

Grundlagen der Künstlichen Intelligenz

6. Klassische Suche: Repräsentation von Zustandsräumen

Malte Helmert

Universität Basel

6. März 2015

Klassische Suche: Überblick

Kapitelüberblick klassische Suche:

- 5.–7. Grundlagen
 - 5. Zustandsräume
 - 6. Repräsentation von Zustandsräumen
 - 7. Beispiele von Zustandsräumen
- 8.–12. Basisalgorithmen
- 13.–20. heuristische Algorithmen

Repräsentation von Zustandsräumen

Repräsentation von Zustandsräumen

- praktisch interessante Zustandsräume sind oft **sehr gross** (10^{10} , 10^{20} , 10^{100} Zustände)
- Wie **repräsentieren** wir sie so, dass wir sie effizient algorithmisch verarbeiten können?

Drei wesentliche Möglichkeiten:

- 1 als **explizite (gerichtete) Graphen**
- 2 mit **deklarativen Repräsentationen**
- 3 als **Black Box**

Explizite Graphen

Zustandsräume als explizite Graphen

Zustandsräume als explizite Graphen

Repräsentiere Zustandsräume als **explizite gerichtete Graphen**:

- Knoten = Zustände
- gerichtete Kanten = Transitionen

↗ repräsentiert über **Adjazenzlisten** oder **Adjazenzmatrix**

Beispiel (expliziter Graph für 8-Puzzle)

`puzzle8.graph`

Zustandsräume als explizite Graphen: Diskussion

Diskussion:

- für **grosse** Zustandsräume **unmöglich** (Platzbedarf zu gross)
- wenn Räume klein genug für explizite Repräsentation
einfach lösbar: **Dijkstras Algorithmus** $O(|S| \log |S| + |T|)$
- interessant für zeitkritische **All-Pairs-Shortest-Path**-Anfragen
(Beispiele: Routenplanung, Pfadplanung in Computerspielen)

Deklarative Repräsentationen

Zustandsräume mit deklarativen Repräsentationen

Zustandsräume mit deklarativen Repräsentationen

Stelle Zustandsräume **deklarativ** dar:

- **kompakte** Beschreibung des Zustandsraums als Eingabe für Algorithmen
 - ↪ Zustandsraum **exponentiell grösser** als Eingabe
- Algorithmen arbeiten **direkt auf kompakter Beschreibung**
 - ↪ erlaubt automatische Problemumformulierung, Vereinfachung, Abstraktion, usw.

Beispiel (deklarative Repräsentation für 8-Puzzle)

```
puzzle8-domain.pddl + puzzle8-problem.pddl
```

Black Box

Zustandsräume als Black Box

Zustandsräume als Black Box

Definiere ein **abstraktes Interface** für Zustandsräume.

Für Zustandsraum $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$

benötigen wir die Methoden:

- **init()**: erzeugt Anfangszustand
Ergebnis: Zustand s_0
- **is_goal(s)**: testet, ob s ein Zielzustand ist
Ergebnis: **true** wenn $s \in S_\star$; **false** sonst
- **succ(s)**: erzeugt anwendbare Aktionen und Nachfolger von s
Ergebnis: Folge von Paaren $\langle a, s' \rangle$ mit $s \xrightarrow{a} s'$
- **cost(a)**: liefert Kosten der Aktion a
Ergebnis: $cost(a) (\in \mathbb{N}_0)$

Anmerkung: wir werden dieses Interface später leicht,
aber entscheidend erweitern

Zustandsräume als Black Box: Beispiel und Diskussion

Beispiel (Black-Box-Repräsentation für 8-Puzzle)

Demo: `puzzle8.py`

- im Folgenden: Konzentration auf Black-Box-Modell
- explizite Graphen nur als illustrierende Beispiele
- gegen Ende des Semesters: deklarative Zustandsräume
(Handlungsplanung)

Zusammenfassung

Zusammenfassung

- Zustandsräume oft **sehr gross** ($> 10^{10}$ Zustände)
 $\rightsquigarrow$ **Wie repräsentieren?**
- **explizite Graphen**: Adjazenzlisten oder -matrix;
 nur für kleine Probleme geeignet
- **deklarativ**: kompakte Beschreibung als Eingabe
 für Suchalgorithmen
- **Black Box**: abstraktes Interface implementieren