

# Intelligente Systeme

## Computational Intelligence

**Prof. Dr. R. Kruse    C. Braune**

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

Institut für Wissens- und Sprachverarbeitung

Fakultät für Informatik

Otto-von-Guericke Universität Magdeburg

# Übersicht

## 1. Computational Intelligence

## 2. Techniken der Computational Intelligence

## 3. Reale Beispiele

## 4. Schwarm- und populationsbasierte Optimierung

## 5. Teilchenschwarmoptimierung

## 6. Ameisenkolonieoptimierung

# Computational *intelligence* Society

*Home of Neural Networks, Fuzzy Systems and Evolutionary Computation*



Computational Intelligence ist charakterisiert durch:

Meist modellfreie Ansätze,

Kein explizites Modell (z.B. Differentialgleichungen) notwendig

Approximation statt exakte Lösung (nicht immer ausreichend!)

schnelleres Finden einer brauchbaren Lösung (u.U. auch ohne tiefgehende Problemanalyse)

# CI: Technologien und Anwendungen

## Kerntechnologien

Neuronale Netze (NN)  
Fuzzy-Logik (FL)  
Probabilistisches Schließen (PR)  
Evolutionäre Algorithmen (EA) und  
weitere Metaheuristiken  
Hybride Systeme

## Verwandte Technologien

Fallbasiertes Schließen (CBR)  
Regelbasierte Expertensysteme (RBR)  
Maschinelles Lernen (z.B.  
Induktionsbäume)  
Bayessche Belief-Netze (BBN)

## Anwendungen

### Klassifikation

- Überwachung/  
Anomalieerkennung
- Diagnose
- Prognose
- Konfiguration/Initialisierung

### Vorhersage

- Qualitätskontrolle
- Alterungsprozessmodellierung

### Terminplanung

- Zeit-/Ressourcenzuweisung

### Regelung

- Maschinen-/Prozesskontrolle
- Prozessinitialisierung
- Überwachungskontrolle

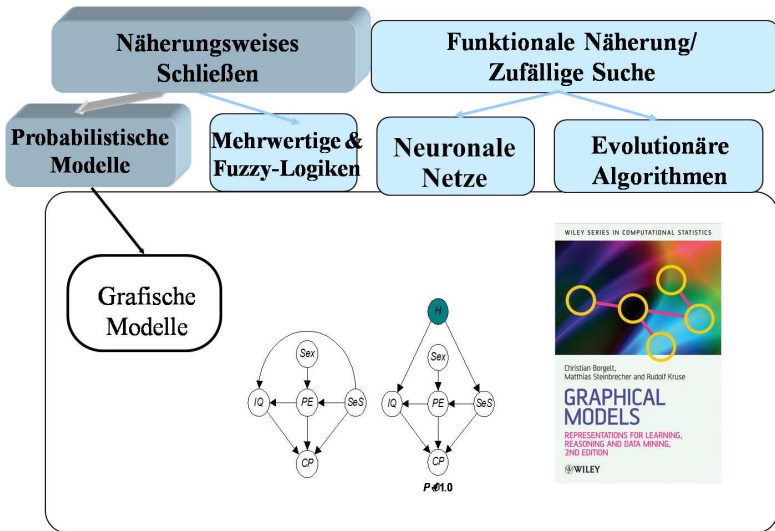
### Automatische Entscheidung

- Kosten-/Risikoanalyse
- Einkommensoptimierung

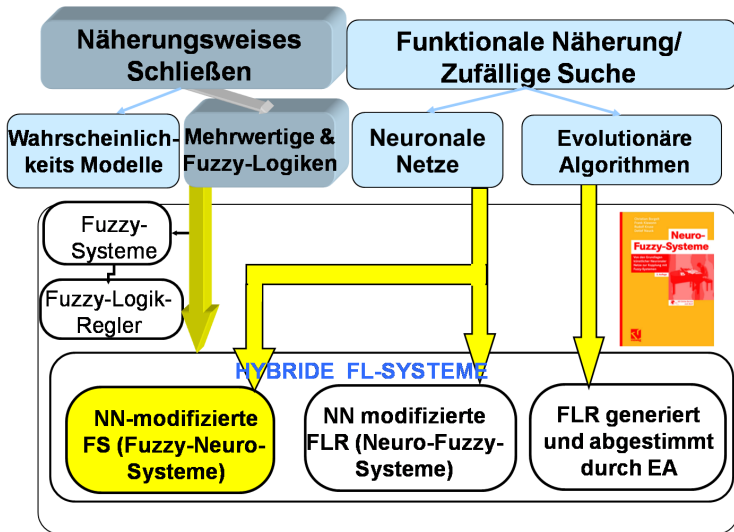
# Übersicht

1. Computational Intelligence
- 2. Techniken der Computational Intelligence**
3. Reale Beispiele
4. Schwarm- und populationsbasierte Optimierung
5. Teilchenschwarmoptimierung
6. Ameisenkolonieoptimierung

# Computational Intelligence



# Hybride Neuro-Fuzzy-Systeme



# CI: Anwendungen



## Haushaltsgeräte

- Bevorzugte Serviceverträge (Stat.)
- Call-Center-Support (CBR)



## Kapitalservice

- Kreditwürdigkeitsbewertung (Fusion/FL/CBR)



## Finanzversicherungen

- Bevorzugte Kunden (Stat./NN)



## Plastik

- Automatische Farbanpassung (CBR)



## Föderierte Systeme

- Terminwartung von Satellitenkonstellationen (GA)



## Schifffahrt

- Schiffmanagementsysteme (AI/GA)



## Medizinische Systeme

- Automatische Analyse von MRT (FL)
- Reverse Engineering von Picker (FL)
- Finite-Element-Analyse (FL)
- Analyse von Röntgenfehlern (CBR)



## Flugzeugmotoren

- Zentrum für Ferndiagnosen (CBR)
- Kundenanfragecenter (CBR)
- Ausreißerererkennung (FL/Stat.)
- Wartungsberater (NN/FL)
- Sensorfusion (FL)



## Transportsysteme

- Erfassung von Transport-DB (CBR)
- Prototyp. Zugführerkontrolle (FL/GA)
- Prototyp. Trendanalyse (Stat.)
- Eingebettete/Ferndiagnose (BBN)



## Stromerzeugung

- Fernausreißerererkennung (Stat.)
- Eingebettete/Ferndiagnose (BBN)
- Call-Center-Problem/Lösung (CBR)

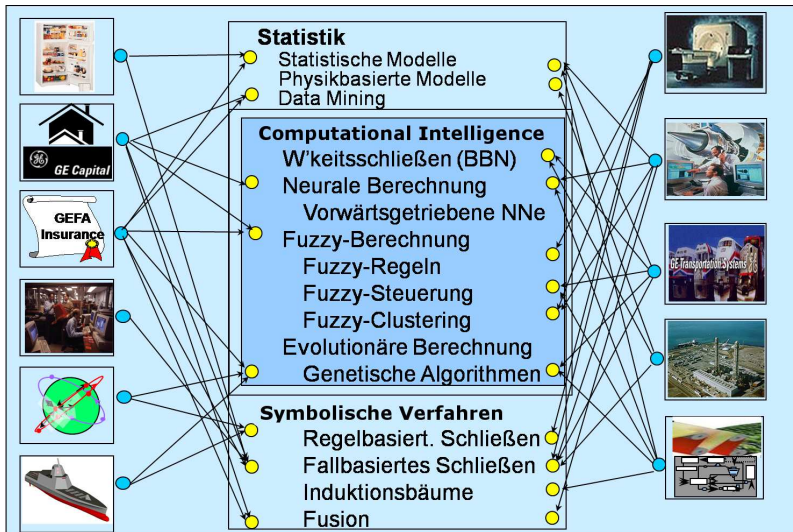


## Industriesysteme

- Vorhersage eines Abbruchs der Papierbahn (NN/Stat./Induction)
- Rührkontrolle von Zement (FL/GA)



# CI: Grundlage und verwandte Technologien



# Übersicht

1. Computational Intelligence

2. Techniken der Computational Intelligence

**3. Reale Beispiele**

DAX-Prognosen

Neuro-Fuzzy-Systeme

Qualitätskontrolle

4. Schwarm- und populationsbasierte Optimierung

5. Teilchenschwarmoptimierung

6. Ameisenkolonieoptimierung

# Beispiel: DAX-Prognosen

Datenbasis: Zeitreihe von 1986–1997

DAX	Composite-DAX
Deutsche 3-Monate-Zinsrate	Return Germany
Deutscher Morgan-Stanley-Index	Industrie-Index des Dow Jones
DM / US-\$	US Schatzanweisungen
Goldpreis	Japanischer Nikkei-Index
Europäischer Morgan-Stanley-Index	Verhältnis von Preis und Ertrag

# Fuzzy-Regeln in der Finanzwelt

## Entwicklungsregel

**WENN** DAX = fallend **UND** US-\$ = fallend  
**DANN** DAX-Voraussage = fallend  
**MIT** hoher Gewissheit

## Wendepunktregel

**WENN** DAX = fallend **UND** US-\$ = steigend  
**DANN** DAX-Voraussage = steigend  
**MIT** niedriger Gewissheit

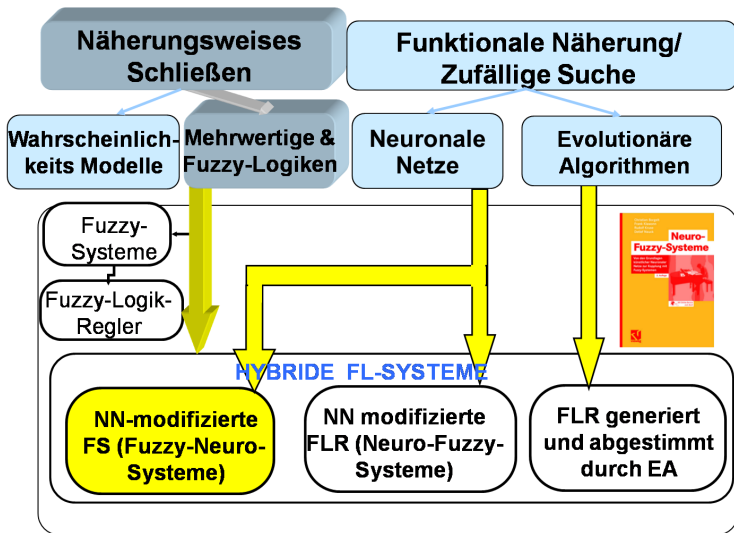
## Verzögerungsregel

**WENN** DAX = stabil **UND** US-\$ = fallend  
**DANN** DAX-Voraussage = fallend  
**MIT** sehr hoher Gewissheit

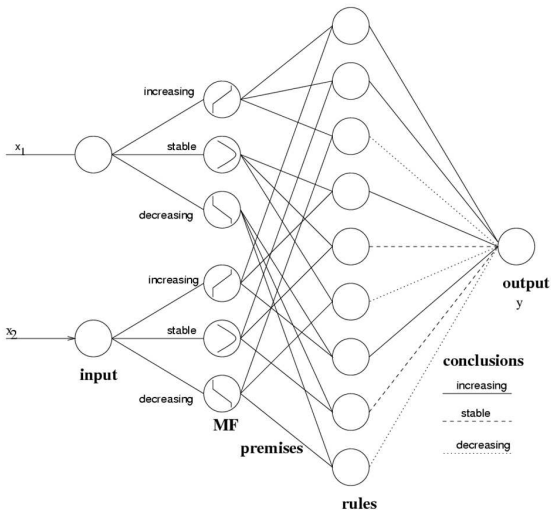
## generell

**WENN**  $x_1$  ist  $\mu_1$  **UND**  $x_2$  ist  $\mu_2$  **UND** ... **UND**  $x_n$  ist  $\mu_n$   
**DANN**  $y = \eta$   
**MIT** Gewicht  $k$

# Neuro-Fuzzy-Architektur



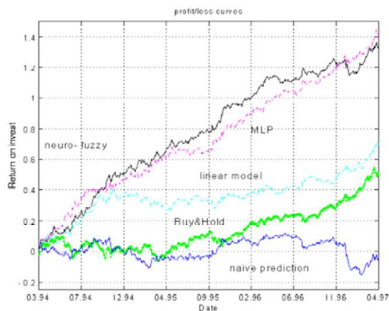
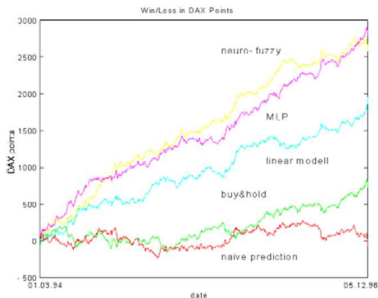
# Neuro-Fuzzy-Architektur



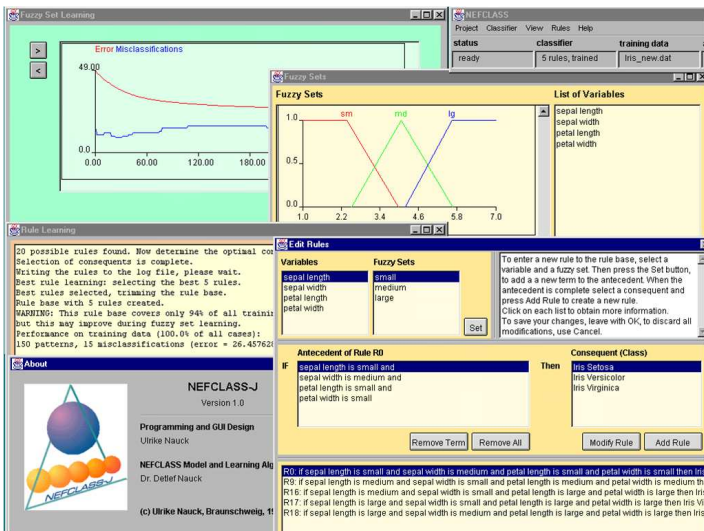
# Return-on-Investment-Kurven

verschiedene Modelle

Validierungsdaten: 01. März 1994 bis April 1997



# NEFCLASS-J



The screenshot displays the NEFCLASS-J software interface, which is used for fuzzy set learning and rule-based classification. The interface is divided into several windows:

- Fuzzy Set Learning:** A graph showing the error rate (misclassifications) over time. The error rate starts at 49.00 and decreases to approximately 26.45% after 180 iterations.
- Fuzzy Sets:** A graph showing the membership functions for three fuzzy sets: 'sm' (small), 'md' (medium), and 'lg' (large). The x-axis represents a variable value (e.g., sepal length), and the y-axis represents the membership degree (0.0 to 1.0).
- List of Variables:** A list of variables used in the fuzzy sets: sepal length, sepal width, petal length, and petal width.
- Rule Learning:** A text window showing the progress of rule learning. It indicates that 20 possible rules were found, the best rule learning is complete, and a rule base with 5 rules was created. A warning message states: "WARNING: This rule base covers only 94% of all training but this may improve during fuzzy set learning. Performance on training data (100.0% of all cases): 150 patterns, 15 misclassifications (error = 26.45762)".
- Edit Rules:** A window for editing rules. It shows the antecedent and consequent of a rule. The antecedent is "sepal length is small and sepal width is medium and petal length is small and petal width is small". The consequent is "Iris Setosa", "Iris Versicolor", and "Iris Virginica".
- About:** A window providing information about the software, including the logo, version (1.0), and the authors (Ulrike Nauck and Dr. Delfel Nauck).



# Beispiel: Qualitätskontrolle

## Heutiges Verfahren

Oberflächenkontrolle: manuell durchgeführt

Erfahrener Arbeiter bearbeitet Oberfläche  
mit Schleifstein

Experten klassifizieren Abweichungen durch  
sprachliche Beschreibungen

Umständlich, subjektiv, fehleranfällig,  
zeitaufwendig



## Vorgeschlagener Ansatz:

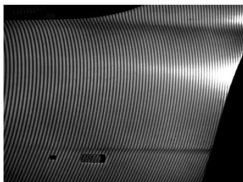
Digitalisierung der Oberfläche mit optischen Mess-Systemen

Charakterisierung der Formabweichungen durch mathematische  
Eigenschaften (nahe der subjektiven Merkmale)

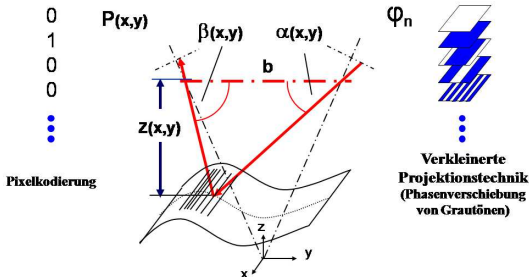
# Topometrisches 3D-Mess-System



breuckmann 

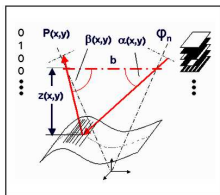


## Triangulation und Gitterprojektion

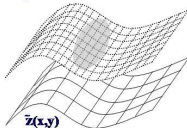


- hohe Punktdichte
- schnelle Datenansammlung
- genaue Messung
- kontakt- und harmlos

# Datenverarbeitung



• Annäherung durch  
Polynomiale Oberfläche



• Differenz

$z(x,y)$

$\tilde{z}(x,y)$

$Dz(x,y)$

• Farbkodierte  
Darstellung



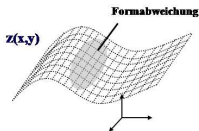
3-D-Daten-  
aufnahme

Nachverarbeitung

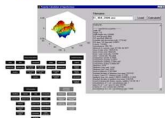
Erkennung von  
Abweichungen

Merkmalsanalyse

• 3-D-Punktwolke



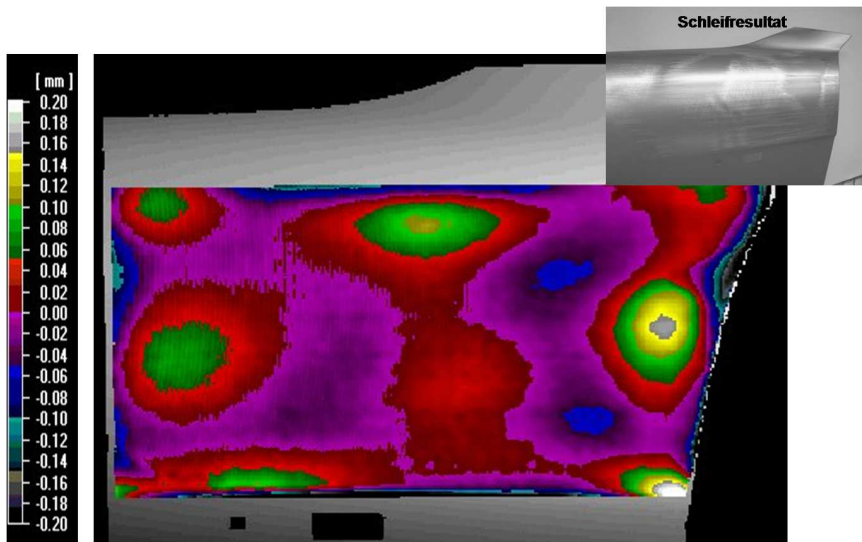
• Merkmalsberechnung



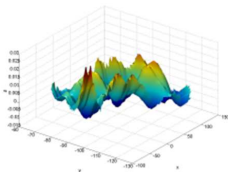
• Klassifikation (Data-Mining)



# Farbkodierte Darstellung

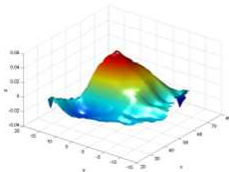


# 3D-Darstellung lokaler Oberflächendefekte



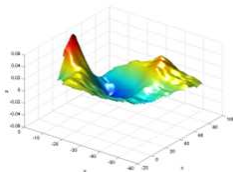
**Unebene Oberfläche**

Mehrere Einfallstellen in Serie/benachbart



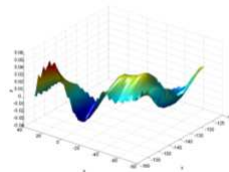
**Walzenmarkierung**

Lokale Glättung der Oberfläche



**Einfallstelle**

Leichte flach basierte Senke einwärts



**Wellplatte**

Mehrere schwerere Faltungen in Serie

# Charakteristik der Daten

9 Meisterstücke mit insgesamt 99 Defekten analysiert

Für jeden Defekt wurden 42 Merkmale berechnet

Typen sind eher unbalanciert

Seltene Klassen wurden verworfen

Einige extrem korrelierte Merkmale wurden verworfen (31 übrig)

Rangfolge der 31 Merkmale nach Wichtigkeit

Geschichtete 4-fache Kreuzvalidierung fürs Experiment

# Anwendung und Ergebnisse

## Regelbasis für NEFCLASS:

### Rule base

- ☞  Rule 1: IF (max\_distance\_to\_cog IS fun 2 AND min\_extrema IS fun 1 AND max\_extrema IS fun 1) THEN type IS press\_mark
- ☞  Rule 2: IF (max\_distance\_to\_cog IS fun 2 AND all\_extrema IS fun 1 AND max\_extrema IS fun 2) THEN type IS sink\_mark
- ☞  Rule 3: IF (max\_distance\_to\_cog IS fun 3 AND min\_extrema IS fun 2 AND max\_extrema IS fun 2) THEN type IS uneven\_surface
- ☞  Rule 4: IF (max\_distance\_to\_cog IS fun 2 AND min\_extrema IS fun 2 AND max\_extrema IS fun 2) THEN type IS uneven\_surface
- ☞  Rule 5: IF (max\_distance\_to\_cog IS fun 2 AND all\_extrema IS fun 1 AND min\_extrema IS fun 2) THEN type IS press\_mark
- ☞  Rule 6: IF (max\_distance\_to\_cog IS fun 3 AND all\_extrema IS fun 2 AND max\_extrema IS fun 3) THEN type IS uneven\_surface
- ☞  Rule 7: IF (max\_distance\_to\_cog IS fun 3 AND min\_extrema IS fun 3) THEN type IS uneven\_surface

## Klassifikationsgenauigkeit:

	NBC	DTree	NN	NEFCLASS	DC
Trainingsmenge	89.0%	94.7%	90%	81.6%	46.8%
Testmenge	75.6%	75.6%	85.5%	79.9%	46.8%

# Übersicht

1. Computational Intelligence
2. Techniken der Computational Intelligence
3. Reale Beispiele
- 4. Schwarm- und populationsbasierte Optimierung**
5. Teilchenschwarmoptimierung
6. Ameisenkolonieoptimierung



# Schwarm- und populationsbasierte Optimierung

## Schwarm-Intelligenz

Bereich der KI, der intelligente Multi-Agentensysteme entwickelt  
Inspiration durch das Verhalten bestimmter Tierarten, speziell sozialer Insekten (z.B. Ameisen, Termiten, Bienen, etc.) und in Schwärmen lebender Tiere (z.B. Fische, Vögel, etc.)

Tiere dieser Arten können recht komplexe Aufgaben lösen (Finden von Nahrungsquellen, Wegesuche, Nestbau, etc.), indem sie kooperieren.

## Wesentliche Ideen

Ziemlich einfache Einzelindividuen mit begrenzten Fähigkeiten  
Koordination ohne zentrale Steuerung, lediglich Selbstorganisation  
Austausch von Informationen zwischen Individuen (Kooperation)  
Klassifizierung der Verfahren nach Art des Informationsaustausches

# Verfahren

## Genetische/Evolutionäre Algorithmen

Biologisches Vorbild: Evolution der Lebewesen

Informationsaustausch durch Rekombination der Genotypen

Jedes Individuum ist ein Lösungskandidat

# Verfahren

## Teilenschwarmoptimierung

Biologisches Vorbild: Futtersuche von Fisch- und Vogelschwärmen  
Informationsaustausch über einfache Aggregation der  
Einzellösungen  
Jedes Individuum ist ein Lösungskandidat

## Ameisenkolonialgorithmen

Biologisches Vorbild: Wegesuche zu Futterquellen durch Ameisen  
Informationsaustausch über Veränderung der Umgebung  
(Stigmergie, erweiterter Phänotyp nach Dawkins)  
Individuen konstruieren Lösungskandidaten

# Übersicht

1. Computational Intelligence
2. Techniken der Computational Intelligence
3. Reale Beispiele
4. Schwarm- und populationsbasierte Optimierung
- 5. Teilenschwarmoptimierung**  
Implementierung
6. Ameisenkolonieoptimierung

# Teilenschwarmoptimierung



© Eric T. Schulz <http://www.eeb.uconn.edu/courses/eeb296/>



© Ariel Bravy <http://www.skphoton.com/albums/>

Fische, Vögel suchen in Schwärmen nach ergiebigen Futterplätzen. Orientierung anhand individueller Suche (kognitiver Anteil) und an anderen Mitgliedern des Schwarmes in ihrer Nähe (sozialer Anteil) Außerdem: Leben im Schwarm schützt Individuen gegen Fressfeinde.

# Biologische Vorbilder: Fische



©[http://s3.amazonaws.com/xlsuite\\_production/assets/51840/fish-swarm.jpg](http://s3.amazonaws.com/xlsuite_production/assets/51840/fish-swarm.jpg)

2-3 Artgenossen geben individuellem Fisch im Schwarm Orientierung  
Anpassung von Geschwindigkeit und Richtung, sodass Winkel  
zwischen Schwimmrichtung und nächsten Artgenossen:  $45-135^\circ$   
Andere Fische werden ignoriert

Garantie, dass Schwarm ständig vorwärtsschwimmt und  
Abweichler nicht Gesamtrichtung beeinflussen

# Teilenschwarmoptimierung

Particle Swarm Optimization [Kennedy and Eberhart, 1995]

**Motivation:** Verhalten von z.B. Fischschwärmen bei der Futtersuche: zufälliges Ausschwärmen, aber stets auch Rückkehr zum Schwarm, Informationsaustausch zwischen Mitgliedern

**Ansatz:** verwende statt nur einzelnen aktuellen Lösungskandidaten „Schwarm“ von  $m$  Lösungskandidaten

**Voraussetzung:**  $\Omega \subseteq \mathbb{R}^n$  und somit zu optimierende (o.B.d.A.: zu maximierende) Funktion  $f : \mathbb{R}^n \rightarrow \mathbb{R}$

**Vorgehen:** fasse jeden Lösungskandidaten als „Teilchen“ auf, das Ort  $\mathbf{x}_i$  im Suchraum und Geschwindigkeit  $\mathbf{v}_i$  hat ( $i = 1, \dots, m$ )  
Vereinigt Elemente der bahnoorientierten Suche (z.B. Gradientenverfahren) und populationsbasierter Suche (z.B. EA)

# Teilenschwarmoptimierung

**Aktualisierung** für Ort und Geschwindigkeit des  $i$ -ten Teilchens:

$$\mathbf{v}_i(t+1) = \alpha \mathbf{v}_i(t) + \beta_1 \left( \mathbf{x}_i^{(\text{lokal})}(t) - \mathbf{x}_i(t) \right) + \beta_2 \left( \mathbf{x}^{(\text{global})}(t) - \mathbf{x}_i(t) \right)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t)$$

Parameter:  $\beta_1, \beta_2$  zufällig in jedem Schritt,  $\alpha$  mit  $t$  abnehmend

$\mathbf{x}_i^{(\text{lokal})}$  ist **lokales Gedächtnis** des Individuums (Teilchens): der beste Ort im Suchraum, den Teilchen bisher besucht hat, d.h.

$$\mathbf{x}_i^{(\text{lokal})} = \mathbf{x}_i \left( \arg \max_{u=1}^t f(\mathbf{x}_i(u)) \right)$$

$\mathbf{x}^{(\text{global})}$  ist **globales Gedächtnis** des Schwarms: der beste Ort im Suchraum, den Individuum des Schwarms bisher besucht hat (beste bisher gefundene Lösung), d.h.

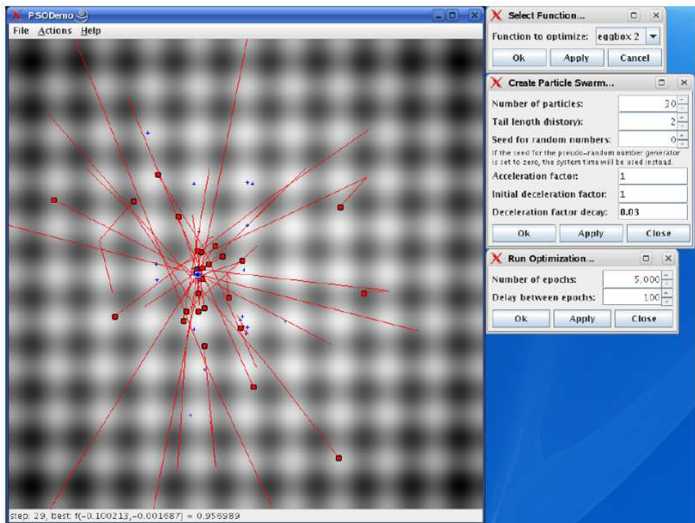
$$\mathbf{x}^{(\text{global})}(t) = \mathbf{x}_j^{(\text{lokal})}(t) \quad \text{mit} \quad j = \arg \max_{i=1}^m f \left( \mathbf{x}_i^{(\text{lokal})} \right)$$



## Algorithmus 1 Teilenschwarmoptimierung

```
1: for each Teilchen  $i$  {
2:    $\mathbf{x}_i \leftarrow$  wähle zufällig im Suchraum  $\Omega$ 
3:    $\mathbf{v}_i \leftarrow 0$ 
4: }
5: do {
6:   for each Teilchen  $i$  {
7:      $y \leftarrow f(\mathbf{x}_i)$ 
8:     if  $y \geq f(\mathbf{x}_i^{(\text{lokal})})$  {
9:        $\mathbf{x}_i^{(\text{lokal})} \leftarrow \mathbf{x}_i$ 
10:    }
11:    if  $y \geq f(\mathbf{x}_i^{(\text{global})})$  {
12:       $\mathbf{x}_i^{(\text{global})} \leftarrow \mathbf{x}_i$ 
13:    }
14:  }
15:  for each Teilchen  $i$  {
16:     $\mathbf{v}_i(t+1) \leftarrow \alpha \cdot \mathbf{v}_i(t) + \beta_1 \left( \mathbf{x}_i^{(\text{lokal})}(t) - \mathbf{x}_i(t) \right) + \beta_2 \left( \mathbf{x}_i^{(\text{global})}(t) - \mathbf{x}_i(t) \right)$ 
17:     $\mathbf{x}_i(t+1) \leftarrow \mathbf{x}_i(t) + \mathbf{v}_i(t)$ 
18:  }
19: } while Terminierungskriterium ist nicht erfüllt
```

# Teilenschwarmoptimierung



PSODemo

File Actions Help

step 29, best  $f(-0.100213, -0.001687) = 0.956989$

Select Function...

Function to optimize: eggbox 2

Ok Apply Cancel

Create Particle Swarm...

Number of particles: 30

Tail length history: 2

Seed for random numbers: 0

Acceleration factor: 1

Initial deceleration factor: 1

Deceleration factor decay: 0.03

Ok Apply Close

Run Optimization...

Number of epochs: 5,000

Delay between epochs: 100

Ok Apply Close

# Übersicht

1. Computational Intelligence
2. Techniken der Computational Intelligence
3. Reale Beispiele
4. Schwarm- und populationsbasierte Optimierung
5. Teilchenschwarmoptimierung
- 6. Ameisenkolonieoptimierung**

# Ameisenkolonieoptimierung



© PeTA <http://www.helpingwildlife.com/ants.asp>



© NickLyonMedia <http://nicklyon.orchardhostings4.co.uk>

Da gefundenes Futter zur Versorgung der Nachkommen zum Nest transportiert werden muss, bilden Ameisen Transportstraßen  
Weglängen zu Futterplätzen werden annähernd minimiert

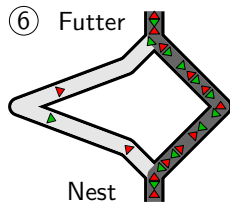
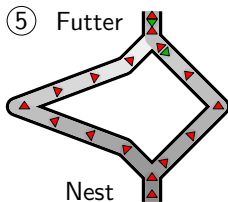
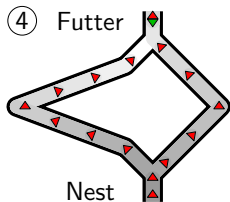
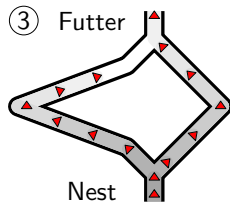
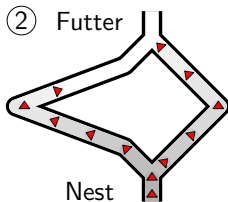
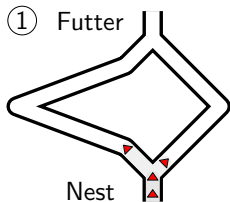
## Doppelbrückenexperiment [Goss et al., 1989]

Ameisennest und Futterquelle werden durch Doppelbrücke verbunden (beiden Zweige der Brücke sind verschieden lang)

Experiment mit argentinischer Ameise *Iridomyrmex Humilis*: diese Ameisenart ist (wie fast alle anderen auch) fast blind (Ameisen können also nicht sehen, welcher Weg kürzer ist)

Schon nach wenigen Minuten: in meisten Versuchen benutzten fast alle Ameisen kürzeren Weg

# Doppelbrückenexperiment



# Doppelbrückenexperiment [Goss et al., 1989]

## Erklärung

Auf kürzerem Weg erreichen Ameisen Futter schneller (Ende des kürzeren Weges erhält daher (am Anfang) mehr Pheromon)  
Auf Rückweg wird wegen entstandener Pheromondifferenz mit höherer W'keit kürzerer Weg gewählt (führt zu einer Verstärkung der Pheromondifferenz)

# Ameisenkolonieoptimierung

**Motivation:** Ameisen einiger Arten finden kürzeste Wege zu Futterquellen durch Legen und Verfolgen von Pheromon („Duftmarken“)

Intuitiv: kürzere Wege erhalten in gleicher Zeit mehr Pheromon  
Wege werden zufällig nach vorhandenen Pheromonmenge gewählt  
(es ist um so wahrscheinlicher, dass ein Weg gewählt wird, je mehr Pheromon sich auf Weg befindet)

Menge des ausgebrachten Pheromons kann von Qualität und Menge des gefundenen Futters abhängen

**Grundprinzip:** Stigmergie (engl. stigmergy)

Ameisen kommunizieren indirekt über Pheromonablagerungen  
Stigmergie (indirekte Kommunikation durch Veränderung der Umgebung) ermöglicht global angepasstes Verhalten aufgrund lokaler Informationen



# Natürliche und künstliche Ameisen

abstrahiere zu Suche nach bestem Weg in gewichtetem Graphen

**Problem:** Kreise, die sich selbst verstärken (durchläuft Ameise Kreis, erzeugt sie durch abgelegtes Pheromon Tendenz, Kreis erneut zu durchlaufen)

**Abhilfe:** Ablegen von Pheromon erst nach Konstruktion des ganzen Weges (Entfernen von Kreisen, bevor Pheromon abgelegt wird)

**Problem:** Ggf. konzentriert sich Suche auf am Anfang konstruierte Lösungskandidaten (vorzeitige Konvergenz)

**Abhilfe:** Pheromonverdunstung (spielt in Natur geringe Rolle)

## Nützliche Erweiterungen/Verbesserungen

Abgelegte Pheromonmenge hängt von Lösungsgüte ab  
Einbringen von Heuristiken in Kantenwahl (z.B. Kantengewicht)

# Ameisenkolonieoptimierung

**Voraussetzungen:** kombinatorisches Optimierungsproblem mit konstruktiver Methode, zur Erzeugung eines Lösungskandidaten

**Vorgehen:** Lösungen werden durch Folge von Zufallsentscheidungen konstruiert, wobei jede Entscheidung Teillösung erweitert

Entscheidungsfolge = Pfad in Entscheidungsgraphen (auch: Konstruktionsgraphen)

Ameisen sollen Pfade durch Entscheidungsgraphen erkunden und besten (kürzesten, billigsten) Weg finden

Ameisen markieren benutzte Kanten des Graphen mit Pheromon  
⇒ andere Ameisen werden zu guten Lösungen geleitet

Pheromon verdunstet in jeder Iteration, damit einmal ausgebrachtes Pheromon System nicht zu lange beeinflusst („Vergessen“ veralteter Information)

## Anwendung auf TSP

Darstellung des Problems durch  $n \times n$  Matrix  $\mathbf{D} = (d_{ij})_{1 \leq i, j \leq n}$   
 $n$  Städte mit Abständen  $d_{ij}$  zwischen Städte  $i$  und  $j$

Beachte:  $\mathbf{D}$  kann asymmetrisch sein, aber  $\forall i \in \{1, \dots, n\} : d_{ii} = 0$

Pheromoninformation als  $n \times n$  Matrix  $\Phi = (\phi_{ij})_{1 \leq i, j \leq n}$

Pheromonwert  $\phi_{ij} (i \neq j)$  gibt an, wie wünschenswert es ist, Stadt  $j$  direkt nach Stadt  $i$  zu besuchen ( $\phi_{ii}$  nicht benötigt)

$\Phi$  muss nicht notwendig symmetrisch sein/gehalten werden

Alle  $\phi_{ij}$  werden mit gleichen kleinen Wert initialisiert (anfangs liegt auf allen Kanten gleiche Menge Pheromon)

Ameisen durchlaufen (durch Pheromon) Hamiltonkreise (sie markieren Kanten des durchlaufenden Hamiltonkreises mit Pheromon, wobei ausgebrachte Pheromonmenge der Lösungsqualität entspricht)

# Lösungskonstruktion

Jede Ameise hat „Gedächtnis“  $C$ , welche Indizes der noch nicht besuchten Städte enthält

Jede besuchte Stadt wird aus Menge  $C$  entfernt

Gedächtnis gibt es im biologischen Vorbild nicht!

Ameise wird in zufällig bestimmter Stadt gesetzt (Anfang der Rundreise)

Ameise wählt noch nicht besuchte Stadt und begibt sich in diese:  
in Stadt  $i$  wählt Ameise (unbesuchte) Stadt  $j$  mit  $W$ 'keit

$$p_{ij} = \frac{\phi_{ij}}{\sum_{k \in C} \phi_{ik}}.$$

Wiederhole Schritt 2 bis alle Städte besucht

# Pheromonaktualisierung

## 1. Verdunstung/Evaporation

alle  $\phi_{ij}$  werden um Bruchteil  $\eta$  (evaporation) verringert:

$$\forall i, j \in \{1, \dots, n\} : \phi_{ij} = (1 - \eta) \cdot \phi_{ij}$$

## 2. Verstärkung konstruierter Lösungen

Kanten der konstruierten Lösungen werden mit zusätzlicher Menge an Pheromon belegt, die Lösungsqualität entspricht:

$$\forall \pi \in \Pi_t : \phi_{\pi(i)\pi((i \bmod n)+1)} = \phi_{\pi(i)\pi((i \bmod n)+1)} + Q(\pi)$$

$\Pi_t$  ist Menge der im Schritt  $t$  konstruierten Rundreisen (Permutationen), Qualitätsfunktion: z.B. inverse Reiselänge

$$Q(\pi) = c \cdot \left( \sum_{i=1}^n d_{\pi(i)\pi((i \bmod n)+1)} \right)^{-1}$$

„Je besser Lösung, desto mehr Pheromon erhalten deren Kanten.“

# Problem des Handlungsreisenden

---

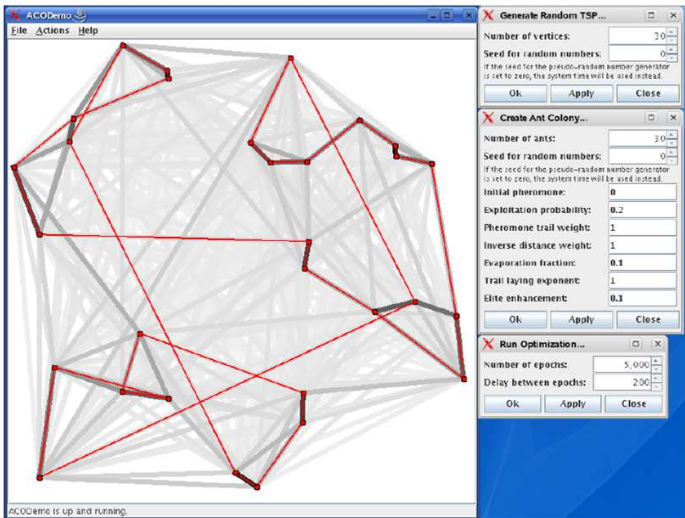
## Algorithmus 2 Ameisenkolonieoptimierung für das TSP

---

```
1: initialisiere alle Matricelemente  $\phi_{ij}$ ,  $1 \leq i, j \leq n$ , auf kleinen Wert  $\epsilon$ 
2: do {
3:   for each Ameise {                               /* konstruiere Lösungskandidaten */
4:      $C \leftarrow \{1, \dots, n\}$                    /* Menge der zu besuchenden Städte */
5:      $i \leftarrow$  wähle zufällig Anfangsstadt aus  $C$ 
6:      $C \leftarrow C \setminus \{i\}$                  /* entferne sie aus den unbesuchten Städten */
7:     while  $C \neq \emptyset$  {                       /* solange nicht alle Städte besucht wurden */
8:        $j \leftarrow$  wähle nächste Stadt der Reise aus  $C$  mit  $W$ 'keit  $p_{ij}$ 
9:        $C \leftarrow C \setminus \{j\}$                /* entferne sie aus den unbesuchten Städten */
10:       $i \leftarrow j$                                /* und gehe in die ausgewählte Stadt */
11:    }
12:  }
13:  aktualisiere Pheromon-Matrix  $\Phi$  nach Lösungsgüte
14: } while Terminierungskriterium ist nicht erfüllt
```

---

# Ameisenkolonieoptimierung



The screenshot displays the ACODemo application window, which is used for demonstrating Ant Colony Optimization (ACO) on a Traveling Salesman Problem (TSP). The main window shows a graph with 30 vertices and numerous edges. A red path highlights the current solution, while a grey path shows a previous iteration. The status bar at the bottom indicates "ACODemo is up and running."

Three dialog boxes are open on the right side of the window:

- Generate Random TSP...:**
  - Number of vertices: 30
  - Seed for random numbers: 0
  - Buttons: Ok, Apply, Close
- Create Ant Colony...:**
  - Number of ants: 30
  - Seed for random numbers: 0
  - Initial pheromone: 0
  - Exploitation probability: 0.2
  - Pheromone trail weight: 1
  - Inverse distance weight: 1
  - Evaporation fraction: 0.1
  - Trail laying exponent: 1
  - Elite enhancement: 0.1
  - Buttons: Ok, Apply, Close
- Run Optimization...:**
  - Number of epochs: 5,000
  - Delay between epochs: 200
  - Buttons: Ok, Apply, Close

# Allg. Anwendung zur Optimierung

## Grundsätzliches Prinzip

Formuliere Problem als Suche in (Entscheidungs-)Graphen, Lösungskandidaten müssen durch Kantenmengen beschreibbar sein, (beachte: es muss sich nicht notwendigerweise um Pfade handeln!)

**Allgemeine Beschreibung:** im folgenden jeweils angegeben:

- Knoten, Kanten des Entscheidungs-/Konstruktionsgraphen
- Einzuhaltende Nebenbedingungen
- Bedeutung des Pheromons auf Kanten (und evtl. Knoten)
- Nutzbare heuristische Hilfsinformation
- Konstruktion eines Lösungskandidaten

Algorithmisches Vorgehen ist i.W. analog zu Vorgehen beim TSP



## Allg. Anwendung zur Optimierung: TSP

*Knoten und Kanten des Entscheidungs-/Konstruktionsgraphen:* die zu besuchenden Städte und ihre Verbindungen, die Verbindungen sind gewichtet (Abstand, Zeit, Kosten)

*einzuhaltende Nebenbedingungen:* besuche jede Stadt genau 1x

*Bedeutung des Pheromons auf den Kanten:* wie wünschenswert ist es, Stadt  $j$  nach Stadt  $i$  zu besuchen

*nutzbare heuristische Hilfsinformation:* Abstand der Städte, bevorzuge nahe Städte

*Konstruktion eines Lösungskandidaten:* ausgehend von zufällig gewählter Stadt wird stets zu einer weiteren, noch nicht besuchten Stadt fortgeschritten

# Allg. Anwendung zur Optimierung: Zuordnungsproblem

Ordne  $n$  Aufgaben zu  $m$  Arbeiter (Personen, Maschinen): Minimierung der Summe der Zuordnungskosten  $d_{ij}$  unter Einhaltung maximaler Kapazitäten  $\rho_j$  bei gegebenen Kapazitätskosten

$$r_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$$

Jede Aufgabe und jeder Arbeiter = Knoten des Konstruktionsgraphen (Kanten tragen die Zuordnungskosten  $d_{ij}$ )

Jede Aufgaben muss genau einem Arbeiter zugeordnet werden (Kapazitäten der Arbeiter nicht überschreiten)

Pheromonwerte auf Kanten beschreiben, wie wünschenswert Zuordnung einer Aufgabe an Arbeiter ist

Inverse absolute oder relative  $r_{ij}$  oder inverse  $d_{ij}$

Wähle schrittweise Kanten (müssen keinen Pfad bilden), übergehe Kanten von bereits zugeordneten Aufgaben (bestrafe Lösungen, die Nebenbedingungen verletzen (Kostenerhöhung))

# Allg. Anwendung zur Optimierung: Rucksackproblem

Wähle aus  $n$  Objekten mit zugeordnetem Wert  $w_i$ , Gewicht  $g_i$ , Volumen  $v_i$ , etc.  $1 \leq i \leq n$ , Teilmenge maximalen Wertes aus, sodass Maximalwerte für Gewicht, Volumen, etc. eingehalten

Jedes Objekt = Knoten des Konstruktionsgraphen (Knoten tragen Objektwerte  $w_i$ , Kanten nicht benötigt)

Maximalwerte für Gewicht, Volumen, etc. müssen eingehalten werden

Pheromonwerte: nur Knoten zugeordnet (sie beschreiben, wie wünschenswert Auswahl des zugehörigen Objektes)

Verhältnis von Objektwert zu relativem Gewicht, Volumen, etc. wobei in den Verhältnissen die Maximalwerte berücksichtigt werden können

Wähle schrittweise Knoten aus, wobei in jedem Schritt sichergestellt, dass Maximalwerte eingehalten

# Konvergenz der Suche

betrachte „Standardverfahren“ mit folgenden Eigenschaften:

Verdunstung des Pheromons mit konstantem Faktor von allen Kanten

Nur auf Kanten des besten, bisher gefundenen Lösungskandidaten wird Pheromon abgelegt (strenges Eliteprinzip)

$\exists$  Untergrenze  $\phi_{\min}$  für Pheromonwerte der Kanten, welche nicht unterschritten wird

Standardverfahren konvergiert in  $W'$ keit gegen Lösung, d.h. mit  $t \rightarrow \infty$  geht  $W'$ keit, dass Lösung gefunden wird, gegen 1

Lässt man Untergrenze  $\phi_{\min}$  für Pheromonwerte „genügend langsam“ gegen 0 gehen ( $\phi_{\min} = \frac{c}{\ln(t+1)}$  mit Schrittzahl  $t$  und Konstante  $c$ ), kann man zeigen, dass für  $t \rightarrow \infty$  jede Ameise der Kolonie Lösung mit gegen 1 gehender  $W'$ keit konstruiert

# Zusammenfassung

Schwarm- und populationsbasierte Algorithmen: **Heuristiken zur Lösung von Optimierungsproblemen**

Ziel: Finden guter Näherungslösungen

**Problem der lokalen Optima:** Reduzierung durch bessere Durchforstung des Suchraums

Wichtig: **Informationsaustausch** zwischen Individuen (je nach Prinzip: verschiedene Algorithmentypen)

## **Teilchenschwarmoptimierung**

Optimierung einer Funktion mit reellen Argumenten





Informationsaustausch durch Orientierung an Nachbarn

## **Ameisenkolonieoptimierung**

Suche nach besten Wegen (abstrakt: in einem Entscheidungsgraphen)

Informationsaustausch durch Veränderung der Umgebung (Stigmergie)

# Literatur

-  Birattari, M., Di Caro, G. A., Doursat, R., Engelbrecht, A. P., Floreano, D., Gambardella, L. M., Groß, R., Sahin, E., Stützle, T., and Sayama, H. (2010).  
*Swarm Intelligence*.  
Springer.
-  Clerc, M. (2010).  
*Particle Swarm Optimization*.  
John Wiley & Sons.
-  Goss, S., Aron, S., Deneubourg, J.-L., and Pasteels, J. M. (1989).  
Self-organized shortcuts in the argentine ant.  
*Naturwissenschaften*, pages 579–581.
-  Kennedy, J. and Eberhart, R. (1995).  
Particle swarm optimization.  
In *Proc. IEEE Int'l. Conf. on Neural Networks*, pages 1942–1948, Perth,