WSI – Raport z zadania 3

Michał Pędziwiatr

Numer indeksu: 331 421

Wstęp

Algorytm „Minimax”, metoda minimalizowania maksymalnych możliwych strat, to jeden z najważniejszych fundamentów programów specjalizujących się w optymalizacji ruchów w grach logicznych. Jego potencjału możemy dopatrywać się w praktycznie wszystkich grach dwuosobowych o sumie zerowej z pełną informacją, czyli takich w których przewaga jednego gracza oznacza stratę drugiego, a każdy z nich ma przed wykonaniem ruchu pełny dostęp do informacji o obecnym stanie gry.

Przykładem gry, która spełnia wszystkie te wymagania, oraz która dzięki swojej prostocie dobrze i przejrzyście wizualizuje potencjał algorytmu Minimax to gra w „kółko i krzyżyk”. Rozgrywka toczy się na planszy o wymiarach 3x3, na której to gracze próbują na zmianę ułożyć z określonego znaku 3 elementowy szyk, jednocześnie próbując utrudnić to przeciwnikowi.

Zamysł algorytmu

Jak zostało już wspomniane, algorytm Minimax, jest algorytmem znanym z minimalizowania maksymalnych możliwych strat. W praktyce, oznacza to, że w swoim działaniu poza poszukiwaniem najlepszego dostępnego dla siebie ruchu, symulować będzie również ruchy przeciwnika. Podejście to sprawia, że oceniając potencjalny ruch, algorytm może ustalić jego wartość biorąc pod uwagę także najlepsze odpowiedzi przeciwnika. W ten sposób, znając potencjalne kontr-odpowiedzi swoich oponentów, algorytm może zminimalizować straty jakie mogą one spowodować dla jego pozycji wybierając inną dostępną alternatywę.

Parametry algorytmu

Mimo że użycie algorytmu Minimax w kontekście gry w kółko i krzyżyk nie wymaga złożonej konfiguracji parametrów ze względu na prostotę zasad gry,

nie oznacza to, że nie wpływają one na proces optymalizacji oraz wyniki działania algorytmu. Grając samodzielnie przeciwko graczowi komputerowemu używającemu algorytmu Minimax, oraz przydzielając mu przeciwników wykonujących losowe ruchy lub również wykorzystujących algorytm z różnymi parametrami, doszedłem do następujących wartości parametrów:

- głębokość = 6, dająca możliwość symulacji po 3 ruchy w przód na gracza, okazała się kluczową wartością, dzielącą wartości parametru na te, dla których algorytm wykonywał wyłącznie ruchy perfekcyjne, oraz te dla których, mimo wysokiej skuteczności przeciwko graczom losowym pomyłki były liczne. Ponieważ wartość 6 jest dość szczególna w tym kontekście, ze względu na możliwość symulacji wygranej obu graczy, podejrzewam, że wprowadzenie bardziej skomplikowanej metody oceny obecnego stanu gry mogłoby ją nieco obniżyć. W obecnym stanie, dopóki algorytm nie dojdzie za pomocą symulacji w przód do wygranej lub przegranej, będzie on wykonywał pierwszy z dostępnych ruchów, co nie jest optymalne. Ostatecznie jednak po próbach implementacji takiego mechanizmu, które nie zmieniły rażąco optymalności algorytmu, zdecydowałem się na uznaniu tej wartości za akceptowalną oraz zostawienie jej w prostszej i przejrzystszej wersji.
- Wartości oceny stanu gry = {-1, 0, 1}, wydają się tak jak zostało to wspomniane w punkcie dotyczącym głębokości, w pełni wystarczające dla prostszych implementacji algorytmu Minimax. W obecnym stanie zakładając, że przeciwnik również wykonuje najoptymalniejsze ruchy algorytm rozróżnia więc 3 grupy wartości: dające pewną porażkę, dającą pewną wygraną oraz te nie rozstrzygające wyniku rozgrywki.
- Początkowe wartości Alfa = - ∞ oraz Beta = + ∞, posiadają wartości skrajne, mające na celu zobrazować ich cel jakim jest przechowywanie wartości najlepszego oraz najgorszego wynik, aktualizując je używając porównań. W algorytmie Minimax, do poprawnej optymalizacji jego działania, wystarczy by były one większe oraz mniejsze od skrajnych wartości funkcji oceniającej stan rozgrywki, co pozwoli na nadpisanie ich wartości i ewentualne „odcięcie” danej gałęzi rozwiązań w obliczeniach, którą algorytm uzna za niewartą symulacji.

Eksperymenty

Eksperyment 1 : Algorytm Minimax vs Gracz losowy

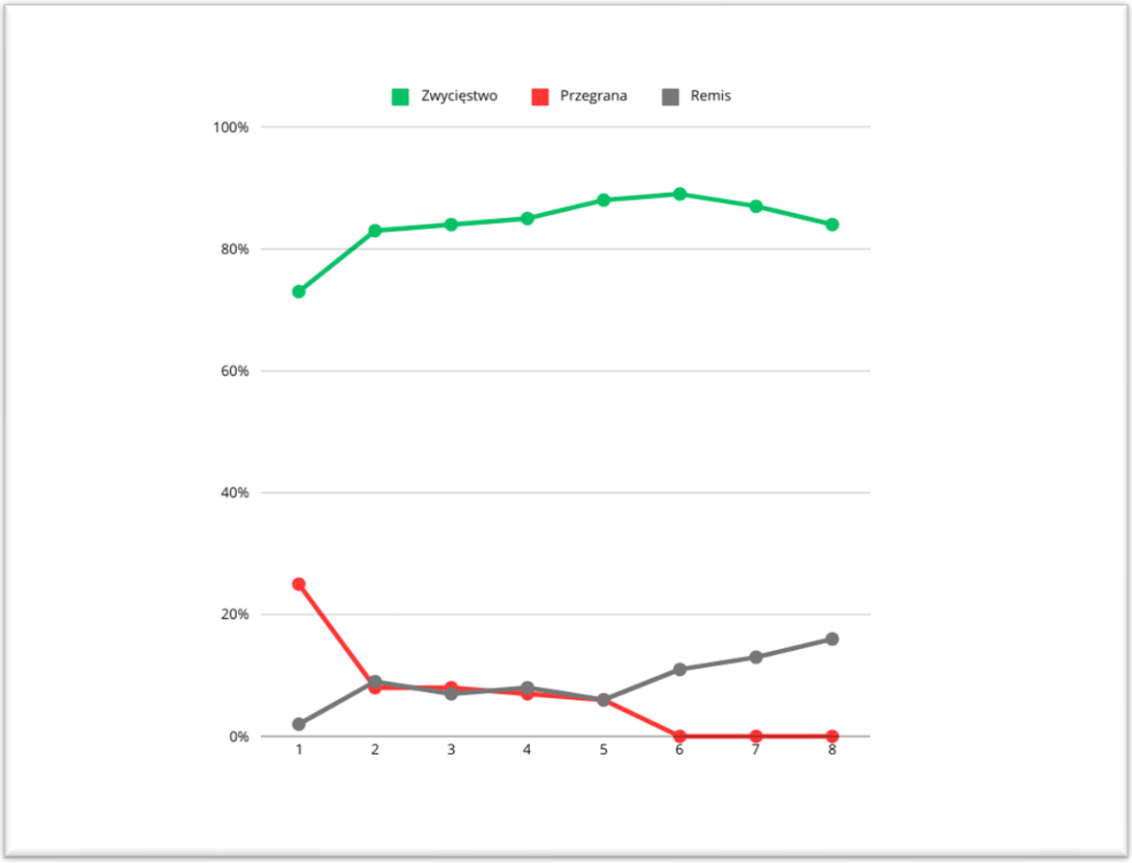
W poniższych eksperymentach:

Krzyżyk – komputerowy gracz losowy

Kółko – komputerowy gracz Minimax, parametr głębokości = N

Gracze zmieniają się stronami co grę.

- N = 1:
```
x wins: 125 / o wins: 361 / draws: 9
```
- N = 2:
```
x wins: 41 / o wins: 408 / draws: 45
```
- N = 3:
```
x wins: 39 / o wins: 419 / draws: 36
```
- N = 4:
```
x wins: 28 / o wins: 341 / draws: 34
```
- N = 5:
```
x wins: 21 / o wins: 321 / draws: 21
```
- N = 6:
```
x wins: 0 / o wins: 436 / draws: 56
```
- N = 7:
```
x wins: 0 / o wins: 210 / draws: 30
```
- N = 8:
```
x wins: 0 / o wins: 42 / draws: 9
```



Oś pozioma – parametr głębokości odcinania algorytmu minima

Eksperyment 2 : Gracz Minimax vs Gracz Minimax

W poniższych eksperymentach:

Krzyżyk – komputerowy gracz Minimax, parametr głębokości = N

Kółko – komputerowy gracz Minix, parametr głębokości = M

Gracze zmieniają się stronami co grę.

- N = 6, M = 9
  ```
  x wins: 0 / o wins: 0 / draws: 200
  ```
- N = 5, M = 9
  ```
  x wins: 0 / o wins: 54 / draws: 55
  ```
- N = 1, M = 6
  ```
  x wins: 0 / o wins: 53 / draws: 0
  ```
- N = 3, M = 4
  ```
  x wins: 0 / o wins: 62 / draws: 62
  ```
- N = 6, M = 5
  ```
  x wins: 42 / o wins: 0 / draws: 42
  ```

Eksperyment 3: Gracz Minimax vs Gracz Minimax

W poniższych eksperymentach:

Krzyżyk – komputerowy gracz Minimax, parametr głębokości = N

Kółko – komputerowy gracz Minix, parametr głębokości = M

Zaczyna gracz grający jako krzyżyk.

- N = 6, M=6

```
x wins: 0 / o wins: 0 / draws: 65
```

- N = 5, M = 6

```
x wins: 0 / o wins: 0 / draws: 75
```

- N = 5, M = 1

```
x wins: 140 / o wins: 0 / draws: 0
```

- N = 4, M = 1

```
x wins: 132 / o wins: 0 / draws: 0
```

Eksperyment 4: Gracz Minimax vs Gracz Losowy

W poniższych eksperymentach:

Krzyżyk – komputerowy gracz Minimax, parametr głębokości = N lub gracz losowy (oznaczony jako N = R)

Kółko – komputerowy gracz Minix, parametr głębokości = M lub gracz losowy (oznaczony jako M = R)

Zaczyna gracz grający jako krzyżyk.

- N = R, M = 6

```
x wins: 0 / o wins: 101 / draws: 17
```

- N = 6, M = R

```
x wins: 122 / o wins: 0 / draws: 0
```

- N = R, M = 3

```
x wins: 26 / o wins: 150 / draws: 35
```

- N = 3, M = R

```
x wins: 174 / o wins: 1 / draws: 5
```

- N = R, M = R

```
x wins: 294 / o wins: 135 / draws: 67
```

Wnioski

Wpływ głębokości na przebieg rozgrywki

Tak jak już to zostało wspomniane w punkcie „Parametry algorytmu”, wartość 6 wydaje się po przeprowadzeniu testowych symulacji wartością graniczną (dla gier ze zmiennym graczem rozpoczynającym), od której wzwyż algorytm przestaje przegrywać, niezależnie od swojego przeciwnika. Fakt, że aby algorytm był w pełni efektywny, potrzebuje on symulacji po 3 ruchów na gracza, sugeruje iż istotne jest, już na początku rozgrywki, przeanalizowanie swoich najszybszych możliwych zwycięstwo oraz porażek. W przeciwnym razie, nie może on skutecznie ocenić swoich dostępnych ruchów. Dopóki rozgrywka więc nie zbliży się do wygranej jednej ze strony, będzie on wykonywał przewidywalne ruchy po kolei.

Co ciekawe wartość 6 okazuje się również najoptymalniejszym wyborem biorąc pod uwagę nie tylko optymalizację czasu kompilacji, ale także i wyników. W wykonanych pomiarach, algorytm mimo dalszego skutecznego omijania błędów doprowadzających do przegranej, zdaje się radzić sobie minimalnie gorzej dla głębokości równej 7 oraz 8. Mogłoby to oznaczać, że algorytm dla głębokości 6 wygrywa wykorzystując słabości przeciwnika, a dla wyższych głębokości gra bardziej zapobiegawczo, idąc na remis. Według tego jednak, algorytm o głębokości równej 6 nie powinien sobie radzić z dokładniejszymi algorytmami. Konfliktuje to więc bezpośrednio z faktem iż , jak widzimy w eksperymencie 2, gdy zestawimy ze sobą algorytm z parametrem głębokości równym 6 oraz taki z głębokością równą 9, rozgrywki ich kończą się jedynie remisami. Należy jednak pamiętać, że różnica pomiędzy działaniem algorytmu dla głębokości = 6 oraz dla głębokości równej 8 jest minimalna, dlatego też może ona wynikać po prostu z losowego błędu pomiarowego.

Wpływ bycia graczem rozpoczynającym na przebieg rozgrywki

Bycie graczem rozpoczynającym w grze w kółko i krzyżyk ma olbrzymie znaczenie na przebieg rozgrywki i potencjalne najoptymalniejsze dostępne ruchy graczy. Dowodem tego, są pomiary z eksperymentu 3 oraz 4, według których gdy zestawimy ze sobą dwóch graczy losowych, ten zaczynający będzie wygrywał średnio aż 59% razy, przegrywając ich tylko 27%. W przypadku algorytmów używających algorytm Minimax, różnica ta może być

jeszcze większa. Ze względu na mechanizm działania głębokości w algorytmie, można stwierdzić, że gracz zaczynający ma przewagę jednego punktu głębokości. Dzieje się tak, ponieważ do ilości symulowanych potencjalnych ruchów narzuconych przez głębokość, dochodzi także ruch obecny (zaczynający). Sprawia to więc, że algorytm o np. głębokości równej 5, gdy zaczyna rozgrywkę, grać będzie na poziomie niezaczynającego algorytmu o głębokości równej 6. Widzimy to m.in. w wynikach eksperymentu, gdzie w potyczce zawsze zaczynającego algorytmu o głębokości równej 5, z algorytmem o głębokości równej 6 zawsze remisują, dokładnie tak jak w potyczce 2 algorytmów o głębokości równej 6, kiedy grają zaczynając na zmianę. Na podobny wniosek wskazuje fakt, iż grając na zmianę, algorytm o głębokości równej 5 remisuje w połowie gier z algorytmem o głębokości równej 6 (ponieważ można powiedzieć, że co drugą grę „gra jak algorytm o głębokości równej 6”).