ШАХМАТНЫЙ ДВИЖОК

Гудков А. С., Навроцкий А. А.
Кафедра информационных технологий автоматизированных систем, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: gudkou_fitu@mail.ru

Рассматривается алгоритм создания шахматного движка, позволяющий эффективно рассчитывать наилучшие ходы в шахматной партии.

Введение

В настоящее время становятся популярными компьютерные шахматные программы. Для реализации этих программ используются шахматные движки, которые позволяют на основе анализа шахматных позиций возвращать наиболее оптимальные варианты ходов.

I. Принцип работы шахматного движка

Создание шахматной программы включает несколько этапов:

- визуализация доски;
- перемещение фигур;
- оценка доски;
- дерево поиска;
- Alpha-Beta;
- улучшенная функция оценки.

Первые два этапа реализуют оболочку программы (интерфейс, обеспечивающий взаимодействие пользователя с программой). Они легко осуществляются, например, с помощью JavaScript фреймворка Vue.js. Остальные этапы представляют собой алгоритм вычисления следующего хода (шахматный движок). Исследуя текущую позицию, движок просчитывает большое количество возможных вариантов шахматных ходов, численно оценивая каждый вариант и представляя наилучший результат пользователю. Огромное количество комбинаций, которые необходимо перебрать программе существенно замедляет работу движка, поэтому большой интерес представляют разработки критерием отсечения заведомо невыигрышных шагов.

II. Оценка доски

Общее количество уникальных партий в шахматы превышает количество атомов во Вселенной, отчего невозможно найти наилучший ход простым перебором возможных вариантов позиций. Поэтому в шахматных движках используются алгоритмы оценки позиций и деревьев поиска возможных ходов.

Для оценки расстановки силы на шахматной доске, в самом простом случае можно рассчитать относительную силу фигур, установив стоимость для каждой фигуры (пешка – 1, конь и офицер – 3, тура – 5, ферзь – 9, а король – 900, т. к. его утрата приводит к проигрышу партии). Положим для белых цену фигур положительной, а для черных – отрицательной. Исходная матрица ходов представлена на рисунке 1.

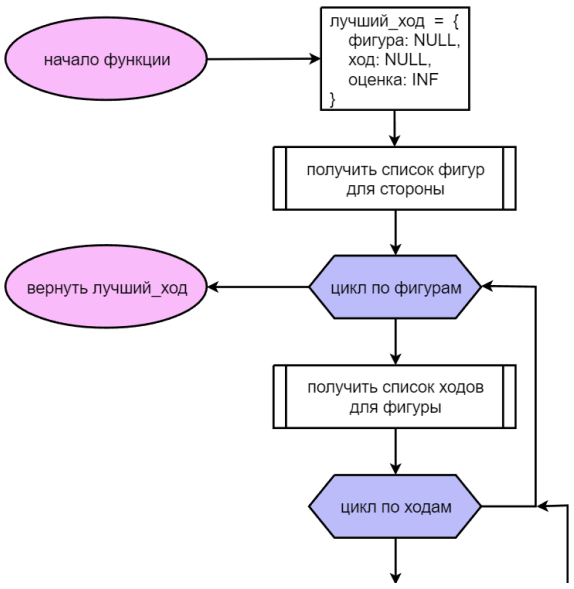
```
const BOARD = [
    [-5, -3, -4, -9, -900, -4, -3, -5],
    [-1, -1, -1, -1,  -1,  -1, -1, -1],
    [ 0,  0,  0,  0,   0,   0,  0,  0],
    [ 0,  0,  0,  0,   0,   0,  0,  0],
    [ 0,  0,  0,  0,   0,   0,  0,  0],
    [ 0,  0,  0,  0,   0,   0,  0,  0],
    [ 1,  1,  1,  1,   1,   1,  1,  1],
    [ 5,  3,  4,  9,  900,  4,  3,  5]
];
```

Рис. 1 – Матрица ходов

Предложенная оценка доски позволяет выбирать ход с максимальной оценкой.

III. Дерево поиска

Поиск оптимального хода для выигрыша в шахматной партии требует расчета ситуации на несколько ходов вперед. Для этого создаётся дерево поиска, анализирующее все возможные ходы до заданной глубины. Полученная оценка возвращается в родительский узел. Блок-схема дерева поиска представлена на рисунке 2. В жёлтом блоке происходит вызов аналогичной рекурсивной функции поиска, для создания новой ветви в дереве.



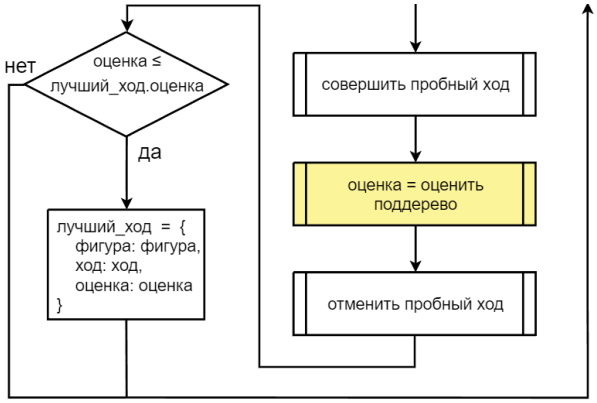


Рис. 2 – Блок-схема корня дерева поиска

С деревом поиска алгоритм начинает понимать базовую тактику шахмат и уже способен не только составить конкуренцию, но и обыграть большинство игроков.

IV. Alpha-Beta

Стоит отметить, что при увеличении глубины поиска алгоритма повышается его эффективность, но также возрастает и время. Используя переменную-счётчик, было получено количество итераций для разработанного движка с различными уровнями глубины поиска. Результаты представлены на таблице 1.

Таблица 1 – зависимость количества итераций от глубины дерева

Глубина дерева	Количество итераций	Время, с
1	10-20	0.1-0.2
2	500-700	0.3-0.9
3	14000-98000	1.1-5.2
4	500000-4000000	5.9-8.7
5	6000000 и более	9.6-14.6

Для шахмат среднее количество возможных перемещений из одной позиции примерно равно 40. Следовательно, для расчета дерева с глубиной, равной 4, необходимо перебрать 40 * 40 * 40 * 40 = 2560 тыс. позиций, а на глубину 5 – 10240 тыс. Дерево поиска, растёт экспоненциально, поэтому для большой глубины дерева простой перебор использовать невозможно. На сегодняшний день на самых мощных процессорах при самом оптимальном коде можно рассчитать глубину дерева равную 6 в реально оцениваемый промежуток времени.

Так как при простом переборе алгоритм оценивает все возможные ходы партии, в том числе и бессмысленные, то следует предусмотреть возможность отсечения заведомо невыигрышных цепочек. Эля этого хорошо подходит алгоритм Alpha-Beta, который является основой всех современных шахматных движков.

Для определения отсекаемых вариантов в рекурсивной функции требуется ввести две новые переменные максимум для белых (ALPHA) и максимум для черных (BETA). При первом вызове обе эти величины равны минимально возможному значению (-INFINITY). Если найденная позиция, лучше текущей, то это значение запоминается. Ветвь, оценка которой не улучшает уже найденное оптимальное значение, будет «отсечена». Использование алгоритма Alpha-Beta позволяет значительно увеличить глубину поиска, затрачивая прежний объём ресурсов.

V. Улучшенная функция оценки

Качество работы алгоритма зависит от способа оценки относительной силы фигур, поэтому в работе предложена оценка, учитывающая кроме типа фигуры ещё и позицию фигуры на шахматной доске. Например, конь, размещенный в центре шахматной доски, более ценен, так как количество доступных ходов для этой фигуры увеличивается. Ценность пешки возрастает с приближением к противоположной стороне доски (при достижении противоположной стороны пешка становится ферзём). Маска ценности пешки на шахматной доске представлена на рисунке 3.

```
const BOARD_MASK_WHITE_PAWN = [
    [7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0],
    [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]
];
```

Рис. 3 – Маска ценности пешки в зависимости от её позиции на шахматной доске

Аналогичным образом, основываясь на шахматной логике, составляются маски ценностей для остальных фигур.

VI. Выводы

Представлен алгоритм, на основе которого разработан шахматный движок, позволяющий рассчитывать оптимальные дебюты в шахматной игре.

VII. Список литературы

1. Программирование шахмат и других логических игр / Е. Н. Корнилов. – СПб.: БХВ-Петербург, 2005. – 272 с.: ил.
2. Simple chess AI [Электронный ресурс]. – Режим доступа: https://github.com/lhartikk/simple-chess-ai. – Дата доступа: 14.03.2021.
3. Chess program [Электронный ресурс]. – Режим доступа: https://github.com/da-nie/Chess. – Дата доступа: 14.03.2021.
4. Разработка шахматной программы [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/329528. – Дата доступа: 14.03.2021.