Dawid Woźniak
Informatyka - Uczenie Maszynowe I Sztuczna Inteligencja

Uczenie ze Wzmocnieniem

Przeszukiwanie drzewa metodą Monte Carlo

1. Opis ćwiczenia.

Celem laboratorium było dokończenie implementacji algorytmu przeszukiwania drzewa metodą Monte Carlo zastosowanego w grze planszowej Izolacja.

2. Wyniki

Poprawność implementacji została wstępnie przetestowana poprzez przesymulowanie 50 gier z losowym graczem na planszy o wymiarach 4x4. W takiej konfiguracji przeszukiwanie drzewa Monte Carlo wygrywało wszystkie 50 gier. Następnie przystąpiłem do tworzenie kolejnych przypadków testowych, które zostały opisane poniżej. Jako politykę drzewa używaną w selekcji zastosowałem dwie metody - Upper Confidence Bound oraz Epsilon Greedy. Planszę zwiększyłem do rozmiarów 8x8.

2.1 Upper Confidence Bound

W tej sekcji badałem wpływ 2 parametrów na jakość rozwiązania: c oraz t_{limit} podanym w sekundach. Zgodnie ze wzorem:

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

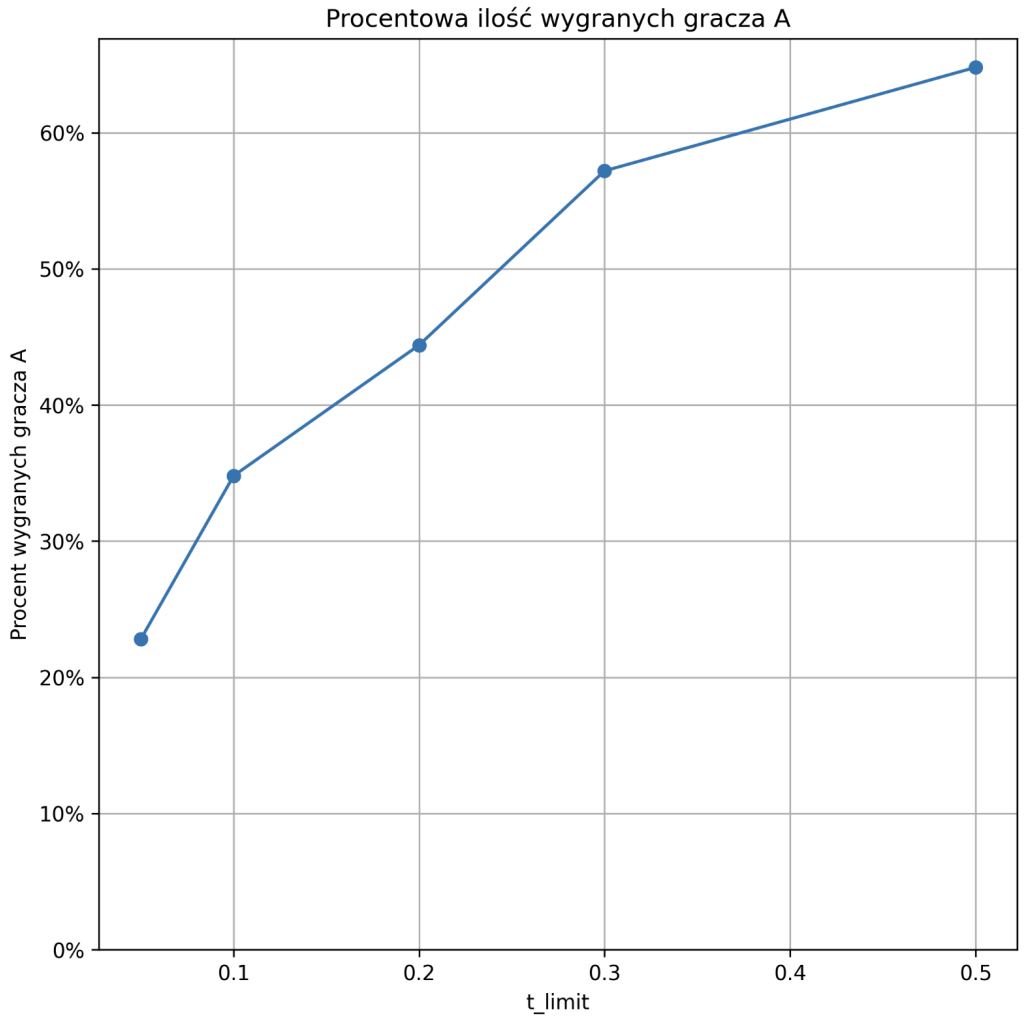
parametr c odpowiada za balans między eksploracją a eksploatacją. Algorytm ten będzie premiował węzły, które były do tej pory rzadko odwiedzane w stosunku do ilości odwiedzin ich rodzica. Parametr t_{limit} odpowiada za ilość czasu przeznaczoną na wykonanie każdej akcji.
Zorganizowałem pojedynki dwóch graczy sterowanych algorytmem Monte Carlo. Każdy pojedynek miał 500 epizodów.

2.1.1 Wpływ czasu na wybór akcji

W tym etapie, przyjąłem parametr $c = 0.5$ dla obu graczy. Dla gracza A, badałem kolejno parametry $t_{limit} \in \{0.05,\ 0.1,\ 0.2,\ 0.3,\ 0.5\}$. Gracz B miał stałą konfigurację, $c = 0.5$ oraz $t_{limit} = 0.2$. Poniższa tabela prezentuje procentową ilość wygranych gracza A.

Gracz A \ Gracz B	$t_{limit} = 0.05$	$t_{limit} = 0.1$	$t_{limit} = 0.2$	$t_{limit} = 0.3$	$t_{limit} = 0.5$
$t_{limit} = 0.2$	22.8%	34.8%	44.4%	57.2%	64.8%

Tabela 1. Ilość wygranych gracza A w zależności od parametru t_{limit}



Rys. 1. Ilość wygranych gracza A w zależności od parametru t_{limit}

Analizując otrzymane wyniki możemy zauważyć, że zwiększenie czasu na wybór akcji skutkuje otrzymaniem lepszych wyników. Jest to spodziewany efekt, gdyż większy czas na wybór akcji oznacza, że metoda Monte Carlo przeanalizuje więcej możliwości a co za tym idzie wartościowanie stanów będzie dokładniejsze. Ciekawy przypadek wystąpił dla $t_{limit} = 0.2$. W tej konfiguracji grali ze sobą dokładnie tacy sami gracze a gracz A uzyskał

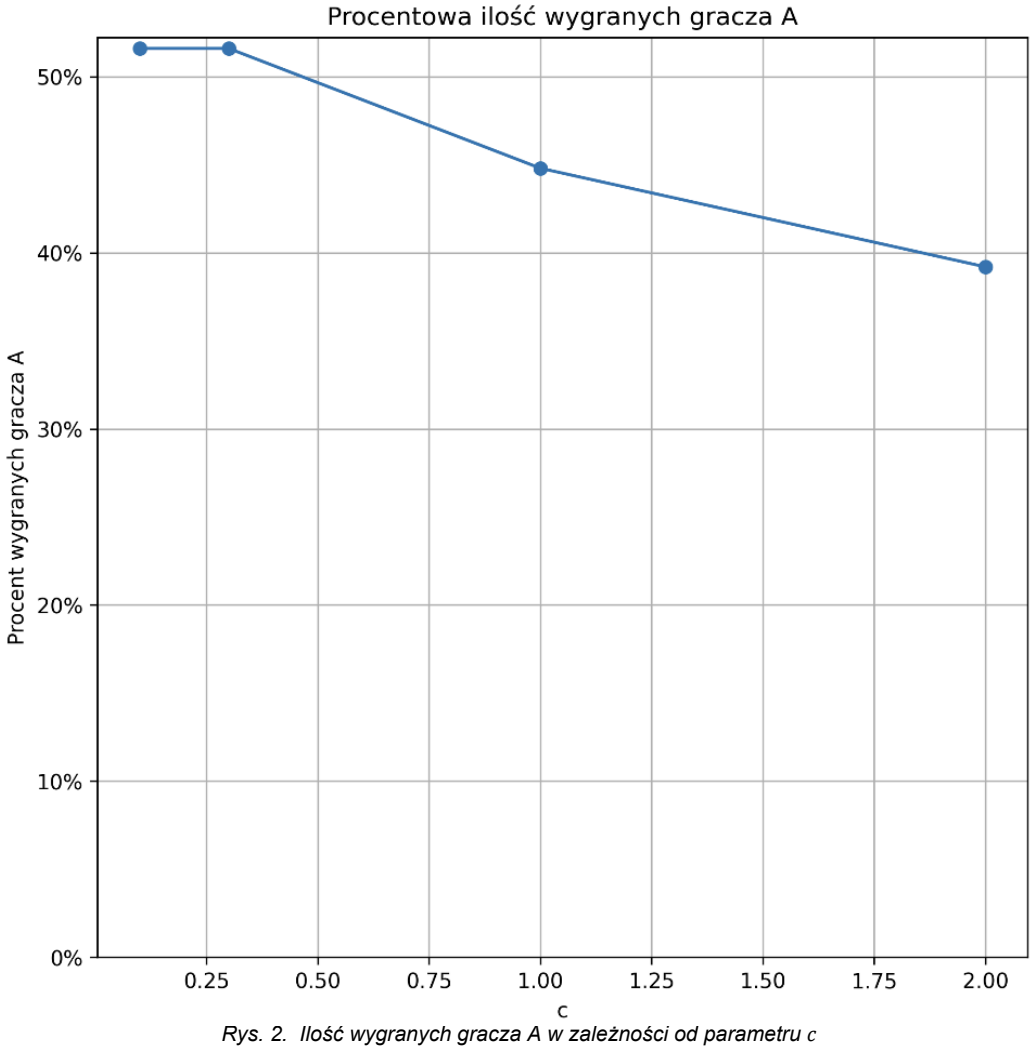
44.4% wygranych. Wydaje się, że w przypadku Monte Carlo, bardziej opłaca się zaczynać rozgrywkę jako drugi.

2.1.2 Wpływ parametru c

W tym etapie przyjąłem parametr $t_{limit} = 0.2$ dla obu graczy. Dla gracza A badałem kolejno parametry $c \in \{0.1,\ 0.3,\ 1,\ 2\}$. Dla gracza B przyjąłem konfigurację $c = 0.5$ oraz $t_{limit} = 0.2$

Gracz A \ Gracz B	$c = 0.1$	$c = 0.3$	$c = 1$	$c = 2$
$c = 0.5$	51.6%	51.6%	44.8%	39.2%

Tabela 2. Ilość wygranych gracza A w zależności od parametru c



Rys. 2. Ilość wygranych gracza A w zależności od parametru c

Tym razem możemy zauważyć odwrotny trend niż w przypadku parametru t_{limit}.
Zwiększenie parametru c do zbyt dużej wartości powoduje spadek ilości zwycięstw gracza A.

Możemy to tłumaczyć w ten sposób, że zbyt duży parametr c powoduje za dużo eksploracji w stosunku do eksploatacji i w ten sposób nie wykorzystujemy wystarczająco zdobytej wiedzy o poszczególnych stanach a w zamian tego zbytnio staramy się wybierać mniej znane stany.

2.2 Epsilon Greedy

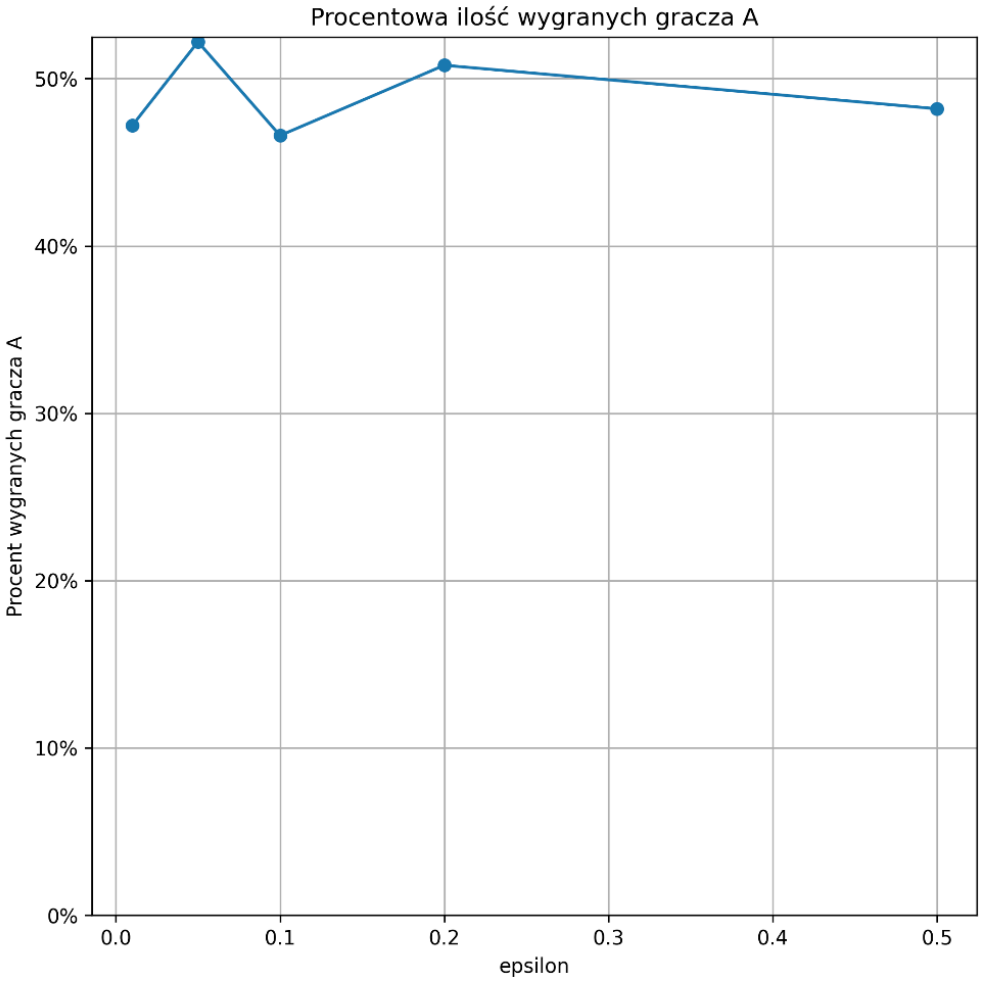
W tej sekcji badałem wpływ parametru ε w podejściu epsilon greedy. Zdecydowałem się nie porównywać wpływu parametru t_{limit} gdyż spodziewałem się podobnych wyników jak we fragmencie 2.1.1 - wraz ze wzrostem czasu na wybór akcji wzrasta skuteczność rozwiązania. Metoda Epsilon greedy działa zgodnie z następującym wzorem:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

Z prawdopodobieństwem równym ε dajemy sobie szansę na wybór losowej akcji, przez co eksplorujemy zamiast eksploatować. Przez dobór parametru ε balansujemy między eksploracją a eksploatacją. Kolejny raz zorganizowałem pojedynki dwóch graczy sterowanych algorytmem Monte Carlo, tym razem z polityką wybierania akcji Epsilon Greedy. Każdy pojedynek miał 500 epizodów. Przyjąłem parametr $t_{limit} = 0.2$ dla obu graczy. Dla gracza A badałem kolejno parametry $\varepsilon \in \{0.01,\ 0.05,\ 0.1,\ 0.2,\ 0.5\}$. Dla gracza B użyłem konfiguracji $\varepsilon = 0.5$ oraz $t_{limit} = 0.2$.

Gracz A \ Gracz B	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.5$
$\varepsilon = 0.05$	47.2%	52.2%	46.6%	50.8%	48.2%

Tabela 3. Ilość wygranych gracza A w zależności od parametru ε



Rys. 3. Ilość wygranych gracza A w zależności od parametru ε

Analizując otrzymane wyniki, najlepiej wypadł wariant dla $\varepsilon = 0.2$. Zbyt małe i zbyt duże wartości mają zły wpływ na działanie algorytmu - dla zbyt małych eksplorujemy zbyt mało a dla dużych zbyt dużo, tak naprawdę zbliżając się do losowego gracza. Co ciekawe, dla dwóch algorytmów o tych samych parametrach zwyciężył gracz A gdy dla wariantu UCB zwyciężał gracz B i to z jeszcze większą przewagą. Porównując z sekcją 2.1.2 za pomocą wariantu Epsilon Greedy otrzymałem podobne wyniki jak dla wariantu UCB.

2.3 Upper Confidence Bound vs Epsilon Greedy

Jako ostatni test postanowiłem stworzyć pojedynek algorytmów Monte Carlo z politykami wybierającymi Upper Confidence Bound oraz Epsilon Greedy. Dla obu graczy ustawiłem parametr $t_{limit} = 0.2$. Dla UCB wybrałem $c = 0.1$ a dla Epsilon Greedy $\varepsilon = 0.2$, zgodnie z wynikami otrzymanymi wcześniej. Zorganizowałem 2 pojedynki po 500 epizodów raz w kolejności grania UCB vs Epsilon Greedy a raz Epsilon Greedy vs UCB. W pierwszym pojedynku UCB wygrał 74.6% potyczek a w drugim 76.4% co świadczy o znacznej

przewadze tej polityki. Potwierdził się znowu zaobserwowany wcześniej fakt, że dla wariantu UCB metoda Monte Carlo lepiej działa jeżeli zaczyna rozgrywkę jako druga.

3. Wnioski

Udało się poprawnie dokończyć implementację przeszukiwania drzewa metodą Monte Carlo, co zostało potwierdzone prostym przypadkiem testowym. Studium parametryczne w politykach Upper Confidence Bound oraz Epsilon Greedy kolejny raz pokazało podstawowy problem w uczeniu ze wzmocnieniem - utrzymanie balansu między eksploracją a eksploatacją oraz to jak ważne jest ustawienie odpowiednich parametrów przy pracy z tego typu algorytmami. W czasie testów, gdy tworzyłem pojedynki graczy o tych samych politykach otrzymałem zbliżone wyniki. Po wyborze najlepszych konfiguracji dla UCB oraz Epsilon Greedy stworzyłem pojedynek dwóch graczy z różnymi politykami. Polityka UCB okazała się znacznie lepsza od Epsilon Greedy osiągając 74.6% i 76.4% wygranych. Warty odnotowania jest fakt, że dla polityki UCB opłacalne jest zaczynać grę jako drugi. Tak duża różnica na korzyść UCB może wynikać z sposobu balansu między eksploracją a eksploatacją. Dla UCB wydaje się on być trochę lepiej skonstruowany, gdyż za pomocą podanego wzoru możemy premiować węzły, które były rzadko odwiedzane lub minęło dużo czasu od ich odwiedzenia a więc ich potencjał mógł się zmienić. Dla Epsilon Greedy po prostu wybieramy losową akcję co może mieć wpływ na jakość wyniku.