
- Erklärungen der Molekulargenetik: Watson und Crick (1953)

Biologisches Modell



Biologisches Modell

langfristige Anpassung von Faktoren wie

- Nahrungsaufnahme
- Partnerfindung
- Fortpflanzung
- Tarnung

führt zur Entwicklung hochkomplexer Lösungen

Biologische Evolutionsfaktoren

- Evolution nur dann möglich, wenn sich Genfrequenz ändert
- ⇒ Auftreten von Faktoren wie
- Mutation
 - Selektion
 - Genfluss
 - Gendrift

Mutation

- direkte Änderung der Genfrequenz durch kleine Veränderungen an Individuen auf molekulargenetischer Ebene
- natürlicher Vervielfältigungsfehler
- Mutationsrate beim Menschen: 10^{-10}
- also bei 10^5 Genen mit 10^4 Bausteinen: 1 Mutation bei jeder 10. Zellteilung
- nur kleine Mutationen meist überlebensfähig

Selektion

- Änderung der Genfrequenz durch unterschiedlich viele Nachkommen der einzelnen Individuen
- Fitness eines Allels = $\frac{\# \text{Nachkommen}}{\# \text{Nachkommen des besten Allels}}$
- Unterschiedliche Ursachen:
 - Überlebenschancen
 - Fruchtbarkeit
 - Fähigkeit, einen Partner zu finden
 - Länge der Generationsdauer

Genfluss und Gendrift

Genfluss:

- Migration zwischen sonst getrennten Populationen
- ⇒ Änderung der Genfrequenz
- bei Standard-EA nicht imitiert (für Parallelrechner relevant)

Gendrift:

- große Populationen: stabil bezüglich Zufallsereignissen
- kleinen Populationen: zufälliges „Driften“ des gesamten Genpools
- Faktor bei EA (aber nicht gezielt, da eher negativ)

Rekombination

- laut Populationsgenetik kein Faktor: Genfrequenz gleichbleibend in großen Populationen
- Sichtweise der Populationsgenetik ist mechanistisch: Gene sind Bauplan, wobei Rekombination sie nur neu zusammenstellt
- modernere Sichtweise: genetisches Netzwerk, wobei Gene abhängig von anderen Genen aktiv sind
- selbstorganisierter, zyklischer „Wachstumsprozess“: durch Rekombination neue Zusammenhänge erschaffen
- Benutzung der Rekombination in EA: zufälliges Durchprobieren anderer Kombinationen (ohne Selbstorganisation)

Übersicht

1. Biologische Grundlagen
- 2. Grundlagen evolutionärer Algorithmen**
 - Grundbegriffe
 - Elemente
3. Genetische Programmierung
4. Reale Anwendungsbeispiele

Grundbegriffe und ihre Bedeutung I

Begriff	Biologie	Informatik
Individuum	Lebewesen	Lösungskandidat
Chromosom	DNS-Histon-Protein-Strang	Zeichenkette
	legt „Bauplan“ bzw. (Teil der Eigenschaften) eines Individuums in kodierter Form fest	
	meist mehrere Chromsomen je Individuum	meist nur ein Chromosom je Individuum
Gen	Teilstück eines Chromosoms	ein Zeichen
	grundlegende Einheit der Vererbung, die eine (Teil-)Eigenschaft eines Individuums festlegt	
Allel (Allelomorph)	Ausprägung eines Gens	Wert eines Zeichens
	je Chromosom gibt es nur eine Ausprägung eines Gens	
Locus	Ort eines Gens	Position eines Zeichens
	in einem Chromosom gibt es an jedem Ort genau ein Gen	

Grundbegriffe und ihre Bedeutung II

Begriff	Biologie	Informatik
Phänotyp	äußeres Erscheinungsbild eines Lebewesens	Umsetzung/Implementierung eines Lösungskandidaten
Genotyp	genetische Konstitution eines Lebewesens	Kodierung eines Lösungskandidaten
Population	Menge von Lebewesen	Familie/Multimenge von Chromosomen
Generation	Population zu einem Zeitpunkt	
Reproduktion	Erzeugen von Nachkommen aus einem oder mehreren (meist zwei) Lebewesen	Erzeugen von (Kind-)Chromosomen aus 1 oder mehreren (Eltern-)Chromosomen
Fitness	Tauglichkeit/Angepaßtheit eines Lebewesens	Güte/Tauglichkeit eines Lösungskandidaten
	bestimmt Überlebens- und Fortpflanzungschancen	

Elemente eines EAs I

Kodierungsvorschrift für Lösungskandidaten

- problemspezifische Kodierung der Lösungskandidaten
- keine allgemeinen Regeln
- später: einige Aspekte, zur Beachtung bei Wahl einer Kodierung

Methode, die **Anfangspopulation** erzeugt

- meistens Erzeugen zufälliger Zeichenketten
- auch komplexere Verfahren je nach gewählter Kodierung

Elemente eines EAs II

Bewertungsfunktion (Fitnessfunktion) für die Individuen

- stellt Umgebung dar und gibt Güte der Individuen an
- meist identisch mit zu optimierender Funktion
- enthält auch zusätzliche Elemente (z.B. Nebenbedingungen)

Auswahlmethode basierend auf Fitnessfunktion

- bestimmt Individuen für Erzeugung von Nachkommen
- wählt Individuen unverändert in nächste Generation

Elemente eines EAs III

genetischen Operatoren, die Lösungskandidaten ändern

- *Mutation* — zufällige Veränderung einzelner Gene
- *Crossover* — Rekombination von Chromosomen
 - richtig: “crossing over” (Meiose-Vorgang, Zellteilungsphase)
 - Chromosomen werden zerteilt und dann überkreuzt zusammengefügt

Parameterwerte (Populationsgröße, Mutationsw'keit, etc.)

Abbruchkriterium, z.B.

- festgelegte Anzahl von Generationen berechnet
- festgelegte Anzahl von Generationen keine Verbesserung
- vorgegebene Mindestlösungsgüte erreicht

Beispiele für Optimierungsprobleme I

Parameteroptimierung

- z.B. Krümmung von Rohren für minimalen Widerstand
- allgemein: Parametersatz finden, der gegebene reellwertige Funktion (möglichst global) optimiert

Packprobleme

- z.B. Rucksack füllen mit maximalem Wert
- oder Packen möglichst weniger Kisten mit gegebenen Gütern

Wegeprobleme

- z.B. Problem des Handlungsreisenden (z.B. Bohren von Platinen)
- Reihenfolge anzufahrender Ziele, Fahrtroutenoptimierung, Verlegen von Leiterbahnen auf Platinen/integrierten Schaltkreisen

Beispiele für Optimierungsprobleme II

Anordnungsprobleme

- z.B. Steinerproblem (engl. facility allocation problem):
- Positionierung von Verteilerknoten z.B. in Telefonnetz

Planungsprobleme

- z.B. Ablaufpläne (Scheduling), Arbeitspläne, Operationenfolgen (auch Optimierung in Compilern — Umordnung der Befehle)

Strategieprobleme

- z.B. Gefangenendilemma und andere Modelle der Spieltheorie
- Verhaltensmodellierung von Akteuren im Wirtschaftsleben

biologische Modellbildung

- z.B. Netspinner (beschreibende Regeln zum Spinnennetz-Bau)
- EA optimiert Parameter, Vergleich mit Realität \Rightarrow gutes Modell

Generischer Grundalgorithmus

Algorithmus 1 EA-Schema

Eingabe: Optimierungsproblem

$t \leftarrow 0$

$\text{pop}(t) \leftarrow$ erzeuge Population der Größe μ

bewerte $\text{pop}(t)$

while Terminierungsbedingung nicht erfüllt {

$\text{pop}_1 \leftarrow$ selektiere Eltern für λ Nachkommen aus $\text{pop}(t)$

$\text{pop}_2 \leftarrow$ erzeuge Nachkommen durch Rekombination aus pop_1

$\text{pop}_3 \leftarrow$ mutiere die Individuen in pop_2

 bewerte pop_3

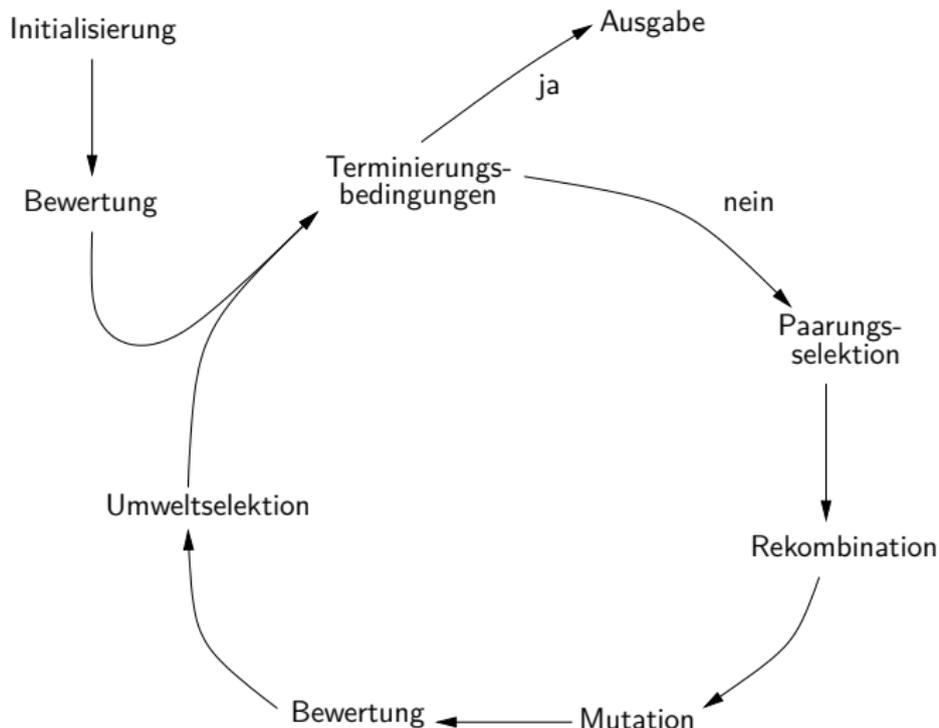
$t \leftarrow t + 1$

$\text{pop}(t) \leftarrow$ selektiere μ Individuen aus $\text{pop}_3 \cup \text{pop}(t - 1)$

}

return bestes Individuum aus $\text{pop}(t)$

Zyklischer Ablauf des EA



Übersicht

1. Biologische Grundlagen

2. Grundlagen evolutionärer Algorithmen

3. Genetische Programmierung

- Terminal- und Funktionssymbole

- Symbolische Ausdrücke

- Ablauf eines GPs

- Genetische Operatoren

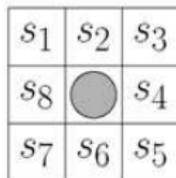
- 11-Multiplexer

- Stimulus-Response-Agent

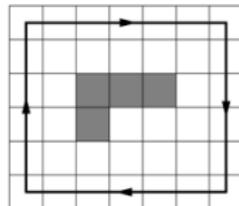
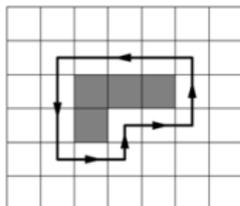
4. Reale Anwendungsbeispiele

Beispiel: Stimulus-Response-Agent

- betrachte Stimulus-Response-Agenten in Gitterwelt:



- 8 Sensoren s_1, \dots, s_8 liefern Zustand der Nachbarfelder
- 4 Aktionen: go east, go north, go west, go south
- direkte Berechnung der Aktion aus s_1, \dots, s_8 , kein Gedächtnis
- **Aufgabe:** umlaufe ein im Raum stehendes Hindernis oder laufe Begrenzung des Raumes ab!



Genetische Programmierung

Idee:

- Beschreibung einer Problemlösung durch Computerprogramm"
 - dieses verbindet gewisse Eingaben mit gewissen Ausgaben
- ⇒ Suche nach passendem Computerprogramm
- GP = genereller Weg, Computerprogramme zu lernen/zu erzeugen
 - Darstellung der Programme durch sog. Parse-Bäume

Darstellung der Lösungskandidaten

- **EA:** Chromosomen fester Länge (Vektor von Genen)
- **GP:** Funktionsausdrücke bzw. Programme
 - ⇒ **genetische Programmierung**
 - ⇒ komplexere Chromosomen variabler Länge
- formale Grundlage: Grammatik zur Beschreibung der Sprache
- Festlegung zweier, problemspezifischer Mengen \mathcal{F}, \mathcal{T}
 - \mathcal{F} – Menge der Funktionssymbole und Operatoren
 - \mathcal{T} – Menge der Terminalsymbole (Konstanten und Variablen)
- sollten nicht zu groß sein (Beschränkung des Suchraums)
- dennoch reichhaltig genug, um Problemlösung zu ermöglichen

Beispiele zu Symbolmengen

Beispiel 1: Erlernen einer Booleschen Funktion

- $\mathcal{F} = \{\text{and, or, not, if } \dots \text{ then } \dots \text{ else } \dots, \dots\}$
- $\mathcal{T} = \{x_1, \dots, x_m, 1, 0\}$ bzw. $\mathcal{T} = \{x_1, \dots, x_m, t, f\}$

Beispiel 2: Symbolische Regression

- Regression: Bestimmung einer Ausgleichsfunktion zu geg. Daten unter Minimierung der Fehlerquadratsumme – *Methode der kleinsten Quadrate*
- $\mathcal{F} = \{+, -, *, /, \sqrt{}, \sin, \cos, \log, \exp, \dots\}$
- $\mathcal{T} = \{x_1, \dots, x_m\} \cup \mathbb{R}$

Abgeschlossenheit von \mathcal{F} und \mathcal{T}

- **wünschenswert:** \mathcal{F} und \mathcal{T} sollten abgeschlossen sein
 - Funktionen $\in \mathcal{F}$ akzeptieren alle möglichen Eingabewerte
 - z.B. Division durch 0 würde Ausführung mit Fehler beenden
 - falls keine Abgeschlossenheit, dann Lösen eines Optimierungsproblems mit Bedingungen
- ⇒ Reparaturmaßnahmen oder Strafterme u.U. nötig

Abgeschlossenheit von \mathcal{F} und \mathcal{T}

verschiedene Strategien garantieren Abgeschlossenheit, z.B.

- Implementierung von gesicherten Versionen anfälliger Operatoren
 - gesicherte Division: Divisor 0 \Rightarrow 0/Maximalwert zurückgeben
 - gesicherte Wurzelfunktion: operieren mit Absolutwert
 - gesicherte Logarithmusfunktion: $\forall x \leq 0$: $\log(x) = 0$ o.Ä.
- Kombination verschiedener Funktionsarten, z.B. numerische und boolesche Werte (FALSE = 0, TRUE \neq 0)
- Implementierung von bedingten Vergleichsoperatoren, z.B. *IF*
 $x < 0$ *THEN* ...
- ...

Vollständigkeit von \mathcal{F} und \mathcal{T}

- GP ist nur effizient und effektiv, wenn \mathcal{F} und \mathcal{T} hinreichend/vollständig zum Finden eines angemessenen Programms
- z.B. Boolesche Aussagenlogik: $\mathcal{F} = \{\wedge, \neg\}$ und $\mathcal{F} = \{\rightarrow, \neg\}$ sind vollständig, $\mathcal{F} = \{\wedge\}$ nicht
- generelles Problem des maschinellen Lernens: **Merkmalsauswahl**
- Finden der kleinsten vollständigen Menge: (meistens) NP-schwer
 \Rightarrow gewöhnlich: mehr Funktionen in \mathcal{F} als notwendig

Symbolische Ausdrücke

- Chromosomen = Ausdrücke (zusammengesetzt aus Elementen aus $\mathcal{C} = \mathcal{F} \cup \mathcal{T}$ und ggf. Klammern)
 - allerdings: Beschränkung auf „wohlgeformte“ Ausdrücke
- ⇒ üblich: **rekursive Definition** (Präfixnotation):
- Konstanten- und Variablensymbole = symbolische Ausdrücke
 - t_1, \dots, t_n symbolische Ausdrücke und $f \in \mathcal{F}$ (n -stelliges) Funktionssymbol $\implies (ft_1 \dots t_n)$ symbolischer Ausdruck
 - keine anderen Zeichenfolgen sind symbolische Ausdrücke
 - **Beispiele** zu dieser Definition:
 - „ $(+ (* 3 x) (/ 8 2))$ “ ist symbolischer Ausdruck
Lisp- bzw. Scheme-artige Schreibweise, Bedeutung: $3 \cdot x + \frac{8}{2}$
 - „ $27 * (3 /$ “ ist kein symbolischer Ausdruck

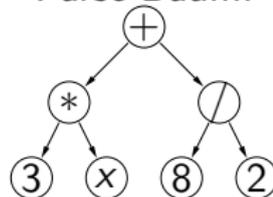
Implementierung

- günstig für GPs: Darstellung der Ausdrücke als Parse-Bäume (werden im Parser z.B. eines Compilers verwendet, um arithmetische Ausdrücke darzustellen und anschließend zu optimieren)

symbolischer Ausdruck:

$(+ (* 3 x) (/ 8 2))$

Parse-Baum:



- in Lisp/Scheme sind Ausdrücke verschachtelte Listen:
erstes Listenelement ist Funktionssymbol (Operator)
nachfolgende Elemente sind Argumente (Operanden)

Ablauf einer Genetischen Programmierung

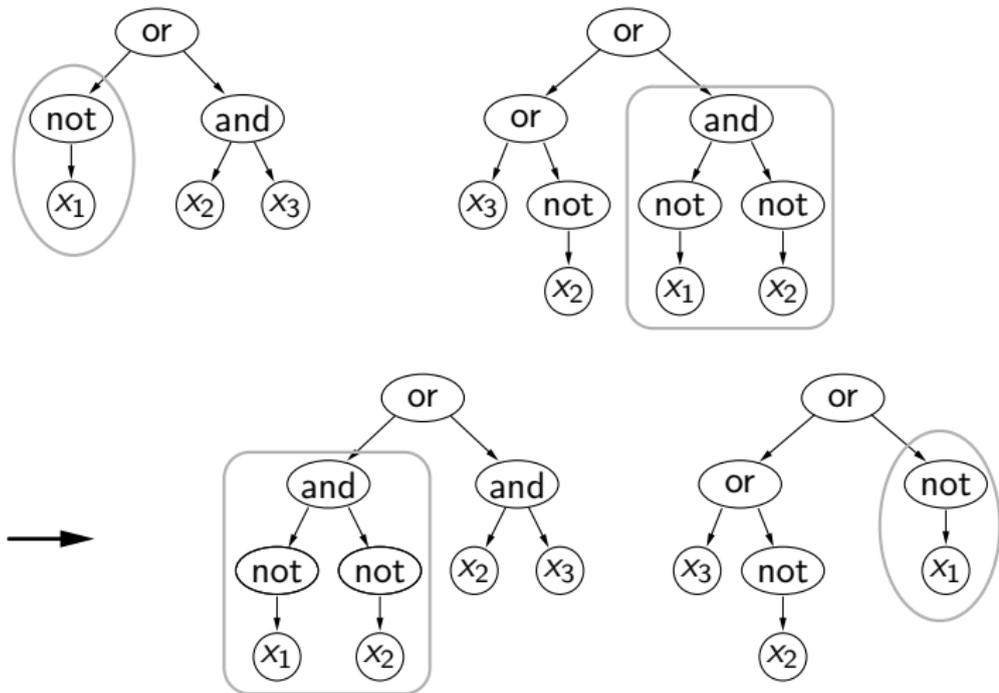
- Erzeugen einer **Anfangspopulation** zufälliger symbolischer Ausdrücke
- **Bewertung** der Ausdrücke durch Berechnung der Fitness
 - Erlernen Boolescher Funktionen: Anteil korrekter Ausgaben für alle Eingaben bzgl. einer Stichprobe
 - Symbolische Regression: Summe der Fehlerquadrate über gegebene Messpunkte
1-D: Daten (x_i, y_i) , $i = 1, \dots, n$, Fitness
$$f(c) = \sum_{i=1}^n (c(x_i) - y_i)^2$$
- **Selektion** mit einem der besprochenen Verfahren
- Anwendung **genetischer Operatoren**, meist nur Crossover

Genetische Operatoren

- für gewöhnlich: initiierte Population hat sehr geringe Fitness
- evolutionärer Prozess verändert anfängliche Population durch genetische Operatoren
- für GPs: viele verschiedene genetische Operatoren
- wichtigsten drei:
 - Crossover
 - Mutation
 - klonale Reproduktion (Kopieren eines Individuums)

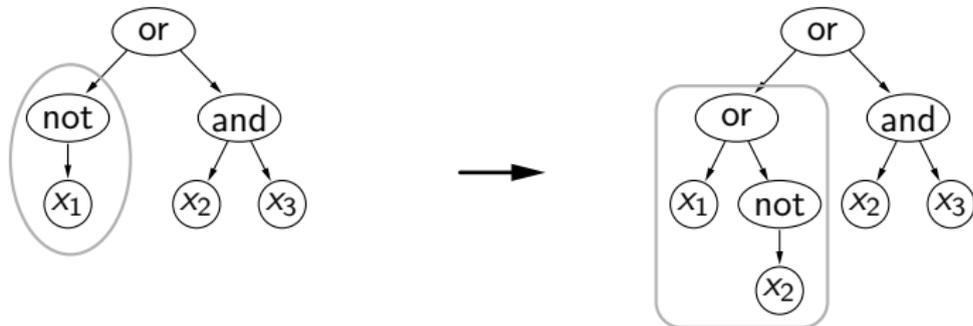
Crossover

- Austausch zweier Teilausdrücke (Teilbäume)



Mutation

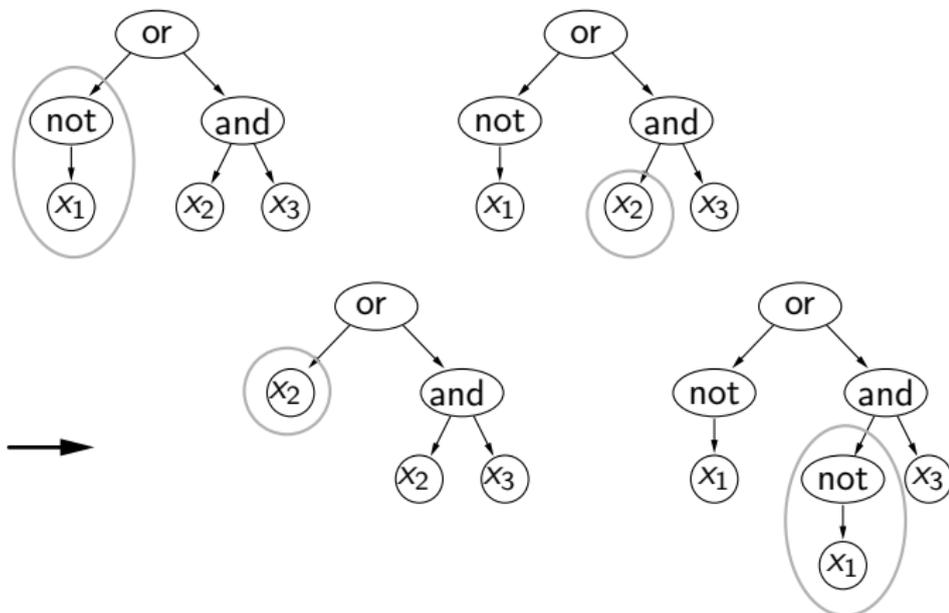
- Ersetzen eines Teilausdrucks (Teilbaums) durch zufällig erzeugten:



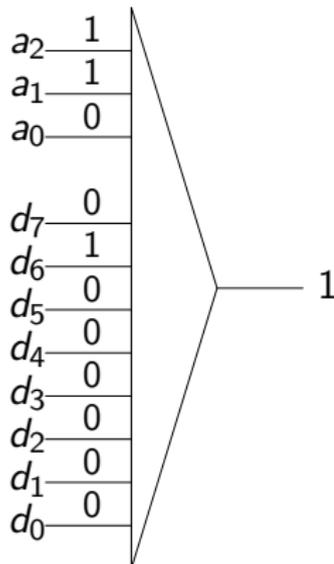
- möglichst nur kleine Teilbäume ersetzen
- bei großer Population: meist nur Crossover und keine Mutation, da hinreichender Vorrat an „genetischem Material“

Vorteil des Crossover

- Crossover von GPs ist mächtiger als Crossover von Vektoren
- Grund: Crossover identischer Elternprogramme führt u.U. zu verschiedenen Individuen



Beispiel: 11-Multiplexer



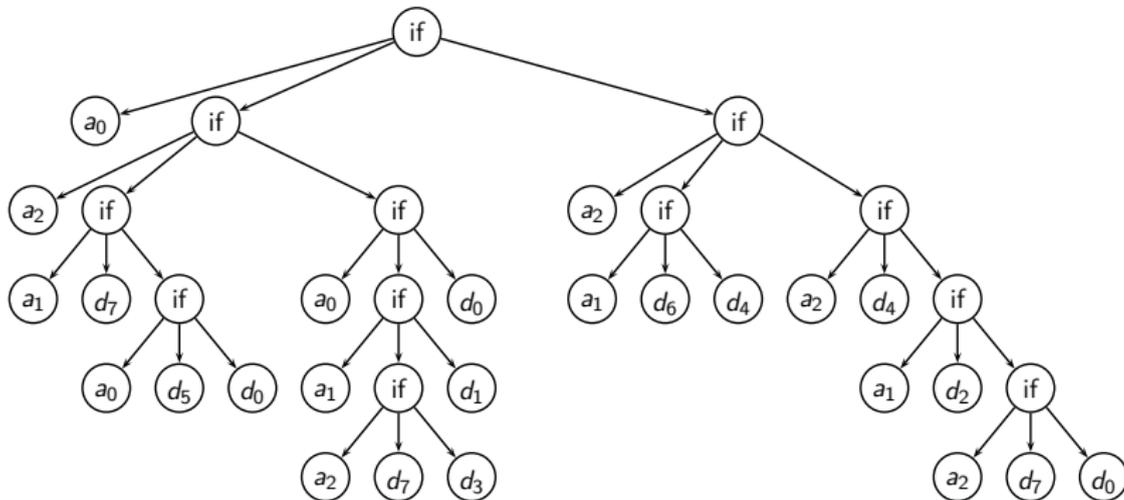
- Multiplexer mit 8 Daten- und 3 Adressleitungen (Zustand der Adressleitungen gibt durchzuschaltende Datenleitung an)
- $2^{11} = 2048$ mögliche Eingaben mit je einer zugehörigen Ausgabe
- Festlegung der Symbolmengen:
 - $\mathcal{T} = \{a_0, a_1, a_2, d_0, \dots, d_7\}$
 - $\mathcal{F} = \{\text{and, or, not, if}\}$
- Fitnessfunktion: $f(s) = 2048 - \sum_{i=1}^{2048} e_i$, wobei e_i Fehler für i -te Eingabe ist

11-Multiplexer

- Populationsgröße $|P| = 4000$
- Anfangstiefe der Parse-Bäume: 6, maximale Tiefe: 17
- Fitnesswerte in Anfangspopulation zwischen 768 und 1280, mittlere Fitness von 1063
(Erwartungswert ist 1024, da bei zufälliger Ausgabe im Durchschnitt Hälfte der Ausgaben richtig)
- 23 Ausdrücke haben Fitness von 1280, einer davon entspricht 3-Multiplexer: (if a_0 d_1 d_2)
- fitnessproportionale Selektion
- 90% (3600) der Individuen werden Crossover unterworfen
- 10% (400) werden unverändert übernommen

11-Multiplexer

- nach nur 9 Generationen: Lösung mit Fitness 2048



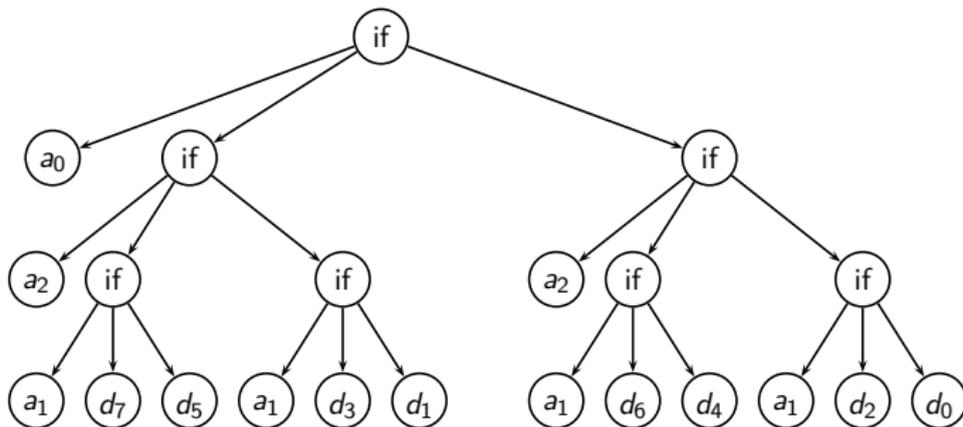
- eher schwer zu interpretieren für Menschen
- kann vereinfacht werden durch Umformung (engl. editing)

Umformung

- asexuelle Operation eines Individuums
 - dient der Vereinfachung durch generelle und spezielle Regeln
 - **generell:** falls Funktion ohne Nebeneffekte im Baum mit konstanten Argumenten, dann evaluiere Funktion und ersetze Teilbaum mit Ergebnis
 - **speziell:** hier Aussagenlogik
 - $\neg(\neg A) \rightarrow A$, $(A \wedge A) \rightarrow A$, $(A \vee A) \rightarrow A$, usw.
 - de Morgan'schen Gesetze, usw.
 - Umformung: z.B. als Operator während GP-Suche
- ⇒ Reduktion aufgeblähter Individuen auf Kosten der Diversität
- normalerweise: Umformung nur zur Interpretation der Ergebnisse

11-Multiplexer

- beste Lösung gestützt durch Aufbereitung:



11-Multiplexer

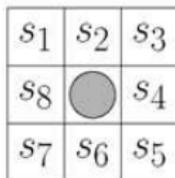
- **beachte:** beste Lösung durch GP ist hierarchisch
- Zerlegung des 11-Multiplexer-Problems anhand zwei kleinerer:
 - 6-Multiplexer: 2 Adressbits und 4 Datenbits
 - 3-Multiplexer: 1 Adressbit und 2 Datenbits
- GP-Lösung = Komposition zweier 6-Multiplexer:
 a_0 wird genutzt, zu entscheiden ob entweder Adressbit a_1 oder a_2 auf d_7, d_5, d_3, d_1 oder d_6, d_4, d_2, d_0 verweisen sollten
- weiterhin: unterste Ebene der 6-Multiplexer ist Komposition zweier 3-Multiplexer

11-Multiplexer

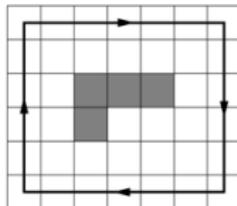
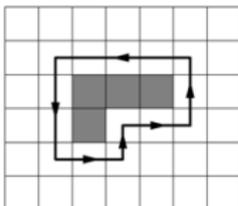
- bestes Individuum in 9. Generation erreicht bestmögliche Fitness
- Frage: wie wahrscheinlich ist dies anhand blinder Suche?
- Schätzung der Zahl aller booleschen Funktionen:
 - Wie viele boolesche Funktionen gibt es für 11 Variablen?
 - Warum ist dieser Wert nicht hinreichend für GPs?
 - Wie viele Möglichkeiten gibt es mit unbeschränkter Baumtiefe?

Beispiel: Stimulus-Response-Agent

- betrachte Stimulus-Response-Agenten in Gitterwelt:



- 8 Sensoren s_1, \dots, s_8 liefern Zustand der Nachbarfelder
- 4 Aktionen: go east, go north, go west, go south
- direkte Berechnung der Aktion aus s_1, \dots, s_8 , kein Gedächtnis
- **Aufgabe:** umlaufe ein im Raum stehendes Hindernis oder laufe Begrenzung des Raumes ab!



Stimulus-Response-Agent

- $\mathcal{T} = \{s_1, \dots, s_8, E, N, W, S, 0, 1\}$, $\mathcal{F} = \{\text{and, or, not, if}\}$
- Vervollständigung der Funktionen, z.B. durch

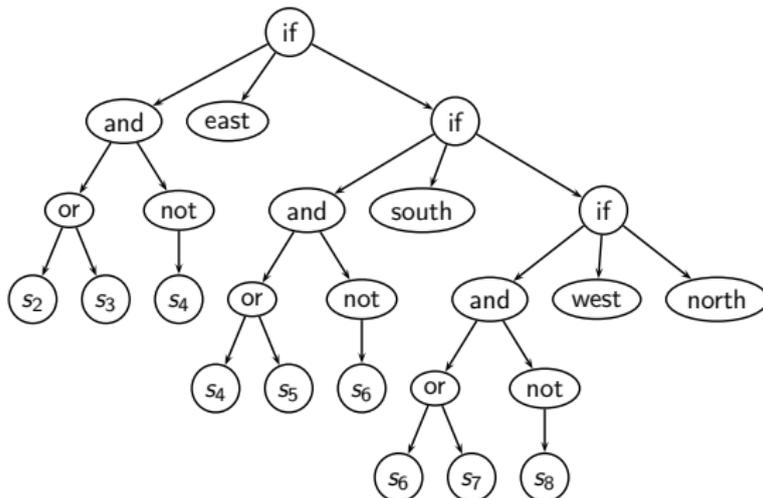
$$(\text{and } x \text{ y}) = \begin{cases} \text{false,} & \text{falls } x = \text{false,} \\ y, & \text{sonst.} \end{cases}$$

(beachte: logische Operation liefert Aktion)

- Populationsgröße $|P| = 5000$
- Aufbau der Nachfolgepopulation
 - 10% (500) Lösungen unverändert übernommen
 - 90% (4500) Lösungen durch Crossover erzeugt
 - <1% Lösungen mutiert
- 10 Generationen (ohne Anfangspopulation) berechnet

Stimulus-Response-Agent

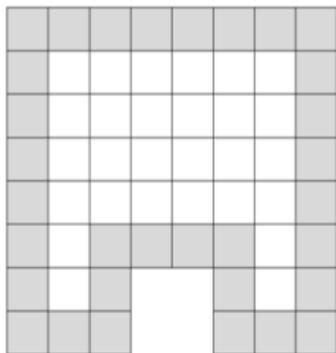
- optimale, von Hand konstruierte Lösung:



- Finden genau dieser Lösung: höchst unwahrscheinlich
- ⇒ Strafterm als Maß der Komplexität eines Ausdrucks
- ⇒ Einfachheit der Chromosomen

Stimulus-Response-Agent

- Bewertung einer Kandidatenlösung anhand eines Testraumes:



- perfekt arbeitendes Steuerprogramm: Agenten läuft grau gezeichneten Felder ab
- Startfeld zufällig
- falls Aktion nicht ausführbar oder statt Aktion Wahrheitswert geliefert, dann Ausführung des Steuerprogramms abbrechen
- pro Chromosom: 10 zufällige Startfelder, Bewegung wird verfolgt
- Zahl der insgesamt besuchten Randfelder = Fitness (maximal $10 \cdot 32 = 320$)

Wandverfolgung

meisten der 5000 Programme in Generation 0 sind nutzlos:

(and sw ne)

- wertet nur aus und terminiert dann
- Fitness von 0

(or east west)

- liefert manchmal west und geht somit einen Schritt nach Westen
- landet manchmal neben einer Wand
- Fitness von 5

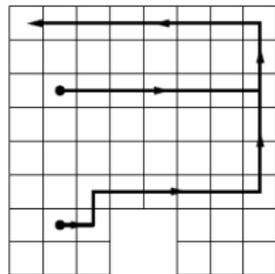
bestes Programm hat Fitness von 92:

- schwer zu lesen, hat redundante Operatoren
- Weg mit zwei Startpunkten auf nächster Folie beschrieben (Osten bis zu Wand, dann Norden bis nach Osten oder Westen möglich und dann in Ecke oben links gefangen)

Bestes Individuum der Generation 0

```
(and (not (not (if (if (not s1)
                    (if s4 north east)
                    (if west 0 south))
                (or (if s1 s3 s8) (not s7))
                (not (not north))))))
(if (or (not (and (if s7 north s3)
                  (and south 1)))
        (or (or (not s6) (or s4 s4))
            (and (if west s3 s5)
                (if 1 s4 s4))))
    (or (not (and (not s3)
                  (if east s6 s2)))
        (or (not (if s1 east s6))
            (and (if s8 s7 1)
                (or s7 s1))))
    (or (not (if (or s2 s8)
                  (or 0 s5)
                  (or 1 east)))
        (or (and (or 1 s3)
                  (and s1 east))
            (if (not west)
                (and west east)
                (if 1 north s8))))))
```

Bestes Individuum in Generation 0:

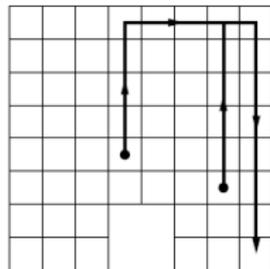


(Bewegung von zwei
Startpunkten aus)

Beste Individuen der Generationen 2 und 6

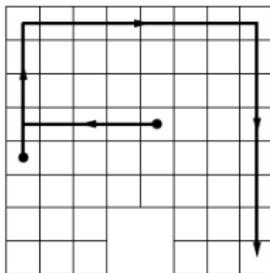
Bestes Individuum in Generation 2:

```
(not (and (if s3
           (if s5 south east)
           north)
         (and not s4)))
```

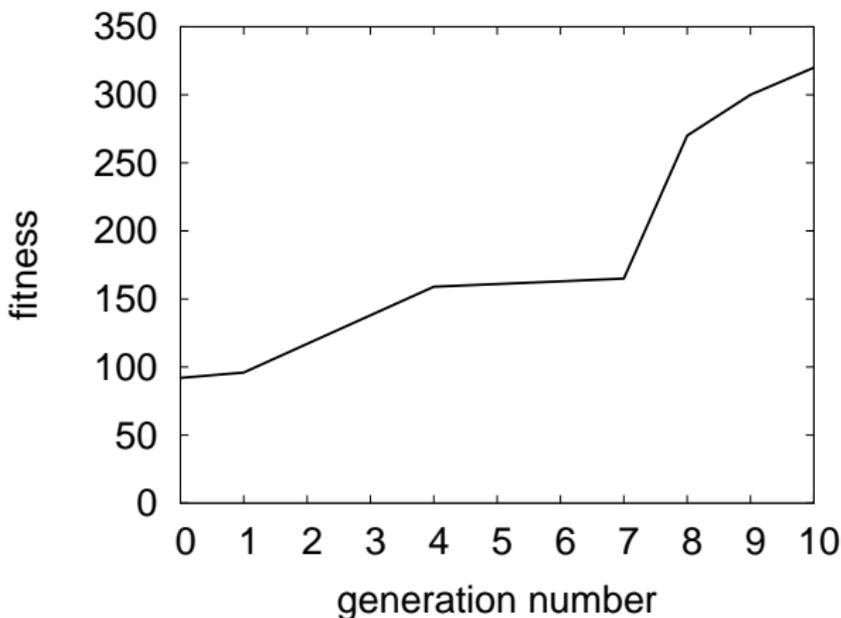


Bestes Individuum in Generation 6:

```
(if (and (not s4)
         (if s4 s6 s3))
    (or (if 1 s4 south)
        (if north east s3))
    (if (or (and 0 north)
            (and s4 (if s4
                       (if s5 south east)
                       north))))
        (and s4 (not (if s6 s7 s4)))
        (or (or (and s1 east) west) s1)))
```



Entwicklung der Fitness



Entwicklung der Fitness im Laufe des Lernvorgangs
(bestes Individuum der jeweiligen Generation)

Übersicht

1. Biologische Grundlagen
2. Grundlagen evolutionärer Algorithmen
3. Genetische Programmierung
- 4. Reale Anwendungsbeispiele**
Antennenplatzierung

Antennenplatzierung

[Weicker et al., 2003]

- Basisantennen für Mobilfunknetze
- erstes Ziel: hohe Netzverfügbarkeit
- zweites Ziel: geringe Kosten
- übliche Vorgehensweise:
 - Basisantennen platzieren und Größe/Reichweite konfigurieren
⇒ Bedarf abdecken
 - Frequenzen zuweisen ⇒ Interferenzen minimal halten

Ausgangssituation

- beide Probleme sind \mathcal{NP} -hart
- Platzierung kann Frequenzzuweisung stark einschränken
- in einer Iteration können Ergebnisse der Frequenzzuweisung nur bedingt in Platzierung wieder einfließen

Grundsatzentscheidung:

- beide Probleme werden gleichzeitig bearbeitet

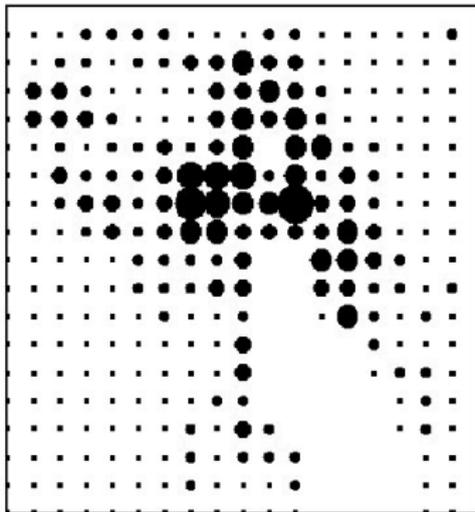
Formalisierung

- rechteckiges Gebiet (x_{\min}, y_{\min}) und (x_{\max}, y_{\max}) mit Rasterung res
- Menge aller (mögliche) Positionen:

$$Pos = \left\{ (x_{\min} + i \cdot res, y_{\min} + j \cdot res) \mid 0 \leq i \leq \frac{x_{\max} - x_{\min}}{res} \right. \\ \left. \text{und } 0 \leq j \leq \frac{y_{\max} - y_{\min}}{res} \right\}$$

Gesprächsbedarf Zürich

- statistisch ermitteltes Gesprächsaufkommen $bedarf(zelle) \in \mathbb{N}$
für einige $zelle \in Pos$



Formalisierung: Antenne

- Antenne $t = (pow, cap, pos, freq)$
- Sende-/Empfangsstärke $pow \in [MinPow, MaxPow] \subset \mathbb{IN}$
- Gesprächskapazität $cap \subset [0, MaxCap] \subset \mathbb{IN}$
- Frequenzen/Kanäle $freq \subset Frequ = \{f_1, \dots, f_k\}$ mit $|freq| \leq cap$
- alle möglichen Antennenkonfigurationen:

$$T = [MinPow, MaxPow] \times [0, MaxCap] \times Pos \times Frequ$$

Genotyp

- problemnaher Genotyp

$$\Omega = \mathcal{G} = \{ \{t_1, \dots, t_k\} \mid k \in \mathbb{N} \text{ und } \forall i \in \{1, \dots, k\} : t_i \in T \}$$

- variable Länge

Randbedingungen

- Netzverfügbarkeit hat oberste Priorität \Rightarrow als harte Randbedingung formuliert
- erreichbare Positionen gemäß Wellenverbreitungsmodell:
 $wp : Pos \times [MinPow, MaxPow] \rightarrow \mathcal{P}(Pos)$
- $A.G = (t_1, \dots, t_k)$ heißt legal, wenn für jedes t_i eine Zuordnung $bedient(t_i, zelle) \in \mathbb{N}$ (mit $zelle \in Pos$) existiert, sodass
 - $bedient(t_i, zelle) > 0 \Rightarrow zelle \in wp(t_i)$
 - $\sum_{i=1}^k bedient(t_i, zelle) \geq bedarf(zelle)$
 - $\sum_{zelle \in Pos} bedient(t_i, pos) \leq cap$ mit $t_i = (pow, cap, pos, freq)$

Bewertungsfunktionen

- Störungen durch Antennen mit gleichen oder eng beieinander liegenden Frequenzen in einer Zelle

$$f_{interferenz}(A) = \frac{\sum_{i=1}^k \#gestörteGespräche(t_i)}{\sum_{zelle \in Pos} bedarf(zelle)}$$

- Kosten $kosten(pow_j, cap_i)$ pro Antenne

$$f_{kosten}(A) = \sum_{i=1}^k kosten(t_i)$$

„Entwurfsmuster“

- nur legale Individuen, daher Reparaturfunktion notwendig
- jede Antennenkonfiguration muss noch erreichbar sein
- verlängernde und verkürzende Operatoren halten sich die Waage
- Feinabstimmung und Erforschung sind ausgeglichen
⇒ problemspezifische und zufällige Operatoren

Reparaturfunktion

- Zellen in einer zufälligen Reihenfolge besuchen
- falls deren Bedarf nicht gedeckt ist:
 - bei Existenz mindestens einer Antennen mit freier Kapazität: die stärkste Antenne wählen und Frequenzen zuweisen
 - ggf. diejenige Antenne ermitteln, die kostenminimal durch Erhöhung der Stärke den Bedarf decken kann
 - ggf. prüfen, welche Kosten durch eine neue Antenne unmittelbar bei der Zelle entstehen
 - ggf. Lösung (2) oder (3) umsetzen

Reparaturfunktion

Einsatz

- auf jedes neu erzeugte Individuum
- zur Initialisierung der Anfangspopulation
 - Reparaturfunktion auf leeres Individuum
 - max. $2^{|Pos|}$ Individuen durch mögliche zufällige Reihenfolge der Bedarfzellen

Mutationsoperatoren

- 6 „gerichtete“ Mutationen, die spezieller Idee folgen
- 5 „zufällige“ Mutationen

Gerichtete Mutationsoperatoren

Name	Wirkung
DM1	falls Antenne unbenutzte Frequenzen hat ⇒Kapazität entsprechend reduzieren
DM2	falls Antenne maximale Kapazität nutzt ⇒weitere Antenne mit Standardeinstellungen in der Nähe platzieren
DM3	falls Antennen große überlappende Regionen haben ⇒eine Antenne entfernen
DM4	falls Antennen große überlappende Regionen haben ⇒Stärke einer Antenne reduzieren, so dass dennoch alle Anrufe bedient
DM5	falls Interferenzen vorkommen ⇒involvierende Frequenzen verändern
DM6	falls Antenne nur kleine Anzahl von Anrufen hat ⇒Antenne entfernen

Zufällige Mutationsoperatoren

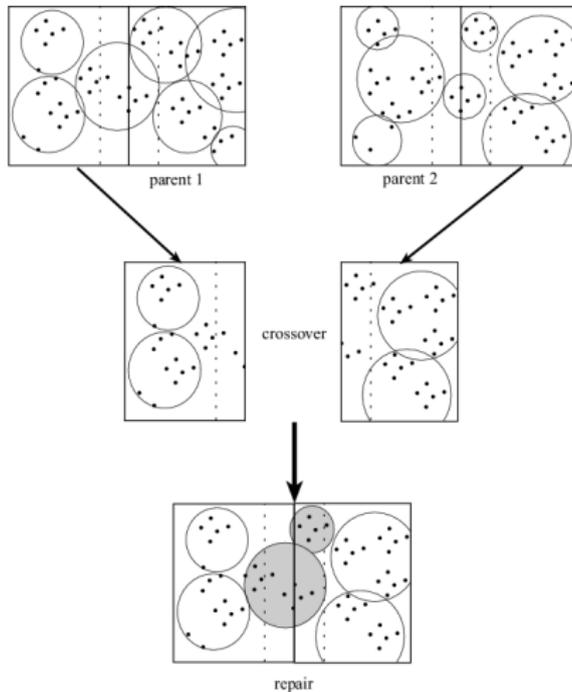
Name	Wirkung
RM1	Position einer Antenne ändern (Stärke und Kapazität unverändert, Frequenzen neu durch Reparaturfunktion)
RM2	komplett zufälliges Individuum (wie in Initialisierung)
RM3	Stärke einer Antenne zufällig ändern ⇒Ausgleich zu <i>DM4</i>
RM4	Kapazität einer Antenne zufällig verändern ⇒Ausgleich zu <i>DM1</i>
RM5	zugeordnete Frequenzen einer Antenne ändern ⇒Ausgleich zu <i>DM5</i>

Rekombination

- Gesamtheit in zwei Hälften teilen (vertikal oder horizontal)
- pro Hälfte Antennen eines Individuums übernehmen
- Korridor um Grenze durch Reparaturalgorithmus füllen

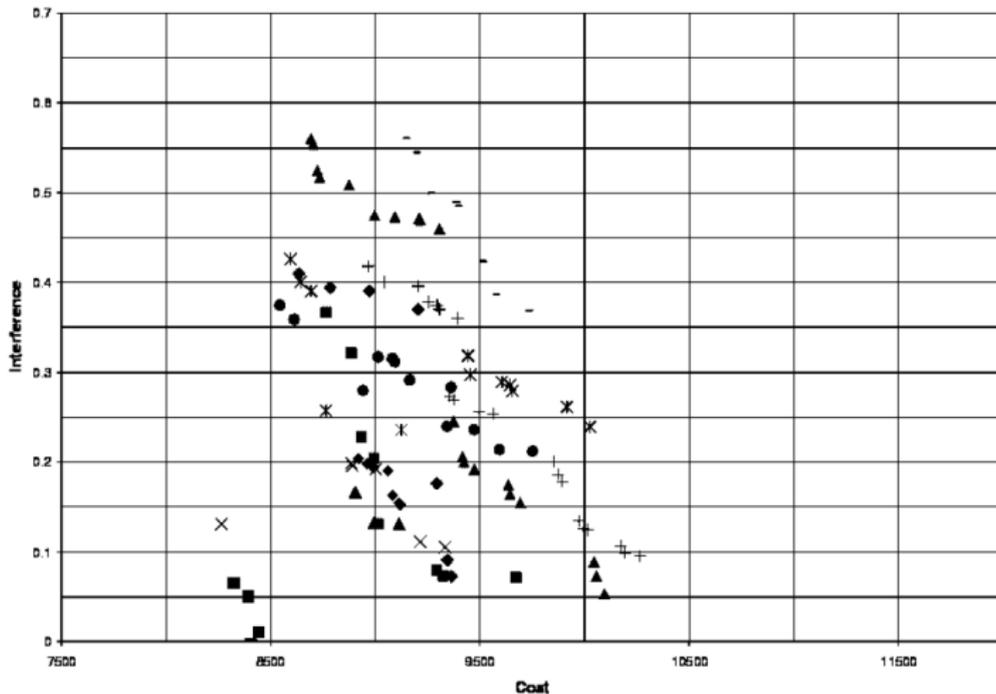
Rekombination

Ein Beispiel



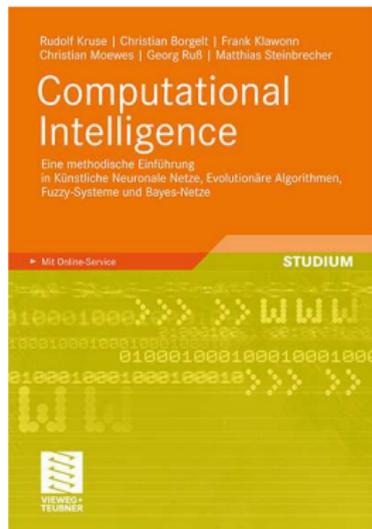
Pareto-Front

eigene Selektion, $p_{RM} = p_{DM} = 0.3$ und $p_{Rek} = 0.4$



Wer mehr wissen will, liest. . .

[Kruse et al., 2011]
oder besucht unsere **EA-Vorlesung** im
Sommer



Schätzen verdeckter Fahrereigenschaften

[Held, 2011]

Zielstellung:

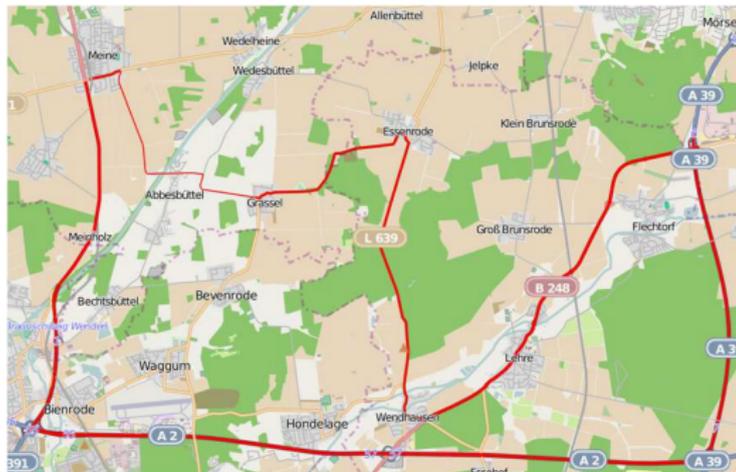
- Analyse des Fahrverhaltens
- Extraktion von Kenngrößen
- Algorithmus zum Schätzen verdeckter Fahrparameter

Grundannahme:

- Fahrer mit ähnlichen Eigenschaften verhalten sich ähnlich

Fahrerleistungsdatenbank

- 54 aufgezeichnete Fahrten
- 11 Probanden
- 108 km Gesamtstrecke
- größtenteils Landstraße und Autobahn



Simulatorstudie

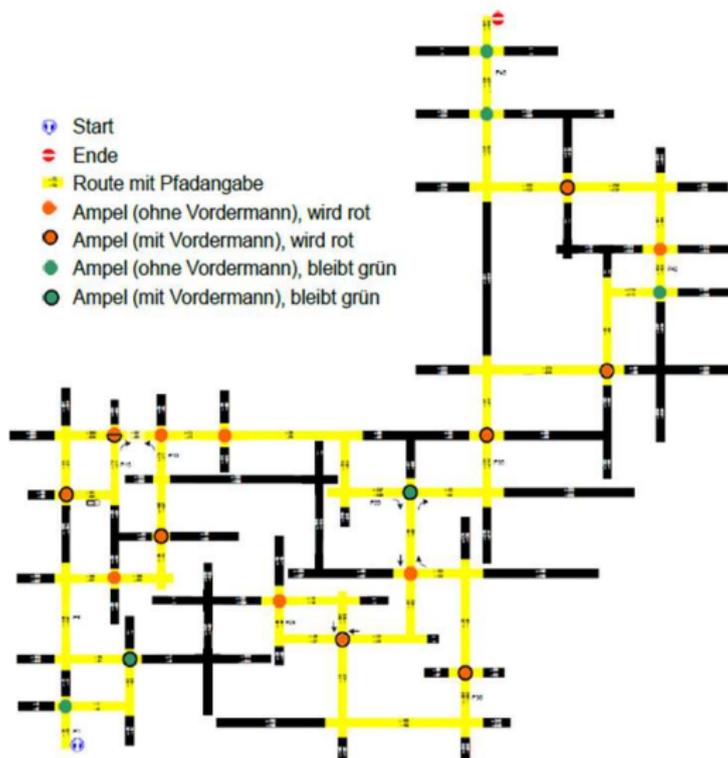
21 aufgezeichnete Fahrten

Landstraße:

- 12 km Strecke
- 11 min Dauer
- 80 km/h
Richtgeschwindigkeit
- Überholmanöver

innerorts:

- 15 km Strecke
- 25 min Dauer
- Ampelsituationen



Zusatzstudie

26 aufgezeichnete Fahrten

Autobahn:

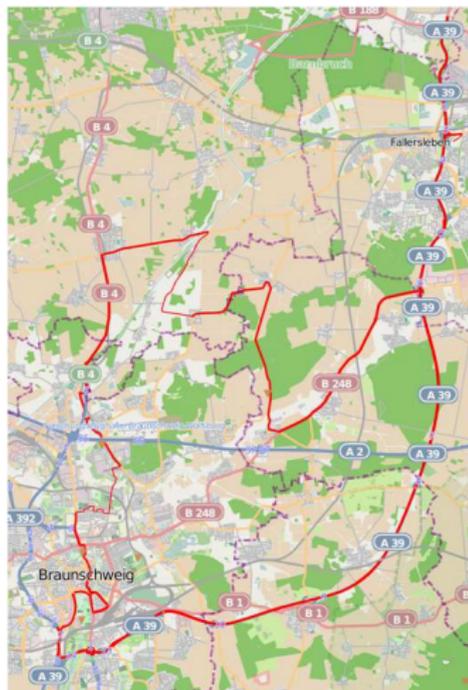
- 44 km Strecke
- zweispurig

Stadt:

- 15 km Strecke
- Ampeln und Wohngebiete

außerorts:

- 42 km Strecke
- Bundes- und Landstraßen, Autobahn



Fahrparameter und Kenngrößen

Abstand:

- Absoluter Abstand
- Zeitlücke
- Time To Collision

Beschleunigung:

- Längsbeschleunigung
- Querbeschleunigung

Fahrpedalwert

Geschwindigkeit:

- Differenz zur erlaubten Geschwindigkeit
- Richtgeschwindigkeit

berechnete Größen:

- arithmetisches Mittel
- Varianz
- Anteile über bestimmte Werte

Ähnliche Fahrten

untersuchte Fahrt

0.2	0.7	0.4	0.8	0.6	0.3	0.1	0.2
-----	-----	-----	-----	-----	-----	-----	-----

Referenzfahrt 1

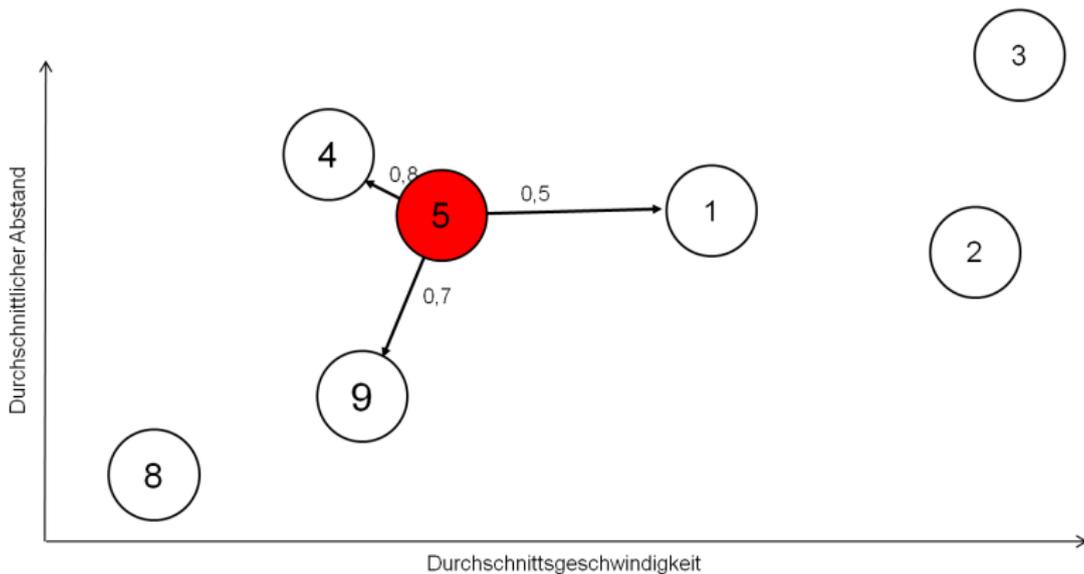
0.6	0.8	0.9	0.6	0.4	0.4	0.3	0.6
-----	-----	-----	-----	-----	-----	-----	-----

Referenzfahrt 2

0.4	0.3	0.6	0.5	0.1	0.7	0.4	0.9
-----	-----	-----	-----	-----	-----	-----	-----

$$s(x, y) = 1 - \sum_{i=1}^n k_i (x_i - y_i)^2$$

Schätzen der Zielgröße



$$\hat{z} = \frac{\sum_{i=1}^3 S_i \cdot Z_i}{\sum_{i=1}^3 S_i}$$

Evolutionärer Algorithmus

Anfangspopulation:	50 zufällige Individuen
Zwischenpopulation:	50 Individuen
Zufallsauswahl:	Glücksradauswahl mit dynamisch skaliertes Fitness
keine Veränderung:	10 zufällige Individuen (darunter Top-3)
Mutation:	20 Individuen (Mutationsw'keit: $4/n$)
Crossover:	20 Individuen aus je zwei Eltern (uniformes Crossover)

Kodierung der Individuen

Beispielindividuum

0	0.3	0	0.2	0.1	0.4	0	0
---	-----	---	-----	-----	-----	---	---

- jedes Gen repräsentiert Gewicht für jeweiligen Parameter
- Summe der Gewichte: normiert auf 1
- Normierung der Individuen nach Mutation und Crossover

Fitness

- r : Korrelationskoeffizient zwischen z_i und \hat{z}_i
- n : Anzahl der genutzten Parameter
- f : Fitness der Parameterkonfiguration

$$f = r - \frac{r \cdot n}{100}$$

- dynamische Skalierung für Glücksradauswahl:

$$f^* = \max\{f, 0.01\}^{\max_{i \in \{1, \dots, n\}} \{f_i\}}$$

Fazit

- Einschätzung bzgl. Zielwerte möglich
- hohe Korrelationen bei realen Zielwerten
- niedrige Korrelation bei Zufallswerten
- automatisches Lernen mit evolutionären Algorithmen

Untersuchung	f	n	r
Simulatorstudie	0.72	10	0.80
Zielgröße A	0.62	13	0.71
Zielgröße B	0.50	12	0.57
Zielgröße C	0.62	9	0.67
Zufallswerte	0.30	16	0.36

Literatur zur Lehrveranstaltung I



Held, P. (2011).

Schätzen verdeckter fahreigenschaften auf basis des fahrverhaltens.

Masterarbeit, Otto-von-Guericke-Universität Magdeburg.



Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Ruß, G., and Steinbrecher, M. (2011).

Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze.

Vieweg+Teubner-Verlag, Wiesbaden.



Weicker, N., Szabo, G., Weicker, K., and Widmayer, P. (2003).

Evolutionary multiobjective optimization for base station transmitter placement with frequency assignment.

In *IEEE Trans. on Evolutionary Computing*, volume 7, pages 189–203.