COMP34120 Exam - 2011

Todd Davies

May 26, 2016

1. Not doing q1.
2. (a) Kalah is a two person, zero sum game of perfect information without chance. It is zero sum because if each move were to be assigned a score based on its benefit to the player playing the move, then the benefit of the move for the opposing player would be the negation of the score.

    It is a game of perfect information because each player can always see the whole state of the board, and every information set contains just one node in the game tree.

    Because of these properties, and according to Theorem 1.10 in the notes, which states that for any two person zero sum game of perfect information, without chance and where play ends after a finite number of moves, then either player one has a winning strategy, player two has a winning strategy, or both player one and two have strategies that force a draw.

    (b) The bot we wrote for the Kalah project used alpha beta pruning to determine the best moves to play. The program consisted of three main components:

    - The framework that interfaced with the game engine
    - A kalah-rule module that was able to identify legal moves that could be made from each position and build a game tree.
    - A heuristic module, consisting of a board analysis routine to assign as score to a board layout depending on its potential further down the game tree.
    - A minimax module that would take a game tree, and run the minimax algorithm on it. This used alpha-beta pruning and negamax to make the search as efficient as possible.

    The framework module would instansiate the starting board and then when it was our turn, start a minimax search with a specific depth (between 12 and 14 levels, depending on the time available and the speed of the computer running the program).

    The minimax module would run an alpha-beta negamax search on the game tree, expanding the game tree (using the kalah-rule module) as it went until it was the desired number of levels deep. When it had reached the search depth, it would have the heuristic module assign a score to the leaf node based on its likely outcome. An accurate heuristic was important here, since it would decrease how much the search would be affected by the horizon effect.

    (c) The greatest strength of our program was its efficiency; we implemened a very lightweight form of minimax; negamax, which relies on the zero sum property of kalah to run exactly the same code for both players in the game tree, but just negate the value of each leaf in every recursion. This was tricky when kalah allowed two consecutive moves by the same player in some situations, but it paid off in the end.

This simplicity made it easy to run the program with multiple threads. We implemented a thread pool, that we would submit tasks to (such as 'score the game tree from this node down for five levels'), which was flexible and allowed us to efficiently use all of the machine's resources.

Of course, alpha-beta pruning also made our AI more efficient by allowing us to eliminate branches of the tree that could be guarenteed to be 'bad'.

All of these measures allowed us to increase the depth to which we could search the game tree, and therefore mitigte the horizon effect.

(d) The greatest weakness of our program was the fact that we discarded the game tree at the end of every turn. If we could have preserved the game tree and used iterative deepening to increase the size of the existing tree, then we could have saved lots of computation.

If we implemented iterative deepening, not only could we preserve node scores between moves, but we could also improve the accuracy of alpha beta pruning since a more accurate score of each node could be ascertained since we wouldn't have to rely on a heuristic.

Also, we could have been more flexible when it came to timing of moves; if we were running out of time towards the end of the game, we could simply cut off the iterative deepening search early it would still be able to give an answer for the score of

3. (a) A two person Stackelberg game with perfect infromation is one where one player is the leader (i.e. they move first), and the other player is the follower (they react to the leader's move), and the leader knows the follower's reaction function. That is to say that the leader knows what response every 'move' he sets will evoke from the follower.

In a two person Stacelberg game with imperfect information, the leader does not know the follower's reaction function. In this case, the leader must try to guess or learn the follower's reaction function in order to determine what the best move is to make in order to maximise his payoff and minimse the payoff of the follower.

(b) First, lets find u_F in terms of u_L:

$$\begin{aligned}\frac{d}{du_F}J_f(u_L, u_F) &= 2u_L - 2u_F + 4\\ 0 &= 2u_L - 2u_F + 4\\ u_F &= u_L + 2\end{aligned}$$

Now we can sub this in to $J_L(u_L, u_F)$:

$$\begin{aligned}
J_L(u_L, u_F) &= (u_L - 1)(3 - 2u_L + u_F) \\
&= 3u_L - 2u_L^2 + u_L u_F - 3 + 2u_L - u_F \\
&= -2u_L^2 + 5u_L + u_L u_F - u_F - 3 \\
&= -2u_L^2 + 5u_L + u_L(u_L + 2) - u_L + 2 - 3 \\
&= -u_L^2 + 6u_L - 1
\end{aligned}$$

Now lets differentiate that to find our max profit:

$$\begin{aligned}
\frac{d}{du_L} J_L(u_L, u_F) &= -2u_L + 6 \\
0 &= -2u_L + 6 \\
u_L &= 3
\end{aligned}$$

Now we can find $u_F = u_L + 2 = 5$, and sub the prices into the payoff functions:

$$\begin{aligned}
J_L(u_L, u_F) &= (3 - 1)(3 - 6 + 5) \\
&= 4 \\
J_F(u_L, u_F) &= -3^2 + 2(3 \cdot 5) - 5^2 - 4(3) + 4(5) + 5 \\
&= -9 + 30 - 25 - 12 + 20 + 5 \\
&= 9
\end{aligned}$$

(c) i. The moving window approach involves an agent remembering the last n data points (analogous to moves in a Stackelberg game), and using those to derive the follower's reaction function. As more data points become available, the oldest ones are dropped from the data set and the new ones are added. Some learning algorithm such a regression is applied to the data points to derive the follower's reaction function. This is a form of online updating.

ii. Forgetting factor is when every data point is remembered in the history of the game, but they have progressively less influence on the learning process. This is achieved by multiplying the data points by a constant on each iteration (usually between 0.95 and 0.99).

iii. $\theta_{T+1} = \theta_T + L_{T+1}(y(T+1) - \phi^\tau(X(T+1))\theta_T)$
N.b. I don't understand the formula. One mark for quoting it though, right ;)

4. Not applicable