Künstliche Intelligenz

Vorlesung 5: Schwarmintelligenz

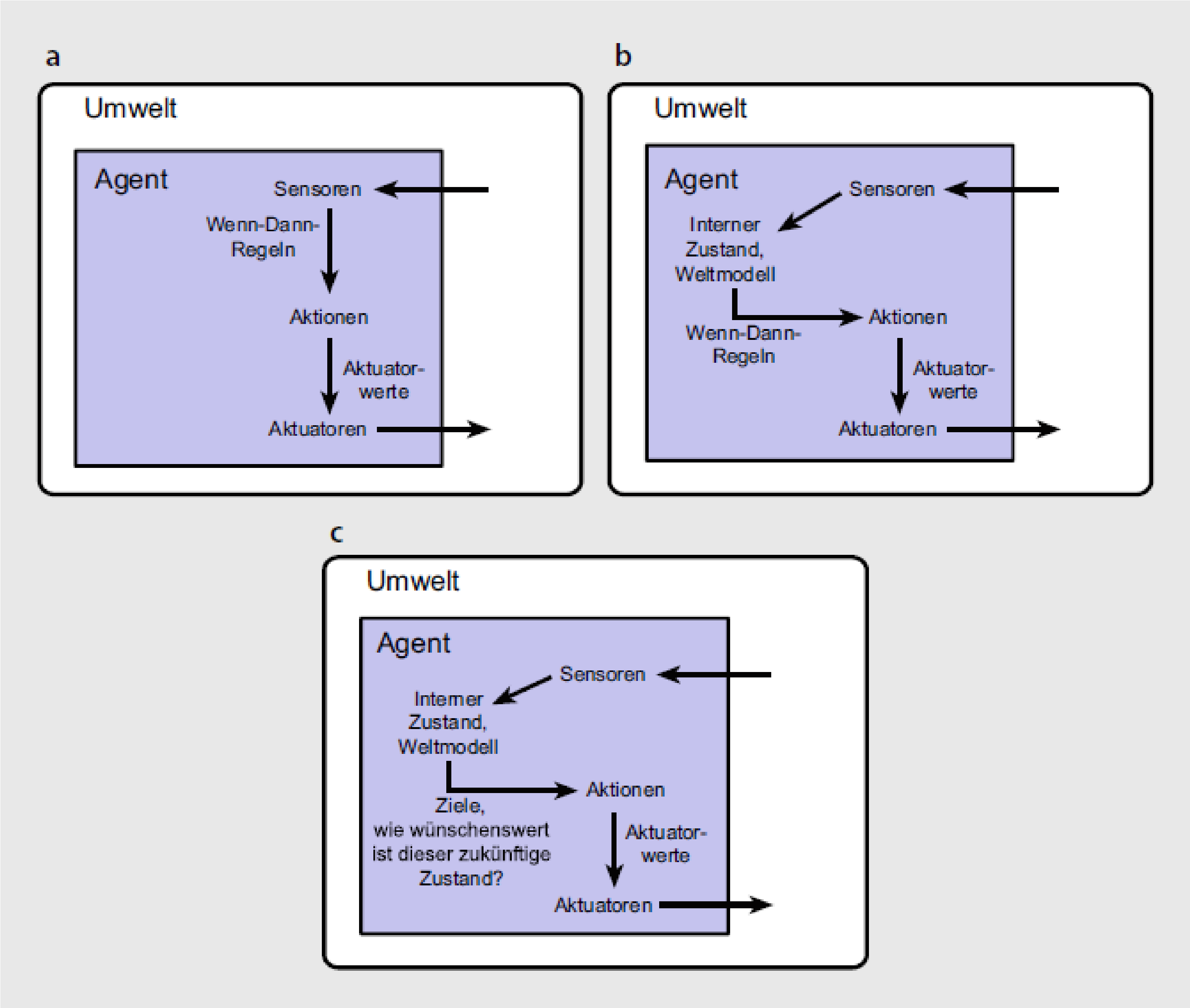
Schwarmintelligenz

- Gruppen von Individuen durch Zusammenarbeit, unabhängig von der Intelligenz der einzelnen Mitglieder, intelligente Entscheidungen treffen können.
- Basiert auf Agententechnologie
- *Verteilte Künstliche Intelligenz* (VKI). Das Arbeitsgebiet versucht, komplexe vernetzte Softwareagentensysteme nach dem Vorbild staatenbildender Insekten wie Ameisen, Bienen und Termiten, sowie teilweise auch Vogelschwärmen (Schwarmverhalten) zu modellieren.

Agenten (Wiederholung)

- Ein technischer **Agent** ist eine abgrenzbare (Hardware- oder/und Software-) Einheit mit definierten Zielen.
- Ein technischer Agent ist bestrebt, diese Ziele durch selbstständiges Verhalten zu erreichen und interagiert dabei mit seiner Umgebung und anderen Agenten.

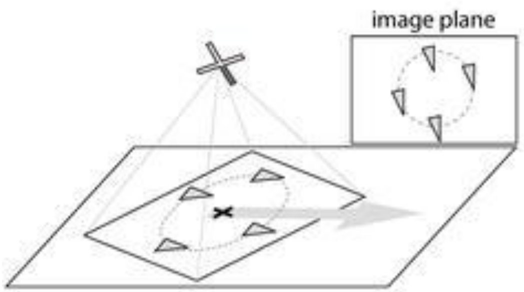
1. Reflexagent
2. Modellbasierte Reflexagenten
3. Zielbasierte Agenten


a
Umwelt
Agent
Sensoren
Wenn-Dann-Regeln
Aktionen
Aktuator-werte
Aktuatoren
b
Umwelt
Agent
Sensoren
Interner Zustand, Weltmodell
Aktionen
Wenn-Dann-Regeln
Aktuator-werte
Aktuatoren
c
Umwelt
Agent
Sensoren
Interner Zustand, Weltmodell
Aktionen
Ziele, wie wünschenswert ist dieser zukünftige Zustand?
Aktuator-werte
Aktuatoren

Multiagenten

Wie können Agenten zusammenwirken und miteinander interagieren?
- Direkte Interaktion
- Indirekte Interaktion: eines verändert die Umwelt und das andere Individuum reagiert zu einem späteren Zeitpunkt auf diese Umweltveränderung.

- Aktion und Reaktion sind zeitlich versetzt




Multiagenten und Schwarmintelligenz

- Das Zusammenwirken einiger Agenten durch eine Vielzahl an Interaktionen kann dem Schwarm als Ganzes Fähigkeiten verleihen, die über die Fähigkeiten des Individuums hinausgehen.
- Ameisen finden kürzeste Wege zwischen Futterquelle und Nest
- Kollektive Wahrnehmung: ermöglicht dem Schwarm, viele einzelne Sinneswahrnehmungen zu vereinigen.
- Schwarmroboter: Jeder Roboter führt local Messungen durch, die er dann mit seinen Nachbarn teilt, die wiederum die Daten mit ihren eigenen Messungen verbinden und weiterleiten.
- Über die Zeit verbreitet sich im Schwarm die gesammelte Information, auf die jeder einzelne Roboter dann lokal reagieren kann.

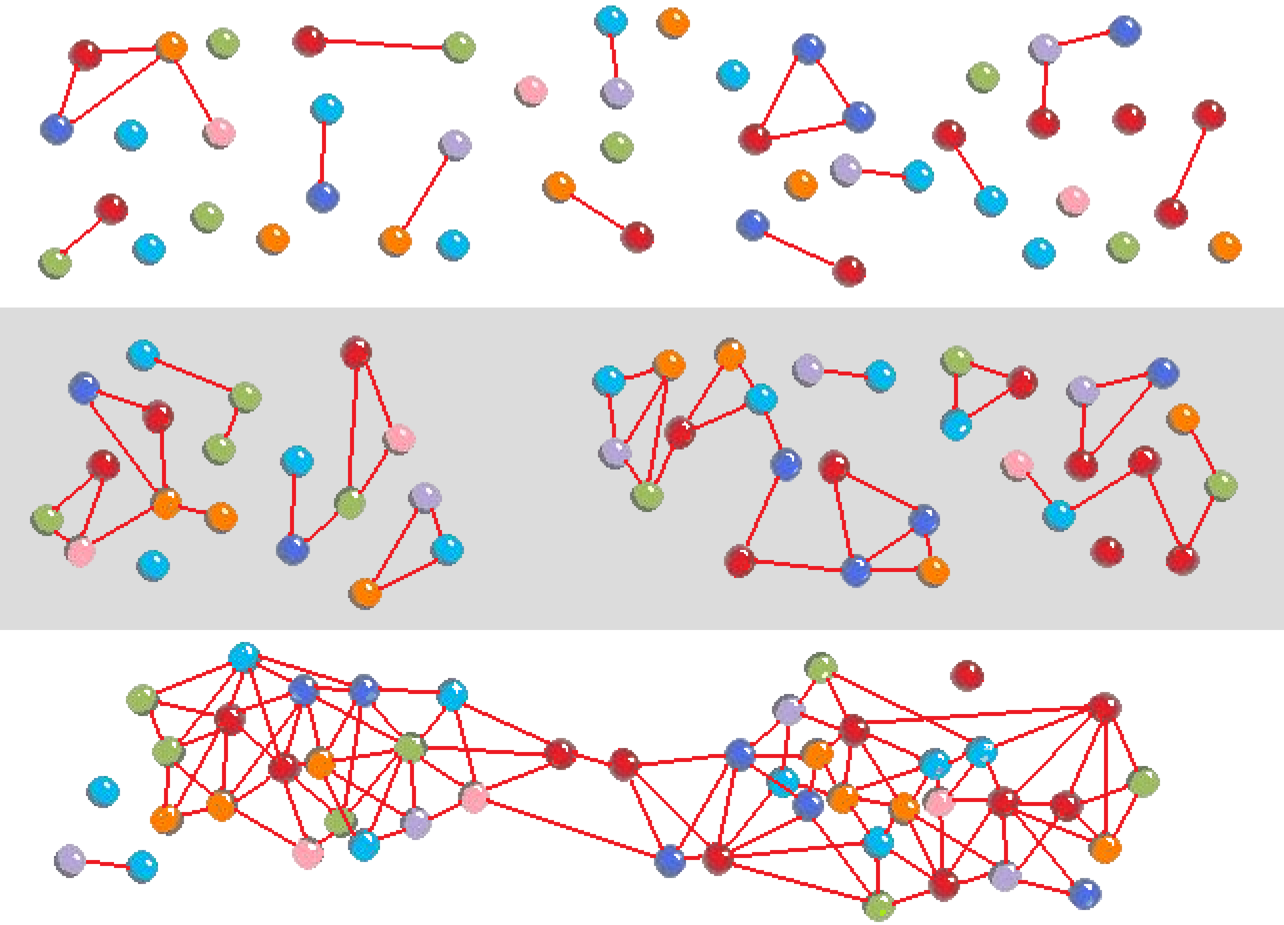
Grundkonzepte: Schwarm

- Ein **Schwarm** ist eine Aggregation von Tieren/Agenten, die sich kollektiv bewegen.
- Ein Schwarm benötigt nicht wirklich eine Mindestanzahl an Agenten, sondern definiert sich über sein Verhalten.
- Selbst wenn nur drei oder vier Agenten miteinander interagieren, so können sie doch einem Schwarmverhalten folgen und haben somit als Schwarm zu gelten.

Grundkonzepte: Selbstorganisation und Feedback

Selbstorganisation besteht aus vier Komponenten:

- positives Feedback,
- negatives Feedback,
- eine Vielzahl an Interaktionen,
- ein ausgewogenes Verhältnis zwischen der Verwertung und Erforschung von Informationen.



Grundkonzepte: Positives Feedback

Verstärkt Prozesse: Die Veränderung eines Elements A bewirkt die

Vermehrung des Elements B, was dann wiederum positiv auf Element A zurückwirkt.

Beispiel:

1. Rekrutierung bei Ameisen mittels Pheromon
2. Marktblasen/Investoren

Grundkonzepte: Negatives Fedback

- Nicht jede Fluktuation und Abweichung hat sofort extreme Auswirkungen und stabilisiert so das System.
- Ein Element A mag eine Vermehrung in einem Element B bewirken, aber Element B bewirkt dann eine Verminderung in Element A.

Grundkonzepte: Lokale Interaktion und Kommunikation

- Ein Schwarm agiert dezentral.
- Es kann daher keinen sogenannten single point of failure (einzelne Fehlerstelle) geben

➢ Hohe Robustheit

➢ Unabhängig von der Schwarmgröße entsteht immer in etwa der gleiche Aufwand für Kommunikation und Koordination.

➢ Bei lokal interagierenden Agenten besitzt auch keiner überproportional viel Informationen oder löst sehr spezialisiert eine Teilaufgabe.

➢ Kein too big to fail!

Grundkonzepte: Kommunikation über die Umwelt

- Ein Agent kann hier und jetzt eine Veränderung in der Umwelt bewirken, die später von einem anderen Agenten gefunden, erkannt und interpretiert werden kann.

Grundkonzepte: Skalierbarkeit und Schwarmleistung

- Schwärme können beliebig groß werden.
- Ursprünglich bezogen auf technische Systeme, bezeichnet die **Skalierbarkeit** die Fähigkeit, bei Vergrößerung des Systems weiterhin gut zu funktionieren. Anders gesagt: Ein skalierbares System kann wachsen und bleibt dabei leistungsfähig.
- Bei einem technischen System aus z. B. Schwarmroboten sagt man, dass es skaliert, wenn man die gleichen Steueralgorithmen für zehn, hundert oder tausend Roboter einsetzen kann, ohne dass dabei Probleme entstehen.

Szenarien: Aggregation

- Fundamentales Schwarmverhalten.
- Insbesondere für einen natürlichen Schwarm ist es essenziell zusammenzubleiben. Ansonsten würde der Schwarm drohen, sich in mehrere Gruppen aufzuteilen.
- Abstrakt: Abstand zu allen anderen minimieren, manchmal unter bestimmten Umweltfaktoren
  - Temperatur, Licht oder chemische Konzentration eines bestimmten Moleküls
- Kollektive Entscheidung zur Aggregation, dezentral und ohne konkrete Kommunikation einer globalen Position.

Szenarien: Aggregation

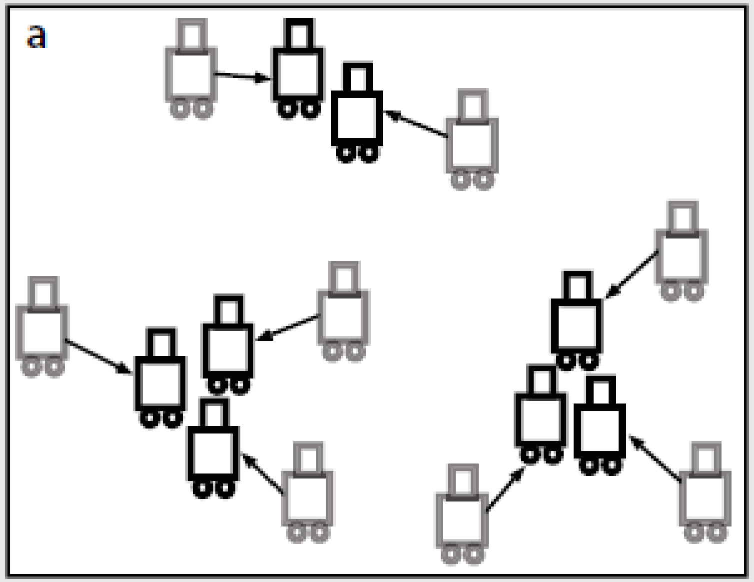
- Schwarmrobotiker muss Schwarm programmieren. Wie?
- Was muss ein einzelner Roboter tun, damit der Schwarm als Ganzes aggregiert?

Szenarien: Aggregation

- Schwarmrobotiker muss Schwarm programmieren. Wie?
- Was muss ein einzelner Roboter tun, damit der Schwarm als Ganzes aggregiert?
  - Infos kommen nur über Sensoren, die nur lokale Informationen liefern: wie weit ein Hindernis entfernt ist, eventuell auch in welcher Entfernung und Richtung sich andere Roboter befinden
  - Der Roboter hat dabei keine globale Sicht, kennt also nicht die Position aller anderen Roboter und kennt eventuell auch nicht seine Position (GPS)
- Wie kann ein Roboter, der also lediglich vage Kenntnis über die relative Position benachbarter Roboter hat, eine Schwarmaggregation provozieren?

Roboter Aggregation

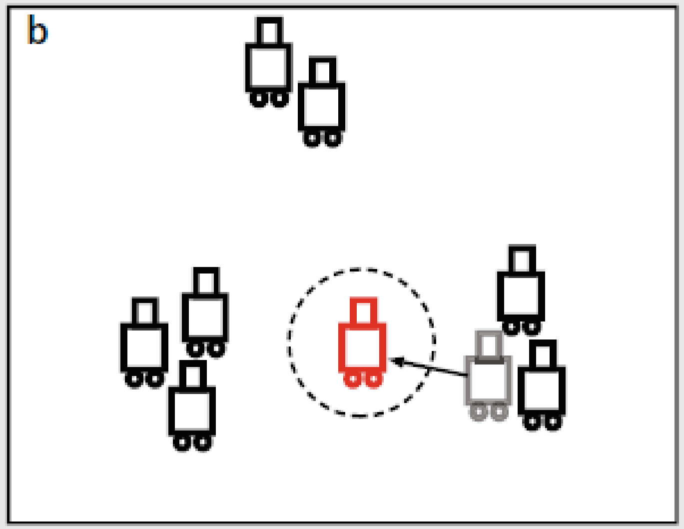
1. Idee: Distanz zu den anderen Roboter verringern -> Bildung von keinen Robotergruppen/Cluster



fahre auf deine Nachbarn zu

Roboter Aggregation

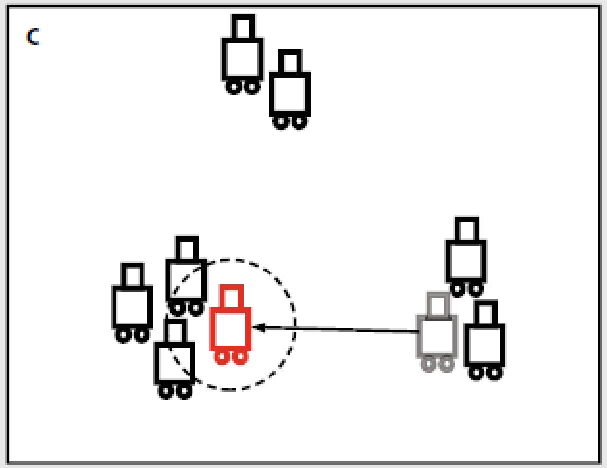
2. Idee: einen einzelnen Roboter ausschicken -> Distanz zu benachbarten Roboter wird größer und steuert durch Zufall ggf. auf eine andere Gruppe.



verlasse auch mal deine Gruppe

Roboter Aggregation

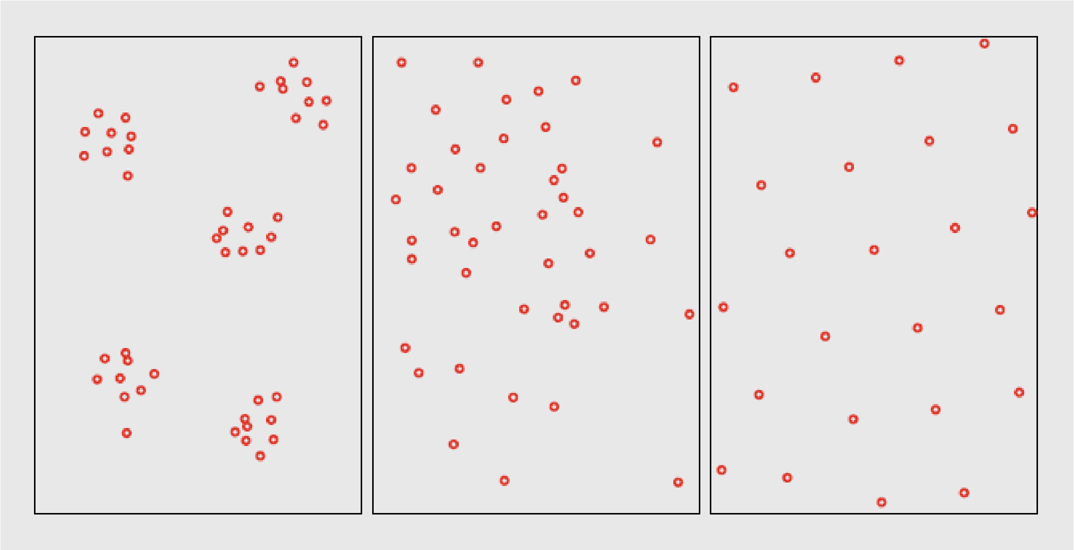
- 3. Idee: Baue positives Feedback ein! -> Roboter in großen Gruppen länger verweilen zu lassen als in kleinen Gruppen.
- Somit werden bereits große Gruppen eher wachsen, während kleinere Gruppen eher schrumpfen werden und sich hoffentlich sogar auflösen -> selbstorganisierter Prozess, in dem keine Gruppe sich bewusst auflöst, aber trotzdem das Verhalten eines jeden Roboters dazu beiträgt, die Gruppen im Durchschnitt zu vergrößern.



schließe dich neuer Gruppe an

Szenarien: Dispersion

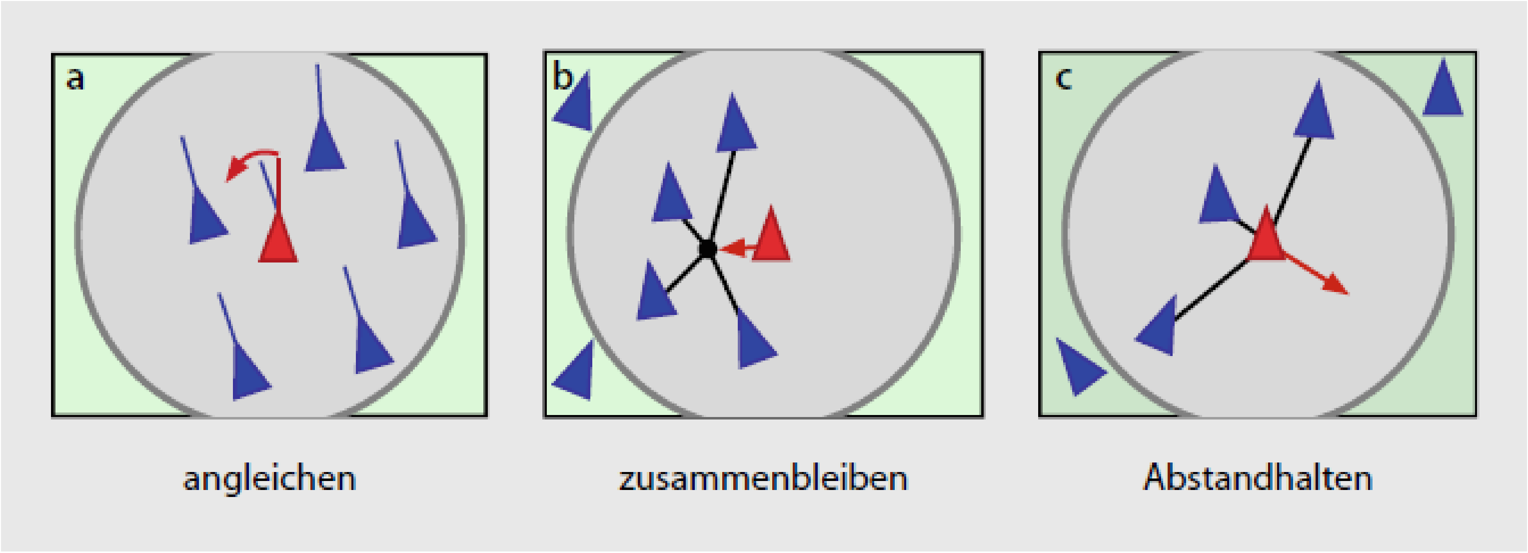
- Der Schwarm soll sich über einen möglichst großen Bereich verteilen.
- Bei Roboterschwärmen kann Dispersion sinnvoll eingesetzt werden, um einen Bereich zu überwachen oder zu erforschen.



In der Ökologie sind verschiedene Populationsverteilungen bekannt: gruppiert, zufällig und regelmäßig

Szenarien: Flocking

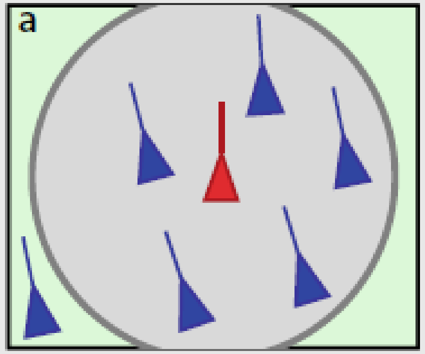
Szenarien: Flocking



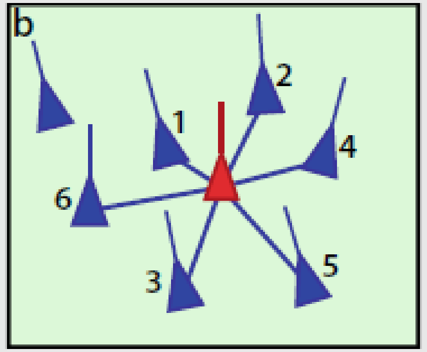
Die drei Regeln für Flocking nach Reynolds (1987): **a** angleichen (alignment), **b** zusammenbleiben (cohesion) und **c** Abstand halten (separation)

Szenarien: Flocking

- Metrischer Ansatz vs. Topologischer Ansatz
- nur die sechs nächsten Nachbarn sind relevant, egal wie weit entfernt (oder wie nah) diese sind.



Metrische Nachbarschaft: Alle Agenten innerhalb einer definierten Distanz sind Nachbarn (hier alle, die auf dem Kreis liegen).



Topologische Nachbarschaft: Die Nachbarschaft hat eine feste Größe, hier sechs; unabhängig von einer maximalen Distanz sind die nächsten sechs Agenten Nachbarn.

Memetische Algorithmen

- Populationsbasierte Algorithmen und lokale Suche zeichnen sich durch unterschiedliche Vor- und Nachteile aus:
  - Während der populationsbasierte Ansatz langsam in der Breite den Suchraum durchforscht, geht die lokale Suche schnell in die Tiefe und steuert das nächste lokale Optimum an.
- Memetische Algorithmen verbinden beide Ansätze.
- Ihr Name geht auf den Begriff Meme des Biologen Richard Dawkins zurück, der damit Verhaltenselemente bezeichnet, die sich im Gegensatz zu Genen individuell ändern können, indem sie beispielsweise durch Nachahmung erworben werden.

Memetische Algorithmen

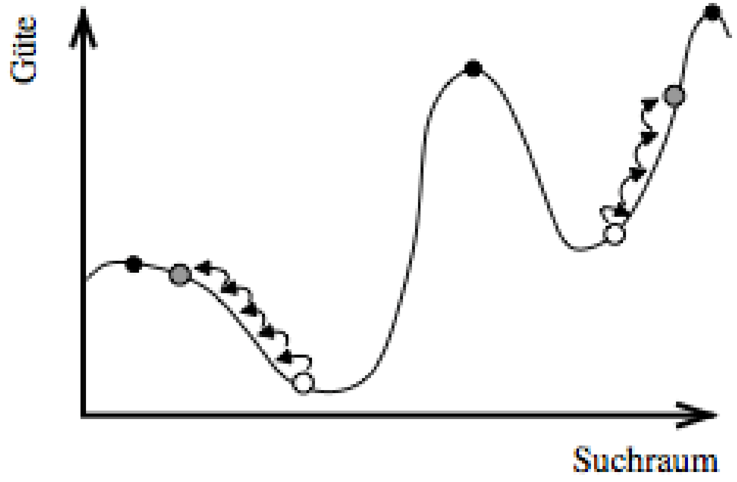
- Die Grundidee nahezu aller memetischer Algorithmen ist, alle durch einen evolutionären Algorithmus erzeugten Individuen zunächst lokal zu optimieren und sie dann erst in die Population aufzunehmen.

MEMETISCHER-ALGORITHMUS(Bewertungsfunktion F)

$t \leftarrow 0$
$P(t) \leftarrow$ initialisiere Population der Größe μ
$P(t) \leftarrow$ LOKALE-SUCHE(F) für jedes Individuum in $P(t)$
bewerte $P(t)$ durch F
while Terminierungsbedingung nicht erfüllt
do $E \leftarrow$ selektiere Eltern für λ Nachkommen aus $P(t)$
    $P' \leftarrow$ erzeuge Nachkommen durch Rekombination aus E
    $P'' \leftarrow$ mutiere die Individuen in P'
    $P''' \leftarrow$ LOKALE-SUCHE(F) für jedes Individuum in P''
    bewerte P''' durch F
    $t \leftarrow t+1$
    $P(t) \leftarrow$ Umweltselektion auf P'''
return bestes Individuum aus $P(t)$

Memetische Algorithmen

- Memetische Algorithmen schränken die Bereiche des Suchraums ein, in denen sich Lösungskandidaten befinden können.
- Im Extremfall entspricht tatsächlich jeder Lösungskandidat einem lokalen Optimum.
- Beispielhaft wird für die Gütelandschaft eines Maximierungsproblems gezeigt, wie sich die neu erzeugten Individuen (weiße Punkte) durch lokale Suche den lokalen Optima (schwarze Punkte) annähern. Im Extremfall wird solange lokal optimiert, bis die lokalen Optima erreicht sind.



Populationsbasiertes inkrementelles Lernen

- PBIL, engl. population based incremental learning
- In genetischen Algorithmen mit binärer Kodierung $\mathcal{G} = \mathbb{B}^l$ wird die Population nicht mehr explizit gespeichert, sondern nur noch eine Populationsstatistik der Genfrequenz geführt.
- Für jedes der l Bits wird protokolliert, wie häufig der Wert 1 in den Individuen der Population vorhanden ist.
- Selbstverständlich wird hierbei die relative Häufigkeit betrachtet.

Populationsbasiertes inkrementelles Lernen

- Für jedes Gen wird der Mittelwert der Allele aller vier Individuen in der Beispielpopulation berechnet.

1	0	1	1	Individuum 1
0	0	0	1	Individuum 2
1	1	1	1	Individuum 3
1	0	0	1	Individuum 4
0,75	0,25	0,5	1,0	Populationsstatistik

(a) Elternpopulation

Individuum				Wahrscheinlichkeit
1	0	1	1	$0{,}75 \cdot 0{,}75 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}281$
1	1	0	1	$0{,}75 \cdot 0{,}25 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}094$
0	0	1	1	$0{,}25 \cdot 0{,}75 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}094$
0	1	1	1	$0{,}25 \cdot 0{,}25 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}031$
1	0	0	0	$0{,}75 \cdot 0{,}75 \cdot 0{,}5 \cdot 0{,}0 \approx 0{,}0$

(b) Kindindividuen

Abbildung 3 : Aufbau und Verwendung einer Populationsstatistik: (a) Berechnung der Statistik aus vier Individuen und (b) die Erzeugung von fünf neuen Individuen aus der Populationsstatistik.

Populationsbasiertes inkrementelles Lernen

- Eine Populationsstatistik allein reicht jedoch nicht aus, um ein Optimierungsproblem zu lösen - dafür müssen konkrete Individuen bewertet werden.
- Die statistischen Werte werden als Wahrscheinlichkeiten aufgefasst, entsprechend derer neue Individuen aus der virtuellen Population gezogen werden.

Populationsbasiertes inkrementelles Lernen

1	0	1	1	Individuum 1
0	0	0	1	Individuum 2
1	1	1	1	Individuum 3
1	0	0	1	Individuum 4
0,75	0,25	0,5	1,0	Populationsstatistik

(a) Elternpopulation

Individuum				Wahrscheinlichkeit
1	0	1	1	$0{,}75 \cdot 0{,}75 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}281$
1	1	0	1	$0{,}75 \cdot 0{,}25 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}094$
0	0	1	1	$0{,}25 \cdot 0{,}75 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}094$
0	1	1	1	$0{,}25 \cdot 0{,}25 \cdot 0{,}5 \cdot 1{,}0 \approx 0{,}031$
1	0	0	0	$0{,}75 \cdot 0{,}75 \cdot 0{,}5 \cdot 0{,}0 \approx 0{,}0$

(b) Kindindividuen

DieTabelle (b) demonstriert für verschiedene Kindindividuen, mit welcher Wahrscheinlichkeit sie erzeugt werden, wenn die Populationsstatistik aus Tabelle (a) zugrunde gelegt wird. Das Individuum 1000 kann dabei gar nicht mehr erzeugt werden, weil das letzte Bit mit Wahrscheinlichkeit 1,0 den Wert 1 annimmt.

Populationsbasiertes inkrementelles Lernen

- Da die einzelnen Bits völlig unabhängig voneinander erzeugt werden, entspricht diese Erzeugung eines neuen Individuums bereits der Rekombination UNIFORMER-CROSSOVER als globale Variante, sodass hier kein zusätzlicher Rekombinationsoperator mehr angewandt wird.
- Als Selektionsmechanismus wird per BESTEN-SELEKTION das beste erzeugte Individuum ausgewählt und zur Aktualisierung der Populationsstatistik herangezogen
- Eine Mutation wird nicht direkt auf den erzeugten Individuen durchgeführt, sondern es wird stattdessen die Statistik für einige Bits zufällig leicht verschoben.

PBIL ALGORITHMUS

PBIL(Bewertungsfunktion F)

$t \leftarrow 0$
bestInd $\leftarrow$ erzeuge ein zufälliges Individuum aus $\mathcal{G} = \mathbb{B}^{\ell}$
bewerte *bestInd* durch F
$Prob^{(0)} \leftarrow (0.5, \ldots, 0.5) \quad \in [0,\ 1]^{\ell}$
while Terminierungsbedingung nicht erfüllt
do $P \leftarrow \langle\rangle$
    for $i \leftarrow 1, \ldots, \lambda$
    do $A \leftarrow$ erzeuge Individuum aus $\mathbb{B}^{\ell}$ gemäß $Prob^{(t)}$
        $P \leftarrow P \circ \langle A \rangle$
    bewerte P durch F
    $\langle B \rangle \leftarrow$ Selektion aus P mittels BESTEN-SELEKTION
    if $B.F \succ bestInd.F$
    then $bestInd \leftarrow B$
    $t \leftarrow t + 1$
    for each $k \in \{1, \ldots, \ell\}$
    do $Prob_k^{(t)} \leftarrow B_k \cdot \alpha$ (Lernrate) $+ Prob_k^{(t-1)} \cdot (1 - \alpha)$
    for each $k \in \{1, \ldots, l\}$
    do $u \leftarrow$ wähle Zufallszahl gemäß $U((0,\ 1])$
        if $u \leq p_m$ (Mutationswahrscheinlichkeit)
        then $u' \leftarrow$ wähle Zufallszahl gemäß $U(\{0,\ 1\})$
            $Prob_k^{(t)} \leftarrow u' \cdot \beta$ (Mutationskonstante) $+ Prob_k^{(t)} \cdot (1 - \beta)$
return *bestInd*

Populationsbasiertes inkrementelles Lernen

- Im Gegensatz zu den genetischen Algorithmen können beim populationsbasierten inkrementellen Lernen keine internen Abhängigkeiten zwischen den einzelnen Bits erlernt werden.
- Im Algorithmus bestimmt die Lernrate α den Grad, mit welchem Erforschung und Feinabstimmung betrieben werden.
- Ein niedriger Wert betont mehr die Erforschung, während bei einem hohen Wert die Suche sich sehr schnell fokussiert.

Scatter Search

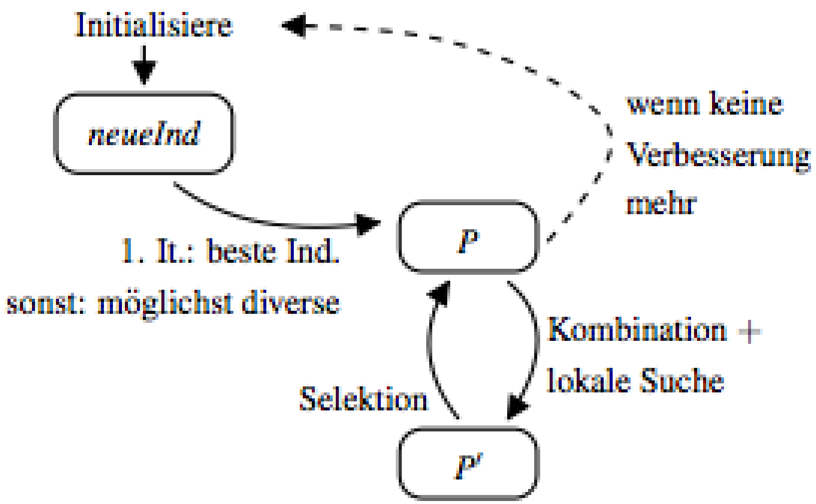
- Scatter Search wurde als deterministisches Optimierungsverfahren konzipiert.
- Es weist es viele Ähnlichkeiten zu den evolutionären Algorithmen auf:
- Es arbeitet auf Populationen, benutzt Variationsoperatoren und erzeugt einen Selektionsdruck für die neu erzeugten Individuen.
- Eine breite Initialisierung und eine umfassende systematische Erzeugung neuer Individuen garantieren eine weiträumige Erforschung des Suchraums.
- Die Feinabstimmung wird wie bei den memetischen Algorithmen durch eine lokale Suche für jedes Individuum erreicht. Der SCATTER-SEARCH Algorithmus ist für reellwertige Problemräume mit $\mathcal{G} = \Omega = \mathbb{R}^n$ gedacht.

Scatter Search

```
SCATTER-SEARCH( Bewertungsfunktion F )
P = ⟨⟩
neueInd ← ⟨⟩
for t ← 1, . . . , maxIter
do ⌈ while #neueInd < μ
     do ⌈ A ← erzeuge ein Individuum mit einem Diversitätsgenerator
          A ← LOKALE-SUCHE(F) angewandt auf A
          bewerte A durch F
          if A ∉ neueInd ∘ P
        ⌊ then [ neueInd ← neueInd ∘ ⟨A⟩
     if t = 1
     then ⌈ P ← selektiere α Individuen aus neueInd mit BESTEN-SELEKTION
          ⌊ neueInd ← streiche Individuen aus P in neueInd
     for k ← 1, . . . , β
     do ⌈ A ← dasjenige Individuum aus neueInd, das min_{B∈P} d(A.G, B.G) maximiert
          neueInd ← streiche Individuum A in neueInd
        ⌊ P ← P ∘ ⟨A⟩
     repeat ⌈ P′ ← ⟨⟩
              Mengen ← erzeuge Teilmengen von P durch einen Teilmengengenerator
              for each M ∈ Mengen
              do ⌈ A ← wende einen Kombinationsoperator auf M an
                   A ← LOKALE-SUCHE(F) angewandt auf A
                   bewerte A durch F
                   if A ∉ P ∪ P′
                 ⌊ then [ P′ ← P′ ∘ ⟨A⟩
            ⌊ P ← selektiere α + β Ind. aus P ∘ P′ mit BESTEN-SELEKTION
     until P hat sich nicht geändert
   ⌊ P ← selektiere α Individuen aus P mit BESTEN-SELEKTION
return bestes Individuum aus P
```

Scatter Search

- Dabei wird die Population P immer wieder mit neuen, möglichst andersartigen Individuen erweitert und anschließend geprüft, ob sich durch die Rekombination mit den neuen Individuen neue bessere lokale Optima finden lassen.



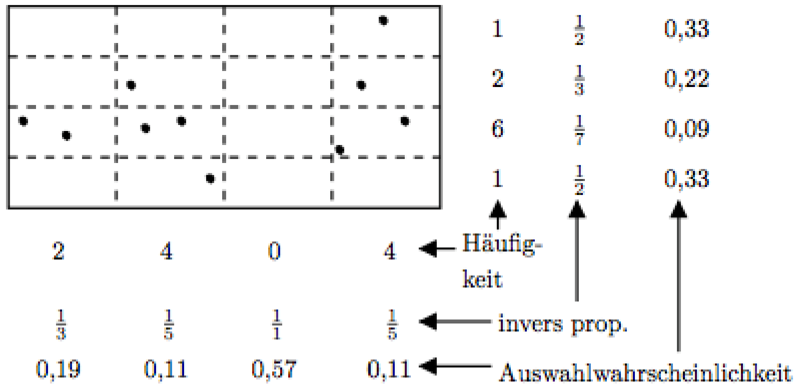
Scatter Search

- Für die Erzeugung neuer Individuen kommt ein Diversitätsgenerator zum Einsatz, der Individuen in möglichst noch unberücksichtigten Regionen des Suchraums erzeugt.
- Diese werden anschließend lokal optimiert.

BEISPIEL: DIVERSITÄTSGENERATOR FÜR REELLWERTIGE PROBLEME

- Für jede Suchraumdimension wird der gültige Wertebereich in vier Teile zerlegt und für jeden Teil wird gespeichert, wieviele Individuen x in diesem Teil bereits erzeugt wurden.
- Dann wird invers proportional zur Häufigkeit $1/(1+x)$ für jede Suchraumdimension der Wertebereich und ein zufälliger Wert aus diesem Bereich gewählt.

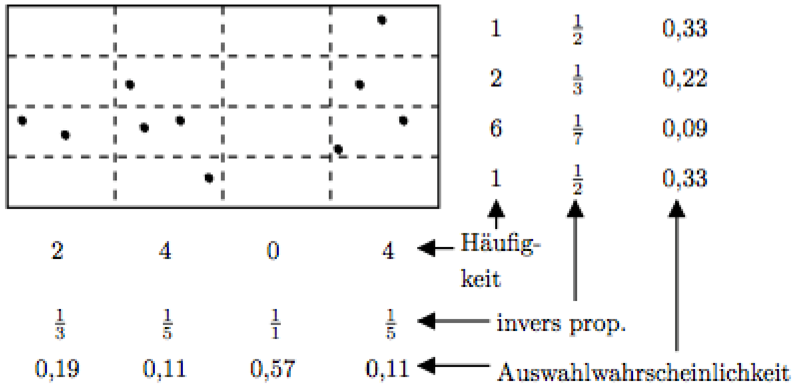
BEISPIEL: DIVERSITÄTSGENERATOR FÜR REELLWERTIGE PROBLEME



Verteilung von Individuen in einem zweidimensionalen Suchraum: Häufigkeiten der Individuen auf den einzelnen Achsenabschnitten sowie die Ableitung der Auswahlwahrscheinlichkeit daraus.

Im ersten X-Achsenabschnitt sind x = 2 Individuen enthalten, was der inversen Häufigkeit 1/(x + 3) = 1/3 entspricht. Da die Summe der inversen Häufigkeiten etwa 1,73 ist, resultiert die Auswahlwahrscheinlichkeit als 1/3 x 1/1, 73 = 0, 19.

BEISPIEL: DIVERSITÄTSGENERATOR FÜR REELLWERTIGE PROBLEME



Die Wahrscheinlichkeit, dass ein Individuum im linken oberen Sektor erzeugt wird, ergibt sich damit als 0,19 x 0,33 = 0,062.
Der Sektor zwei Felder darunter hat die Wahrscheinlichkeit 0,19 x 0,09 =0,017. Die größte Wahrscheinlichkeit hat u. a. das dritte Feld in der oberen Reihe mit 0,57 x 0,33 = 0,188.

Scatter Search

- Die Kombination der Individuen ist nicht zufällig wie bei den evolutionären Algorithmen.
- Über einen Teilmengengenerator werden systematisch Rekombinationen durchgeführt.
- Aus den zusammengestellten Individuen erzeugt der Kombinationsoperator jeweils ein neues Individuum.
- Dieses wird wieder lokal optimiert und in die Population der Besten übernommen, falls es noch nicht bekannt ist.
- Diese Kombinationsphase wird solange wiederholt, bis sich die Menge der besten Individuen nicht mehr ändert.

Schwarmoptimierung: Ameisenkolonien

- Der Vorgang der Evolution ist nicht die einzige Inspirationsquelle aus der Natur für die Lösung von Optimierungsaufgaben.
- Auch Insektenkolonien sind ein interessanter Betrachtungsgegenstand, da sie ohne eine zentrale Steuerung mit relativ einfacher Basiskommunikation sehr komplexe Aufgabenstellungen bewältigen.
- Als ein Beispiel wird hierfür die Futtersuche von Ameisen betrachtet.
- Dabei wurde in Experimenten festgestellt, dass die Ameisen über einen Duftstoff, das sog. Pheromon, ihre Wege markieren und sich mit größerer Wahrscheinlichkeit an solchen Wegen orientieren, auf denen sich mehr Duftstoff befindet.

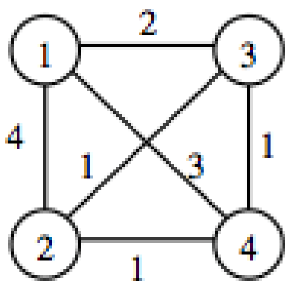
Schwarmoptimierung: Ameisenkolonien

- Dieses Verhalten wird zur Lösung von solchen Problemen imitiert, bei denen die Lösung als ein Weg in einem Graphen dargestellt werden kann.
- Ein Beispiel hierfür ist das Handlungsreisenden Problem
- Beim evolutionären Ansatz wurde durch Veränderung der Permutation in den Individuen immer eine komplette Rundreise betrachtet und variiert.
- Im Gegensatz dazu wird bei der Ameisenkolonieoptimierung durch μ virtuelle Ameisen immer wieder eine neue Rundreise schrittweise konstruiert.
- Dabei hat jede Ameise nur ein sehr beschränktes lokales Wissen über das Problem.

Schwarmoptimierung: Ameisenkolonien

- Sie nutzt einerseits ein Erinnerungsvermögen, welche Knoten sie bereits besucht hat, um im Beispiel des Handlungsreisendenproblems nicht zu früheren Knoten auf dem Rundweg zurückzuspringen.
- Andererseits benutzt sie das Pheromon, das von anderen Ameisen auf den Kanten platziert wurde, um häufig benutzte Kanten mit einer größeren Wahrscheinlichkeit auszuwählen.
- Hat eine Ameise einen vollständigen Lösungskandidaten erstellt, wird der Lösungskandidat bewertet und aufgrund seiner Güte eine bestimmte Menge Pheromon auf den Kanten verteilt.
- Das Pheromon wird zur Zeit t in einer Matrix $PM^{(t)}$ gespeichert, bestehend aus den Werten $(\tau_{i,j})_{1 \leq i,j \leq n}$.

Schwarmoptimierung: Ameisenkolonien/Beispiel



(a) Handlungsreisenden-problem

	1	2	3	4
1	0	1	6	3
2	5	0	2	3
3	3	3	0	4
4	2	4	4	0

(b) Pheromonverteilung

Beispiel für eine Pheromonverteilung auf einem Handlungsreisendenproblem mit 4 Knoten.

Der hohe Pheromonwert z. B. auf der Kante von 1 nach 3 zeigt an, dass diese Kante besonders oft von den Ameisen benutzt wurde.

Schwarmoptimierung: Ameisenkolonien

- Konkret bestimmt sich die Wahrscheinlichkeit, dass von dem aktuellen Knoten v_i der Knoten $v_j \in \mathit{verfuegbar}$ aus der Menge der noch nicht besuchten Knoten gewählt wird, aus zwei Faktoren:
  - der Pheromonmenge $\tau_{i,j}$, das auf der Kante liegt, - je mehr Pheromon desto höher ist die Wahrscheinlichkeit - und
  - der inversen Entfernung $\mathit{nah}_{i,j} = \frac{1}{\gamma(v_i, v_j)}$ zwischen den Knoten, wobei γ das Gewicht der Kante im Graphen darstellt.
  - $\mathit{nah}_{i,j}$ gibt an wie attraktiv die Strecke ij für eine Ameise ist, wenn bisher noch keine Pheromone ausgeschüttet worden wären.

Schwarmoptimierung: Ameisenkolonien

- Dann ist die Auswahlwahrscheinlichkeit

$$Pr[v_j \mid v_i] = \begin{cases} \frac{\tau_{i,j} \cdot (nah_{i,j})^{\beta}}{\sum_{v_k \in verfuegbar} \tau_{i,k} \cdot (nah_{i,k})^{\beta}} & \text{falls } v_j \in verfuegbar \\ 0 & \text{sonst.} \end{cases}$$

- β ist ein Wert mit dem man den Einfluss der Pheromonkonzentration festlegen kann.

Beispiel

- Wird eine neue Ameise mit dem Startknoten 1 auf den Weg geschickt, ergeben sich für die nächste Stadt die folgenden Wahrscheinlichkeiten (mit der Summe $s = 1/16 + 3/2 + 1/3 = 91/48$, $\beta = 2$):
  - von 1 nach 2: $1 \cdot (1/4)^2 \cdot 1/s = 1/16 \cdot 1/s \approx 0,033$
  - von 1 nach 3: $6 \cdot (1/2)^2 \cdot 1/s = 3/2 \cdot 1/s \approx 0,79$
  - von 1 nach 4: $3 \cdot (1/3)^2 \cdot 1/s = 1/3 \cdot 1/s \approx 0,17$.

Beispiel

- Wenn wir weiter davon ausgehen, dass die Kante (1, 3) gewählt wird, dann sind noch die Knoten 2 und 4 verfügbar. Dann ergeben sich dort (mit $s = 7$) die Wahrscheinlichkeiten:
  - von 3 nach 2: $3 \cdot (1/2)^2 \cdot 1/2 = 3 \cdot 1/s \approx 0,43$
  - von 3 nach 4: $\cdot(1/1)^2 \cdot 1/s = 4 \cdot 1/s \approx 0,57.$
- Folgt die Ameise auch hier der größeren Wahrscheinlichkeit, folgt die Rundreise $(1, 3, 4, 2)$ mit der Länge 7.

Schwarmoptimierung: Ameisenkolonien

- Durch einen Explorationsregler Θ wird im Algorithmus AMEISENKOLONIE-TSP bestimmt, wie häufig die nächste Stadt gemäß dieser Auswahlwahrscheinlichkeit bestimmt werden soll oder ob einfach die Stadt mit der größten Wahrscheinlichkeit genommen wird.
- Ein kleiner Wert Θ kann stabilere Ergebnisse produzieren.

Schwarmoptimierung: Ameisenkolonien

- Sind alle Ameisen die Städte abgelaufen, wird die Pheromonmatrix *PM* durch die Länge ihrer Reise modifiziert.
- Dabei verdunstet ein Teil α des Pheromons und auf den benutzten Kanten wird die Pheromonmenge gemäß der inverse Länge der konstruierten Rundreise erhöht, wodurch erreicht wird, dass kürzere Rundreisen den Pheromonwert auf ihren Kanten stärker erhöhen als lange Rundreisen.

$$\tau_{i,j} \leftarrow \alpha \cdot \tau_{i,j} + \sum_{k=1}^{\mu} wert(A^{(k)}, i, j)$$

mit

$$wert(A, i, j) = \begin{cases} \frac{1}{A.F} & \text{falls } (i, j) \text{ in } A \text{ enthalten ist} \\ 0 & \text{sonst.} \end{cases}$$

Beispiel

	1	2	3	4
1	0	0,7	4,2	2,1
2	3,5	0	1,4	2,1
3	2,1	2,1	0	2,8
4	1,4	2,8	2,8	0

(a) Verdunstung

	1	2	3	4
1	0	0,7	4,34	2,1
2	3,5	0	1,4	2,24
3	2,1	2,24	0	2,8
4	1,54	2,8	2,8	0

(b) Wegmodifikation

Modifikation der Pheromontabelle: (a) durch die Verdunstung und (b) mit dem Faktor 1/7 entlang der gewählten Rundreise.

Wird die Pheromontabelle mit der Rundreise (1, 3, 4, 2) modifiziert, werden alle Pheromonwerte auf den Anteil $\tau = 0{,}7$ reduziert, wie es in (a) dargestellt ist. Die Kanten der Rundreise werden um den Wert 1/7 erhöht, weil die Rundreise die Länge 7 hat. Das Ergebnis ist in (b) dargestellt.

Schwarmoptimierung: Ameisenkolonien

```
AMEISENKOLONIE-TSP( Bewertungsfunktion F (TSP mit n Städten) )
t ← 0
PM^(t) ← initialisiere Pheromon
while Terminierungsbedingung nicht erfüllt
do ⌈ for i ← 1, ..., μ (Anzahl der Ameisen)
     do ⌈ A^(i).G ← ⟨1⟩ (initialisiere neue Ameise)
          aktuell ← 1
          for k ← 2, ..., n
          do ⌈ u ← wähle Zufallszahl gemäß U([0, 1))
               if u < θ (Regler für Exploration)
               then [ nächster ← wähle Knoten gemäß Pr[v_j|aktuell]
               else [ nächster ← Knoten j mit maximalem Pr[v_j|aktuell]
               A^(i).G ← A^(i).G ∘ ⟨nächster⟩
             ⌊ aktuell ← nächster
        ⌊ bewerte A^(i) durch F
     t ← t + 1
   ⌊ PM^(t) ← aktualisiere PM^(t−1) gemäß Gleichung 4.2
return beste gefundene Rundreise
```

Schwarmoptimierung: Ameisenkolonien

Anwendungen:

- Busrouten, Müllabfuhr, Post- und Auslieferungsrouten.
- Maschinenbelegungsproblem: Minimierung der Transportzeit bei räumlich weit auseinander liegenden Produktionsstätten: Realisiert bei Unilever in England und Vincent Darley von der Bios-Gruppe in Santa Fe (New Mexico).
- Routenoptimierung zur Nachschubversorgung von Fertigungslinien mit minimalem Transportmitteleinsatz (Routenbildung, -führung, -taktung in Verbindung mit der Behälterauswahl); realisiert im layoutbasierten Logistikplanungssystem MALAGA mit Algorithmen von INPRO.
- Beschickung von Lackieranlagen: Losbildung zur Minimierung der Farbwechsel: Realisiert bei DaimlerChrysler.

Schwarmoptimierung: Ameisenkolonien

- Fertigungssteuerung: Losbildung zur Minimierung der Rüstzeiten und der Einhaltung von Endterminen bei ebm-papst, St. Georgen; realisiert durch ein Programm von Carpe Retem.
- Proteinfaltung: 20 Aminosäuren werden zu Proteinen mit 100 Aminosäuren kombiniert → 20^{100}, dies ergibt etwa 10^{130} verschiedene Proteine.
- Telefonnetzwerk und Internet: Durch ACO können schnell freie Strecken gefunden werden und zwar während des Betriebs (z. B. Antnet).
- Personaleinsatzplanung bei Fluggesellschaften: Flugbegleiter und Piloten werden unter Berücksichtigung von Ruhephasen etc. monatlich geplant.
- Staplerleitsysteme: Optimale Steuerung und Auslastung von Fahrzeugen und Fahrwegen.

Partikelschwärme

- Partikelschwärme (engl. particle swarms) sind eine Optimierungsmethode für reellwertige Optimierungsprobleme, die auf der Modellierung sozialer Interaktionen beruht.
- Zunächst waren die Partikelschwärme reine Simulationsmodelle für Sozialverhalten.
- Daher unterscheiden sie sich von evolutionären Algorithmen in erster Linie darin, dass sie Verbesserungen nicht durch einen Selektionsmechanismus erreichen sondern durch Nachahmung und Lernen von anderen benachbarten Individuen.
- Damit wird das Schwarmverhalten von Vögeln oder Fischen hinsichtlich optimaler Futterplätze etc. auf die Lösung von reellwertigen Optimierungsproblemen übertragen

Partikelschwärme

- Ist biologisch inspiriert, in diesem Falle durch Flocking, also z. B. das Schwarmverhalten von Vögeln.
- Statt von Vögeln sprechen wir aber von Partikeln (engl. particles). Die Idee ist, dass diese N Partikel sich kollektiv durch den Suchraum bewegen auf der Suche nach dem Optimum.
- Die Position jedes Partikels ist direkt mit einer Güte assoziiert, letztlich einem Zahlenwert, der die Qualität dieser Position als Lösung eines Suchproblems beschreibt.
- Die Annahme ist, dass ein Partikel seinen aktuellen Wert mit seinen Nachbarn kommunizieren kann, dass er die Position des besten Nachbars kennt und dass er die Position seines bisher besten Wertes erinnern kann.

Partikelschwärme

- Jede Partikel bewegt sich mit einer bestimmten Geschwindigkeit und Richtung durch den Suchraum. Dazu aktualisiert jede Partikel seine Position als Kompromiss aus drei Richtungsindikatoren:

- der aktuellen Richtung des Partikels,
- der Richtung, in der der bisher beste Wert des Partikels liegt,
- und der Richtung, in der der momentan beste Nachbar liegt.

Partikelschwärme

Die Aktualisierungsregel der Position *xi* eines Partikels *i* ist

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

für die neue Geschwindigkeit v_i^{t+1} des Partikels.

Diese neue Geschwindigkeit erhalten wir, indem wir über die drei oben genannten Richtungen summieren:

$$v_i^{t+1} = av_i^t + b(x_i^b - x_i^t) + c(x_j^t - x_i^t)$$

mit den Konstanten $a > 0$, $b > 0$ und $c > 0$ zur Gewichtung.

Partikelschwärme

$(x_i^b - x_i^t)$ ist die Differenz aus der Position x_i^b des bisher besten Wertes, den Partikel *i* gesehen hat, und seiner aktuellen Position x_i^t.

$(x_j^t - x_i^t)$ ist die Differenz aus der Position x_j^t des aktuell besten Nachbarns und der aktuellen Position x_i^t des Partikels.

Was fehlt noch?

Partikelschwärme

$(x_i^b - x_i^t)$ ist die Differenz aus der Position x_i^b des bisher besten Wertes, den Partikel i gesehen hat, und seiner aktuellen Position x_i^t.

$(x_j^t - x_i^t)$ ist die Differenz aus der Position x_j^t des aktuell besten Nachbarns und der aktuellen Position x_i^t des Partikels.

Es fehlt das Element der Exploration, also der Erforschung neuer Möglichkeiten!

WIE???

Partikelschwärme

- Einfache Randomisierung über zwei Zufallszahlen
- Die Multiplikation mit den gleichverteilten Zufallsvariablen $z_b \in [0, o_b]$ und $z_c \in [0, o_c]$ mit Obergrenzen $o_b > 0$ und $o_c > 0$ liefert

$$v_i^{t+1} = av_i^t + bz_b(x_i^b - x_i^t) + cz_c(x_j^t - x_i^t).$$

Partikelschwärme

- Die typische Schwarmgröße liegt im Bereich von $N \in [20, 200]$ Partikeln und die Nachbarschaftsgröße definiert man so, dass ca. 10% des Schwarms im Durchschnitt Nachbarn sind.
- Typische Werte für a, b, c, o_b und o_c liegen auf dem Intervall $[0.1, 1]$.

Partikelschwärme

```
for each particle i = 1, ..., S do
   Initialize the particle's position with a uniformly distributed random vector: xi ~ U(blo, bup)
   Initialize the particle's best known position to its initial position: pi ← xi
   if f(pi) < f(g) then
       update the swarm's best known  position: g ← pi
   Initialize the particle's velocity: vi ~ U(-|bup-blo|, |bup-blo|)
while a termination criterion is not met do:
   for each particle i = 1, ..., S do
      for each dimension d = 1, ..., n do
         Pick random numbers: rp, rg ~ U(0,1)
         Update the particle's velocity: vi,d ← ω vi,d + φp rp (pi,d-xi,d) + φg rg (gd-xi,d)
      Update the particle's position: xi ← xi + vi
      if f(xi) < f(pi) then
         Update the particle's best known position: pi ← xi
         if f(pi) < f(g) then
            Update the swarm's best known position: g ← pi
```

Partikelschwärme evolutionär

PARTIKELSCHWARM(Bewertungsfunktion F)

$t \leftarrow 0$
$P(t) \leftarrow$ initialisiere die Population der Größe μ
bewerte $P(t)$ durch F
$best \leftarrow$ Genotyp des besten Individuums in $P(t)$
while Terminierungsbedingung nicht erfüllt
do $P' \leftarrow \langle\rangle$
  for $i \leftarrow 1, \ldots, \mu$
  do (sei $A^{(i)}.S = (v_1^{(i)}, \ldots, v_\ell^{(i)}, B_1^{(i)}, \ldots, B_\ell^{(i)})$)
    $u_1, u_2 \leftarrow$ wähle Zufallszahlen gemäß $U([0,\ 1])$
    for each $k \in \{1, \ldots, \ell\}$
    do $v_k' \leftarrow \beta \cdot v_k^{(i)} + \alpha_1 \cdot u_1 \cdot (B_k^{(i)} - A^{(i)}.G_k) + \alpha_2 \cdot u_2 \cdot (best_k - A^{(i)}.G_k)$
    if $\|(v_1', \ldots, v_\ell')\| > MAX$ (maximale Veränderung)
    then $(v_1', \ldots, v_\ell') \leftarrow$ skaliere den Veränderungsvektor auf die Länge MAX
    for each $k \in \{1, \ldots, \ell\}$
    do $A'.G_k \leftarrow A_k^{(i)} + v_k'$
    bewerte A' durch F
    if $A'.F \succeq B^{(i)}.F$
    then $P' \leftarrow P' \circ \langle(A_1', \ldots, A_\ell', v_1', \ldots, v_\ell', B_1^{(i)}, \ldots, B_\ell^{(i)})\rangle$
    else $P' \leftarrow P' \circ \langle(A_1', \ldots, A_\ell', v_1', \ldots, v_\ell', A_1', \ldots, A_\ell')\rangle$
  $t \leftarrow t + 1$
  $P(t) \leftarrow P'$
  $best \leftarrow$ Genotyp des besten Individuums in $P(t)$
return bestes Individuum aus $P(t)$

Partikelschwärme

- Die Individuen bestehen dabei aus dem Genotyp $A.G \in \mathbb{R}^l$ und den Zusatzinformationen $A.S = (v_1, \ldots, v_l, B_1, \ldots, B_l) \in \mathcal{Z} = \mathbb{R}^{2l}$.
- Dabei stellt $v = (v_1, \ldots, v_l)$ einen Veränderungsvektor dar, der bei der Modifikation der Individuen benutzt wird und $B = (B_1, \ldots, B_l)$ repräsentiert den besten bisher auf dem Weg des Individuums gefundenen Punkt im Suchraum.
- In jeder Generation wird der Veränderungsvektor der Individuen durch die soziale Interaktion mit benachbarten Individuen modifiziert (von v zu v') und anschließend auf den Genotyp angewandt, d. h. für alle $k \in \{1, \ldots, l\}$ gilt

$$A'.G_k = A.G_k + v'_k.$$

Partikelschwärme

- In die Modifikation von gehen zwei Komponenten ein:
  - das Bestreben eines Individuums, zu seinen Erfolgen zurückzukehren, d. h. den Veränderungsvektor so zu modifizieren, dass er zum besten Lösungkandidaten B zurückführt und
  - eine Orientierung des Individuums an den besten Erfolgen seiner Nachbarn.

Partikelschwärme

- Sei also best der beste bisher gefundene Lösungskandidat in einer Nachbarschaft, die oft für ein Individuum $A^{(i)}$ einfach aus den Individuen $A^{(i1)}, A^{(i)}$ und $A^{(i+1)}$ besteht.
- Dann wird der Veränderungsvektor mittels zweier Zufallszahlen $u_1, u_2 \equiv U([0,1])$ folgendermaßen modifiziert.

$$v' = \beta \cdot v_k^{(i)} + \alpha_1 \cdot u_1 \cdot (B^{(i)} - A^{(i)}.G_k) + \alpha_2 \cdot u_2 \cdot (best - A^{(i)}.G_k).$$

- Dabei ist β ein Trägheitsfaktor, α_1 bestimmt, wie stark die gespeicherte beste Position eingeht, und α_2 ist ein sozialer Faktor, wie stark ein Individuum sich an den Nachbarn orientiert.
- In Algorithmus PARTIKELSCHWARM wird eine globale Nachbarschaft benutzt, d.h. best ist das beste Individuum der aktuellen Population.