UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR ROBOTIK
UND KOGNITIVE SYSTEME

Künstliche Intelligenz
Sommersemester 2020
20. April 2020

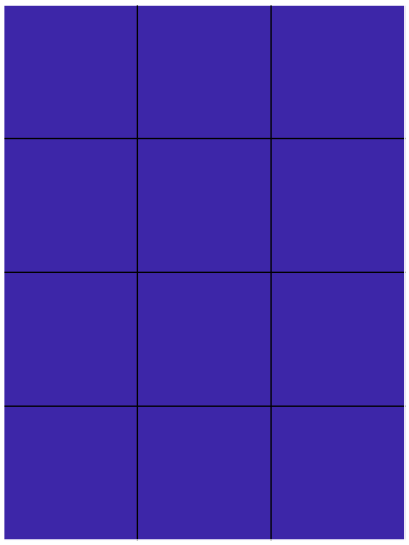
Jannis Hagenah
hagenah@rob.uni-luebeck.de

Projekt 1: Malen nach Zahlen

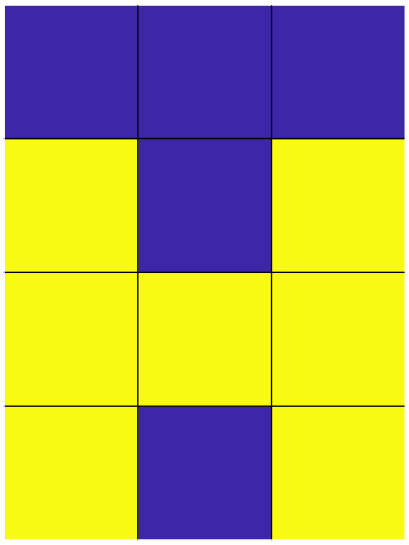
Abgabe bis zum 18.05.2020, 12 Uhr im Moodle. Abgegeben wird eine `.zip`-Datei, in der alle abzugebenden Dateien enthalten sind. Zur Abgabe gehört der kommentierte, im entpackten Verzeichnis lauffähige Code sowie eine Zusammenfassung (mind. 3 Seiten im IEEE Proceedings-Format, `https://www.ieee.org/conferences/publishing/templates.html`), in der die genutzten Ansätze und Verfahren sowie eventuelle Annahmen beschrieben, erläutert und diskutiert werden. Hier werden auch die Ergebnisse vorgestellt, kritisch geprüft und Vorschläge für künftige Verbesserungen beschrieben. Bitte achten Sie darauf, dass die Namen aller Gruppenmitglieder in diesem Dokument vollständig und konsistent angegeben sind.

Aufgabe 1: Da steh ich vor dem Holstentor und bin so klug als wie zuvor (25 Punkte)

Jeder Lübecker, der etwas auf sich hält, könnte auf Anhieb das Holstentor malen, wenn er oder sie danach gefragt würde. Doch kann das auch ein Computer? Bringen Sie es ihm bei!
Dabei sollen uninformierte Suchverfahren zum Einsatz kommen, um ein binäres Bild des Holstentores auf eine anfangs leere, digitale Leinwand zu malen. Die Leinwand ist eine (zugegeben recht grob aufgelöste) Bildfläche von 4×3 Pixeln, wobei jeder Pixel nur den Wert 0 oder den Wert 1 annehmen kann. Gestartet wird mit einer leeren Leinwand (alle Pixelwerte = 0, siehe Abbildung 1 links), das Zielbild ist in Abbildung 1 rechts dargestellt. In jedem Schritt kann nur ein Pixelwert verändert werden.



Leinwand



Holstentor

Laden Sie sich zunächst die vorgegebenen MATLAB-Funktionsrümpfe aus dem Moodle-Kurs herunter. Halten Sie sich bei Ihrer Implementierung an die in den Funktionsrümpfen vorgegebene Nomenklatur! Folgende Skripte sind vorgegeben:

- `createAdjacency.m`

  Dieses Skript definiert die Adjazenzmatrix für das gegebene Problem und speichert sie in der Datei `Adjacency.mat` ab, damit sie nicht mit jedem Durchlauf neu berechnet werden muss.

- `TestSearch.m`

  Dieses Skript führt die einzelnen Suchverfahren aus. Hier müssen noch Start- und Zielknoten definiert werden. Außerdem existiert die Datei `Adjacency.mat` noch nicht, bevor Sie diese nicht mittels `createAdjacency.m` erstellt haben.

- `bfs_gs.m`

  Dieses Skript stellt einen Funktionsrumpf für die Breitensuche als Graphsuche bereit.

- `dfs_gs.m`

  Dieses Skript stellt einen Funktionsrumpf für die Tiefensuche als Graphsuche bereit.

- `dfs.m`

  Dieses Skript stellt einen Funktionsrumpf für die einfache Tiefensuche bereit.

Sie beschließen, das Malen des Holstentores als Suchproblem in einem ungerichteten Graphen aufzufassen, bei dem jeder Knoten ein Bild repräsentiert. Um diese Repräsentation einfach zu gestalten nutzen Sie das vektorisierte Bild (12-Bit String) als Zustandsbeschreibung.

a) Wie viele Knoten hat der resultierende Graph?

b) Implementieren Sie eine Funktion, die Ihnen den ungerichteten Graph als Adjazenzmatrix beschreibt. Nutzen Sie diese Implementierung, um die Adjazenzmatrix A zu berechnen. Weitere Informationen zu Adjazenzmatrizen finden Sie zum Beispiel bei Wikipedia (`http://de.wikipedia.org/wiki/Adjazenzmatrix`). Ein bei der Zustandskodierung hilfreicher Befehl könnte `dec2bin` sein.

c) Implemetieren Sie in `bfs_gs.m` die Breitensuche als Graphsuche. Es kann hilfreich sein, den aktuell besuchten Knoten auszugeben. Bauen Sie außerdem ein Abbruchkriterium ein, falls die Lösung nach sehr vielen Schritten noch nicht gefunden wurde.

d) Implemetieren Sie in `dfs_gs.m` die Tiefensuche als Graphsuche. Es kann hilfreich sein, den aktuell besuchten Knoten auszugeben. Bauen Sie außerdem ein Abbruchkriterium ein, falls die Lösung nach sehr vielen Schritten noch nicht gefunden wurde.

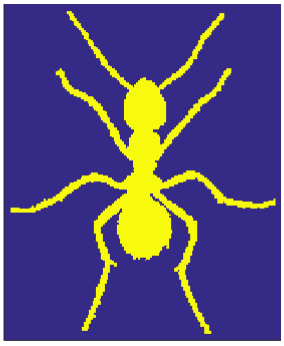
e) Implemetieren Sie in `dfs.m` die einfache Tiefensuche. Es kann hilfreich sein, den aktuell besuchten Knoten auszugeben. Bauen Sie außerdem ein Abbruchkriterium ein, falls die Lösung nach sehr vielen Schritten noch nicht gefunden wurde.

f) Testen Sie Ihre Suchverfahren anhand des Skripts `TestSearch.m`. Wird das Holstentor von allen Verfahren gefunden? Wie lange benötigen die verschiedenen Verfahren? Starten Sie auch von einem zufälligen Bild statt einer leeren Leinwand. Ändert sich die Performance der Verfahren?

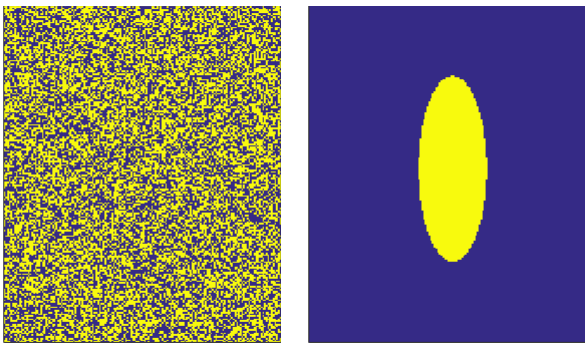
Aufgabe 2: Waiting for the ant to come... (25 Punkte)

Wollten Sie sich schon immer mal Ihre eigenen Ameisen züchten? Haben Sie sich als Kind immer eine eigene Ameisenfarm zum Geburtstag gewünscht, sie aber nie bekommen? Dann bekommen Sie jetzt endlich die Gelegenheit, sich wie Gott (oder Charles Darwin) zu fühlen und digitale Ameisen aus (mehr oder weniger) nichts zu erschaffen! Alles, was Sie dafür brauchen, ist ein wenig Wissen über Genetische Algorithmen.

Ein Individuum besteht aus einem 160×198 Pixel großen Binärbild. Das Ziel Ihrer Kreationen ist folgende Ameise:



Gehen Sie dabei von zwei unterschiedlichen Startpopulationen aus, *Zufalls-Ameisen* and *Blob-Ameisen*:

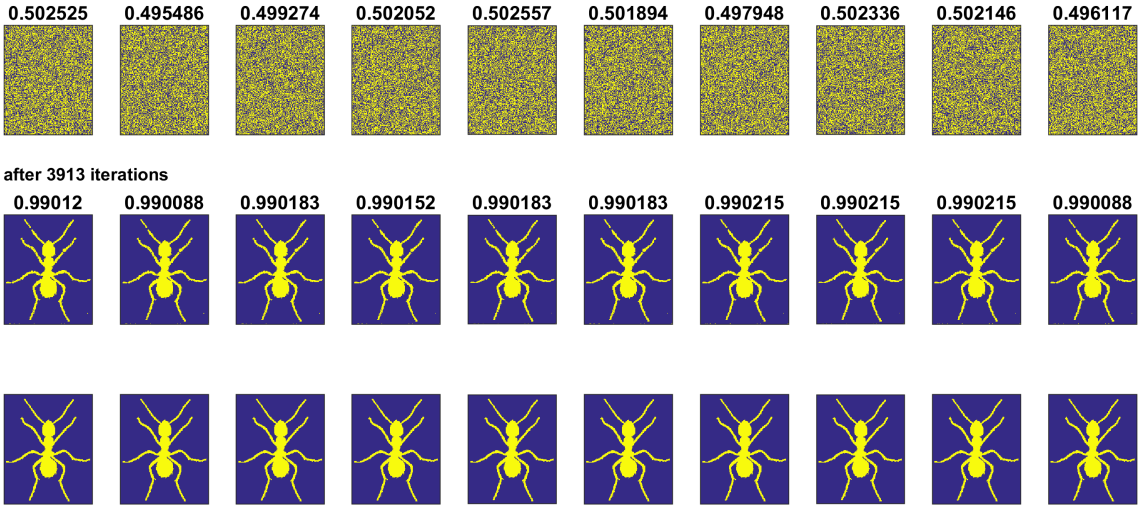


Laden Sie sich die vorgegebenen MATLAB-Funktionsrümpfe aus dem Moodle-Kurs herunter.

- `antFarm.m`

  Dieses Skript führt den gesamten Genetischen ALgorithmus aus. Setzen Sie die drei notwendigen Parameter: die Anzahl der überlebenden Individuen, die Anzahl der Mutationen pro Individuum, and die maximale Anzahl der Iterationen. Die Populationsgröße ist auf 10 festgelegt und soll nicht verändert werden! Wählen Sie eine der zwei Startpopulationen aus. Implementieren Sie Ihre Fitness-Funktion, die Selektion der fittesten Individuen, sowie Ihre Rekombinations- und Mutationsstrategie. Das Skript generiert eine Graphik, die automatisch als png-Bild gespeichert wird.

Das Ergebnis könnte beispielsweise so aussehen:



Die erste Zeile zeigt die Startpopulation und die zugehörigen Fitness-Werte. Das Ergebnis der Kreation steht in der zweiten Zeile. Zum Vergleich zeigt die letzte Zeile eine *perfekte Population*.

a) Welches Problem ergäbe sich, wenn Sie das Züchten der Ameise analog zu Aufgabe 1 mit uninformierten Suchverfahren lösen wollten?

b) Definieren Sie eine sinnvolle Fitness-Funktion und erläutern Sie diese. Zur besseren Vergleichbarkeit sollte die Funktion ein relatives Verhätnis richtiger Pixel darstellen.

c) Definieren Sie eine sinnvolle Rekombinationsstrategie und erläutern Sie diese. Neue Individuen sollen nur aus der Rekombination zweier zufällig ausgewählter, selektierter Individuen der aktuellen Generation erzeugt werden. Jedes neue Individuum sollte gleich viele Gene von *Mutter* und *Vater* bekommen. Der Einfachheit halber soll es möglich sein, dass *Mutter* und *Vater* das gleiche Individuum sind ("Das Leben findet einen Weg").

d) Definieren Sie eine sinnvolle Mutationsstrategie und erläutern Sie diese. In diesem Fall können wir Mutation sehr weit auffassen, da zufälliges invertieren einzelner Pixel zu langen Rechenzeiten führen kann. Sie dürfen also eine Art *Evolutions-Strategie* einbauen. Aber: halten Sie den Grad an Vorwissen so gering wie möglich! Während der Mutation darf kein Pixel systematisch auf seinen "richtigen" Wert gesetzt werden! Die Mutation findet nach der Rekombination statt und insgesamt werden $n_{mutation}$ Individuen mutiert.

e) Implementieren Sie Ihre Fitness-Funktion, die Rekombinations- und die Mutationsstrategie. Evaluieren Sie Ihre Methode anhand beider Startpopulationen. Finden Sie Parameter-Kombinationen, die *ameisenähnliche* Populationen erzeugen, also Populationen mit hoher Fitness. Die Parameterkombinationen können für beide Startpopulationen unterschiedlich sein. Versuchen Sie, die Parameter so zu wählen, dass die Rechenzeit 5 Minuten nicht überschreitet (dieser Richtwert hängt stark von Ihrer Rekombinations- und Mutationsstrategie ab)! **Achtung: Bei jeder Ausführung des Skripts wird das generierte Bild überschrieben.**

f) Kommentieren und diskutieren Sie Ihre Ergebnisse. Versuchen Sie zu erklären, wie Ihre (guten oder schlechten) Ergebnisse mit Ihren Evolutionsstrategien und Parameterwahlen zusammenhängen. Welchen Einfluss haben unterschiedliche Wahlen der Parameter auf die Laufzeit und auf die Qualität der Ergebnisse?

Aufgabe 3: Und wenn alles anders wäre? (10 Punkte)

Beantworten Sie folgende Fragen innerhalb Ihres Abgabedokumentes. Begründen Sie Ihre Antworten!

a) Könnten Sie die gegebenen Probleme aus den Aufgaben 1 und 2 auch mittels Hill-Climbing lösen?Wenn ja, wie müssten Sie das Problem formulieren? Falls nein, wieso nicht? Welche Probleme würden auftreten? Welche eventuellen Vorteile hätte das Hill-Climbing gegenüber den hier verwendeten Verfahren?

b) Stellen Sie sich vor, ein nerviges Kind würde immer, wenn Sie mehrere Pixel abgeändert hätten, einen weiteren Pixel umändern, einfach nur um Sie zu ärgern. Könnten Sie auf das Problem algorithmisch eingehen? Gibt es einen Unterschied, ob das Kind die Pixel zufällig oder nach einem Muster ändert?