Я. А. Евтушенко, Т. Н. Кульман

РЕШЕНИЕ ЗАДАЧ ПЕРЕБОРНОГО ТИПА НА ПРИМЕРЕ ШАХМАТ

Филиал «Протвино» университета «Дубна»
Кафедра информационных технологий

Представлена программа решения шахматных задач на поиск мата и разыгрывания партий. Рассматривается применение метода перебора с возвратом. Созданная программа была проверена на большом количестве реальных задач.

Особенно интригующая область
программирования – задачи так называемого
искусственного интеллекта.
Никлаус Вирт

Направление искусственного интеллекта связано с решением задач тех видов человеческой деятельности, которые традиционно считаются интеллектуальными. К таким задачам относятся и шахматные задачи, для решения которых необходимо изучение и применение многих математических и программистских методов.

Шахматными программами занимались и занимаются многие видные математики и программисты, такие как К. Шеннон, М. М. Ботвинник, Г. М Адельсон-Вельский [1, 2, 3] и др. В шахматных программах достаточно точно сформулированы цели и формализованы ситуации по сравнению с другими задачами, имеющими отношение к искусственному интеллекту. Методы, создаваемые в шахматном программировании, имеют более широкие области применения, некоторые из них имеют прикладное значение [2, 3]. Кроме того существуют объективные критерии оценки силы шахматных программ, выгодно отличающих их от других задач, моделирующих сложные процессы в системах управления.

Целью данной работы являлось создание программы, позволяющей решать шахматные задачи на поиск мата и разыгрывать партии.

Основными задачами в рамках поставленной цели являются:

1. Создание графического интерфейса доски.
2. Разработка компьютерного представления доски и правил игры (разработка соответствующих классов).
3. Создания алгоритма, позволяющего искать мат на заданную глубину**.**
4. Изменение данного алгоритма с целью добавления возможности игры с компьютером.

Для решения задач 3 и 4 применяется алгоритм перебора с возвратами. Как правило,

задача разделяется на отдельные подзадачи [4], решаемые с помощью рекурсии. Алгоритм поиска решения сводится к перебору вариантов и отбрасыванию ложных путей [5]. Рассмотрим реализацию схемы перебора с возвратом на примере поиска мата на заданную глубину:

1. Получение позиции.
2. Выбор фигуры ходящей стороны.
3. Перебор возможных ходов фигуры.
4. Проверка на наличие мата, если он есть, то выход из алгоритма.
5. Иначе, перебор ответов оппонента.
6. При получении информации о возможности мата, выполнение хода и выход.
7. Иначе, переход на п. 2.
8. Если перебор всех фигур закончен, — выход из алгоритма.

Программа написана на объектно-ориентированном языке программирования C# с использованием библиотеки *.NET* 3.5. Объём кода составляет около 25 тыс. строк.

Рассмотрим задачи, решаемые в работе.

1. Создание графического интерфейса доски

В интерфейсе пользователя шахматная доска представлена в виде формы с 64 кнопками. Для удобства работы с ними, каждой кнопке дается название по следующему правилу: *button.name* = “button”+x+y, где *x* и *y* — координаты клетки.

Все кнопки поддерживают коллекцию рисунков для отображения на них изображений клеток и фигур. Кнопки вызывают событие, которое передает координаты клетки, по которой был совершён щелчок, виртуальному представлению доски.

2. Разработка компьютерного представления доски и правил игры

Для реализации этих задач были созданы несколько классов. Рассмотрим некоторые из них.

Класс *Position.* Этот класс содержит массив структур, в которых отображается тип стоящей фигуры, её цвет и цвет клетки, на которой она стоит. Также этот класс содержит данные, позволяющие узнать, стоит ли белым или чёрным шах, ходил ли белый или черный король или ладья во время партии, что позволяет узнать, возможна ли длинная или короткая рокировки. Класс *Position* также содержит методы для работы с позицией.

Класс *MoveTest.* Перед тем, как совершить ход с помощью метода *move(),* необходимо проверить, возможен ли он. Класс *MoveTest* содержит группу методов, проверяющих корректность хода, если на клетке [*xs, ys*] стоит король (ферзь, слон, ладья, конь или пешка). Кроме этих методов класс *MoveTest* включает методы, проверяющие возможен ли ход; методы, проверяющие наличие шаха белым или чёрным и другие методы.

Класс *CheckMate.* Класс содержит методы для проверки наличия в позиции мата или пата.

3. Создания алгоритма, позволяющего искать мат на заданную глубину

Для поиска мата на заданную глубину создан класс *NiceMove.* Он включает:

- структуру, которая содержит координаты хода, оценку хода и фигуру на которую будет заменена текущая фигура после хода (применяется для превращения пешки);
- метод, определяющий лучший по оценке ход из данной коллекции ходов.
- метод, ищущий мат в 1 ход. Мат ищется путем перебора ходов с возвратом, пока не буден найден матующий ход, или пока возможные ходы не закончатся, т.е. мат в 1 ход отсутствует;
- рекурсивный метод, позволяющий искать мат на заданную глубину. Имеет параметр, который показывает текущую глубину перебора. Текущая глубина увеличивается при совершении хода стороной, за которую ищется мат, и уменьшается при возврате хода. При нахождении мата в 1 ход передается информация вызывающей функции о результатах поиска.
- методы, выполняющие перебор с возвратом. Для каждой фигуры реализован свой метод, что позволяет ускорить программу. Эти методы вызываются методами для поиска мата.

Задача поиска мата в заданное количество ходов включает также в себя реализацию ввода позиции для дальнейшего поиска решения.

4. Изменение алгоритма, с целью добавления возможности разыгрывания партий с компьютером

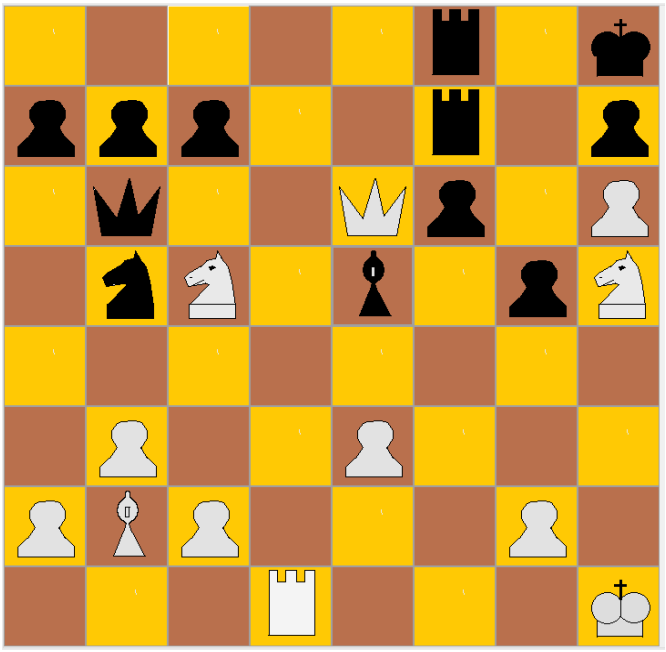
Для реализации этой задачи потребовалось в классе *NiceMove* создать новый метод оценки позиции, цель которого – вычислять и возвращать оценку текущей позиции в виде вещественного числа. Разработка этого метода дала возможность искать не только мат, но и оптимальный ход на данную глубину при отсутствии мата.

Эта оценка зависит от многих параметров, например: материальное соотношение фигур, наличие сдвоенных пешек, ситуация в центре доски (какая сторона его контролирует) и т. д.

Также в классе *NiceMove* был доработан метод, определяющий в текущей позиции не только матующий ход, но и ход с максимальной оценкой, если матующий ход отсутствует.

Работа программы была проверена на большом количестве реальных задач.

Для иллюстрации решения шахматных задач приведём позицию для поиска мата в 3 хода (рис. 1).



Мат в 3 хода.
Решение:
1.Q:f7 R:f7
2.Rd8+ Rf8
3.R:f8#

Рис. 1 Решение шахматной задачи для поиска мата в 3 хода

Библиографический список

1. Ботвинник, М. М. О кибернетической цели игры / М. М. Ботвинник. — М.: Советское радио, 1975. — 88 с.
2. Ботвинник, М. М. Шахматный метод решения переборных задач. / М. М. Ботвинник. — М.: Советский спорт, 1989. — 112 с.
3. Адельсон-Вельский, Г. М. и др.. Машина играет в шахматы. / Г. М. Адельсон-Вельский, В. Л. Арлазаров, А. Р. Битман, М. В. Донской — М. : Наука, 1983.—208 с.
4. Вирт Н. Алгоритмы и структуры данных / Н. А. Вирт — СПб. : Невский диалект, 2001. — 352 с.
5. Корнилов Е.Н. Программирование шахмат и других логических игр. / Е. Н. Корнилов — СПб.: БХВ-Петербург, 2005.—252 с.