#pragma once

#include "Player.h"

const int simulation\_times = 100;

class MonteCarloPlayer :

public Player

{

public:

MonteCarloPlayer() : Player() {};

MonteCarloPlayer(int i, string name = "MonteCarlo Player") : Player(i, name) {};

~MonteCarloPlayer() {};

Move getMove(const Board &board)

{

list<Move> rMoves = board.getRemainingMoves(); 25

priority\_queue<Move, vector<Move>, less<vector<Move>::value\_type>> bestMove;

for (Move : rMoves) 25x m\*n\*

{

Board tempBoard(board); 25 + 25

list<Neighbour>targets = tempBoard.getTargets(move); 25

if (targets.size() == 2)

{

move.captureTargets = targets; 4 log4 + 4 + 4log4

}

else if (targets.size() > 2)

{

move.pickTargets(targets, id); 4

}

tempBoard.addMove(move, id);

move.utility = simulation(tempBoard); 100X 100\*

bestMove.push(move);

}

cout << " X " << bestMove.top().x << " Y " << bestMove.top().y << " Utility: " << bestMove.top().utility << endl;

return bestMove.top();

}

int simulation(Board board)

{

int gamesWon = 0;

for (int i = 0; i < simulation\_times; i++)

{

Board tempBoard(board);

gamesWon += expansion(tempBoard, id);

}

cout << "Games Won " << gamesWon << endl;

return gamesWon;

}

int expansion(Board, int playerId)

{

Board tempBoard(board); 25 + 25

list<Move> rMoves = tempBoard.getRemainingMoves();25

if (rMoves.empty())

{

if ((playerId == 0 && tempBoard.getScore() > 0) || playerId == 1 && tempBoard.getScore() < 0)

{

return 1;

}

else

{

return 0;

}

}

list<Move>::const\_iterator moveIt = rMoves.begin();

advance(moveIt, rand() % rMoves.size());

Move = \*moveIt;

list<Neighbour>targets = tempBoard.getTargets(move);

if (targets.size() == 2)

{

move.captureTargets = targets; 4log4 + 4 + 4log4

}

else if (targets.size() > 2)

{

move.pickTargets(targets, playerId); 4

}

tempBoard.addMove(move, playerId); log4

return expansion(tempBoard, (playerId + 1) % 2); 150X m\*n\*6

}

};

m\*n\*100\*m\*n\*6

m\*n\*m\*n = k^2 Big 0 of n^2