Министерство науки и высшего образования РФ

ФГБОУ ВО «РГРТУ» имени В.Ф. Уткина

Кафедра «Космические технологии

ОТЧЕТ

к лабораторной работе № 1

по курсу "Экспертные системы и искусственный интеллект"

по теме

«Изучение алгоритмов Minimax, Negmax и Alpha-Beta отсечения»

Выполнил:

студент гр. 748

Чуйко А.А.

Проверил:

доцент кафедры КТ

Наумов Д.А.

Рязань, 2020 г.

**Оглавление**

[«Изучение алгоритмов Minimax, Negmax и Alpha-Beta отсечения» 1](#_Toc56289313)

[Цель работы 3](#_Toc56289314)

[Описание игры и её правил 4](#_Toc56289315)

[Описание состояние игры, ход игры в терминах структур данных выбранного языка реализации алгоритмов 4](#_Toc56289316)

[Вывод 5](#_Toc56289317)

[Приложение 6](#_Toc56289318)

Цель работы

Знакомство с задачей поиска пути на дереве игры, реализация и ислле-дование алгоритмов Minimax, Negmax и Alpha-Beta отсечения.

**Задачи**:

– выбрать детерменированную игру для двух игроков с открытой информацией (шашки, крестики-нолики, "точки" и т.д.);

– описать состояние игры, ход игры в терминах структур данных выбранного языка реализации алгоритмов;

– придумать и реализовать следующие алгоритмы:

1) инициализация начального состояния игры;

2) получение списка всех возможных ходов для заданного состояния;

3) выполнение хода;

4) отмена хода;

5) проверка, что игрок выиграл/проиграл;

– придумать и реализовать как минимум одну оценочную функцию;

– изучить учебный пример – реализацию игры в крестики-нолики;

– реализовать алгоритмы Minimax, Negmax и Alpha-Beta (или использовать их реализацию из учебного примера);

– сравнить количество вершин дерева игры для трех алгоритмов и время поиска хода в зависимости от заданной грубины дерева;

– реализовать игру компьютера с самим собой или с человеком (или использовать реализацию из учебного примера).

Описание игры и её правил

Ладейная игра

На шахматной доске, или, в более общей ситуации, на прямоугольной доске произвольного фиксированного размера стоит ладья (рис. 1) — шахматная фигура, которая может ходить по вертикали или по горизонтали, причем в данной игре только вниз или влево, на любое число клеток, но хотя бы на одну. Ходы игроков чередуются, и проигрывает игрок, который не может сделать ход. Пронумеруем клетки по горизонтали слева направо и по вертикали снизу вверх, начиная с 1.

Терминальная позиция — (1,1).

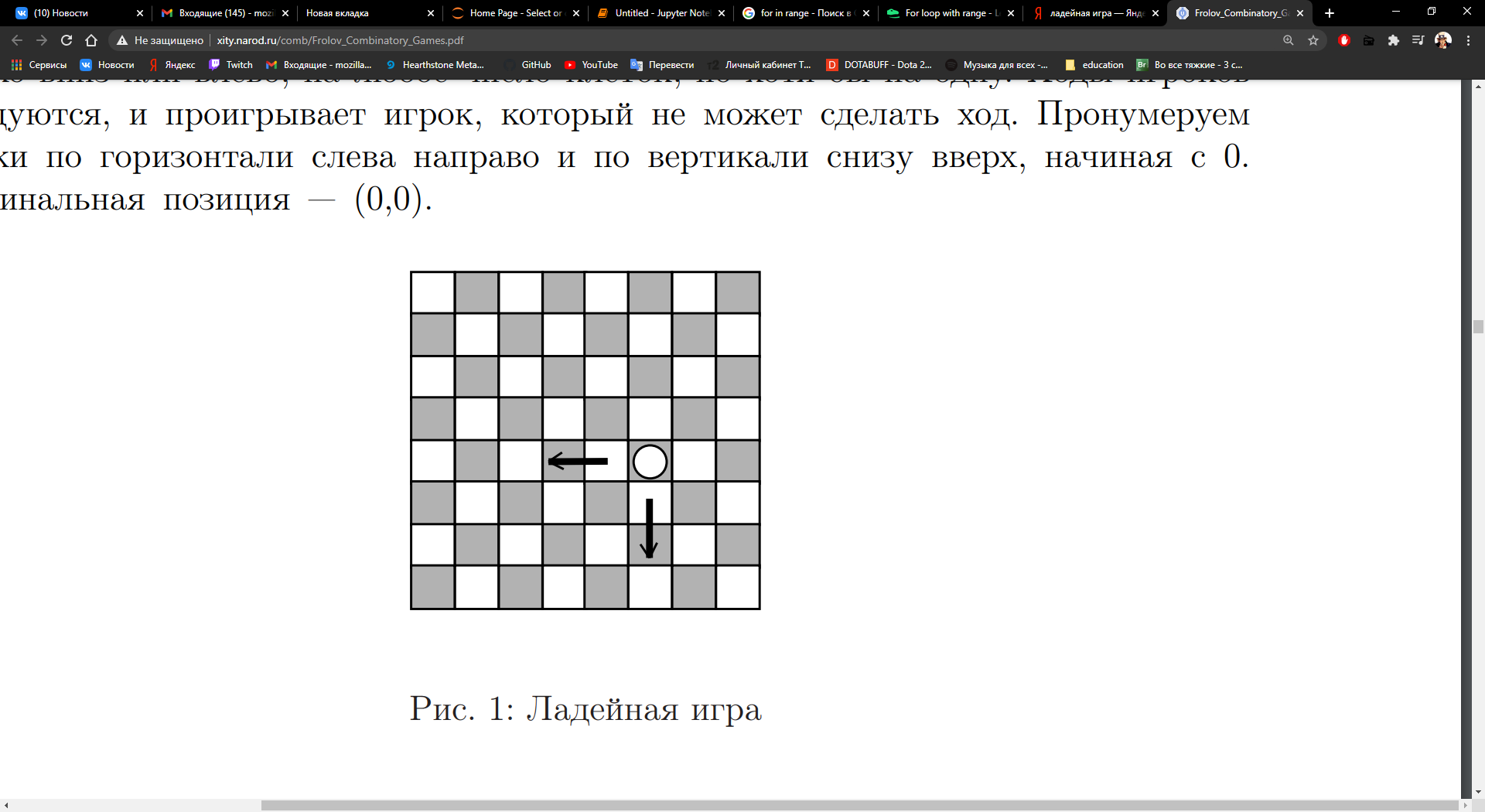


Рис. 1 – Ладейная игра

Описание состояние игры, ход игры в терминах структур данных выбранного языка реализации алгоритмов

При начале игры инициализируем начальное положение фигуры (координаты на поле 1, 1). Т.к. целью игры является достижение координаты (8, 8), то мы не будем инициализировать всё поле, а при совершении или отмене хода будем опираться на смещение по той или иной оси.

Ситуация в игре будет считаться выигрышной, если следующий ход будет иметь координату (8, 8), то есть обе оси должны быть равны 8, что и будет условием выигрышной ситуации.

Оценочная функция будет возвращать отрицательный модуль разницы между координатами, что будет соответствовать отклонению от диагонали, т.к. эмпирическим способом было выяснено, что нахождение на диагонали (x,x) является выигрышным.

Модуль будет отрицательным, т.к. цель игрока удержаться на диагонали, что принесет ему победу.

Вывод

В ходе работы была спроектирована игра для двух игроков, к которой был подключен искусственный интеллект с использованием оценочной функции.

Приложение

**Модуль игры**

from base import state

class state\_rook(state):

infinity = 100

players = ["1", "2"]

opponent = {"1": "2", "2": "1"}

def \_\_init\_\_(self):

self.x = 1

self.y = 1

def do\_move(self, move):

# для более простой отмены хода запишем в кортеж смещения по x и y и прибавим их к

# текущим значениям переменных

x, y = move

self.x += x

self.y += y

def undo\_move(self, move):

x, y = move

self.x -= x

self.y -= y

def is\_win(self, player):

if self.x == self.y == 8:

return True

return False

def get\_moves(self, player):

# если ситуация выигрышная или проигрышная

# то ходов нет

if self.is\_win(player) or self.is\_win(self.opponent[player]):

return []

moves = []

# всего имеем 8 - x и 8 - y ходов по каждой оси

for x in range(8-self.x):

moves.append((x + 1, 0))

for y in range(8-self.y):

moves.append((0, y + 1))

return moves

#Будем оценивать отклонение фигуры от диагональной позиции

#т.к. нахождение на позиции (x,x) является выигрышной

def score(self, player):

oppenent = state\_rook.opponent[player]

# если выиграл игрок, то +бесконечность

if self.is\_win(player):

return state\_rook.infinity

# если игрок проиграл, то -бесконечность

elif self.is\_win(oppenent):

return (-1) \* state\_rook.infinity

else:

return -abs(self.x - self.y)

**Модуль тестирования**

from alpha\_beta import bestmove

def calc\_nodes(state, level, player, opponent):

''' Расчет количества сгенерированных узлов

- state - начальное состояние

- level - максимальная глубина рекрсии (количество полуходов)

- player - игрок

- opponent - оппонент

'''

\_, \_, nodes = bestmove(state, level, player, opponent)

return nodes

def test\_count():

from rook import state\_rook

import numpy as np

import matplotlib.pyplot as plt

s = state\_rook('1')

player, opponent = "1", "2"

# <<<<<<< HEAD

lag = 1

x = np.arange(1, 8, lag)

y = np.array([calc\_nodes(s, level, player, opponent) for level in x])

\_ = plt.figure()

plt.plot(x, y)

plt.title('Count of nodes')

plt.ylabel('nodes')

plt.xlabel('alpha-beta')

plt.grid(True)

plt.show()

def test\_play():

from rook import state\_rook

s = state\_rook('1')

level = 4

player, opponent = "1", "2"

print('start!')

step = 1

while not (s.is\_win(player) or s.is\_win(opponent)):

move, \_, \_ = bestmove(s, level, player, opponent)

if move == None:

print('finish... draw')

break

print(f'Step {step}: {move} (x: {s.x} y: {s.y}) Player: {player}')

move, \_, \_ = bestmove(s, level, player, opponent)

s.do\_move(move)

print(f'Step {step}: {move} (x: {s.x} y: {s.y}) Player: {player}')

if s.is\_win(opponent):

print(f'Player {player} win!')

break

step += 1

player, opponent = opponent, player

print('the end')

if \_\_name\_\_ == "\_\_main\_\_":

test\_count()

test\_play()