Raport laboratorium 3

Cel laboratorium

Zadaniem laboratorium jest napisanie programu / skryptu, który buduje drzewo zadanej gry typu Isolation a następnie gra sam ze sobą wykorzystując do tego algorytm minimax. Należało sprawdzić dwa przypadki:

1. jeden z graczy gra w sposób losowy (tzn. nie używa naszego algorytmu) a drugi stara się optymalizować swoje ruchy (random vs minimax)
2. Obaj gracze podejmują optymalne decyzje (minimax vs minimax)

Zasady gry

- każdy z graczy zaczyna z pionkiem, który może ruszyć się o 1 pole na turę (pionowo, poziomo i na ukos)
- gracze wykonują ruchy na przemian, 1 ruch na raz
- gracz nie może postawić pionka na już wcześniej odwiedzonym polu lub na polu przeciwnika - gra kończy się jeżeli przeciwnik nie może wykonać już ruchu

Założenia

- Przyjąłem wersje gry typu Isolation, w której nie występuje remis. Przegrywa gracz, który jako pierwszy nie może wykonać ruchu
- Zdefiniowana przeze mnie heurystyka oblicza w danym stanie planszy ile ruchów może wykonać gracz, a ile ruchów jego przeciwnik. Funkcja zwraca różnice pomiędzy ilością możliwych ruchów u każdego z graczy. Im większy wynik tym stan planszy jest korzystniejszy dla gracza Max i im mniejszy wynik tym korzystniejszy jest stan planszy dla gracza Min.

Biblioteki

- `copy`
- `numpy`
- `random`
- `time`

Badanie zależności

Algorytm minimax służy do wyznaczania optymalnej strategii w grach dwuosobowych. Zależy on od:

1. Zdefiniowanej heurystyki, czy daje ona optymalną strategie. Od niej zależy ocena sytuacji na planszy, czy dany ruch jest korzystny dla gracza Max czy Min.
2. Ustalonej głębokości przeszukiwania, tym samym zmienia się wygląd drzewa gry, po którym przechodzi algorytm. W zależności od głębokości jaką przyjmiemy rozpatrujemy różną ilość potencjalnych sekwencji ruchów
3. Strategii przeciwnika, w zależności od niej przeciwnik może ruszać się w sposób optymalny lub nieoptymalny.

Pomiary minimax vs random

Podczas pomiaru gracz Max korzystał z algorytmu `minimax`, gracz Min wykorzystywał funkcje `chooseRandomMove` , która wybiera losowy ruch z dostępnych w danym momencie. Wszystkie pomiary zostały przeprowadzone dla stałych punktów startowych w dwóch przeciwległych rogach planszy.

ilość gier	win Max	win Min	min - ruchy	max - ruchy	średnia -ruchy	plansza	głębokość
100	82%	18%	45	217	131,45	20x20	5
100	81%	19%	46	217	129,32	20x20	3
100	74%	26%	13	92	56,71	10x10	4
100	70%	30%	13	94	56,44	10x10	3
100	56%	44%	14	43	30,47	7x7	3
100	57	43%	12	46	32,13	7x7	4
100	31%	69%	6	14	10,81	4x4	8
100	59%	41%	8	23	17.39	5x5	8
100	60%	40%	18	46	34,08	7x7	5
100	61%	39%	18	43	33,21	7x7	6
100	68%	32%	22	47	33,88	7x7	8

Wnioski - wpływ głębokości

- Zwiększenie głębokości przeszukiwań poprawia wyniki działania algorytmu minimax, jednak gdy zwiększamy głębokość przeszukiwania dużo bardziej rozrasta się drzewo gry, po którym algorytm przechodzi, zatem znacząco wzrasta czas wykonania programu. Widać to na podstawie trzech ostatnich pomiarów gdy nawet zwiększenie o jeden głębokości przeszukiwania dla tych samych ustawień może wydłużyć czas działania programu nawet o godzinie. (w przypadku implementacji bez obcięcia alfa - beta). Jest więcej możliwości do sprawdzenia, zatem im większa jest głębokość tym program działa dłużej.

Wnioski - wielkość planszy

- Algorytm dużo lepiej działa dla większych plansz, przy stopniowym wzroście rozmiaru plansz widać stopniowe zwiększanie skuteczności algorytmu minimax.
- Dla planszy 4x4 przy głębokości 8, algorytm zwyciężał tylko w 31%. Jest to skrajny przypadek, dla większych plansz algorytm jest stabilny i za każdym razem wygrane gracza używającego algorytm minimax stanowią większość ze wszystkich rozegranych gier.
- Zależność pomiędzy wielkością planszy a działaniem algorytmu jest prawdopodobnie spowodowana zdefiniowaną przeze mnie funkcją oceny i typem gry Isolation. Im więcej jest przestrzeni na planszy tym lepiej algorytm sobie radzi w podejmowaniu decyzji gdzie postawić ruch, ponieważ preferuje miejsca, w których później będzie miał jak najwięcej kolejnych ruchów.

Wnioski - heurystyka

Zdefiniowana przeze mnie heurystyka, oceniająca sytuacje na planszy dokonuje oceny na podstawie różnicy w ilości ruchów gracza Max od ilości ruchów gracza Min w danym momencie.

- Wzrost głębokości jest korzystny dla mojej heurystyki, ponieważ im większa głębokość tym więcej zmian badamy, co za tym idzie funkcja oceny może wskazać bardziej różniące się od siebie wyniki oceny.

- Im większa plansza tym więcej możliwości ruchu, co sprzyja funkcji oceny, która opiera się na badaniu ilości ruchów w danym momencie przez graczy

Pomiary Minimax vs Minimax

Pomiary dla przypadku, w którym oboje graczy korzysta z algorytmu minimax zostały przeprowadzone dla losowych punktów startowych.

ilość gier	win Max	win Min	min - ruchy	max - ruchy	średnia -ruchy	plansza	głębokość gracza Max
50	46%	54%	12	46	30,7	7x7	4
50	58%	42%	12	44	29,53	7x7	2
50	48%	52%	26	97	68,2	10x10	4
50	46%	54%	21	96	70,02	10x10	3
50	40%	60%	18	96	69,72	10x10	2

Wnioski - minimax vs minimax

- W przypadku, w którym obaj gracze korzystają a algorytmu minimax wyniki są zbliżone, w każdej konfiguracji. Wyniki oscylują w okolicach 50% na wygraną każdego z graczy, z przewagą dla gracza posiadającego większą głębokość przeszukiwania.
- W pomiarze zostało zastosowane losowanie punktów startowych zatem również wylosowany punkt w danej grze ma wpływ na jej przebieg. Może się okazać, że wylosowany punkt startowy determinuje dla określonego gracza wygraną lub przegraną.
- Podobnie jak w przypadku poprzednich pomiarów dla rozgrywki minimax vs random, przy zwiększaniu głębokości przeszukiwania tym samym zwiększa się czas działania programu ze względu na rozbudowanie się drzewa gry.

Wnioski - własności algorytmu

- Algorytm minimax działa na podstawie drzewa gry, które przedstawia wszystkie możliwe ruchy i ruchy będące odpowiedzią przeciwnika aż do końca gry lub do wyznaczonej głębokości przeszukiwania, zatem działa biorąc pod uwagę optymalną grę przeciwnika.
- W zależności od dobranej heurystyki należy dobrać odpowiednie parametry wywołania programu aby algorytm działał jak najlepiej. Po wszystkich pomiarach widać, że większe plansze i większe głębokości przeszukiwań są korzystne dla zdefiniowanej przeze mnie funkcji oceny i samego algorytmu.