

Zaimplementować grę w kółko i krzyżyk na planszy 3x3 (choć warto NxM i  $k=4$  [~connect4]) z użyciem algorytmu Minimax oraz przycinania alfa-beta. Grać powinny ze sobą dwa AI z ustawianymi oddzielnie parametrami: - głębokość przeszukiwania  $D$  - czy włączone przycinanie - czy losuje swój ruch.

### Założenia wstępne

Zaimplementowana wersja dla gry na planszy 3x3. Jako funkcję heurystyczną oceny stanu gry przyjęto sumę liczby punktów za pola zgodnie z macierzą pokazującą w ilu konfiguracjach wygrywających bierze udział dane pole. Jeżeli pole należy do gracza MAX, to punkty są dodawane do sumy, a jeżeli należy do gracza MIN, to punkty są odejmowane.

3	2	3
2	4	2
3	2	3

Macierz używana w funkcji heurystycznej oceny stanu gry

### Raport z przeprowadzonych eksperymentów

Przeprowadzono eksperymenty polegające na grze dwóch komputerów z różnymi parametrami, przeciwko sobie. Oznaczenia parametrów:

- $d$  – głębokość przeszukiwania (0 oznacza, że komputer robi losowe ruchy i nic nie przeszukuje)
- $\alpha\beta$  – czy przycinanie  $\alpha\beta$  jest włączone

#### P1:( $d=9$ , $\alpha\beta=false$ ) vs P2:( $d=9$ , $\alpha\beta=false$ )

W tej konfiguracji oba komputery widzą całe drzewo gry, więc grają optymalnie. Każda gra wygląda tak samo i kończy się remisem.

Statystyki gracza P1		
Wygrane	Remisy	Porażki
0	5	0

Liczba przeszukanych stanów	
P1	P2
557 487	60 688

#### P1:( $d=9$ , $\alpha\beta=true$ ) vs P2:( $d=9$ , $\alpha\beta=false$ )

W tym przypadku, podobnie jak poprzednio, gra zawsze kończy się remisem. Warto jednak zwrócić uwagę, że po zastosowaniu przycinania  $\alpha\beta$  bardzo znacząco spadła liczba przejranych stanów gry przez gracza P1, a co za tym idzie, algorytm działa dużo szybciej.

Statystyki gracza P1		
Wygrane	Remisy	Porażki
0	5	0

Liczba przeszukanych stanów	
P1	P2
19 212	60 688

P1:( $d=3$ ,  $\alpha\beta=false$ ) vs P2:( $d=9$ ,  $\alpha\beta=false$ )

W tym przypadku, gracz P1 nie był w stanie wygrać żadnej gry ze względu na to, że jego przeciwnik widzi całe drzewo gry, a P1 tylko 3 ruchy do przodu.

Statystyki gracza P1		
Wygrane	Remisy	Porażki
0	0	5

Liczba przeszukanych stanów	
P1	P2
940	56 605

P1:( $d=4$ ,  $\alpha\beta=false$ ) vs P2:( $d=4$ ,  $\alpha\beta=false$ )

W tym przypadku, ze względu na takie same głębokości przeszukiwań cały czas pada remis.

Statystyki gracza P1		
Wygrane	Remisy	Porażki
0	5	0

Liczba przeszukanych stanów	
P1	P2
4785	2554

P1:( $d=4$ ,  $\alpha\beta=true$ ) vs P2:( $d=4$ ,  $\alpha\beta=true$ )

Po włączeniu przycinania  $\alpha\beta$  liczba przejrzanych stanów przez obu graczy ponownie znacznie spadła, ale przebieg gry i wyniki nie zmieniły się. Wynika z tego, że opłaca się używać przycinania.

Statystyki gracza P1		
Wygrane	Remisy	Porażki
0	5	0

Liczba przeszukanych stanów	
P1	P2
847	399

P1:( $d=3$ ,  $\alpha\beta=true$ ) vs P2:( $d=3$ ,  $\alpha\beta=true$ )

Dla takich samych parametrów graczy, gra zawsze kończy się porażką gracza P1. Gra układa się tak, że ze względu na małą wartość parametru  $d$ , gracz P1 nie widzi wygranej gracza P2 i priorytetyzuje zdobywanie punktów zgodnie z funkcją heurystyczną. Funkcja ta jest dość prosta i nie uwzględnia wielu czynników.

Statystyki gracza P1		
Wygrane	Remisy	Porażki
0	0	5

Liczba przeszukanych stanów	
P1	P2
334	194

P1:(losowe ruchy) vs P2:( $d=1$ ,  $\alpha\beta=true$ )

Pomimo, że gracz P1 wykonuje losowe ruchy, to udało mu się wygrać kilka razy. Gracz P2 przy głębokości  $d=1$  w ogóle nie patrzy na możliwości ruchu gracza P1 i działa według słabej funkcji heurystycznej. Po raz kolejny okazuje się, że dobry dobór tej funkcji jest kluczowy dla dobrego działania algorytmu.

Statystyki gracza P1		
Wygrane	Remisy	Porażki
3	0	15

Liczba przeszukanych stanów	
P1	P2
0	18 lub 20

## Odpowiedzi na pytania

*Czy zaczynając zawsze tak samo (iz tymi samymi ustawieniami) przebieg rozgrywki jest deterministyczny?*

Tak, oba komputery grają optymalnie z punktu widzenia ich algorytmu, więc rozgrywka będzie wyglądała zawsze tak samo. Nie ma elementu losowości.

*Czy można wygrać z komputerem? Jeżeli tak to kiedy?*

W ogólności komputer liczy kilka tysięcy wariantów na sekundę (zależy od gry), a człowiek maksymalnie kilka. Przy takim samym ograniczeniu czasowym na ruch, dużej głębokości przeszukiwania i dobrej funkcji heurystycznej nie ma szans wygrać z komputerem. Wygrana jest możliwa jeżeli ustawimy mały parametr  $d$  (np.  $d=1$ ) lub jeżeli dostarczymy bardzo słabą funkcję heurystyczną.

*Czy jeżeli wiemy że przegramy/zremisujemy (nie wygramy) to czy taka gra może sprawiać przyjemność;)?*

To jest przypadek, w którym chyba lepiej żyć w niewiedzy.