Was bisher geschah

- Daten, Information, Wissen
- Wissensrepräsentation und -verarbeitung

Wissensrepräsentation: Beschreibung von

Wissen: Zustandsübergangssystem: gerichteter Graph $G = (V, E)$ mit

- Knotenmarkierungen $l_v : V \to L_V$ mit L_V: Eigenschaften der Zustände
- Startzustand $s \in V$
- Eigenschaften der Zielzustände (z.B. Variablenwerte)
- Kantenmarkierungen $l_E : V \to L_E$ mit L_E: mögliche / zulässige Aktionen (Übergänge)

Lösung: zulässiger Weg (Zustandsfolge $p \in V^*$) vom Start- zu einem Zielzustand

Wissensverarbeitung: Pfadsuche im Graphen

- blinde Suchverfahren: Tiefensuche, Breitensuche

Allgemeiner Suchalgorithmus

1. aktuelle Menge der zu untersuchenden Knoten $L_a = \{s\}$
2. aktuelle Menge der erledigten $L_x = \emptyset$
3. solange nicht (gefunden oder $L_a = \emptyset$) wiederhole:
   - 3.1 Verschiebe einen festgelegten Knoten u aus L_a in L_x
   - 3.2 Füge alle Nachbarn von u, die $L_a \cup L_x$ nicht enthält, (auf eine festgelegte Art) in L_a ein

Verschiedene Suchverfahren unterscheiden sich nur in der Auswahl des expandierten (festgelegten) Knotens aus L_a

nach Festlegung durch Datenstruktur zur Verwaltung von L_a

- Stack: Tiefensuche
- Queue: Breitensuche

Schrittweise Vertiefung (ID)

(iterative deepening)

Ziel: Verbindung der Vorteile von

- Tiefensuche (geringer Speicherbedarf)
- Breitensuche (Vollständigkeit)

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

nicht vollständig (Lösungszustände, die mehr als m von der Wurzel entfernt sind, werden nicht gefunden)

2. Idee: schrittweise Vertiefung

Nacheinanderausführung beschränkter Tiefensuchen für alle $m \in \mathbb{N}$ ($<$-geordnet), solange keine Lösung gefunden wurde

Vorteil: vollständig, optimal

Nachteil:

Knoten nahe des Startzustandes werden mehrfach expandiert

aber (asymptotischer) Zeit- und Platzbedarf wie Tiefensuche

Bidirektionale Suche

- simultane Suche ab Startknoten und ab Zielknoten
  Vorwärtssuche mit L_{xs}, L_{as}, Rückwärtssuche mit L_{xg}, L_{ag}
- Lösung (Pfad $p(s, g)$ von Start s zu Ziel g) gefunden, wenn ein Zustand u von s und g erreichbar ist (also in beiden Suchen entdeckt wurde)
  Lösung $p(s, g) = p(s, u) \circ p(g, u)^{-1}$
- Bidirektionale Suche endet, wenn sich die „Grenzen“ der durch die Suche bisher entdeckten Mengen überschneiden
  $((L_{xs} \cup L_{as}) \cap (L_{xg} \cup L_{ag} \neq \emptyset)$

Speicherbedarf geringer als bei Breitensuche

- eindeutiger (gesuchter) Zielzustand muss bekannt sein
  z.B. bei Kannibalen-Missionare-Rätsel, Navigation
- Erweiterung auf endliche Mengen explizit gegebener Zustände möglich (Betrachtung von Zustandsmengen in Suchknoten)
- meist ungeeignet, wenn Zielzustände durch zu erfüllende Bedingung definiert sind
  (z.B. Spiele mit Zielbedingung wie Schach-Matt, kein Zug möglich)
  mehreren Zielzuständen verschiedener Güte

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)

Bewertungsfunktion für Knoten $k : V \to \mathbb{R}_{\geq 0}$

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

Datenstruktur zur Verwaltung von L_a: Priority Queue

Priorität eines Knotens u: $k(u)$

Beispiele:

- I Breitensuche (Kosten = Tiefe des Knotens) I kürzeste Wege (Kosten = Abstand des Knotens vom Startknoten) Dijkstra-Algorithmus

Uniforme Kostensuche ist wie Breitensuche und Tiefensuche ein uninformiertes Suchverfahren

Heuristische Suche – Motivation

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

Anwendung bei

- 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)
- falls vollständige Suche zu aufwendig

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) $h : V \to \mathbb{R}_{\geq 0} \cup \{\infty\}$
(oder in eine andere geordnete Menge)
Schätzung der erwartete Restkosten vom Knoten u bis zum Ziel

repräsentiert die Zusatzinformation

Eigenschaften von Heuristiken

Schätzfunktion $h : V \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ 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. $\forall u : (h(u) = \infty \to H(u) = \infty)$

konsistent gdw. für jeden Knoten $u \in V$ und alle Nachbarn 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.
Aus zielerkennend und konsistent folgt nicht-überschätzend.

Besten-Suche

(best-first-search)

Allgemeines Suchverfahren mit Bewertungsfunktion
$f : V \to \mathbb{R}_{\geq 0} \cup \{\infty\}$
mit folgender Strategie zur Auswahl der in jedem Schritt zu expandierenden Knoten:

- Knoten werden aufsteigend nach Bewertung $f(u)$ expandiert,
- Expansion des Knotens u mit dem geringsten Wert $f(u)$ zuerst
- Verwaltung von L_a als priority queue

Beispiel: Suche eines kürzesten Weges zwischen Orten A und B

- Bewertungsfunktion $f(u)$: bisherige Kosten bis zum Ort u (ohne Schätzfunktion, uniforme Kostensuche, Dijkstra)
- Bewertungsfunktion $f(u)$:
  Luftlinienentfernung des Ortes u von B (nur Schätzfunktion)

Besten-Suche – Eigenschaften

zwei Methoden:

1. Knoten mit großen Werten möglichst spät expandieren
2. Knoten mit großen Werten nicht expandieren

- Bestensuche mit einer beliebigen Besertungsfunktionfunktion ist nicht immer optimal.
- Bestensuche nach Methode 1 (fair) ist vollständig
- Bestensuche nach Methode 2 ist nicht immer vollständig

Greedy-Suche (kleinste Restkosten)

Idee: Suche zuerst in Teilbäumen der noch nicht besuchten Knoten mit den geringsten (geschätzten) noch aufzuwendenden Kosten

Heuristische Funktion $h : V \to \mathbb{R}_{\geq 0} \cup \{\infty\}$

$h(v)$ ist Abschätzung des von Knoten v aus den noch notwendigen Kosten zum Erreichen eines Zielzustandes

Greedy-Suche:
Besten-Suche mit Bewertungsfunktion $f : V \to \mathbb{R}_{\geq 0} \cup \{\infty\}$,
wobei für jeden Knoten $v \in V$ gilt

$$f(v) = h(v)$$

Eigenschaften der Greedy-Suche:

- optimal?
- vollständig?

Beispiel Schiebefax

- Zustände $u \in \{0, \ldots, 8\}^{3\times 3}$, 3×3-Matrix mit Einträgen $\{0, \ldots, 8\}$ (jede Zahl genau einmal, 0 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		4
7	6	5

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

8		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)

Schiebefax – Heuristische Funktionen

Heuristische Funktionen $h_i : \{0, \ldots, 8\}^{3\times 3} \to \mathbb{N}$ 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 $h(u) \leq n$

Bisherige Kosten

Kostenfunktion $k : V \to \mathbb{R}_{\geq 0}$

$k(u)$ Kosten des besten (bisher bekannten) Pfades vom Startzustand zum Zustand u

Kostenfunktion $k : V \to \mathbb{R}_{\geq 0}$ heißt

streng monoton wachsend , falls für alle Knoten v und alle Nachfolger u von v gilt $k(u) < k(v)$

Beispiele für Kostenfunktionen:

- Tiefe des Knotens im Suchbaum,
- maximale Entfernung vom Startknoten

A*-Suche (kleinste Gesamtkosten)

Idee: Suche zuerst in Teilbäumen der noch nicht besuchten Knoten mit dem geringsten Wert der Schätzfunktion
(Summe von bisherigen und geschätzen zukünftigen Kosten)

Funktionen

- $k : V \to \mathbb{R}_{\geq 0}$ – bisher bekannte Kosten von einem Startzustand zu v
- $h : V \to \mathbb{R}_{\geq 0}$ – geschätzte Kosten von v zu einem Endzustand

A*-Suche:

Besten-Suche mit Schätzfunktion $f : V \to \mathbb{R}_{\geq 0}$, wobei für jeden Knoten $v \in V$ gilt

$$f(v) = k(v) + h(v)$$

Eigenschaften der A^*-Suche:

- vollständig?
- optimal?

Anwendungen

Planungsprobleme und kombinatorische Suchprobleme, z.B.

- Routenplanung
- TSP
- Verlegen von Leitungen
- Schaltkreis-Layout
- Navigation (z.B. von Robotern)
- Scheduling
- Produktionsplanung

Reading Group KW 20
Robert C. Holte, 2010: Common Misconceptions Concerning Heuristic Search
`https://aaai.org/ocs/index.php/SOCS/SOCS10/paper/view/2073/2500`
ÜA (zur Information):
Serie 2 aus BA-Modul Grundlagen der Künstlichen Intelligenz