Project: Adversarial Game Playing Agent

Student: Serhat Güngör

Project Description

In this project, you will experiment with adversarial search techniques by building an agent to play knights Isolation. Unlike the examples in lecture where the players control tokens that move like chess queens, this version of Isolation gives each agent control over a single token that moves in L-shaped movements--like a knight in chess.

Isolation

In the game Isolation, two players each control their own single token and alternate taking turns moving the token from one cell to another on a rectangular grid. Whenever a token occupies a cell, that cell becomes blocked for the remainder of the game. An open cell available for a token to move into is called a "liberty". The first player with no remaining liberties for their token loses the game, and their opponent is declared the winner.

In knights Isolation, tokens can move to any open cell that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board. On a blank board, this means that tokens have at most eight liberties surrounding their current location. Token movement is blocked at the edges of the board (the board does not wrap around the edges), however, tokens can "jump" blocked or occupied spaces (just like a knight in chess).

Finally, agents have a fixed time limit (150 milliseconds by default) to search for the best move and respond. The search will be automatically cut off after the time limit expires, and the active agent will forfeit the game if it has not chosen a move.

Report Requirements

Your report must include a table or chart with data from an experiment to evaluate the performance of your agent as described above. Use the data from your experiment to answer the relevant questions below. (You may choose one set of questions if your agent incorporates multiple techniques.)

Option 1: Develop a custom heuristic (must not be one of the heuristics from lectures, and cannot only be a combination of the number of liberties available to each agent)

- What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during the search?
- Analyze the search depth your agent achieves using your custom heuristic. Does search speed matter more or less than accuracy to the performance of your heuristic?

Option 2: Develop an opening book (must span at least depth 4 of the search tree)

- Describe your process for collecting statistics to build your opening book. How did you choose states to sample? And how did you perform rollouts to determine a winner?

- What opening moves does your book suggest are most effective on an empty board for player 1 and what is player 2's best reply?

Option 3: Build an agent using advanced search techniques (for example: killer heuristic, principle variation search (not in lecture), or monte carlo tree search (not in lecture))

- Choose a baseline search algorithm for comparison (for example, alpha-beta search with iterative deepening, etc.). How much performance difference does your agent show compared to the baseline?
- Why do you think the technique you chose was more (or less) effective than the baseline?

Report

Advanced Search Technique

MTD(f) algorithm is used for Advnced Search Tehnique. According to Wikipedia, “MTD(f) is an alpha-beta game tree search algorithm modified to use ‘zero-window’ initial search bounds, and memory (usually a transposition table) to reuse intermediate search results”".

Pseudo Code is

```
function MTDF(root, f, d) is
    g := f
    upperBound := +∞
    lowerBound := −∞

    while lowerBound < upperBound do
        β := max(g, lowerBound + 1)
        g := AlphaBetaWithMemory(root, β − 1, β, d)
        if g < β then
            upperBound := g
        else
            lowerBound := g

    return g
```

More information may be accessed from the below link.

MTD(f): https://en.wikipedia.org/wiki/MTD(f)

The baseline algorithm is **Alpha–beta pruning** algorithm. According to Wikipedia, “Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree”.

Pseudo Code is

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
```

```
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β,
FALSE))
            if value ≥ β then
                break (* β cutoff *)
            α := max(α, value)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β,
TRUE))
            if value ≤ α then
                break (* α cutoff *)
            β := min(β, value)
        return value
```

More information may be accessed from the below link.

Alpha–beta pruning: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

100 games were run with the both algorithms for different time limits of 50ms, 100ms, 150ms, 200ms and 250ms.

TIME_LIMIT	Baseline	MTD(f)
50ms	75%	28%
100ms	74%	21%
150ms	77%	27%
200ms	70%	25%
250ms	70%	28%

As it may seen in the table, baseline algoritm performs better thatn the MTD(f) in depth of 9. Baseline perfromds around %70 and MTD(f) perfroms atodun %25. MTD(f) has completed the run in faster duration but performance is lower.

When the depth is decreased the performance of the MTD(f) does not change significantly but baseline algorithm performance decrease. When the depth is 4 the performance go down to around %45.

Effectivenes of the baseline algorithm is changing with the depth of search. Since MTD(f) stores the previosly visited nodes, it runs faster.

Output of the run

```
root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 50
Running 100 games:
--++----------+---+----+---+--+-+-+-----+----------+-+------+-+-+----+--+-+-+----+++--+++-----++----+--
Your agent won 28.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 100
Running 100 games:
-------------+--+---+--+----------+--+---+--++-----++---++-++-----+---------+------+------+-+-+----+
Your agent won 21.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 150
Running 100 games:
-----+--+-+----+----+-----+--+---+--+-----+----+-------+-----+-----++-----+++++--++-+---+--+--+-+----++-
Your agent won 27.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 200
Running 100 games:
----+-----+----+-----+-+-+---+----------+--+--+----------++--++----++------++---+---+-----+++-+--++----
Your agent won 25.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 250
Running 100 games:
----------+---+-+---+-+-+--+-+---+----+--+-+-----+------+-++-+----+-++-+++-------+-++-+----+--------+
Your agent won 28.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# #With baseline algorithm
root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 50
Running 100 games:
+++-+++-+--++++-++++-+--+----+-++++++-++++-++-++++-++++++++++-+++++++-++++++-++++++-+++--+-++-++-++++
Your agent won 75.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 100
Running 100 games:
+++++-++---++++++++-+--+++++++++--++++++-+--++-++-++++++++++-+-++++++-+++-+--++-++++++++++++-+-++++---
+++++-
Your agent won 74.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 150
Running 100 games:
+++++++++--+++-+++++-++-+++-+-++++++-+++-+-++++-++-+++-++++++-+++++++++++++++--++-+-++++++++-+++--+-
+++-
Your agent won 77.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 200
Running 100 games:
+++++++--++++---++-+++--+-+++-+++++++-++++++-+---++++++++++-++-+-++--++++--++++++++++++++-+++-+----++++-
Your agent won 70.0% of matches against Minimax Agent

root@52fd39c1cebb:/home/workspace#
root@52fd39c1cebb:/home/workspace# python run_match.py -r 50 -t 250
Running 100 games:
---+++++++-++++++-+-+++++-+-++++--+-++-++++++--+++-+-+--+-++++----++++-++--++++++++++++++++-++-+++-+++
Your agent won 70.0% of matches against Minimax Agent
```