УДК 681.12

М. С. Шестаков
І. Р. Арсенюк

ПІДХІД ДО ГРИ У ШАХИ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

Вінницький національний технічний університет

Анотація

Запропоновано підхід щодо прорахунку ходів шахових фігур, яка побудована на основі Minimax алгоритму з використанням Альфа-Бета відсікання. Програмний додаток гри розроблено на мові програмування C# у програмному середовищі Visual Studio 2022 та Unity 2021.2.4f. Запропонована технологія дозволяє скоротити час прорахунку ходів шахових фігур на 5 – 8 %.

Ключові слова: інформаційна технологія, глибоке прогнозуваня, шахи.

Abstract

The approach of calculating the moves of chess pieces is proposed, which is built on the basis of the Minimax algorithm with the alpha-beta clipping. The software implementation is developed in the C# programming language in the Visual Studio 2022 and Unity 2021.2.4f software environment. The proposed system allows to reduce the time of calculating the moves of chess pieces by 5 – 8 %.

Key words: information technology, deep prediction, chess.

Вступ

Шахи це гра з повною інформацією [1]. Цей теоретико-ігровий термін позначає гру, де гравцям відомі функція корисності, правила гри, а також ходи інших гравців. Задачею даного дослідження є підхід до гри у шахи, що передбачає глибокий прорахунок ходів шахових фігур у реальному часі.

Якщо ні в яких аспектах гри (правилах, можливості або черговості ходів, визначенні моменту завершення гри тощо) не має місця елементу випадковості, така гра буде ще і детермінованою.

Для будь-якої детермінованою гри з повною інформацією, теоретично, можна прорахувати все дерево можливих ходів гравців і визначити послідовність ходів, яка гарантовано призведе, принаймні одного з них, до виграшу або нічиєї. Тобто, завжди може бути побудовано алгоритм виграшу або зведенню гри у нічию.

До ігор з повною інформацією відноситься більшість детермінованих настільних ігор (наприклад, шахи, таврелі, шашки, го, рендзю, сянці, сьогі, хрестики-нулики, реверсі, точки). Проте, для більшості з них, алгоритм виграшу (або гарантованою нічиєї) невідомий: хоча теоретично він існує і може бути знайдений, на практиці дерево варіантів занадто велике, щоби його можна було побудувати та проаналізувати за прийнятний час.

Основна частина

Для розробки даної програми будемо враховувати нижченаведене.

Для того, щоби зрозуміти, яка сторона сильніша у певній позиції можна підрахувати відносну міцність фігур на дошці за допомогою таблиці 1.

Далі створимо дерево пошуку, з якого алгоритм зможе вибрати найкращий хід. Для цього можна використати алгоритм Minimax який створений на базі розгорткової нейронної мережі (дана нейромережа дозволяє проаналізувати усі можливі ходи шахових фігур до певної глибини) [2]. У цьому алгоритмі рекурсивне дерево всіх можливих ходів досліджується на задану глибину, а позиція оцінюється у кінцевих «листках» дерева.

Ефективність мінімаксного алгоритму значною мірою залежить від глибини пошуку, яку ми можемо досягти.

Альфа-бета відсікання [3] – це метод оптимізації мінімаксного алгоритму, який дозволяє не враховувати деякі гілки дерева пошуку. Це допомагає оцінити мінімаксне дерево пошуку набагато глибше, використовуючи при цьому ті самі ресурси. Альфа-бета відсікання передбачає можливість

припинити оцінювання частини дерева пошуку, якщо знайдемо хід, що призводить до гіршої ситуації, ніж раніше виявлений хід.

Таблиця 1 – цінності шахових фігур

Доцільно зазначити, що альфа-бета відсікання не впливає на результат мінімаксного алгоритму, а лише робить його швидшим. Альфа-бета алгоритм також є ефективнішим, якщо спочатку відвідати ті шляхи, які ведуть до «хороших» ходів.

Функція початкової оцінки досить наївна, оскільки ми враховуємо лише ту інформацію, що є на дошці. Щоби покращити це, ми додаємо до оцінки фактор, який враховує положення фігур на гральній дошці як на таблиці 2. Наприклад, кінь у центрі дошки краще (оскільки має більше варіантів і, отже, більш активний), ніж лицар на краю дошки. Тобто, чим більший коефіцієнт у таблиці, тим фігура має кращу позицію на дошці.

Таблиця 2 – Фактор оцінки положення фігур

```
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0],
[ -1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0],
[  2.0,  2.0,  0.0,  0.0,  0.0,  0.0,  2.0,  2.0 ],
[  2.0,  3.0,  1.0,  0.0,  0.0,  1.0,  3.0,  2.0 ]
```

```
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -0.5,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[  0.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[ -1.0,  0.5,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0]
```

```
[  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[  0.5,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[  0.0,   0.0, 0.0,  0.5,  0.5,  0.0,  0.0,  0.0]
```

```
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  1.0,  1.0,  0.5,  0.0, -1.0],
[ -1.0,  0.5,  0.5,  1.0,  1.0,  0.5,  0.5, -1.0],
[ -1.0,  0.0,  1.0,  1.0,  1.0,  1.0,  0.0, -1.0],
[ -1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0, -1.0],
[ -1.0,  0.5,  0.0,  0.0,  0.0,  0.0,  0.5, -1.0],
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]
```

```
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0],
[-4.0, -2.0,  0.0,  0.0,  0.0,  0.0, -2.0, -4.0],
[-3.0,  0.0,  1.0,  1.5,  1.5,  1.0,  0.0, -3.0],
[-3.0,  0.5,  1.5,  2.0,  2.0,  1.5,  0.5, -3.0],
[-3.0,  0.0,  1.5,  2.0,  2.0,  1.5,  0.0, -3.0],
[-3.0,  0.5,  1.0,  1.5,  1.5,  1.0,  0.5, -3.0],
[-4.0, -2.0,  0.0,  0.5,  0.5,  0.0, -2.0, -4.0],
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]
```

```
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0],
[1.0,  1.0,  2.0,  3.0,  3.0,  2.0,  1.0,  1.0],
[0.5,  0.5,  1.0,  2.5,  2.5,  1.0,  0.5,  0.5],
[0.0,  0.0,  0.0,  2.0,  2.0,  0.0,  0.0,  0.0],
[0.5, -0.5, -1.0,  0.0,  0.0, -1.0, -0.5,  0.5],
[0.5,  1.0, 1.0,  -2.0, -2.0,  1.0,  1.0,  0.5],
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0]
```

Програмне забезпечення прорахунку ходів шахових фігур розроблено на мові програмування C# у програмному середовищі Visual Studio 2022 та Unity 2021.2.4f. Тестування основних функціональних можливостей програми показало, що даний підхід дозволяє скоротити час прорахунку ходів шахових фігур на 5 – 8 %.

Висновки

Запропоновано підхід щодо прорахунку ходів шахових фігур. Підхід передбачає використання алгоритму Minimax (у якому рекурсивне дерево всіх можливих ходів досліджується на задану глибину, а позиція оцінюється у кінцевих «листках» дерева) та альфа-бета відсікання (метод оптимізації мінімаксного алгоритму, який дозволяє не враховувати деякі гілки у дереві пошуку). Це допомагає оцінити мінімаксне дерево пошуку набагато глибше, використовуючи, при цьому, ті самі ресурси. На базі запропонованого підходу розроблено програмний додаток, що дозволяє скоротити час прорахунку ходів шахових фігур на 5 – 8 %.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Котов А. А. Как стать гроссмейстером / Издательство: Russian Chess House / Русский Шахматный Дом. – 2007. – 296 с.
2. Sandhya Samarasinghe Neural Networks for Applied Sciences and Engineering. From Fundamentals to Complex Pattern Recognition. – 2007. – 570 p.
3. Томас Эрл Основы Big Data: концепции, алгоритмы и технологии. – 2018. – 320 p.
4. Очеретний А. Б. Аналіз та порівняння методів покращення контрастності зображення шахових фігур. / А. Б. Очеретний, І. Р. Арсенюк // Науковий журнал “Альманах науки” № 6 (51) листопад 2021. – С 70 – 73

Шестаков Максим Сергійович **–** студент групи 2КН-20м, факультет інформаційних технологій та комп’ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, email: maxsh284@gmail.com

Арсенюк Ігор Ростиславович – к. т. н., доцент кафедри комп’ютерних наук, Вінницький національний технічний університет, м. Вінниця

Shestakov Max S. – Department Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, email: maxsh284@gmail.com

Arseniyk Igor R. – Cand. Sc. (Eng.), Assistant Professor of the Chair of Computer Science, Vinnytsia National Technical University, Vinnytsia