Wykorzystanie sztucznej inteligencji w symulowaniu zachowań biologicznych

(Utilizing artificial intelligence to model biological behaviours)

Tymoteusz Kaczorowski

Praca inżynierska

Promotor: dr Jakub Kowalski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

9 lutego 2018 r.

Tymoteusz Kaczorowski

.....

.....
(adres zameldowania)

.....

.....
(adres korespondencyjny)

PESEL: .....

e-mail: .....

Wydział Matematyki i Informatyki
stacjonarne studia I stopnia
kierunek: informatyka
nr albumu: 273882

Oświadczenie
o autorskim wykonaniu pracy dyplomowej

Niniejszym oświadczam, że złożoną do oceny pracę zatytułowaną *Wykorzystanie sztucznej inteligencji w symulowaniu zachowań biologicznych* wykonałem/am samodzielnie pod kierunkiem promotora, dr Jakuba Kowalskiego. Oświadczam, że powyższe dane są zgodne ze stanem faktycznym i znane mi są przepisy ustawy z dn. 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tekst jednolity: Dz. U. z 2006 r. nr 90, poz. 637, z późniejszymi zmianami) oraz że treść pracy dyplomowej przedstawionej do obrony, zawarta na przekazanym nośniku elektronicznym, jest identyczna z jej wersją drukowaną.

Wrocław, 9 lutego 2018 r.

(czytelny podpis)

Streszczenie

Nie mam na razie pomysłu na streszczenie, niestety.

Yeah, not in English either

Spis treści

Rozdział 1.

Wstęp

W ramach tej pracy stworzyłem aplikację pozwalającą uruchomić i obserwować, jak sztuczna inteligencja uczy się grać w grę przedstawiającą uproszczony model środowiska naturalnego.

1.1. Motywacja

Założeniem projektu było stworzenie symulacji inspirowanej światem rzeczywistym, w której egzystują byty zdolne do nauczenia się strategii pozwalającej na optymalne funkcjonowanie. Celem było zbudowanie modelu reprezentującego pewne zjawiska obserwowane w naturze.

1.2. Realizacja

Na potrzeby realizacji celu powstały trzy elementy:

1. Środowisko graficzne służące do wizualnego reprezentowania i kontrolowania symulacji
2. Silnik symulacji, będący grą dla jednego gracza, w której Osiołek (gracz) ma za zadanie przeżyć jak najdłużej odżywiając się roślinami
3. Ucząca się sztuczna inteligencja kontrolująca Osiołka

1.3. Narzędzia

Projekt w całości został zrealizowany w języku C# z wykorzystaniem silnika WPF do części wizualnej.

Rozdział 2.

Zasady symulacji

Symulacja stworzona na potrzeby tej pracy to prosta, turowa gra dla jednego gracza rozgrywająca się na ciągłej, dwuwymiarowej, kwadratowej planszy, na której umieszczone są byty. Gra rozróżnia dwa rodzaje bytów: Osiołek i oset.

2.1. Byty

Wszystkie byty posiadają następujące cechy:

1. Pozycja na planszy
2. Identyfikator

2.1.1. Osiołek

Osiołek jest bytem kontrolowanym przez gracza. W danej turze na planszy znajduje się dokładnie jeden Osiołek. Osiołek posiada następujące cechy zdefiniowane w ustawieniach:

1. **Mass** - masa, która opisuje stopień wypełnienia żołądka i umożliwiająca Osiołkowi funkcjonowanie poprzez spalanie jej
2. **MovementSpeed** - maksymalny dystans, który może pokonać w ciągu tury
3. **InteractionDistance** - zasięg, na którym możliwe jest jedzenie ostu
4. **BiteSize** - maksymalna masa, którą może przyswoić z rośliny w ciągu tury
5. **StomachCapacity** - maksymalny rozmiar żołądka
6. **SightRange** - zasięg, w którym inne byty są dla Osiołka widoczne
7. **PassiveWork** - ilość masy zużywanej co turę

8. **MovementWork** - ilość masy zużywanej na potrzeby ruchu

2.1.2. Oset

Osiołek żywi się ostem. Na planszy może znajdować się co najwyżej określona w ustawieniach liczba roślin. Każdy oset ma nastepujące cechy:

1. **Mass** - określa ilość dostępnego pożywienia w danej roślinie
2. **MaxMass** - maksymalna masa, do której roślina może urosnąć
3. **RegrowthRate** - stały przyrost masy na turę

Maksymalna masa i przyrost masy na turę danej rośliny są losowane z rozkładem jednostajnym na przedziałach zdefiniowanych w ustawieniach.

2.2. Przebieg tury

Każda tura przebiega następująco:

1. Wszystkie rośliny, których obecna masa jest większa niż zero zwiększają swoją masę w oparciu o **RegrowthRate**, nie przekraczając **MaxMass**
2. Wszyskie byty, których masa nie przekracza zera umierają i znikają. Jeśli umiera Osiołek symulacja jest resetowana.
3. Osiołek otrzymuje opis otoczenia i swojego stanu
4. Kontroler wybiera akcję dla Osiołka w oparciu o ten opis
5. Osiołek wykonuje akcję i spala część swojej masy określoną wzorem:
   $\mathbf{PassiveWork} + \mathbf{MovementWork} * \frac{\text{przebyty dystans}}{\mathbf{MovementSpeed}}$
6. Jeżeli liczba roślin na planszy nie przekracza maksymalnej pojawia się w losowym miejscu nowa roślina, nie częściej niż raz na liczbę tur opisaną w ustawieniach

2.3. Akcje

W każdej turze Osiołek może wykonać jedną z poniższych akcji:

1. Pominięcie tury

2. Próba zjedzenia wybranej rośliny, która przekazuje część jej masy Osiołkowi jeśli ten znajduje się w odpowiednim zasięgu. Ilość zaabsorbowanej masy wynosi

   $min(\textbf{BiteSize}, \textbf{plant.Mass}, \textbf{StomachCapacity - donkey.Mass})$

3. Poruszenie się w stronę zadanego punktu o wybraną część maksymalnego dystansu

Rozdział 3.

Aplikacja

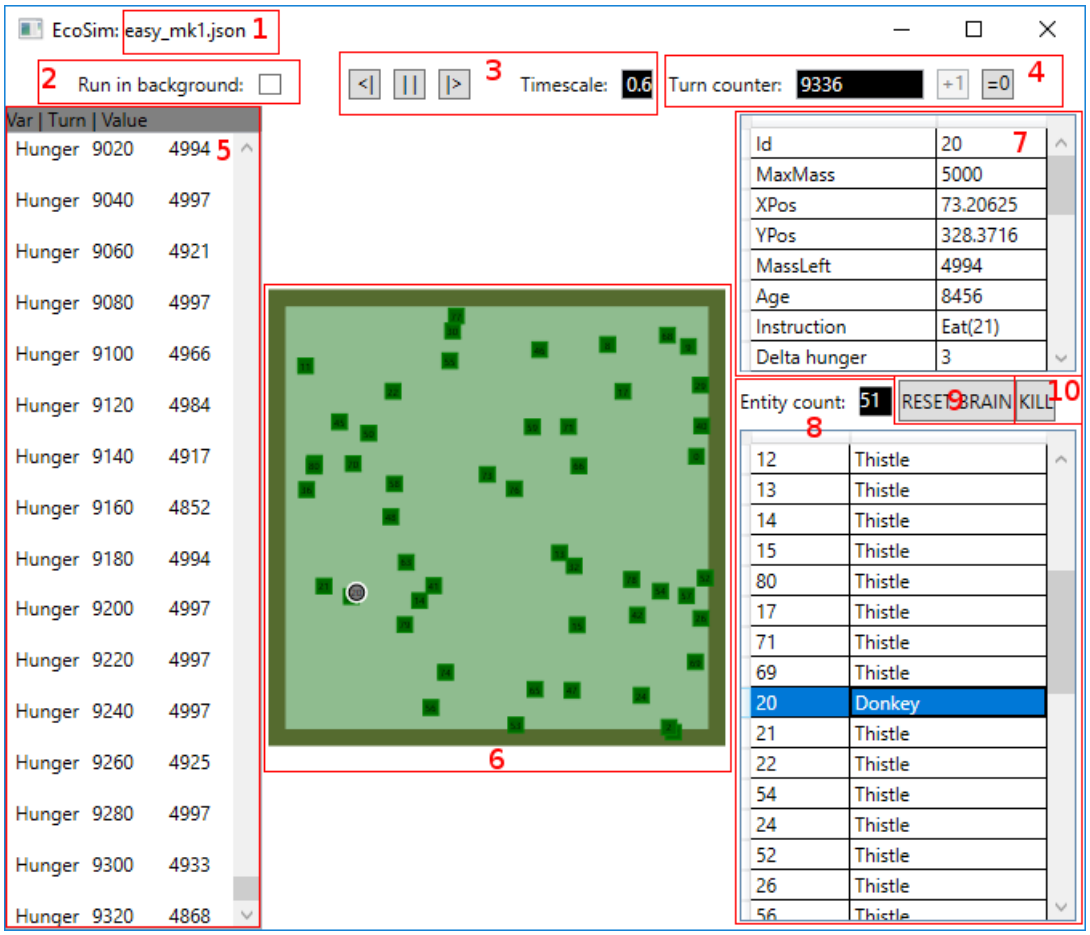
Aplikacja udostępnia prostą wizualizację oraz metody kontrolowania i podglądu stanu symulacji. Ten rozdział opisuje interfejs, obsługę i działanie aplikacji.

3.1. Działanie aplikacji

Po uruchomieniu użytkownik proszony jest o wybranie pliku .json z ustawieniami. Następnie tworzona jest instancja kontrolera Osiołka i uruchamiane są zadania cykliczne: odświeżanie grafiki i obliczanie kolejnych tur symulacji. Program przyjmuje jeden, opcjonalny argument wywołania, liczbę całkowitą określającą liczbę odświeżeń grafiki na sekundę. Wartość domyślna wynosi 20.

Gdy nastąpi śmierć Osiołka symulacja jest resetowana, ale licznik tur i kontroler zachowują swój stan.

3.2. Opis interfejsu



1. Nazwa wybranego pliku z ustawieniami
2. Checkbox pozwalający wyłączyć odświeżanie grafiki i obliczać kolejne tury symulacji tak szybko, jak jest to możliwe.
3. Przyciski kontrolujące częstotliwość obliczania kolejnych tur symulacji.
4. Licznik tur. Przycisk *+1* pozwala obliczyć kolejną turę symulacji gdy **Timescale** jest równe 0. Przycisk *=0* resetuje licznik tur.
5. Log, w którym pojawiają się wiadomości tworzone przez symulację. W przykładzie jest to bieżąca tura i stan żołądka Osiołka wypisywane co 20 tur.
6. Plansza reprezentująca stan symulacji.
7. Szczegóły obecnie wybranego bytu. Byt można wybrać klikając go na planszy lub na liście bytów.
8. Lista bytów znajdujących się na planszy
9. Przycisk pozwalający wywołać metodę *Reset()* na kontrolerze Osiołka

10. Przycisk pozwalający natychmiastowo zabić obecną instancję Osiołka.

3.3. Ustawienia

Symulacja oraz kontroler są parametryzowane plikiem .json. Poniżej przedstawiam przykładowy plik z ustawieniami oraz ich opisami:

```
{
    "Brain": "Mark1", //"Mark1" lub "Mark0", wybiera rodzaj
       kontrolera

    "LogDeaths" : false, //wartosc true sprawia, ze po kazdej
       smierci Osiolka zalogowana zostanie tura jego urodzin i
       wiek w momencie zgonu

    "LogHungerEvery": 20, //loguje stan zoladka Osiolka co kazde
       LogHungerEvery tur. Wartosc 0 wylacza te opcje.

    "InitialPlantCount": 20, //liczba roslin pojawiajaca sie na
       poczatku symulacji

    "MaxPlantCount": 50, //maksymalna liczba roslin na planszy

    "NewPlantFrequency": 5, //liczba tur miedzy pojawieniami sie
       nowych roslin

    "MinPlantMass": 100, //dolna granica przedzialu, z ktorego
       losowana jest maksymalna masa rosliny

    "MaxPlantMass": 500, //gorna granica w/w przedzialu

    "MinGrowthRate": 30, //dolna granica przedzialu, z ktorego
       losowane jest RegrowthRate rosliny

    "MaxGrowthRate": 50, //gorna granica w/w przedzialu

    "MovementSpeed": 20, // opisane w rozdz. 2 pracy

    "InteractionDistance": 22, // opisane w rozdz. 2 pracy

    "BiteSize": 100, // opisane w rozdz. 2 pracy

    "StomachCapacity": 5000, // opisane w rozdz. 2 pracy

    "SightRange": 88, // opisane w rozdz. 2 pracy

    "PassiveWork": 3, // opisane w rozdz. 2 pracy

    "MovementWork": 5, // opisane w rozdz. 2 pracy

    "BrainBaseActionScore": 0.0022, // opisane w rozdz. 4 pracy

    "BrainDiscount": 0.6, // opisane w rozdz. 4 pracy

    "BrainLearningRate": 0.148, // opisane w rozdz. 4 pracy
```

```
        "BrainLearningRateDamping": 0.99999, // opisane w rozdz. 4
            pracy

        "BrainBaseActionScoreDamping" : 0.99952, // opisane w rozdz.
            4 pracy

        "BrainProbabilityExponent": 1.6 // opisane w rozdz. 4 pracy
    }
```

Rozdział 4.

Sztuczna inteligencja

Poczynaniami Osiołka kieruje kontroler implementujący algorytm Q-learning. W tym rozdziale opiszę szczegóły implementacji kontrolera.

4.1. Rodzaje kontrolerów

Zaimplementowane zostały dwa rodzaje kontrolerów, przezwane *Mark1* i *Mark0*. Różnią się jedynie liczbą stanów, które rozróżniają - *Mark0* jest "krótkowzroczny"i nie analizuje pożywienia poza zasięgiem interakcji.

4.2. Kwantyzacja stanu

Kontroler w każdej turze symulacji otrzymuje opis stanu Osiołka:

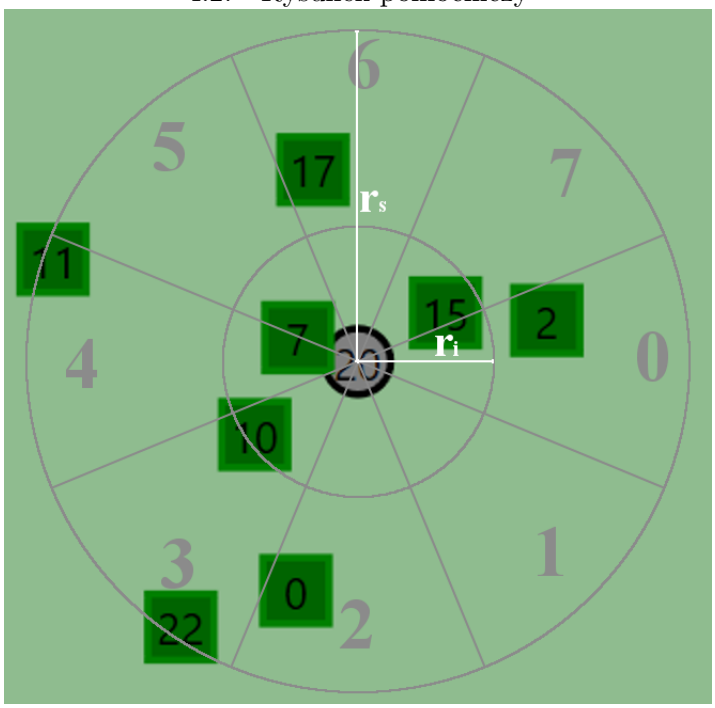
1. **Hunger** - liczba z zakresu 0-1 równa $1 - \frac{\textbf{donkey.Mass}}{\textbf{donkey.MaxMass}}$
2. Pozycję Osiołka
3. Pozycje i masy wszystkich roślin w zasięgu wzroku Osiołka

Na potrzeby działania algorytmu stan ten sprowadzany jest do liczby 19-bitowej (11-bitowej dla kontrolera *Mark0*). Składa się na nią kolejno:

1. **Penalty** - liczba 3-bitowa równa $\lfloor \textbf{Hunger} * 7 \rfloor$
2. **FoodRichDirections** - liczba 8-bitowa wyznaczająca kierunki bogate w pożywienie (tylko w kontrolerze *Mark1*)
3. **CloseFoodDirections** - liczba 8-bitowa wyznaczająca kierunki, w których pożywienie znajduje się w zasięgu interakcji

Kontroler rozróżnia osiem kierunków. Kolejne bity liczb **FoodRichDirections** i **CloseRichDirections** wyznaczają kolejne kierunki.

4.2. - Rysunek pomocniczy



Niech:

- r_i oznacza zasięg interakcji
- $$L(a,b) = \begin{cases} 1, & \text{gdy } a \leqslant b \\ 0, & \text{w p. p.} \end{cases}$$
- $P_{k,0}...P_{k,n_k}$ reprezentuje wszystkie rośliny w zasięgu wzroku (r_s) w kierunku k, $k = 0...7$.
- $M(P)$ oznacza masę danej rośliny P
- $D(P)$ oznacza odległość rośliny P od Osiołka
- $C_k = \sum_{i=0}^{n_k} M(P_{k,i}) \cdot L(D(P_{k,i}), r_i)$
- $C = \sum_{d=0}^{7} C_d$
- $R'_k = \sum_{i=0}^{n_k} \frac{M(P_{k,i})}{D(P_{k,i}) - r_i + 1} \cdot (1 - L(D(P_{k,i}), r_i))$
- $R_k = \frac{R'_{(k-1) \mod 8} + R'_{(k+1) \mod 8}}{2} + R'_k$
- $R = \sum_{d=0}^{7} R_d$

Wtedy:
CloseFoodDirections = $\sum_{d=0}^{7} 2^d \cdot L(\frac{C}{9}, C_d)$
FoodRichDirections = $\sum_{d=0}^{7} 2^d \cdot L(\frac{R}{9}, R_d)$

Czyli bit **CloseFoodDirections** jest równy 1, jeśli w wyznaczanym przez niego kierunku znajduje się więcej jedzenia w zasięgu interakcji niż przeciętnie.

Bit **FoodRichDirections** jest równy 1, jeśli w wyznaczanym przez niego kierunku, znajduje się poza zasięgiem interakcji (po przeskalowaniu względem odległości i uwzględnieniu kierunków sąsiednich) więcej jedzenia niż przeciętnie.

4.3. Akcje

Na potrzeby algorytmu akcje także są uproszczone. Na akcję składają się:

- Rodzaj - pominięcie tury, ruch, jedzenie
- Kierunek - liczba od 0 do 7

Akcja taka jest jest po wybraniu tłumaczona na instrukcję w rozumieniu silnika symulacji. Jedzenie obiera na cel najbliższą roślinę w danym kierunku, zaś ruch obiera kierunek odpowiadający zadanemu, dla uproszczenia odchylany w stronę najbliższej rośliny w tym kierunku.

4.4. Parametry kontrolera

Parametry kontrolera definiowane są w pliku .json z ustawieniami. Są to:

1. **ProbabilityExponent** - zwiększa różnicę w prawdopodobieństwie między najlepszymi a najgorszymi akcjami
2. **StartLearningRate** - początkowa wartość parametru α, który wpływa na skalę zmian wprowadzanych do stanu kontrolera po każdej decyzji
3. **LearningRateDamping** - zmniejsza wartość parametru α kontrolera z czasem poprzez mnożenie go przez tę liczbę w każdej turze
4. **StartBaseActionScore** - początkowy **BaseActionScore**, który zapewnia niezerowe prawdopodobieństwo wszystkim możliwym akcjom
5. **BaseActionScoreDamping** - zmniejsza faktyczny **BaseActionScore** kontrolera z czasem poprzez mnożenie go przez tę liczbę w każdej turze
6. **Discount** - wartość parametru γ, który wpływa na balans między rozważaniem natychmiastowej nagrody płynącej z akcji i jej konsekwencji

4.5. Nauka

Kontroler wykorzystuje prosty Q-Learning oparty o tabelkę definiującą funkcję Q, która każdej parze (stan,akcja) przyporządkowuje pewną wartość, która jest modyfikowana w czasie nauki. Na początku funkcja ta ma wartość 0 dla wszystkich argumentów.

Przyjmijmy, że w turze t Osiołek znajdował się w stanie S_t, jego **Hunger** wynosił H_t i podjął akcję A_t. Doprowadziło go to do stanu S_{t+1}, w którym **Hunger** wynosi H_{t+1}. Niech $\Delta H = H_t - H_{t+1}$, $\alpha_t = \textbf{StartLearningRate} \cdot \textbf{LearningRateDamping}^t$ Wtedy:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) \cdot (1 - \alpha_t) + \alpha_t \cdot (\Delta H + \gamma \cdot max_a Q_t(s_{t+1}, a))$$

Dla wszystkich (s, a) różnych od (s_t, a_t) $Q_{t+1}(s, a) = Q_t(s, a)$

4.6. Podejmowanie decyzji

Po zaktualizowaniu funkcji Q wybierana jest akcja do podjęcia w danej turze. Każda akcja ma pewne pradopodobieństwo zostania wybraną, zależne od wartości funkcji Q. Niech $\beta_t = \textbf{StartBaseActionScore} \cdot \textbf{BaseActionScoreDamping}^t$. Wtedy prawdopodobieństwo akcji a w stanie s_t wynosi:

$P_t(s_t, a) = \frac{P'_t(s_t,a)}{\sum_{a'} P'_t(s_t,a')}$, gdzie

$P'_t(s_t, a) = (Q_t(s_t, a) - min_{a'} Q_t(s_t, a') + \beta_t)^{\textbf{ProbabilityExponent}}$

Parametr β zapewnia pewne prawdopodobieństwo każdej możliwej akcji, co umożliwia eksplorację możliwości. **ProbabilityExponent** zwiększa szanse wybrania dobrych akcji zmniejszając szanse wybrania złych.

Rozdział 5.

Wnioski

asdf

Bibliografia

[1] test test entry, 2017