

CS4618 Artificial Intelligence I

Today: Adversarial Search (\cong Games)

Thomas Jansen

October 12th

Plans for Today

- 1 Optimal Game Play
Alpha-Beta-Pruning And Beyond
- 2 Real-Time Game Play
Evaluation and Cutting Off
Forward Pruning and Lookup
- 3 Other Games
Stochastic Games
Partially Observable Games
- 4 Summary
Summary & Take Home Message



Optimal Game Play

Remember

Optimal Game Play

Remember

- two-player zero-sum games

Optimal Game Play

Remember

- two-player zero-sum games
- minimax algorithm to compute optimal moves
 - go over all actions
 - recursively compute values for resulting nodes
 - maximise or minimise (as required by current player) over these values

Optimal Game Play

Remember

- two-player zero-sum games
- minimax algorithm to compute optimal moves
 - go over all actions
 - recursively compute values for resulting nodes
 - maximise or minimise (as required by current player) over these values
- alpha-beta pruning to improve over minimax search without sacrificing optimality
 - keep bounds α and β of what Max and Min can achieve
 - prune subtrees where nodes have values above/below these values

Optimal Game Play

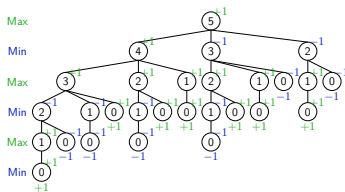
Remember

- two-player zero-sum games
- minimax algorithm to compute optimal moves
 - go over all actions
 - recursively compute values for resulting nodes
 - maximise or minimise (as required by current player) over these values
- alpha-beta pruning to improve over minimax search without sacrificing optimality
 - keep bounds α and β of what Max and Min can achieve
 - prune subtrees where nodes have values above/below these values

Now one further improvement
without sacrificing optimality

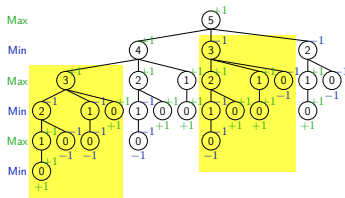
Transposition Table

Remember



Transposition Table

Remember

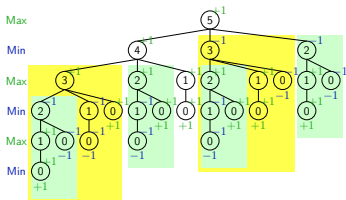


Observation

identical sub-trees explored multiple times

Transposition Table

Remember

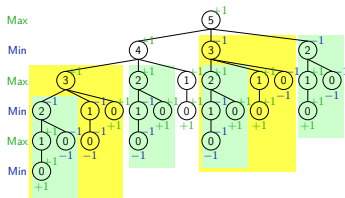


Observation

identical sub-trees explored multiple times

Transposition Table

Remember



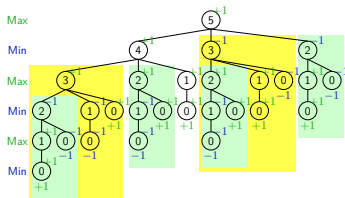
Observation identical sub-trees explored multiple times

Improvement store **states** in hash table
 check before exploring
transposition table
 ('transposing' moves leading to identical states)

Problem size of transposition table
 heuristics for deciding about which states to store

Transposition Table

Remember



Observation identical sub-trees explored multiple times

Improvement store **states** in hash table
 check before exploring
transposition table
 ('transposing' moves leading to identical states)

Problem size of transposition table
 heuristics for deciding about which states to store

Remark minimax search ↔ graph search
 transposition table ↔ explored queue

Real-Time Game Play

Real-Time Game Play

Sad Fact alpha-beta pruning still **much too slow**
for realistic games

Real-Time Game Play

Sad Fact alpha-beta pruning still **much too slow**
for realistic games

Idea make search practical by introducing
cutoff-test to decide when to stop search at a non-terminal

Real-Time Game Play

Sad Fact alpha-beta pruning still **much too slow**
for realistic games

Idea make search practical by introducing
cutoff-test to decide when to stop search at a non-terminal
eval assigns heuristic value to non-terminals
(effectively turning them into terminals)

Real-Time Game Play

Sad Fact alpha-beta pruning still **much too slow**
for realistic games

Idea make search practical by introducing
cutoff-test to decide when to stop search at a non-terminal
eval assigns heuristic value to non-terminals
(effectively turning them into terminals)

Definition for an arbitrary state s

$$h\text{-minimax}(s, d) = \begin{cases} \text{eval}(s) & \text{if cutoff-test}(s, d) \\ \max_{a \in \text{actions}(s)} h\text{-minimax}(\text{result}(s, a), d + 1) & \text{if player}(s) = \text{Max} \\ \min_{a \in \text{actions}(s)} h\text{-minimax}(\text{result}(s, a), d + 1) & \text{if player}(s) = \text{Min} \end{cases}$$

On Evaluation Functions

On Evaluation Functions

Facts

- evaluation functions are **heuristics**

On Evaluation Functions

Facts

- evaluation functions are **heuristics**
- evaluation functions are **crucial**

On Evaluation Functions

Facts

- evaluation functions are **heuristics**
- evaluation functions are **crucial**

Guidelines

- respect ordering for terminals

On Evaluation Functions

Facts

- evaluation functions are **heuristics**
- evaluation functions are **crucial**

Guidelines

- respect ordering for terminals
- be strongly correlated with 'chances' of winning

On Evaluation Functions

Facts

- evaluation functions are **heuristics**
- evaluation functions are **crucial**

Guidelines

- respect ordering for terminals
- be strongly correlated with 'chances' of winning
- be not too computationally costly

On Evaluation Functions

Facts

- evaluation functions are **heuristics**
- evaluation functions are **crucial**

Guidelines

- respect ordering for terminals
- be strongly correlated with ‘chances’ of winning
- be not too computationally costly

Observation **trade-off** for evaluation functions
computational complexity vs. quality

Example: Evaluation Functions for Chess

Example: Evaluation Functions for Chess

- expected values

Example: Evaluation Functions for Chess

- expected values
 - classify states into categories

Example: Evaluation Functions for Chess

- expected values
 - classify states into categories
(e. g., two pawns vs. one pawns)

Example: Evaluation Functions for Chess

- expected values
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)
 - add values as result of evaluation

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)
 - add values as result of evaluation ($\hat{=}$ weighted linear function)

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)
 - add values as result of evaluation ($\hat{=}$ weighted linear function)
 - **problem** often too simple and inaccurate (lacking incorporation of structural properties)

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)
 - add values as result of evaluation ($\hat{=}$ weighted linear function)
 - **problem** often too simple and inaccurate (lacking incorporation of structural properties)
- **non-linear weighting**

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)
 - add values as result of evaluation ($\hat{=}$ weighted linear function)
 - **problem** often too simple and inaccurate (lacking incorporation of structural properties)
- **non-linear weighting**
 - use other factors to change values, incorporate structure

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)
 - add values as result of evaluation ($\hat{=}$ weighted linear function)
 - **problem** often too simple and inaccurate (lacking incorporation of structural properties)
- **non-linear weighting**
 - use other factors to change values, incorporate structure
(e. g., number of pieces in total, number of moves played, ...)

Example: Evaluation Functions for Chess

- **expected values**
 - classify states into categories
(e. g., two pawns vs. one pawns)
 - obtain statistical evidence about outcomes
(e. g., 72% wins (+1), 8% draw (+1/2), 20% loss (-1))
 - use expected value as result of evaluation
(e. g., $.72 \cdot 1 + .08 \cdot (1/2) + .2 \cdot 0 = .76$)
 - **problem** (too) many categories, difficult to get sufficient data
- **material values**
 - assign values to pieces
(e. g., pawn 1, knight 3, bishop 3, rook 5, queen 9)
 - add values as result of evaluation ($\hat{=}$ weighted linear function)
 - **problem** often too simple and inaccurate (lacking incorporation of structural properties)
- **non-linear weighting**
 - use other factors to change values, incorporate structure
(e. g., number of pieces in total, number of moves played, ...)
 - **problem** concrete values difficult to obtain

On Cutting Off

Goal implement $\text{cutoff-test}(s, d)$ for state s and depth d

On Cutting Off

Goal implement $\text{cutoff-test}(s, d)$ for state s and depth d

Simple Idea fixed depth

On Cutting Off

Goal implement $\text{cutoff-test}(s, d)$ for state s and depth d

Simple Idea fixed depth
return true of $d \geq d_{\max}$ or terminal reached

On Cutting Off

Goal implement $\text{cutoff-test}(s, d)$ for state s and depth d

Simple Idea fixed depth
 return true of $d \geq d_{\max}$ or terminal reached

Slight Improvement iterative deepening search

On Cutting Off

Goal implement cutoff-test(s, d) for state s and depth d

Simple Idea fixed depth
return true of $d \geq d_{\max}$ or terminal reached

Slight Improvement iterative deepening search

Problems

- non-quiet positions ($\hat{=}$ extreme changes in evaluation after one move)

On Cutting Off

Goal implement cutoff-test(s, d) for state s and depth d

Simple Idea fixed depth
return true of $d \geq d_{\max}$ or terminal reached

Slight Improvement iterative deepening search

Problems

- non-quiet positions ($\hat{=}$ extreme changes in evaluation after one move)
solution attempt do not include these in evaluation (is called quiescence search)
- horizon effect (delay an ultimately unavoidable bad event thus making bad moves look good)

On Cutting Off

Goal implement cutoff-test(s, d) for state s and depth d

Simple Idea fixed depth
return true of $d \geq d_{\max}$ or terminal reached

Slight Improvement iterative deepening search

Problems

- non-quietest positions ($\hat{=}$ extreme changes in evaluation after one move)
solution attempt do not include these in evaluation (is called quiescence search)
- horizon effect (delay an ultimately unavoidable bad event thus making bad moves look good)
solution attempt consider very promising move in greater depth (is called singular extension)

Forward Pruning

Forward Pruning

- Idea** for speeding-up search
- forward pruning** $\hat{=}$ considering only a limited number of moves already in the beginning

Forward Pruning

Idea for speeding-up search

forward pruning $\hat{=}$ considering only a limited number of moves already in the beginning

Implementation **beam search**

consider only k most promising moves (parameter k)

Forward Pruning

Idea for speeding-up search

forward pruning $\hat{=}$ considering only a limited number of moves already in the beginning

Implementation **beam search**

consider only k most promising moves (parameter k)

Problem dangerous as best move may be difficult to identify (and thus missed)

Forward Pruning

Idea for speeding-up search

forward pruning $\hat{=}$ considering only a limited number of moves already in the beginning

Implementation **beam search**

consider only k most promising moves (parameter k)

Problem dangerous as best move may be difficult to identify (and thus missed)

Implementation **probabilistic cut**

extend alpha-beta pruning by

estimating $[\alpha'; \beta']$ -windows

and cutting off like in alpha-beta pruning

Lookup

Idea avoid search completely
by **looking up** optimal solutions in a dictionary
when there are only sufficiently few states

Lookup

Idea avoid search completely
by **looking up** optimal solutions in a dictionary
when there are only sufficiently few states

When are there only 'sufficiently few' states?

Lookup

Idea avoid search completely
by **looking up** optimal solutions in a dictionary
when there are only sufficiently few states

When are there only 'sufficiently few' states?

Observation at the beginning and the end

Lookup

Idea avoid search completely
by **looking up** optimal solutions in a dictionary
when there are only sufficiently few states

When are there only 'sufficiently few' states?

Observation at the beginning and the end

Opening choose 'optimal' moves based on
expert domain knowledge

Lookup

Idea avoid search completely
by **looking up** optimal solutions in a dictionary
when there are only sufficiently few states

When are there only 'sufficiently few' states?

Observation at the beginning and the end

Opening choose 'optimal' moves based on
expert domain knowledge
experience/statistics

Lookup

Idea avoid search completely
by **looking up** optimal solutions in a dictionary
when there are only sufficiently few states

When are there only 'sufficiently few' states?

Observation at the beginning and the end

Opening choose 'optimal' moves based on
expert domain knowledge
experience/statistics

Endgames compute 'optimal' moves using
retrograde minimax search (considering moves in reverse)

Lookup

Idea avoid search completely
by **looking up** optimal solutions in a dictionary
when there are only sufficiently few states

When are there only 'sufficiently few' states?

Observation at the beginning and the end

Opening choose 'optimal' moves based on
expert domain knowledge
experience/statistics

Endgames compute 'optimal' moves using
retrograde minimax search (considering moves in reverse)
Note game tree size limited
↪ exhaustive search feasible

Stochastic Games

Fact some games include randomness

Stochastic Games

Fact some games include randomness
example backgammon

Stochastic Games

Fact some games include randomness

example backgammon

Observation standard game trees **fail** to work
since choices of players are limited by random events

Stochastic Games

Fact some games include randomness

example backgammon

Observation standard game trees **fail** to work
since choices of players are limited by random events

Idea introduce **chance nodes**
formally, another player 'Chance' with random choices
using the appropriate probability distribution

Stochastic Games

Fact some games include randomness

example backgammon

Observation standard game trees **fail** to work
since choices of players are limited by random events

Idea introduce **chance nodes**
formally, another player 'Chance' with random choices
using the appropriate probability distribution

Definition for an arbitrary state s

$$\text{exp-MM}(s, d) = \begin{cases} \text{utility}(s) & \text{if terminal-test}(s) \\ \max_{a \in \text{actions}(s)} \text{exp-MM}(\text{result}(s, a)) & \text{if player}(s) = \text{Max} \\ \min_{a \in \text{actions}(s)} \text{exp-MM}(\text{result}(s, a)) & \text{if player}(s) = \text{Min} \\ \sum_{r \in \text{rand. events}(s)} \text{Prob}(r) \text{exp-MM}(\text{result}(s, a)) & \text{if player}(s) = \text{Chance} \end{cases}$$

On Order-Preserving Transformations

On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important

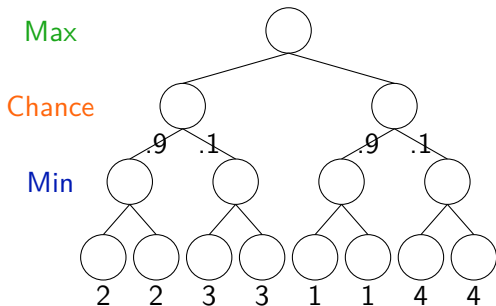
On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

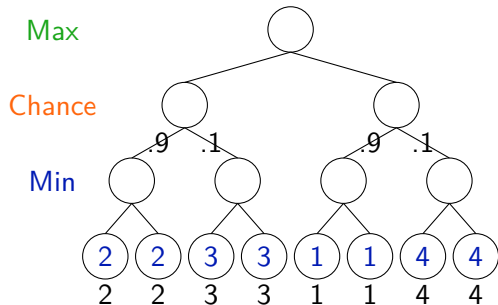
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

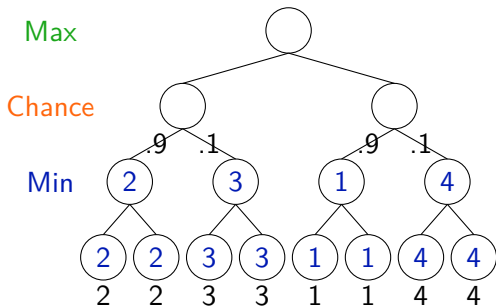
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
 only **ordering** of terminal-values important
 ⇒ order-preserving transformation without effect

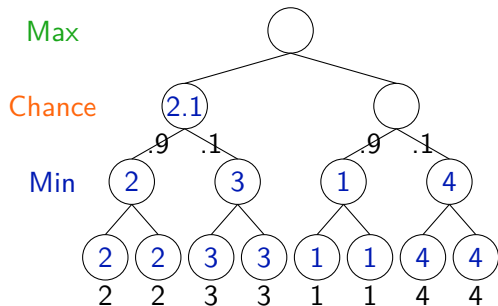
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
 only **ordering** of terminal-values important
 \Rightarrow order-preserving transformation without effect

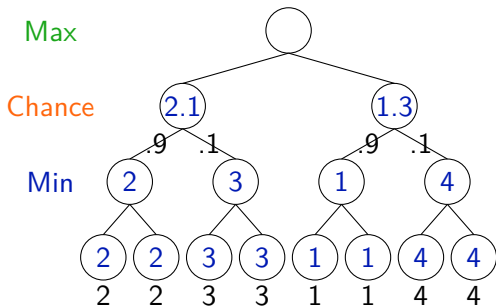
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
 only **ordering** of terminal-values important
 \Rightarrow order-preserving transformation without effect

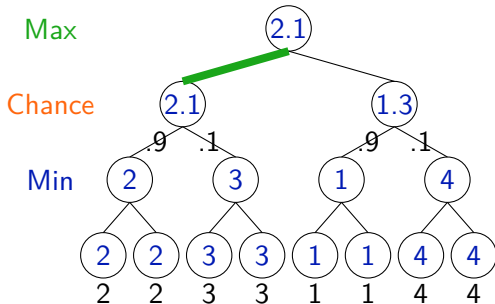
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
 only **ordering** of terminal-values important
 \Rightarrow order-preserving transformation without effect

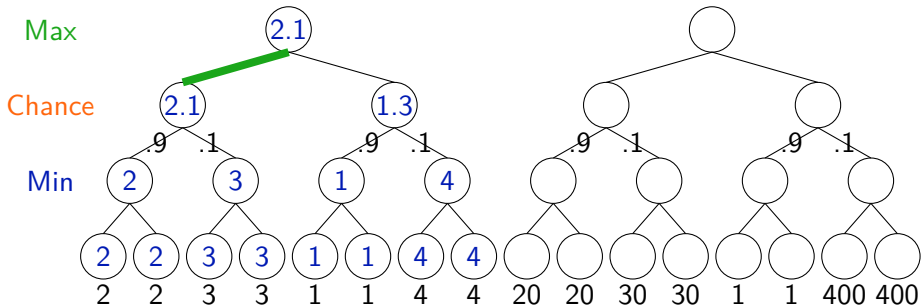
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

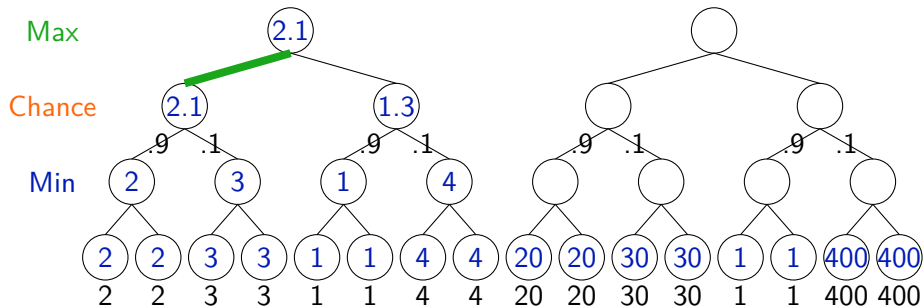
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

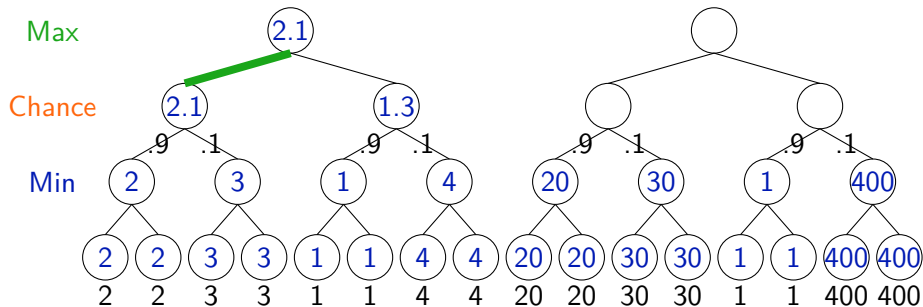
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

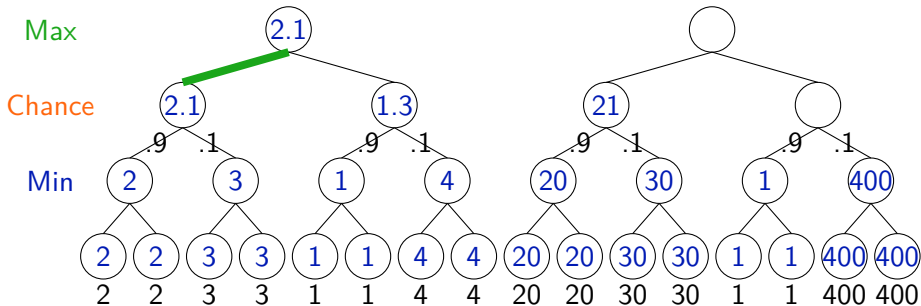
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

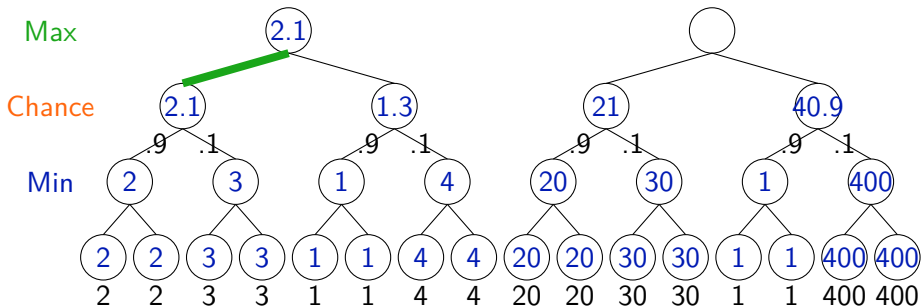
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
 only **ordering** of terminal-values important
 ⇒ order-preserving transformation without effect

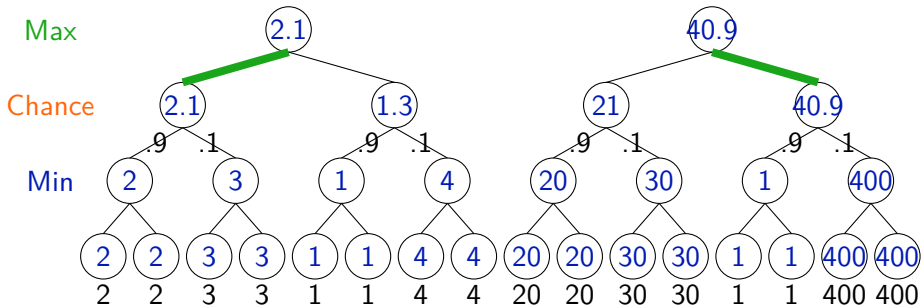
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
only **ordering** of terminal-values important
⇒ order-preserving transformation without effect

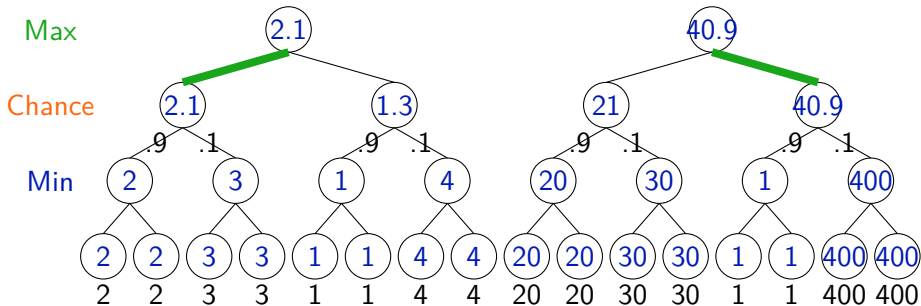
Consider game tree for a tiny stochastic game



On Order-Preserving Transformations

Observation for deterministic games
 only **ordering** of terminal-values important
 \Rightarrow order-preserving transformation without effect

Consider game tree for a tiny stochastic game



Observation optimal move **changed**

On Stochastic Games

On Stochastic Games

Observations and Facts

- special care with evaluation function needed
(due to sensitivity with respect to order-preserving transformations)

On Stochastic Games

Observations and Facts

- special care with evaluation function needed
(due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**

On Stochastic Games

Observations and Facts

- special care with evaluation function needed
(due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**
 - **needs** upper and lower bounds on utility values

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**
 - **needs** upper and lower bounds on utility values
 - use bounds to bound expected utility values at nodes

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**
 - **needs** upper and lower bounds on utility values
 - use bounds to bound expected utility values at nodes
 - apply alpha-beta pruning with respect to expected utility values

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**
 - **needs** upper and lower bounds on utility values
 - use bounds to bound expected utility values at nodes
 - apply alpha-beta pruning with respect to expected utility values
- Monte Carlo simulation helps

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**
 - **needs** upper and lower bounds on utility values
 - use bounds to bound expected utility values at nodes
 - apply alpha-beta pruning with respect to expected utility values
- Monte Carlo simulation helps
 - replace chance nodes by actual random decisions

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**
 - **needs** upper and lower bounds on utility values
 - use bounds to bound expected utility values at nodes
 - apply alpha-beta pruning with respect to expected utility values
- Monte Carlo simulation helps
 - replace chance nodes by actual random decisions
 - apply 'normal' search algorithm (e. g., alpha-beta pruning)

On Stochastic Games

Observations and Facts

- special care with evaluation function needed (due to sensitivity with respect to order-preserving transformations)
- chance nodes **increase game tree size vastly**
- alpha-beta pruning **hardly useful**
- adaptation of alpha-beta pruning **feasible**
 - **needs** upper and lower bounds on utility values
 - use bounds to bound expected utility values at nodes
 - apply alpha-beta pruning with respect to expected utility values
- Monte Carlo simulation helps
 - replace chance nodes by actual random decisions
 - apply 'normal' search algorithm (e. g., alpha-beta pruning)
 - repeat many tries, average results and pick 'average best' move

Partially Observable Games

Partially Observable Games

Fact in some games the player's knowledge about states is restricted
example Kriegsspiel

Partially Observable Games

Fact in some games the player's knowledge about states is restricted
example Kriegsspiel

Kriegsspiel

- partially observable variant of chess

Partially Observable Games

Fact in some games the player's knowledge about states is restricted
example Kriegsspiel

Kriegsspiel

- partially observable variant of chess
- each player can only see the own pieces

Partially Observable Games

Fact in some games the player's knowledge about states is restricted
example Kriegsspiel

Kriegsspiel

- partially observable variant of chess
- each player can only see the own pieces
- a referee announces illegal moves, takes them back and allows for another move

Partially Observable Games

Fact in some games the player's knowledge about states is restricted
example Kriegsspiel

Kriegsspiel

- partially observable variant of chess
- each player can only see the own pieces
- a referee announces illegal moves, takes them back and allows for another move
- a referee announces captures, check (with additional information), and end of game (stalemate, checkmate)

Partially Observable Games

Fact in some games the player's knowledge about states is restricted
example Kriegsspiel

Kriegsspiel

- partially observable variant of chess
- each player can only see the own pieces
- a referee announces illegal moves, takes them back and allows for another move
- a referee announces captures, check (with additional information), and end of game (stalemate, checkmate)

Approach perform **state estimation**
and search to **belief states**

Partially Observable Stochastic Games

Partially Observable Stochastic Games

Fact in some games the player's knowledge about states is restricted and the game includes random elements

Partially Observable Stochastic Games

- Fact** in some games the player's knowledge about states is restricted and the game includes random elements
- example** card games

Partially Observable Stochastic Games

Fact in some games the player's knowledge about states is restricted and the game includes random elements

example card games

Approach assume cards are dealt and treat as simple ($\hat{=}$ fully observable) deterministic game

Partially Observable Stochastic Games

Fact in some games the player's knowledge about states is restricted and the game includes random elements

example card games

Approach assume cards are dealt and treat as simple ($\hat{=}$ fully observable) deterministic game

Problems

- probability space **too large**

Partially Observable Stochastic Games

Fact in some games the player's knowledge about states is restricted and the game includes random elements

example card games

Approach assume cards are dealt and treat as simple ($\hat{=}$ fully observable) deterministic game

Problems

- probability space **too large**
solution attempt apply Monte Carlo simulation

Partially Observable Stochastic Games

Fact in some games the player's knowledge about states is restricted and the game includes random elements

example card games

Approach assume cards are dealt and treat as simple ($\hat{=}$ fully observable) deterministic game

Problems

- probability space **too large**
solution attempt apply Monte Carlo simulation
- assumption of fully observability **unrealistic** making reasonable moves impossible (e. g., information gathering, bluffing)

Partially Observable Stochastic Games

Fact in some games the player's knowledge about states is restricted and the game includes random elements

example card games

Approach assume cards are dealt and treat as simple ($\hat{=}$ fully observable) deterministic game

Problems

- probability space **too large**
solution attempt apply Monte Carlo simulation
- assumption of fully observability **unrealistic** making reasonable moves impossible (e. g., information gathering, bluffing)
solution attempt incorporate belief states

AI for Games – An Overview

AI for Games – An Overview

- Chess
 - routinely outperform best human players (at least since 2008)

AI for Games – An Overview

- Chess
routinely outperform best human players (at least since 2008)
- Checkers
optimal game play feasible (\Rightarrow routinely outperform best human players)

AI for Games – An Overview

- **Chess**
routinely outperform best human players (at least since 2008)
- **Checkers**
optimal game play feasible (\Rightarrow routinely outperform best human players)
- **Othello/Reversi**
routinely outperform best human players (at least since 2000)

AI for Games – An Overview

- **Chess**
routinely outperform best human players (at least since 2008)
- **Checkers**
optimal game play feasible (\Rightarrow routinely outperform best human players)
- **Othello/Reversi**
routinely outperform best human players (at least since 2000)
- **Backgammon**
competitive with top human players (since 1992)

AI for Games – An Overview

- **Chess**
routinely outperform best human players (at least since 2008)
- **Checkers**
optimal game play feasible (\Rightarrow routinely outperform best human players)
- **Othello/Reversi**
routinely outperform best human players (at least since 2000)
- **Backgammon**
competitive with top human players (since 1992)
- **Go**
amateur level (good results on smaller boards)

AI for Games – An Overview

- **Chess**
routinely outperform best human players (at least since 2008)
- **Checkers**
optimal game play feasible (\Rightarrow routinely outperform best human players)
- **Othello/Reversi**
routinely outperform best human players (at least since 2000)
- **Backgammon**
competitive with top human players (since 1992)
- **Go**
amateur level (good results on smaller boards)
- **Bridge**
acceptable level of game play

AI for Games – An Overview

- **Chess**
routinely outperform best human players (at least since 2008)
- **Checkers**
optimal game play feasible (\Rightarrow routinely outperform best human players)
- **Othello/Reversi**
routinely outperform best human players (at least since 2000)
- **Backgammon**
competitive with top human players (since 1992)
- **Go**
amateur level (good results on smaller boards)
- **Bridge**
acceptable level of game play
- **Scrabble**
competitive with top human players (since 2006)

Summary & Take Home Message

Things to remember

Summary & Take Home Message

Things to remember

- transposition tables

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games
- partially observable (stochastic) games

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games
- partially observable (stochastic) games

Take Home Message

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games
- partially observable (stochastic) games

Take Home Message

- Games are a fascinating and motivating area for AI applications.

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games
- partially observable (stochastic) games

Take Home Message

- Games are a fascinating and motivating area for AI applications.
- Many games are a AI success story.

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games
- partially observable (stochastic) games

Take Home Message

- Games are a fascinating and motivating area for AI applications.
- Many games are a AI success story.
- If feasible 'brute force' is better than 'intelligence'.

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games
- partially observable (stochastic) games

Take Home Message

- Games are a fascinating and motivating area for AI applications.
- Many games are a AI success story.
- If feasible 'brute force' is better than 'intelligence'.
- Stochastic games are much more challenging.

Summary & Take Home Message

Things to remember

- transposition tables
- evaluation functions & cutoff, forward pruning, lookup
- stochastic games
- partially observable (stochastic) games

Take Home Message

- Games are a fascinating and motivating area for AI applications.
- Many games are a AI success story.
- If feasible 'brute force' is better than 'intelligence'.
- Stochastic games are much more challenging.
- Partially observable (stochastic) games present difficulties beyond our scope here.