Zuzanna Misztal

Sztuczna Inteligencja do gry Tetris

Czym jest Tetris?

Tetris to komputerowa gra logiczna stworzona w roku 1984 w Rosji przez Aleksieja Pażytnowa. Podstawą do gry jest plansza 20 x 10, na której układa się klocki zwane tetromino. Klocki pojedynczo pojawiają się na środku górnych wierszy planszy, po czym zaczynają opadać. Gracz może klocek obracać oraz przesuwać w prawo bądź lewo. Gdy klocek nie ma już możliwości zejść niżej, wtapiany jest w planszę, a na górze pojawia się nowy klocek. Zadaniem gracza jest takie układanie tetromines, aby tworzyły pełne wiersze, gdyż każdy pełny wiersz zostaje usunięty (wyższe wiersze przesuwają się odpowiednio w dół), dając punkty, oraz więcej miejsca na planszy dla nowo przybyłych klocków. Gra kończy się, gdy na planszy nie ma miejsca dla nowego klocka. W klasycznej wersji gry wraz z narastającą liczbą usuniętych rzędów zwiększa się tempo spadania klocków, jednak na potrzeby AI aspekt ten został przeze mnie pominięty. Jest wiele różnych wariantów Tetrisa – ten, który ja napisałam przechowuje dla gracza dodatkową informację o kształcie następnego klocka.

Podejście do AI

Na planszy 20 x 10 możliwych kombinacji jest około decylion, więc z góry można porzucić proste podejście, w którym dla każdej możliwej planszy przypisane jest jak ułożyć dany klocek. Zamiast tego stosuje się podejście heurystyczne bazujące na obliczeniu “jak dobra” jest dana plansza, poprzez przydzielenie jej wartości punktowej.

Heurystyki

1. Łączna wysokość planszy.

Jest to pierwsza heurystyka, którą zaimplementowałam, i już tylko z jej użyciem programowi udało się usunąć 25 linijek. Idea jest prosta: wynikiem tej funkcji jest suma wysokości wszystkich kolumn, gdzie wysokość kolumny to odległość jej najwyższego elementu od dołu. Generalnie “im wyżej tym gorzej”, wynik tej funkcji jest wzięty z minusem przy obliczaniu wartości planszy. Sprawia to, że jeśli jest taka możliwość, to klocek zostanie położony tak, aby nie powstały pod nim dziury, bo każda dziura to dodatkowy punkt wysokości. Ponadto naturalnie jeśli będzie miał okazję to postawi klocek tak, aby usunąć linię, bo każda usunięta linia to –10 punktów wysokości. O ile w pierwszym etapie pisania programu była to heurystyka dużo dająca (szczególnie, że była pierwsza), tak po dopisaniu heurystyki usuniętych linii i heurystyki dziur stała się redundantna. Powodem rozbicia tej heurystyki na dwie wymienione wyżej jest możliwość zróżnicowania wag.

2. Usunięte linie.

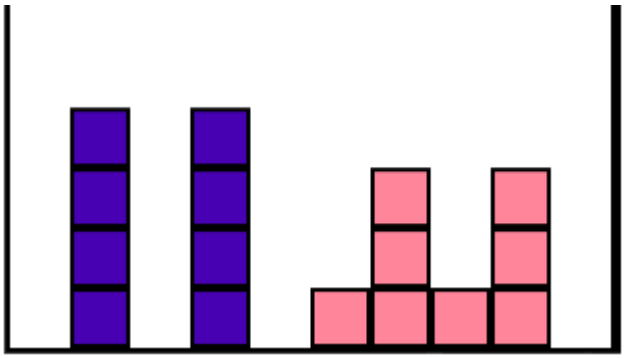
Tak proste jak tylko może być - jeśli konkretne ułożenie klocka spowodowało usunięcie linii, plansza która powstała dzięki takiemu ułożeniu zyskuje 50 punktów.

3. Dziury.

Dziura w kolumnie to każde puste pole pod najwyższym niepustym polem. Każda dziura na planszy obniża jej wartość o 30 punktów.

4. Zróżnicowanie wysokości kolumn.

Wartością tej heurystyki jest różnica wysokości pomiędzy każdymi dwoma sąsiednimi kolumnami, poza tym jeśli kolumna druga (odpowiednio przedostatnia) jest wyższa od pierwszej (odpowiednio ostatniej) - różnica ich wysokości jest jeszcze raz dodawana do wyniku. Im większe jest zróżnicowanie wysokości w sąsiednich kolumnach tym trudniejsze jest późniejsze działanie na takiej planszy, tak aby nie powstawały dziury, więc wynik tej heurystyki odejmowany jest od dotychczas obliczonej wartości planszy (ja przyznałam mu wagę -2).



Powyższy przykład ilustruje dlaczego taka heurystyka jest potrzeba – bez niej algorytm może ustawić klocki np w taki właśnie sposób, i wydaje mu się to optymalne, bo nie powstają (jeszcze) żadne dziury. Jednak aby dalej mógł stawiać klocki tak, aby nie powstały dziury i linijki się usuwały potrzebuje aż trzech tetromines typu ‘I’.

Strategie algorytmu

W napisanym przeze mnie programie dostępne są trzy strategie doboru najlepszego ruchu dla danego klocka.

1. Greedy

W tym rozwiązaniu algorytm zupełnie pomija wiedzę o następnym klocku. Jedyne co robi, to dla wszystkich legalnych ruchów kalkuluje (na podstawie wypisanych wyżej heurystyk) wartość planszy, która przez dany ruch powstała, a następnie wskazuje ruch, dzięki któremu wartość ta jest największa.

2. Dwuetapowy pełny

Do algorytmu wprowadzona została informacja o następnym tetromino. Kalkulacja punktów dla ruchu rozszerza się o sprawdzenie wszystkich możliwych “konsekwencji” tego ruchu, czyli wszystkich możliwych ruchów drugiego klocka na planszy powstałej z ruchu klocka pierwszego. Z tych wyników wybierany jest największy i on jest przypisywany jako wartość dla rozpatrywanego ruchu (klocka pierwszego). Następnie podobnie jak wcześniej wybierany jest ruch, któremu przyznana była najwyższa wartość. Algorytm ten działa znacznie lepiej niż poprzedni, jest to jednak kosztem czasu poświęconego na obliczanie najlepszego ruchu.

3. Dwuetapowy przyspieszony (N)

Ten algorytm wymyślony został, aby uwzględniać informacje o drugim klocku, a jednocześnie zmniejszyć czas na wykonanie ruchu. Bazuje on na tym, że licząc ruch klocka pierwszego już przegląda wszystkie możliwe ruchy drugiego, więc zapamiętuje najlepsze, i gdy algorytm jest powtarzany mając

dopasować kolejny klocek, sięga do tych zapamiętanych i te zapamiętane ewaluuje na podstawie nowo nabytych informacji o swoim następniku. (N) odnosi się do ilości zapamiętywanych klocków.

Testy

	usunięte linie (średnio)	ilość prób	czas ułożenia klocka
1. Greedy	143	10	0.043 sekundy
2. Dwuetapowy pełny	2368	5	0.82 sekundy
3. Dwuetapowy przyspieszony (4)	2033	6	0.13 sekundy
3. Dwuetapowy przyspieszony (8)	1976	9	0.25 sekundy

Post scriptum

W trakcie robienia testów uświadomiłam sobie, że brakuje ważnej heurystyki – dobrze by było nie przegrać. Bez niej zdarzało się, że lewa lub prawa strona planszy mają miejsce na nowo przybyły klocek, ale np. tworzyłby tam dziurę, więc kładły na środkowej kolumnie (która w wyniku gry była już bardzo wysoka) nie zdając sobie sprawy, że coś może być nie tak. Przedstawiam wyniki strategii Greedy, po dodaniu heurystyki zapobiegającej przegraniu gry jeśli jest na to możliwość.

	Usunięte linie	ilość prób	czas ułożenia klocka
1. Greedy with death fear	244	10	0.03 sekundy

Wynik jednorazowej rozgrywki strategią dwuetapową pełną z nową heurystyką wyniósł 4161 linijek.