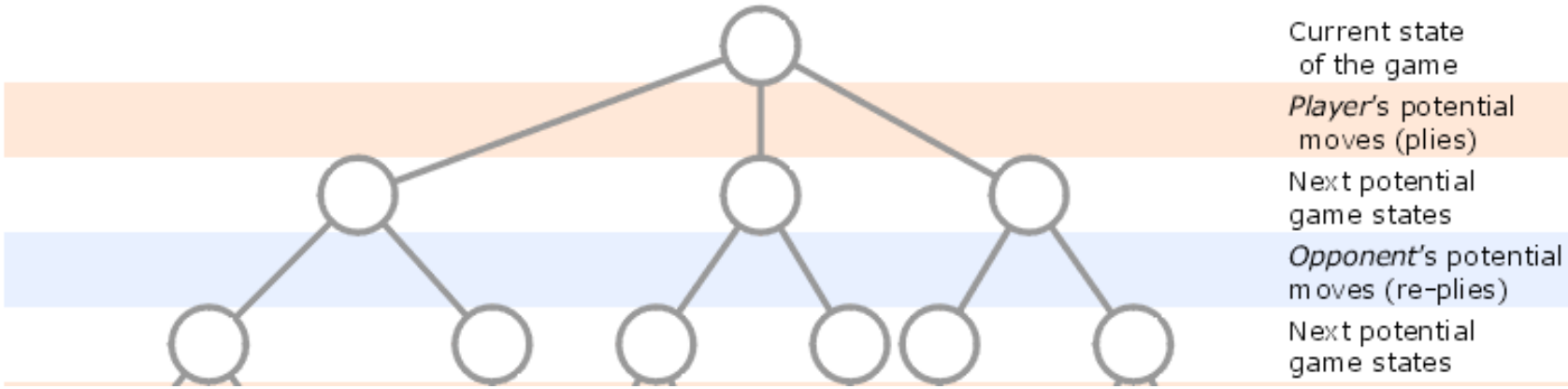
ETH zürich

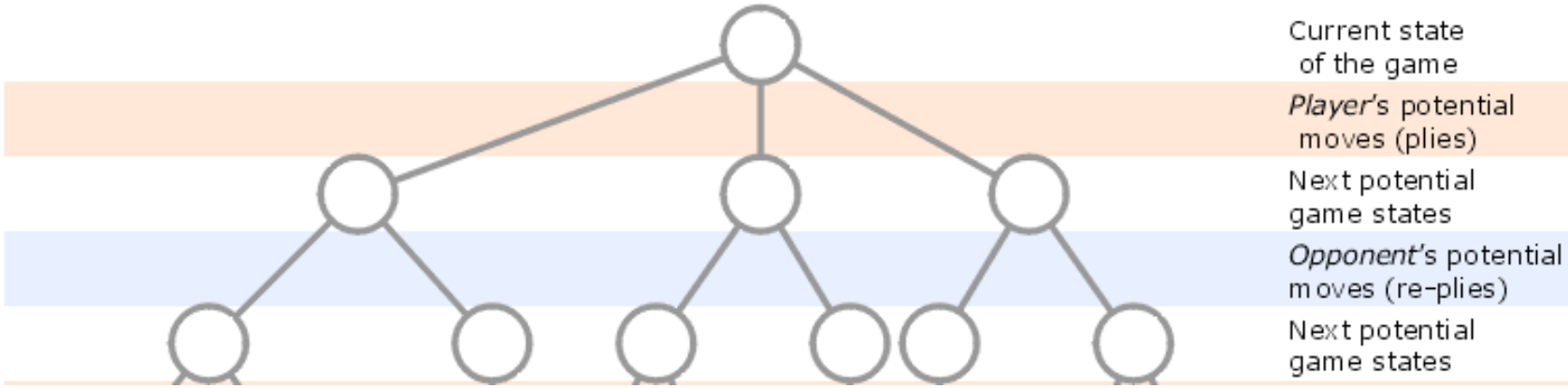
# Design of a Parallel Chess Engine

Jonathan Maurer, Jonathan Rosenthal, Jonas Kuratli

# Simple Approach



# Simple Approach




- Evaluate states up to a pre-defined depth d bottom-up
- Leaf evaluated using an evaluation function
- Other nodes evaluated by finding min or max of children

# Simple Approach

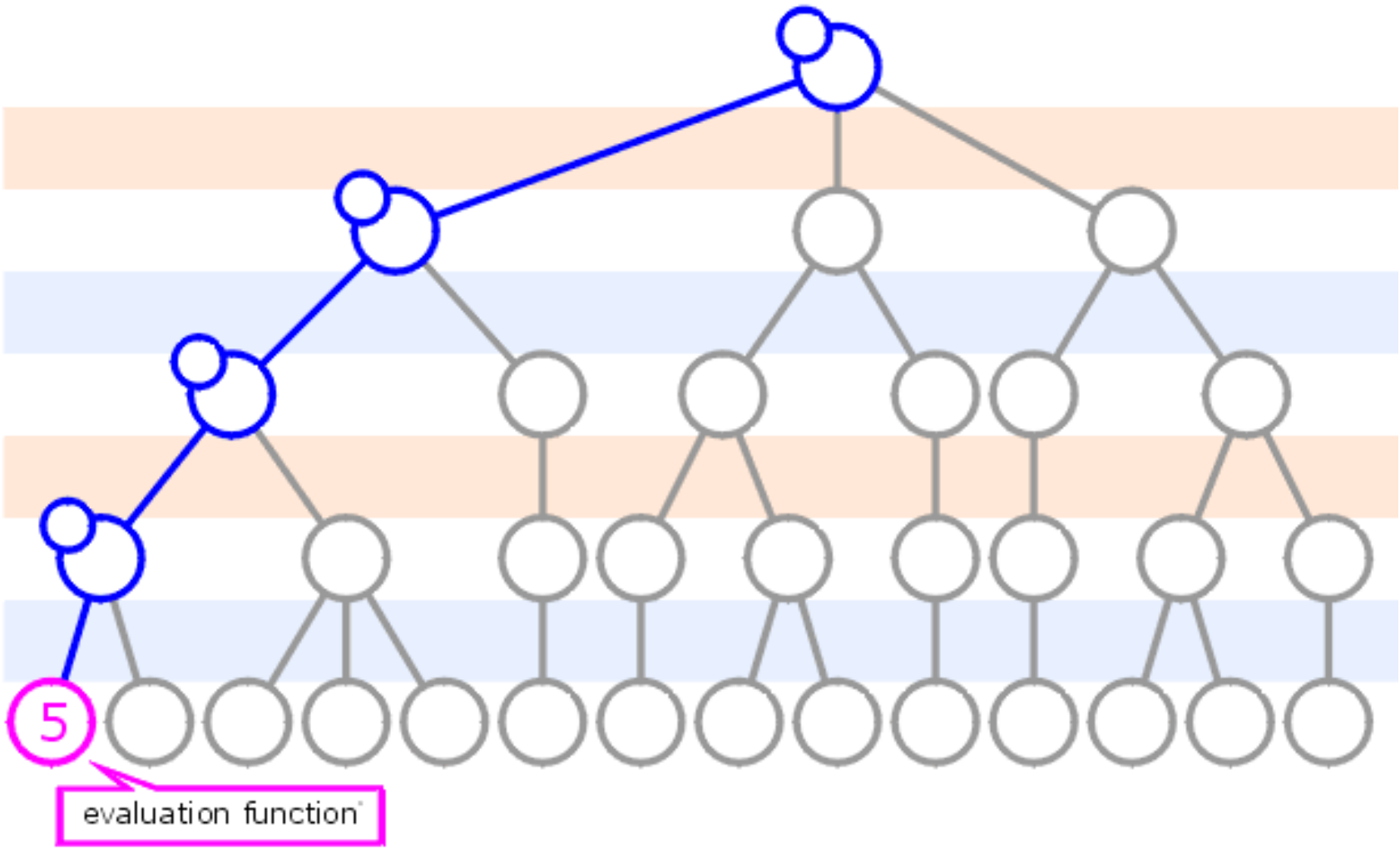
Pros:

- Easy to implement
- Easy to parallelize (Distribute evaluation of children)

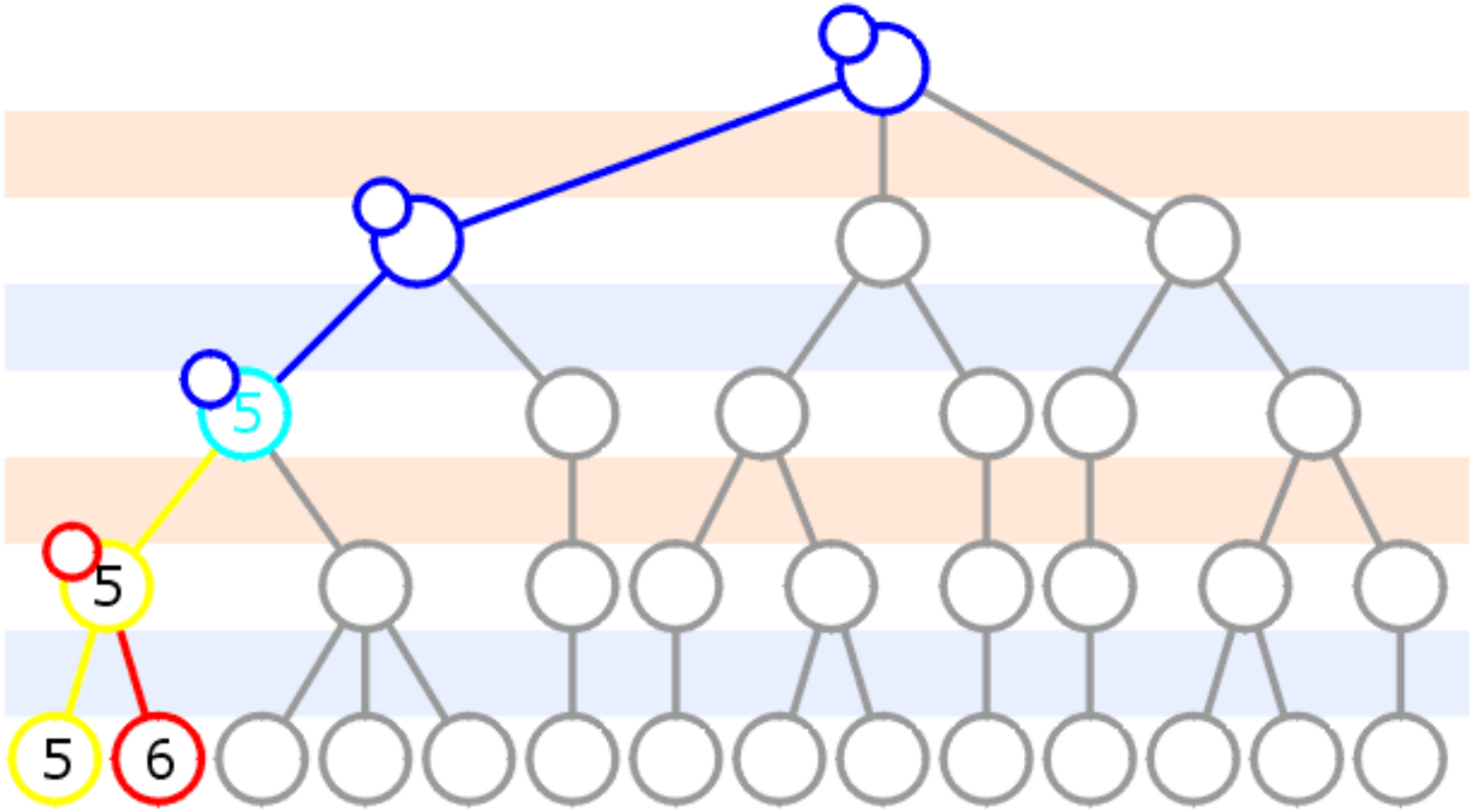
Cons:

- Tree becomes broad very quickly
- Message passing overhead if parallelized

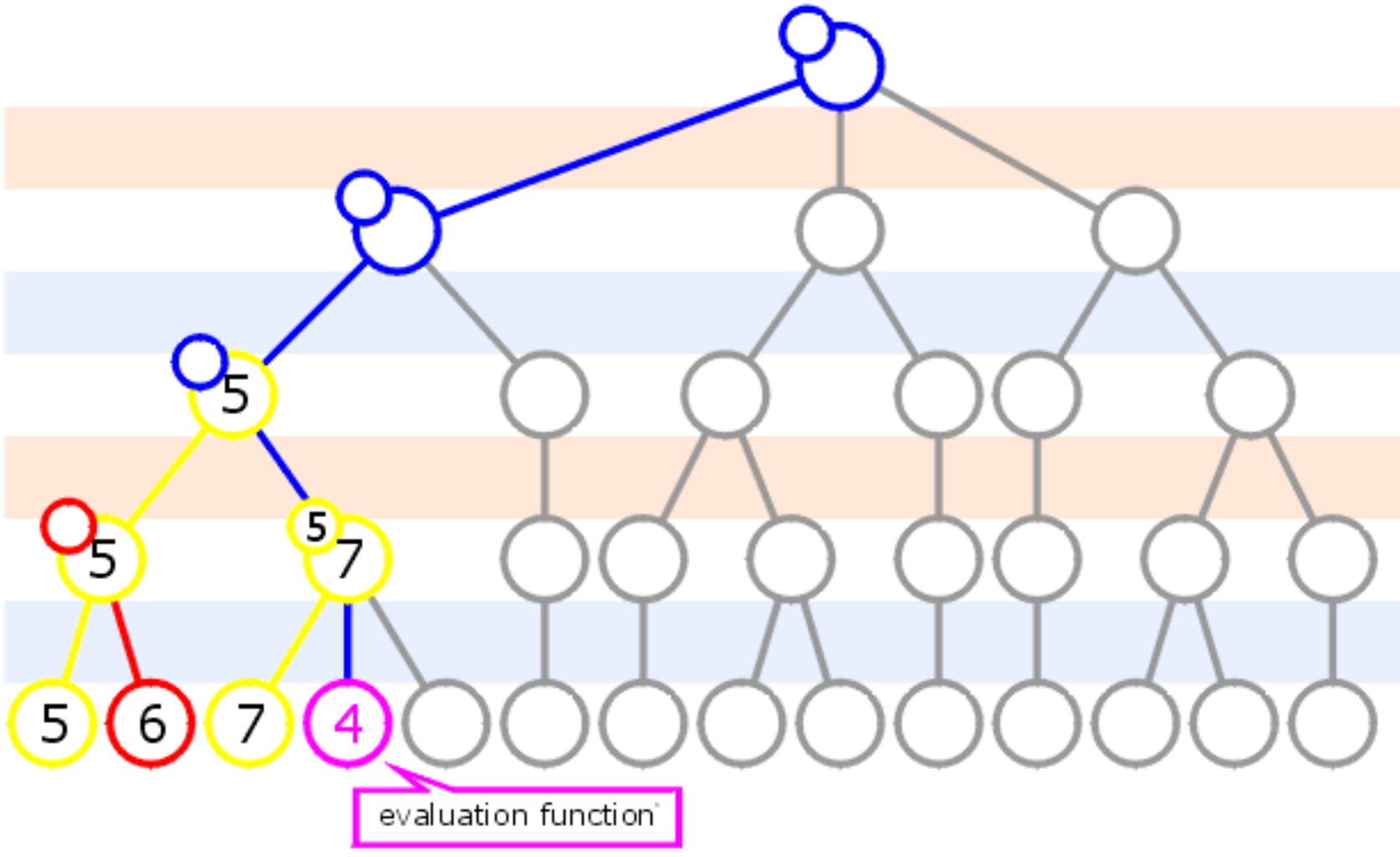
# Alpha-Beta Pruning



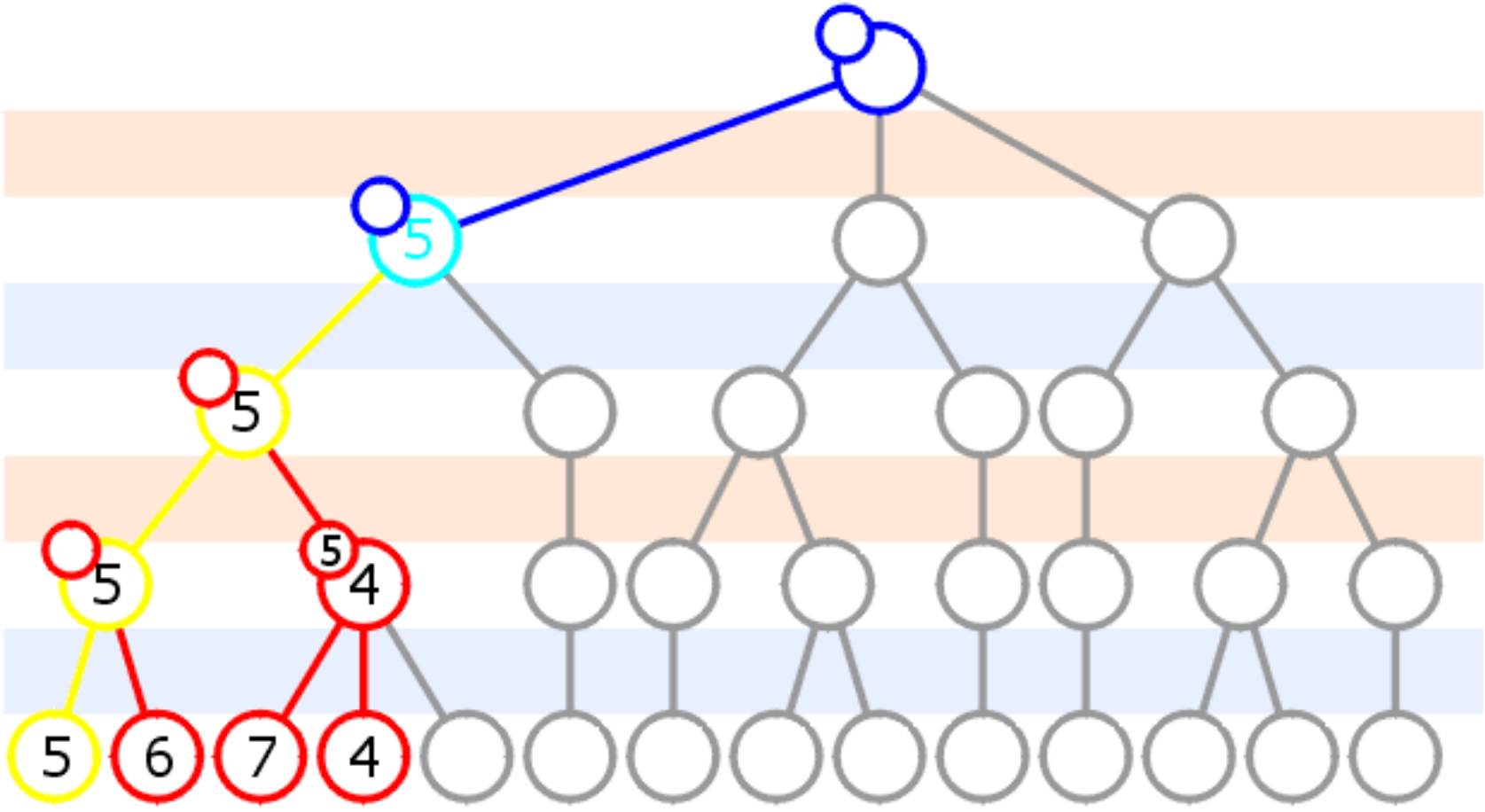
# Alpha-Beta Pruning



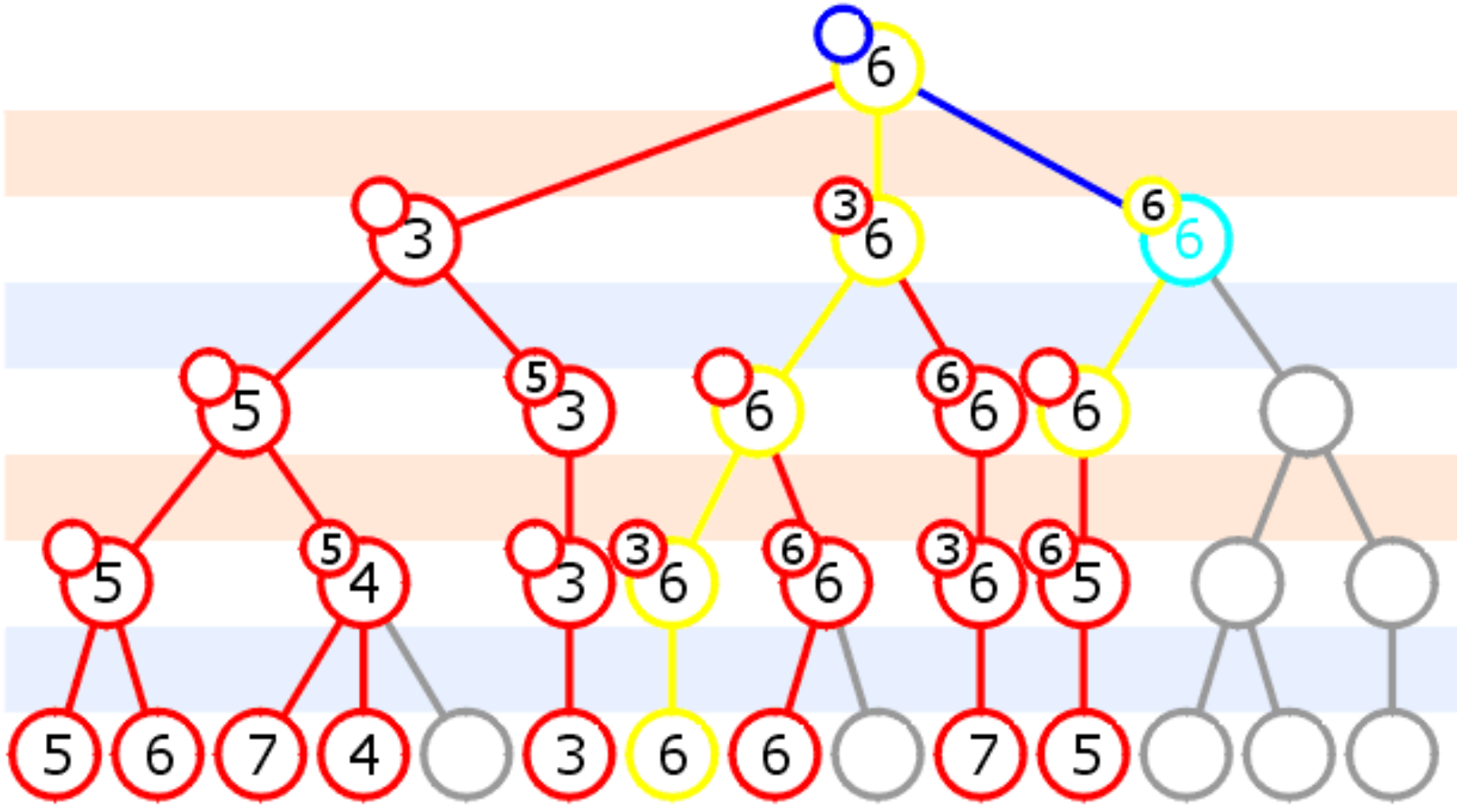
# Alpha-Beta Pruning



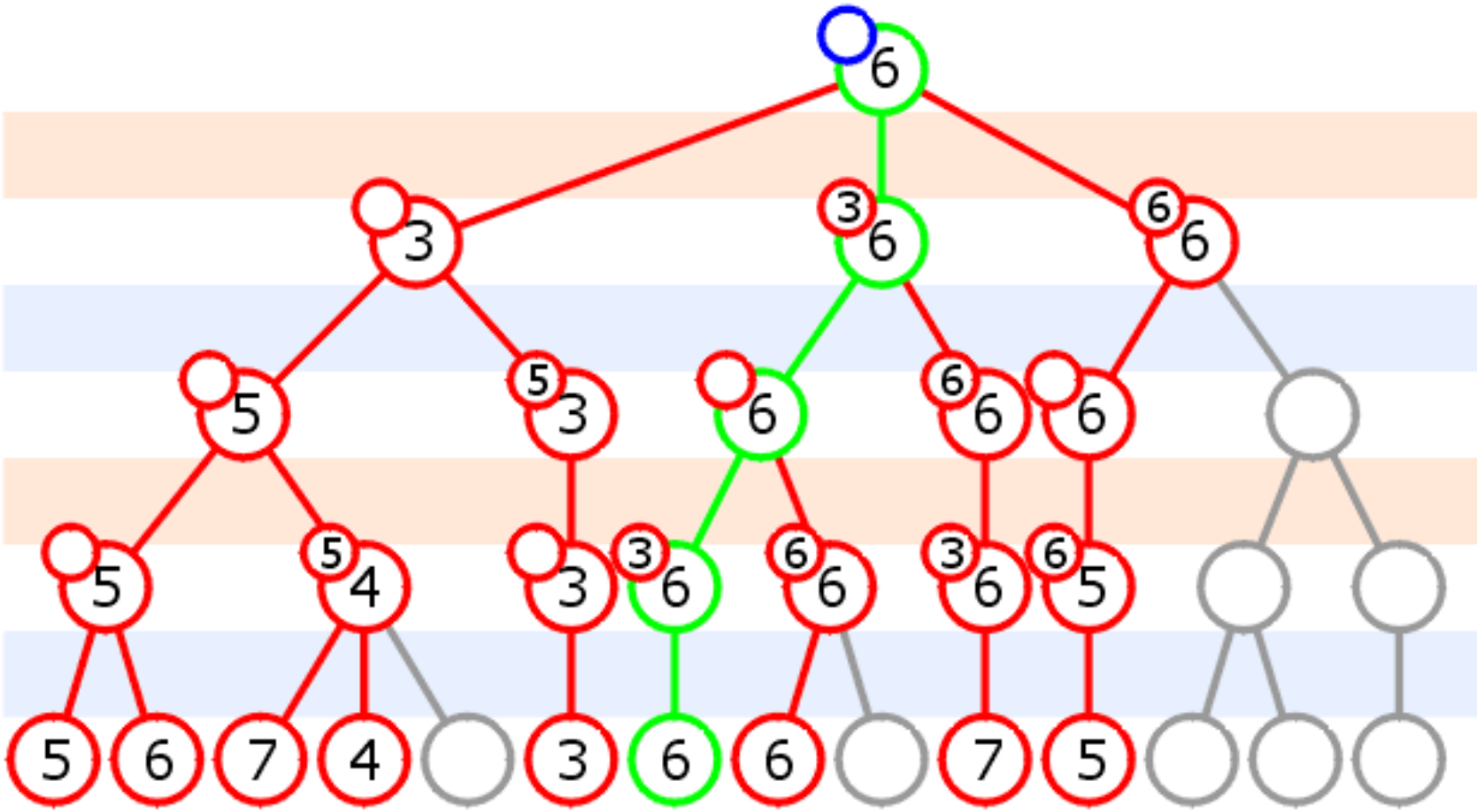
# Alpha-Beta Pruning



# Alpha-Beta Pruning



# Alpha-Beta Pruning



# Alpha-Beta Pruning

Pros:

- Reduces # of visited nodes

Cons:

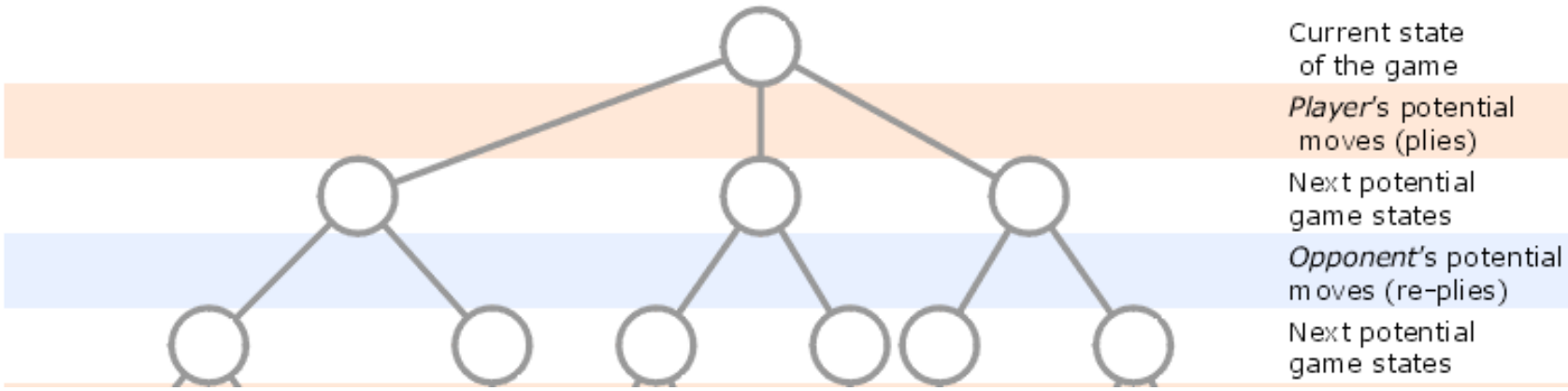
- Harder to parallelize (Message propagation top-down AND bottom-up)

# Parallel Alpha-Beta Pruning

Rough Plan:

- Parallel evaluation of nodes at a certain depth
- Cut nodes with significantly worse evaluation than best node
- Evaluate nodes ordered by evaluation value
- Allocate resources by value

# Parallel Alpha-Beta Pruning



# Goals

- Serial version of both approaches
- Parallelize both appraoches
- Determine gain from parallelization
- Compare approaches depending on resources
- Compare our approaches with existing ones

# Existing Work

- First parallel engine was less efficient than its serial version
- By 2013, 2 out of top 3 engines ran in parallel
- By now, almost no top engines run serial implementation
- Different approaches taken by developers, most use 40-80 cores

# Current state

- Serial versions of both approaches exist
- Depth considered dependent on time remaining
- Demonstration coming up...