Разработка программного модуля искусственного интеллекта для игры в шахматы

А. В. Рягузов, email: ryaguzov2411@mail.ru
Д. В. Борисенков, email: xuser@relex.ru

Воронежский государственный университет

Аннотация. *На основе существующих алгоритмов для антагонистических игр с поочередными ходами была реализована программа для игры в шахматы, поддерживающая возможность игры против ИИ – компьютерного соперника. Намечено направление доработки программы с использованием нейросетей.*

Ключевые слова: *Теория игр, шахматный движок, альфа-бета отсечение, нейросети, Zobrist-hashing, нейросети.*

Введение

Игра в шахматы - традиционная область для исследования возможностей искусственного интеллекта. Данная статья посвящена реализации алгоритма выбора хода в шахматах, основанного на принципе минимакса и альфа-бета-отсечении. Рассмотрены способы оптимизации работы этого алгоритма по времени выполнения и по объему используемой памяти.

1. Общая характеристика игры в шахматы

Шахматы – антагонистическая игра с полной информацией. В ней нет элементов случайности, и отсутствует кооперация между игроками. Игра начинается со стандартной начальной позиции, два игрока делают ходы строго по очереди и не могут пропускать очередь хода. Игра может завершиться либо приходом к какой-то конечной позиции (мат, пат, недостаточно материала для мата у обеих сторон), либо к позиции, где ни одна из сторон не может добиться выигрыша при оптимальной игре другой стороны (вечный шах, позиционная ничья и т.п.) [1]. На практике также распространены сдача партии игроком, который не видит в своей позиции ресурсов борьбы за ничью или победу, и обоюдное соглашение игроков на ничью.

2. Дерево позиций

Для выбора наилучшего хода в заданной позиции необходимо построить дерево игры, представляющее собой ориентированный граф. Вершинами графа являются позиции, ребрами – ходы, ведущие из одной

позиции в другую. Граф может быть разделен на два непересекающихся подмножества вершин W (позиции, в которых очередь хода у белых) и B (позиции, в которых очередь хода у черных. Каждой вершине графа ставится в соответствие некоторое число, представляющее оценку соответствующей позиции (0 – ничейная позиция, положительное число – позиция выгодна для белых, отрицательное число – позиция выгодна для черных, модуль оценки – степень выгоды позиции для сильнейшей стороны) [2][3] Текущая позиция является корнем дерева. В процессе анализа программа должна определить оценку текущей позиции и всех позиций, содержащихся в дереве игры.

Сложность игрового дерева вычисляется по формуле

$$c = w^d$$

где w– среднее количество возможных ходов, d – глубина дерева.

3. Оценка позиции

Статическая оценка позиции – численное выражение степени выгодности позиции для одной из сторон, по факту является объективной интерпретацией человеческого взгляда на позицию и определения ее качества (насколько сильной она является) [3][4].

Разработка алгоритма, ответственного за расчет статической оценки, является самой сложной задачей с креативной точки зрения. В отличие от перебора ходов с помощью дерева, подходящего под огромное множество различных игр, функция оценки позиции должна быть специфицирована под конкретную игру (в данном случае – шахматы). Именно в функции оценки позиции учитываются особые факторы, свойственные данной игре [4].

Численное значение, которое ставится в соответствие позиции, определяет способность программы различать сильные и слабые позиции, что в целом определяет уровень искусственного интеллекта. Функция оценки, возвращающая хорошее значение для одного игрока, должна возвращать плохое значение для его оппонента, и наоборот. Таким образом, крайне важна симметричность оценочной функции.

Существует множество методов, используемых при реализации оценочной функции, например [3]:

– отдельное представление оценок каждой фигуры на каждой позиции, выраженное в виде таблиц;

– генетический алгоритм;

– корректировка оценки в зависимости от отдельных позиционных факторов.

4. Алгоритм минимакс

Принцип минимакса заключается в чередовании решений задачи выбора наилучшего хода между оппонентами, каждый из которых на каждом ходу выбирает оптимальную для себя оценку, возвращая решение в корень дерева с учетом того, что оппонент также будет выбирать лучшие ходы. В узлах последнего уровня (листьях дерева) осуществляется статическое оценивание.

При помощи алгоритма минимакса может быть осуществлен полный перебор всех позиции дерева заданной глубины (рис. 1) [3][5]. Однако сам по себе, без оптимизации, данный алгоритм имеет ряд недостатков, к которым относится экспоненциальный рост количества рассматриваемых позиций в зависимости от роста глубины дерева.

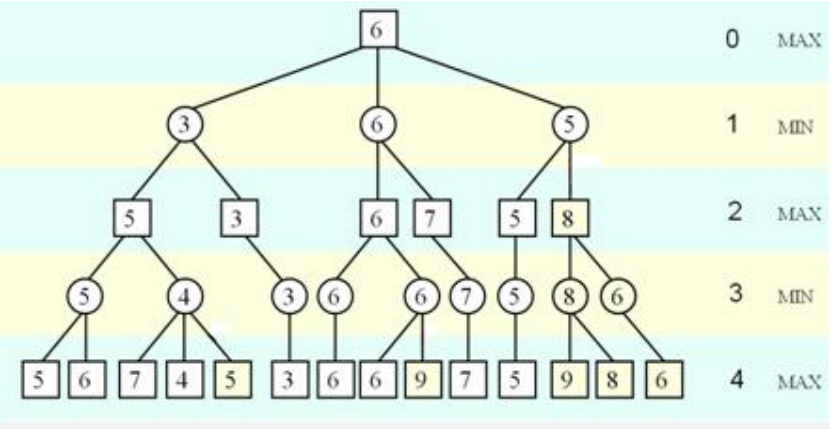

Рис. 1. Пример дерева игры, где вес каждого узла – оценка текущей позиции

5. Методы оптимизации алгоритма минимакс. Альфа-бета отсечение

Альфа-Бета отсечение (англ. Alpha-Beta pruning) – алгоритм поиска, основной задачей которого является сокращение количества узлов, оцениваемых в минимакс-дереве. Является важнейшим оптимизационным алгоритмом, применяемым практически во всех позиционных играх между двумя игроками [2][3][4].

Главная идея заключается в следующем: если при каком-либо ходе соперник имеет возможность сделать заведомо неблагоприятный для рассматриваемого игрока ход, то уже можно исключить из рассмотрения все остальные ответные ходы соперника.

Название данного алгоритма происходит от параметров alpha и beta, подающихся на вход для его выполнения. Они отвечают за границы отсечения на первом уровне, при проходе в глубину параметры меняются. Изначально $\alpha = +\infty$, $\beta = -\infty$. Принцип работы алгоритма изображен ниже (рис. 2).

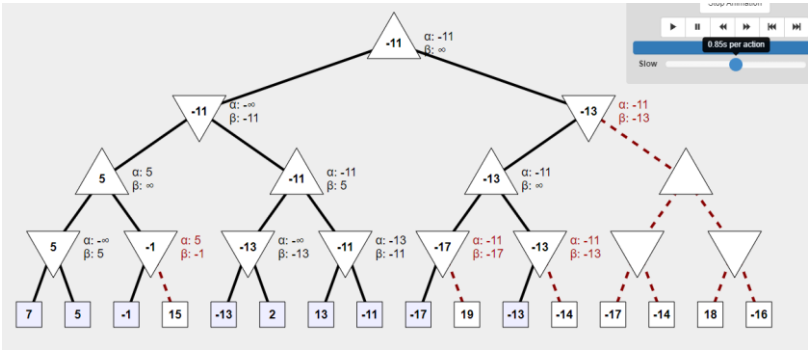

Рис. 2. Пример работы альфа-бета отсечения

6. Итерационное погружение

Основной принцип итерационного погружения (англ. Iterated Deepening) заключается в последовательном вызове функции перебора ходов на фиксированной глубине с увеличением глубины вплоть до достижения максимально заданной (либо пока не будет превышен лимит времени перебора). Основное преимущество данного метода – отсутствие необходимости выбора глубины поиска заранее с возможностью использования результата последнего поиска. Возвращенные значения могут быть также использованы для корректировки при следующем поиске [3].

7. Сортировка ходов

Результат альфа-бета отсечения достаточно сильно зависит от того, в каком порядке проверяются ходы. Для разработки алгоритма сортировки ходов необходимо определить, какие ходы нужно исследовать в первую очередь. Для этого существует так называемая эвристика убийства. Она заключается в том, что нужно попробовать первым ход, вызвавший отсечение на предыдущей ветке. Вводится массив, хранящий наилучшие ходы каждой глубины, и если существуют ходы из данного массива на текущей глубине, их нужно проверять в первую очередь [2].

Для всех остальных ходов приоритет должен отдаваться взятиям и шахам. Такие ходы гораздо важнее и заметнее «тихих» ходов, которые могут быть рассмотрены позже.

8. Zobrist-хеширование

Zobrist-хеширование (названо в честь Альберта Зобриста, р. 1942) – наиболее известный метод хеширования, получивший свое распространение в программировании настольных игр для двоих, в частности шахмат [2][4][6].

Zobrist-хеширование начинается со случайной генерации битовых строк для каждого возможного элемента шахмат, то есть для каждой комбинации фигур и позиции. В шахматах таковыми элементами являются 12 фигур на 64 поля или 16 фигур, если король может рокироваться и пешка может совершать взятие на проходе (En Passant). Далее любая конфигурация доски может быть разбита на независимые компоненты фигуры или позиции, которые соответствуют случайным битовым строкам, сгенерированным ранее. Окончательное хеш-значение вычисляется путем применения к битовым строкам операции побитового исключающего «или» (XOR).

9. Использование книги дебютов

Как уже было отмечено, дебют (начальная стадия игры) неразрывно связан с шахматной теорией. За века развития шахмат великими игроками, гроссмейстерами и просто шахматными теоретиками было открыто множество вариантов дебютов, например, каким образом максимально быстро развить свои фигуры, обезопасить короля, захватить инициативу и т.д [2].

Безусловно, возникает проблема обучения программы определенным дебютам, особенно с учетом того, что это относительно несложная задача, нежели обучение анализу более абстрактных и неоднозначных позиций, возникающих в миттельшпиле и эндшпиле. Для реализации данного принципа достаточно добавить в программу список дебютов, чтобы программа могла сравнивать позиции с уже имеющимися.

10. Разработка программы

Во время разработки шахматной программы должны учитываться следующие требования:

– Оптимальность реализованных методов по времени и качеству работы
– Простота реализованных методов

– Дружелюбный пользовательский интерфейс, предусматривающий игру на виртуальной шахматной доске с виртуальными шахматными фигурами, а также некоторые пользовательские настройки, например, выбор уровня сложности [7].

Программа разработана на языке C# с применением языка xml для создания элементов интерфейса.

Интерфейс (рис. 3, 4) и структура программы (рис. 5) приведены ниже. Автор дизайна: Jacques Fournier.

Рис. 3. Игровая доска

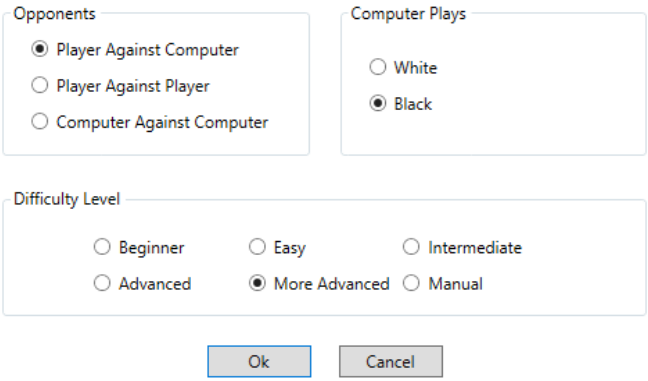

Рис. 4. Меню настроек

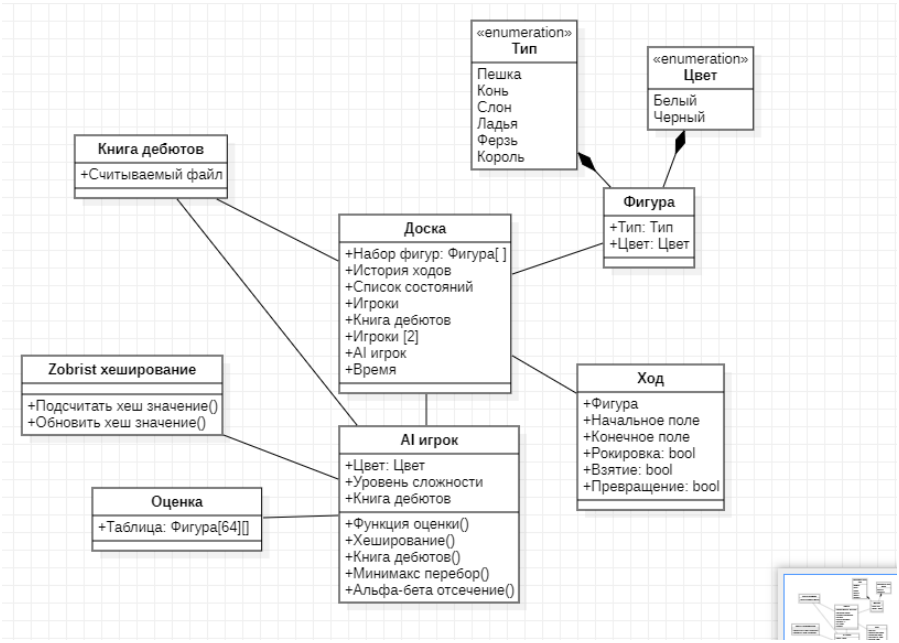

Рис. 5. Структура программы

Заключение

Были описаны основные принципы и алгоритмы, обеспечивающие работу шахматной искусственного интеллекта, в результате чего был

получен конечный результат в виде полноценной программы. В дальнейшем на основе уже изученных алгоритмов планируется усовершенствование шахматного ИИ с использованием нейронных сетей [8].

Список литературы

1. Компьютерные алгоритмы игры в шахматы [Электронный ресурс]. – Режим доступа: https://postnauka.ru/video/47550
2. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/329528/
3. Tproger [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/simple-chess-ai-step-by-step/
4. Chessprogramming WIKI [Электронный ресурс]. – Режим доступа: https://www.chessprogramming.org/
5. Harmonia Philosophia [Электронный ресурс]. – Режим доступа: https://harmoniaphilosophica.com/2019/02/13/huo-chess-c-micro- chess-updated/
6. Zobrist Hashing [Электронный ресурс]. – Режим доступа: https://levelup.gitconnected.com/zobrist-hashing-305c6c3c54d0
7. Tearth's blog. .NET and low-level programming [Электронный ресурс]. – Режим доступа:https://tearth.dev/posts/performance-of- chess-engines-written-in-csharp-part-1/
8. Dennis DeCoste. The Future of Chess-Playing Technologies and the Significance Kasparov Versus Deep Blue [Электронный ресурс]. – Режим доступа: https://www.aaai.org/Papers/Workshops/1997/WS- 97-04/WS97-04-003.pdf