Foundations of Artificial Intelligence

43. Board Games: Stochastic Games

Thomas Keller and Florian Pommerening

University of Basel

May 22, 2023

Board Games: Overview

chapter overview:

Expected Value

Discrete Random Variable

- a random event (like the result of a die roll)
  - is described in terms of a random variable X
  - with associated domain $\mathrm{dom}(X)$
  - and a probability distribution over the domain

Discrete Random Variable

- a random event (like the result of a die roll)
  - is described in terms of a random variable X
  - with associated domain $\mathrm{dom}(X)$
  - and a probability distribution over the domain
- if the number of outcomes of a random event is finite (like here), the random variable is a discrete random variable
- and the probability distribution is given as a probability $P(X = x)$ that the outcome is $x \in \mathrm{dom}(X)$

Discrete Random Variable: Example

informal description:

- you plan to invest in stocks
- your analyst expects these stock price changes:

Bellman Inc.	Howard Corp.
+2 with 30%	+3 with 40%
+1 with 60%	±0 with 10%
±0 with 10%	−1 with 50%

Markov Tec.
+4 with 20%
+2 with 30%
−1 with 50%

Discrete Random Variable: Example

informal description:

- you plan to invest in stocks
- your analyst expects these stock price changes:

Bellman Inc.
+2 with 30%
+1 with 60%
±0 with 10%

Howard Corp.
+3 with 40%
±0 with 10%
−1 with 50%

Markov Tec.
+4 with 20%
+2 with 30%
−1 with 50%

formal model:

- discrete random variables B, H and M
- $\mathrm{dom}(B) = \{2, 1, 0\}$
  $\mathrm{dom}(H) = \{3, 0, -1\}$
  $\mathrm{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$ $\quad P(H = 3) = 0.4$
  $P(B = 1) = 0.6$ $\quad P(H = 0) = 0.1$
  $P(B = 0) = 0.1$ $\quad P(H = -1) = 0.5$

$P(M = 4) = 0.2$
$P(M = 2) = 0.3$
$P(M = -1) = 0.5$

Expected Value

- the expected value $\mathbb{E}[X]$ of a random variable X is a weighted average of its outcomes
- it is computed as the probability-weighted sum of all outcomes $x \in \mathrm{dom}(X)$, i.e.,

$$\mathbb{E}[X] := \sum_{x \in \mathrm{dom}(X)} P(X = x) \cdot x$$

- in stochastic environments, it is rational to deal with uncertainty by optimizing expected values

Expected Value: Example

formal model:

- discrete random variables B, H and M
- $\text{dom}(B) = \{2, 1, 0\}$
  $\text{dom}(H) = \{3, 0, -1\}$
  $\text{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$
  $P(B = 1) = 0.6$
  $P(B = 0) = 0.1$

$$
\begin{aligned}
P(H = 3) &= 0.4 & P(M = 4) &= 0.2 \\
P(H = 0) &= 0.1 & P(M = 2) &= 0.3 \\
P(H = -1) &= 0.5 & P(M = -1) &= 0.5
\end{aligned}
$$

expected gain:

Expected Value: Example

formal model:

- discrete random variables B, H and M
- $\mathrm{dom}(B) = \{2, 1, 0\}$
  $\mathrm{dom}(H) = \{3, 0, -1\}$
  $\mathrm{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$
  $P(B = 1) = 0.6$
  $P(B = 0) = 0.1$

$$\begin{aligned} P(H=3) &= 0.4 \qquad & P(M=4) &= 0.2 \\ P(H=0) &= 0.1 & P(M=2) &= 0.3 \\ P(H=-1) &= 0.5 & P(M=-1) &= 0.5 \end{aligned}$$

expected gain:

$$\begin{aligned} \mathbb{E}[B] &= P(B=2)\cdot 2 + P(B=1)\cdot 1 + P(B=0)\cdot 0 \\ &= 0.3\cdot 2 + 0.6\cdot 1 + 0.1\cdot 0 = 1.2 \end{aligned}$$

$$\begin{aligned} \mathbb{E}[H] &= P(H=3)\cdot 3 + P(H=0)\cdot 0 + P(H=-1)\cdot -1 \\ &= 0.4\cdot 3 + 0.1\cdot 0 + 0.5\cdot -1 = 0.7 \end{aligned}$$

$$\begin{aligned} \mathbb{E}[M] &= P(M=4)\cdot 4 + P(M=2)\cdot 2 + P(M=-1)\cdot -1 \\ &= 0.2\cdot 4 + 0.3\cdot 2 + 0.5\cdot -1 = 0.9 \end{aligned}$$

Expected Value: Example

formal model:

- discrete random variables B, H and M
- $\text{dom}(B) = \{2, 1, 0\}$
  $\text{dom}(H) = \{3, 0, -1\}$
  $\text{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$
  $P(B = 1) = 0.6$
  $P(B = 0) = 0.1$

$$\begin{aligned} P(H = 3) &= 0.4 \qquad & P(M = 4) &= 0.2 \\ P(H = 0) &= 0.1 & P(M = 2) &= 0.3 \\ P(H = -1) &= 0.5 & P(M = -1) &= 0.5 \end{aligned}$$

expected gain:

$$\begin{aligned} \mathbb{E}[B] &= P(B = 2) \cdot 2 + P(B = 1) \cdot 1 + P(B = 0) \cdot 0 \\ &= 0.3 \cdot 2 + 0.6 \cdot 1 + 0.1 \cdot 0 = 1.2 \end{aligned}$$

$$\begin{aligned} \mathbb{E}[H] &= P(H = 3) \cdot 3 + P(H = 0) \cdot 0 + P(H = -1) \cdot -1 \\ &= 0.4 \cdot 3 + 0.1 \cdot 0 + 0.5 \cdot -1 = 0.7 \end{aligned}$$

$$\begin{aligned} \mathbb{E}[M] &= P(M = 4) \cdot 4 + P(M = 2) \cdot 2 + P(M = -1) \cdot -1 \\ &= 0.2 \cdot 4 + 0.3 \cdot 2 + 0.5 \cdot -1 = 0.9 \end{aligned}$$

rational decision:

Expected Value: Example

formal model:

- discrete random variables B, H and M
- $\text{dom}(B) = \{2, 1, 0\}$
  $\text{dom}(H) = \{3, 0, -1\}$
  $\text{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$
  $P(B = 1) = 0.6$
  $P(B = 0) = 0.1$

$$
\begin{aligned}
P(H = 3) &= 0.4 & P(M = 4) &= 0.2 \\
P(H = 0) &= 0.1 & P(M = 2) &= 0.3 \\
P(H = -1) &= 0.5 & P(M = -1) &= 0.5
\end{aligned}
$$

expected gain:

$$
\begin{aligned}
\mathbb{E}[B] &= P(B = 2) \cdot 2 + P(B = 1) \cdot 1 + P(B = 0) \cdot 0 \\
&= 0.3 \cdot 2 + 0.6 \cdot 1 + 0.1 \cdot 0 = 1.2
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}[H] &= P(H = 3) \cdot 3 + P(H = 0) \cdot 0 + P(H = -1) \cdot -1 \\
&= 0.4 \cdot 3 + 0.1 \cdot 0 + 0.5 \cdot -1 = 0.7
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}[M] &= P(M = 4) \cdot 4 + P(M = 2) \cdot 2 + P(M = -1) \cdot -1 \\
&= 0.2 \cdot 4 + 0.3 \cdot 2 + 0.5 \cdot -1 = 0.9
\end{aligned}
$$

rational decision: buy Bellman Inc.

Stochastic Games

Definition

Definition (stochastic game)

A stochastic game is a 7-tuple $\mathcal{S} = \langle S, A, T, s_I, S_\star, \mathit{utility}, \mathit{player}\rangle$ with

- finite set of positions S
- finite set of moves A
- transition function $T : S \times A \times S \mapsto [0,1]$ that is well-defined for $\langle s, a\rangle$ (see below)
- initial position $s_I \in S$
- set of terminal positions $S_\star \subseteq S$
- utility function $\mathit{utility} : S_\star \to \mathbb{R}$
- player function $\mathit{player} : S \setminus S_\star \to \{\mathit{max}, \mathit{min}\}$

A transition function is well-defined for $\langle s, a\rangle$ if $\sum_{s' \in S} T(s, a, s') = 1$ (then a is applicable in s) or $\sum_{s' \in S} T(s, a, s') = 0$.

Reminder: Bounded Inc-and-square Game

informal description:

- Players alternatingly apply a
  - increment-mod10 (*inc*) or
  - square-mod10 (*sqr*) move
- on the natural numbers from 0 to 9
- starting from the number 1;
- if the game reaches the number 6 or 7
- or after a fixed number of moves n
- *max* obtains utility r (*min*: $-r$) where x is the current number.

formal model:

- $S = \{s_0^k, s_1^k, \dots, s_9^k \mid 0 \leq k \leq n\}$
- $A = \{inc, sqr\}$
- for $0 \leq i \leq 9$ and $0 \leq k \leq n$:
  - $\langle s_i^k, inc, s_{(i+1) \bmod 10}^{k+1} \rangle \in T$
  - $\langle s_i^k, sqr, s_{i^2 \bmod 10}^{k+1} \rangle \in T$
- $s_I = s_1^0$
- $S_\star = \{s_6^k, s_7^k \mid 0 \leq k \leq n\} \cup \{s_i^n \mid 0 \leq i \leq 9\}$
- $utility(s_i^k) = i$ for all $s_i^k \in S_\star$
- $player(s_i^k) = max$ if k even and $player(s_i^k) = min$ otherwise

Example: Stochastic Inc-and-square Game

informal description:

- rules like bounded inc-and-square game
- except applying square-mod10 move in state s
- results with probability $\frac{s}{10}$ in $(s^2 \bmod 10)$
- and otherwise in $((2 \cdot s) \bmod 10)$

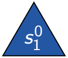
formal model:

- $S = \{s_0^k, s_1^k, \dots, s_9^k \mid 0 \le k \le n\}$
- $A = \{inc, sqr\}$
- for $0 \le i \le 9$ and $0 \le k < n$:
  - $T(s_i^k, inc, s^{k+1}_{(i+1) \bmod 10}) = 1$
  - $T(s_i^k, sqr, s^{k+1}_{i^2 \bmod 10}) = \frac{s}{10}$
  - $T(s_i^k, sqr, s^{k+1}_{(2 \cdot i) \bmod 10}) = \frac{10-s}{10}$

  $(T(s, a, s') = 0$ for all other $\langle s, a, s' \rangle)$
- $s_I = s_1^0$
- $S_\star = \{s_6^k, s_7^k \mid 0 \le k \le n\} \cup \{s_i^n \mid 0 \le i \le 9\}$
- $utility(s_i^k) = i$ for all $s_i^k \in S_\star$
- $player(s_i^k) = max$ if k even and $player(s_i^k) = min$ otherwise

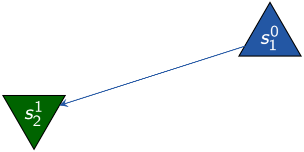
Expectiminimax

Idea and Example



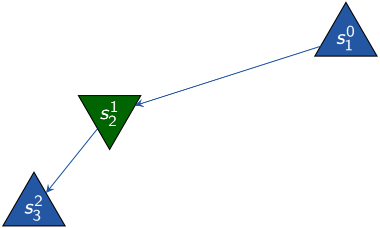
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



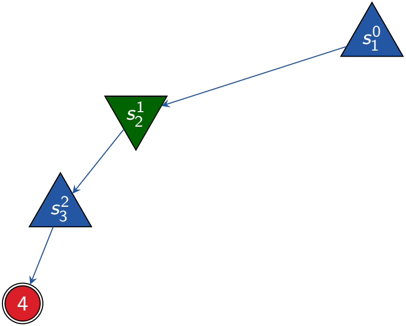
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



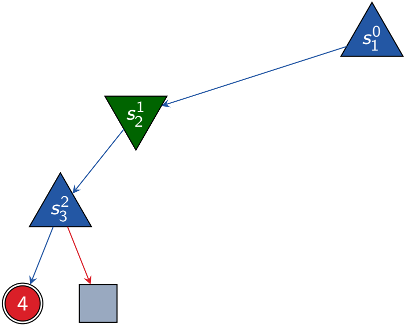
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



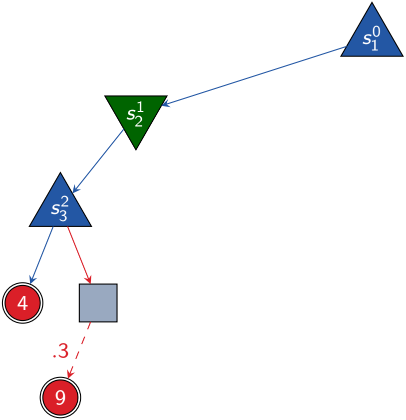
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



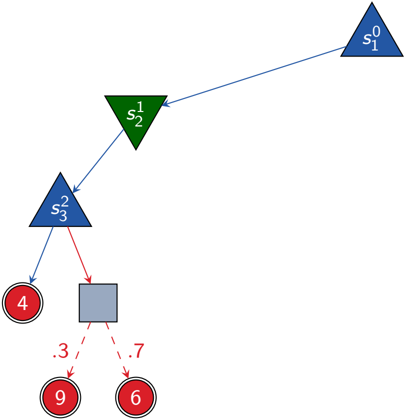
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



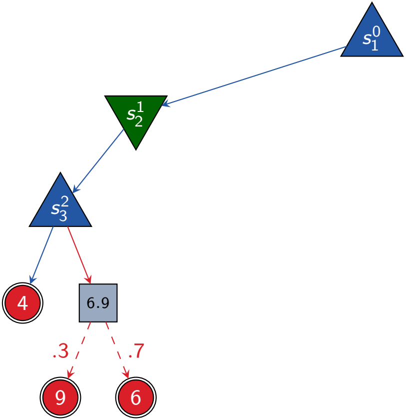
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



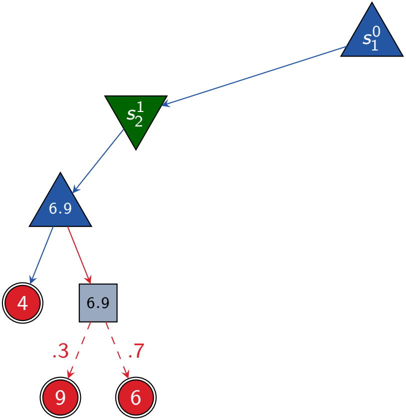
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



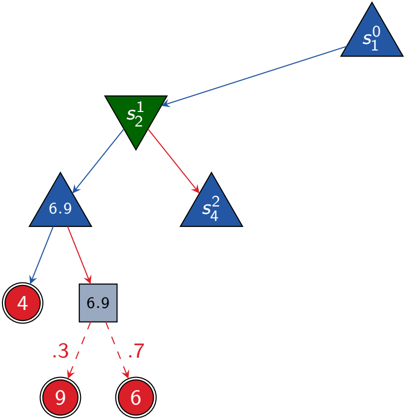
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



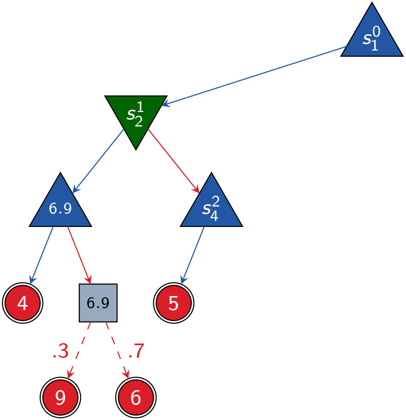
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



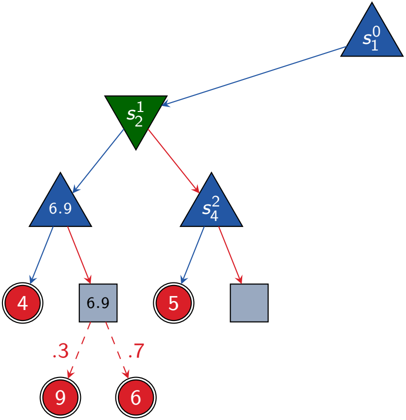
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



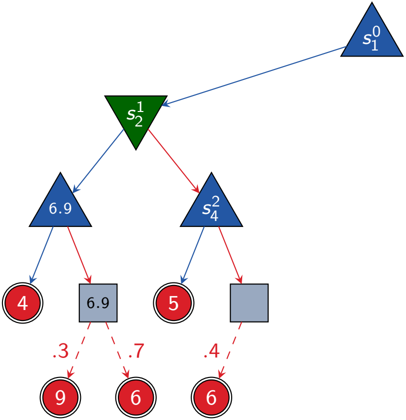
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



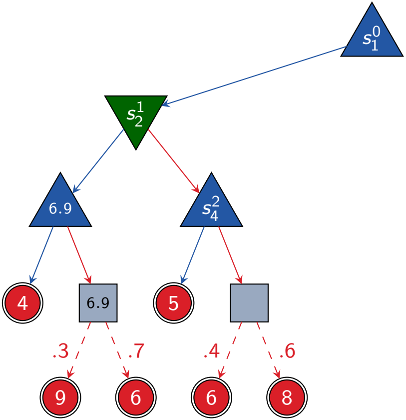
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



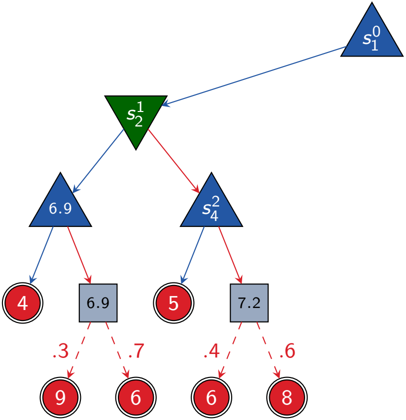
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



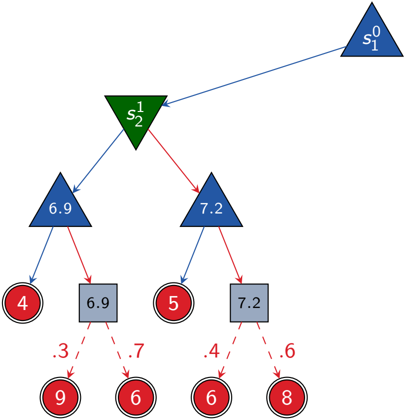
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



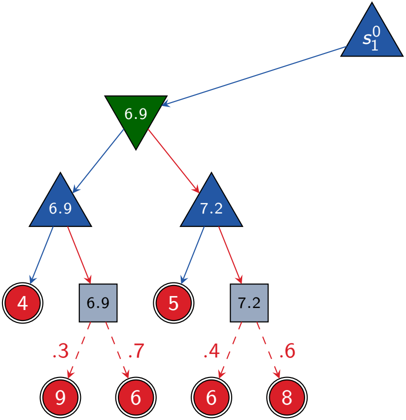
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



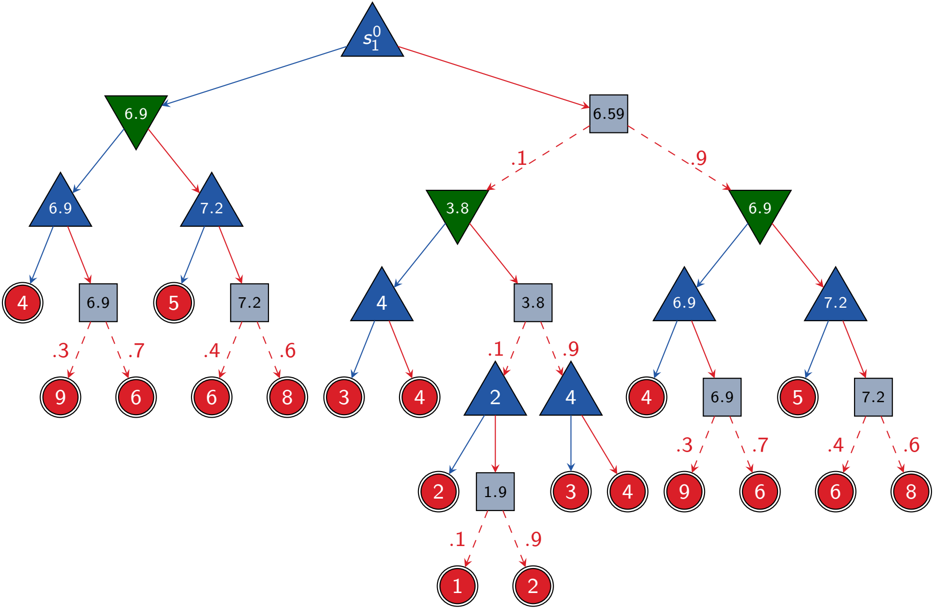
- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example

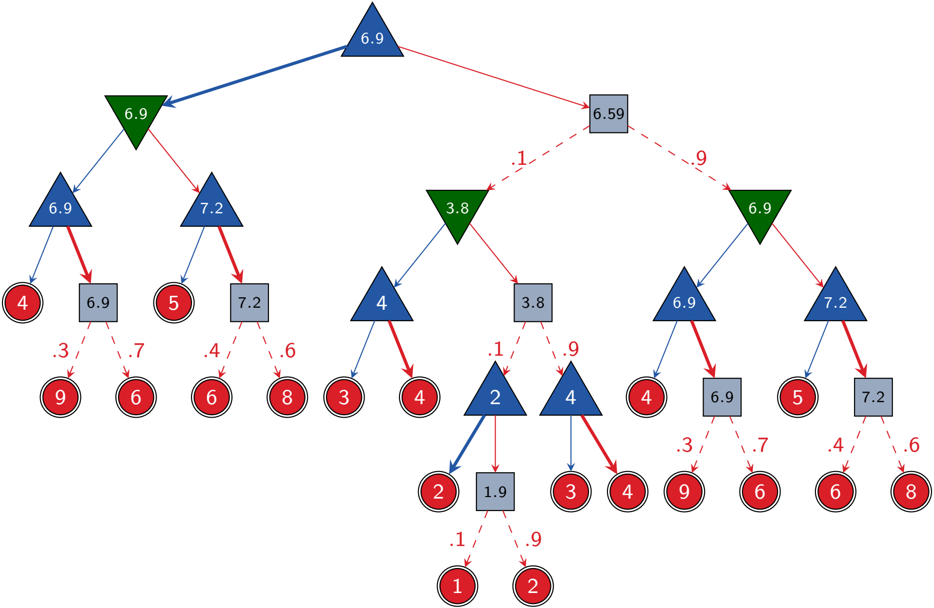


- depth-first search in game tree
- determine utility value of terminal positions with utility function
- compute utility value of inner nodes from below to above through the tree:
  - *min*'s turn: utility value is minimum of utility values of children
  - *max*'s turn: utility value is maximum of utility values of children
  - chance: utility value is probability-weighted sum of utility values of children
- policy: action that maximizes utility value

Idea and Example



Idea and Example



Discussion

- expectiminimax is the simplest (decent) search algorithm for stochastic games
- yields optimal policy* (in the game-theoretic sense, i.e., under the assumption that the opponent plays perfectly)
- *max* obtains at least the state value computed for the root in expectation, no matter how *min* plays
- if *min* plays perfectly, *max* obtains exactly the computed value in expectation
- the reward for *max* that is actually obtained may be a higher or lower than the state value computed for the root, independently of how *min* plays

(*) for finite trees; otherwise things get more complicated

Summary

Summary

- Stochastic games are board games with an additional element of chance.
- Expectiminimax is a minimax variant for stochastic games with identical behavior in *max* and *min* nodes.
- In chance nodes, it propagates the probability-weighted sum of all successors.
- Expectiminimax has similar guarantess as minimax but in expectation.