Alicja Kubiszyn, 318685

Algorytm Q-learning

1. Opis algorytmu

Zaimplementowany algorytm to algorytm q-learning z epizodami, czyli q-learning z dużą ilością niezależnych epizodów/prób. Parametrami w tym algorytmie są: emax – maksymalna liczba epizodów, tmax – maksymalna liczba iteracji w jednym epizodzie, gamma – współczynnik dyskontowania, beta – współczynnik uczenia, epsilon – wykorzystywany przy wyborze akcji – im większy epsilon tym większe prawdopodobieństwo, że akcja zostanie wybrana losowo, w przeciwnym przypadku będzie wybrana ta, dla której wartość w tablicy Q_table jest największa , n_states i n_actions – odpowiednio liczba stanów oraz liczba akcji.
Na początku inicjowana jest tablica Q_table wypełniona zerami o rozmiarze n_states x n_actions. Następnie w pętli przeprowadzane są kolejne próby/epizody. Resetuje się środowisko oraz ustawiany jest początkowy stan. Następnie znajduje się właściwa, główna pętla algorytmu. Najpierw wybierana jest akcja, potem generowane są: next_state, reward (następny stan oraz nagroda w tym stanie) i zmienna done, która określa, czy zadanie zostało wykonane. Następnie aktualizowana zostaje tablica Q_table:

Q(state, action) = (1 – beta) * Q(state, action) + beta * (reward + gamma * max(Q(next_state, all actions)))

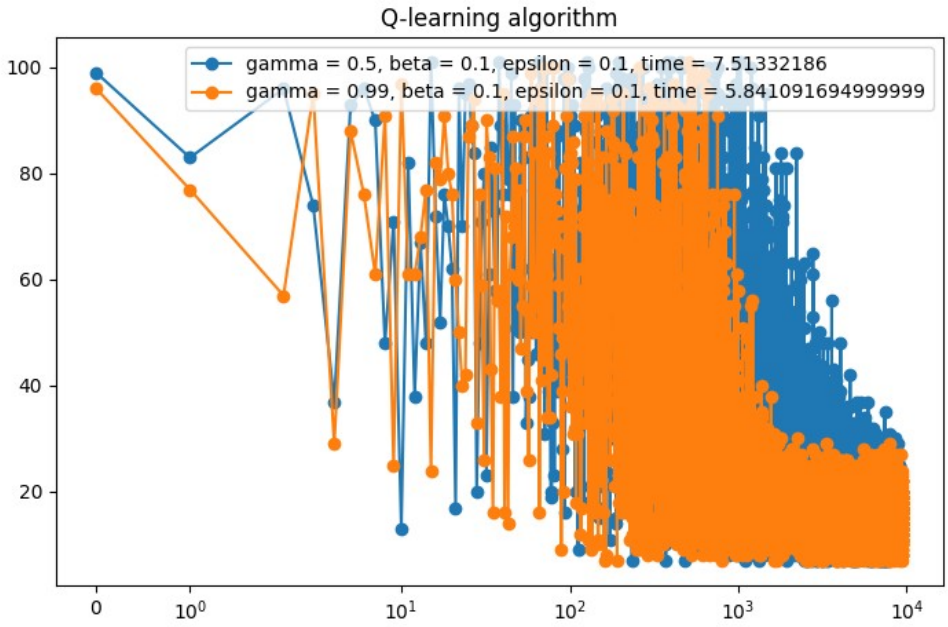
Oprócz zwiększenia licznika t oraz zastąpienia aktualnego stanu następnym, sprawdzany jest warunek czy zmienna typu bool, done jest równa True, jeśli tak to pętla zostaje przerwana.

W funkcji check_agent zostaje sprawdzony ten algorytm poprzez wygenerowanie animacji.

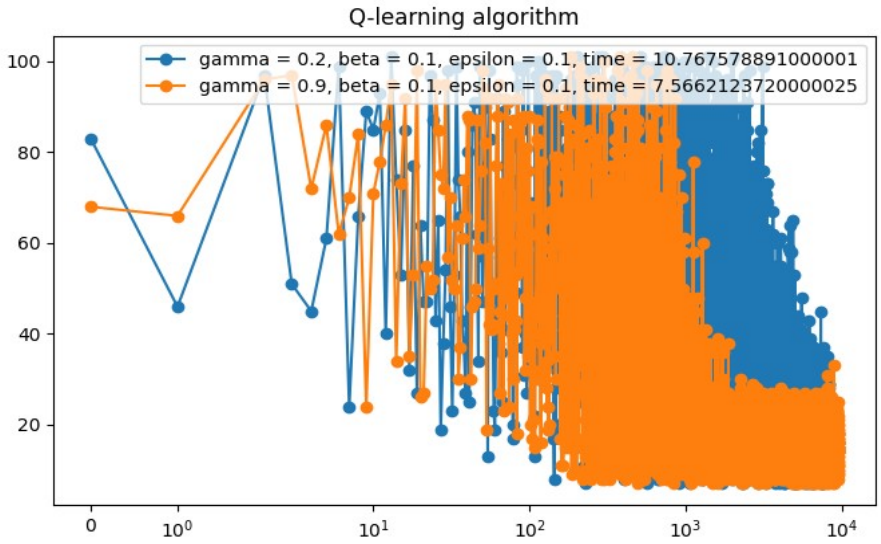
2. Opis planowanych eksperymentów

Zmieniane będą następujące parametry – parametr gamma, beta oraz epsilon. Podczas wykonywania się algorytmu, dla każdego epizodu liczona jest liczba kroków, po jakich nastąpiło przerwanie pętli – czyli znalezienie celu. Epizodów za każdym razem będzie 10^4. Oprócz tego będzie także liczony czas wykonywania się algorytmu. Będą porównywane do siebie algorytmy, w których różni się tylko jeden parametr.

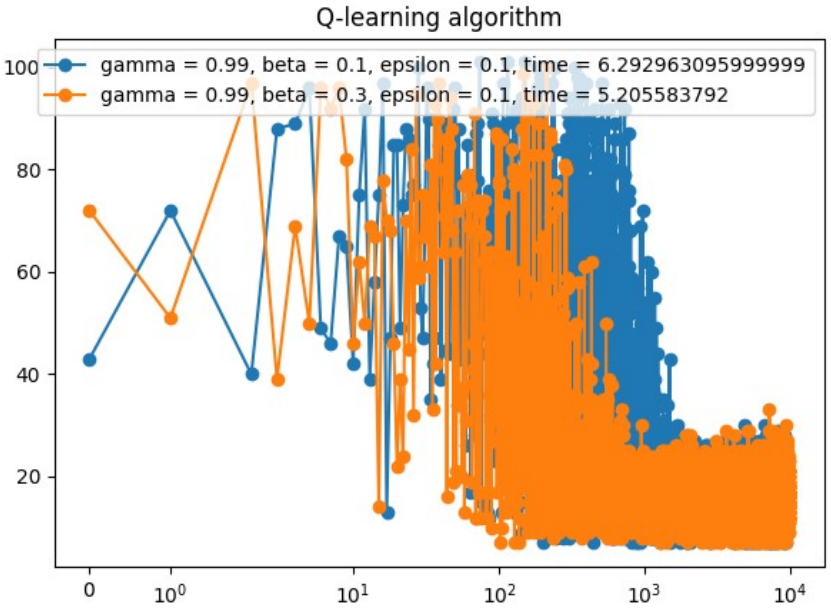
3.Analiza wyników



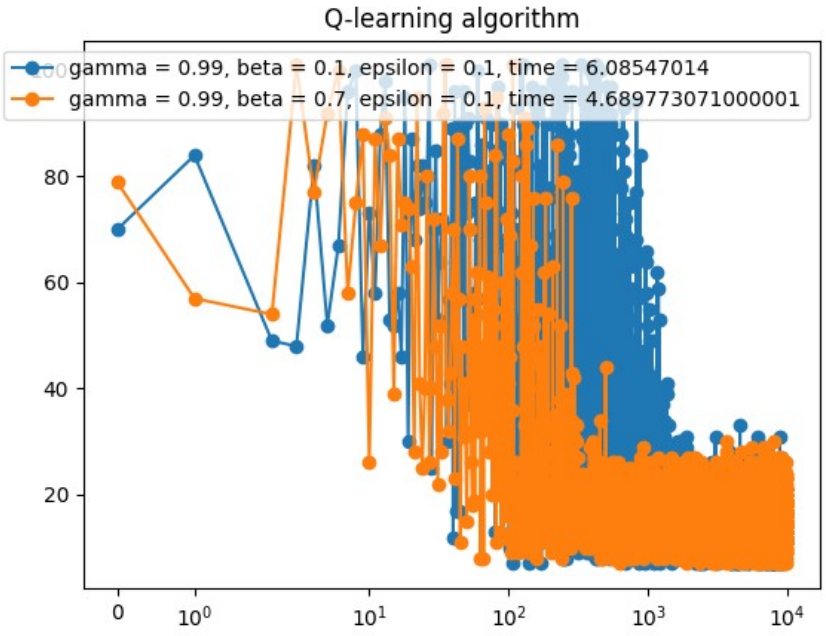
Mniejszy o połowę współczynnik gamma radzi sobie gorzej – czas jest o prawie 2 sekundy dłuższy oraz później dochodzi do celu. Jednak w dwóch przypadkach algorytm działa poprawnie.



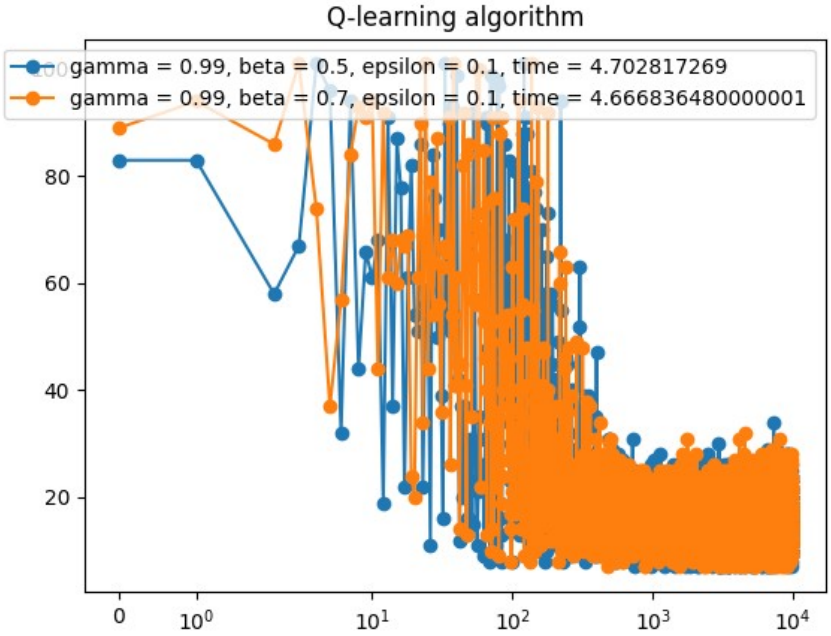
W powyższym przypadku algorytm ze współczynnikiem gamma radzi sobie jeszcze gorzej – czas jest wydłużony oraz ilość kroków nie zmniejsza się przez bardzo długi czas.



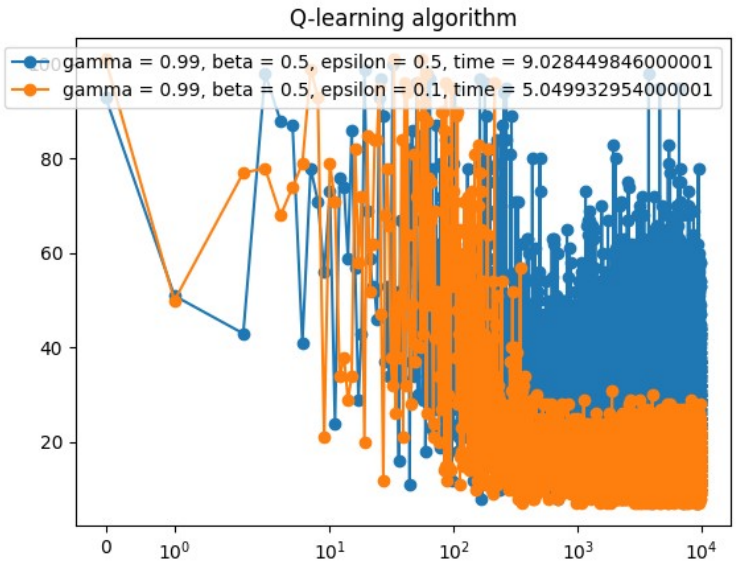
W wypadku niedużego zwiększenia współczynnika beta, czas algorytmu się poprawił oraz algorytm szybciej znajdował cel.



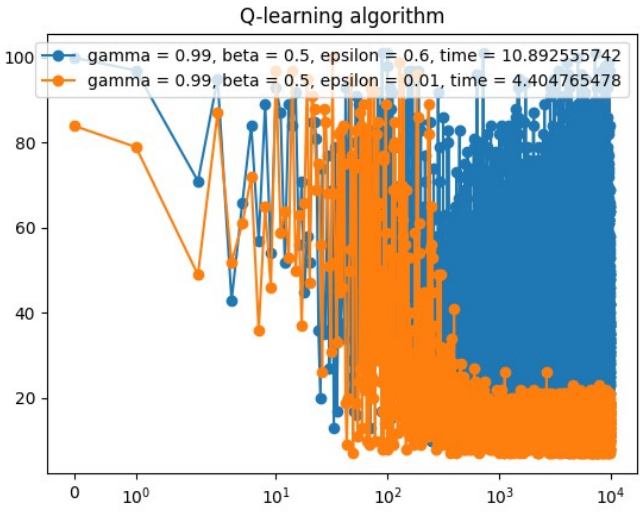
W wypadku jeszcze większego współczynnika beta, sytuacja jest podobna, czas algorytmu jeszcze się skrócił.



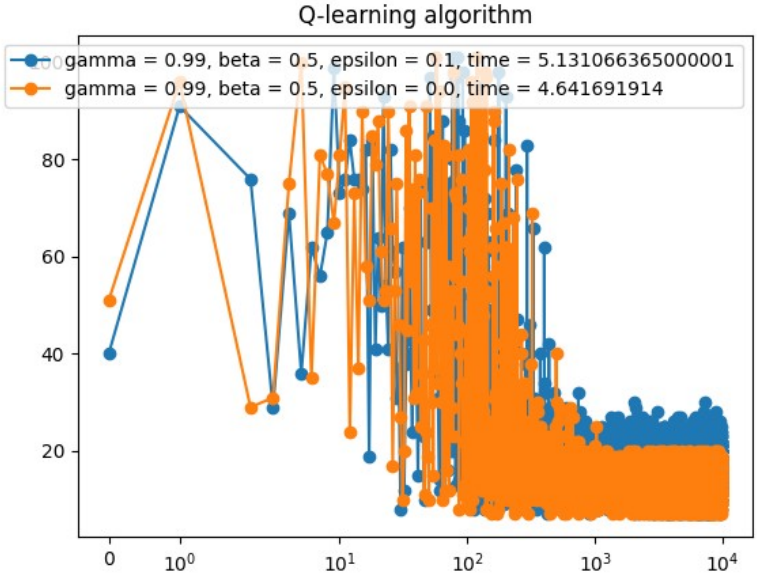
Porównując oba współczynniki beta (0.5 i 0.7) nie widać zbyt dużej różnicy w czasie wykonania algorytmu.



W przypadku zwiększenia epsilona można zaobserwować znaczące pogorszenie się algorytmu – czas się wydłużył o niemal połowę oraz algorytm do samego końca nie znajduje szybko celu.

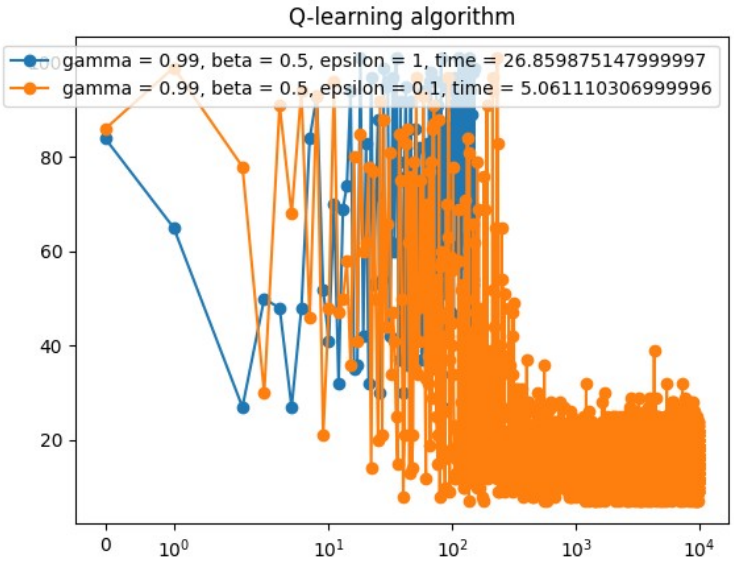


Jeszcze gorzej jest w przypadku jeszcze większego epsilona, co widać na wykresie. Zwiększając epsilon zwiększa się eksploracja algorytmu.



W przypadku zerowego epsilona – wybieramy akcję tylko za pomocą funkcji maks. Oznacza to, że zmniejsza się eskploracja, zwiększa się eksploatacja.

Co się stanie w przypadku epsilona równego 1?



Czas się znacząco wydłużył – aż o 5 razy.

4.Wnioski

Algorytm najlepiej działa dla jak największego współczynnika gamma. Działa on również lepiej dla większych współczynników beta, jednak nie ma zbyt dużej różnicy między wartością współczynnika 0.5 i 0.7. Parametr epsilon wpływa nam na zdolność eksploracji i eskploatacji algorytmu. Im większy tym większa eksploracja, jednak czas się znacząco wydłuża. Z przedstawionych eksperymentów wynika, że bardziej korzystne jest ustawienie bardzo niskiego epsilonu – wtedy algorytm działa szybciej, bo szybciej znajduje cel.