Sztuczna inteligencja i inżynieria wiedzy

Sprawozdanie nr 3

Algorytmy rozwiązywania gier o sumie zerowej

Przemysław Pietrzak, 238083
Środa, 17:05

Spis treści

1 Wstęp

1.1 Opis implementacji

Zaimplementowany program składa się z pięciu głównych części do których należą modele, logika gry, heurystyki wyboru wartości, algorytmy oraz interfejs użytkownika. Modele przedstawiają widok planszy oraz zawierają informacje o znajdujących się w niej polach i możliwych do zdobycia młynkach. Silnik gry następnie wykorzystuje zaimportowany model podczas właściwej rozgrywki do weryfikowania ruchów gracza oraz dokonywania ewaluacji stanu gry. Stany gry są ewaluowane na podstawie heurystyki wartości przypisanej do obecnie aktywnego gracza. Wybrane algorytmy natomiast poprzez wykorzystanie odpowiedniego interfejsu z silnika gry podejmują decyzję o następnym ruchu, a następnie wysyłają tą informację do silnika w celu wykonania ruchu. Interfejs użytkownika został napisany przy pomocy biblioteki JavaFX. Jego zadaniem jest udostępnienie użytkownikowi możliwości zmiany dostępnych opcji jak wybór algorytmu przypisanego do danego gracza i wybór heurystyk poprzez odpowiednie kontrolki

1.2 Opis algorytmów

Badane w następnych etapach algorytmy to min-max oraz alfa-beta. Oba algorytmy są metodami minimalizowania maksymalnych możliwych strat. Drugi z podanych algorytmów jest rozszerzeniem pierwszego, które redukuje liczbę przeszukiwanych węzłów koniecznych do odwiedzienia. Pseudokod obu rozwiązań znajduje się poniżej.

```
// Algorytm min-max
function getbestmove()
    for move in possible_moves
        result = minmax(move, max_depth, true)
        if result > best_result
            best_result = result
            best_move = move
    return move

function minmax(move, depth, maximize)
    make_move(move)

    if depth < 1
        return evaluate()
    if game_finished
        if draw
            return 0
        return winner ? ∞ : -∞

    for next_move in possible_moves
        result = minmax(next_move, depth - 1, !maximize)
        if(maximize)
            best = max(best, result)
        else
            best = min(best, result)
    return best
```

```
// Algorytm alfa-beta
function getbestmove()
    alfa = -∞, beta = ∞
    for move in possible_moves
        alpha = minmax(move, max_depth, true, alfa, beta)
        if alpha > best_result
            best_result = alpha
            best_move = move
    return move

function alfabeta(move, depth, maximize, alfa, beta)
    make_move(move)
    if depth < 1
        return evaluate()
    if game_finished
        if draw
            return 0
        return winner ? ∞ : -∞
    for next_move in possible_moves
        result = alfabeta(next_move, depth - 1, !maximize, alfa, beta)
        if(maximize)
            if(result > best)
                best, alpha = result
        else
            if(result < best)
                best, beta = result
        if(alpha >= beta)
            return best
    return best
```

2 Badanie wpływu algorytmu na czas przetwarzania i liczbę instrukcji

Badanie polega na zmierzeniu czasu rozgrywki oraz liczbę potrzebnych ruchów do uzyskania zwycięstwa dla obu z wykonywanych algorytmów. Badanie przeprowadzone zostało na różnych maksymalnych głębokościach zarówno dla algorytmu Min-Max jak i algorytmu Alfa-Beta. W każdym z badań wykorzystano heurystykę oceny ruchu na podstawie sumy różnicy liczby znajdujących się na planszy pionków oraz różnicy liczby dostępnych ruchów. Wyniki badań przedstawiono w poniższych tabelach.

Tabela 1: Wpływ głębokości na działanie algorytmu Min-Max

Maksymalna głębokość	Czas przetwarzania [s]	Liczba ruchów wygranego	Liczba instrukcji wygranego
1	cell5	celfflf6	celfflf6
2	cellfsdfd8	cell9	cell10
3	cell8	cell9	cell10
4	cell8	cell9	cell10
5	cell8	cell9	cell10
6	cell8	cell9	cell10

Tabela 2: Wpływ głębokości na działanie algorytmu Alfa-Beta

Maksymalna głębokość	Czas przetwarzania [s]	Liczba ruchów wygranego	Liczba instrukcji wygranego
1	cell5	celfflf6	celfflf6
2	cell8	cell9	cell10
3	cell8	cell9	cell10
4	cell8	cell9	cell10
5	cell8	cell9	cell10
6	cell8	cell9	cell10

3 Badanie wpływu heurystyki na czas przetwarzania i liczbę instrukcji

placeHolder

4 Podsumowanie