Sztuczna inteligencja
Pracownia 4b (Szachy)
(termin: do końca regularnych zajęć w semestrze)
(bonus: +1 za każde zadanie oddane do i w trakcie pierwszych zajęć w czerwcu)

Zapowiedziane trzy zadania z tej listy są z gwiazdką. Dotyczą one szachów, można w nich korzystać z bibliotek szachowych odpowiedzialnych za generowanie ruchów, sprawdzanie czy jest mat, itp. Przykładową taką biblioteką jest python-chess.

Zadanie 1. (4+X) Szachy z workiem treningowym[1]

Masz napisać agenta, który gra w szachy z losowym przeciwnikiem (Twój agent gra jako biały i zawsze zaczyna). Losowy przeciwnik wybiera z jednakowym prawdopodobieństwem któryś z możliwych ruchów, Twój agent to wie i może optymalizować swoją strategię dla tego konkretnie oponenta. Każda partia kończy się po 100 ruchach Twojego agenta (o ile wcześniej nie nastąpił mat) i Twój agent dostaje następujące punkty:

- (100 - liczba wykonanych ruchów białego), jeżeli to on wygrał partię
- -100, jeżeli nie udało się zamatować czarnego króla
- -1000, jeżeli to czarne dały mata

Należy rozegrać 50 gier i podać średni wynik agenta. Do zaliczenia zadania wymagane jest, by był on dodatni. Przed zajęciami będą podane dodatkowe warunki na bonusy punktowe. Uwaga: nie wolno korzystać z zewnętrznych silników szachowych.

Zadanie 2. 4p Będziemy rozważać szachy super-błyskawiczne, w którym agenci posługują się wyłącznie funkcją heurystyczną (bez żadnego dodatkowego przeszukiwania typu minimax czy MCTS). Funkcja heurystyczna jest sumą następujących 4 składników

- waga moich bierek (pion ma wagę 1, waga pozostałych bierek jest parametrem agenta)
- waga bierek przeciwnika (ze znakiem przeciwnym)
- $\alpha\times$ liczba-moich-możliwych-ruchów
- $-\alpha\times$ liczba-możliwych-ruchów-przeciwnika

Dodatkowo, jeżeli ruch kończy się matem, to ma on wartość nieskończoną (czyli inne czynniki nie mają znaczenia).

Naszym celem będzie analizowanie partii rozgrywanych przez agentów. Przyjmiemy, że każda partia trwa co najwyżej K ruchów (K=100?). Agent wygrywa partię przez knock-out, jeżeli udało mu się zamatować przeciwnika, jeżeli nie – to oceniamy sytuację na planszy i przyznajemy zwycięstwo 'na punkty'.

Ten drugi przypadek ma 2 warianty (do wyboru przez studenta):

a) Używamy 'standardowego' sposobu oceniania (czyli 1 dla piona, 3 dla skoczka, 3 dla gońca, 5 dla wieży, 9 dla hetmana).[2]

b) Używamy zewnętrznego, popularnego silnika szachowego, jak np. stockfish jako arbitra[3]. Ten wariant jest bardziej wiarygodny i daje dodatkowe 2 punkty.

Twoim celem w tym zadaniu jest wylosować populację około 100 agentów (określanych przez wagi bierek i wartość α) i przeprowadzić pojedynki 'każdy z każdym'. Zliczyć zwycięstwa i porażki agentów i sporządzić raport do pliku, zawierający specyfikację i osiągi każdego agenta (oczywiście warto posortować agentów od najlepszego do najgorszego).

[1] Tak jakoś wyszło, że w dwóch zadaniach będziemy używać terminologii bokserskiej, ale bez związku z Szachowym boksem

[2] Oczywiście taki sposób nie jest całkiem właściwy, bo agent który nieco inaczej rozkłada wagi wydaje się mieć mniejsze szanse. Ale warto sprawdzić tę hipotezę, ponadto wybór α jest jak najbardziej możliwy

[3] Zwróć uwagę, że python-chess umożliwia odwołanie się do stockfisha, za pomocą chess.engine.analyse

Zadanie 3. 2+X Cel i zasady są takie same jak w poprzednim zadaniu. Jedyną różnicą jest wymaganie, żeby szukać optymalnego agenta za pomocą jakiejś heurystycznej metody przeszukiwania przestrzeni stanów (stan == specyfikacja agenta). Przy czym wartością agenta będzie średni wynik z 5-cioma najlepszymi agentami z poprzedniego zadania. Premia jest uznaniowo przyznawana ze względu pracochłonność i jakość rozwiązania.

Zadanie 4. 2 Dobierz parametry StockFisha tak, by czas trwania procedury wybierającej ruch był w przybliżeniu taki, jak w zadaniu 1. Porównaj jakość gry Twojego agenta i StockFisha przeciw agentowi losowemu (jak w zadaniu 1).