Projektowanie algorytmów i metody sztucznej

inteligencji

Kanstantsin Zhuk 250934

**Projekt 3**

**Algorytm MinMax z cięciami alfa-beta - warcaby**

1 Wstęp[1]

Algorytm MinMax wywodzi się z twierdzenia o grze o sumie stałej. Oznacza to, że jeśli dwie osoby grają przeciwko sobie, to poprawa sytuacji jednego z graczy oznacza proporcjonalne pogorszenie się sytuacji gracza drugiego. Dzięki tej informacji można przy użyciu odpowiedniej funkcji heurystycznej przypisać danej sytuacji na planszy konkretną wartość (dodatnią, jeśli jest korzystna dla gracza, dla którego ją rozpatrujemy lub ujemną w przeciwnym wypadku).

2 Opis tworzonej gry

Do przechowania ruchów zaimplementowałem dwie struktury: Coords, zawierającą współrzędne piona przed i po wykonaniu ruchu oraz Move, zawierającą wszystkie koordynaty dla danego ruchu

(kiedy składa się na przykład z 3 bić).

Graficzna reprezentacja gry została wykonana przy użyciu biblioteki SFML 2.5.1

**Zasady gry:**

* Bicie jest obowiązkowe
* Piony mogą poruszać się tylko do przodu, zaś bicie dozwolone we wszystkich kierunkach
* Kolejne bicia jednym pionem są obowiązkowe
* Pion, który doszedł do ostatniego rzędu, staje się damką
* Jeśli pion po wykonaniu bicia znalazł się na ostatnim rzędzie, ale dalej posiada bicie, to damką nie staje się
* Koniec gry następuje kiedy któryś z graczy nie będzie w stanie zrobić kolejny ruch
* Pat następuje, kiedy obydwa graczy wykonają po 15 ruchów damkami bez zmiany ilości figur na planszy

**Techniki SI**

Dla drzewa przeszukań, zawierającego wszystkie możliwe stany planszy dla n-następnych ruchów, zaimplementowałem klasę pojedynczego elementu takiego drzewa. W konstruktorze przyjmuje on obiekt planszy, maksymalną głębokość drzewa, kolor, którym gra gracz oraz jako argument opcjonalny przyjmuje on wykonany ostatnio ruch.

Aby utworzyć drzewo należy utworzyć jego korzeń tj. pierwszy element, który otrzymuje aktualną plansze gry. Do pierwszego elementu nie podajemy jaki ruch został wykonany, ponieważ znajdzie

się w nim ruch wybrany przez algorytm Min-max.

Konstruktor elementu jest funkcją rekurencyjną, która wywołuje się, aż do osiągnięcia maksymalnej głębokości drzewa, lub gdy po wykonanym ruchu następuje zakończenie gry. Konstruktor pobiera z planszy wszystkie możliwe do wykonania dla danego gracza ruchy, następnie dla każdego ruchu tworzy potomka, któremu przekazuje: ruch, obiekt planszy po wykonaniu danego ruchu, oraz kolor pionków, które będą wykonywały ruch następne.

Jeśli zostanie spełniony jeden z warunków zakończenia pogłębiania drzewa, wywołana zostaje funkcja heurystyczna, oceniająca ostateczną sytuacje na planszy.

**Funkcja Heurystyczna**

Funkcja heurystyczna pozwala na przybliżenie rozwiązania danego problemu bez zapewnienia gwarancji poprawności uzyskanego wyniku. Używa się jej wtedy, gdy właściwy algorytm jest zbyt złożony, kosztowny lub nieznany.

Od implementacji tej funkcji zależy efektywność całego algorytmu wybierania najlepszych ruchów. Oczywistym czynnikiem jest ilość posiadanych pionów oraz damek. Starałem się punktować dodatkowo piony ’bezpieczne’, tj. takie, które nie mogą zostać zbite (przylegają do bocznej ściany planszy). Dodatkowe punkty przyznaję też za wyprowadzenie piona z rzędów początkowych, ponieważ uważam, że piony poruszające się w bardziej zwartej formacji są efektywniejsze. Podstawową mechaniką warcabów jest wykonywanie zbić, dlatego każdy pion z możliwością bicia również otrzymuje dodatkowe punkty.

Zastosowany system przydzielania punktów za określone czynniki na planszy:

* Wygrana: 10000 pkt
* Przegrana: -10000 pkt
* Remis: 0 pkt
* Pion: 7 pkt
* Damka: 25 pkt
* Figura przy bocznej ścianie planszy: +2 pkt
* Figura oddalona o 1 pole od bocznej ściany: +1 pkt
* Pion w rzędzie dalszym niż 1: +2 pkt
* Pion w rzędzie dalszym niż 2: +1 pkt
* Pion w rzędzie dalszym niż 3: +1 pkt
* Figura posiada możliwość bicia: +15 pkt

**Algorytm MinMax:**

Algorytm ten jest metodą wybierania ruchu, który pozwala na minimalizację możliwych strat lub maksymalizację zysków. Jeśli dla danej sytuacji na planszy stworzymy drzewo wszystkich możliwych układów dla n-następnych ruchów i każdej przypiszemy wartość przy użyciu określonej funkcji heurystycznej, to możemy wybrać taki ruch, który daje nam największe korzyści. Gracz, dla którego

wywołany jest w/w algorytm będzie dążył do maksymalizacji zysków, natomiast przeciwnik będzie się starał te zyski zminimalizować.

Algorytm MinMax dostaje jako parametr korzeń drzewa stanów. Następnie sprawdza, czy aktualny gracz dąży do maksymalizacji, czy do minimalizacji zysków i zależnie od tego wybiera ruch o największej / najmniejszej wartości. Aby to uzyskać, wywołuje się rekurencyjnie aż do osiągnięcia warunku podstawowego, dzięki któremu może zwrócić konkretną wartość.

Jeśli założymy, że dla każdego ruchu można wykonać n-następnych, to złożoność obliczeniowa algorytmu wyniesie O(), gdzie m to maksymalna głębokość rekurencji. W warcabach liczba możliwych ruchów jest zmienna, więc nie da się dokładnie określić złożoności obliczeniowej algorytmu.

**Cięcia Alfa-Beta**

Cięcia αβ nie są osobnym algorytmem, a raczej modyfikacją do algorytmu MinMax. Zakładają one dodanie dwóch zmiennych, które przechowują minimalną – β i maksymalną – α wartość, jakie MinMax obecnie może zapewnić na danej głębokości drzewa lub wyżej. Przy starcie algorytmu α = −∞ i β = ∞ , wartości te są korygowane w dalszych etapach działania algorytmu. Na ich podstawie, jeśli przy sprawdzaniu kolejnego poddrzewa węzła minimalizującego α ≥ β algorytm może odciąć dane poddrzewo (w tym przypadku cięcie β ), ponieważ już wie, że maksymalna wartość tego poddrzewa przekroczy minimalną, którą węzeł może obecnie zagwarantować. Bardzo korzystnie wpływa to na złożoność algorytmu, pozwalając osiągnąć dużo lepszy czas wyszukiwania optymalnego ruchu lub zwiększyć głębokość rekursji. Przy identycznych założeniach, jak wspomniane w opisie algorytmu MinMax, złożoność obliczeniowa algorytmu wynosi O().

**Literatura:**

* Minimax algorithm en.wikipedia https://en.wikipedia.org/wiki/Minimax
* Michał Mielnicki – Funkcja oceniająca do algorytmu gry w warcaby https://kcir.pwr.edu.pl/~witold/aiarr/2009\_projekty/warcaby/
* Evolutionary-based heuristic generators for checkers and give-away cheekers, Jacek Mańdziuk, Magdalena Kusiak and Karol Walędzik https://www.semanticscholar.org/paper/Evolutionary-based-heuristic-generators-for-and-Mańdziuk-Kusiak/91c9d140267f3b008d00b330b6b0e9182fa4b62e
* Geedx4geeds Minimax algorithm in game theory https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/