Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Uczenie Maszynowe

Dokumentacja Końcowa Projektu – Kolejka Górska

Michał Kwarciński
Tomasz Żebrowski

Warszawa, 2023

1. Przypomnienie dokumentacji wstępnej i założeń

Opis projektu

Celem projektu jest stworzenie agenta sterującego wagonikiem kolejki górskiej z wykorzystaniem technik uczenia ze wzmocnieniem. Zadaniem wagonika jest pokonywanie toru o zmiennym nachyleniu z zachowaniem ograniczeń prędkości. Zmienną sterującą jest siła wywierana przez silnik wagonika – może służyć do przyspieszania lub hamowania.

Wykorzystywać będziemy uczenie ze wzmocnieniem, zatem wyszczególnić należy środowiska, agenta uczącego się, oraz metodę oceny jakości agenta (krytyka).

Środowisko

Głównymi elementami środowiska są wagonik oraz tor, po którym ma on jeździć. Tor ma zmienne nachylenie, co ma zapewnić rozrywkę pasażerom wagonika. Na wagonik oddziałują: siła ciężkości, siła oporów ruchu oraz siła ciągu (którą możemy sterować). Ze względu na konstrukcję urządzeń, zakładamy brak możliwości wypadnięcia wagonika z zakrętu lub inne oderwanie od toru. Agent sterujący wagonikiem jest oceniany przez środowisko z uwzględnieniem następujących kryteriów:

— Nagroda za jazdę – kumuluje się wraz z pokonywaniem odległości przez wagonik. Dzięki temu wzmacniane będą strategię zapewniające ciągłe jechanie do przodu.
— Nagroda za prędkość zbliżoną do ustalonej prędkości optymalnej
— Kara za upływ czasu – mała, ale kumuluje się. Ma zapewnić dążenie agentów do szybkiego pokonywania toru.
— Kara za cofanie
— Kara za przekroczenie prędkości

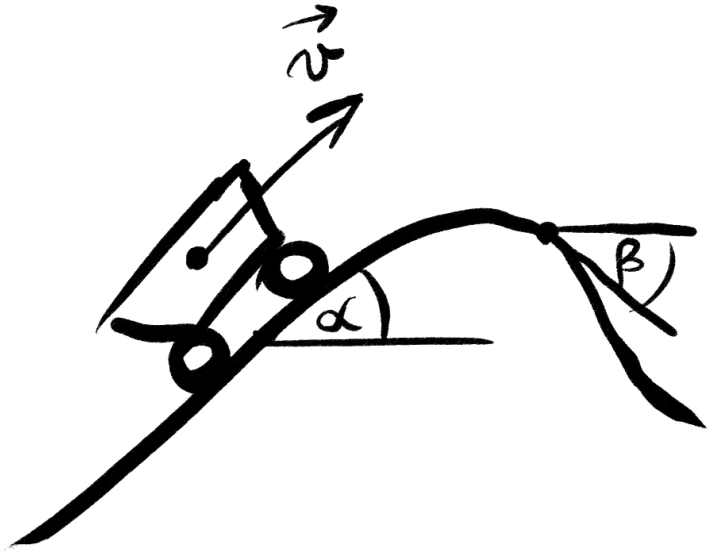
W przypadku przekroczenia limitu czasu próba jest kończona z możliwością dodania dodatkowej kary agentowi.

Opis algorytmów

Wykorzystywać będziemy algorytm Q-Learning. Stan agenta opisują:

— Prędkość wagonika v
— Nachylenie toru pod wagonikiem α
— Nachylenie toru przed wagonikiem β

Rozważaliśmy alternatywne podejście, w którym stan wagonika jest opisywany przez jego aktualną pozycję na torze. Podejście to upraszcza przestrzeń stanów, jednak sprawia, że wagonik nauczony jazdy po jednym torze prawdopodobnie nie poradzi sobie na innym. Uważamy, że istotne jest, aby algorytm generował strategię przenośną między różnymi trasami.



Rys. 1.1. Opis stanu wagonika

Aby algorytm działał na dyskretnej przestrzeni stanów, wartości pomiarów wymienionych sygnałów będą dyskretyzowane. Celem uczenia jest wyznaczenie optymalnej strategii, czyli ustalenia jaka akcja jest najlepsza w danym stanie (Czyli jaką siłę ma generować silnik przy danym nachyleniu toru i prędkości wagonika). Przestrzeń akcji również będzie dyskretna – siła będzie mogła przybierać wartości z określonego zbioru. Nagroda za osiągane stany wyznaczana będzie przez krytyka (stanowiącego element implementowanego przez nas środowiska). Aby zapewnić, że wagonik nie będzie się cofał ani przekraczał dozwolonej prędkości, zdarzenia te będą powodowały osiągnięcie stanów terminalnych z dużym negatywnym wzmocnieniem.

Ponieważ zależy nam na uzyskaniu agenta radzącego sobie dobrze na różnych trasach, po algorytmie Q-Learning zastosujemy walidacje strategii w różnych środowiskach. Na tej podstawie możliwe będzie wybranie najlepszego z nich. W ramach eksperymentów planujemy również przetestować np. uśrednienie strategii lub metody z głosowaniem wielu agentów – być może to poskutkuje lepszym agentem. Dodatkowo, możliwe będzie dostosowanie hiperparametrów algorytmu Q-Learning dla uzyskania możliwie wszechstronnych wagoników.

Plan eksperymentów

Badanie hiperparametrów algorytmu Q-Learning

— Modyfikacja reguł nagradzania/karania agenta w trakcie nauki
— Porównanie startegii ϵ-zachłannej (z różnymi wartościami ϵ) z strategią Boltzmanna
— Z parametrami β, γ, α.

Poszukiwanie najlepszego uniwersalnego agenta

— Wybór najlepszego agenta ze wszysktich
— Agent z uśrednioną strategią
— Metoda ensemble – agenci głosują na akcje

2. Wykonanie projektu

Pracę nad projektem rozpoczęliśmy od implementacji środowiska. Została ona wykonana zgodnie ze specyfikacją podaną w dokumentacji wstępnej. Zrealizowana jest ona jako dwuwymiarowa symulacja fizyki wagonika jadącego po torze o zmiennym nachyleniu, który jest w stanie sterować swoją siłą ciągu. Oprócz możliwości zadawania wartości sterowania, symulacja środowiska pozwala na pomiar jego stanu, tj. prędkości wagonika, nachylenia toru pod nim oraz nachylenia toru przed wagonikiem. Akcje agenta oceniane są przez krytyka, będącego elementem środowiska. Krytyk ten wyznacza nagrodę (lub karę) zdobytą przez agenta w każdym kroku symulacji przy zastosowaniu reguł przytoczonych w dokumentacji wstępnej. Aby zapewnić szybką, ale wciąż dokładną symulację wykorzystaliśmy bibliotekę *PyBind11* pozwalającą kompilować kod C++ do modułu Pythona o wysokiej wydajności. Do wizualizacji symulacji wykorzystujemy moduł *PyGame*.

Agent uczony jest z wykorzystaniem uczenia ze wzmocnieniem. Używamy algorytmu Q-learning, w którym wykorzystywana przez agenta strategia optymalna jest poprawiana po każdym kroku środowiska w zależności od zdobytej w nim nagrody. Strategia ta, reprezentowana jako funkcja Q, przyporządkowująca optymalną akcję każdemu ze stanów środowiska, jest rezultatem działania algorytmu.

Wśród plików projektu, w folderze *demos*, zamieszczamy dwa programy demonstracyjne. Pierwszy z nich – *forHumanPlayer.py* – pozwala użytkownikowi sterować wagonikiem (strzałki prawo/lewo) w symulacji i dostarcza wizualizację środowiska. *forAIPlayer.py* uruchamia wizualizację środowiska, w którym porusza się agent wykorzystujący jedną z wyuczonych podczas realizacji projektu strategii.

Weryfikujemy, że zaimplementowany algorytm Q-learning znajduje strategię zapewniającą agentowi skuteczne pokonywanie zadanego toru. Odnotowawszy sukces, rozpoczynamy testowanie algorytmu.

3. Eksperymenty

3.1. Szukanie optymalnych parametrów gamma, beta, epsilon / T

Testowanie parametrów algorytmu:
Testy zostały wykonane na następujących warunkach:
Nauka:

— 250 iteracji uczących, w każdej 5000 iteracji czasu na naukę
— co 50 iteracji uczących zmieniany jest tor - tzn. jest 5 torów, na których agent uczy się poruszać
— gdy podczas 5000 iteracji agent osiągnie cel wcześniej, następuje przejście do kolejnej iteracji
— gdy podczas 5000 iteracji agent nie osiągnie celu, następuje przejście do kolejnej iteracji

Testowanie:

— 9 iteracji testowych, w każdej 5000 iteracji na dotarcie do celu
— co 3 iteracje testowe następuje zmiana testowanego toru - tzn. są 3 tory, na których nauczony agent jest testowany
— tory testowe są różne od torów uczących

Dane uzyskiwane z testów:

— wartość parametru gamma
— wartość parametru beta
— wartość parametru epsilon lub T (temperatura układu, w przypadku użycia strategii Boltzmanna)
— średnia liczba osiągnięć celu podczas nauki
— średnia liczba osiągnięć celu podczas testowania (najważniejsze kryterium oceny)
— uśredniony najlepszy czas osiągnięcia celu
— uśredniony najgorszy czas osiągnięcia celu
— uśredniony średni czas osiągnięcia celu (ważne kryterium oceny)
— uśrednione odchylenie standardowe czasu osiągnięcia celu
— uśredniona najlepsza nagroda osiągnięta przez agenta (ważne kryterium oceny)

Każdy cykl nauka-test powtarzany jest dla danych testowanych parametrów 10 razy i otrzymane wyniki są uśredniane.

3.2. Wyniki i wnioski

Dane zebrane podczas testów pokazywane są tylko dla tych rekordów, w których agent zawsze osiągnął cel. Wszystkie dane z testów zawarte są w odpowiadających im plikach w Excelu i CSV.

Test parametrów gamma, beta i epsilon (strategia epsilon zachłanna):

Widać przewagę pary parametrów 0.9, 0.1 (3 na 7 wyników które osiągnęły 100% osiągnięć celów), z epsilonem o wartościach [5, 10, 20]:

Gamma	Beta	Epsilon	Learning Reached	Testing Reached	Best	Worst	Average	Best Reward
0,9	0,1	10	230,5	9	309,4	414,9	370,77	41,06
0,9	0,1	5	223,7	9	308,2	423,7	375,09	41,18
0,9	0,1	20	230,1	9	351	473,3	424,23	36,90

Uznajemy, że gamma = 0.9 i beta = 0.1 są optymalnymi parametrami. Patrząc po kolejnych istotnych parametrach (średnim czasie osiągnięcia celu i średniej najlepszej nagrodzie) można dojść do wniosku, że najlepszy będzie epsilon = 10, chodź wyniki dla epsilon = 5 są jedynie minimalnie gorsze.
Przechodząc do analizy danych z testów współczynnika T Boltzmanna:

Gamma	Beta	T	Learning Reached	Testing Reached	Best	Worst	Average	Best Reward
0,5	0,1	0,25	0	9	301,5	439,5	380,5	41,86
0,9	0,1	0,25	0	9	258,7	370,1	329,76	46,17
0,9	0,1	0,5	0	9	294,6	418,9	373,64	42,55
0,9	0,5	0,5	0	9	309,2	420,4	378,84	41,045
0,9	0,1	0,75	0	9	297,1	430	381,82	42,30
0,5	0,5	0,75	0	9	325,6	504,5	428,39	39,43
0,9	0,1	1	0	9	319,5	450,8	397,58	40,01
0,1	0,5	1	0	9	333,8	524,7	438,18	38,65
0,9	0,5	1	0	9	330,5	483,3	419,23	38,94

Znowu można zauważyć przewagę w występowaniu pary gamma = 0.9 i beta = 0.1 (4 na 9 wyników które osiągnęły 100% osiągnięć celów):

Gamma	Beta	T	Learning Reached	Testing Reached	Best	Worst	Average	Best Reward
0,9	0,1	0,25	0	9	258,7	370,1	329,76	46,17
0,9	0,1	0,5	0	9	294,6	418,9	373,64	42,55
0,9	0,1	0,75	0	9	297,1	430	381,82	42,30
0,9	0,1	1	0	9	319,5	450,8	397,58	40,01

Zdecydowanie najlepszy wynik został osiągnięty dla T=0.25. Jest on również najlepszy wynikiem globalnie w całych testach globalnie w testach współczynnika T.

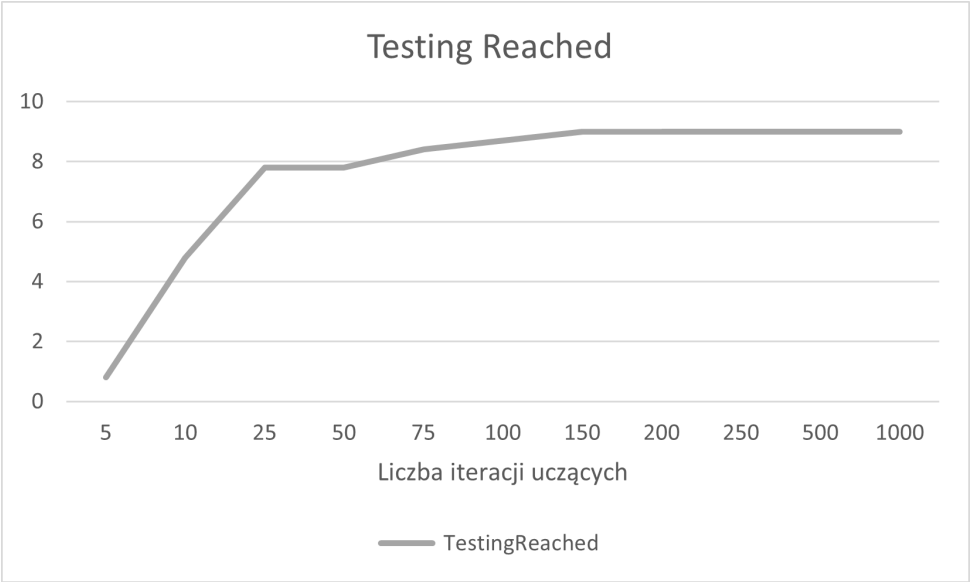
Porównując testy strategi epsilon zachłannej i Boltzmannowskiej można zauważyć ciekawą zależność. W strategi Boltzmannowskiej algorytm podczas nauki ani razu nie osiągnął celu, przy czym podczas testowania osiągnął średnio lepsze wyniki i więcej testów osiągnęło 9/9 (procentowo - 9 na 36 testów strategi Boltzmannowskiej i 7 na 45 strategi epsilon zachłannej). Dlatego do dalszych testów wybieramy strategię Boltzmannowską z parametrami gamma = 0.9, beta = 0.1 i T = 0.25.

3.3. Testowanie "nasycenia" uczenia

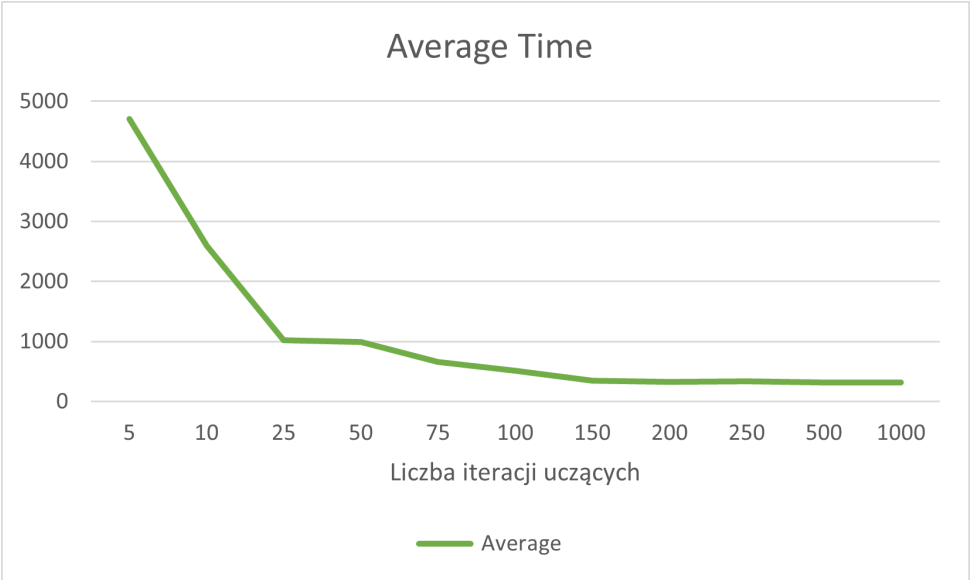
W tym eksperymencie będziemy testować po ilu iteracjach algorytm uczący się "nasyci", tzn. postęp w uczeniu nie będzie znacząco rósł wraz ze wzrostem iteracji uczących.

Testowane warianty to: [5, 10, 25, 50, 75, 100, 150, 200, 250, 500, 1000].
Są one podzielne przez 5, ponieważ dalej uczymy na 5 różnych torach.

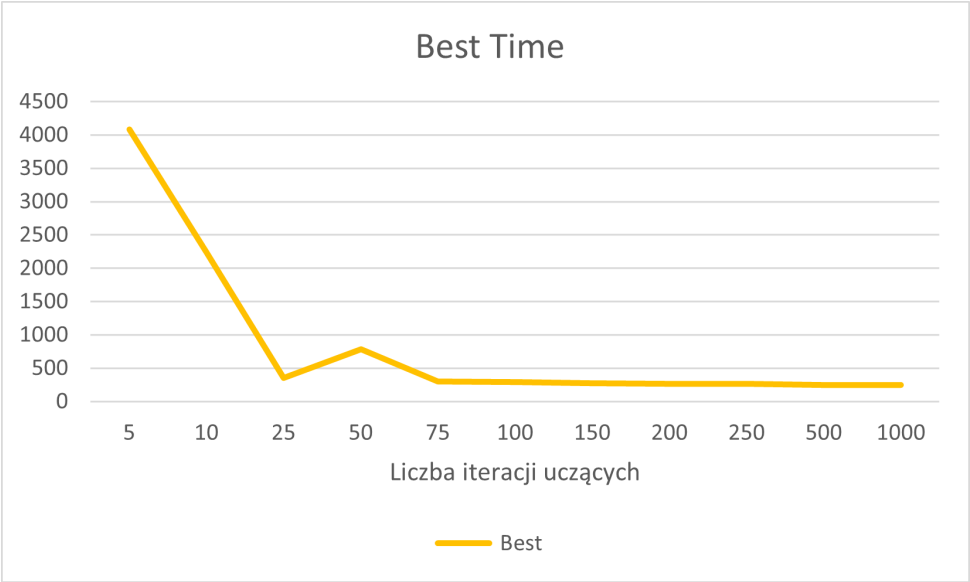
Zebrane dane również znajdują się w odpowiednich plikach CSV i Excel. Poniżej przedstawione są wykresy:



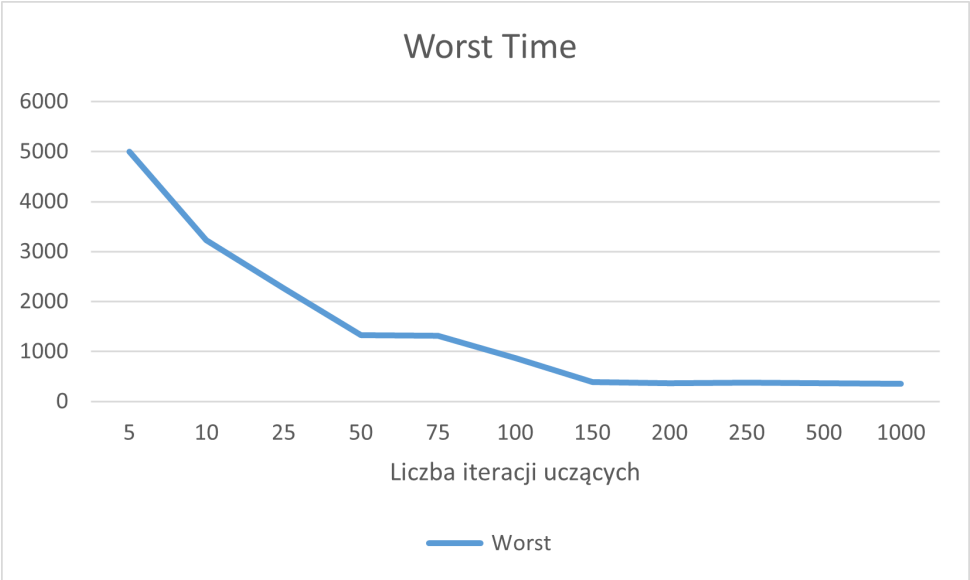
Rys. 3.1. Wykres osiągniętych celi przez agenta



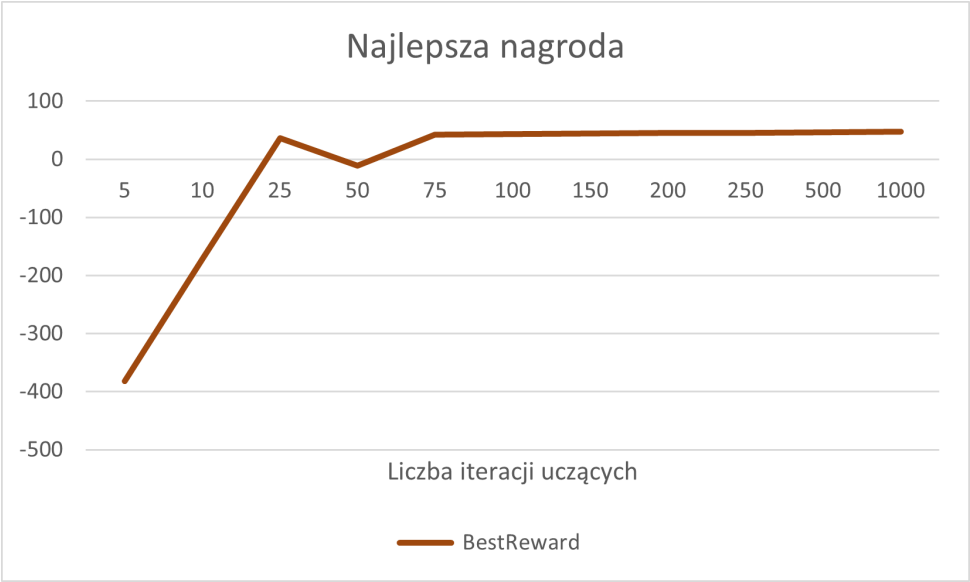
Rys. 3.2. Wykres średniego czasu osiągnięcia celu przez agenta



Rys. 3.3. Wykres najlepszego czasu osiągnięcia celu przez agenta



Rys. 3.4. Wykres najgorszego czasu osiągnięcia celu przez agenta



Rys. 3.5. Wykres średniej najlepszej nagrody zdobytej przez agenta

Na podstawie testów można stwierdzić, że przy najlepszych parametrach (3.2) wystarczy 150 iteracji uczących (po 30 na tor), aby "nasycić" wiedzę agenta.

3.4. 1 dobrze nauczony vs 5 średnio nauczonych

W tym eksperymencie sprawdzamy co lepsze:

— jeden agent uczony na wszystkich torach (po 100 iteracji na tor)
— 5 agentów uczonych (każdy swój tor i 100 iteracji)

Każdy z 5 agentów jest uczony na tylko 1 torze. Podczas testowania wspólnie podejmują decyzję jaki ruch wagonika wybrać - w wyniku głosowania wybierana jest akcja najczęściej wybrana przez "rój" agentów. Jeśli jest więcej niż jedna najczęstsza to wybierana jest losowa z nich. Testy były wykonywane 25 razy i uśredniane.

Test pierwszy - test przy "nasyconym" uczeniu po 30 iteracji na tor dla agenta z "roju" i 150 iteracji na wszystkich torach dla pojedynczego agenta.

Solo Learning Reached	Solo Testing Reached	Solo Best	Solo Worst	Solo Average	Solo Average Reward
0	9	305,89	391	343,84	37,62

Hive Learning Reached	Hive Testing Reached	Hive Best	Hive Worst	Hive Average	Hive Average Reward
0	6,64	1916,89	3970,56	1914,88	-135,88

Test drugi - test przy "nienasyconym" uczeniu po 10 iteracji na tor dla agenta z "roju" i 50 iteracji na wszystkich torach dla pojedynczego agenta.

Solo Learning Reached	Solo Testing Reached	Solo Best	Solo Worst	Solo Average	Solo Average Reward
0.0	7.92	321.67	4487.0	969.78	-35.20

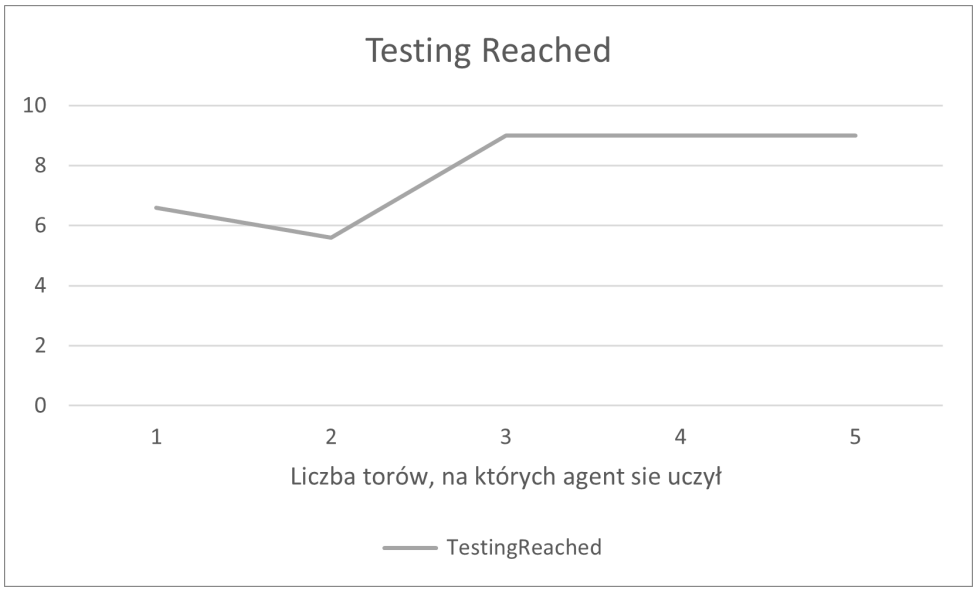
Hive Learning Reached	Hive Testing Reached	Hive Best	Hive Worst	Hive Average	Hive Average Reward
0.0	3.56	3943.67	5001.0	3557.42	-322.78

Z testu wynika jednoznacznie, że lepiej nauczyć jednego agenta ale dobrze i w zróżnicowanym środowisku, niż wielu agentów do wyspecjalizowanych środowisk.

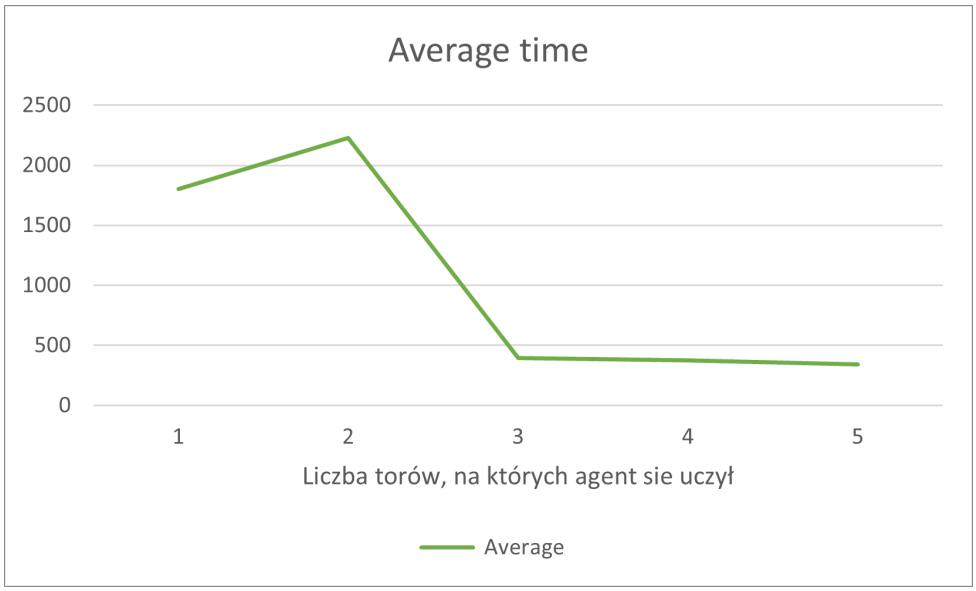
3.5. Test wpływu zróżnicowania środowiska na jakość uczenia

W każdym podejściu pojedynczy agent ma 150 iteracji uczących. W zależności od testu dzieli je na 1, 2, 3, lub 5 torów.

Wykresy:



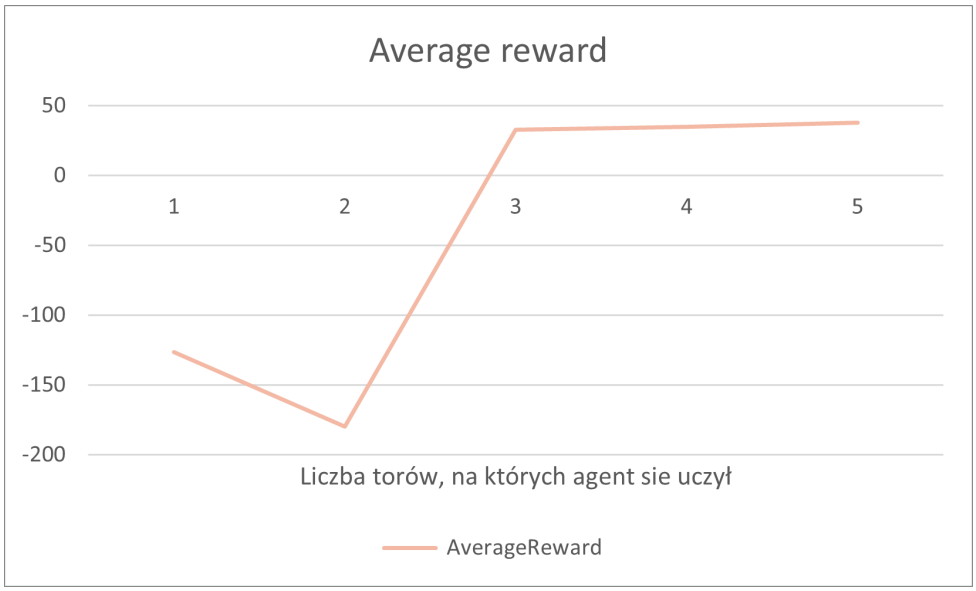
Rys. 3.6. Wykres osiągniętych celi przez agenta



Rys. 3.7. Wykres średniego czasu osiągnięcia celu przez agenta

Rys. 3.8. Wykres najlepszego czasu osiągnięcia celu przez agenta

Rys. 3.9. Wykres najgorszego czasu osiągnięcia celu przez agenta



Rys. 3.10. Wykres średniej najlepszej nagrody zdobytej przez agenta

Mimo tego, że przy każdym teście agent miał tyle samo czasu na naukę, widać, że zdecydowanie wydajniej mu ona wyszła, przy bardziej zmiennym środowisku.
Widać, że w testowanym przez nas przypadku wystarczy nauka na 3 różnych torach, aby stworzyć dobrego agenta.

Podsumowując, najlepszym agentem pod względem jakości nauki i wykorzystanych zasobów jest agent uczony korzystając ze strategii Boltzmannowskiej przez 150 iteracji na 3 torach, z wykorzystaniem parametrów T = 0.25, gamma = 0.9 i beta = 0.1.