move is a valid strategy for O and can be meet with a opposite corner move from X to try a mistake from O in the future exactly the same as in the third case above from the previous branch, and another move would be go to the middle where X eventually achieves strategy 1 or 2.

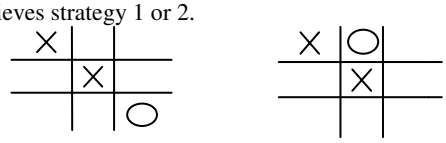

Figure 11: Tic-tac-toe Move 2 after center opening

The fist move will lead to win if O moves to the middle or a draw if it goes for the corners [Figure 11]. In the second case O has to block the lower left corner which leaves X to go for the middle left or corner left which are strategy 1 and 2.

To sum the strategies for the planning, first we have center or corner strategy for the beginning. Then for the center we try to get the corners with the particularly the one opposite to the one O holds. If the center is empty for the second strategy we go for it or we go for the opposite corner. After this point we either block the opponent or try to implement strategies 1, 2 or 3 which lead to victory.

Plan 1: Take center
Preconditions: Center empty
Plan 2: Take corner
Preconditions: All corners empty
Plan 3: Take corner after center
Preconditions: We have center take corner opposite to the one the opponent has
Plan 4: Take diagonal corner
Preconditions: We have a corner, the opponent has the ce-nter and the corner opposite to the one we have is free.
Plan 5: Block
Precondition: The opponent has tree tokens in a row, colu-mn or di agonal
Plan 6: Win
Preconditions: We have two tokens in a row, column or dia-gonal a nd the third place is free
Plan 7: Tie
Preconditions: If all places are taken, it's a tie.

5.1 Hierarchical Task Network

Top level task is Play [Figure 12]. This is a complex task and can be derived into: Win, Block, Tie or Search for Plan. The Search for plan is derived to both Plan 1 and Plan 2 or Plan 3 and Plan 4, which later leads to a call for the opponent's move and a recursive call to Play.

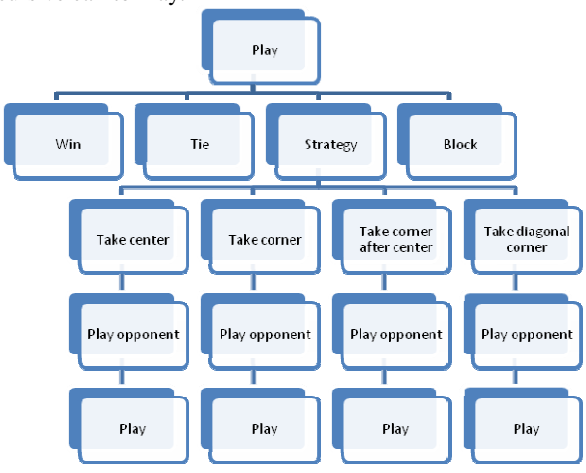

Figure 12: Tic-tac-toe HTN

This HTN when executed will result with plans for possible game scenarios. By creating nodes from each position and linking them with branches with the move of the player we create a game tree for the Tic-tac-toe game over which we can run the minimax algorithm.

This set up with 7 plans with 3 target strategies creates a tree for Tic-tac-toe which considers all possible moves for the second player with only 457 games, 281 of which X wins 176 are draw and 0 where the second opponent wins. This is a significant reduction over the 255, 168 possible games with a complete game tree. These reductions can be very useful for devices with limited computing capabilities but also we prove a very important point that planning can be very efficient if designing meaningful game trees by applying reasoning very similar to human player reasoning.

Further improvements to the game tree are also possible if the opponents moves are also planned, in other words if we drop all the meaningless and symmetrical moves of the opponent.

6. Game AI in Monopoly

6.1 Overview of the AI Implementation

The AI agent is responsible for the moves of the artificial players in the game. The core principle of the AI agent is building a Game Tree with all the sensible moves that all the players would make from the current point of time forward. Then using the minimax algorithm the agent selects the move that in the future would bring the computer player most favorable game position with the highest probability. Building a Game Tree in this game that would be big enough to consider sufficient number of moves is obstructed by the vastness of possible moves in combination with all the possible random landings of the dice. The number of nodes of the game tree exponentially grows at each level. To tackle this problem the AI agents incorporates two already discussed technologies: Case Based Reasoning and AI Planning.

The technologies are employed in the following manner. First the agent searches the CBR database to find the case with the largest similarity with the current state of the board. This case is associated with a playing strategy. The strategy consists of goal that the planner needs to build plans for, and the plans consist of consecutive player moves that bring the player to that goal. This way only moves that are part of that strategy are considered, those being a small fraction of the overall possible moves the number of edges of the game tree at each level decreases immensely.

At each level of the game tree the model considers the moves of a single player. After the strategies of the AI player are considered the response to those strategies needs to be considered by the opponent(s). The move of the opponent(s) depends of the probability distribution of the dice as well as the strategy of the player. A more general strategy needs to be implemented for the opponent's (human player) moves since we cannot be aware of the expertise of the opponent. This general strategy would bring more plausible moves than the focused strategy of the AI player.

After covering all opponents the agent comes back to deducting a feature move of the computer player by using the CBR selected plan strategy. After creating several loops of strategies and reaching a reasonable size of a Game Tree taking into account the memory limits and the rapidly decreasing probabilities that the move is possible due to the distribution of the dice the building of the Game Tree stops. Then the minimax algorithm searches the Game Tree and decides on the most favorable move for the AI player using the minimax algorithm. The process is repeated each time the AI player is up.