Intelligente Systeme

- Skript -

Prüfungsform: Referat

Adrian Helberg
Matr.Nr. 2309051

Abgabe: 18.02.2020

Zusammenfassung

Diese Arbeit beschäftigt sich mit Konzepten der vier Kernbereiche aus *intelligenten Systemen*: **Suchen**, **Lernen**, **Verarbeitung von Sequenzen** und **Ethik**. Zum einen werden Problemstellungen aufgezeigt, zum anderen auf Implementationen verwiesen, die in der zugehörigen Präsentation vorgestellt werden.
Eine Lösungsstrategie für das *Problem des Handlungsreisenden (Traveling Salesman Problem)* wird im Kapitel **Suchen** beschrieben. *Selbstorganisierte Karten (Selforganizing Maps)* zeigen eine Strategie des **Lernen**s, ein *Deep Learning*- Ansatz zur Datenanalyse von Sensordaten mobiler Endgeräte gibt Einblicke in die **Sequenzverarbeitung** und eine Diskussion zum *Trolley Problem* lädt zur Diskussion über die **Ethik** in Verbindung mit *intelligenten Systemen* ein.

Inhalt

1 Suchen

Dieses Kapitel beschäftigt sich mit der Umsetzungen einer Lösungsstrategie für das *Problem des Handlungsreisenden (Traveling Salesman Problem)* mithilfe einer informierten Suche. Im speziellen eine Umsetzung mit einem genetischen Algorithmus.

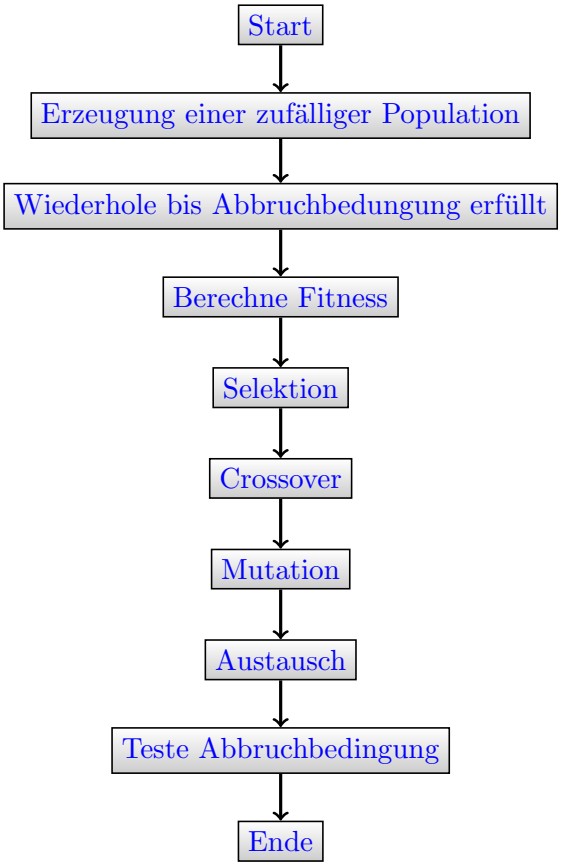
1.1 Problem des Handlungsreisenden

Das Problem des Handlungsreisenden ist ein kombinatorisches Optimierungsproblem der theoretischen Informatik. Dabei muss ein Handlungsreisender eine Menge von Städten besuchen. Er beginnt bei einer bestimmten Stadt und muss, nachdem jede andere Stadt besucht wurde, zu dieser zurückkehren. Das Optimierungsproblem besteht bei der Festlegung der Reihenfolge der zu besuchenden Städte, sodass die gesamte Distanz der Reise minimal ist. Das Problem ist als NP-vollständig klassifiziert.

1.2 Genetischer Algorithmus

Evolutionäre Algorithmen sind eine Klasse von stochastischen, heuristischen Optimierungsverfahren. Der Name lässt sich von der Evolution natürlicher Lebewesen ableiten. Ziel von genetischen Algorithmen ist es optimierte Lösungen zu Aufgabenstellungen zu finden, bei denen ein Auffinden einer akzeptablen Lösung aus Gründen der kombinatorischen Komplexität misslingt. So gibt es beim Problem des Handlungsreisenden mit 10 Orten z.B. bereits $10! = 3628800$ Lösungen. Kern eines genetischen Ansatzes ist die Veränderung von Mengen an Problemlösungen, sodass gute Lösungen mit einer großen Wahrscheinlichkeit und schlechte Lösungen mit geringen Wahrscheinlichkeit erhalten bleiben. Das Zusammenführen von Teilen guter Lösungen kann noch bessere Ergebnisse liefern. So kann verhindert werden, dass ein Algorithmus zur Optimierung an einem lokalen Optimum “hängenbleibt“.

1.2.1 Allgemeiner Ablauf



1.3 Chromosomen

Eine Rundreise des *Handlungsreisenden* wird duch ein Chromosom repräsentiert. Er bereist hierzu alle Städte und endet in der Stadt, in der begonnen hat. Die Population besteht aus der Anzahl solcher Rundreisen, die durch Vektoren modelliert werden.

Eine mögliche Rundreise wäre also: $\left\{ \begin{array}{c} 3 \\ 5 \\ 1 \\ 2 \\ 4 \\ 3 \end{array} \right\}$

1.4 Fitness

Eine Berechnung der Fitness in Abhängigkeit von der Gesamtdistanz einer Rundreise bietet sich für das *Problem des Handlungsreisenden* an. Allerding soll der Fitness-Wert proportional zur Eignung (antiproportional zu den Kosten) eines Chromosoms sein. Für die Kosten wird die Gesamtdistanz eines Chromosoms, für die Fitness der inverse Kostenwert verwendet.

1.5 Selektion

Die Auswahl eines Chromosomenpaars aus der Population soll wahrscheinlicher werden, wenn der Fitness-Wert steigt. Dies kann mittels einer Summe über alle Fitness-Werte gleichverteilte Zufallszahl realisiert werden. Der Wertebereich der Zufallszahl wird in Abschnitte unterteilt, deren Größe genau den Fitnesswerten der einzelnen Chromosomen entsprechen. Der Abschnitt, in den die Zufallszahl fällt, bestimmt die Selektion des zugehörigen Chromosoms.

1.6 Kreuzung

Ein zufälliger Kreuzungspunkt innerhalb des Vektors eines selektierten Chromosoms wird bestimmt. Der neue Vektor ist bis zum Kreuzpunkt identisch mit dem ursprünglichen. Der restliche Teil des neuen Vektors wird wie folgt bestimmt. Gehe den zweiten ausgewählten Vektor vom Anfang bis zum Ende durch und füge die erste Zahl, die noch nicht im neu zu bildenden Chromosoms steht, an die nächste freie Stelle ein. An der letzten Stelle des neuen Chromosomes muss die gleiche Zahl stehen wie an seiner ersten Stelle. Dieser Prozess wird für beide ausgewählten Chromosomen durchgeführt.

1.7 Mutation

Mit einer kleinen Wahrscheinlichkeit findet in der Kreuzung eine Mutation statt. Diese wird wie folgt implementiert. Ein zufälliges Paar von Indizes mit Werten zwischen 1 und der Anzahl der Städte wird erzeugt. Die Werte, die den Indizes entsprechenm werden im Chromosom getauscht. Dabei darf die Eigenschaft, dass Start- und Zielwert identisch sind, nicht verletzt werden.

1.8 Austausch

In jeder Iteration werden in der Population die beiden schlechtesten Chromosomen, d.h. diejenigen mit den geringsten Fitness-Werten, durch die beiden

neuen Chromosomen ersetzt, falls letztere bessere Fitness-Werte haben. Es wird gewährleistet, dass in der Population alle Chromosomen verschieden sind.

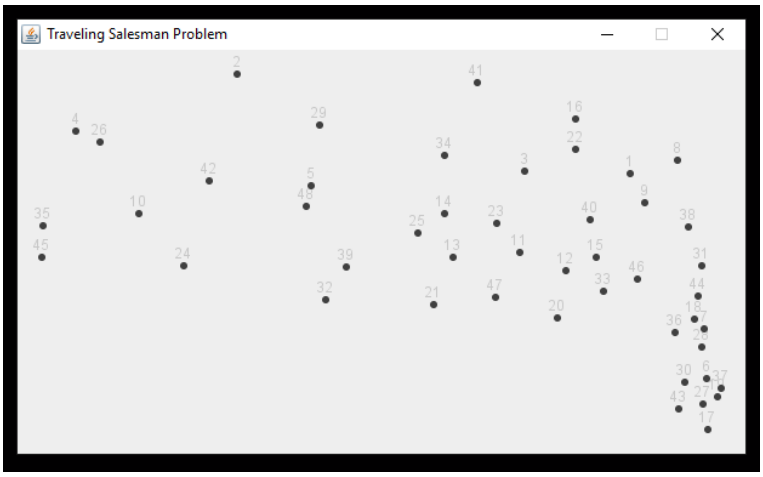
1.9 Realisierung mit Java

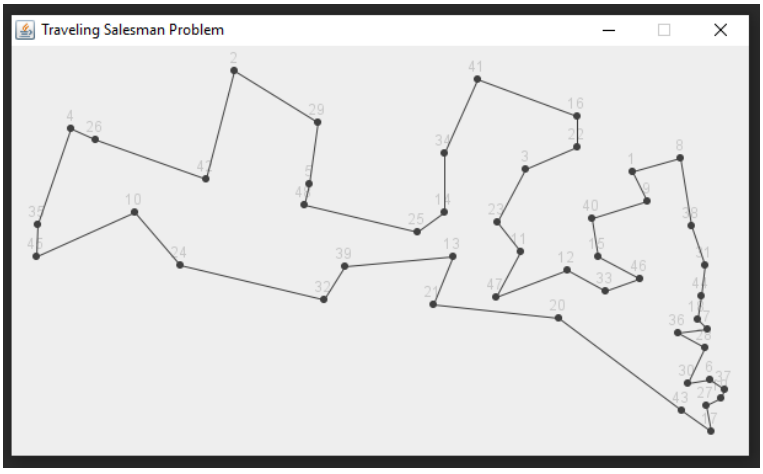
1.9.1 Eigenschaften

```
----------Genetic Algorithm Properties----------
Number of Cities:   48
Population Size:    500
Max. Generation:    500
k Value:            3
Elitism Value:      1
Force Uniqueness:   false
Local Search Rate:  0.0
Crossover Type:     UNIFORM_ORDER
Crossover Rate:     90.0%
Mutation Type:      INSERTION
Mutation Rate:      4.0%
```

```
-----------Genetic Algorithm Results------------
Average Distance of First Generation:  158045
Average Distance of Last Generation:   37931
Best Distance of First Generation:     126405
Best Distance of Last Generation:      36643
Area Under Average Distance:           33718137
Area Under Average Distance:           26872439
```

1.9.2 Visualisierung





2 Lernen

Dieses Kapitel beschreibt ein unüberwachtes Lernverfahren der Neuroinformatik: Die *Selbstorganisierenden Karten*, *Kohonen-Karten*, oder *Kohonen Feature Maps*. Also maschinelles Lernen ohne oder vorher bekannte Ergebniswerte.

2.1 Motivation

Das Funktionsprinzip von *SOM*s (*Self-organizing maps*) beruht auf der Erkenntnis, dass viele Strukturen im Gehirn eine lineare oder planare Topologie aufweisen. Eingehende Signale, wie z.B. visuelle Reize, sind jedoch multidimensional. Abbildung 1 zeigt eine Skizze der topologischen Ansicht des sensorischen Kotex, der die sensorischen Eingaben des Körpers auswertet. Es entsteht eine Klassifizierung, da ähnliche Ausgaben in direkter Umgebung angesiedelt werden. Beispielsweise sind Signale an die Finger in unmittelbarer Nachbarschaft zueinander. Solch eine Nachbarschaftsbeziehung zwischen Neuronen ist ein grundlegendes Merkmal von *SOM*s

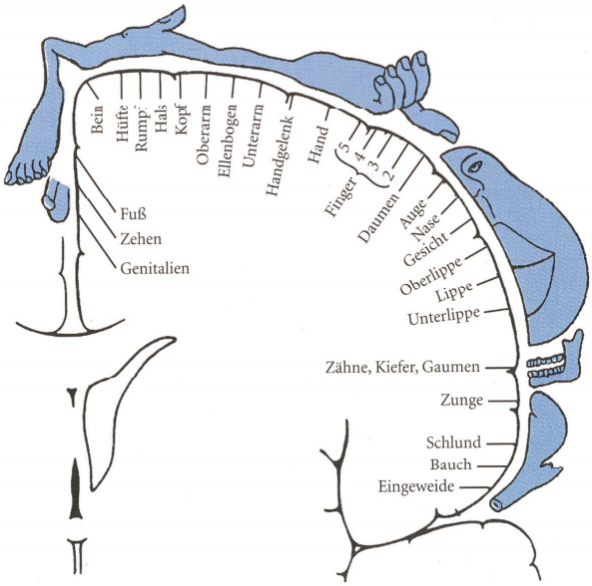


Abbildung 1: Topologische Abbildung des sensorischen Kortex

Eine topologieerhaltende Abbildung auf weniger Dimensionen (siehe Kapitel **Motivation**), hier eine Fläche, wird als Karte bezeichnet. Diese Karten im Gehirn kommen bei allen Säugetieren vor und sind dynamisch. Dies wird an folgendem Beispiel deutlich. Beim Verlust von Körperteilen wird der Platz des entsprechenden Feldes mit der Zeit von anderen Feldern eingenommen (da die eingehenden Signale wegfallen, Abbildung 2).

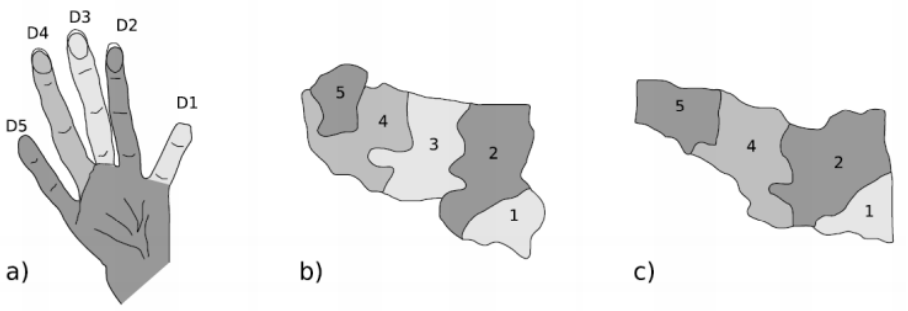


Abbildung 2: a) Schema einer Affenhand b) Bereich des Kortex der Hand c) 2 Monate nach Amputation von Finger D3

2.2 Aufbau

*SOM*s verwenden ein sehr einfaches Neuronenmodell ohne Aktivierungsfunktion (Abbildung 3)

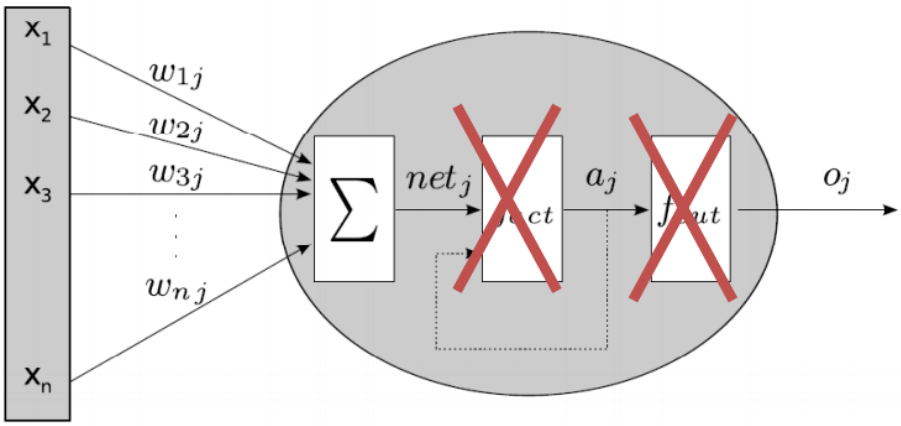


Abbildung 3: Neuron einer *SOM*

2.3 Netz

Organisiert man Neuronen als Eingabe- und Ausgabeneuronen in jeweils einer Schicht, entsteht ein neuronales Netz, in dem die Eingabeneuronen über die Gewichtsvektoren komplett mit den Ausgabeneuronen verbunden sind (Abbildung 4).

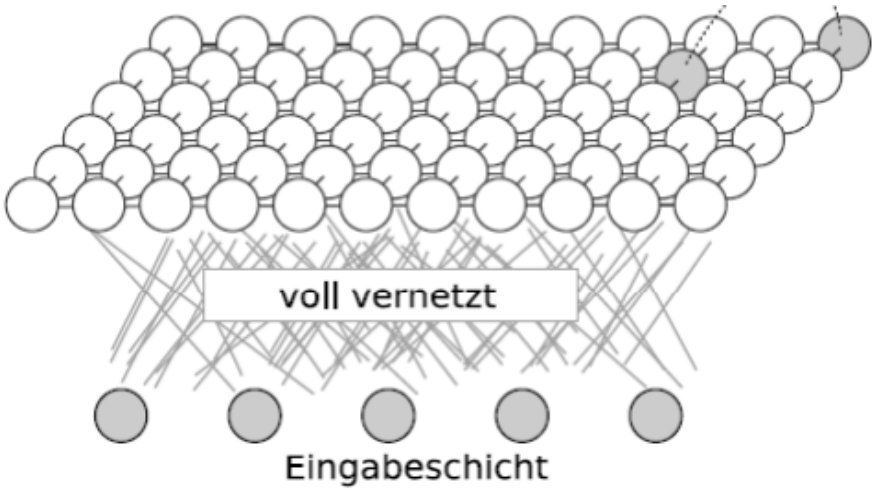


Abbildung 4: *Kohonen*-Netz

2.4 Lernverfahren

*SOM*s lernen durch Auswahl eines Neurons anhand deren „Eignung“. Die Eignung wird über die euklidische Norm berechnet. Hierbei wird der Abstand zwischen Eingabevektor und Gewichtsvektor bestimmt. Besitzen Neuronen die gleiche euklidische Norm, wird über das Zufallsprinzip entschieden, welches Neuron ausgewählt wird. Das Besondere an *SOM*s ist, dass nicht nur der Gewichtsvektor des ausgewählten Neurons (Siegerneuron) verändert wird, sondern auch die der Neuronen in der Umgebung des Siegerneurons.

2.5 Distanzfunktion

Anstelle der euklidischen Norm werden in der Praxis auch andere bewährte Distanzfunktionen verwendet. Einige Beispiele sind die gaussche Glockenfunktion, die Zylinderfunktion und der „Mexican-Hat“ (Abbildung 5).

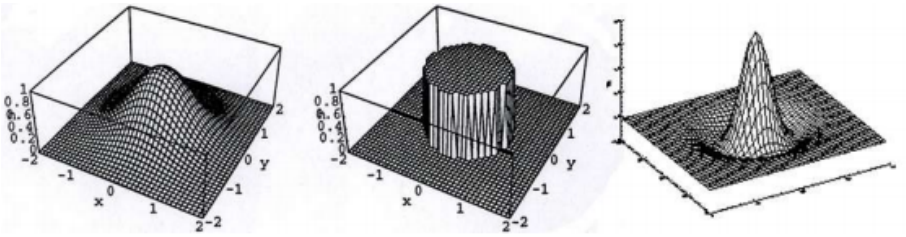


Abbildung 5: Gaussche Glockenfunktion, Zylinderfunktion und Mexican-Hat

2.6 Siegerneuron anpassen

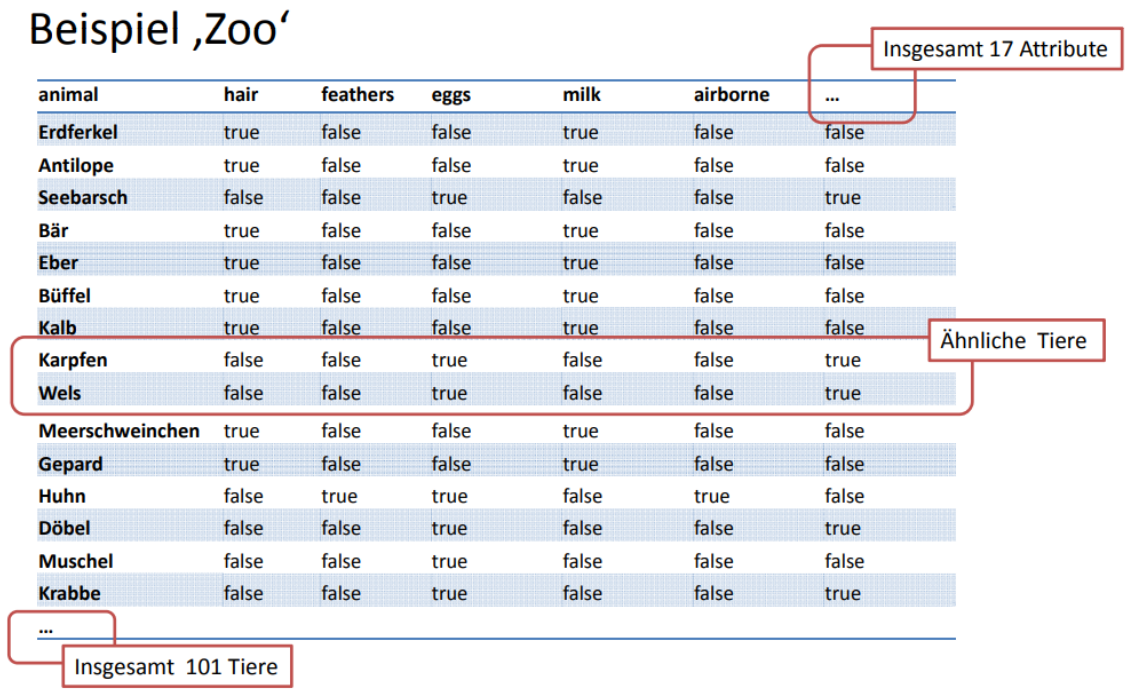
$W_c = W_c + \Delta W_c$ und $\Delta W_c = \eta * (x - W_c)$, mit c als Siegerneuron, W_c als Gewichtungsvektor und Eingabevektor x. Es gilt $0 < \eta < 1$.

2.7 Nachbarneuron des Siegerneurons anpassen

Nachbarn des Siegerneurons lernen umso stärker, je näher sie sich am Siegerneuron befinden:

$\Delta W_j = \eta * h_{cj} * (x - W_j)$

2.8 Visualisierung



animal	hair	feathers	eggs	milk	airborne	...
Erdferkel	true	false	false	true	false	false
Antilope	true	false	false	true	false	false
Seebarsch	false	false	true	false	false	true
Bär	true	false	false	true	false	false
Eber	true	false	false	true	false	false
Büffel	true	false	false	true	false	false
Kalb	true	false	false	true	false	false
Karpfen	false	false	true	false	false	true
Wels	false	false	true	false	false	true
Meerschweinchen	true	false	false	true	false	false
Gepard	true	false	false	true	false	false
Huhn	false	true	true	false	true	false
Döbel	false	false	true	false	false	true
Muschel	false	false	true	false	false	false
Krabbe	false	false	true	false	false	true
...						

Abbildung 6: Beispiel „Zoo“

Reduktion von mehreren Dimension auf 2 (Karte). Sind ähnliche Tiere eines Zoos benachbart? (Abbildung 6+7)

2.9 Anwendungsgebiete

- Dimensionsreduktion
- Optimierung
- Data Mining- Cluster
- Data Mining - Regeln
- Überwachenung und Anomaliedetektion
- Kontextkarten

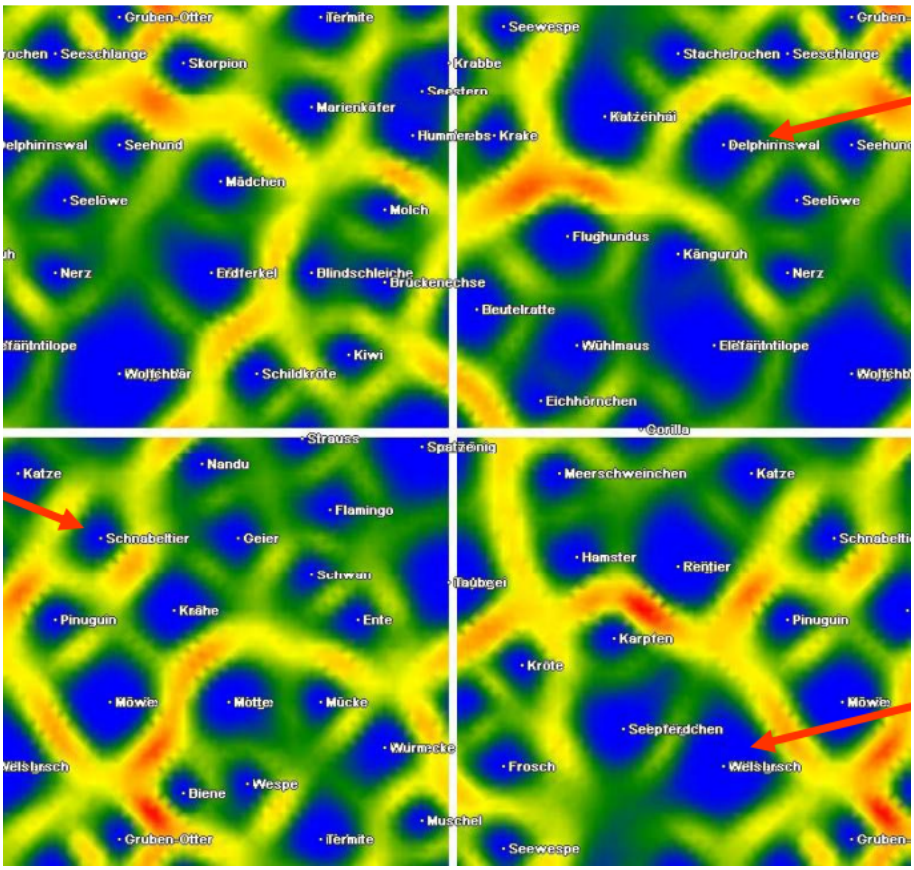


Abbildung 7: Beispiel „Zoo“

3 Sequenzen

4 Ethik

5 Quellen