**Artificial Intelligence in Games**

**Session 2**

1. **A Survey of Monte Carlo Tree Search Methods**(Essential Reading notes):
   1. *Overview*:
      1. A tree is built in an incremental and asymmetric manner (the tree grows asymmetrically, exploring the most promising parts of the search space).
      2. For each iteration of the algorithm, a tree policy is used to find the most urgent node of the current tree. The tree policy attempts to balance considerations of exploration (look in areas that have not been well sampled yet) and exploitation (look in areas which appear to be promising).
      3. A simulation (a random or statistically biased sequence of actions applied to the given state until a terminal condition is reached) is then run from the selected node and the search tree updated according to the result. This involves the addition of a child node corresponding to the action taken from the selected node, and an update of the statistics of its ancestors. Moves are made during this simulation according to some default policy, which in the simplest case is to make uniform random moves.
   2. A great benefit of MCTS is that the values of intermediate states do not have to be evaluated, as for depth-limited minimax search, which greatly reduces the amount of domain knowledge required. Only the value of the terminal state at the end of each simulation is required.
   3. *Markov Decision Process (MDP)*: Models sequential decision problems in fully observable environments using four components:
      1. : A set of states
      2. : A set of actions
      3. : Probability of reaching the initial state if action is applied to state
      4. : A reward function
   4. *Q*-value: Expected reward of an action.
   5. *Bandit problems*: Bandit problems are a well-known class of sequential decision problems, in which one needs to choose amongst actions (e.g., the arms of a multi-armed bandit slot machine) to maximise the cumulative reward by consistently taking the optimal action.
2. **MCTS Algorithm**: involves iteratively building a search tree until some predefined computational budget – typically a time, memory, or iteration constraint – is reached, at which point the search is halted and the best performing root action returned. Each node in the search tree represents a state of the domain and directed links to child nodes represent actions leading to subsequent states.  
     
   Four steps are applied:
   1. *Selection*: Starting at the root node, a child selection policy is recursively applied to descend through the tree until the most urgent expandable node is reached. A node is expandable if it represents a nonterminal state and has unvisited (i.e., unexpanded) children.
   2. *Expansion*: One (or more) child nodes are added to expand the tree, according to the available actions.
   3. *Simulation*: A simulation is run from the new node(s) according to the default policy to produce an outcome.
   4. *Backpropagation*: the simulation result is “backed up” (i.e., backpropagated) through the selected nodes to update their statistics.
3. **MCTS Action choice**: Selecting actions involves a fundamental choice:
   1. *Exploitation*: Make the best decision based on current information.
   2. *Exploration*: Gather more information about the environment. This is, choosing actions that have been sampled less often.

The objective is to gather enough information to make the best overall decision. The best long-term strategy may involve short-term sub-optimal selections.  
  
The action selected must be a maximum of two values (Upper Confidence Bound , UCB): how good the action seems to be, and the uncertainty we have on the value of an action within a state.

1. **MCTS Enhancements**:
   1. *Transpositions*: Use a hash table, where the key is the identifier of the state, and the value is a pointer to the object that holds the state.   
        
      Every time a node needs to be updated or traversed; the object referenced in the transposition table is used instead of the copy in the tree. That way, statistics can be aggregated and centralized in a common place.
   2. *Progressive bias*: Adds domain specific knowledge to MCTS. This is especially useful when a node hasn’t been visited often enough and a game-dependant heuristic can help the selection step.
   3. *All Moves as First* (AMAF): Treats all moves played during selection and simulation steps as if they were played on a previous selection step.

The rationale of this is to reward those actions that are good to be taken independently of when are they taken. If the impact of these actions is not time-dependent and state dependent, the statistics gathered for each move can be based on more samples.