

Was bisher geschah

- ▶ Definition KI (Versuche früher und jetzt)
- ▶ KI-Geschichte
- ▶ KI-Tests (Turing, Chinese Room)
- ▶ Daten, Information, Wissen, Intelligenz
- ▶ symbolische / subsymbolische KI
- ▶ explizites und implizites Wissen

Problemlösung durch Suche in Graphen – Beispiele

- ▶ Finden von Wegen in einem Graphen
 - ▶ Aufgabe:
 - ▶ gegeben: Graph G (Tafel)
 - ▶ gesucht: Weg (Pfad) in G von Knoten u zu Knoten v
 - ▶ Lösungsidee: Suche im Graphen
- ▶ Münzenstapelspiel (für eine Person)
 - ▶ Aufgabe:
 - ▶ gegeben: Stapel von n Münzen
 - ▶ gesucht: Zugfolge durch erlaubte Züge (zwei Münzen von einem Stapel nehmen und auf beide Nachbarn verteilen) bis zu einer Situation, in der kein Zug möglich ist
 - ▶ Lösungsidee:
 - ▶ Modellierung als Zustandsübergangssystem (Graph mit Markierungen)
 - ▶ Suche im Graphen
- ▶ 3 Krüge
 - ▶ Aufgabe:
 - ▶ gegeben: 3 volle Krüge mit Volumen 9l, 7l, 4l,
 - ▶ gesucht: genau 6l in einem der 3 Krüge
 - ▶ Lösungsidee: Zustände als Knoten eines Suchbaumes

Suchprobleme – Kontext, Aufgabe und Lösung

Darstellung von

Kontext (Zustandsübergangssystem)
als Graph $G = (V, E)$ mit

- ▶ Menge V von Zuständen (evtl. unendlich)
oft beschrieben durch Eigenschaften
- ▶ (gerichtete) Kanten:
mögliche Übergänge zwischen Zuständen
Übergangsrelation $E \subseteq V \times V$

Aufgabe definiert durch

- ▶ Startzustand $s \in V$
- ▶ Menge $Z \subseteq V$ von Zielzuständen
(oder Eigenschaften der Zielzustände)

Lösung: Folge von Zuständen
(Weg von einem Start- zu einem Zielzustand)
(Mitunter interessiert nur der erreichte Zielzustand.)

Entfaltung des Graphen zu einem Baum:

Pfade im Graphen = Knoten im Baum

Beispiel Missionare und Kannibalen

informale Problembeschreibung:

- ▶ Zu Beginn: 3 Missionare und 3 Kannibalen am linken Flussufer
- ▶ Ziel ist das Übersetzen aller Personen an das rechte Ufer.
- ▶ Links liegt ein Ruderboot, welches höchstens zwei Personen fasst.
- ▶ Sobald an einem Ufer mehr Kannibalen als Missionare sind, werden alle Missionare dort gefressen.

Aufgaben:

- ▶ formale Darstellung
 - ▶ Zustände
 - ▶ Übergänge
 - ▶ Zielbedingung
- ▶ Heuristiken
- ▶ geeignete Suchverfahren, Verbesserungsmöglichkeit

Problemlösen durch Suchen

- ▶ formale Darstellung des Problemes als Graph (z.B. Baum, DAG)
- ▶ formale Beschreibung der Lösung als Eigenschaft von
 - ▶ Pfaden im Graphen
 - ▶ Knoten im Baum

Möglichkeiten zum Problemlösen:

- ▶ Pfadsuche im Graphen
- ▶ Knotensuche im Baum

Suche in Graphen

(aus dem Modul Algorithmen und Datenstrukturen schon bekannte) Verfahren zur Suche in Graphen (und Bäumen):

Tiefensuche (depth-first search):

Suche zuerst in Teilbäumen eines noch nicht besuchten Nachbarn des aktuellen Knotens

Breitensuche (breadth-first search):

Suche zuerst in Teilbäumen eines noch nicht besuchten Knotens mit der geringsten Tiefe

Allgemeines Suchverfahren

Daten: L_a Menge der noch zu expandierenden Knoten (Agenda)
 L_x Menge der expandierten Knoten
 s Startknoten
 φ Anforderungen an Lösung (Eigenschaft der Zielknoten)

Allgemeiner Suchalgorithmus:

1. $L_a = \{s\}, L_x = \emptyset$
2. solange $L_a \neq \emptyset$:
 - 2.1 Wähle auf **festgelegte Art** einen Knoten u in L_a und verschiebe u aus L_a in L_x .
 - 2.2 Füge alle Nachbarn v von u mit $v \notin L_a \cup L_x$ und $v \notin \text{Mod}(\varphi)$ auf eine **festgelegte Art** in L_a ein.
(falls für ein v gilt $v \in \text{Mod}(\varphi)$: Abbruch mit Lösung v)

prominente Spezialfälle:

Tiefensuche ▶ Wähle Knoten, der **zuletzt** in L_a eingefügt wurde
 ▶ Einfügen der Nachbarn an den **Anfang** der Liste L_a
 ▶ Verwaltung von L_a als **Stack**

Breitensuche ▶ Wähle Knoten, der **zuerst** in L_a eingefügt wurde
 ▶ Einfügen der Nachbarn an das **Ende** der Liste L_a
 ▶ Verwaltung von L_a als **Queue**

Eigenschaften von Suchverfahren

Korrektheit: jede gefundene Lösung erfüllt die Lösungsbedingung

Vollständigkeit: jede Lösung wird gefunden

Fairness: jeder Knoten wird expandiert

Optimalität: (eine) beste Lösung wird zuerst gefunden

Zeitaufwand: (im ungünstigsten Fall) größtmögliche Anzahl der Schritte von einem Start- zu einem Endzustand

(Speicher-)Platzbedarf: (im ungünstigsten Fall) größtmögliche Länge der Anzahl der noch nicht expandierten Knoten in der Agenda

Laufzeit für Baum mit maximaler Tiefe t und maximalem Verzweigungsgrad d :

- ▶ Breitensuche: vollständig,
(optimal, falls beste Lösung = geringe Tiefe)
großer Zeitaufwand: $O(d^t)$
großer Platzbedarf: $O(d^t)$
- ▶ Tiefensuche: unvollständig bei Bäumen mit unendlich langen Zweigen, nicht optimal
großer Zeitaufwand: $O(d^t)$
geringerer Platzbedarf: $O(td)$

Suche mit schrittweiser Vertiefung

beschränkte Tiefensuche:

1. festgelegte Tiefenbeschränkung $m \in \mathbb{N}$
2. Tiefensuche auf allen Pfaden bis zur Tiefe m

nicht vollständig, weiter entfernte Lösungen werden nicht gefunden

Schrittweise Vertiefung (iterative deepening)

Kombination aus Breiten- und Tiefensuche durch
Nacheinanderausführung der beschränkten Tiefensuche für alle
 $m \in \mathbb{N}$, solange keine Lösung gefunden wurde

vollständig, optimal

(asymptotischer) Zeit- und Platzbedarf wie Tiefensuche

Gleiche-Kosten-Suche (kleinste bisherige Kosten)

(uniform-cost-search)

bei Zustandsübergängen mit verschiedenen **Kosten**

Ziel: Lösung (Pfad vom Start- zu einem Lösungsknoten) mit möglichst geringen Pfadkosten

(Pfadkosten = Summe der Kosten aller Übergänge auf dem Pfad)

Kostenfunktion für Knoten $k : V \rightarrow \mathbb{R}_{\geq 0}$

$k(u)$ = minimale (bisher entdeckte) Pfadkosten vom Startknoten zu u

Datenstruktur zur Verwaltung von L_a : Priority Queue

Priorität eines Knotens u : $k(u)$

prominente Spezialfälle:

- ▶ Breitensuche (Kosten = Tiefe des aktuellen Knotens u)
- ▶ Dijkstra-Algorithmus zur Bestimmung kürzester Wege (bekannt aus Algorithmen und Datenstrukturen)
Kosten = minimale bisher bekannte Kosten vom Startknoten zum aktuellen Knoten u

Beispiel Schiebefax

- ▶ Zustände $u \in \{0, \dots, 8\}^{3 \times 3}$, 3×3 -Matrix mit Einträgen $\{0, \dots, 8\}$ (jede Zahl genau einmal, 0 bezeichnet Lücke / leeres Feld)
- ▶ Zulässige Züge: Verschieben des leeren Feldes auf ein Nachbarfeld d. h. Vertauschen von 0 und einem Wert in einem Nachbarfeld (gleicher Zeilen- oder Spaltenindex)
- ▶ Zielkonfiguration

1	2	3
8	0	4
7	6	5

- ▶ Aufgabeninstanz: gegebene Ausgangskonfiguration (Matrix), z.B.

8	0	3
2	1	4
7	6	5

- ▶ Lösung: Folge von zulässigen Zügen (Bewegung der Lücke 0) von der Ausgangs- zur Zielkonfiguration
- ▶ Bewertung der Lösung: Anzahl der Züge (Länge der Lösungsfolge)

Heuristische Suche – Motivation

Heuristik: Effizienzsteigerung durch Zusatzinformationen
(z.B. Erfahrungswerte)

Anwendung bei

- ▶ falls vollständige Suche zu aufwendig
- ▶ Aufgaben mit mehreren Lösungen (z.B. Wege in Graphen)
- ▶ unterschiedliche Qualität der Lösungen
(z.B. Länge des Weges)
- ▶ Suche nach **optimalen** Lösungen (z.B. kürzester Weg)

Ziele:

- ▶ Wahl einer geeigneten Such-Reihenfolge, unter welcher gute Lösungen zuerst gefunden werden
- ▶ Verwerfen von Knoten, die wahrscheinlich nicht zu einer Lösung führen
(beabsichtigte Verletzung der Fairness-Eigenschaft)

Schätzfunktionen

Ziel: sinnvolle Auswahl der in jedem Schritt zu expandierenden Knoten unter Verwendung von Zusatzinformationen

Schätzfunktion (heuristische Funktion) $h : V \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$

(oder in eine andere geordnete Menge)

Schätzung der erwarteten Restkosten vom Knoten u bis zum Ziel

h repräsentiert die Zusatz**information**

ohne Zusatzinformation: **uninformierte** (blinde) Suchverfahren, z.B. Tiefensuche, Breitensuche, Dijkstra

mit Zusatzinformation: **informierte** Suchverfahren
heuristische Suche

Schiebefax – Heuristische Funktionen

Heuristische Funktionen $h_i : \{0, \dots, 8\}^{3 \times 3} \rightarrow \mathbb{N}$ mit

- h_1 Anzahl der Zahlen, die sich nicht an ihrer Zielposition befinden
- h_2 weitester Abstand einer Zahl zu seiner Zielposition
- h_3 Summe der Manhattan-Abstände jeder Zahl zu seiner Zielposition

Tafel: Bestensuche mit Bewertungsfunktionen $f(u) = h_i(u)$

Qualität der Schätzfunktionen:

- ▶ gute Trennung verschiedener Zustände
- ▶ fair: zu jedem $n \geq 0$ existieren nur endlich viele $u \in V$ mit $h(u) \leq n$

Eigenschaften von Heuristiken

Schätzfunktion $h : V \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$ heißt

perfekt (Schätzfunktion $H(u)$), gdw. $\forall u \in V : H(u) =$
genau die Kosten einer optimalen Lösung durch u
($H(u) = \infty$, falls keine Lösung über u existiert)

zielerkennend gdw. für jeden Lösungsknoten $u \in V$ gilt $h(u) = 0$

sicher gdw. aus jedem Knoten $u \in V$ mit $h(u) = \infty$ ist
kein Lösungsknoten erreichbar
d.h. $\forall u : (h(u) = \infty \rightarrow H(u) = \infty)$

konsistent gdw. für jeden Knoten $u \in V$ und alle Folgeknoten v
von u gilt $h(u) \leq w(u, v) + h(v)$
($w(u, v)$ Kosten des Übergangs von u nach v)

nicht-überschätzend gdw. für jeden Knoten $u \in V$ gilt
 $h(u) \leq H(u)$

Aus nicht-überschätzend folgt sicher und zielerkennend. (ÜA)

Aus zielerkennend und konsistent folgt nicht-überschätzend. (ÜA)