For our method of analyzing hockey events, we use a Markov Decision Process (MDP) and Value Iteration to calculate the value of each action in its context.

Since the notation gets a bit messy/overloaded:

State = Node = Vertex = Event with Event History

Edge = State Transition = Action/Event

Sequence = Play sequences, e.g. "PERIOD START, FACEOFF, HIT, SHOT, HIT, ... GOAL" or "FACEOFF, GIVEAWAY, SHOT, SHOT, ... , PENALTY"

**Overview of MDPs:**

The basic intuition of an MDP is that it consists of **states** and **actions** can be taken to **transition** to different states. An MDP can be displayed as a decorated directed graph. We create it using the events recorded in the NHL play-by-play sequences. It is made up of the following

**States S**

- These will be vertices in the graph. They will be one of the following

- The root node of the process (a vertex with no information)

- A context state node (a vertex labeled with only the following context information: GoalDifferential, ManpowerDifferential, and Period)

- This vertex denotes the initial information before any action has occurred in the play sequence

- Play sequences typically begin with a Faceoff and end with either a Goal, Penalty, or Stoppage

- An action-event node (a vertex with context information and the action-event history)

**Edges E = (s,s')**

- These will be directed edges in the graph going from one state s to another state s'

- Each edge denotes an action A taken

- e.g. Let state s be an action-event node labeled with "GoalDifferential = 0, ManpowerDifferential = 0, Period = 1, Name = "HOME:NEUTRAL:FACEOFF"" and state s' be an action-event node labeled with "GoalDifferential = 0, ManpowerDifferential = 0, Period = 1, Name = "HOME:NEUTRAL:FACEOFF, AWAY:NEUTRAL:HIT"". Then the edge e going from s to s' represents the action a = "AWAY:NEUTRAL:HIT"

**Rewards R(s)**

- These represent the value of being in a state s

- This will vary depending on the objective being analyzed

- Expected Goals will have R(s) = 1 when s is a HOME:GOAL, R(s) = -1 when s is an AWAY:GOAL, and R(s) = 0 otherwise

**Transition Probabilities Pa(s,s')**

- These represent the probability of transitioning from state s to state s' on action a

- For example, let state s be a context state node labeled "GoalDifferential = 0, ManpowerDifferential = 0, Period = 1", s' be an action-event node labeled "GoalDifferential = 0, ManpowerDifferential = 0, Period = 1, Name = "HOME:NEUTRAL:FACEOFF"" and s'' be an action-event node labeled "GoalDifferential = 0, ManpowerDifferential = 0, Period = 1, Name = "AWAY:NEUTRAL:FACEOFF"". Suppose the home team typically wins the faceoff 60% of the time in this situation and the away team wins the faceoff the other 40% of the time. Then the transition Pa(s,s') is 0.60 when a = "HOME:NEUTRAL:FACEOFF" and Pa(s,s'') is 0.40 when a = "AWAY:NEUTRAL:FACEOFF"

- These transition probabilities are derived during the MDP construction

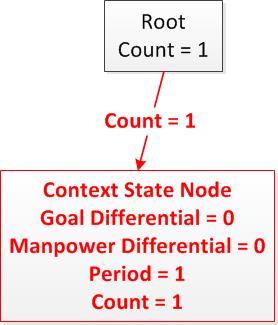
**MDP Construction**

For constructing the MDP, we iterate over each play-by-play event in each game.

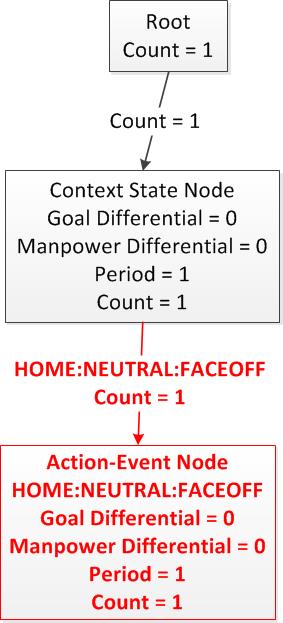


We start from the root node and increase the count (i.e. the number of times we've reached the root node)

Then we look at the first event and get the context information.

If the root node doesn't have this context information as a connecting context state node, we create a new context state node and add an edge from the root to this node.

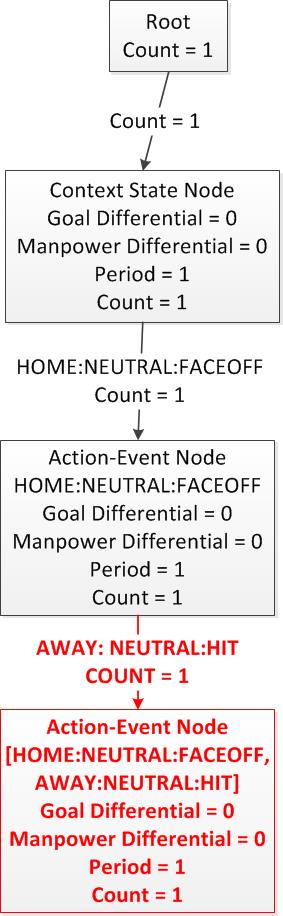
Then we navigate to the context state node, increase the count on the edge from the root to the context state node as well as the count for the context state node.



We then look at the first event in the play-by-play sequence (Typically a FACEOFF)

If the context state node doesn't have this event as a connecting node, we create a new action-event node for this event and add an edge from the context state node to this action-event node.

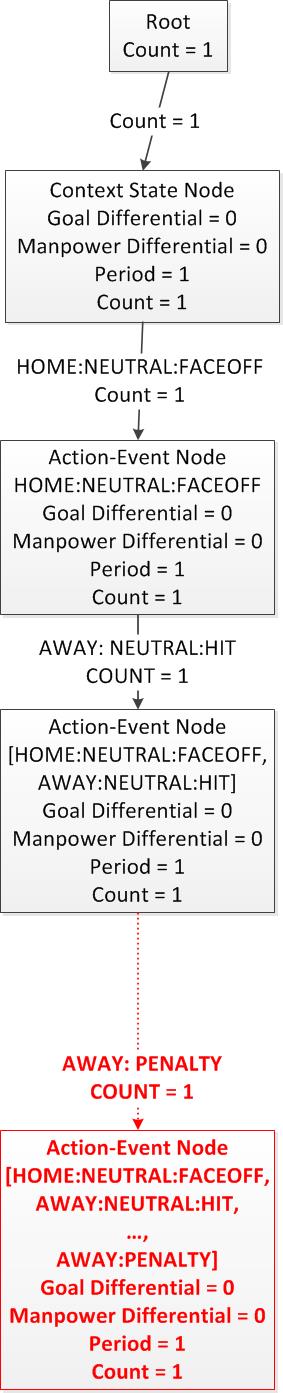
Then we navigate to the action-event node, increase the count on the edge from the context state node to the action-event node, as well as the count for the action-event node.

We then look at the next event.

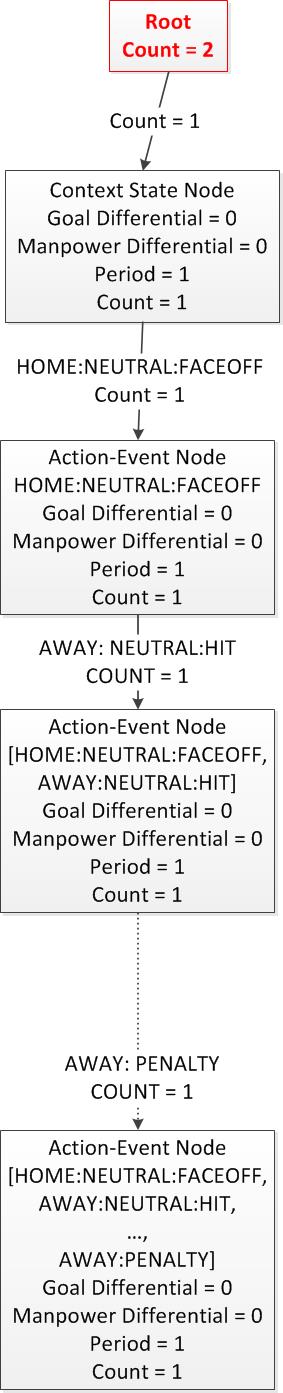
If the previous action-event node doesn't have the next event as a connecting node, we create a new action-event node for this event and add an edge from the previous action-event node to the next action-event node.

Then we navigate to the next action-event node, increase the count on the edge from the previous action-event node to the next action-event node, as well as the count for the next action-event node.

If the action-event node was navigated to from an action in the objective being analyzed (e.g. goals), the reward for the node is set appropriately (i.e. HOME:GOAL has R(s) = 1 in an expected goals model)

This process of connecting action-event nodes is repeated until a terminal state (i.e. an event that stops the play such as GOAL, PENALTY, or STOPPAGE) is reached.

After creating a node for this terminal state and increasing the edge and node counts, we navigate back to the root, but note this terminal state as the previous node.

The process then continues with the following events as we did before.

The difference is this time we add an edge and increase the edge count from the terminal node of the previous sequence to the context node of the current sequence.

After all play sequences in a match have been iterated over, we connect a node for "HOME:WIN" or "AWAY:WIN", depending on which team won the match, to the last node in the match.

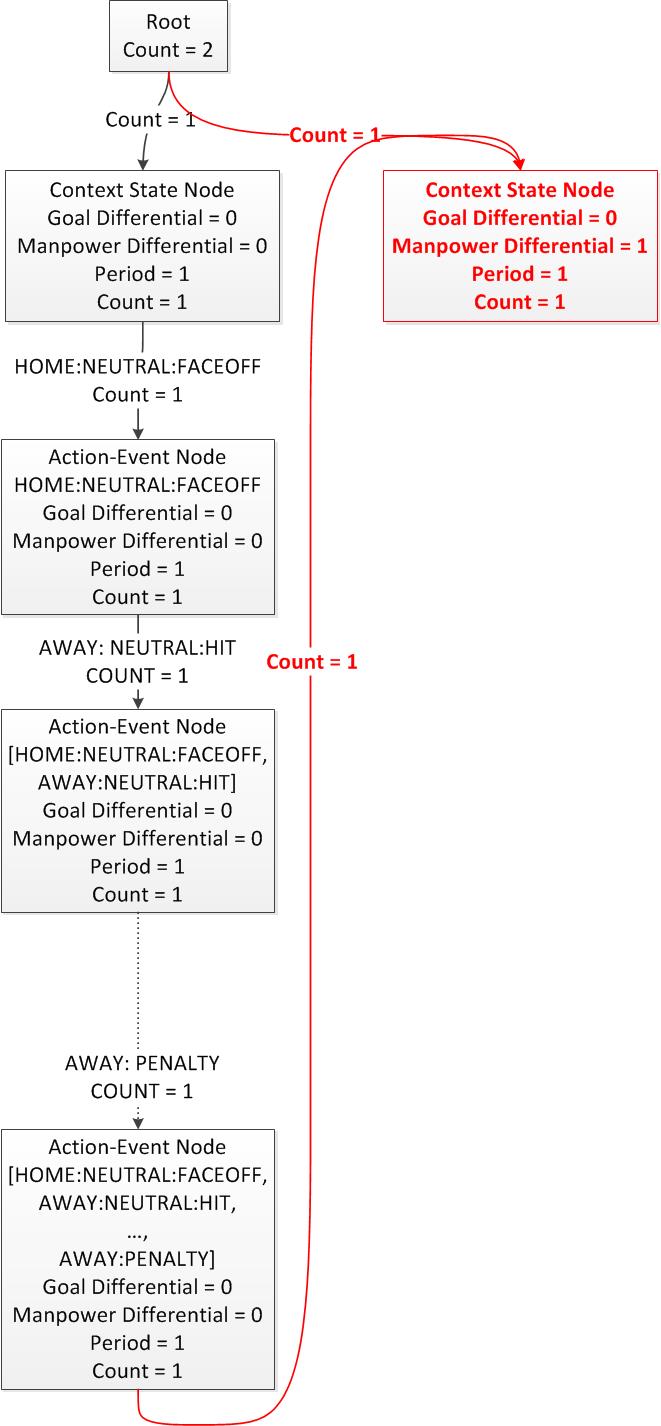
We also increase the edge count to the win node and win node count.

This is to facilitate backing up the probability of winning.

We can then repeat the process from the beginning for the next match.

Nodes, edges, and counts are added and adjusted on the same graph.

This process constructs the MDP.



**Value Iteration:**

Value iteration is a form of reinforcement learning that uses backup computation in the graph to learn values for each state.

In each iteration, the values of child vertices are propagated back up to the parent vertex. Over each iteration, these values will propagate across the graph.

Different calculations are used depending on the objective function being used:

Expected Win, Expected Goals, Expected Penalties:

Probability Next Home Goal, Probability Next Away Goal, Probability Next Home Penalty, Probability Next Away Penalty:

Probability Home Team Wins, Probability Away Team Wins: