# Einführung in die Künstliche Intelligenz

## Intelligente Agenten

### Allgemein

Nehmen ihre Umwelt wahr 🡪 Perzepte  
Manipulieren ihre Umwelt 🡪 Aktionen

Verhalten wird bestimmt durch   
 Perzeptfolge   
 ausführbare Aktionen  
 Leistungsmaße der Agentenfunktion  
 🡪 geeignete Maße wichtig

Rationale Agenten:   
 Nutzt Agentenfunktion:  
 Perzeptfolge × Agentenwissen 🡪 Aktion  
 🡪 Rational ≠ Allwissend

Ideale Rationale Agenten:  
 Maximierung des Lesitungsmaßes bei jeder Aktionsauswahl

### Agententypen

Tabellengesteuerte Agenten:  
 Aktionsauswahl erfolgt über Tabelle mit allen möglichen Perzeptfolgen und Aktionen

Reflex-Agenten:  
 Interpretation der Perzeptfolgen zu Zuständen  
 Auswahl der Aktion abhängig vom jetzigen Zustand

Agenten mit internem Zustand:  
 „Merken“ von der Auswirkung eigener Aktionen und Entwicklung der Umgebung

Agenten mit expliziten Zielen:  
 Agent antizipiert Ergebnis und wählt Aktion nach explizit gegebenen Zielen

Leistungsorientierte Agenten  
 Auswahl der Aktion je nach Qualitätsmaximierung (nach Leistungsmaßen) der Aktion

### Umgebung rationaler Agenten

Vollständig vs. Partiell beobachtbar  
 Sind alle relevanten Aspekte über Sensoren zugänglich?

Deterministisch vs. Indeterministisch  
 Hängt der nächste Weltzustand nur von der Aktionsauswahl ab?

Episodisch vs. Nicht-episodisch  
 Kann die Qualität der Aktion innerhalb eines Wahrnehmen-Agieren-Zyklus berwertet werden?

Statisch vs. Dynamisch  
 Kann sich die Welt verändern während die nächste Aktion ausgewählt wird?

Diskret vs. Kontinuierlich  
 Gibt es klare, diskrete Zustandübergänge?

## Problemlösen durch Suche

### Suche im Zustands baum

Gleicher Ablauf:  
 Auswahl eines Knotens 🡪 Zieltest 🡪 Expansion  
 bis Zieltest bestanden

Darstellung als Baum:  
 Knoten: Zustände  
 haben state, parent-node, children, action, depth, path-cost  
 Wurzel: Anfangszustand  
 Bätter: nicht expandierte oder geschlossene Knoten  
 Kanten: Aktionen

### Uninformierte bzw. Blinde Suche

Breitensuche:  
 Datenstruktur für Expansion: Queue  
 Knoten gleicher Tiefe testen und expandieren  
 Kein Zielknoten gefunden 🡪 Knoten nächster Tiefe  
 Vollständig, Optimal, (b: Expansionsgrad, d: Tiefe)

Uniforme Kostensuche:  
 Expansion nach geringsten Pfadkosten 🡪 Priority Queue  
 Optimal, wenn Atkionskosten > 0  
 ( Kosten optimaler Lösung

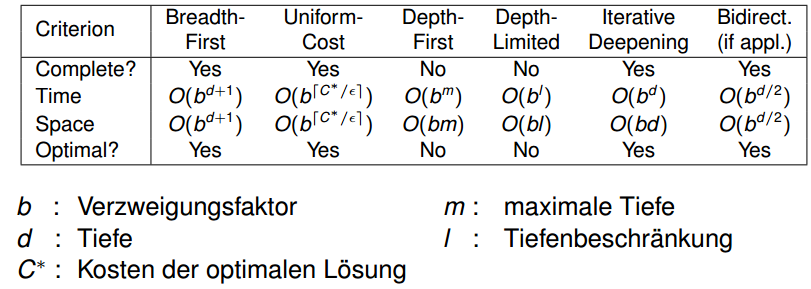
Tiefensuche:  
 Datenstruktur für Expansion: Stack  
 Expansion eines Nachfolgers des aktuellen Knoten  
 Keine Expansion mehr möglich 🡪 Expansion eines anderen Nachfolgers  
 Nicht-optimal, nicht-vollständig,

Backtracking:   
 Expansion eines Nachfolgers des aktuellen Knoten  
 Bei Scheitern ‚backtrack‘ zu vorherigen Knoten  
 Nicht-optimal, nicht-vollständig,

Tiefenbeschränktesuche:  
 Tiefensuche mit festgelegter maximaler Pfadlänge  
 vollständig, nicht-optimal, terminiert bei endlicher Verzweigung

Iterierte Tiefensuche:  
 Tiefenbeschränktesuche mit Iteration über die maximale Pfadlänge  
 Kombination aus Tiefen- und Breitensuche  
 Optimal, vollständig

Bidirektionale Suche  
 Gleichzeitige Entwicklung zweier Suchbäume  
 Reduzierung auf   
 Je nach Problem Erzeugung der Vorgägerknoten nicht möglich



Graphsuche:  
 Enthält zusätzlich closed-Set mit bereits besuchten Knoten

## Heuristische Suche

### Bestensuche

geschätzte Bewertung der Knoten   
 geschätzter Abstand eines Knotens zum Zielknoten

Bestensuche:  
 Instanz des Tree Search bzw. Graph Search Algorithmus  
 Expansion des Knotens mit   
 🡪 Datenstruktur: Priority Queue  
 Weder vollständig noch Optimal

Gierige Suche:  
 Wie Bestensuche jedoch

### A\* Suche

Kombination von gieriger Suche und uniforme Kostensuche  
, wobei tatsächliche Pfadkosten zu einem Knoten  
Als TreeSearch:   
 Optimal effizient  
 Optimal und vollständig, wenn zulässig

Sei die tatsächlichen Kosten von n zum Zielknoten, dann ist zulässig, wenn gilt.

Als GraphSearch:   
 weder vollständig noch optimal, außer:   
 h ist monoton, Aktionskosten sind positiv und

Sei die Kosten der Atkion a um vom Knoten n zum Knoten n‘ zu gelangen. Dann heißt h monoton, wenn