Lecture 7: Adversarial Search

Dr John Levine

CS310 Foundations of Artificial Intelligence
February 9th 2016

Learning Outcomes for Today

After today's lecture, you will know:

- How two-player zero-sum games are an extension of the single agent problems we've examined so far
- What a game tree looks like
- How we can solve a game tree by assuming that the opponent plays rationally
- How the Minimax algorithm works

Which Problems can we Solve?

The task environments which are suitable for the search algorithms we've looked at so far are:

- Fully observable (the easy option)
- Deterministic (the easy option)
- Sequential (the normal option)
- Static (the easy option)
- Discrete (the easy option)
- Single agent (the easy option)

Games

- To play a game, we need to relax the assumption that only one agent can change the state of the world.
- Game theory: any environment with multiple agents in it can be regarded as a game.
- In AI, games are usually what game theorists would call deterministic, turn-taking, two-player, zero-sum games of perfect information.
- Examples: chess, checkers, Connect 4, shogi, Othello, go, tic-tac-toe, dots and boxes, …

Features of these Games

- Fully observable: game state is visible to both players
- Deterministic: no element of chance
- Sequential: action taken now affects future choices
- Static: the world doesn't change during deliberation
- Discrete: the game state can be represented exactly using a finite representation
- Multi agent: in the search, we have to allow for the fact that our opponent can also affect the game state when it is their turn, and will be planning against us

Representing a Game

We need:

- An initial state, including who plays first
- A successors (state) function, like next-states (state)
- A terminal test which determines whether a given state is a end state of the game (i.e. the game is over)
- A utility function – for terminal states only, this is the reward each player gets (e.g. +1 for win, −1 for lose)
- In a zero-sum game, the utilities of the end states sum to the same amount for the two players

The Two Players: Max and Min

- Let's call the two players Max and Min: Max goes first
- Max's task is to maximise Max's reward
- Min's task is to minimise Max's reward (and therefore maximise Min's reward)
- Let's say Max can take actions *a*, *b* or *c* – which one will give Max the best reward when the game ends?
- To answer that question, we need to explore the game tree to a *sufficient depth*, and *assume that Min plays optimally* to minimise the reward that Max gets

University of Strathclyde Science

An Example

- Four coins in a row, each player can pick up one coin or two coins.
- The player who picks up the last coin wins.
- Max plays first. What move should Max make?

The Minimax Value

- The minimax value of a node is the utility of the node if the node is a terminal
- If the node is a non-terminal Max node, the minimax value of the node is the maximum of the minimax values of all of the node's successors
- If the node is a non-terminal Min node, the minimax value of the node is the minimum of the minimax values of all of the node's successors
- Recursive definition: results in a depth-first traversal of the game tree

University of Strathclyde Science

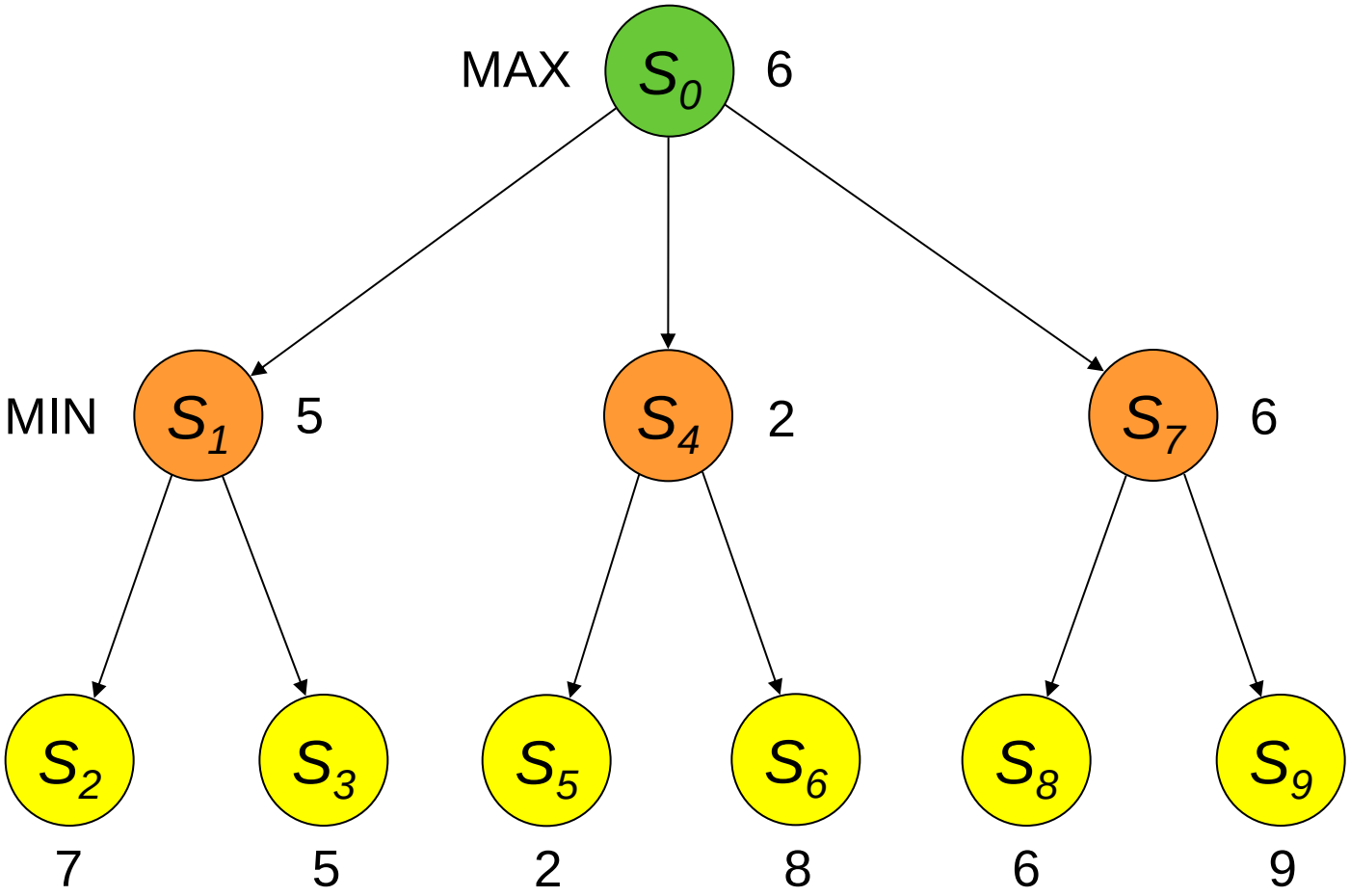
The Minimax Algorithm

```
Max-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← MinimalGameValue
    for s in Successors(state) do
        v ← Max(v, MinValue(s))
    return v

Min-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← MaximalGameValue
    for s in Successors(state) do
        v ← Min(v, Max-Value(s))
    return v
```

University of Strathclyde Science

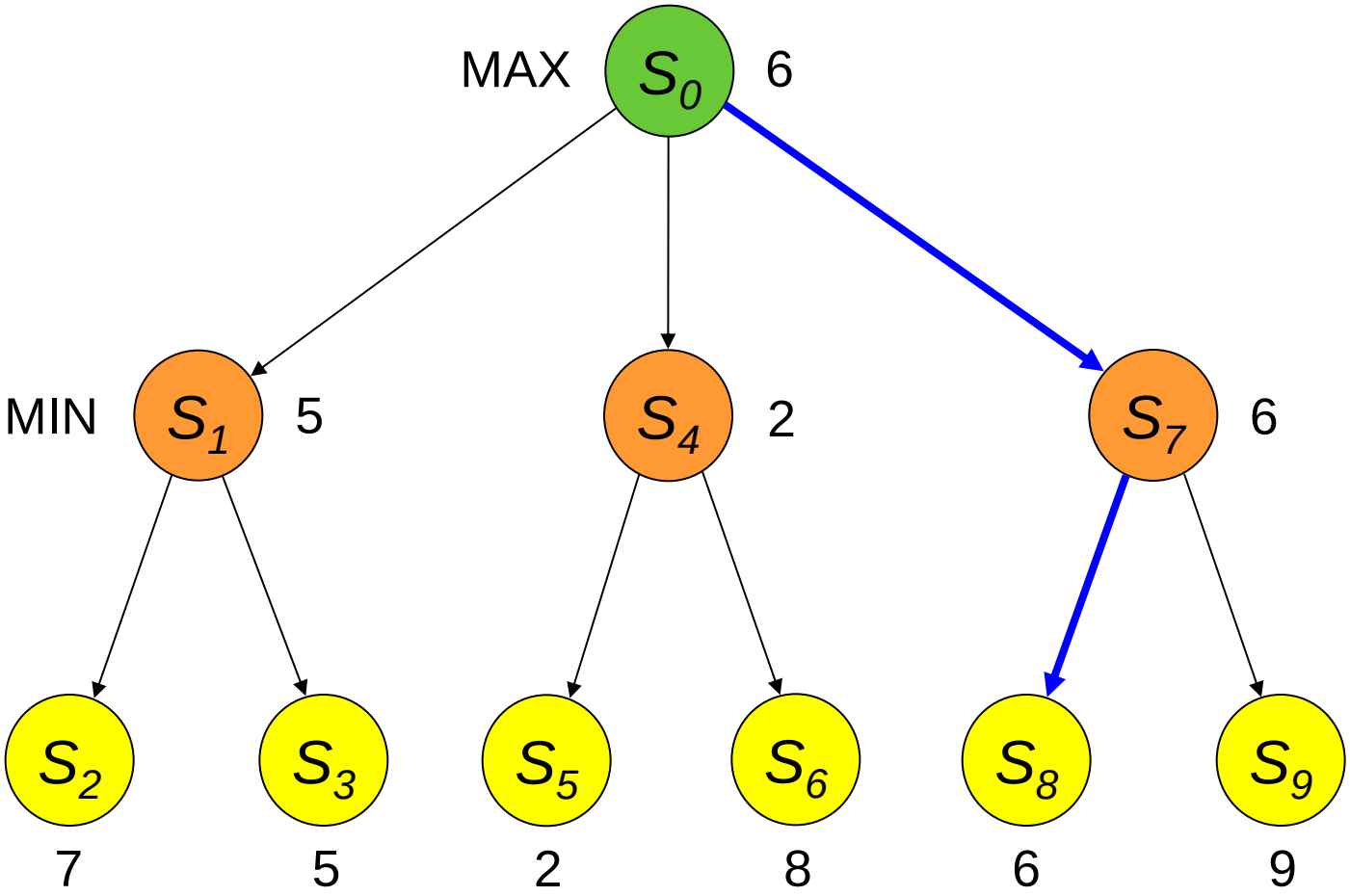
The Minimax Algorithm



CIS COMPUTER and INFORMATION SCIENCES www.cis.strath.ac.uk

University of Strathclyde Science

The Minimax Algorithm



CIS COMPUTER and INFORMATION SCIENCES www.cis.strath.ac.uk

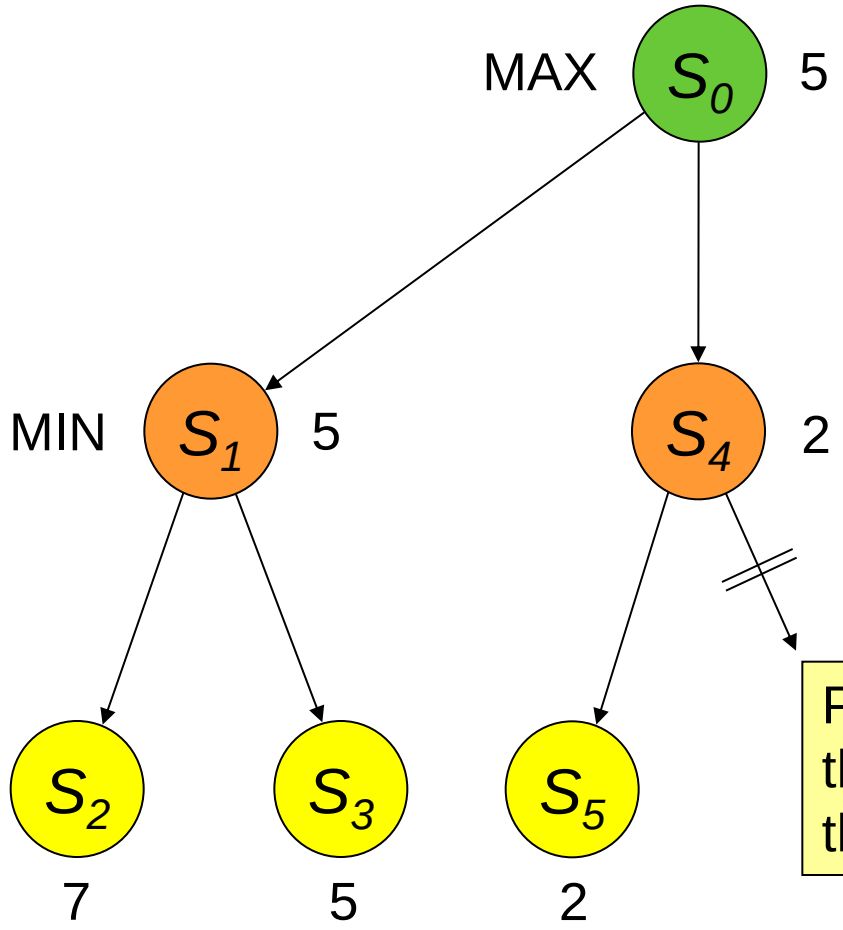
University of Strathclyde Science

Pruning the Search

- Minimax is an exhaustive search algorithm, so it is exponential in the number of moves, i.e. $O(b^m)$
- This is, to say the least, not desirable
- We can only apply full blown minimax to very small games, or games which are close to a terminal state
- However, you may have noticed in the previous example that we don't have to explore the entire tree in order to find the optimal move…

University of Strathclyde Science

Pruning the Search



Prune further search along this branch and return 2 as the minimax value of S_4

Alpha-Beta Pruning

- Alpha-Beta pruning is a method for ignoring branches of the search tree, while still finding the optimal move
- Allows us to find the optimal move much more quickly
- More on this in next Thursday's lecture…