Algorytm Decyzyjny dla Rummikub

Aleksander Strzelecki

Politechnika Gdańska
s179971@student.pg.edu.pl

25 sierpnia 2022

Streszczenie

I. Wstęp

II. Prace powiązane

Rozwiązanie problemu Rummikuba zostało zaproponowane w [**?**]. Algorytm wyliczał punktacje sumy ruchów, przy założeniu rozpoczęciu ruchu od konkretnej płytki. Polegało to na iteracji po wartościach rozpoczynających płytek od 1 do 13. W każdej iteracji układane były maksymalne ciągi przy wykorzystaniu płytek gracza o wartości z aktualnej iteracji. Z płytek o wartości z aktualnej iteracji, których nie można było już wykorzystać do rozbudowywania ciągów były wykorzystywane do rozbudowywania grup. Następnie wartości użytych płytek były sumowane i następowało rekurencyjne wywołanie próby rozbudowywania ciągów dla płytek o wartości z aktualnej iteracji + 1. Pojedyńcza iteracja kończyła się w momencie sprawdzenia możliwości rozbudowy ciągów z płytek o wartości = 13. Następnie cała procedura była wywoływana ponownie, ale dla wartości rozpoczynających płytek +1. W ten sposób uzyskiwaliśmy zestaw ruchów dający maksymalną punktację, jednak zakładając brak możliwośći manipulowania płytkami dostępnymi na stole.

Rummikub polega na wykonywaniu przekształceń na zbiorach liczbowych. Problem dobierania przekształceń został przedstawiony artykule [**?**], gdzie przekształcenia używane są do rozwiązywania całki symbolicznie. Autor podzielił przekształcenia na korzystne i heurystyczne. Przekstałcenia korzystne wykonywane są pierwsze, a heurystyczne w przypadku braku możliwości przekształcenia korzystnego. W miejscach, gdzie możliwe jest kilka przekształceń otrzymujemy rozgałęzienie. Następnie sprawdzana jest złożoność wyrażeń w otrzymanych gałęziach i do dalszego rozwiązywania wybierana jest zawsze gałąź z wyrażeniem o najmniejszej złożoności.

Podczas rozgrywki Rummikuba gracz znając aktualny stan planszy oraz dostępne płytki(stan gry) musi dobrać najlepsze przekształcenia(akcje). Po wykonaniu przekształceń gracz może oszacować wartość przekształceń sumując wartości pozbytych płytek(nagroda). Rozwiązanie tak zdefiniowanego problemu zostało przedstawione w [**?**]. Algorytm Q-learning nie wymaga żadnej początkowej wiedzy na temat badanego problemu. Agent(jednostka, której akcje dyktowane są przez algorytm) uczy się, które akcje są najlepsze w danym stanie gry, podczas rozgrywania kolejnych partii. W [**?**] został również przedstawiony sposób na przechowywanie funcji przejścia[Q(stan, akcja) -> nagroda] za pmocą sieci neuronowej. Połączenie sieci neuronowej z Q-learning tworzy algorytm Deep Q-learning.

Algorytm MinMax przedstawiony w [**?**] służył do dobierania ruchów gracza przy jednoczesnym uwzględnianiu możliwych ruchów przeciwnika. [**?**] przedstawia również zoptymalizowaną pod kątem prędkości działania wesję algorytmu MinMax - Alpha-Beta. Algorytm Alpha-Beta został użyty w maszynie

Deep Blue®

III. Proponowane rozwiązanie

IV. Wyniki

V. Podsumowanie