Chess Match Example

- Setting: A player is about to play a two-game chess match and the goal is maximize the probability of winning the entire match.

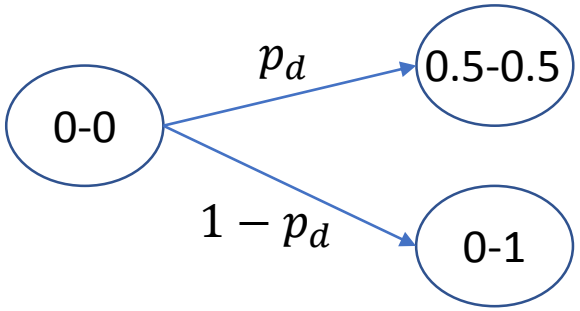
- Winning a game is worth 1 point; Drawing is worth 0.5 points; Losing is worth 0.

- If the final score after two games is 1-1, the we enter sudden-death: the players keep playing until someone wins (a decisive result).

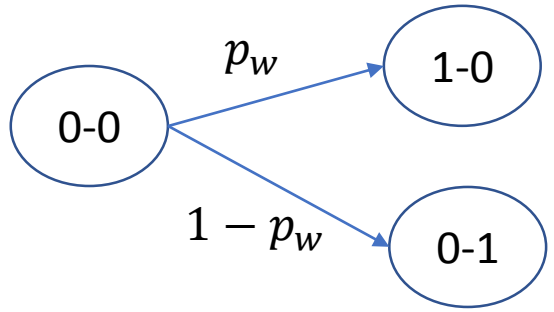
- We are contemplating two possible strategies:
  - Timid Play: By playing timid, the player will draw with probability p_d and will lose with probability $(1 - p_d)$.
  - Bold Play: By playing bold, the player will win with probability p_w and will lose with probability $(1 - p_w)$.

- Assumption: Suppose $p_d > p_w$.

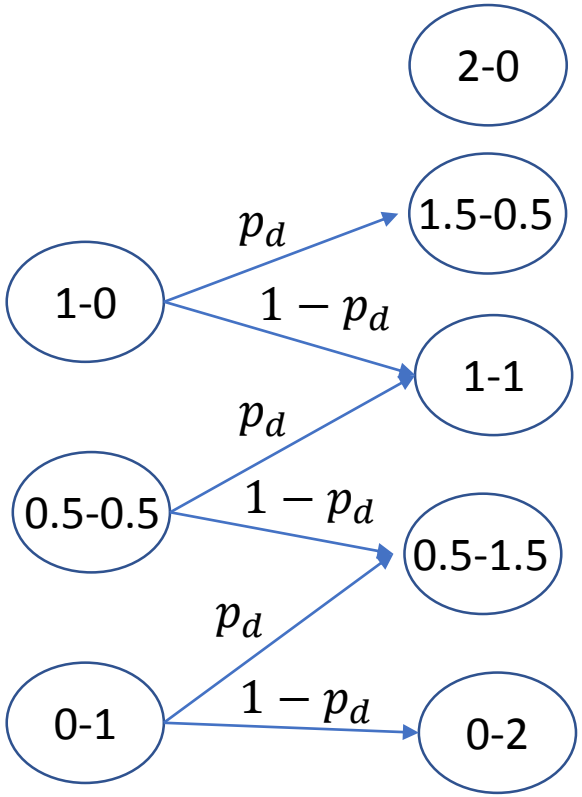
- We will use the DP Algorithm in order to find the optimal closed-loop policy in order to provide the player with the best chance of winning the match.



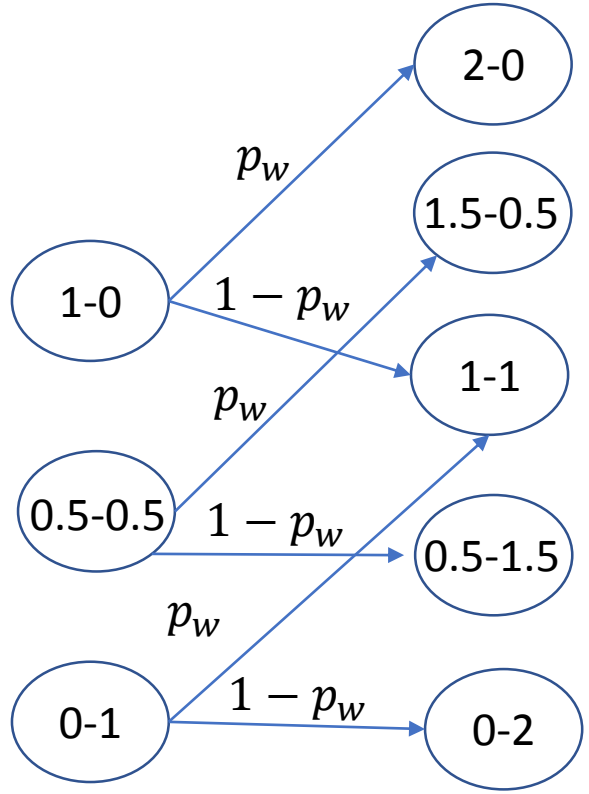
1st game: Timid play



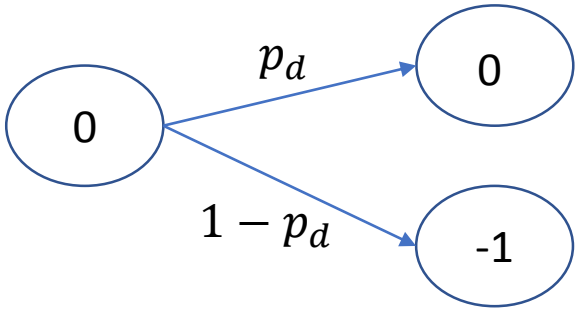
1st game: Bold play



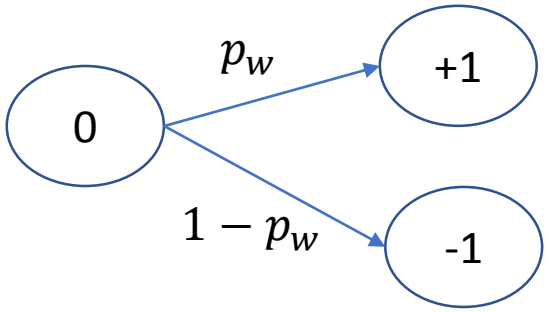
2nd game: Timid play



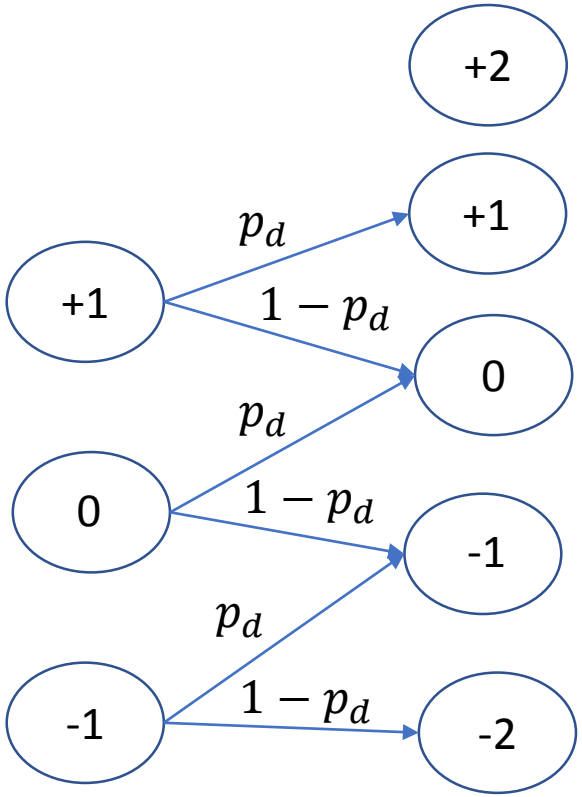
2nd game: Bold play



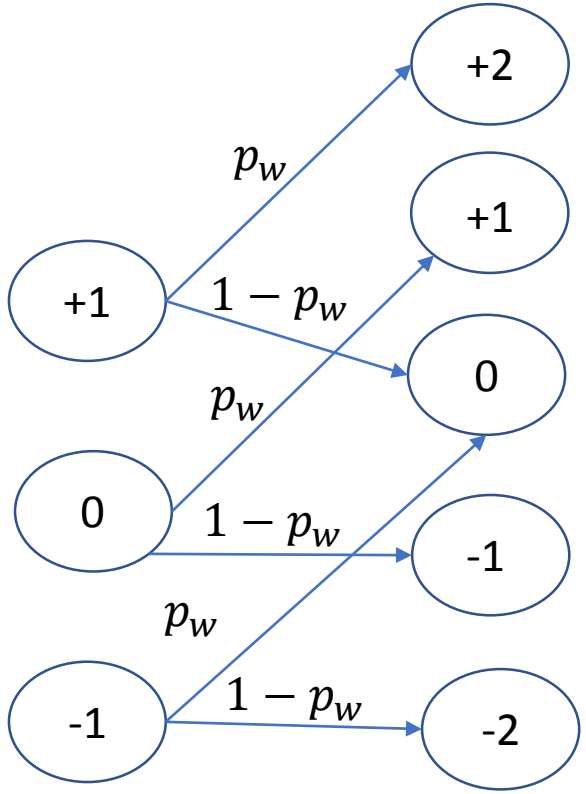
1st game: Timid play



1st game: Bold play



2nd game: Timid play



2nd game: Bold play

DP Algorithm

- Recall the DP-algorithm:

$$J_N(x_N) = g_N(x_N)$$

$$J_i(x_i) = \min_{u_i \in U_i(x_i)} \left\{ \mathbb{E}_{w_i} \left[g_i(x_i, u_i, w_i) + J_{i+1}(f_i(x_i, u_i, w_i)) \right] \right\}, \forall i \in \{0, ..., N-1\}$$

- We need to define all those elements in order to solve our problem.
- States: x_i net score at the beginning of each game.
- Controls/Actions: $\mu_i(x_i) \in \{\text{bold}, \text{timid}\}$.
- Disturbance: $w_i \sim \Pr(\cdot \,|x_i, \mu_i(x_i))$: "outcome of the game given our control decision $\mu_i(x_i)$.
- Hence we have the following DP equation:

$$J_i(x_i) = \max \left[p_d J_{i+1}(x_i) + (1-p_d) J_{i+1}(x_i - 1), p_w J_{i+1}(x_i + 1) + (1-p_w) J_{i+1}(x_i - 1) \right]$$

DP Algorithm

- So our policy simplifies to a ratio test and is given as follows:

$$\mu_i(x_i) = \begin{cases} \text{“play bold” : if } \frac{p_w}{p_d} > \frac{J_{i+1}(x_i) - J_{i+1}(x_i - 1)}{J_{i+1}(x_i + 1) - J_{i+1}(x_i - 1)} \\ \text{“play timid” : otherwise} \end{cases}$$

- Now we apply the DP recursion **<u>backwards</u>** starting from $N = 2$:

$$J_2(x_2) = \begin{cases} 1, \text{if } x_2 > 0 \\ p_w, \text{if } x_2 = 0 \\ 0, \text{if } x_2 < 0 \end{cases}$$

- Note that if after the second game, if the score is a tie (i.e. equal to 0) then the only move is to “play bold”, and the game ends in the first sudden-death game.

DP Algorithm

- Proceeding backwards one more time gives us:

$$J_1(x_1) = \begin{cases} x_1 = 1\text{: } J_1(1) = \max\left[p_d + (1-p_d)p_w, p_w + (1-p_w)p_w\right] = p_d + (1-p_d)p_w, \text{ optimal decision: timid play} \\ x_1 = 0\text{: } J_1(0) = p_w, \text{ optimal decision: bold play} \\ x_1 = -1\text{: } J_1(-1) = p_w^2, \text{ optimal decision: bold play} \end{cases}$$

- Proceeding backwards for the last time for the initial state $x_0 = 0$:

$$J_0(0) = \max\left\{p_d p_w + (1-p_d)p_w^2, p_w(p_d + (1-p_d)p_w) + (1-p_w)p_w^2\right\} =$$

$$p_w(p_w + (p_w + p_d)(1-p_w)); \text{ optimal decision: bold play}$$

- **Optimal policy:** At the start of the match play bold. Then play timid if and only if you are ahead!

Overview of Dynamic Programming