Artificial Intelligence (CS562)

Dodgem

Utkarsh Kunwar (B15338), Utkrisht Dhankar (B15138)

December 1, 2017

- To solve the problem of Game Playing, the approach of *Alpha-Beta* algorithm was used on the given board game, *Dodgem*.
- In Alpha-Beta, there is a *window* of sorts with the upper bound as α and β values as the bounds. The α is for the maximizing and β for the minimizing player. In the worst case, α is $-\infty$ and β is $+\infty$. As the game tree is explored, better values for α and β are achieved as they approach towards each other. The subtree is *pruned* when $\alpha \geq \beta$ and further nodes are not explored saving us time compared to the *Minimax* algorithm.
- In Dodgem, we used the same approach, as we had the list of valid moves from the *movegen* function and the details of the board like player turn, and the position of the pieces, that is, the state of the game.
- In our Alpha-Beta implementation, we had a function which tried the valid moves on a copy of the board to achieve the next state. This new state is now *evaluated* by our evaluation/heuristic function. The heuristic function returns a measure of the *goodness* of the state, and by extension, the move, for the maximizing player.
- The number of plies used was 4, that is, the algorithm looked ahead to a depth of 4 to obtain the best move. This depth was chosen simply because of the fact that it gave us a result in the prescribed move time on our machine.
- A combination of two heuristics were used based on the knowledge of the game. (We played it against each other and against ourselves several times to see which moves are favourable and which are not) The heuristics are -
  - The distance remaining from the finish line for the player's piece (the closer the better).
  - If the player's piece is blocking the opponent's piece or can block it in the next move to cost him more moves to reach the finish line and also if the player is getting blocked or not (currently or in the next move of the opponent).