

Grundlagen der Künstlichen Intelligenz (5/6)

Spielbaumsuche

Dr. Christian Meilicke, Research Group Data and Web Science



Übersicht

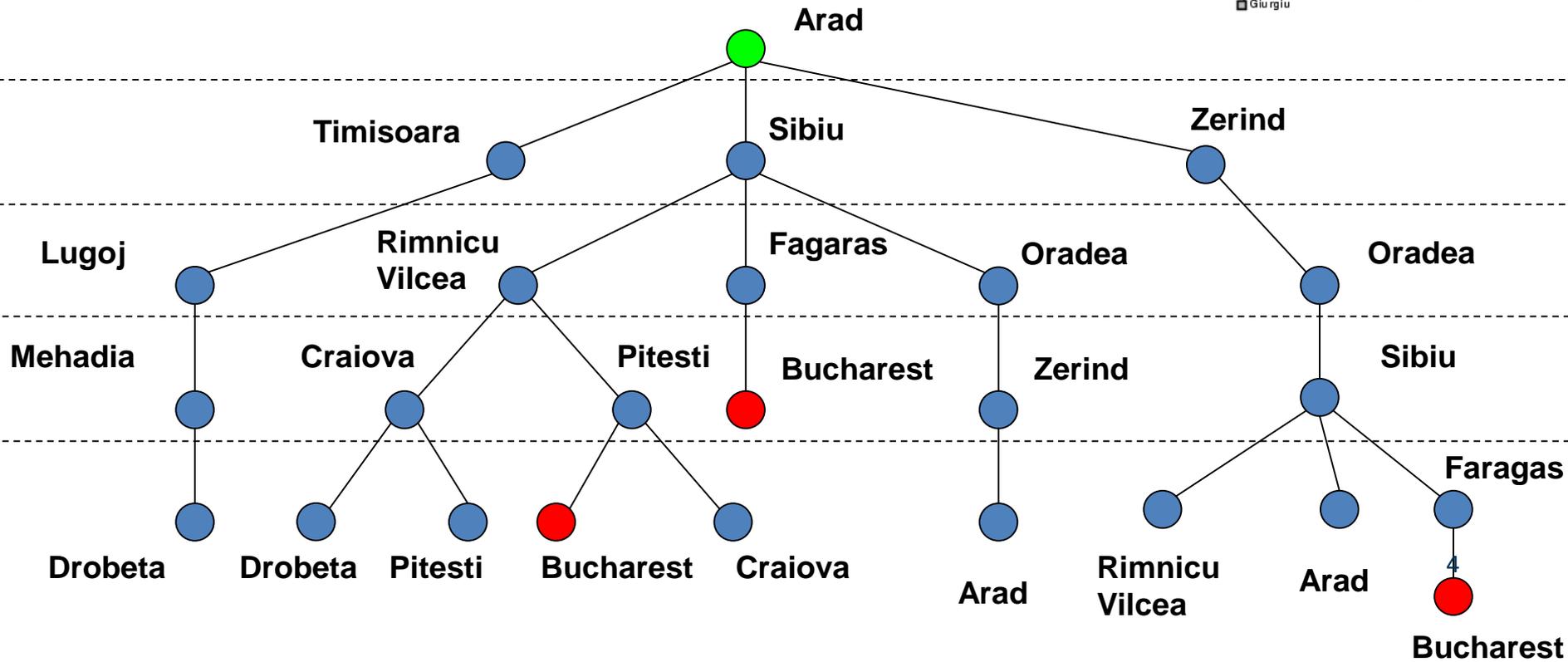
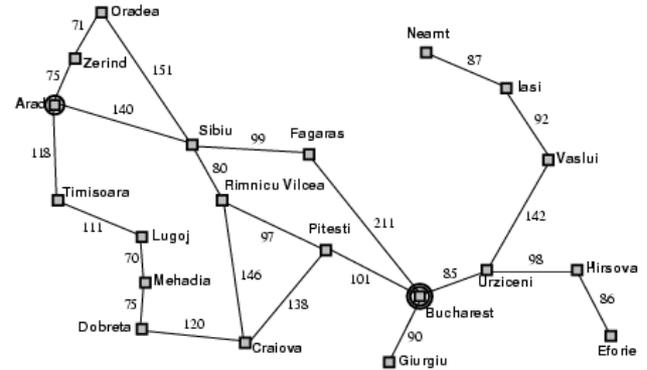
- Vorgestern: Suchverfahren von Breitensuche bis A* Suche
- Heute (erster Foliensatz): Lokale Suche
- Aktueller Foliensatz :
 - Spiele als Suchprobleme
 - Deterministisch und vollständig beobachtbar
 - Heuristiken und Min-Max Bäume, Alpha-Beta Pruning, Eröffnung und Endspiel
 - Nicht deterministische Spiele
- Nächster und letzter Foliensatz: MCTS Spielbaumsuche als Gegenstück zu dem “klassischen” Verfahren

Spieltypen

	Deterministic	Stochastic
Fully Observable	Schach, Go	Backgammon, Monopoly
Partially Observable	Schiffe Versenken	Bridge, Poker

- Single-Agent vs. Multi Agent

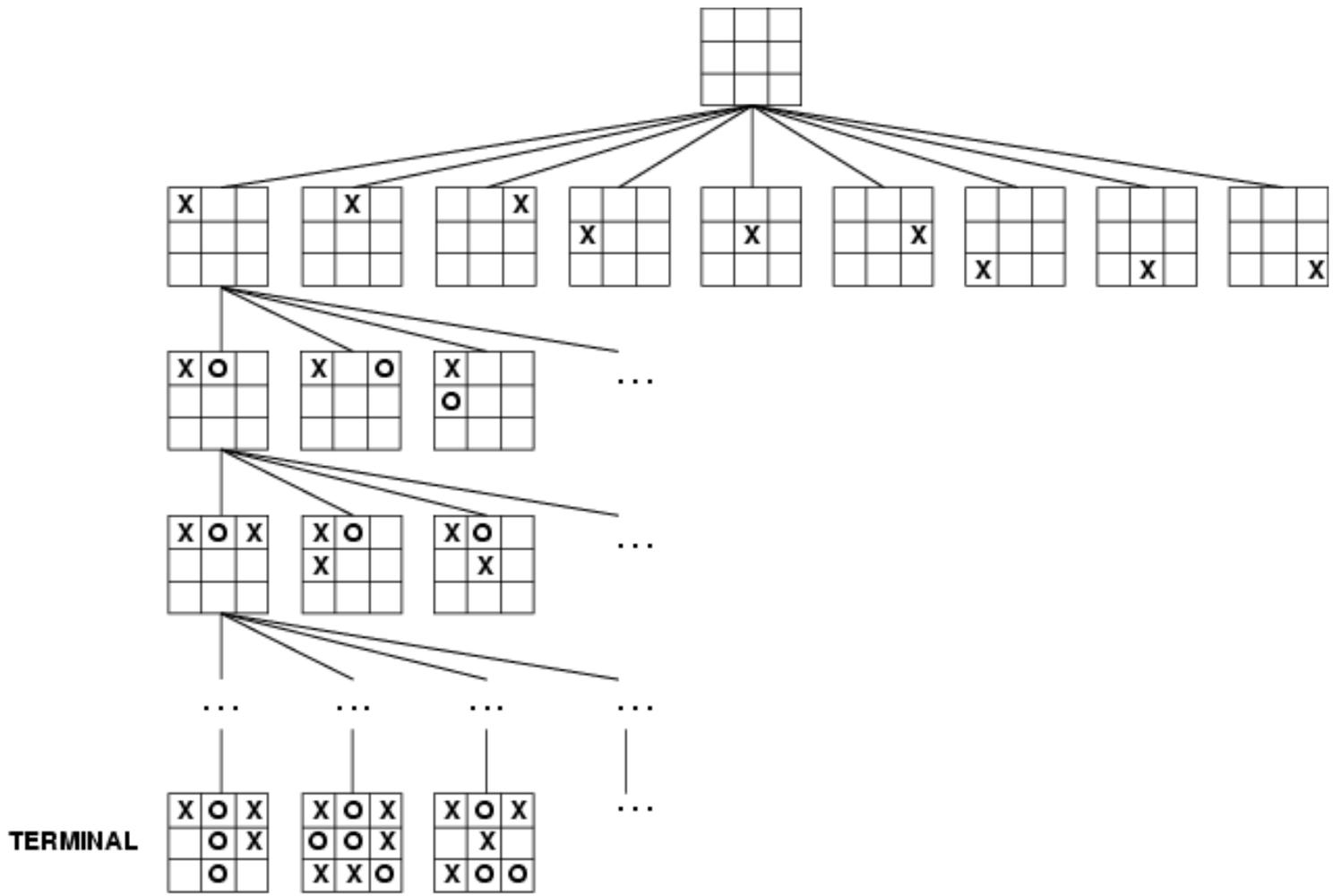
Erinnerung Suchverfahren



Spiele als Suchproblem

- Formulierung
 - Lösungsschritte: Spielzüge
 - Zielzustand: Gewinn oder Verlust
 - Startzustand: Aktuelle Position / Spielstand
 - Kosten?
 - Suchraum: “Game-Tree” (Spielbaum)
- Problem:
 - Züge des Gegners sind nicht planbar, oder doch?

Beispiel: Tic-tac-toe

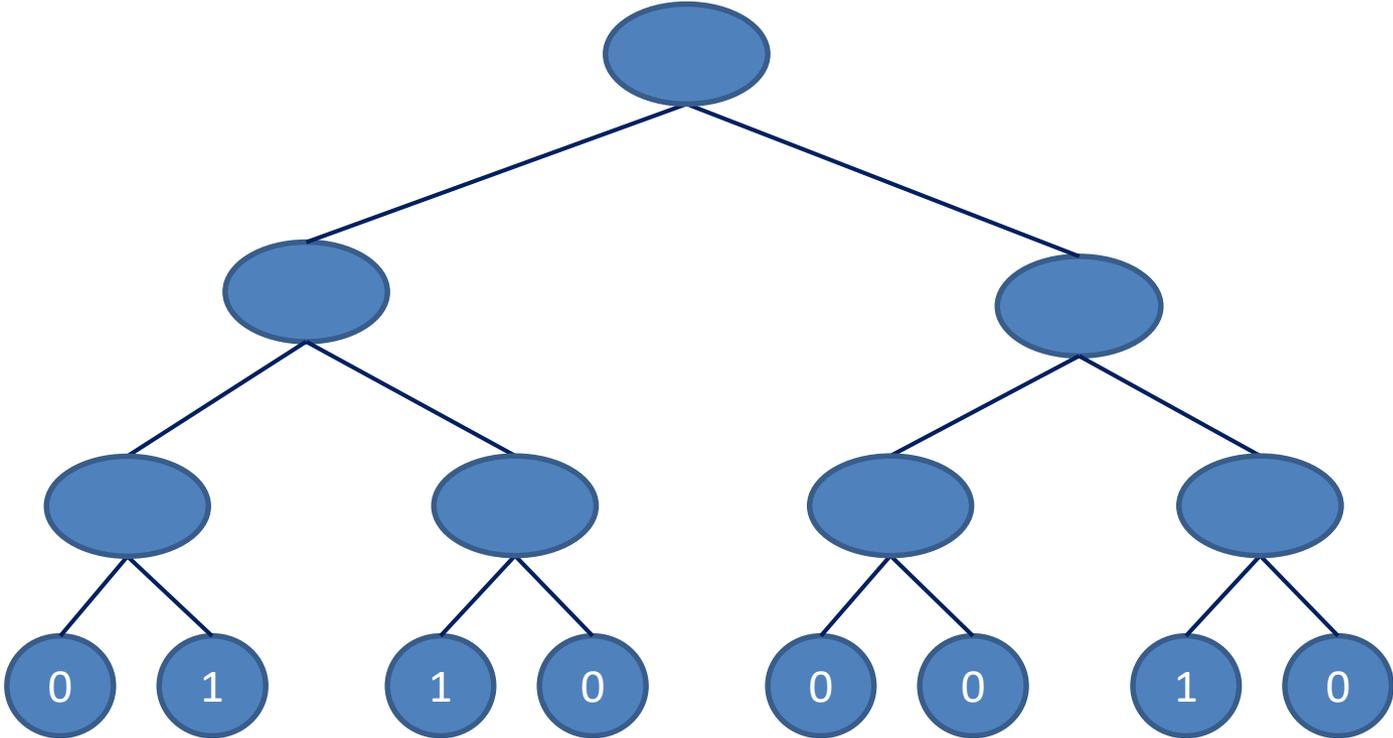


Spielbaum

Spieler A

Spieler B

Spieler A



1 → A hat gewonnen
0 → B hat gewonnen

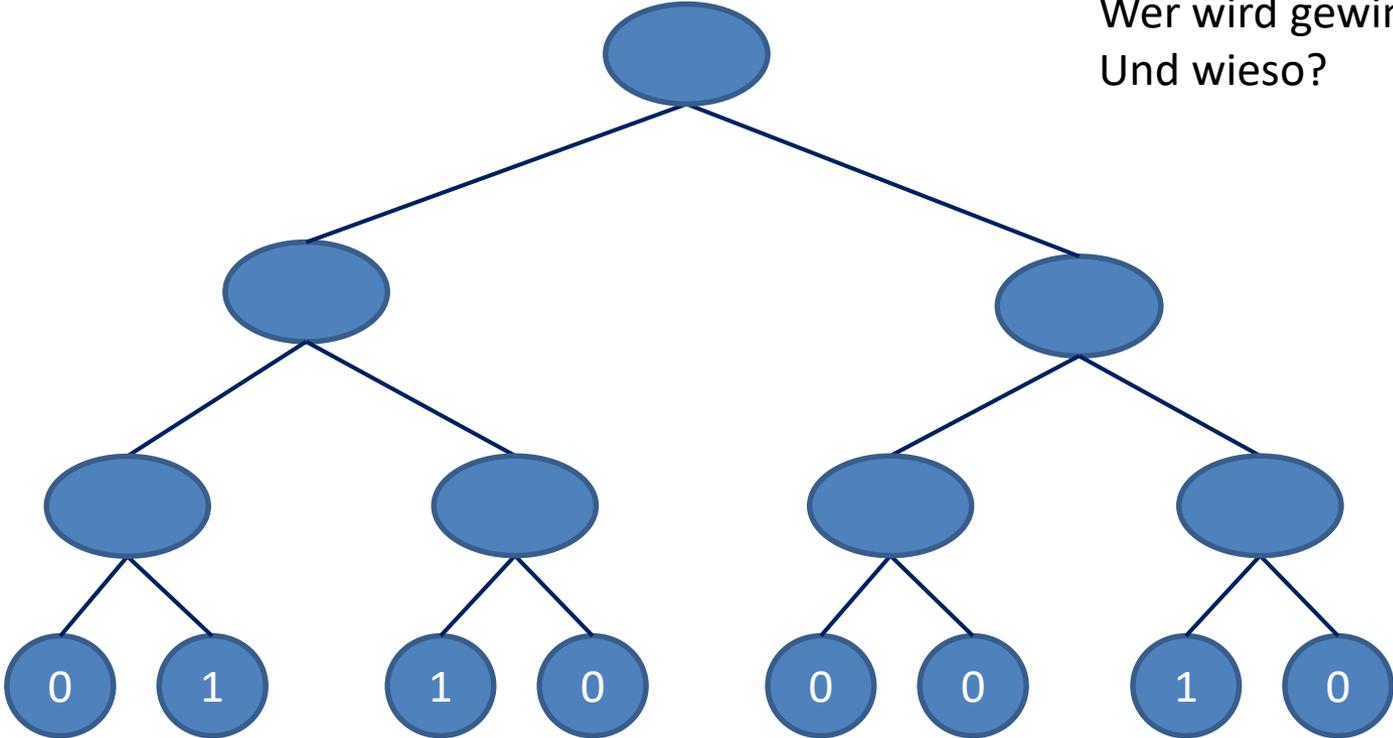
Spielbaum

Spieler A ist am Zug
Wer wird gewinnen?
Und wieso?

Spieler A

Spieler B

Spieler A



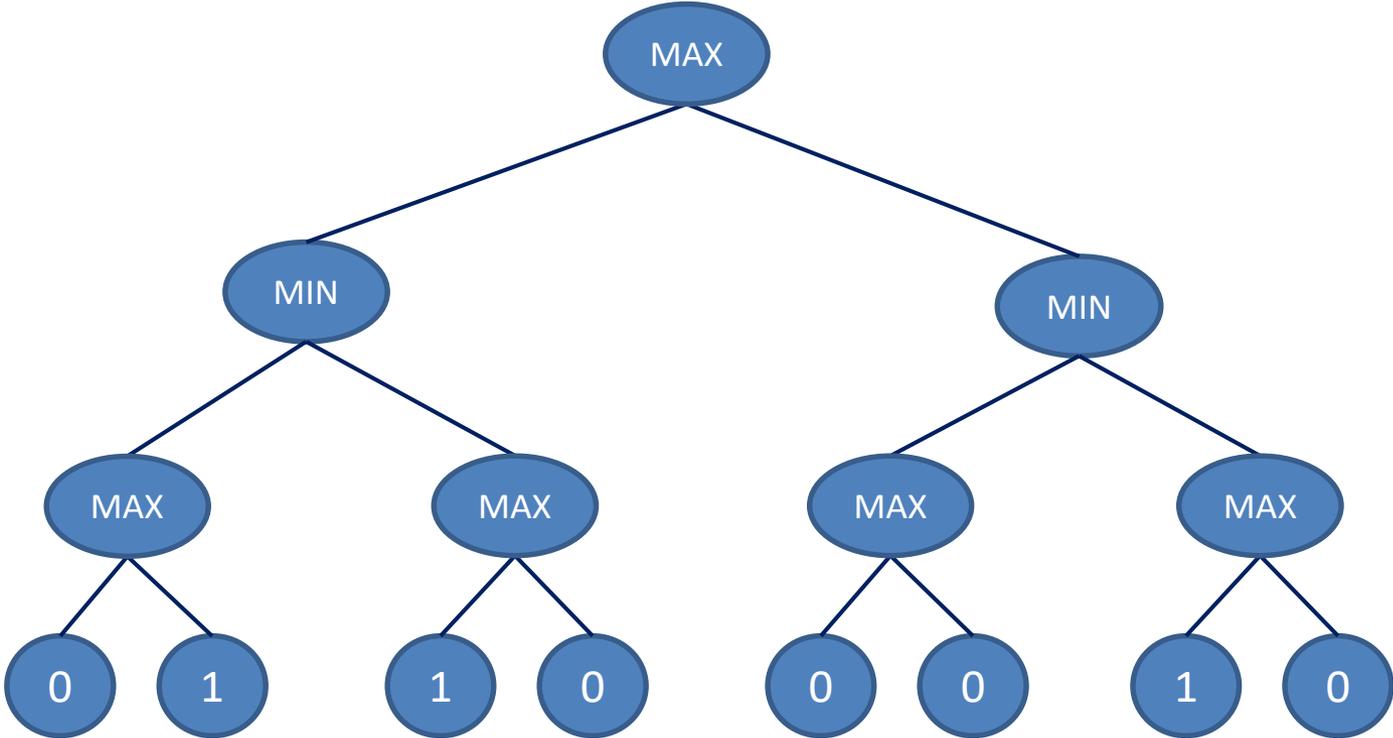
1 → A hat gewonnen
0 → B hat gewonnen

Spiele als Min/Max Bäume

Spieler A

Spieler B

Spieler A



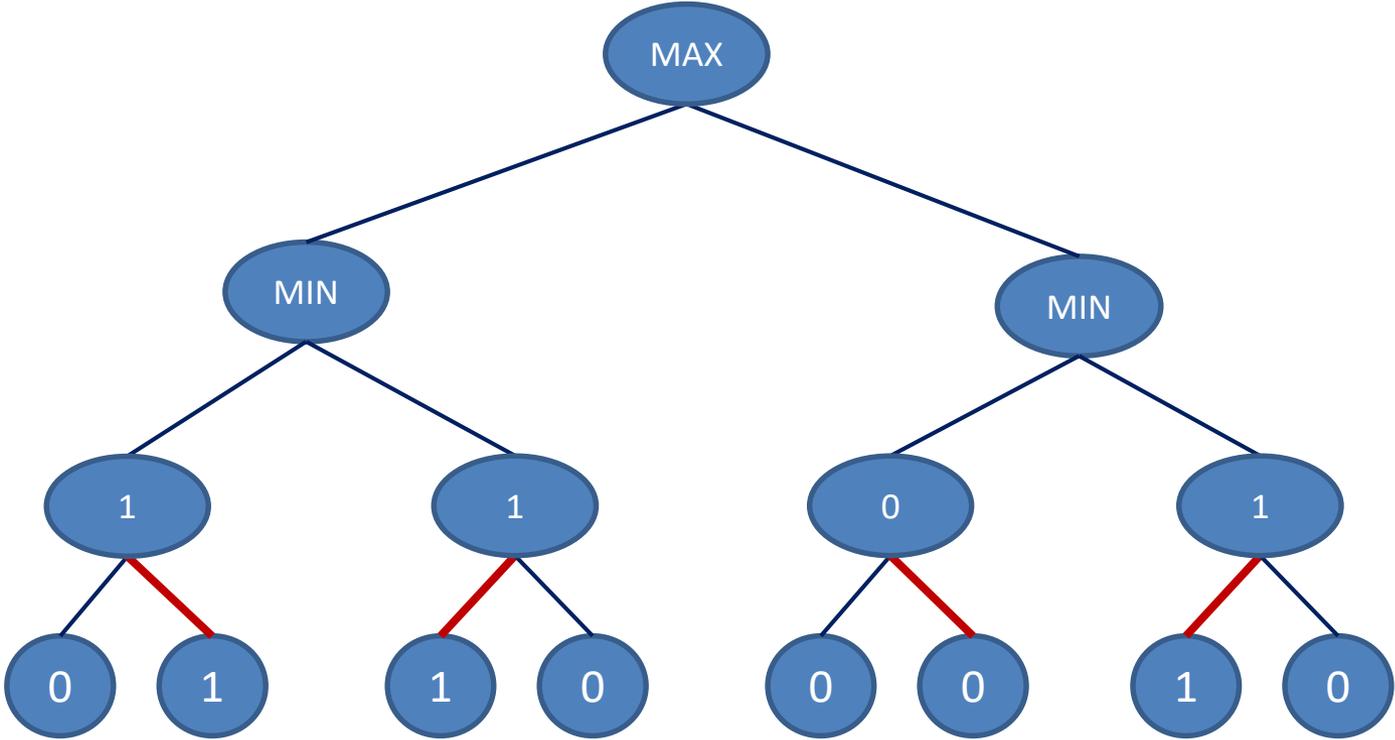
1 → A hat gewonnen
0 → B hat gewonnen

Spiele als Min/Max Bäume

Spieler A

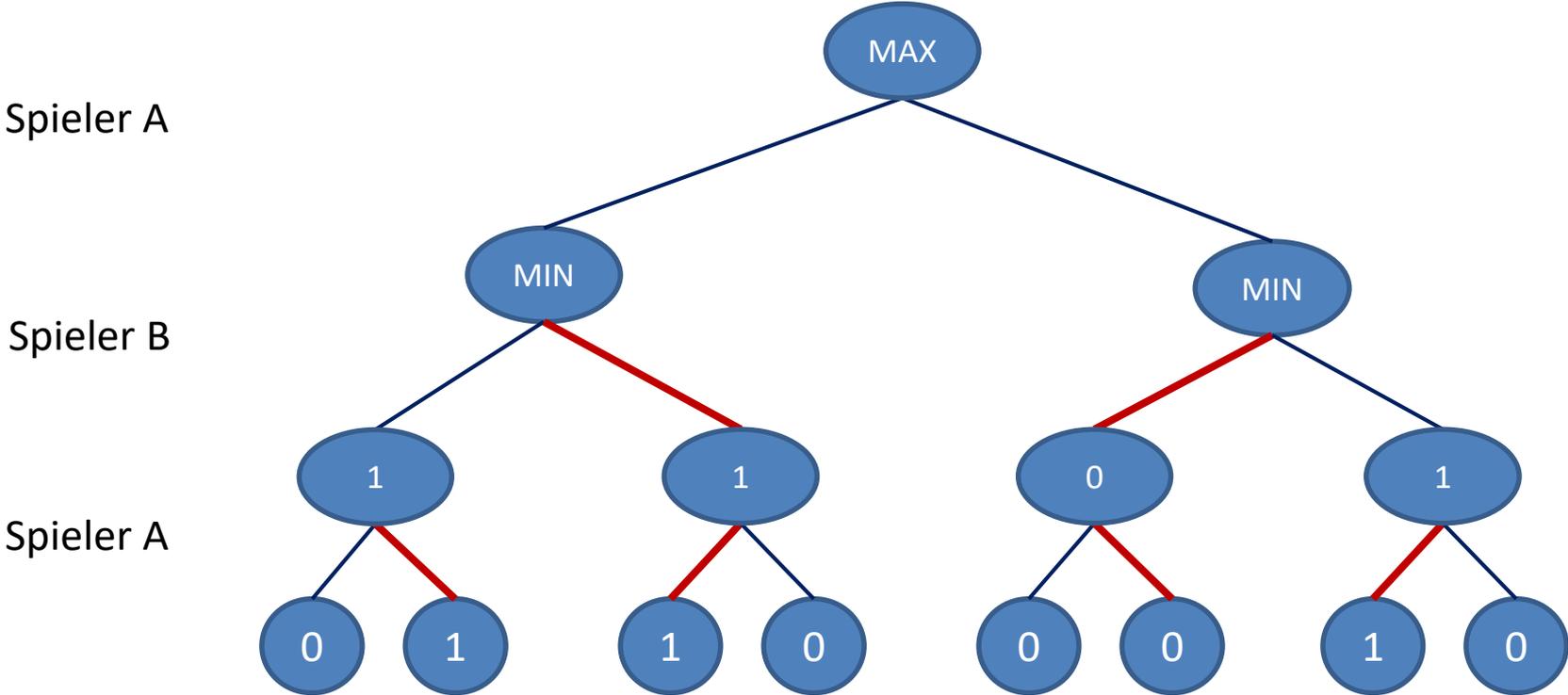
Spieler B

Spieler A



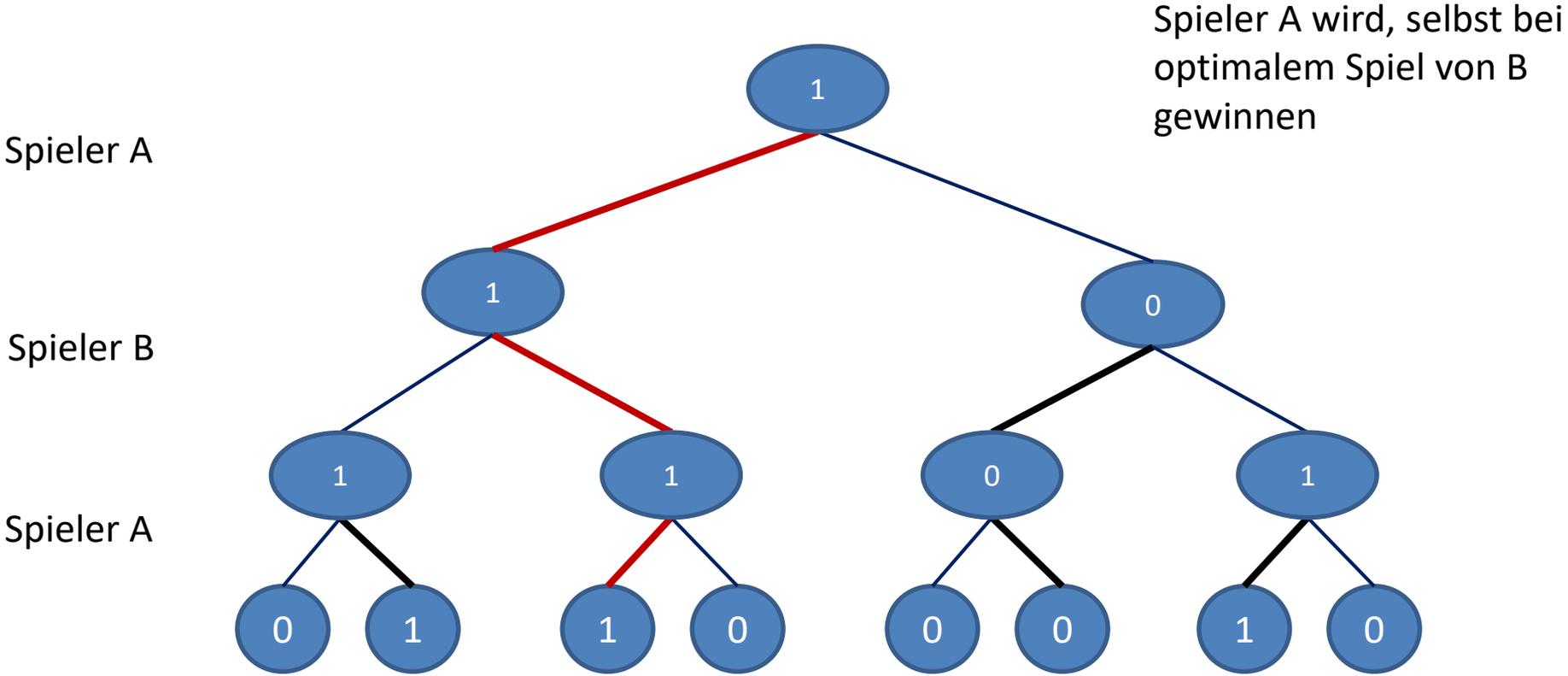
1 → A hat gewonnen
0 → B hat gewonnen

Spiele als Min/Max Bäume



1 → A hat gewonnen
0 → B hat gewonnen

Spiele als Min/Max Bäume



1 → A hat gewonnen
0 → B hat gewonnen

Probleme der Spielesuche

Wir können bestimmen, welche Züge zum Sieg führen, aber:

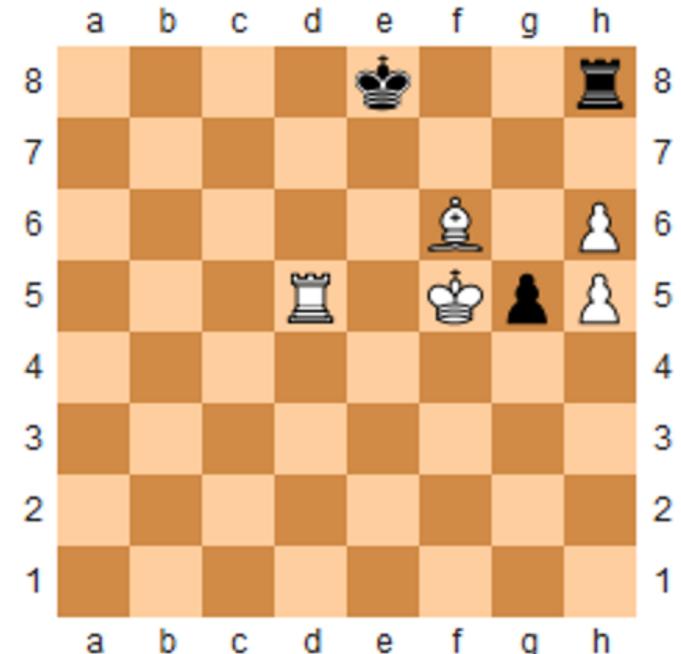
- Problem:
 - Suchbaum in der Regel zu groß, um ihn vollständig zu durchsuchen
- Lösung:
 - Tiefenbeschränkte Suche bis zu einer realisierbaren Tiefe
- Problem:
 - Woher wissen wir, wie das Spiel wirklich ausgeht ?
- Lösung:
 - Bewertungsheuristik für Zwischenzustände

Bewertungsheuristik

- Ähnlich zur Heuristik mit der man die Restkosten schätzt
 - Aber andere Semantik, nicht durcheinander bringen!
- Beispiel Schach:
 - Dame = 8, Turm = 5, Läufer = 3, Springer = 2, Bauer = 1
 - Heuristik = Summe eigene Figuren – Summe Figuren des Gegners
 - Muss erweitert werden um Stellungsbewertungen, z.B. „Springer am Rand bringt Unglück und Schande“
- Terminalzustände: +/- ∞

Anwendung einer Heuristik

- Beispiel
 - Weiß: $1 + 1 + 3 + 5 = 10$
 - Schwarz: $5 + 1 = 6$
 - Ergibt $10 - 6 = 4$
- Oft keine angemessene Bewertung des aktuellen Zustandes
 - Bei komplexen Spielen schwer sinnvoll zu definieren
- Beachten: Wird auf einen Zustand einige Züge in der Zukunft angewendet!



Partielle Spielbäume und das Min-Max Verfahren

Wir wählen den Zug mit dem größten erwarteten Nutzen

max

min

Der Gegner wählt den Zug, der für uns den geringsten Nutzen hat.

min

max

max

max

max

h(n)

2

5

9

3

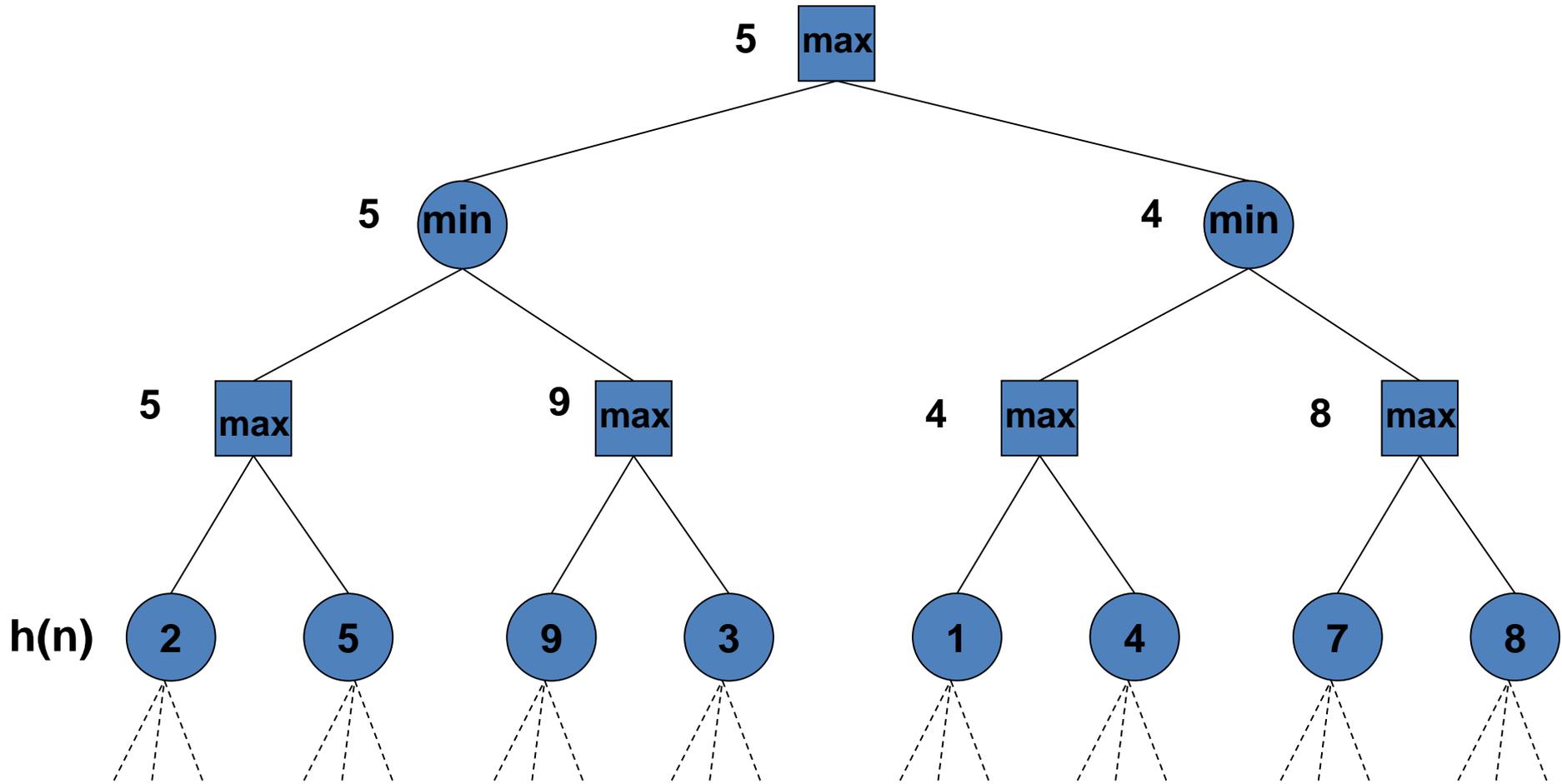
1

4

7

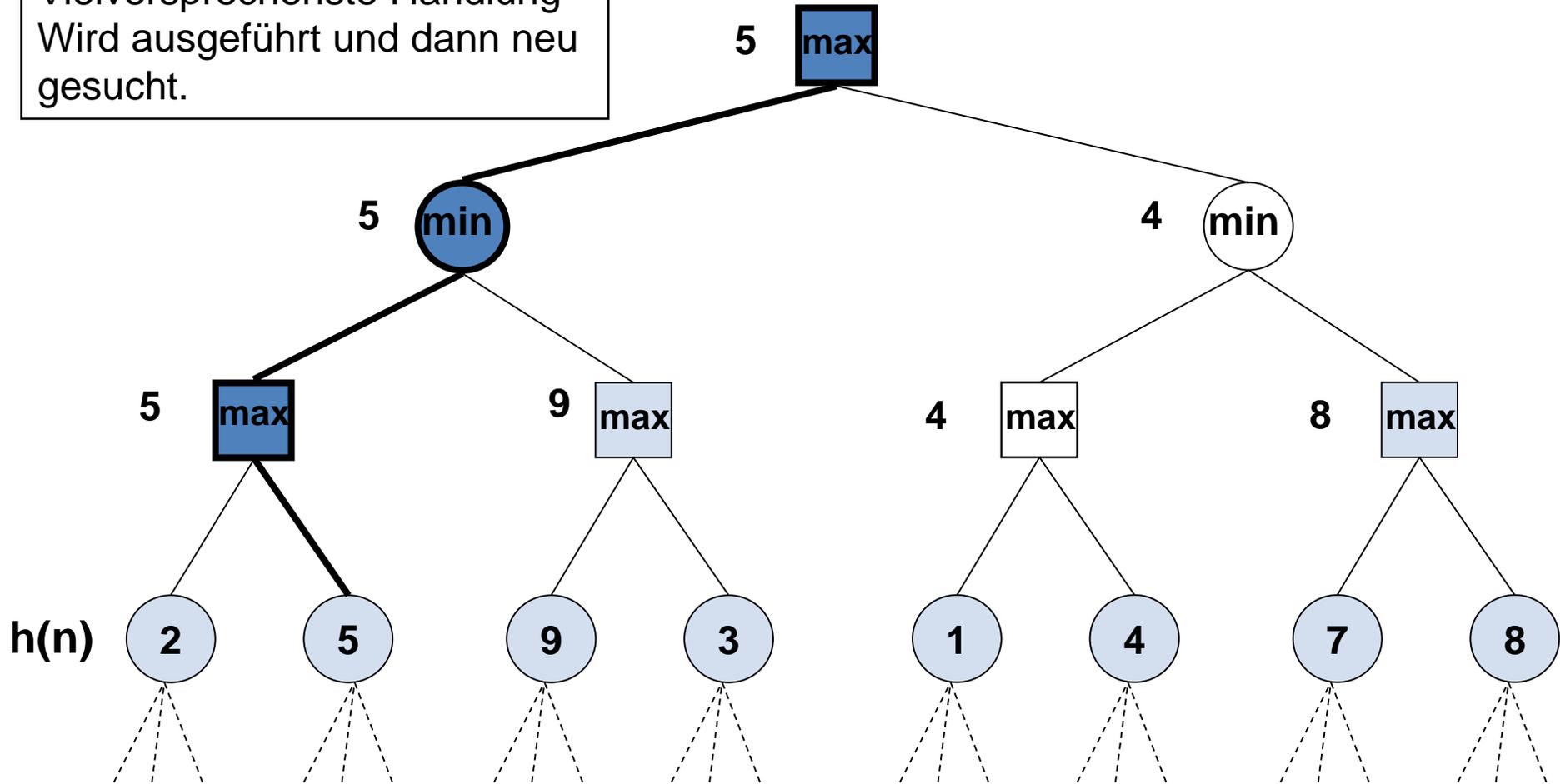
8

Min-Max Verfahren



Min-Max Verfahren

Vielversprechendste Handlung
Wird ausgeführt und dann neu
gesucht.



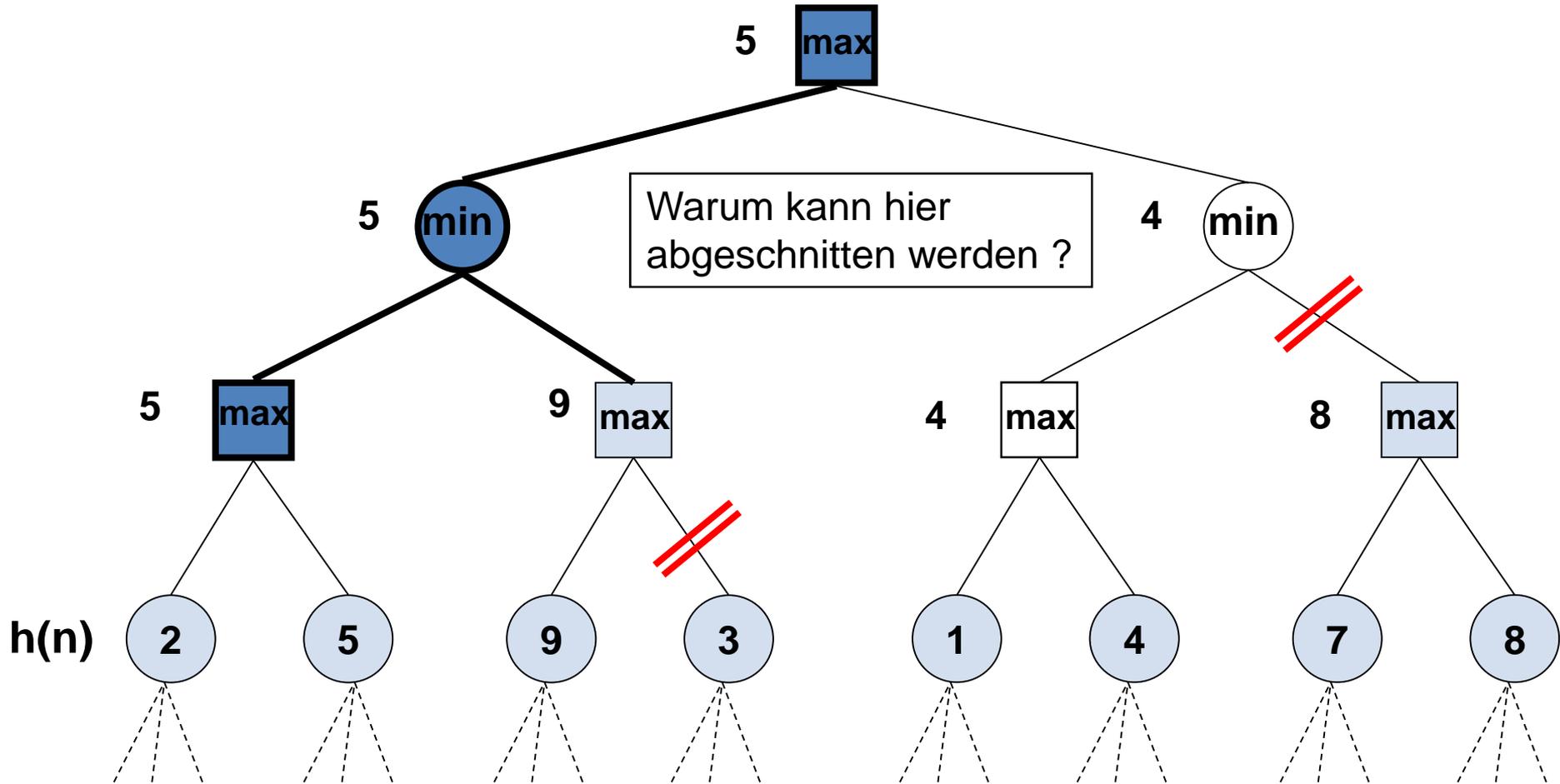
Min-Max als Tiefenbeschränkte Suche

„An der Tafel“

Anwendbarkeit

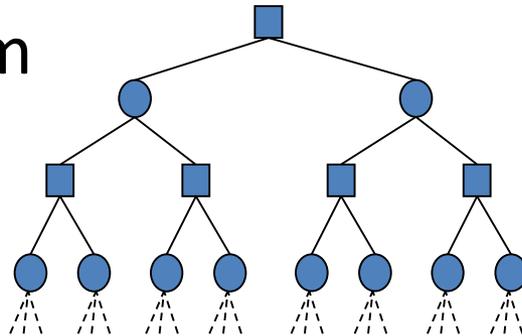
- Komplexität ist ein Problem
 - Reale Spiele haben riesige Suchräume
 - Dame: 10^{78} , Schach 10^{120} , Go 10^{761}
 - Beispiel Schach:
 - ca 40 Zugvarianten auf beiden Seiten
 - D.h. ca 2 Mio Knoten, wenn man 2 Züge im Voraus planen will
 - Selbst Anfänger planen 3 bis 4 Züge, Profis bis zu 10
- Lösung: Einschränkung (Pruning) des Suchraums
 - Idee: Äste, die keine bessere Lösung mehr enthalten können, werden nicht mehr durchsucht
 - Variante von Branch & Bound Verfahren

Pruningmöglichkeiten (α/β Pruning)



Einfluss der Sortierung

- Suchbaum



Pruning wird maximiert, wenn immer zuerst die aus der Sicht des jeweiligen Spielers besten Züge betrachtet werden

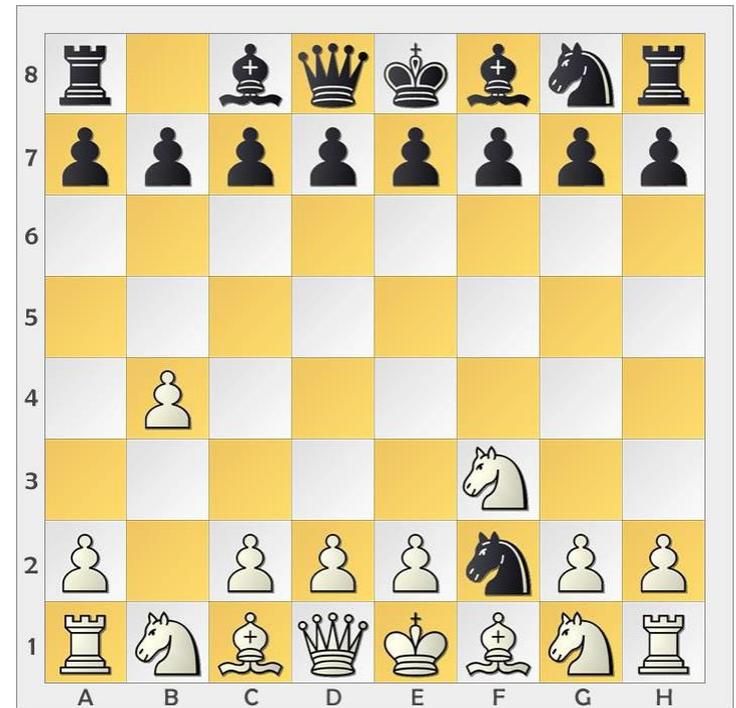
- Vielversprechende Züge zuerst betrachten, erlaubt es mehr Äste zu prunen!

Optimierungen / Erweiterungen

- Vorsortierung der Züge
 - Führe erst vielversprechende Züge aus
 - Verschiedene Ansätze um dies umzusetzen
- Iterative Tiefensuche
 - Verwendung zur besseren Vorsortierung
 - Ergebnisse Level n bestimmt Reihenfolge auf Level n+1
- Spielspezifische Heuristik
 - Schach: Erst Züge betrachten, die Figuren schlagen
- Killer Heuristik
 - Guter Zug in einem Teilbaum = guter Zug in anderem Teilbaum

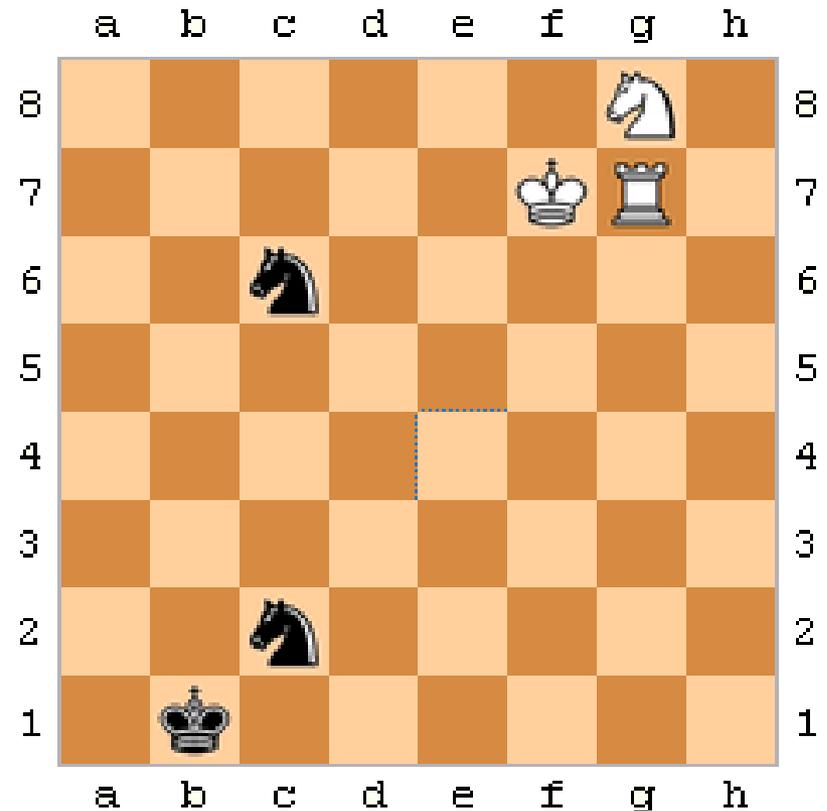
Killer Heuristik

- Weiss am Zug!
- Egal welchen Zug weiss macht, für schwarz ist es immer gut, die weiße Dame zu schlagen
- Wenn man dies für einen weißen Zug durchgerechnet hat, dann sollte man für jeden weiteren weißen Zug stets diesen Zug zuerst betrachten
- Ergibt deutlich mehr Pruning!



Endgame Tablebases

- Datenbank mit allen Positionen, für die das Ergebnis bei optimalem Spiel feststeht.
- Ist eine solche Position erreicht, muss nicht weitergesucht werden
- Für Schach gibt es eine komplette Endgame Datenbank für alle Endgames mit bis zu 6 Spielfiguren
- Variante: Positionen, die sicher zum schlagen einer Figur führen. Züge bis zum Schlagen als Bewertung



matt in 262 Zügen !!

Endgame Algorithmen

- Für Schach sind exakte Verfahren bekannt, wie man in bestimmten Situationen gewinnt
 - Zum Beispiel: Mit König und Turm (oder König, Springer, Läufer) kann man immer einen einzelnen König mattsetzen
- Erkennen (oder herbeiführen durch Suchen) einer solchen Ausgangslage
 - Kennzeichnen eines solchen Zustandes als Terminalzustand
- Deaktivieren des Suchverfahrens und aktivieren des „hartgecodeten“ Algorithmus
- Muss problem(=spiel)-spezifisch gelöst werden

Eröffnungen

- Werden in der Regel nicht über „on-the-fly“ Suchverfahren bestimmt
 - Können vorab berechnet werden mittels bekannter Suchverfahren
 - Können aus „bekannten Partien“ entnommen werden

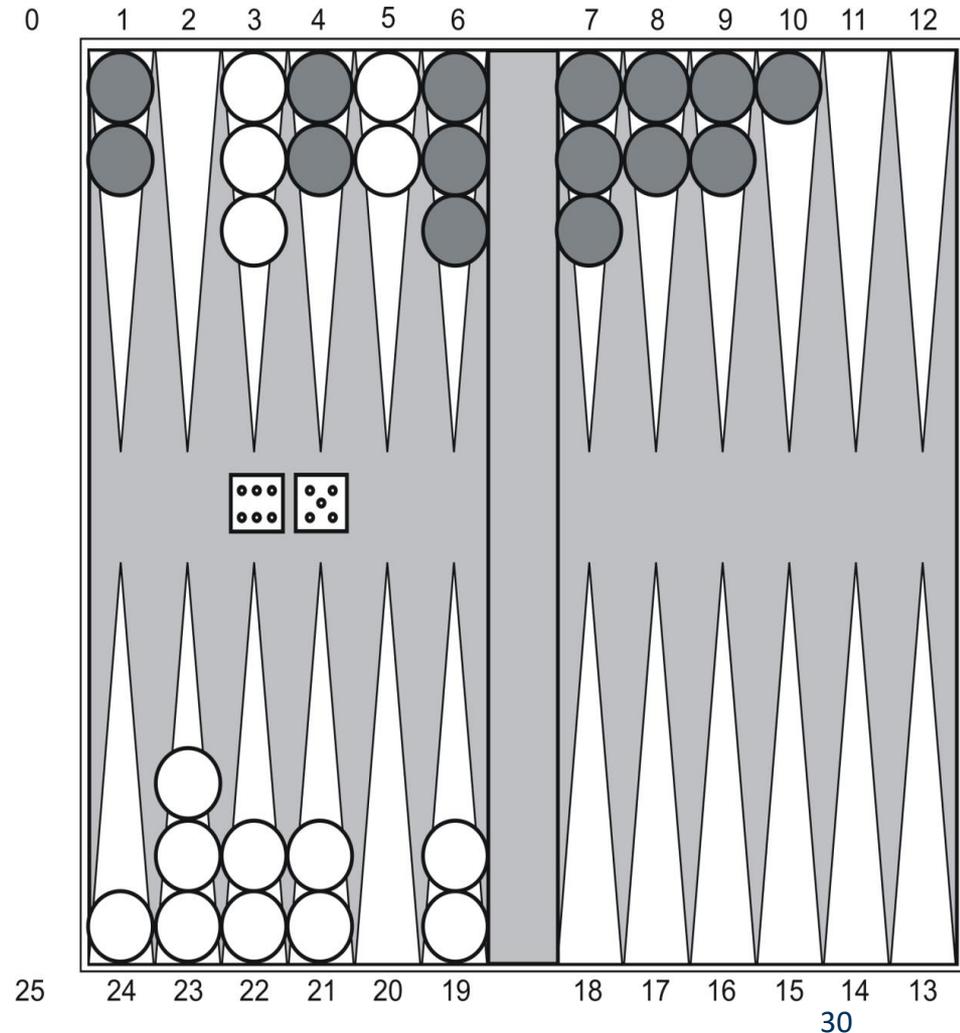
Spieltypen

	Deterministic	Stochastic
Fully Observable	Schach, Go	Backgammon, Monopoly
Partially Observable	Schiffe Versenken	Bridge, Poker



Nicht-Deterministische Spiele

- Beispiel:
Backgammon
- Mögliche Züge abhängig vom
Würfelergebnis
- Minimax-Baum
kann nicht
aufgebaut werden



Erwartetes Minimax

(Expectimax)

- Würfelaktionen werden als zusätzliche Knoten dargestellt
- Minimax-Werte für verschiedene Würfelergebnisse werden gemäß ihrer Wahrscheinlichkeit berücksichtigt

ExpectMM(n) =

Utility(n) , falls n TERMINAL

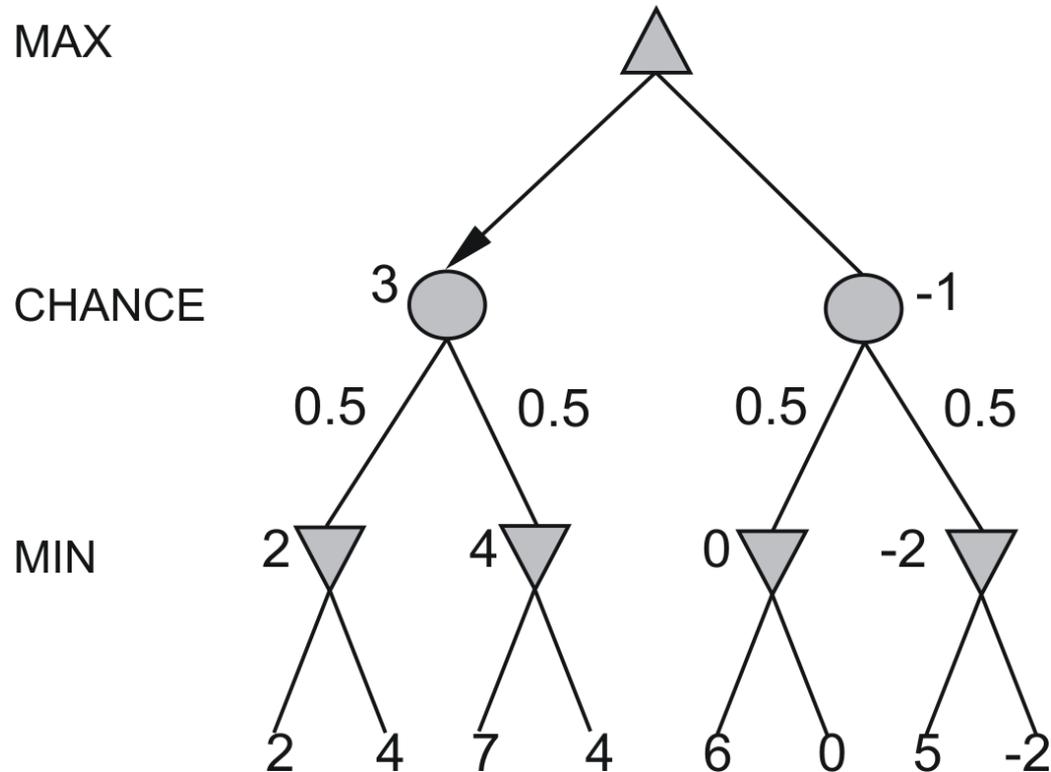
$\max\{\text{ExpectMM}(s) \mid s \text{ ist Nachfolger von } n\}$ bei eigenem Zug

$\min\{\text{ExpectMM}(s) \mid s \text{ ist Nachfolger von } n\}$ bei gegnerischem Zug

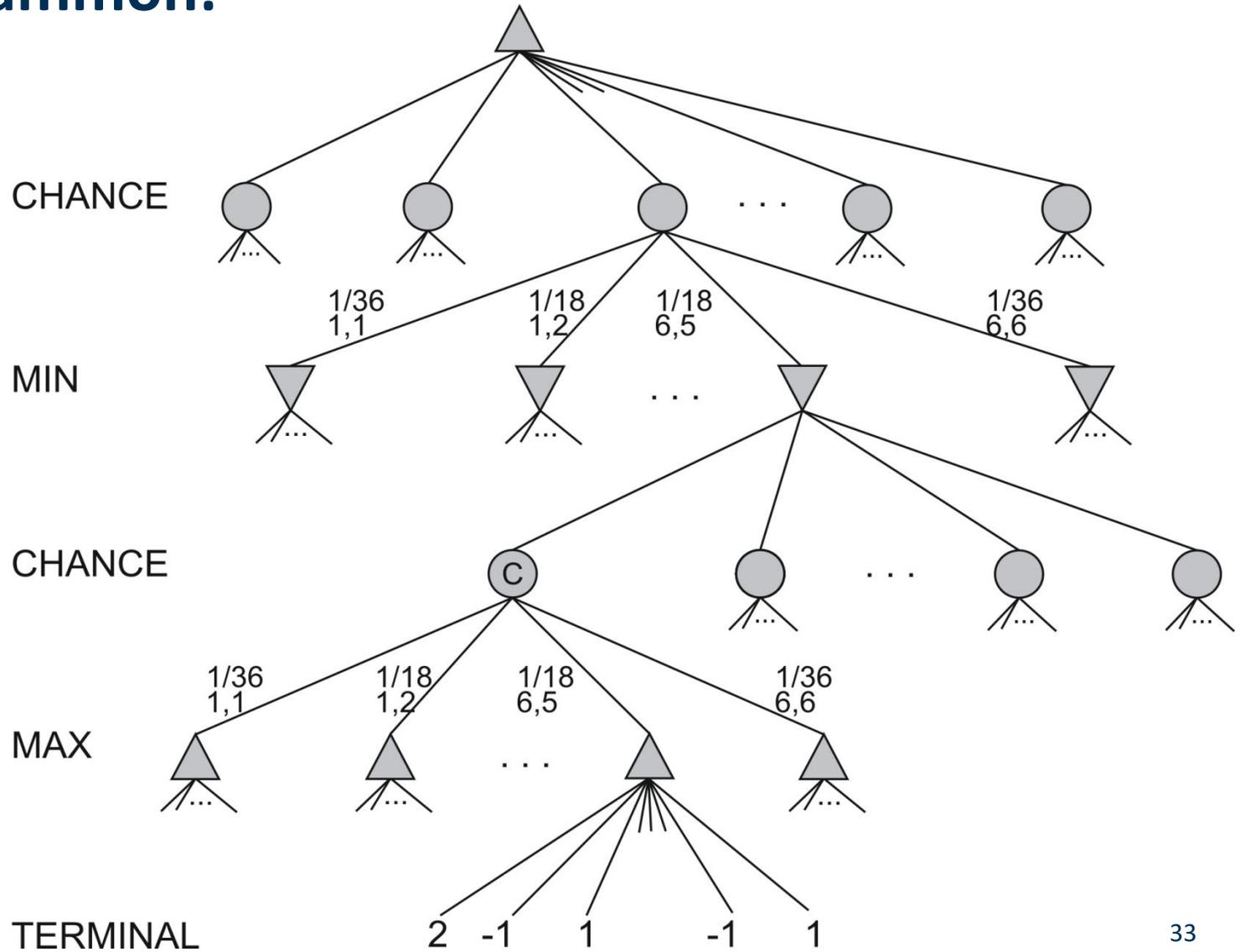
$\sum\{P(s) * \text{ExpectMM}(s) \mid s \text{ ist Nachfolger von } n\}$ bei Würfelaktionen

Ein einfaches Beispiel

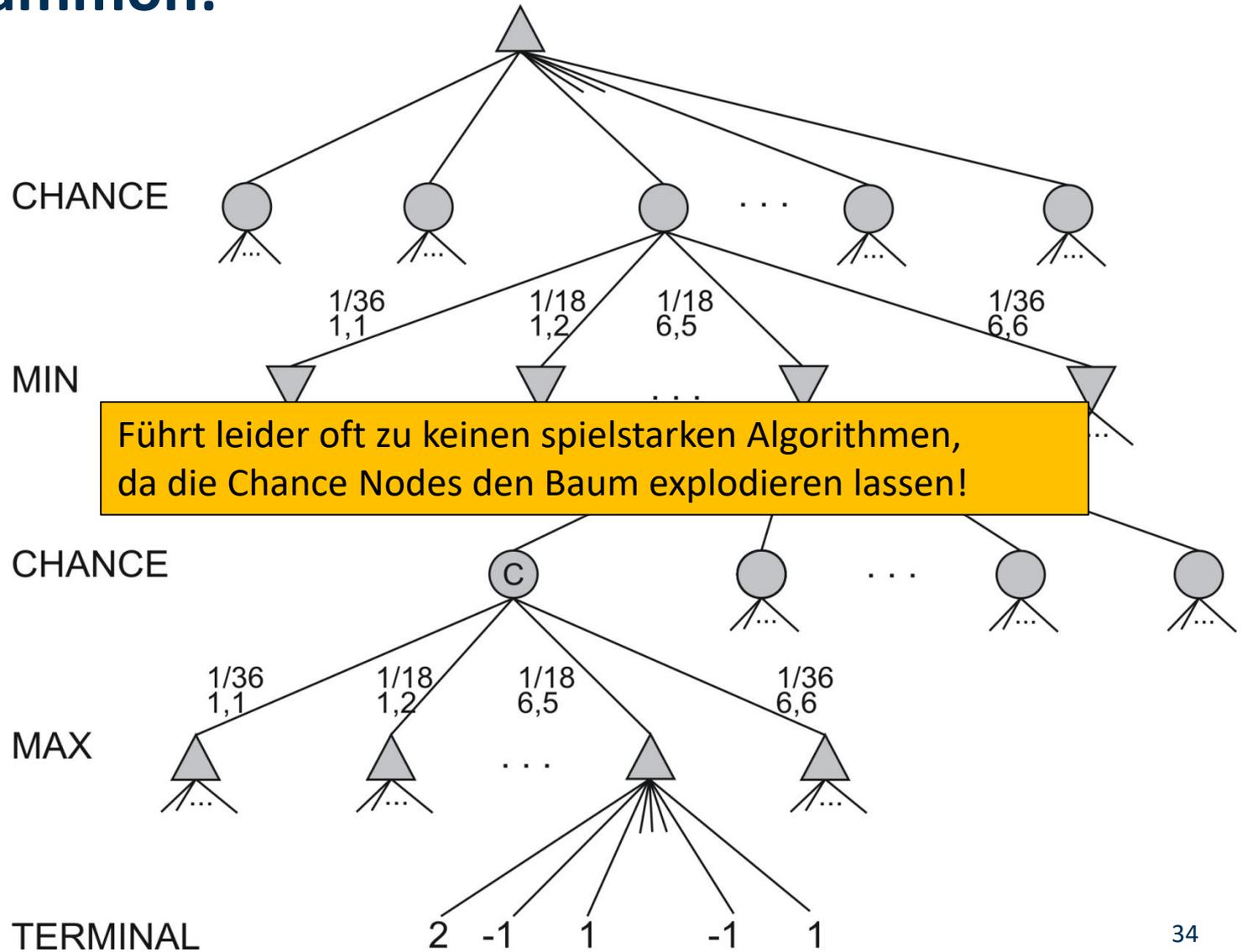
- Spiel mit Münzwurf:



Backgammon:

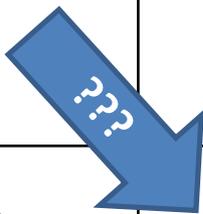


Backgammon:



Spieltypen

	Deterministic	Stochastic
Fully Observable	Schach, Go	Backgammon, Monopoly
Partially Observable	Schiffe Versenken	Bridge, Poker



Kartenspiele

- Möglicher Ansatz:
 - Eigene Karten bekannt; (Variante von) Min-Max für alle möglichen Verteilungen berechnen
 - Mittelwert bilden und danach für einen Zug entscheiden
- Problem: Kann unmöglich für alle Verteilungen berechnet werden
- Lösung: Monte Carlo Simulation, Berechnung für einige zufällige Verteilungen

Zusammenfassung / Fragen

- Spiele als Suchprobleme
- Deterministisch und vollständig beobachtbar
 - Heuristiken und Min-Max Bäume
 - Alpha-Beta Pruning
 - Ansätze um Alpha-Beta Pruning effektiver zu machen
 - Eröffnung und Endspiel
- Nicht deterministische Spiele
- Nächste Vorlesungseinheit: MCTS als Gegenstück zu dem “klassischen” Verfahren