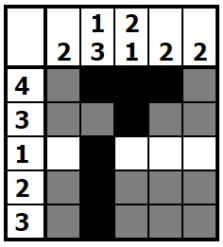
Algorytm Q-learning

Celem algorytmu jest nauczenie agenta próbującego rozwiązać łamigłówkę jak najszybszego dojścia do rozwiązania (w najmniejszej liczbie kroków).

Stan
Stan to konkretna plansza X x Y z ograniczeniami na górze i z lewej strony oraz z tablicą pól oznaczonych jako zamalowane, puste lub nieznane. Zmiana stanu po wykonaniu akcji to zaktualizowany wiersz lub kolumna. Stan wiersza będzie definiowany jako ciąg liczb oznaczających długość bloków, jakie w danym wierszu mają się znaleźć oraz tablica pól z wartościami: wypełnione, puste lub nieznane. Oprócz tego każdy wiersz jest oznaczony jako ukończony lub nie.

Stan początkowy
Wszystkie pola są oznaczone jako nieznane.

Akcja
Ograniczy się do wyboru wiersza lub kolumny, która w danym kroku będzie aktualizowana (maksymalna liczba możliwych akcji: X + Y). Aktualizacja wiersza polega na zmianie tych nieznanych pól w wypełnione lub puste, co do których jest pewność że właśnie takie muszą być. Polega to więc na wykorzystaniu wszystkich możliwości wypełnienia wiersza zgodnie z zasadami łamigłówki i sekwencją wejściową, jaka została wypełniona do tej pory, a następnie zaktualizowaniu tych pól, które we wszystkich rozwiązaniach miały ten sam kolor. W przypadku kiedy dany wiersz zawiera tylko pola ustalone, akcje na tym wierszu nie będą więcej wykonywane, jest on oznaczony jako ukończony. Jeszcze jedno założenie co do wyboru akcji to nie wybieranie tej samej akcji 2 razy z rzędu. Dzięki temu zwiększa to szanse na wybór akcji nieco mniej atrakcyjnych. Przykładowy stan planszy po wykonaniu 4 akcji przedstawiono na rysunku 4.



Rysunek 4. Przykładowa plansza po wykonaniu akcji kolejno na: 2 kolumnie, 1 wierszu, 3 wierszu i 3 kolumnie (pola czarne – wypełnione, białe – puste, szare – nieznane).

Nagroda
Przyznawana jest dla akcji wykonanej na wybranym wierszu lub kolumnie. Wartość od 0 do 1 obliczana wg wzoru: $r_{s,\ a} = \frac{k}{u}$, gdzie u to liczba nieznanych pól w danym wierszu przed wykonaniem akcji, a k to liczba nowych znanych pól, które zostały zaktualizowane po wykonaniu akcji. Wiersz nie może być wybrany, jeśli wszystkie jego pola zostały już „odkryte”. Dla przykładu z rysunku 4, nagroda za wykonanie pierwszej akcji w stanie początkowym (aktualizacja 2 kolumny) to 1, ponieważ wszystkie pola kolumny zostały ustalone.

Proces uczenia
Łamigłówki do procesu uczenia były generowane automatycznie. Do nauki każdego rozmiaru planszy wykorzystano 30 różnych wygenerowanych łamigłówek. Nauka każdej z nich przebiegała w

10 epizodach. Podczas trybu uczenia na początku wykonywane są akcje, w których kolumny są „pewne”, czyli wszystkie pola można od razu wypełnić. Następnie w każdym stanie jest losowo wybierana akcja (czyli aktualizacja losowego wiersza lub kolumny) i obliczana jest przybliżona wartość Q dla wybranej akcji w danym stanie.
Wybrałam 3 cechy charakteryzujące kolumnę:
f1 – liczba znanych pól / liczba wszystkich pól
f2 – suma ograniczeń / liczba wszystkich pól
f3 – liczba ograniczeń / liczba możliwych ograniczeń

Wzór na Q uwzględniający aproksymację liniową:

$$Q(s,a)=w1f1(s,a)+w2f2(s,a)+w3f3(s,a)$$

Nauka algorytmu polega na aktualizowaniu wektora wag w = [w1, w2, w3] według poniższego wzoru :

$$w \leftarrow w+\alpha(r+\gamma * \max a' Q(s',a') - Q(s,a)) \nabla^{*} wQ(s,a)$$

Gdzie $\nabla^{*} wQ(s,a) \nabla \rightarrow wQ(s,a)$ jest gradientem Q(s,a) w wektorze w.

$(r+\gamma * max'aQ(s',a')) - (Q(s,a))$ to błąd pomiędzy oszacowanym Q(s,a) i aktualną wartością Q(s,a), która wynika z otrzymanej nagrody i spodziewanej zdyskontowanej nagrody $\gamma * max'aQ(s',a')$

Pozostałe użyte parametry:
$\alpha = 0.7$
$\gamma = 0.9$

Napotkałam pewien problem z dobraniem odpowiednich wartości fi w stosunku do nagrody r tak aby następowała odpowiednia aktualizacja wektora wag. Okazało się, że kiedy f były sporo większe, niż r, zdarzało się, że wagi otrzymywały ujemną wartość. Jednak w większości przypadków wi uzyskiwały wartości jakich bym oczekiwała. Największa zazwyczaj była wartość w1, natomiast najmniejsza w3.

Po wykonaniu założonej liczby epizodów na planszach do uczenia nastąpiła faza testów. Agent dostaje planszę testową. Zaczynając od stanu początkowego wybiera akcje, które prowadzą do osiągnięcia stanów z najwyższą wartością przewidywaną Q, w ten sposób przechodzi do rozwiązania.

Do implementacji programu został wykorzystany język Python w wersji 3.5.3. Użyłam bibliotek random (do generowania plansz), copy oraz matplotlib (do generowania wykresów).

Kod programu znajduje się w pliku q_learning.py

Raport z testów

Testy przeprowadzałam po trybie nauki trwającym 10 epizodów na 30 planszach tej samej wielkości, co test.

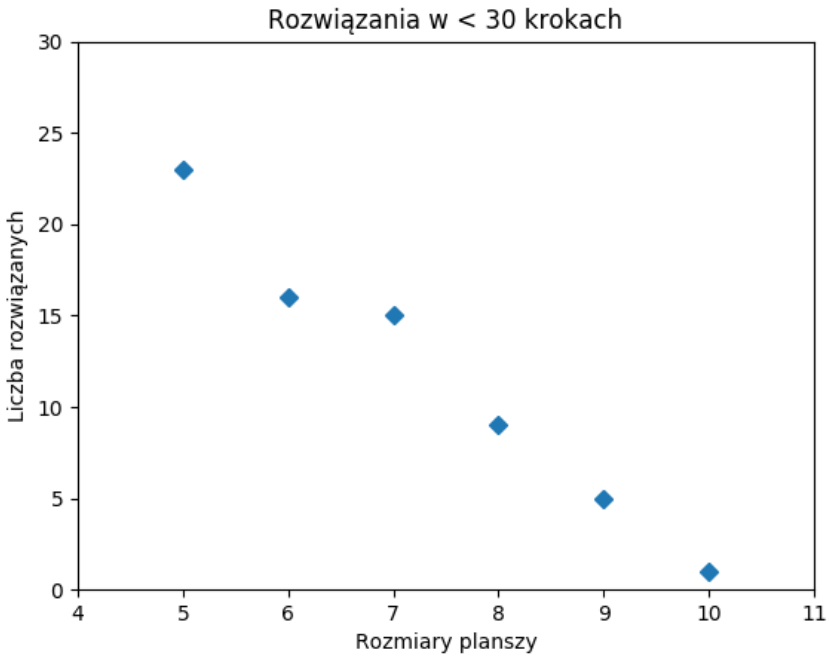
Sprawdziłam rozwiązanie łamigłówek wymienionych w dokumentacji na temat algoytmu A*.
1) budzik – rozwiązano w 27 krokach
2) balon – rozwiązano w 18 krokach
3) nuta – rozwiązano w 39 krokach
4) mysz – rozwiązano w 24 krokach

Krok to dla mnie wybór wiersza do aktualizacji, nic więc dziwnego, że moich kroków jest więcej niż w algorytmie A*.

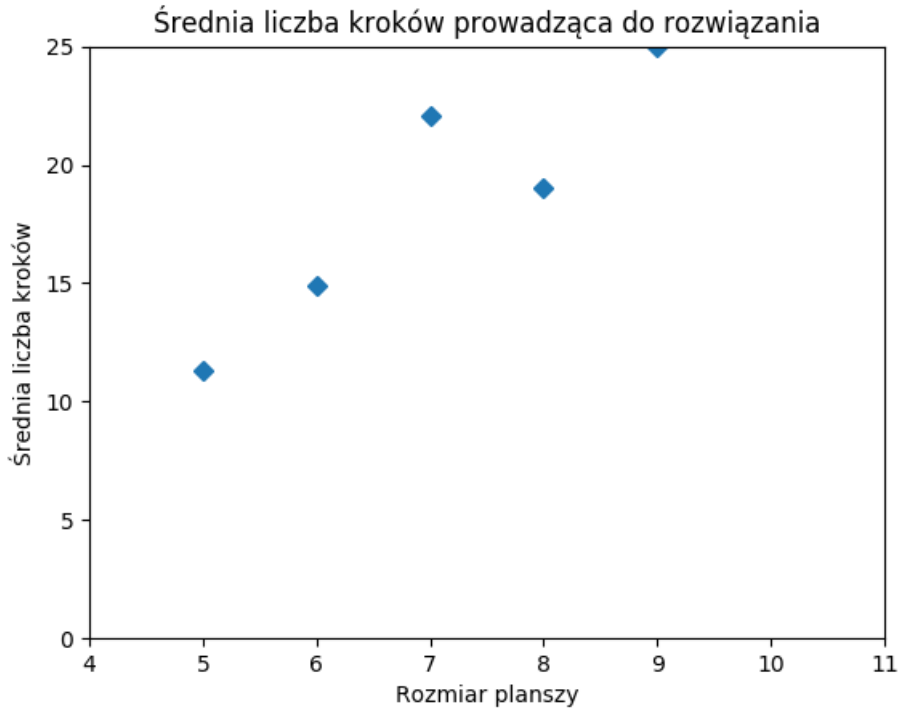
Następnie przeszłam do testowania plansz wygenerowanych automatycznie. Ponieważ okazały się one zdecydowanie trudniejsze od plansz wymienionych wyżej, wyniki są gorsze.

Poniżej przedstawiam wykresy pokazujące rezultaty różnych testów. Wykresy zostały

Pierwszy z nich prezentuje liczbę plansz rozwiązanych w mniej niż 30 krokach na 30 plansz testowanych. Liczba ta waha się w okolicach 25 dla plansz 5x5. Nie rozwiązanie pozostałych plansz świadczy o tym, że algorytm jest wrażliwy na plansze, które mają mało ograniczeń, lub ograniczenia mają małe wartości. Niektórych plansz po prostu nie da się rozwiązać stosując takie podejście do aktualizacji planszy, jakie wymyśliłam.



Poniżej zamieszczam wykres średniej liczby kroków wykonanych w rozwiązywaniu planszy w zależności od rozmiarów planszy. Przykładowo dla plansz 5x5 liczba kroków wahała się mniej więcej pomiędzy 5 a 15.



Postanowiłam przetestować zachowanie algorytmu dla plansz 10x10.
Wykres poniżej przedstawia liczbę prawidłowych rozwiązań planszy 10x10 w zależności od limitu maksymalnej liczby iteracji narzuconych na solver.

Ostatni wykres przedstawia średnią liczbę wykonanych kroków podczas rozwiązywania łamigłówek o rozmiarach 10x10. Największa liczba kroków nie przekraczała 100, co pokazuje, że algorytm dużo lepiej radzi sobie z większymi łamigłówkami.