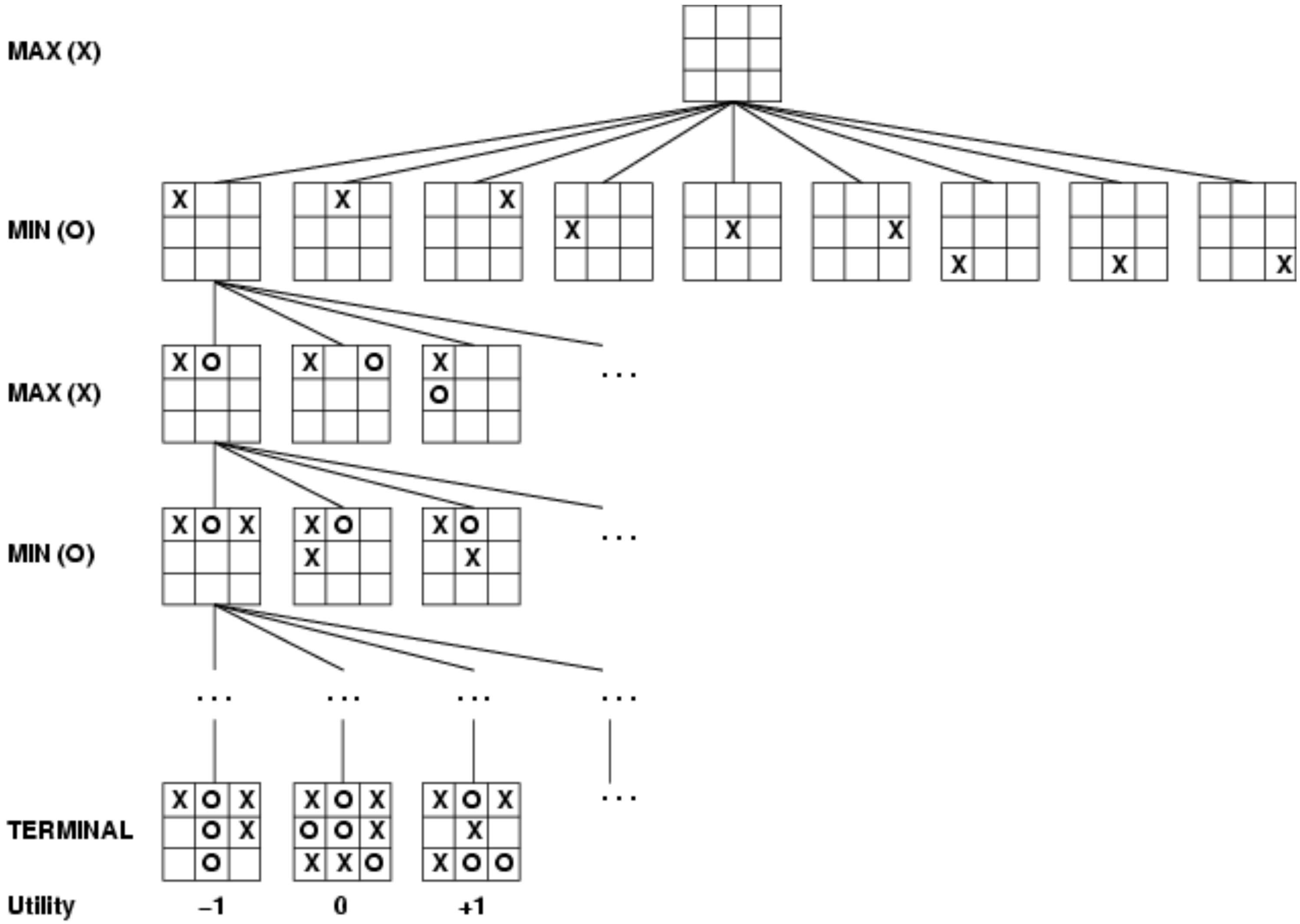
Adversarial Search Continued: Stochastic Games

CMPSCI 383
October 4, 2011

Tip for doing well

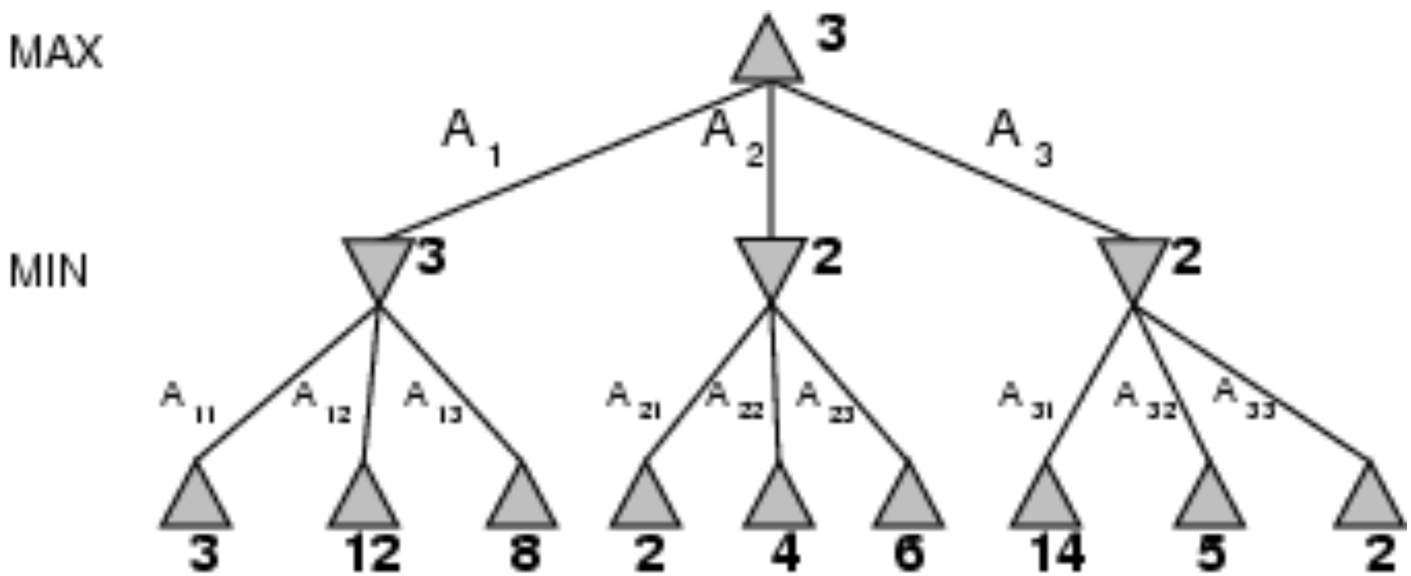
- Begin work on assignments early!

Game tree (2=player, deterministic, turns)

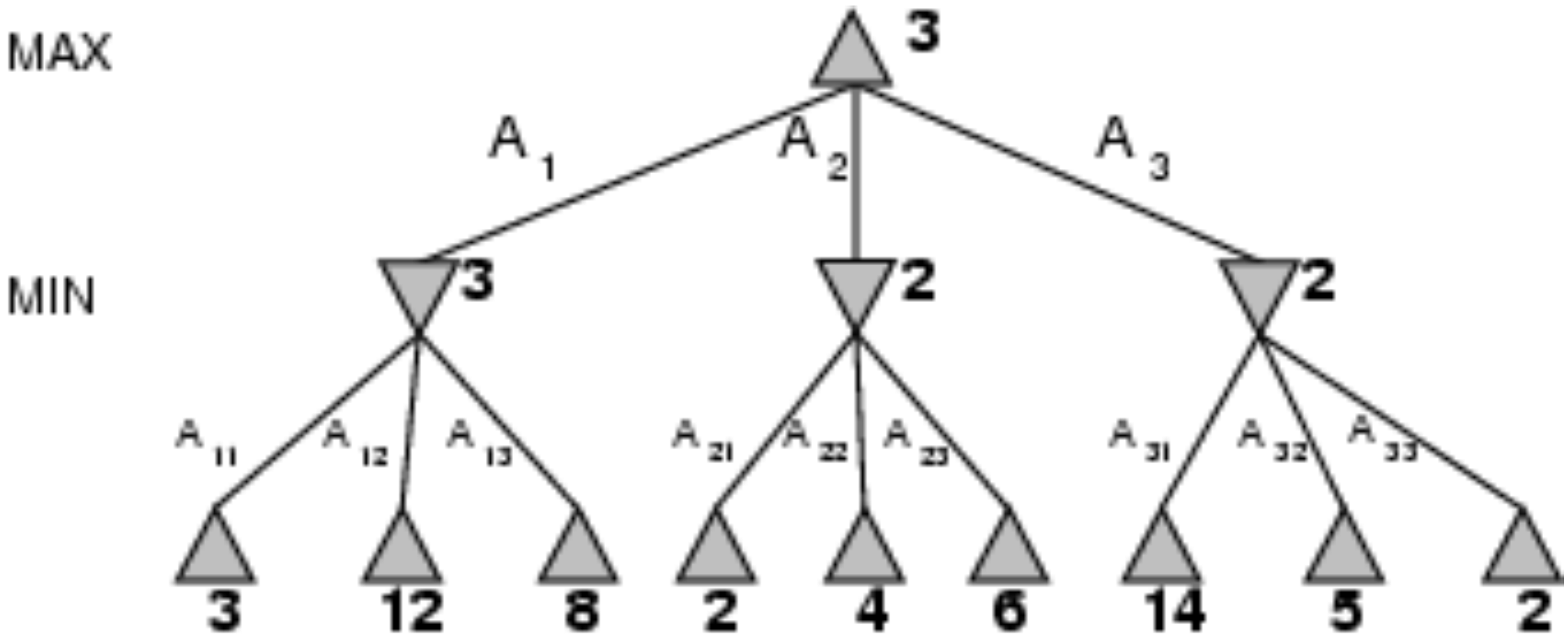


Minimax algorithm

- Perfect play for deterministic games
- Idea — select moves with highest minimax value. That is, select the best achievable payoff against best play by your opponent

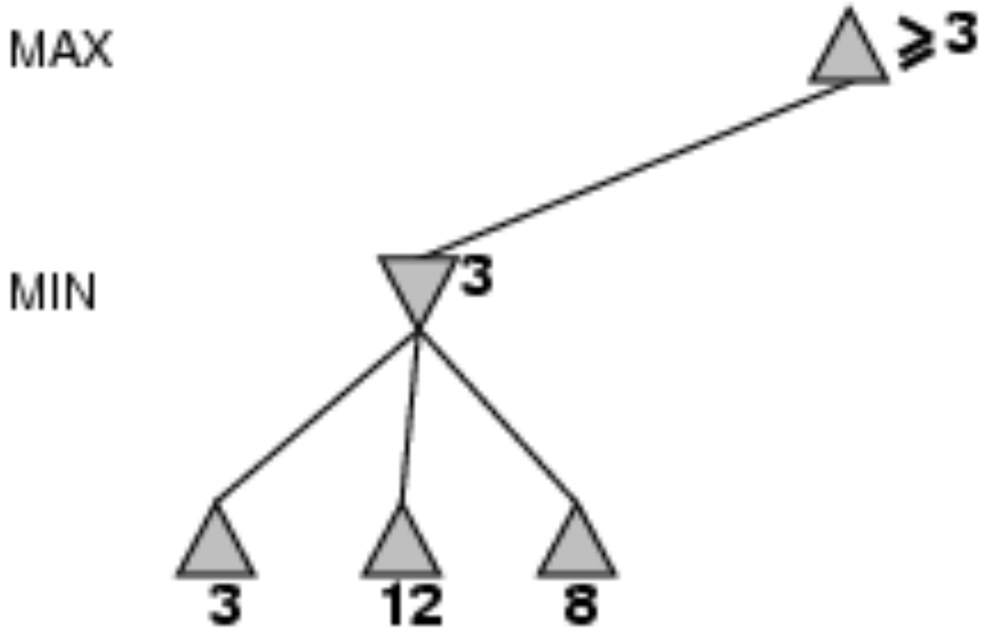


α-β pruning

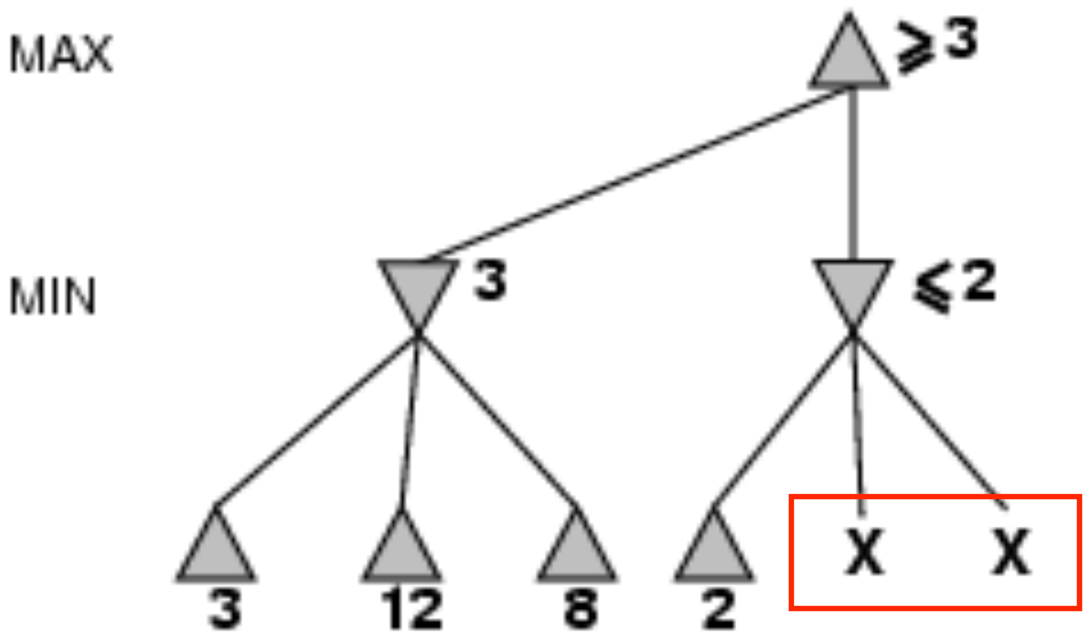


MAX
MIN
3
A 1
A 2
A 3
3
2
2
A 11
A 12
A 13
A 21
A 22
A 23
A 31
A 32
A 33
3
12
8
2
4
6
14
5
2

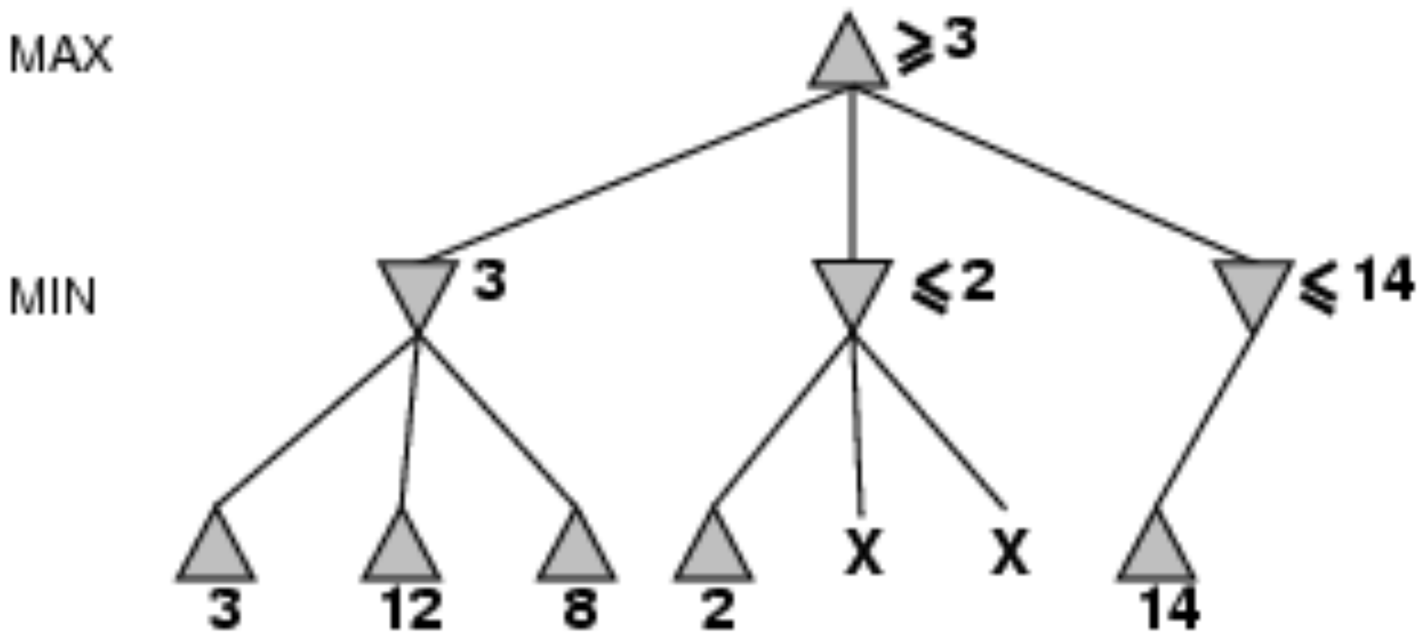
α-β pruning



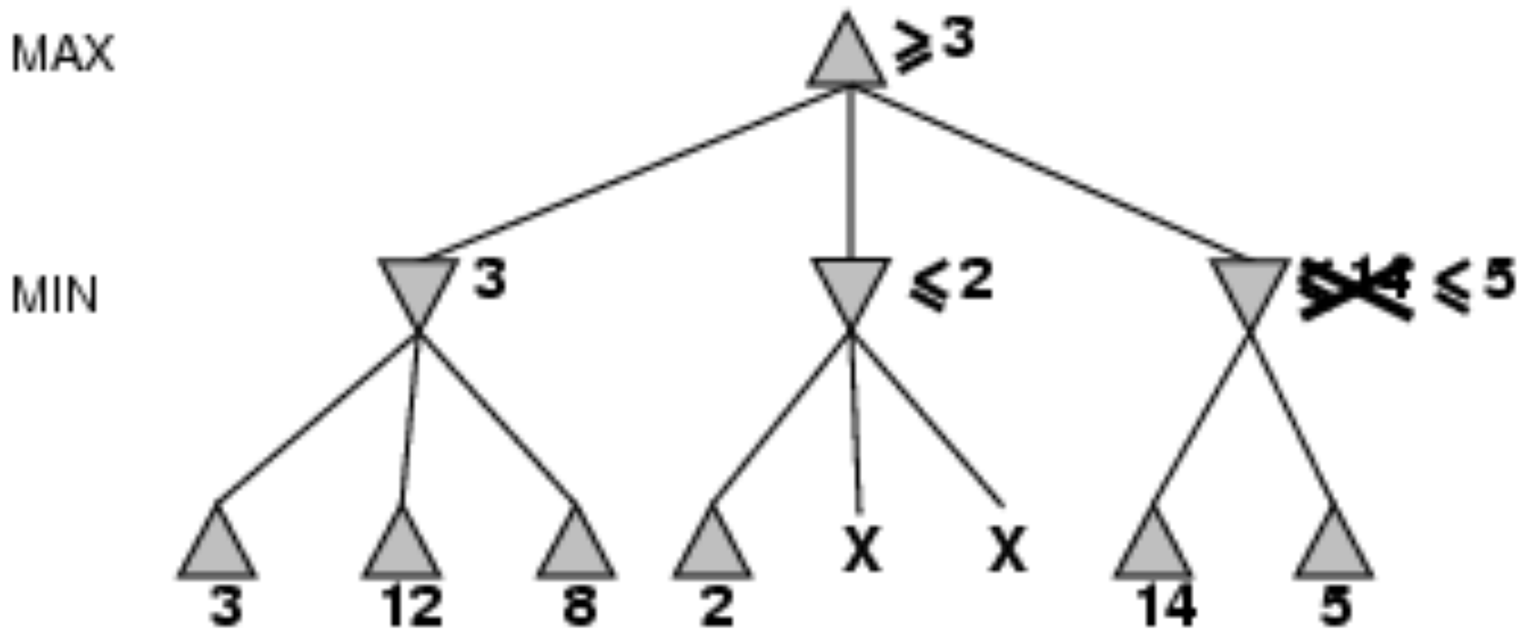
Example: α-β pruning



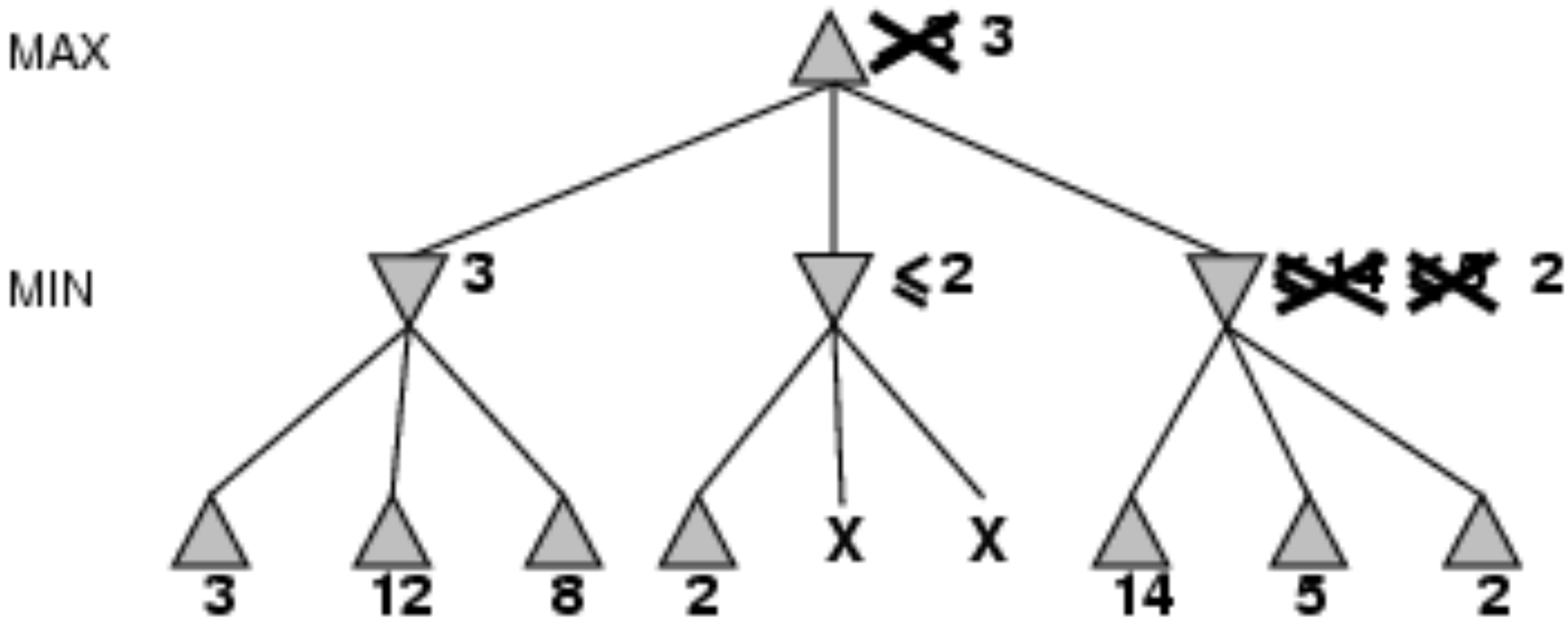
Example: α-β pruning



Example: α-β pruning



Example: α-β pruning

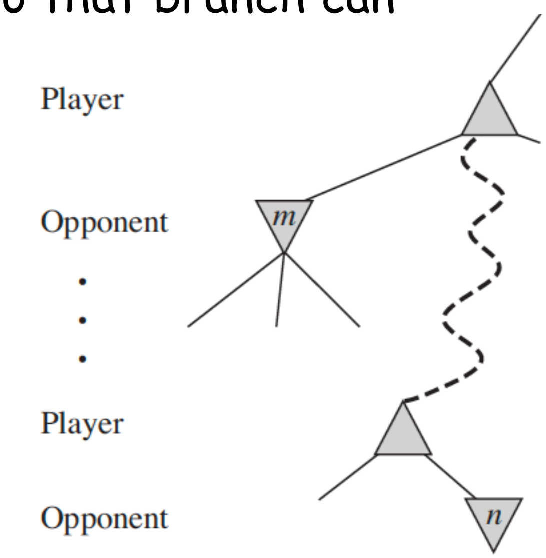


Why is it called α-β?

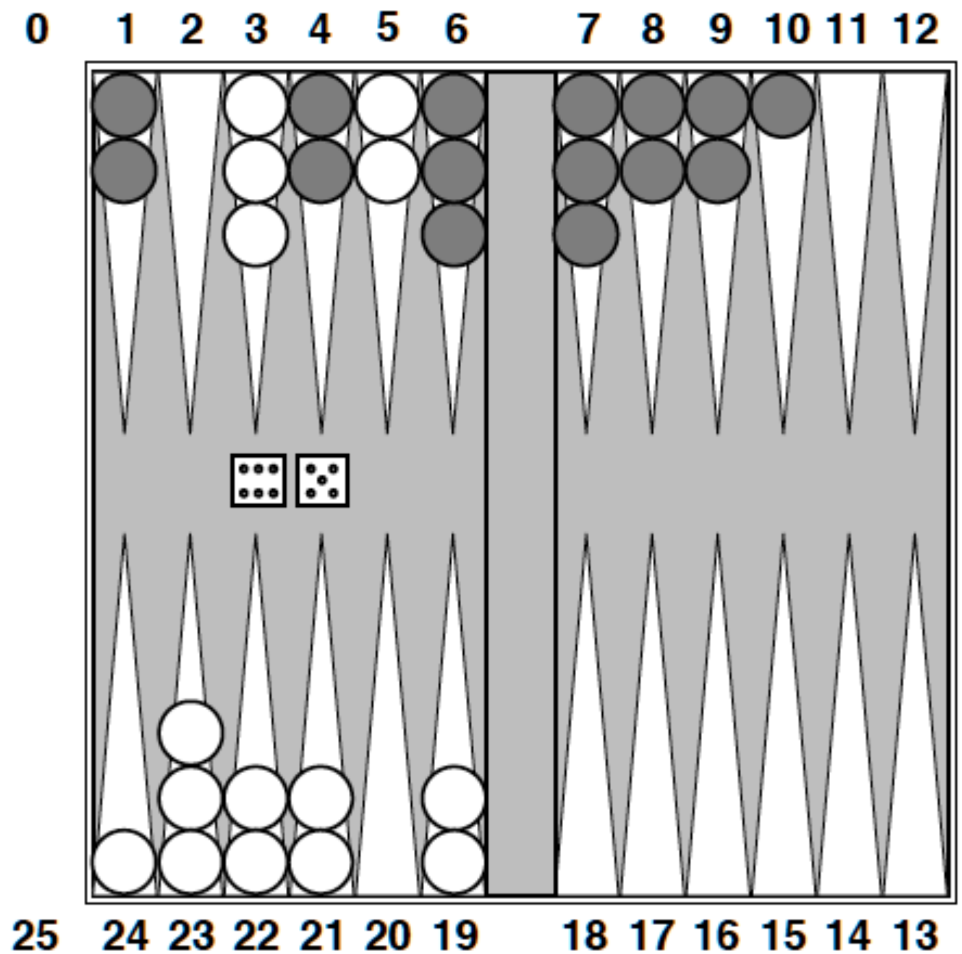
- α is the value of the best (highest-value) choice found so far at any choice point along the path for *MAX*
  - If *v* is worse than α, *MAX* will avoid it, so that branch can be pruned.

- β is the value of the best (lowest-value) choice found so far at any choice point along the path for *MIN*
  - If *v* is worse than β , *MIN* will avoid it, so that branch can be pruned.

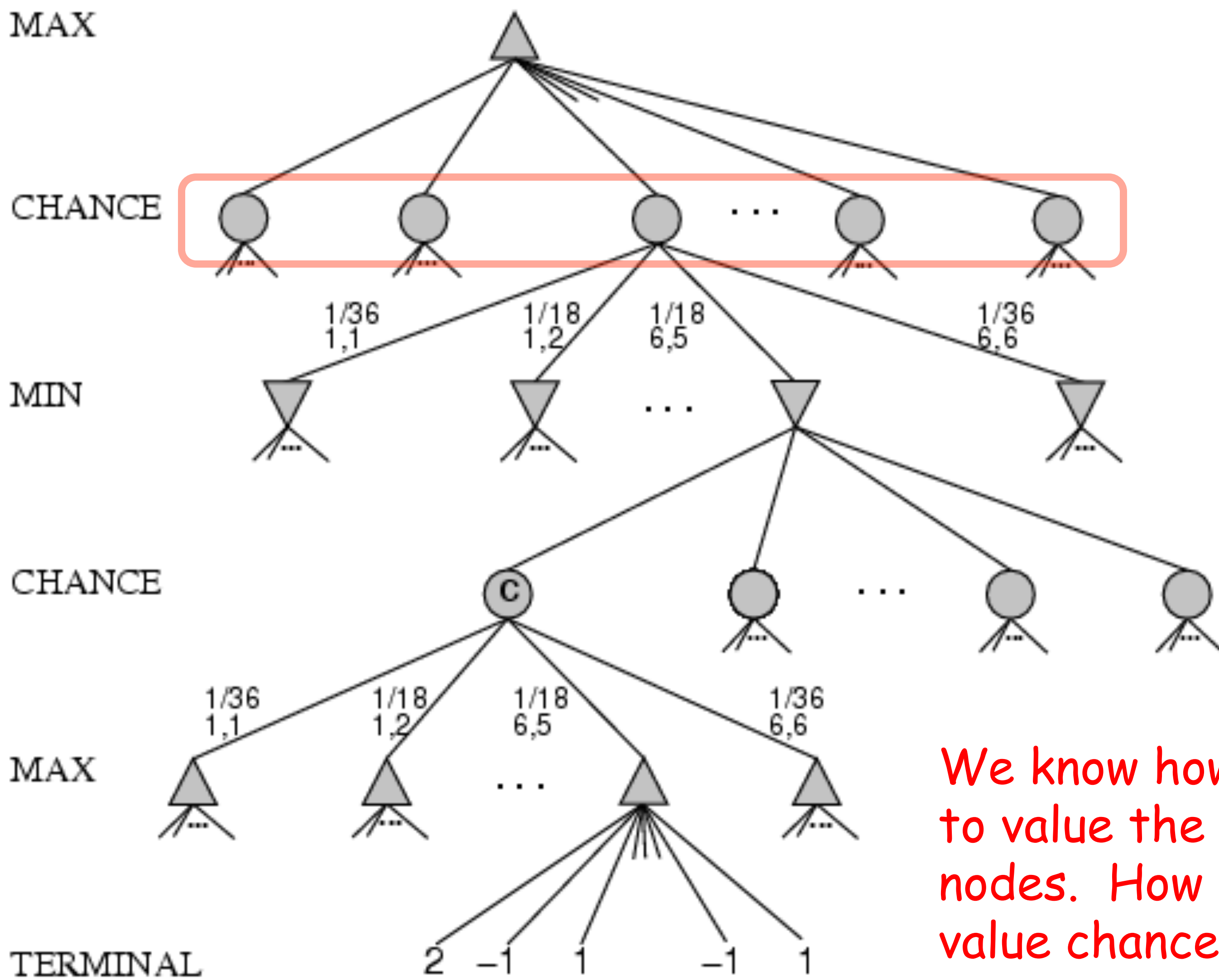
If *m* is better than *n*, we will never get to n



What if a game has a "chance element"?



Chance nodes



We know how to value the other nodes. How do we value chance nodes?

Expected value

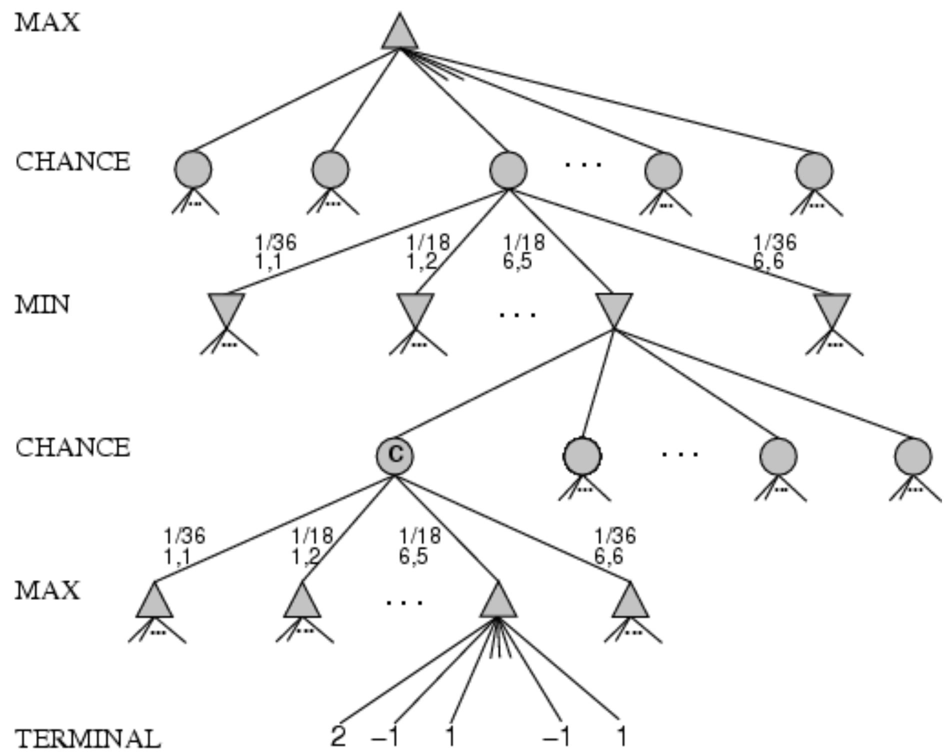
- The sum of the probability of each possible outcome multiplied by its value:

$$E(X) = \sum_i p_i x_i$$

- Are there pathological cases where this statistic could do something strange?
  - Extreme values ("outliers")
  - Functions that are a non-linear transformation of the probability of winning

Expected minimax value

- Now *three* different cases to evaluate, rather than just two.
  - MAX
  - MIN
  - CHANCE



EXPECTIMINIMAX(n) =

UTILITY(n), If terminal node

$\max_{s \in successors(n)}$ EXPRCTIMINIMAX(s), If n is MAX node

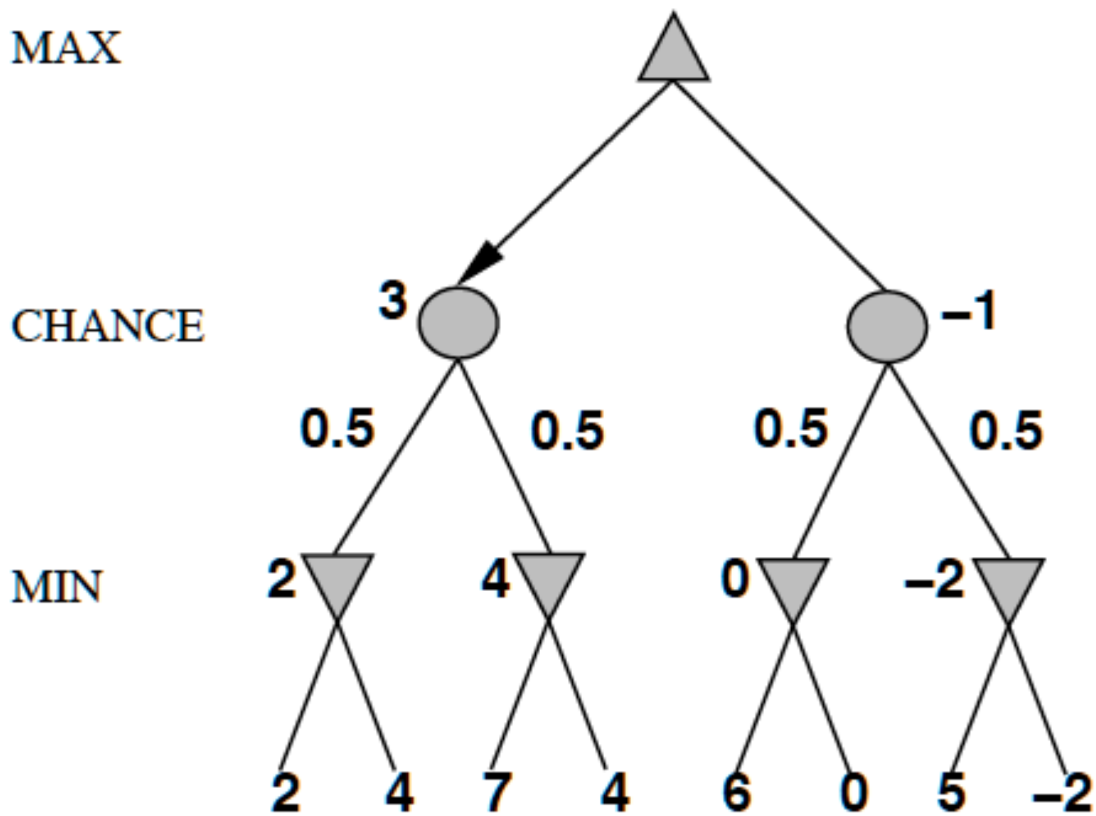
$\min_{s \in successors(n)}$ EXPECTIMINIMAX(s), If n is MIN node

$\sum_{s \in successors(n)} P(s)$ • EXPECTIMINIMAX(s), If n is CHANCE node

Expectiminimaxing

In nondeterministic games, chance introduced by dice, card-shuffling

Simplified example with coin-flipping:



In Backgammon

Dice rolls increase b: 21 possible rolls with 2 dice
Backgammon $\approx$ 20 legal moves (can be 6,000 with 1-1 roll)

$$\text{depth } 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$

As depth increases, probability of reaching a given node shrinks
$\Rightarrow$ value of lookahead is diminished

α–β pruning is much less effective

TDGAMMON uses depth-2 search + very good EVAL
$\approx$ world-champion level

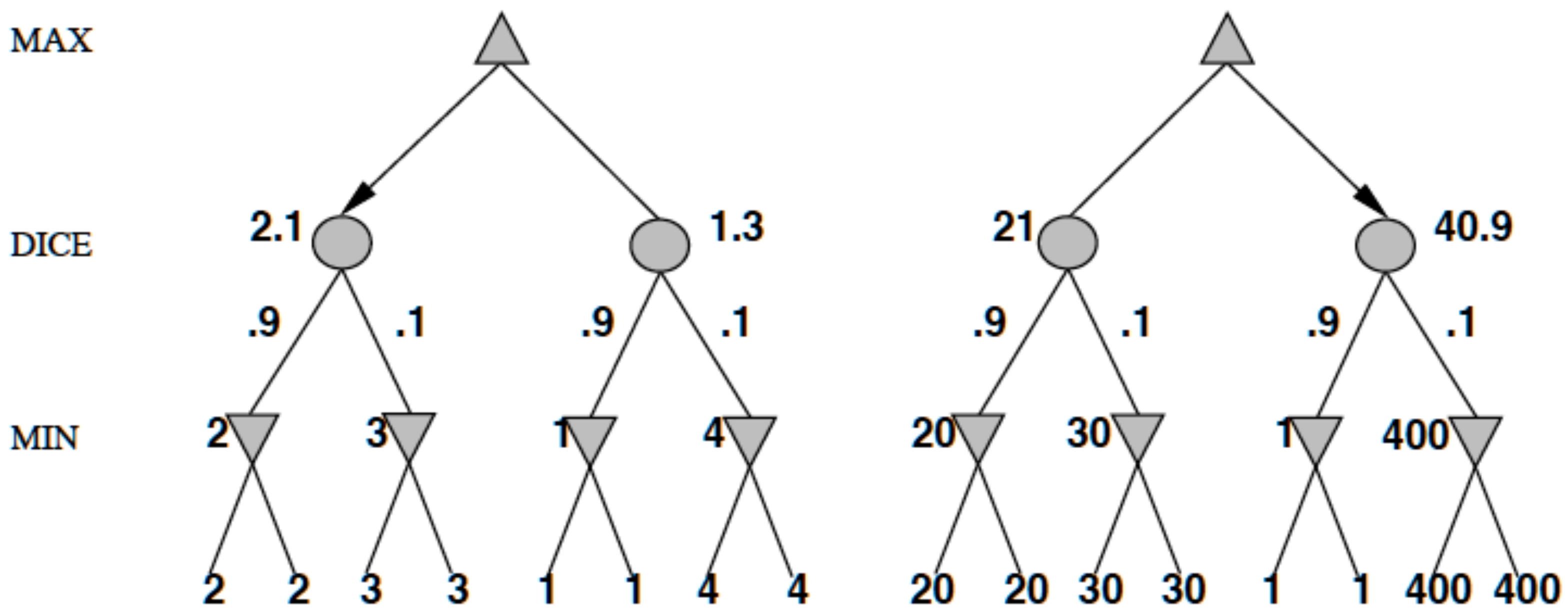
Evaluation functions

- So cut off the search and evaluate leaves with an evaluation function (as in *H-MINIMAX*)
- But do evaluation functions behave the same way in stochastic games?
  - Just need to order nodes in the right way, so particular values are not so important.

- Chance nodes make things more difficult.

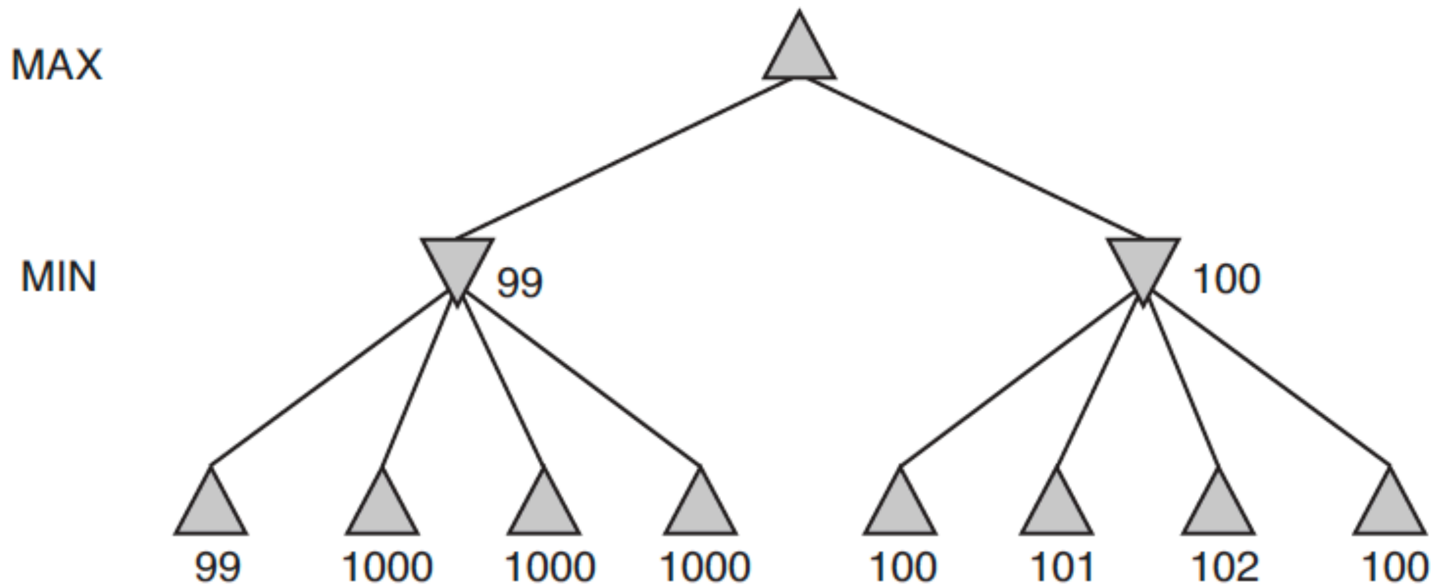
Particular values DO matter

Order-preserving transformation →



Behaviour is preserved only by positive linear transformation of EVAL

Hence EVAL should be proportional to the expected payoff


MAX
MIN
99
100
99
1000
1000
1000
100
101
102
100

Partially Observable Games
(Games of Imperfect Information)

E.g., card games, where opponent's initial cards are unknown

Typically we can calculate a probability for each possible deal

Seems just like having one big dice roll at the beginning of the game*

Idea: compute the minimax value of each action in each deal,
then choose the action with highest expected value over all deals*

$$argmax_a \sum_s P(s)\ \text{MINIMAX}(RESULT(s,a)),$$

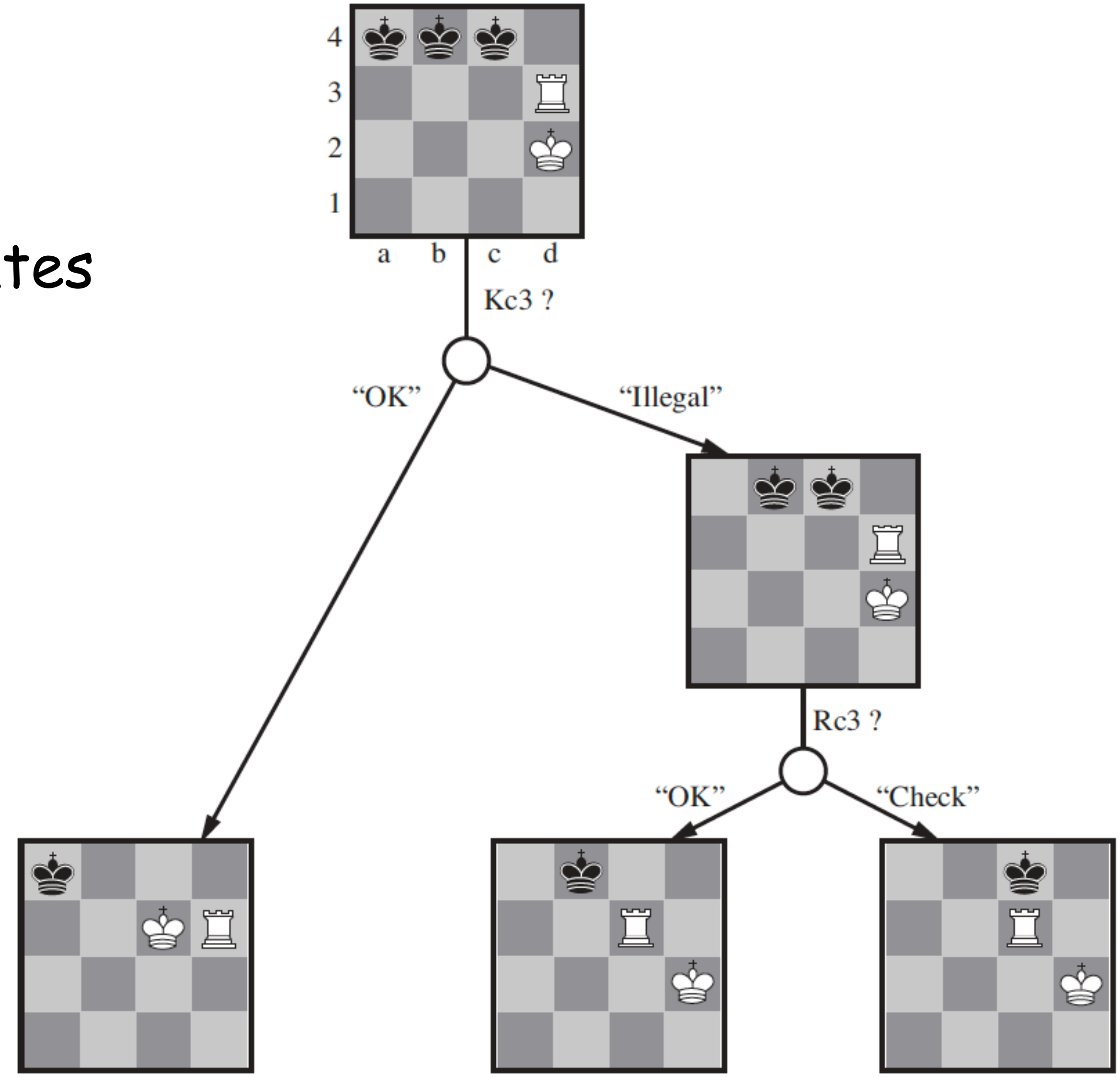
Monte Carlo approximation:

$$argmax_a\ 1/N \sum_{i=1}^{N} \text{MINIMAX}(RESULT(s_i,a)),$$

"averaging over clairvoyance"

Example: Kriegspiel

Maintain
belief states



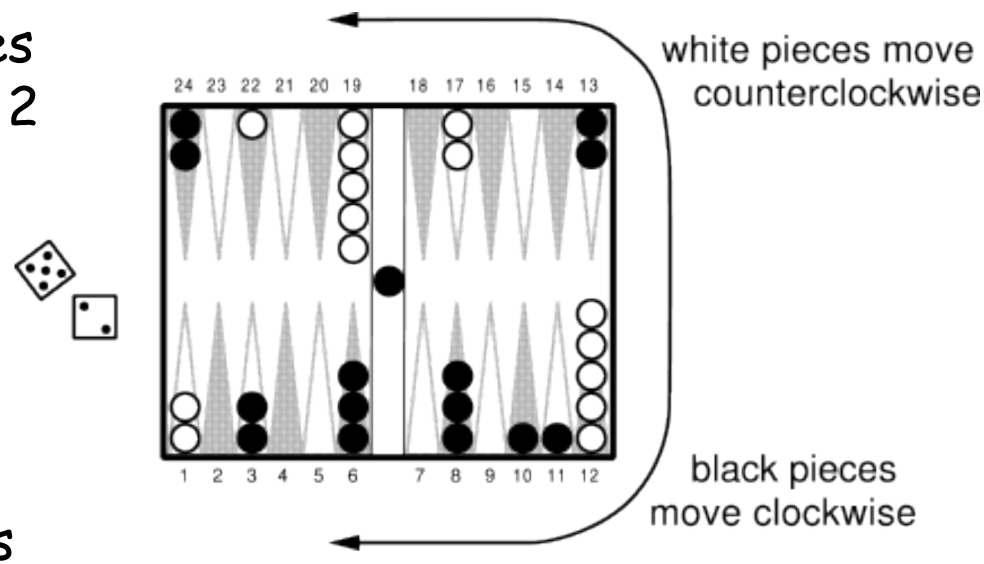
Rollouts

- Play out a position to completion several thousand times with different random dice sequences.
- The best play is assumed to be the one that produced the best outcome statistics in the rollout.
- What program should select moves?

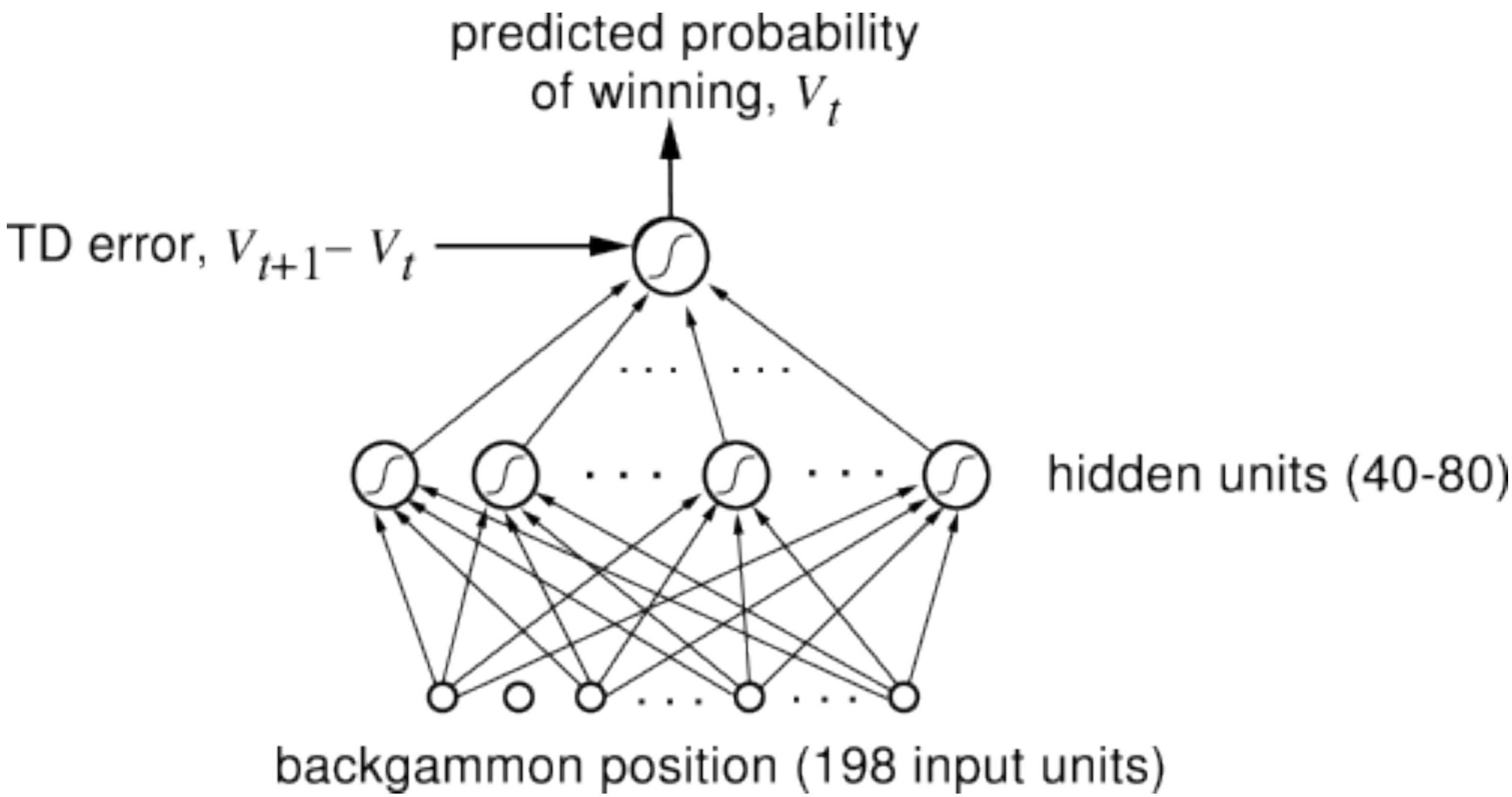
TD Gammon

Tesauro 1992, 1994, 1995, ...

- White has just rolled a 5 and a 2 so can move one of his pieces 5 and one (possibly the same) 2 steps
- Objective is to advance all pieces to points 19-24
- Hitting
- Doubling
- 30 pieces, 24 locations implies enormous number of configurations
- Effective branching factor of 400



Multi-layer Neural Network



Summary of TD-Gammon Results

Program	Hidden Units	Training Games	Opponents	Results
TD-Gam 0.0	40	300,000	other programs	tied for best
TD-Gam 1.0	80	300,000	Robertie, Magriel, . . .	−13 points / 51 games
TD-Gam 2.0	40	800,000	various Grandmasters	−7 points / 38 games
TD-Gam 2.1	80	1,500,000	Robertie	−1 point / 40 games
TD-Gam 3.0	80	1,500,000	Kazaros	+6 points / 20 games

Bill Robertie: world-class human grandmaster and former World Champion

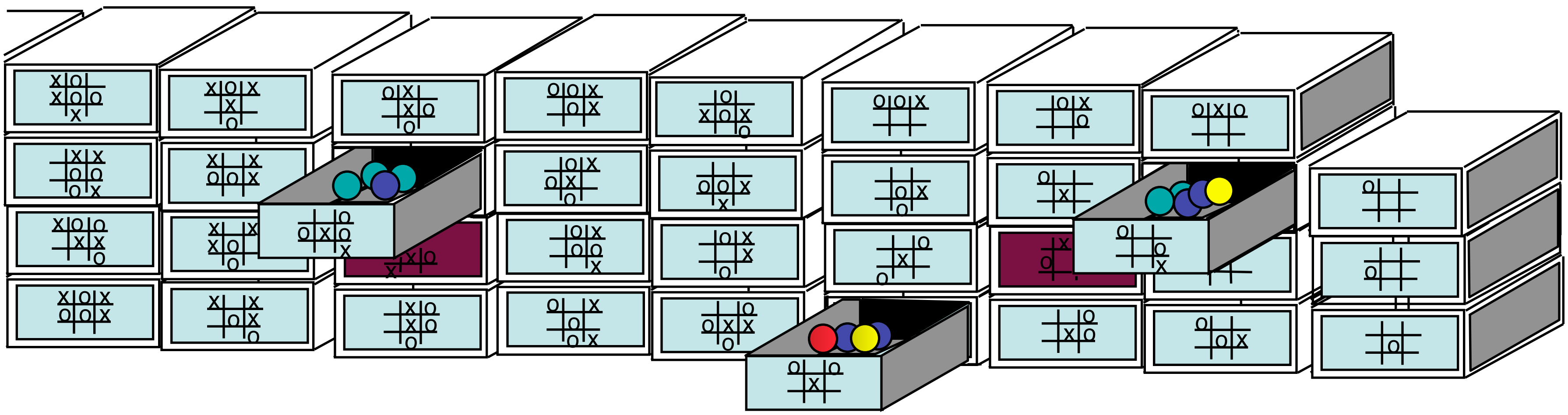
TD-Gammon 2.1 plays at a strong master level that is extremely close to equaling the world's best human planers. ….would be the favorite in a long money game session or grueling tournament like the World Cup competition (it never gets tired or careless).

A Few Details

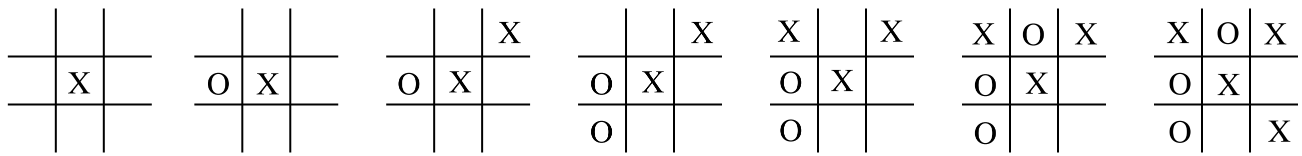
- Reward: 0 at all times except those in which the game is won, when it is 1
- Episodic (game = episode), undiscounted
- Gradient descent TD(λ) with a multi-layer neural network
  - weights initialized to small random numbers
  - backpropagation of TD error
  - four input units for each point; unary encoding of number of white pieces, plus other features
- Use of afterstates
- Learning during self-play

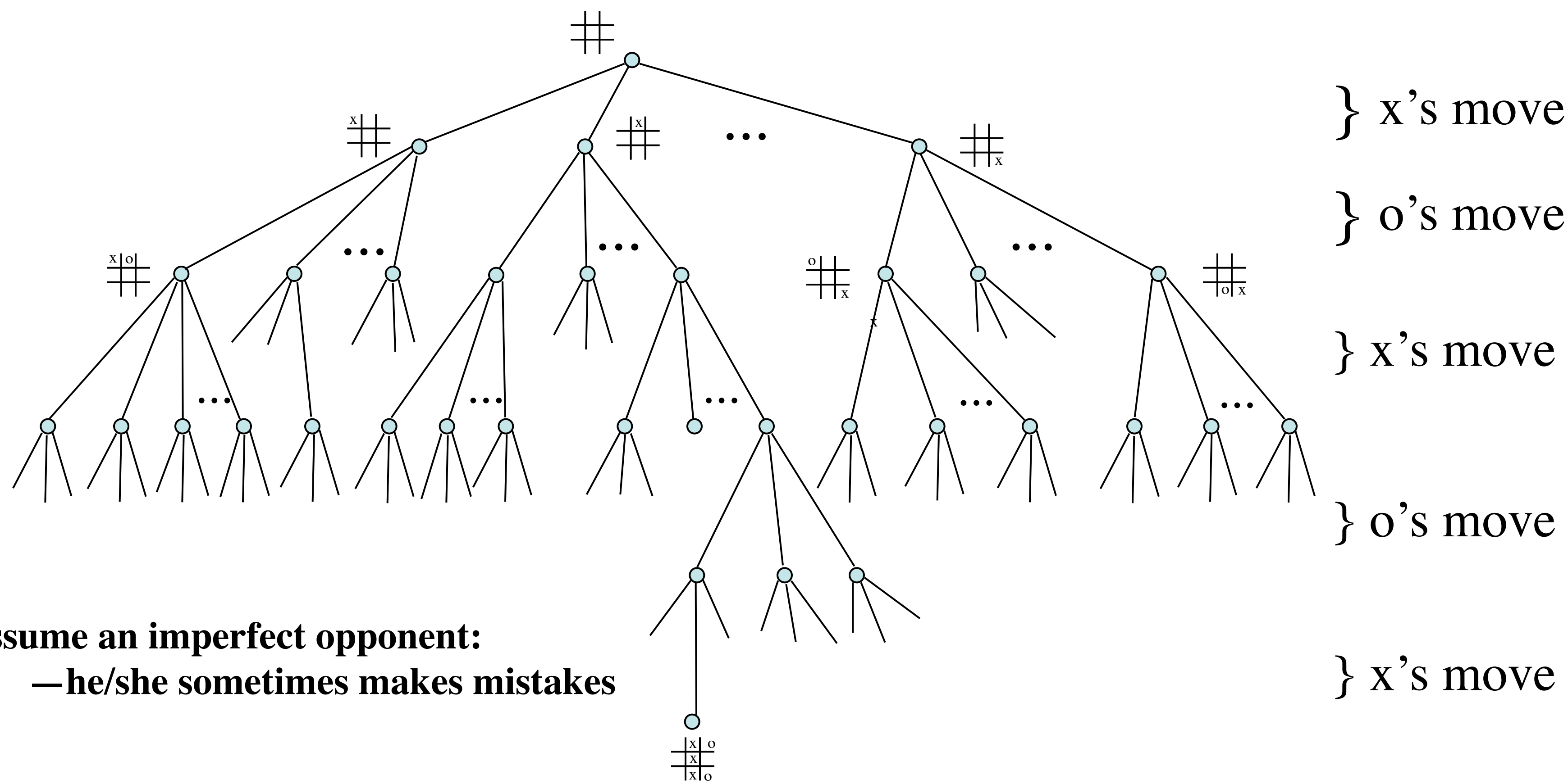
MENACE (Michie, 1961)

"Matchbox Educable Noughts and Crosses Engine"



Tic-Tac-Toe



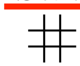
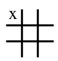
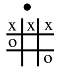
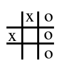
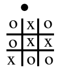


Assume an imperfect opponent:

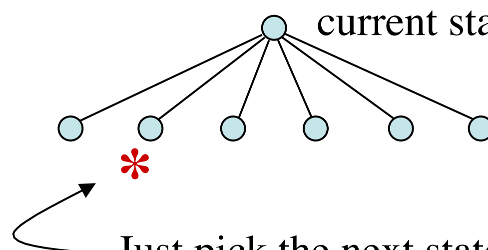
—he/she sometimes makes mistakes

Learning an evaluation function

1. Make a table with one entry per state:

State	$V(s)$ – estimated probability of winning: ***value***	
	.5	?
	.5	?
⋮	⋮	
	1	win
⋮	⋮	
	0	loss
⋮	⋮	
	0	draw

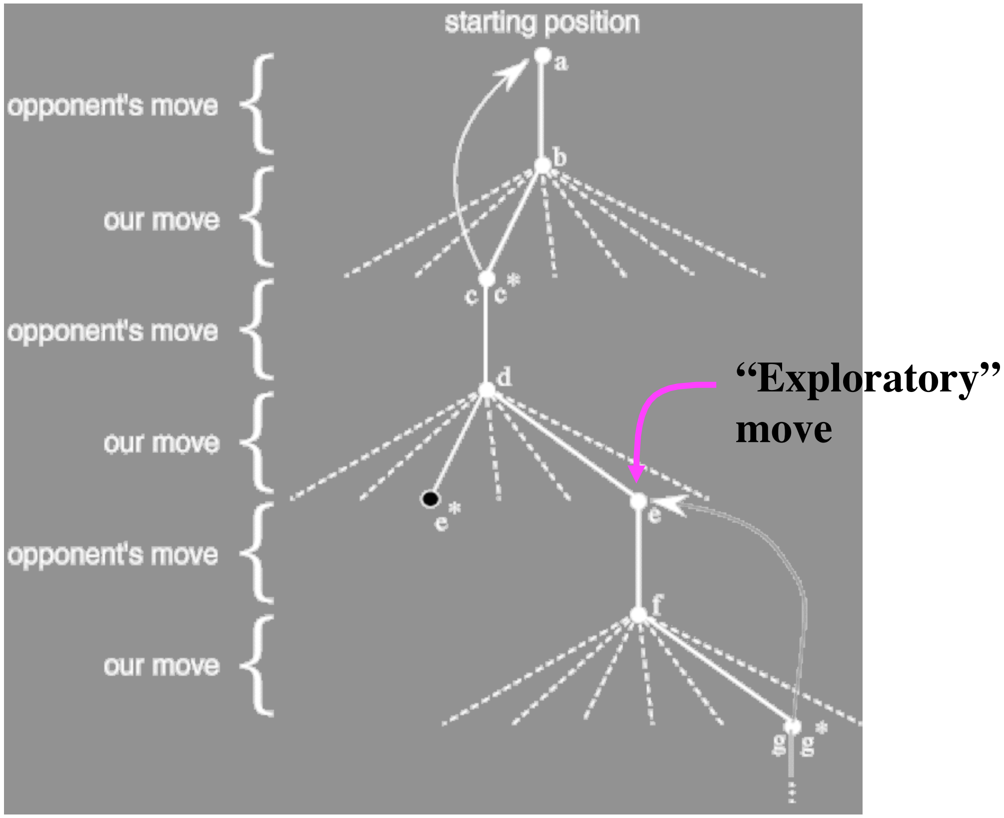
2. Now play lots of games. To pick our moves, look ahead one step:



Just pick the next state with the highest estimated prob. of winning — the largest $V(s)$; a ***greedy*** move.

But 10% of the time pick a move at random; an ***exploratory move***.

Backups



x – the state before our greedy move

y – the state after our greedy move

We increment each $V(x)$ toward $V(y)$ – a ***backup*** :

$$V(x) \leftarrow V(x) + \alpha\,[V(y) - V(x)]$$

α: a small positive fraction, the ***step–size parameter***

A Temporal-Difference (TD) method