DETH Project

Students: Alexandre FLION, Arthur ADAM

Please note that we didn't make a Jupyter Notebook since our program is developed using C / C++ with shared libraries.

Context

This project has been done between November 9th 2022 and January 18 2023. We were asked to use a game to demonstrate our knowledge of decision theory.

Project

As a first task, we chosed a game to experiment our AI. We chose `connect-4` as our game.

Environment

Our environment is a 6x7 connect-4 board. It has 42 states. In order to test multiples AI, we've implemented some AI from a basic random AI to a really challenging AI.

Random AI

The first AI we implemented was the random AI. It's goal is really simple: it plays randomly each turn.

That AI is the worst one and is easily beattable.

In order to test it, you'll have to type some commands:

```
$> make /* It will make our Makefile compile our shared libraries containing
the different AI code. */
$> ./connect-4 ./ai_random.so /* It will launch a game against the random
AI. */
```

After you launch the game, a prompt will appear. Now, you just have to write the column you want to play in.

Kid AI

The kid AI is an implementation of a random AI but also a defensive AI. It's a little advancement of the random AI, beginning to check in the future for a possible victory or a

possible defeat.

In the case, you just have one last piece to place to win, the AI will defend and play to win or draw.

In order to test it, you'll have to type some commands:

```
$> make /* It will make our Makefile compile our shared libraries containing
the different AI code. */
$> ./connect-4 ./ai_kid.so /* It will launch a game against the kid AI. */
```

After you launch the game, a prompt will appear. Now, you just have to write the column you want to play in.

Pro AI

The pro AI is an implementation of a AI powered by pattern matching minmax and alpha-beta pruning. This AI is a big advancement from the kid AI.

Since this AI use minmax, she can see 10 moves into the future. The more she sees into the future, the less this move is considered. This behaviour is given by a falloff variable of 0.9 reduced at each branch depth she visits.

The minmax is powered by pattern matching. She checks pattern and gives a value to the pattern and the 6x7 board. While she's checking the different pattern, alpha-beta pruning removes already seen branches or pattern resulting in the same result.

That AI is also very defensive, it will maybe choose the defensive way over the win.

In order to test it, you'll have to type some commands:

```
$> make /* It will make our Makefile compile our shared libraries containing
the different AI code. */
$> ./connect-4 ./ai_pro.so /* It will launch a game against the kid AI. */
```

Please note that the program will print and show the values of the minmax during the game when the AI thinks about a move.

This gives you an AI that can win 96% of the time (on a 50 handmade games test run).

Discussions

On our Pro AI implementation, we could have planed to implement a branch caching of what we already seen to make the AI be more optimized and quicker than ever at each run.

The Pro AI can also be unbalanced between defense and attack as said in the previous paragraph. As we use pattern checking powered AI, our patterns scores can be sometimes over or underrated facing the current situation of the board. For example, a "OXXO.O" can be considered as a neutral score when the situation should make it a very good move or a very bad move.