Algorytm α-β z iteracyjnym pogłębianiem w grze Neutron

Marcin Kaczyński Jakub Kitaj Michał Proc

6 czerwca 2012

1 Cel projektu

Celem projektu było zaimplementowanie gry planszowej *Neutron*. Do prowadzenia rozgrywki zaimplementowano algorytm α-β z iteracyjnym pogłębianiem uwzględniający ograniczenie czasowe. Celem było również wymyślenie i implementacja funkcji oceny stanu gry.

2 Reguły gry

Neutron jest ogólnie turową, strategiczną grą dwuosobową o sumie zerowej. W każdej turze gracz porusza dwoma różnymi pionkami wg określonych zasad - Neutronem, oraz pionkiem należącym do tego gracza. Gra jest rozgrywana na planszy o rozmiarach 5x5. Każdy z graczy dysponuje 5 pionkami ustawionymi początkowo w polu bazowym (pierwszy lub ostatni wiersz na planszy).

Celem gry jest przesunięcie Neutronu do swojego pola bazowego, lub takie zablokowanie przeciwnika, aby nie mógł wykonać ruchu zgodnego z zasadami gry.

Poniższe punkty określają szczegółowo warunki i reguły gry:

1. 5 pionków każdego gracza jest początkowo ustawionych w polu bazowym gracza.
2. Jeden z graczy gra pionkami białymi, drugi czarnymi (w praktyce kolory mogą być dowolne).
3. Pionki mogą się poruszać wzdłóż wolnych wierszy, kolumn lub po diagonaliach planszy. Pionki muszą poruszać się tak daleko jak jest to możliwe.
4. Początkowo Neutron jest ustawiony na środku planszy. Jest poruszany przez obu graczy, rusza się wg tych samych zasad co pionki.
5. Pierwszy gracz wykonuje ruch w którym porusza tylko pionek. W każdym następnym ruchu gracze muszą w pierwszej kolejności wykonać ruch Neutronem w drugiej pionkiem.

3 Teoria

3.1 Algorytm *MIN-MAX*

MIN-MAX jest ogólnie algorytmem wykorzystywantym w grach dwuosobowych. Zasada działania opieraja się na analizie wszystkich możliwych sekwencji ruchu od tyłu. Analizie towarzyszy założenie, że obaj gracze wybierają ruchy najbardziej korzystne dla siebie. W obec tego każdemu kolejnemu stanowi przypisywana jest ocena najlepszego (w przypadku gracza) lub najgorszego (w przypadku przeciwnika) stanu, do którego prowadzi ruch. Ostatecznie w stanie gry, w którym ta analiza jest prowadzona wybierany jest ruch prowadzący do najlepszego (gracz) lub najgorszego (przeciwnik) kolejnego stanu.

3.2 Algorytm α-β

Algorytm α-β jest prostą modyfikacją *MIN-MAX* dzięki której można obniżyć koszt obliczenia kolejnego ruchu.

Algorytm opiera się na następujących obserawcjach:

- Jeżeli w przeanalizowanej części drzewa jest wybór który jest lepszy dla gracza niż przeciwnik ma zagwarantowany w innym poddrzewie, to nie ma sensu analizować dalej tego poddrzewa.
- Jeżeli w przeanalizowanej części drzewa jest wybór korzystniejszy dla przeciwanika niż gracz ma zagwarantowany w innym poddrzewie, to dalsza analiza poddrzewa nie ma sensu.

Ogólnie na podstawie tych definicji można wprowadzić następujące definicje:

Definicja 1 α *- wartość którą gracz może uzyskać wykonując najlepsze dla siebie, wcześniej analizowane ruchy.*

Definicja 2 β *- wartość którą przeciwnik może uzyskać wykonując najlepsze dla siebie, wcześniej analizowane ruchy.*

Analizowanie poddrzewa ma sens dopóki $\alpha < \beta$. Jeżeli warunek nie jest spełniony, to rozgrywający może wybrać co najwyżej równie dobry dla siebie ruch, a więć można pominąć analizę poddrzewa.

3.3 Iteracyjne pogłębianie

Iteracyjne pogłębianie polega na wielokrotnym, coraz głębszym przeszukiwaniu grafu gry, dopóki obliczenia nie przekraczają zdefiniowanego wcześniej czasu na wykonanie ruchu. Takie podejście pozwala na dostosowanie głębokości do czasu którym dysponujemy na wykonanie ruchu. Za każdym wywołaniem $\alpha - \beta$ następniki z pierwszego poziomu są sortowane, tak aby najbardziej obiecujące węzły były rozpatrywane na początku. Takie podejście zwiększa ilość β-cięć w kolejnych wywołaniach $\alpha - \beta$.

3.4 Zaimplementowane heurystyki

3.5 Heurystyka H1

Zastosowany algorytm umożliwia ocenę sytuacji na planszy z wyróżnieniem pozycji kończących grę z sukcesem dla aktualnego gracza. Jest to sytuacja najwyżej oceniana. W przypadku kiedy ruch zakończyłby się przegraną, wystawiana jest ocena najniższa. Następnie w kolejności od najlepiej ocenianych są następujące sytuacje. Zablkokowanie możliwości ruchu pionków przeciwnika. Kończy to jednocześnie grę z wygraną dla gracza. Sprawdzenie możliwości osiągnięcia przez neutron naszej linii startowej. Pozytywnie oceniane gdyż może się przyczynić do wygranej. Również najniżej oceniana jest możliwosci osiągnięcia linii przeciwnika przez neutron. Utworzenie takiej możliwość mogłoby zakończyć grę, dlatego taki ruch należy eliminować.

3.6 Heurystyka H2

Zastosowany algorytm łączy w sobie ocenę ruchów Heurystyki H1 oraz dodatkowo premiuje ruchy przybliżające neutron do naszej pozycji startowej. Przydatne jest to w przypadku kiedy żadna z innych możliwości oceny ruchu nie została zakwalifikowana i może się przyczynić do wygranej.

3.7 Heurystyka H3

Trzecia zaproponowana heurystyka opiera się jedynie na ocenie tego, czy neutron w danym ruchu znajduje się w polu bazowym.

3.8 Heurystyka H4

Zawiera elementy poprzednich algorytmów i dodatkowo premiuje ruchy, które ograniczają możliwość ruchu neutronu dla przeciwnika. Jeszcze wyżej oceniane są ruchy, które ograniczają możliwość ruchu dla przeciwnika a dodatkowo jednym z tych ruchów jest ruch w stronę naszej linii.

3.9 Heurystyka H5

Jest odmianą Heurystyki H4. Wysoko premiuje ruchy ograniczające ruchy neuronu dla przeciwnika oraz posiadające ruch w stronę naszej linii. Jednak wyżej niż samo ograniczenie ruchu przeciwnika premiowane są sytuacje gdzie neutron może dotrzeć do naszej linii.

3.10 Heurystyka H6

Jest ulepszoną odmianą Heurystyki H4, dodatkowo wspierając te ruchy, w których neutron znajduje się jak najbliżej naszej linii.

4 Wnioski

- Na podstawie przeprowadzonych prostych testów, wydaje się, że dla gry tego rodzaju, dodawanie warunków takich jak blokowanie przeciwnika nie ma większego sensu. Przestrzeń gry jest na tyle mała, że skuteczne zablokowanie praktycznie nie ma szans nastąpić, chyba, że drugi z przeciwników nie będzie dążył do wygranej przez postawienie Neutronu w swoim polu. Z tego powodu zaimplementowane heurystyki często grają bardzo podobnie. Różnice pojawiają się w drzewie w początkowych ruchach, ale są szybko zastępowane przez lepiej oceniane decyzje.

- Proste testy pokazują, że algorytm może nie radzić sobie z przeciwnikiem który nie spełnia założenia o maksymalizacji własnego ruchu. Często wykonanie losowego ruchu wprowadza duże zamieszanie w rozgrywce.

- Na podstawie obserwacji wydaje się także, że algorytm nie powinien stanowić problemu dla doświadczonego gracza. Być może, duże zwiększenie głebokości przeszukiwania znacznie poprawiło by poziom rozgrywki, nie mniej jednak głębokość na którą algorytm schodzi w akcptowalnym czasie (5), nie odznacza się wyjątkową jakością rozgrywki.

- W przygotowanych heurystykach problem może być małe zróżnicowanie wartości ocen planszy. Generalnie powoduje to, że często wybieramy ruch który potencjalnie niewiele wnosi, a gdyby przeszukać drzewo głębiej może się wręcz okazać, że wkonanie tego ruchu prowadzi do przegranej.

- Być może warto rozważyć wprowadzenie pewnej losowości w algorytmie. W zaproponowanej implementacji, jako ruch do wykonania jest wybierany pierwszy o maksymalnej ocenie. Mając na uwadze fakt, że plansze są generowane w sposób deterministyczny, algorytm może mieć tendencje do mało zróżnicowanych rozgrywek.