Maximilian Mühlfeld, Informatik MA, 580070,
Malte Schmitz, Informatik MA,xxxxxx,
Eike von Tils, Informatik MA,xxxxxx

1 Einleitung

Das Ziel des Versuches war es, anhand des Beispieles des Snacecubes einen Evolutionären (Genetischen) Algorithmus zu implementieren. In unserem Fall handelt es sich bei den Individuen des Algorithmus um "snakes" (Schlangen). Diese werden durch einen String repräsentiert, der für jedes Glied der Schlange die Richtung angibt, in der das Folgeglied zu finden ist. Da die Schlange sich im 2D aufhält, gibt es die Richtungen I (geradeaus), R (rechts) und L (links). Die Fitness der Schlange wird aus einer Energiefunktion bezogen auf ihre Form und Lage im Raum berechnet.
Hierzu wurde ein Algorithmus nach dem folgenden Schema entwickelt:

1. Initiale Population erzeugen
2. Bis maximale Epoche oder minimale Energie erreicht wird, wiederhole:
   a) Evaluation
   b) Selektion
   c) Reproduktion

1.1 Initiale Population erzeugen

Die Initiale Population besteht bei unserem Problem aus einer Menge von zufällig erstellten Schlangen. Eine Schlange wird durch einen Bitstring repräsentiert. Da die Position der 'I'-Glieder der Schlange konstant bleibt, sind nur die Richtungen 'R' und 'L' von Bedeutung. Wir codieren diese Richtungen als Richtungsänderungen bezüglich der vorherigen Position. Eine 1 steht hierbei für Richtungsänderung, eine 0 für gleichbleibende Richtung. Beginn ist immer eine Rechtsdrehung.

1.2 Evaluation

Die Berechnung der Fitness zu jedem Individuum erfolgt aus zwei Energiefunktionen, E_a und E_b , nach Aufgabenstellung.
E_a berechnet hierbei den Durchmesser des kleinsten Kreises auf der Ebene, der die Schlange umschließt.
E_b berechnet die gesamte Fläche aller Löcher die die Schlange bildet.
Die genauere Erläuterung zur Implementierung folgt bei der Funktionsbeschreibung.

1.3 Selektion

Die Selektion erfolgt als zufällige Auswahl von Individuen der aktuellen Population. Die Wahrscheinlichkeit, das ein Individuum gewählt wird, ist hierbei von der Fitnessfunktion abhängig. Individuen mit höherer Fitness werden dabei bevorzugt.

1.4 Reproduktion

Reproduktion wird durch 1. Mutation und 2. Rekombination ausgeführt.

1. Mutation erfolgt durch das zufällige Kippen einer zufälligen Anzahl an Bits im Bitstring. Durch die gewählte Kodierung ist sichergestellt, das alle Folgenden Gelenke mitgedreht werden.
2. Rekombination wird durch das zufälligen Wählen eines Einsprungpunktes im Bitstring initiiert. Die beiden, ebenfalls zufällig gewählten, Individuen werden an den Entsprechenden Stellen gekappt. Ein neues Individuum wird dann geboren, bestehend aus einem Teil des ersten Individuums bis zum Einsprungpunkt und dem zweiten Teil des zweiten Individuums ab dem Einsprungpunkt.

Maximilian Mühlfeld, Informatik MA, 580070,
Malte Schmitz, Informatik MA,xxxxxx,
Eike von Tils, Informatik MA,xxxxxx

2 Funktionsbeschreibung

Im Folgenden wird auf die einzelnen Funktionen und Klassen eingegangen.

2.1 class Point

Die Klasse Point ist eine Hilfsklasse, die zur Repräsentation eines Punktes im 2D Raum mit den Koordinaten x und y dient.
Sie verfügt im Wesentlichen über folgende Methoden:

- +(other)
  Die Methode + addiert zwei Instanzen von Point. Hierbei wird jeweilts auf die Koordinaten x und y die der Übergebenen Instanz addiert, und eine neue Instanz zurückgegeben.
- rotate!(d)
  Rotiert die Koordinaten nach links (L) oder rechts (R). Die Rotation erfolgt hierbei jeweils um 90°
- .min(a,b) bzw. .max(a,b)
  Bestimmt die minimale bzw. maximale Lage eines Puntes im Raum bezüglich der Koordinatenwerte x und y.

class snake

Die Klasse snake dient zur Repräsentation einer Schlange auf einem 2D Gitter. Zusätzlich beinhaltet diese Klasse die implementierten Energiefunktionen.

- (to_ board)
  Hier wird versucht die Schlange auf ein 2D Gitter zu legen. Kommt es zu Überschneidungen, so wird nill zurückgegeben. Ansonsten das Board mit Schlange. Das zurückgegebene Board ist zudem minimal groß.
- (to_ String)
  Ersetzt die Kodierte Darstellung der Schlange (1 für Richtungsänderung, 0 für Gleichbleibend) durch die Richtungsangaben R und L.
- (energy_ a)
  Berechnet die Energiefunktion anhand der Größe des Boardes auf dem die Schlange liegt. Der Durchmesser des kleinsten Kreises ist identisch mit der Diagonalen des Boardes. Diese wird mithilfe des Satzes von Pythagoras berechnet und zurückgegeben.
- (energy_ b)
  Berechnet die Energiefunktion anhand der Anzahl der Löcher. Hierzu wird das Board um die Schlange herum mit einem Flooding Algorithmus gefüllt. Alle Stellen auf dem Board, die dann noch leer sind, sind Löcher die von der Schlange umschlossen sind. Die Anzahl dieser Stellen wird zurückgegeben.
- (energy_ c)
  Invertiert die Energiefunktion energy_ b, damit eine minimierung zu einer maximal großen Lochfläche führt.

2.2 class Evolution

Die Klasse Evolution implementiert die Funktionalität des Evolutionären Algorithmus.

- (start)
  Die Methode erzeugt eine zufällige Population von snakes mit vorgegebener Anzahl an Individuen.
- (crossover)
  Erzeugt neue Individuen durch Crossover. Hierzu wird ein zufälliger Einsprungpunkt gewählt an dem der Crossover stattfindet.
- (mutation)
  Erzeugt neue Individuen mit zufälligen Mutationen. Zuerst wird hierzu eine Bitmaske erstellt, die pro Bitstelle mit einer vorgegebenen Wahrscheinlichkeit eine 1 entählt. Diese wird dann mit der Kodierten Schlange xor genommen, so das ein bit gekippt wird, wenn die Maske 1 ist.

Maximilian Mühlfeld, Informatik MA, 580070,
Malte Schmitz, Informatik MA,xxxxxx,
Eike von Tils, Informatik MA,xxxxxx

- (step)
  Führt einen Kompletten Iterationsschritt aus. Durch Aufruf von crossover und mutation werden neue Individuen erzeugt.
  Anschließend werden diese nach ihrer Energie, gewichtet mit einer Zufallszahl, sortiert.
  Von den so sortierten werden die alle bis zu der vorgegebenen Populationsgröße gewählt.

- (iterate)
  Diese Methode startet die komplette Berechnung des Algorithmus. Sie ruft die start methode auf und dann in jeder Iteration einmal step. Sie iteriert solange, bis entweder die maximale Anzahl erreicht wurde, oder eine Energiegrenze unterschritten wurde.

- (bear)
  Erzeugt ein neues Individuum bestehend aus dem Kodierten Bitstring und dem Wert der Energiefunktion.

2.3 snake_ evolution

Das Skript erzeugt eine neue Schlange und setzt die Parameter des Algorithmus. Es startet anschließend den Algorithmus und gibt die Ergebnisse aus.
Die Parameter sind hierbei:

- length = Länge der Schlange
- energy = Zu benutzende Energiefunktion
- size = Größe einer Population
- crossover = Anzahl an crossover Schritten pro Iteration
- mutation = Anzahl an Mutationsschritten pro Iteration
- flip = Wahrscheinlichkeit das ein Bit bei der Mutation gekippt wird
- selection = Anteil der Zufallskomponente bei der Selektion nach Fitness der Individuen (0 bis 1)
- n = Maximale Anzahl an Iterationen
- energy = Maximale zu erreichende Energie bei der Minimierung
- logging = Flag ob Ausgabe erzeugt wird oder nicht.

3 Durchführung

Das Programm wird auf der Kommandozeile durch den Befehl "ruby snake_ evolution.rb" gestartet.
Die Ausgabe besteht aus der Anzahl an Iterationen, der endgültigen Population sowie einer Liste aller entstandenen Individuen (1 pro Energielevel). Zum Ende ist die Schlange auf ihrem 2D Gatter gezeichnet.

3.1 Energiefunktion a)

gewählte Parameter:

- length = 27
- energy = energy_ a
- size = 15
- crossover = 5
- mutation = 15
- flip = 0.3
- selection = 0.5

Maximilian Mühlfeld, Informatik MA, 580070,
Malte Schmitz, Informatik MA,xxxxxx,
Eike von Tils, Informatik MA,xxxxxx

- n = 10000
- energy = 9
- logging = true

```
number of iterations:
52

final population:
8.602325        1110110110010010        IILRLILRILLRILILLRRIRILILII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
9.899495        1111100101011010        IILRLIRLILLRIRILLRLILIRIRII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
9.899495        0011100101011010        IIRRLIRLILLRIRILLRLILIRIRII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
9.899495        1111100101011010        IILRLIRLILLRIRILLRLILIRIRII
9.899495        1111100101011010        IILRLIRLILLRIRILLRLILIRIRII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
10.000000       1011100111011011        IILLRILRIRRLIRILLRLILIRILII
14.212670       1111010101111010        IILRLIRRILLRIRILRLRIRILILII
14.866069       1011101111110101        IILLRILRIRLRILIRLRRILILIRII

all energies and first found individual with that energy:
8.602325        1110110110010010        IILRLILRILLRILILLRRIRILILII
9.899495        0111100101011010        IIRLRILRIRRLILIRRLRIRILILII
10.000000       0111101101101010        IIRLRILRIRLRIRILRRLILIRIRII
...
17.804494       0110101111101011        IIRLRIRLILRLIRILRRLILIRILII
Inf     1011000010010101        IILLRILLILLLIRIRRLLIRIRILII

fittest individual in final population:
8.602325        1110110110010010        IILRLILRILLRILILLRRIRILILII

  LL
LIRI
IRRRL
LLI I
LIRRL
IIIL
LIIX
```

3.2 Energiefunktion c (Logisch b)

gewählte Parameter:

- length = 27
- energy = energy_ c
- size = 15
- crossover = 5
- mutation = 15

- flip = 0.3
- selection = 0.5
- n = 10000
- energy = -15
- logging = true

```
number of iterations:
71

final population:
-15.000000    0010111101111011    IIRRLILRILRLILIRLRLILIRILII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
-10.000000    0110111101010010    IIRLRIRLIRLRIRILLRRIRILILII
-10.000000    0110111101010010    IIRLRIRLIRLRIRILLRRIRILILII
-10.000000    0110111101010010    IIRLRIRLIRLRIRILLRRIRILILII
-10.000000    0110111101010010    IIRLRIRLIRLRIRILLRRIRILILII
-10.000000    0110111101010010    IIRLRIRLIRLRIRILLRRIRILILII
-10.000000    0110111101010010    IIRLRIRLIRLRIRILLRRIRILILII
-10.000000    0110111101010010    IIRLRIRLIRLRIRILLRRIRILILII

all energies and first found individual with that energy:
-15.000000    0010111101111011    IIRRLILRILRLILIRLRLILIRILII
-13.000000    0110111101111011    IIRLRIRLIRLRIRILRLRIRILIRII
...
-1.000000     1101110100111110    IILRRILRILLRIRIRLRLIRILILII
0.000000      1011101011010011    IILLRILRIRLLIRILLRRIRILIRII

fittest individual in final population:
-15.000000    0010111101111011    IIRRLILRILRLILIRLRLILIRILII

  XIIL
IIR  I
 LR  RIL
 I     I
 LR   RL
  I  RL
  LR I
   LIL
```