

Foundations of Artificial Intelligence

G5. Board Games: Stochastic Games

Malte Helmert

University of Basel

May 13, 2026

Board Games: Overview

chapter overview:

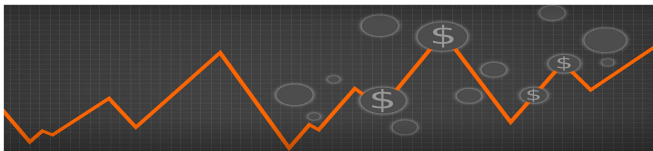
- G1. Introduction and State of the Art
- G2. Formal Definition and Minimax Search
- G3. Evaluation Functions
- G4. Alpha-Beta Search
- G5. Stochastic Games
- G6. Monte-Carlo Tree Search Framework
- G7. Monte-Carlo Tree Search Variants

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** $\text{dom}(X)$
 - 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 $x \in \text{dom}(X)$

Discrete Random Variable: Example

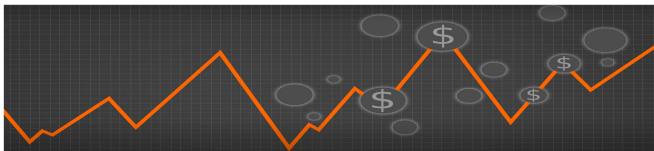


informal description:

- you plan to **invest** in **stocks** and can afford **one share**
- your analyst **expects** these **stock price changes**:

Bellman Inc.	Markov Tec.
+2 with 30%	+4 with 20%
+1 with 60%	+2 with 30%
± 0 with 10%	-1 with 50%

Discrete Random Variable: Example



informal description:

- you plan to **invest** in **stocks** and can afford **one share**
- your analyst **expects** these **stock price changes**:

Bellman Inc.	Markov Tec.
+2 with 30%	+4 with 20%
+1 with 60%	+2 with 30%
± 0 with 10%	-1 with 50%

formal model:

- discrete random variables B and M
- $\text{dom}(B) = \{2, 1, 0\}$
 $\text{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$ $P(M = 4) = 0.2$
 $P(B = 1) = 0.6$ $P(M = 2) = 0.3$
 $P(B = 0) = 0.1$ $P(M = -1) = 0.5$

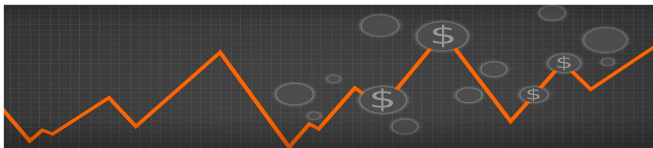
Expected Value

- 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 $x \in \text{dom}(X)$, i.e.,

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

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

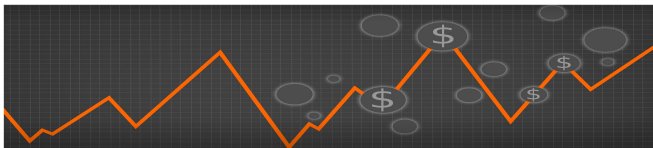
Expected Value: Example



formal model:

- discrete random variables
 B and M
- $\text{dom}(B) = \{2, 1, 0\}$
 $\text{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$ $P(M = 4) = 0.2$
 $P(B = 1) = 0.6$ $P(M = 2) = 0.3$
 $P(B = 0) = 0.1$ $P(M = -1) = 0.5$

Expected Value: Example



formal model:

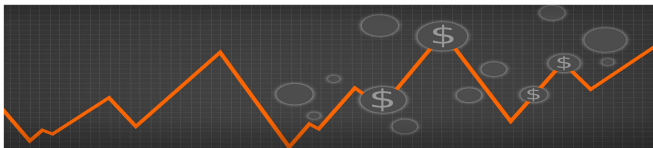
- discrete random variables
 B and M
- $\text{dom}(B) = \{2, 1, 0\}$
 $\text{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$ $P(M = 4) = 0.2$
 $P(B = 1) = 0.6$ $P(M = 2) = 0.3$
 $P(B = 0) = 0.1$ $P(M = -1) = 0.5$

expected gain:

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

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

Expected Value: Example



formal model:

- discrete random variables B and M
- $\text{dom}(B) = \{2, 1, 0\}$
 $\text{dom}(M) = \{4, 2, -1\}$
- $P(B = 2) = 0.3$ $P(M = 4) = 0.2$
 $P(B = 1) = 0.6$ $P(M = 2) = 0.3$
 $P(B = 0) = 0.1$ $P(M = -1) = 0.5$

expected gain:

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

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

rational decision: buy Bellman Inc.

Stochastic Games

Definition

Definition (stochastic game)

A **stochastic game** is a

7-tuple $\mathcal{S} = \langle S, A, T, s_1, S_G, \text{utility}, \text{player} \rangle$ 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** $s_1 \in S$
- set of **terminal positions** $S_G \subseteq S$
- **utility function** $\text{utility} : S_G \rightarrow \mathbb{R}$
- **player function** $\text{player} : S \setminus S_G \rightarrow \{\text{MAX}, \text{MIN}\}$

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 $\sum_{s' \in S} T(s, a, s') = 0$.

Example: Stochastic Inc-and-Square Game

- As an example, we consider a variant of the bounded inc-and-square game from Chapter G2.
- The **sqr** move now acts stochastically:
 - It **squares** the current value $v \pmod{10}$ with probability $\frac{v}{10}$.
 - Otherwise it **doubles** the current value $v \pmod{10}$ (with prob. $1 - \frac{v}{10}$).
- We also reduce the maximum game length to 3 moves (counting both players) to make the example smaller.
- Everything else stays the same.

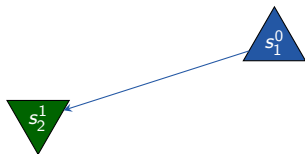
Expectiminimax

Idea and Example



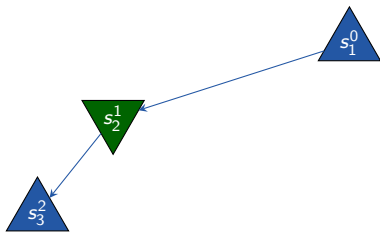
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



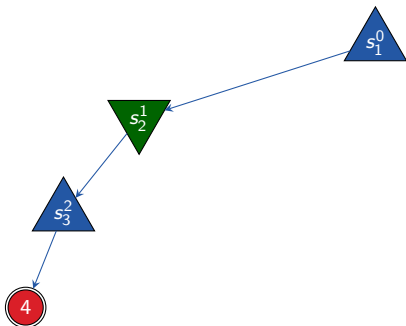
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



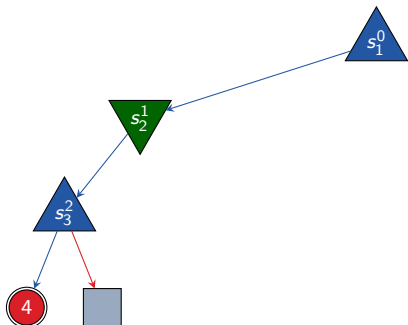
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



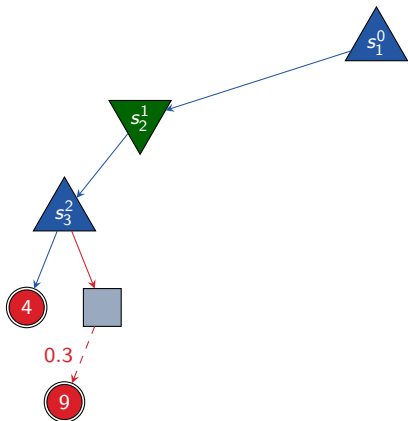
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



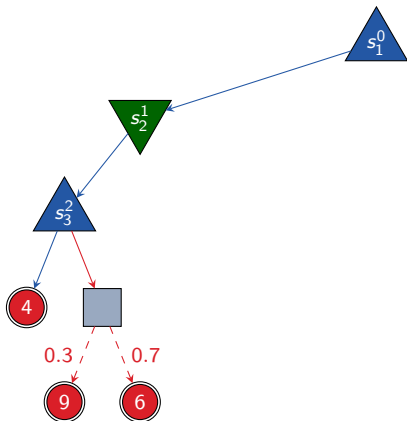
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



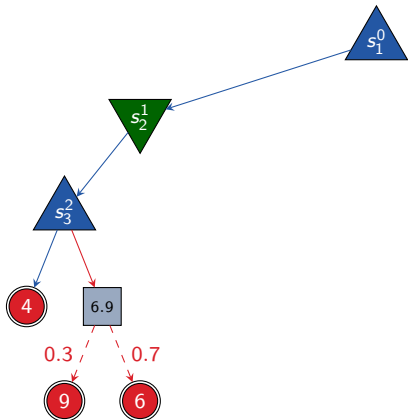
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



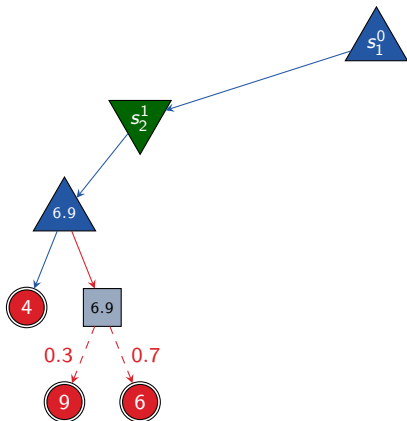
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



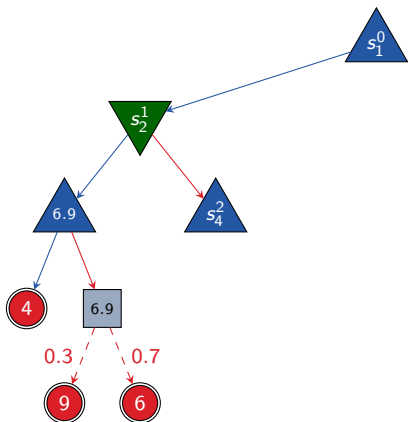
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



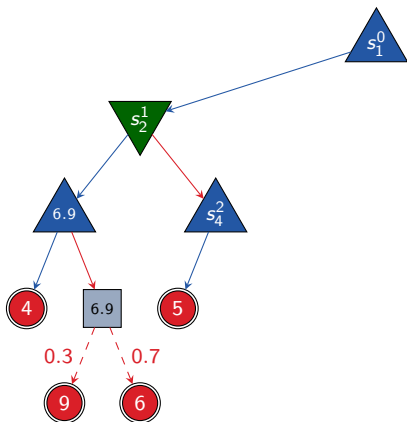
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



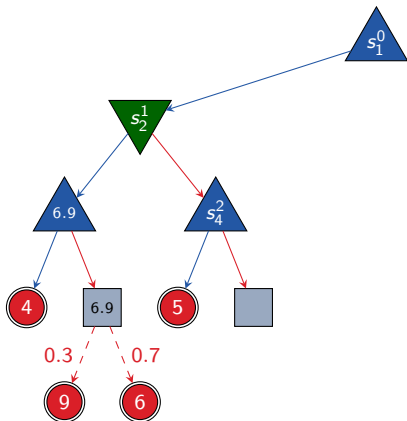
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



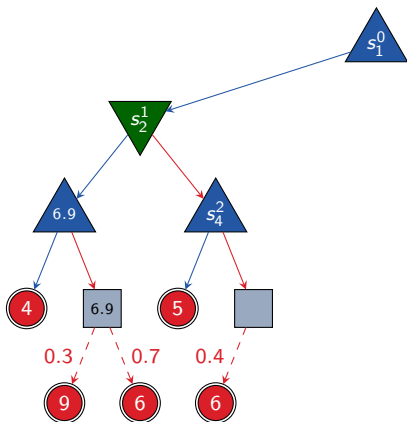
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



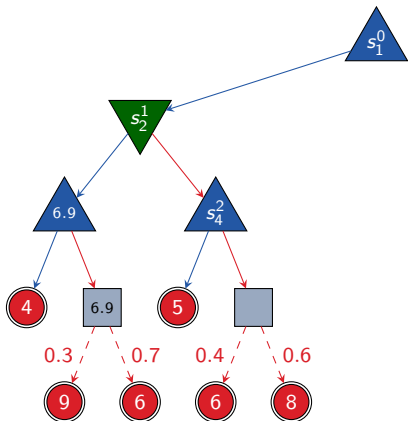
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



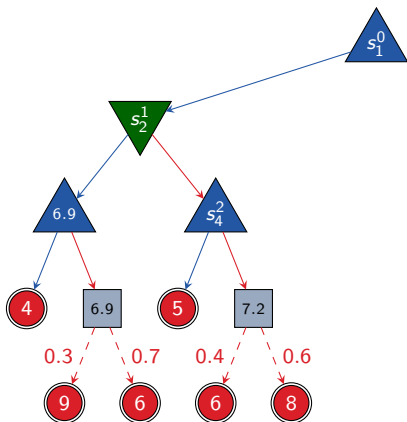
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



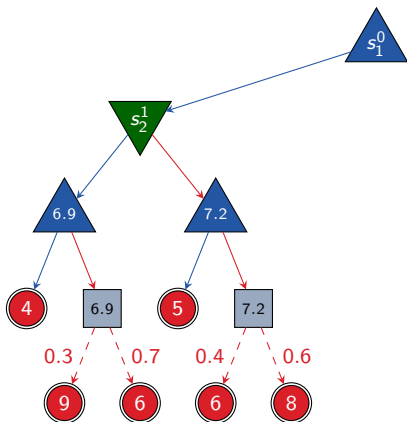
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



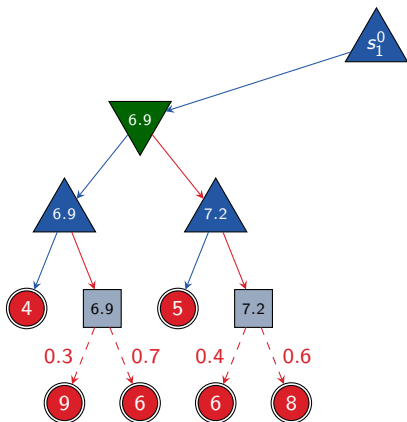
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example



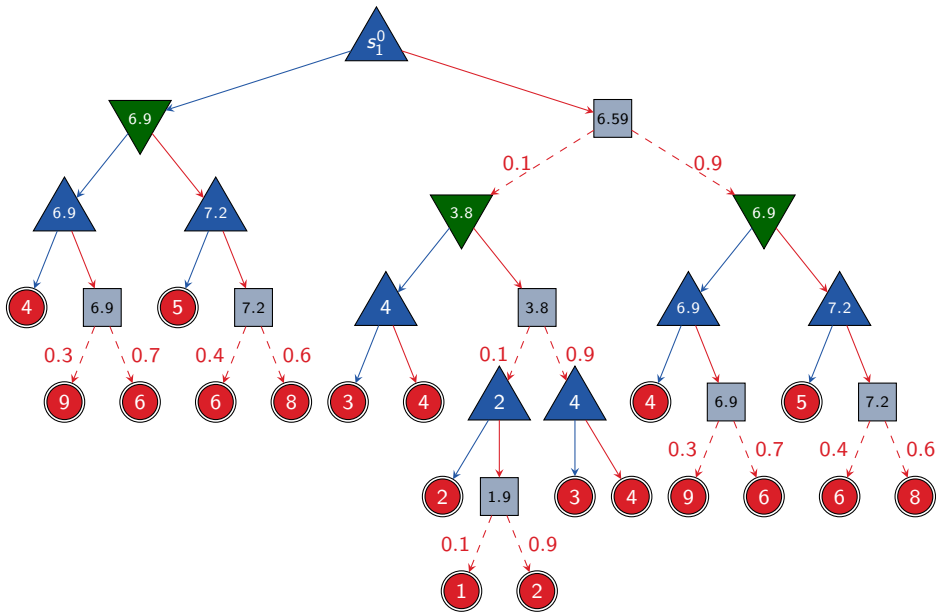
- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

Idea and Example

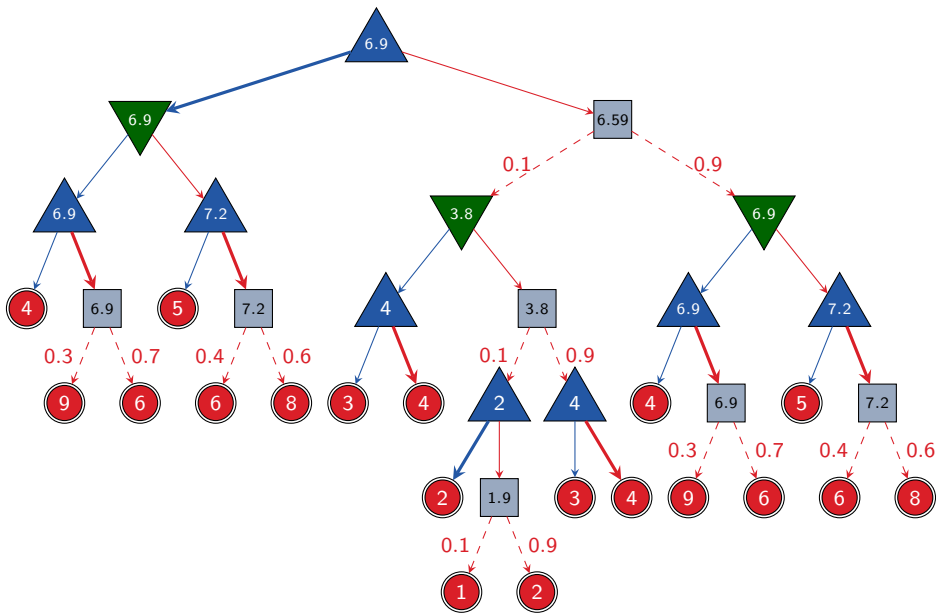


- **depth-first search** in game tree
- determine **utility value of terminal positions** with **utility function**
- compute **utility value of inner nodes** bottom-up through the tree:
 - MIN's turn: utility value is **minimum** of utility values of children
 - MAX's turn: utility value is **maximum** of utility values of children
 - chance: utility value is **expected value** of utility values of children
- **policy** for MAX: select action that leads to maximum utility value of children

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 utility 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 same improvements as for minimax are possible (evaluation functions, alpha-beta search).

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 **expected value** (probability-weighted sum) of all successors.
- Expectiminimax has **same guarantees** as minimax, but **in expectation**.