Australian National University

COMP3620 / 6320 – S1 2022

This lecture will be recorded

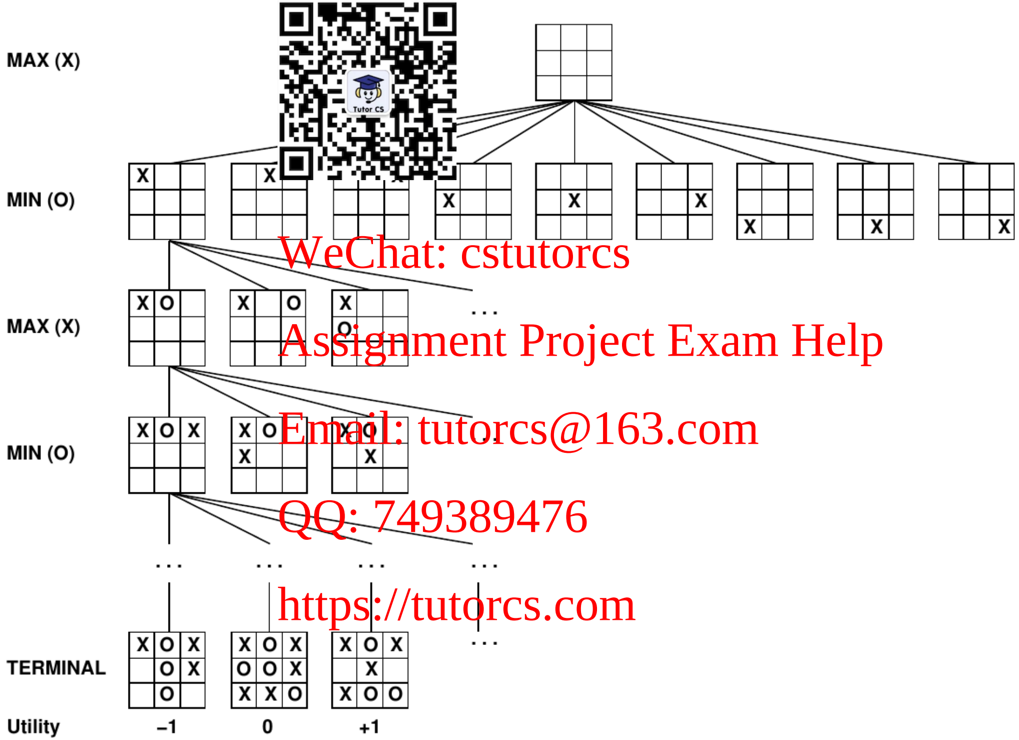
Turn off your camera if you do not want to be in the recording

If you cannot see my whole slide, go to **View Options -> Zoom Ratio** and select **Fit to Window**

If you want to ask a question, either:

- Raise your hand and I will open your mic, or
- Type the question in the chat box

Last Lecture Tic-tac-toe

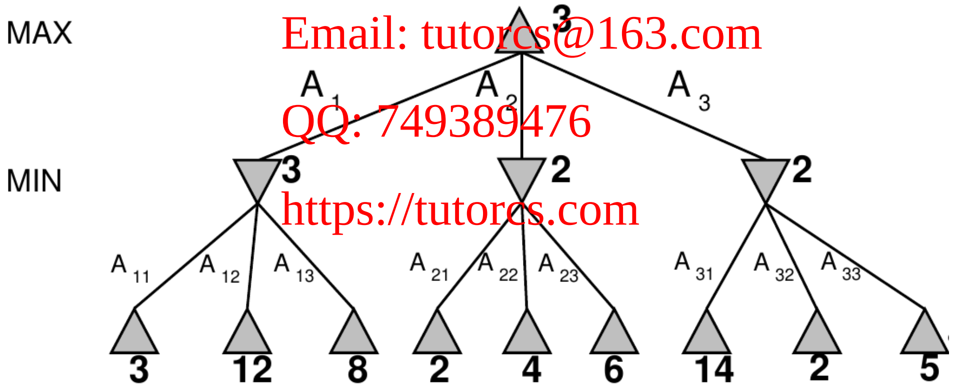


Last Lecture: Minimax

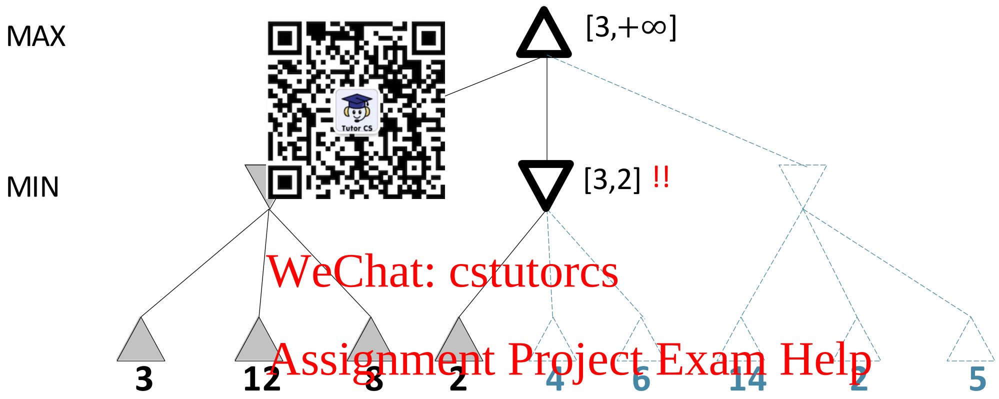
- **Perfect play** for deterministic, two-player, zero-sum, perfect-information games
- **Idea**: choose move to position with highest minimax value
  - ➢ best achievable utility **against best possible opponent**

$$\textsc{Minimax-Value}(s) = \begin{cases} \textsc{Utility}(s, \textsc{max}) & \text{if } \textsc{Terminal}(s) \\ \max_{m \in \textsc{Moves}(s)} \textsc{Minimax-Value}(\textsc{Result}(s, m)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{m \in \textsc{Moves}(s)} \textsc{Minimax-Value}(\textsc{Result}(s, m)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

E.g. 2-ply game:



Last Lecture: α–β pruning

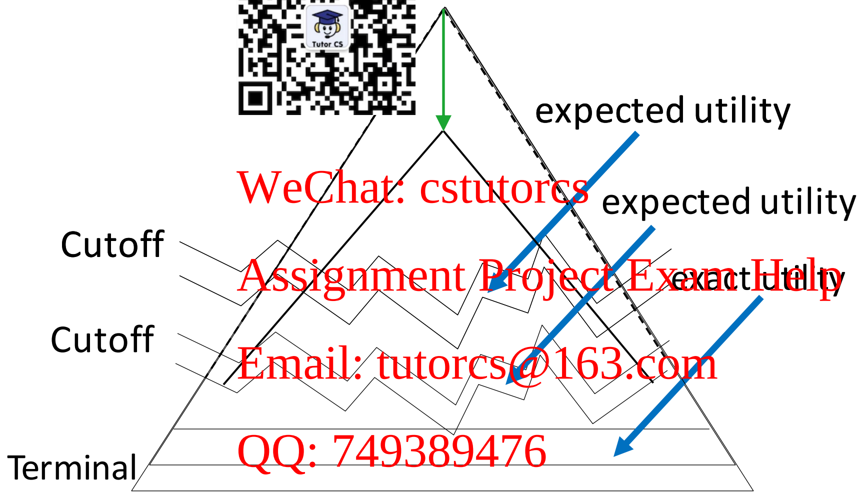


Outline for today

- Handling large state spaces in Games
- Stochastic Games

Imperfect decisions in real-time

- Approach: limit search depth and estimate expected utility



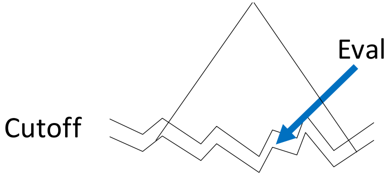
Suppose we have 100 seconds, explore 10^4 nodes/second

$\Rightarrow 10^6$ nodes per move $\approx 35^{8/2}$

$\Rightarrow \alpha\text{–}\beta$ reaches depth 8 $\Rightarrow$ pretty good chess program

Changes to Minimax

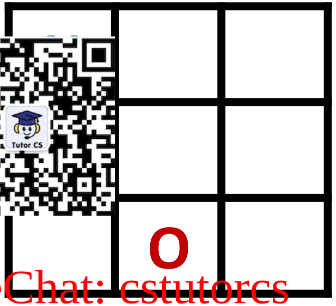
- Use Cutoff test instead of T[illegible] test
  - **Cutoff(s,d)**: true iff the stat[illegible]ntered at depth d in the tree must be considered as a leaf (or s is [illegible].
    - e.g., depth limit, estimated n[illegible]odes expanded
  - perhaps add quiescence search



- Use Eval instead of Utility
  - **Eval(s,p)** i.e., evaluation function that estimates the expected utility of cutoff state s wrt player p, and correlates with chances of winning
  - should order the terminal states in the same way as Utility
  - should not take too long

$$\text{D-MINIMAX-VALUE}(s,d) = \begin{cases} \text{EVAL}(s, \text{MAX}) & \text{if } \text{CUTOFF}(s,d) \\ \max_{m \in \text{MOVES}(s)} \text{D-MINIMAX-VALUE}(\text{RESULT}(s,m), d+1) & \text{if } \text{PLAYER}(s) = \text{MAX} \\ \min_{m \in \text{MOVES}(s)} \text{D-MINIMAX-VALUE}(\text{RESULT}(s,m), d+1) & \text{if } \text{PLAYER}(s) = \text{MIN} \end{cases}$$

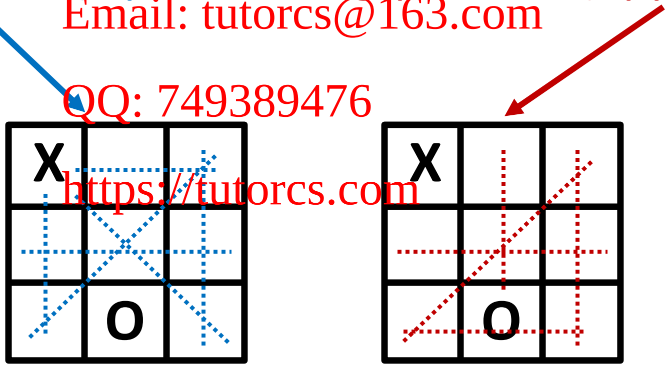
Evaluation functions



X plays next

What would be a good evaluation function for tic tac toe??

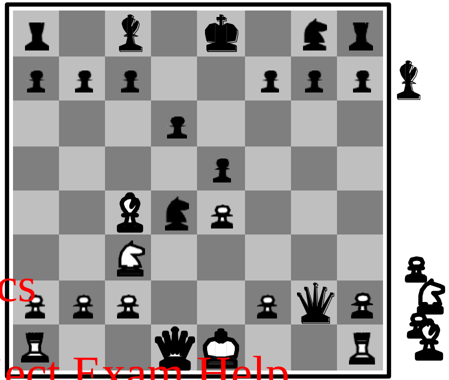
- Eval(s,p) = winning-patterns(s,p) - winning-patterns(opponent(s,p))
- Eval(s,X) = 6 - 5 =1



Evaluation functions

Black to move

White slightly better



White to move

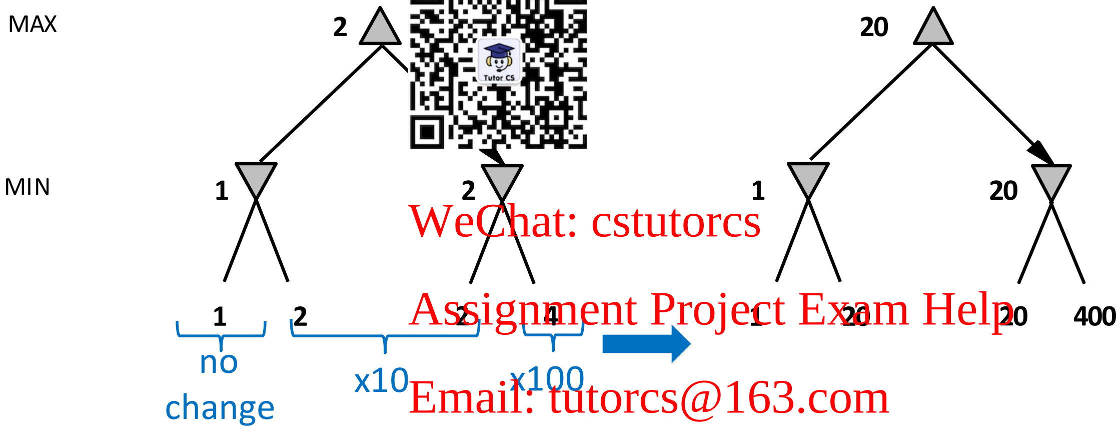
Black winning

For chess, typically linear weighted sum of features

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

e.g., $w_2 = 5$ with $f_2(s)$ = (number of white castles) – (number of black castles), etc.

Observation: Exact values don't matter



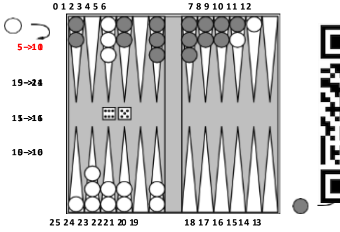
- Behavior is preserved under any non-decreasing monotonic transf. of Eval
- Only the order matters:
  - payoff in deterministic games acts as an ordinal utility function

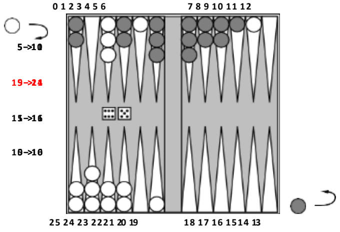
Australian National University

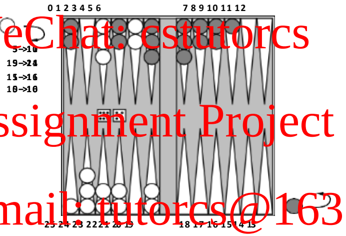
Stochastic Games

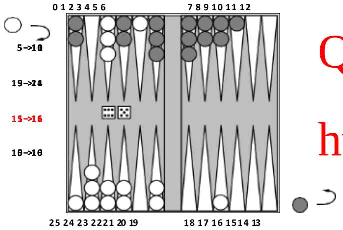
Chapter 5

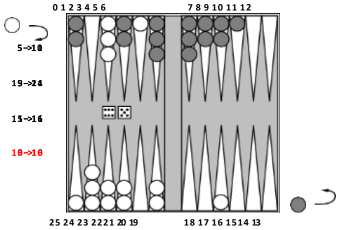
Stochastic games: backgammon





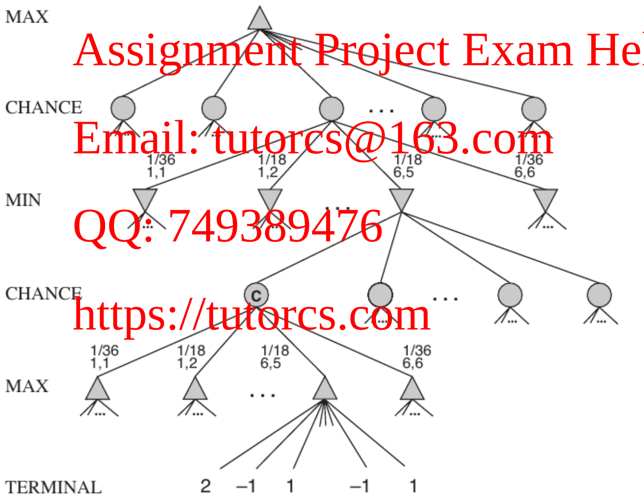






Stochastic games in general

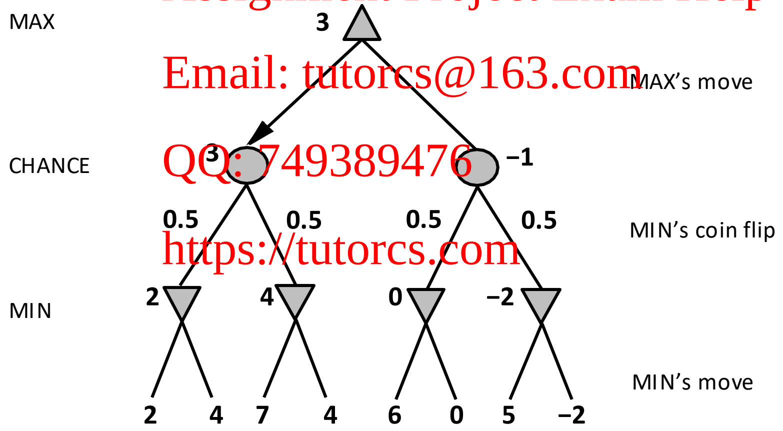
- Chance is introduced by dice, card shuffling, coin flipping.
- "Chance" can be seen as special kind of player whose move is the outcomes of a random event, which determines the space of legal moves down the tree.
- Chance is not adversarial: the value of chance positions is the expectation (average) over all possible outcomes of the value of the result.



Minimax in Stochastic Games

- ExpectiMinimax gives perfect play, also like Minimax, expect we must handle chance nodes

$$\text{EXPECTIMINIMAX-VALUE}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{m \in \text{MOVES}(s)} \text{EXPECTIMINIMAX-VALUE}(\text{RESULT}(s,m)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{m \in \text{MOVES}(s)} \text{EXPECTIMINIMAX-VALUE}(\text{RESULT}(s,m)) & \text{if PLAYER}(s) = \text{MIN} \\ ? & \text{if PLAYER}(s) = \text{CHANCE} \end{cases}$$

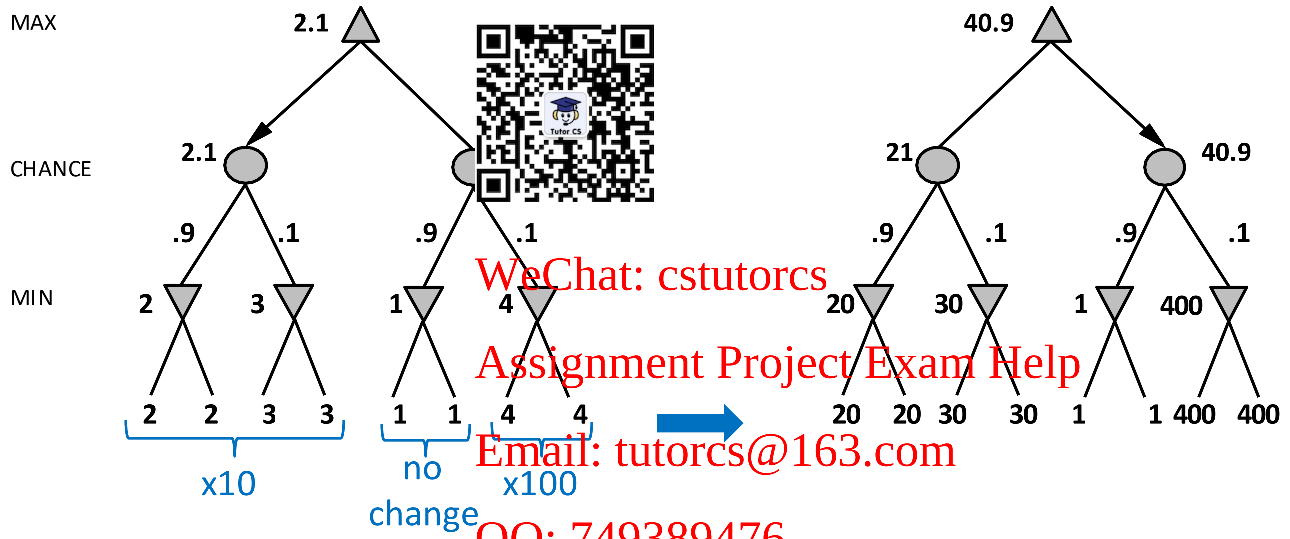


Stochastic games in practice

b: max. branching factor
m: max. depth of the state space

- Time complexity: $O(b^m n^m)$
  - n is the maximum number of outcomes of a chance event
- Dice rolls increase the effective branching factor
  - n = 21 possible rolls with 2 dice
  - b ≈ 20 legal moves in Backgammon (can be 4,000 with 1-1 roll)
  - ≈ 10^9 nodes at depth 4 of the tree
- α–β pruning is much less effective
  - value of lookahead is diminished
- TDGammon [1992] uses depth-2 search + very good Eval
  - ≈ world-champion level
  - evaluation function learnt over millions of games
  - method combined reinforcement learning with neural nets

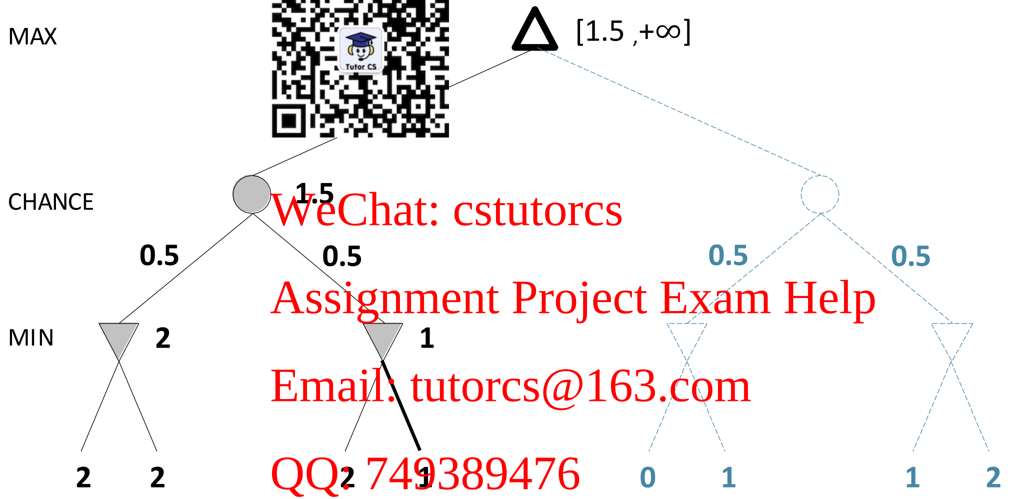
Evaluation function: exact values DO matter



- Behavior is preserved only by positive linear transformation
- Hence Eval should be proportional to the expected payoff determined by Utility

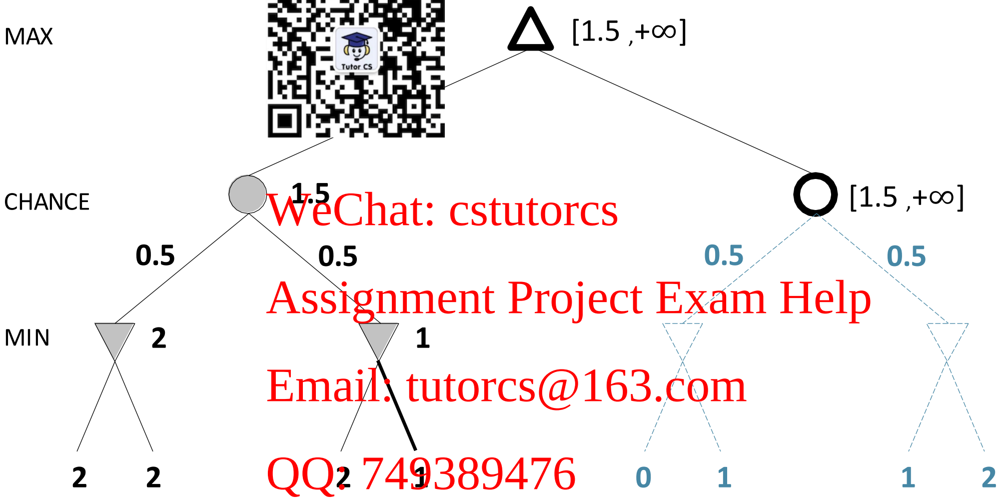
Pruning in stochastic game trees

A version of α-β pruning is possible:



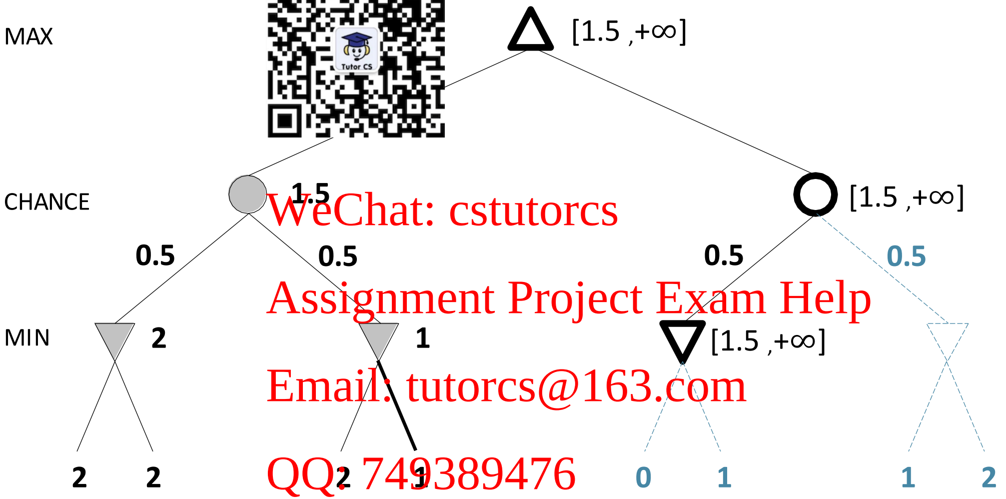
Pruning in stochastic game trees

A version of α-β pruning is possible:



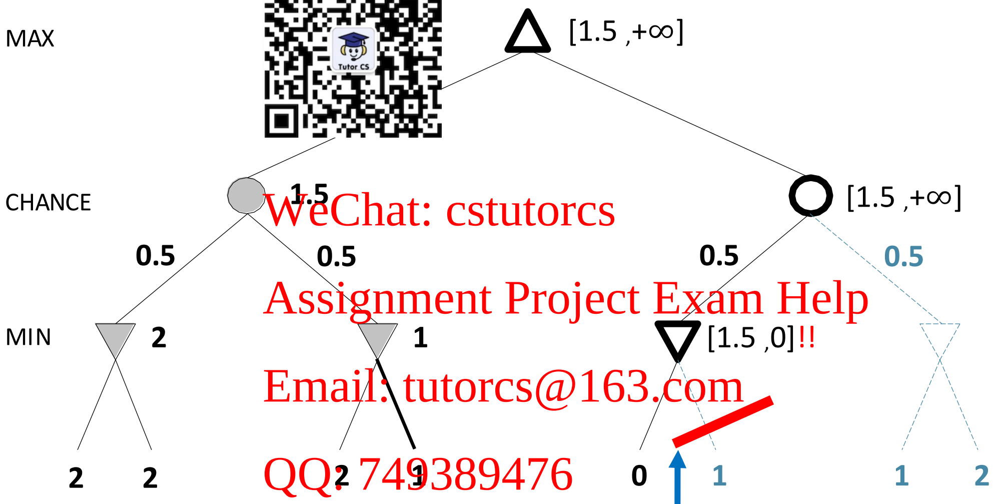
Pruning in stochastic game trees

A version of α-β pruning is possible:



Pruning in stochastic game trees

A version of α-β pruning is possible:

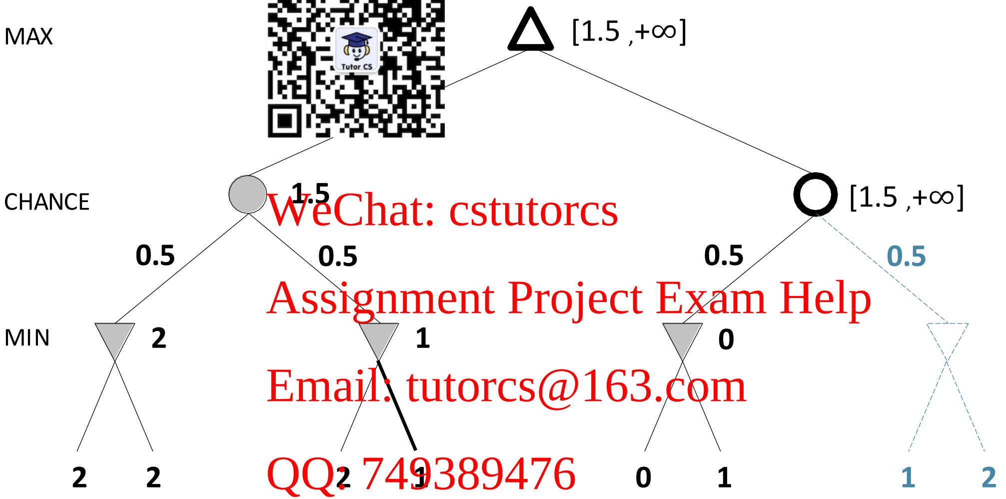


Can we prune here?

- No, the value of the chance node could still be high enough to be MAX’s choice and we need to find out exactly how high it is.

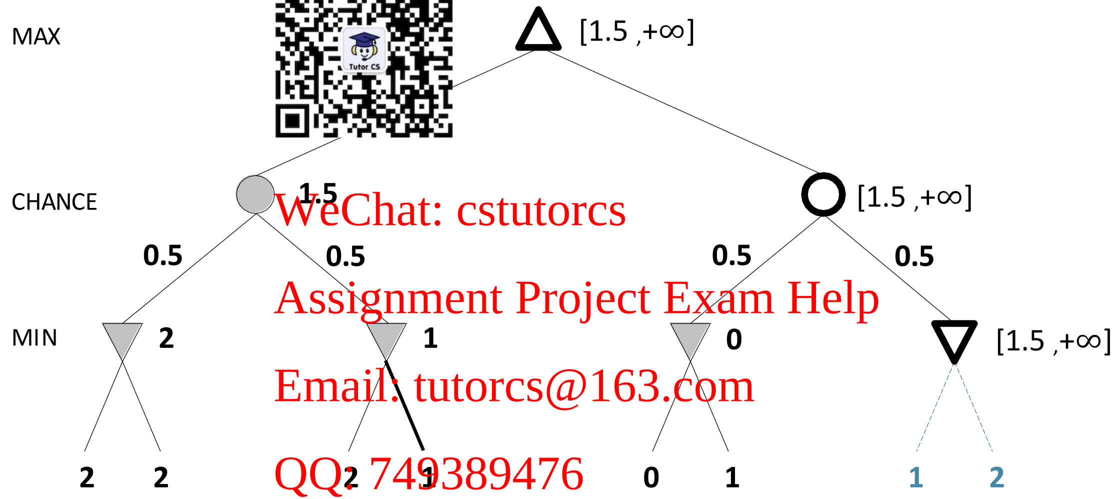
Pruning in stochastic game trees

A version of α-β pruning is possible:



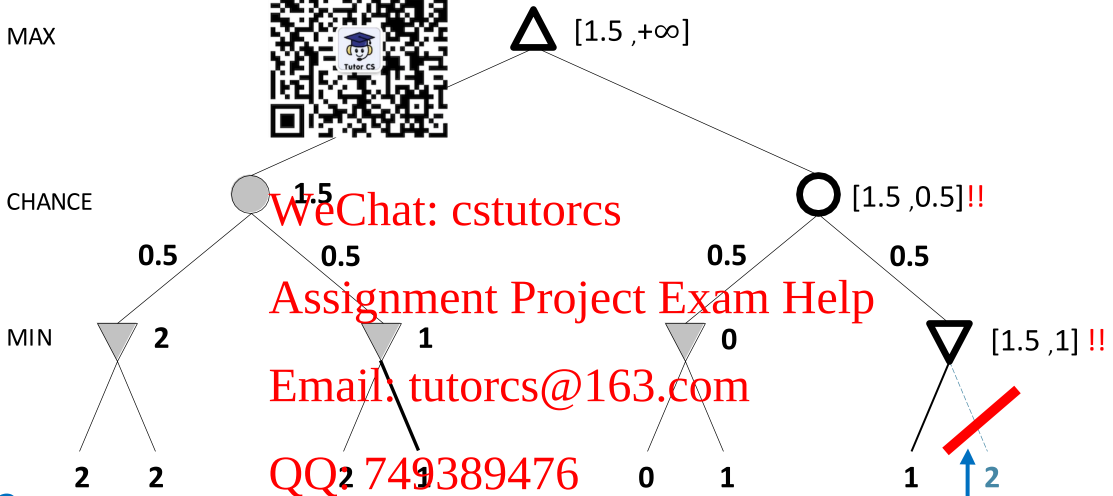
Pruning in stochastic game trees

A version of α-β pruning is possible:



Pruning in stochastic game trees

A version of α-β pruning is possible:

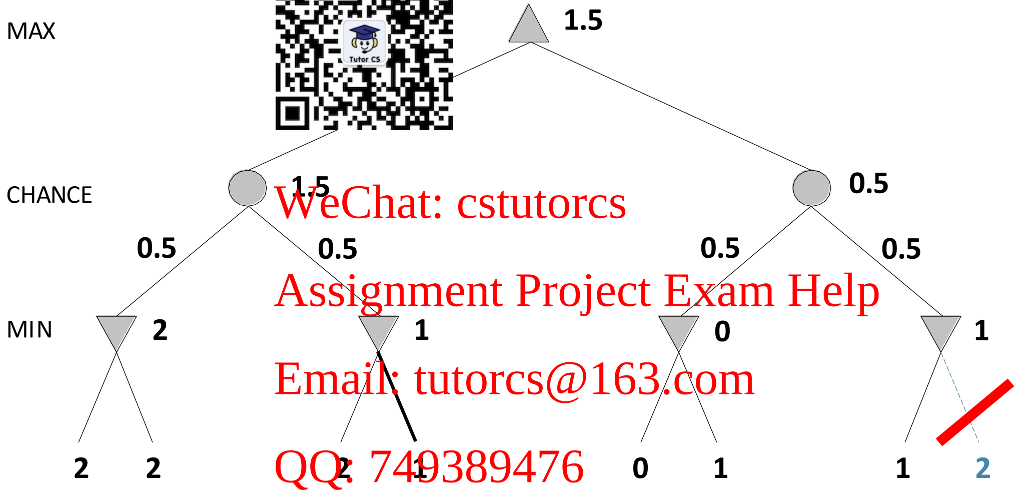


Can we prune here?

- Yes, because
  - the value of the MIN node will be **at most 1**,
  - hence the value of the CHANCE node will be **at most 0.5**,
  - which is provably **insufficient** to be MAX's choice.

Pruning in stochastic game trees

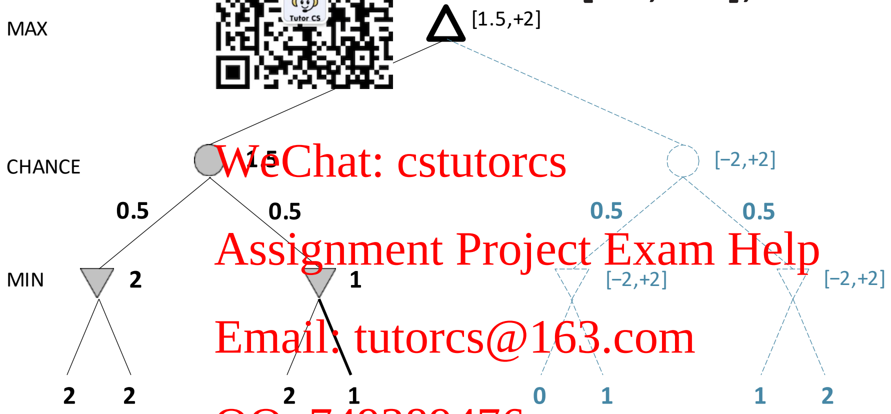
A version of α-β pruning is possible:



Pruning in stochastic game trees

More pruning occurs if we can bound the leaf values

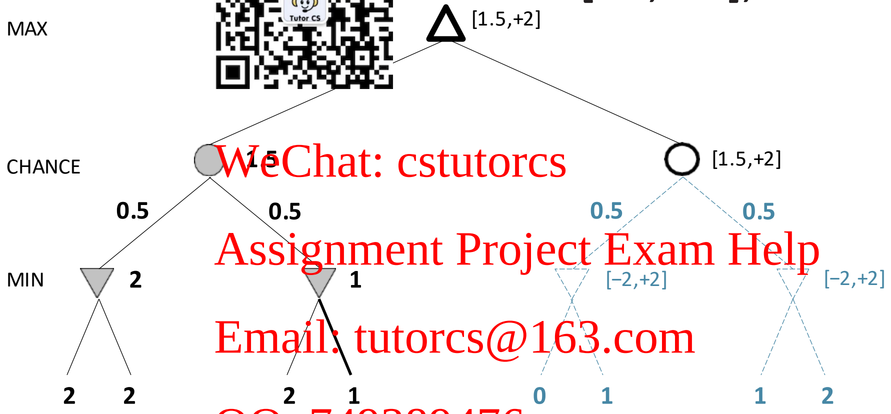
⇒ Suppose our Eval function returns values in [−2, +2], then



Pruning in stochastic game trees

More pruning occurs if we can bound the leaf values

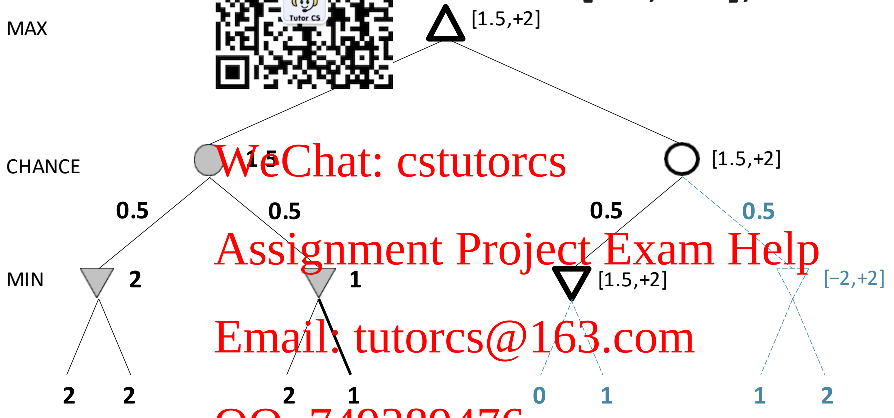
⇒ Suppose our Eval function returns values in [−2, +2], then



Pruning in stochastic game trees

More pruning occurs if we can bound the leaf values

⇒ Suppose our Eval function returns values in [−2, +2], then

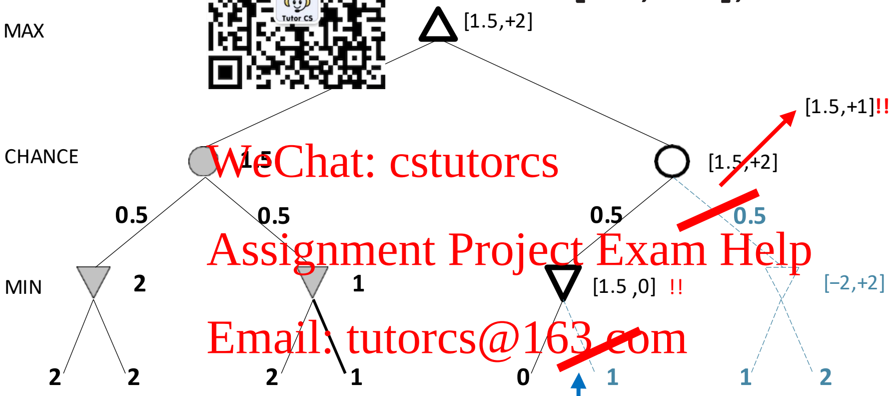


Pruning in stochastic game trees

程序代写代做 CS编程辅导

More pruning occurs if we can bound the leaf values

⇒ Suppose our Eval function returns values in [−2, +2], then



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

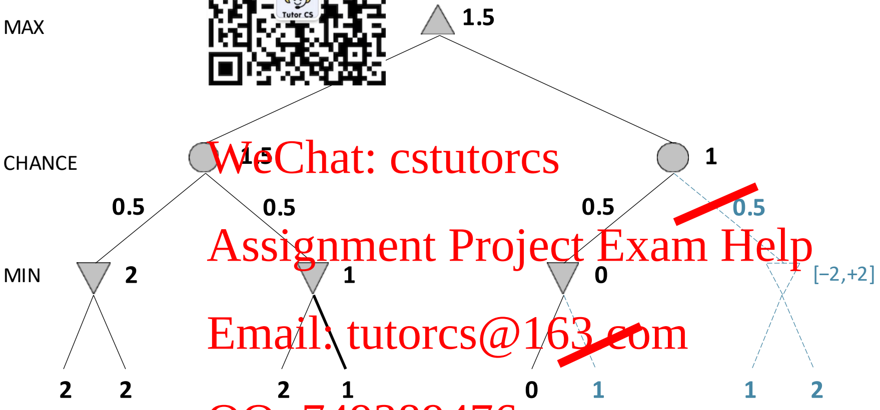
Can we prune here?

- Yes, because
  - we know that the right-hand MIN node will be worth **at most 2**
  - therefore the CHANCE node is worth **at most 1**
  - which is **not high enough** to be MAX's choice.

Pruning in stochastic game trees

More pruning occurs if we can bound the leaf values

⇒ Suppose our Eval function returns values in [−2, +2], then



Other techniques to tame complexity

- Symmetry pruning
- Monte Carlo sampling
- Iterative deepening
- Pattern databases
- Deep learning
- Monte Carlo Tree Search (MCTS)

Deterministic games in practice

- Checkers: Chinook ended 40-year reign of world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board (500 billion states). In 2007 Checkers became the largest game to be completely **solved** (with 500 × 10^{20} states!) at the time.
- Chess: Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per sec, i.e. 300 billion per move (depth 14), uses very sophisticated evaluation (8000 features), and singular extensions to extend some search lines up to 40 plays.
- Go: branching factor b > 300 made this more challenging. Monte Carlo Tree Search (MCTS) is the method of choice. Zen defeated a 9 Dan in 2013. In 2016 AlphaGo won 4-1 against Lee Sedol, using deep learning to learn a value function and policy to guide MCTS.
- General game playing: playing any game from the rules of the game. Poker: the next big thing! But it's a stochastic game!

Summary

- A Game is defined by an initial state, a successor function, a terminal test, and a utility function
- The minimax algorithm select optimal actions for two-player zero-sum games of perfect information by a depth first exploration of the game-tree
- α-β pruning does not compromise optimality but increases efficiency by eliminating provably irrelevant subtrees
- It is not feasible to consider the whole game tree (even with α-β), so we need to cut the search off at some point and apply an evaluation function that gives an estimate of the expected utility of a state
- Game trees and minimax can be extended to stochastic games by introducing chance nodes whose value is the expectation of that of their successors
- The value of α-β and lookahead is limited in stochastic games
  - ➢ the evaluation function needs to compensate