Zustandsraumsuche: Blinde und Heuristische Suche

Einführung in die KI
Übungsstunde am 01.11.04

Benjamin Altmeyer

1

Heute im Angebot

- Was ist „Suche“?
  - Suche als Probemlösung
  - Zustandsraumsuche
  - Vollständigkeit und Korrektheit
- Allgemeines Schema S
  - Expansionstrategien
  - Blinde und Heuristische Suche
  - Tiefen- und Breitensuche

2

Was ist „Suche“?

3

Was ist „Suche“?

- Beispiele für Suche
  - Suche nach Spielzug
  - Suche nach Parametern
  - Suche nach Antwort, nach Beweis
  - Suche nach einem Weg (Graphentheorie)

„Suche ist systematisches Probieren"

- Ziel muss klar sein
  - schnellste Möglichkeit?
  - billigste Möglichkeit?

4

Suchen: Bitte unterscheiden

- Suchen als Wiederfinden:
  - Datenbank, Suchmaschine
  - Suche nach ähnlichen Begriffen
  - Suche nach Inhalten
- Suche als Problemlösung
  - **E**: Existiert eine Lösung?
  - **L**: Finde eine Lösung!
  - **O**: Finde die beste Lösung!

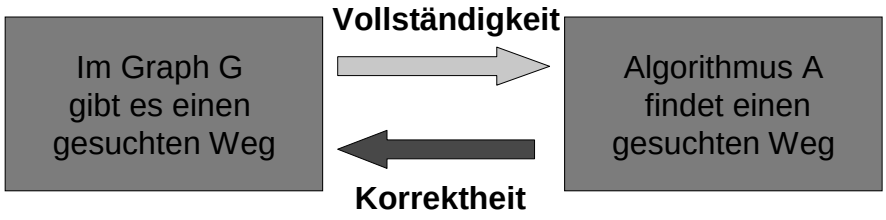
5

Zustandsraum

- Graphen, Bäume (Z, Op, z_{inital}, Z_{final}) mit
  - Zuständen Z
  - Operatoren Op $\subseteq$ ZxZ (Zustandsübergänge), evtl. mit Kostenfunktion
  - einem Anfangszustand z_{inital} ϵ Z
  - Zielzuständen $Z_{final} \subseteq Z$

6

Vollständigkeit und Korrektheit



- Alle folgenden Verfahren sind korrekt
- nicht alle sind vollständig

7

Allgemeines Schema S

Suchverfahren
ohne Kostenfunktion

8

Blinde und Heuristische Suche

- Blinde Suche
  - Keine zusätzlichen Informationen
  - Beispiele: Tiefensuche, Breitensuche, iterat. Tiefensuche
- Heuristische Suche
  - Wissen über die Zukunft
  - Abschätzung des Restweges
  - Beispiele: Hill climbing, beam search, best first (Greedy Search)

9

Allgemeines Schema S

- Arbeitet mit zwei Listen:
  - OPEN: Enthält bekannte Zustände, deren Nachfolger noch nicht berechnet wurden (d.h. nicht expandiert wurden)
  - CLOSE: Zustände, die vollständig expandiert sind
- Unterschiede bei den Verfahren vor allem
  - Bei der Reduzierung der zu betrachtenden Nachfolger (V1)
  - Bei der Reihenfolge der Nachfolger (V2)

10

Allgemeines Schema S

S	Kurzbeschr.	Beschreibung
S0	Start	Falls Anfangszustand z_0 ein Zielzustand: EXIT („yes:“ z_0) Sonst OPEN := [z_0], CLOSED := [ ]
S1	neg. Abbruch-bedingung	Falls OPEN = []: EXIT („no“)
S2	expandieren	Sei z der erste Zustand aus OPEN OPEN := OPEN – {z} CLOSED := CLOSED ∪ {z} Bilde die Menge Succ(z) der Nachfolger von z Falls Succ(z) = {}: Goto S1
S3	pos. Abbruch-bedingung	Falls ein Zustand z_1 aus Succ(z) ein Zielknoten ist: EXIT („yes:“ z_1)
S4	Organisation von OPEN	Reduziere[1] Menge Succ(z) zu Menge NEW(z) durch Streichen nicht weiter zu betrachtenden Zustände. Bilde neue Liste[2] OPEN durch einfügen der Elemente aus NEW(z) Goto S1.

Heuristische Suche

- Schätzfunktion σ(z): geschätzter Konstruktionsaufwand für Erreichen eines Zielzustandes von z aus. (dabei σ(z) = 0 für Zielzustände z)
- Beschriftung der Knoten, nicht Beschriftung der Kanten!! **(keine Kostenfunktion!)**
- Heuristik: Zielzustände mit optimaler Schätzung bevorzugen
- Trade-Off bzgl. Aufwand zur Berechnung der Schätzung

12

Blind: Breiten- vs. Tiefensuche

	Tiefensuche	**Breitensuche**
V2: Einfügen von NEW(z) in OPEN	NEW(z) an den Anfang von OPEN	NEW(z) an das Ende von OPEN
Implementation	Keller	Warteschlange
Speicheraufwand OPEN	linear d*b (b: fan-out, d: Tiefe)	exponentiell b^d (b: fan-out, d: Tiefe)
Vorteile	• geringer Speicherbedarf • Hält sich gerade bei schwierigen Problemen nicht in oberen Ebenen auf	• Findet immer kürzeste Lösung • kein Zyklenproblem (doppelte Zustände können nicht optimal sein)
Nachteile	• kann sich im Graph verrennen • man weiss nie, ob längeres suchen bessere Lösung finden würde	• hoher Speicherbedarf, oft nicht durchführbar

13

Mit und ohne Test auf Wiederholung

	Mit Test auf Wiederholung	**Ohne Test auf Wiederholung**
V1: Reduzierung von oben	NEW(z) = Succ(z) - (OPEN∪CLOSED)	NEW(z) = Succ(z)
Korrekt (endl. Graph.)	ja	ja
Vollständig (endl. Gr.)	ja (findet Lösung im Fall der Existenz)	Tiefensuche: nicht immer Breitensuche: ja
Nachteil	hoher Speicheraufwand für CLOSED (evtl. gesamter Graph), da immer nur neue Zustände	hoher Zeitaufwand bei Zyklen/Maschen
Bermerkung		Statt Graph wird also „abgewickelter Baum“ untersucht

14

Spezielle Algorithmen

- Backtracking
  - Tiefensuche mit „Schichten“
  - Nach Abarbeiten aller Zustände einer Schicht backtracking auf davor liegende Schicht
- Iterative Tiefensuche:
  „**Depth-first-iterative deepening (DFID)“**
  - Stufe 1: begrenzte Tiefensuche bis zur Tiefe 1
  - Stufe 2: begrenzte Tiefensuche bis zur Tiefe 2 usw.
  - **DFID hat Speicherbedarf für OPEN wie Tiefensuche**
  - **DFID findet Lösung wie Breitensuche (kürzeste Lösung..)**

15

Heuristische Verfahren

(alles ohne Test)	**Bergsteigen / “hill climbing“**	**Strahlensuche**	**Bestensuche / “Greedy Search“**
V1: Reduzierung von NEW	NEW(z) = Succ(z)	NEW(z) = „Gute“ Auswahl aus Succ(z)	NEW(z) = Succ(z)
V2: Aufbau von OPEN	NEW(z) nach Aufwand sortiert an Anfang von OPEN	NEW(z) nach Aufwand sortiert an Ende von OPEN	OPEN∪NEW(z) nach Aufwand sortiert
Korrekt (endl. Gr.)	Ja	ja	ja
Vollständig (endl. Gr.)	nicht immer (ähnlich Tiefensuche)	nicht immer (einge-schränkte Breitensuche)	nicht immer
Bemerkung	„Lokale Optimierung“		

16

Zusammenfassung

	Vollständigkeit	Korrektheit	Komplexität (Schrittzahl!)	Speicher-platz
Breitensuche	ja	ja	O(b^d)	b^d
Tiefensuche	nein	ja	O(b^d)	b*d
it. Tiefensuche	ja	ja	O(b^d)	b*d
Hill Climbing	nein (Vorgebirgsgipfel) ja: mit Gedächtnis der schon besuchten	ja	O(b^d)	b*d
Beamsearch	nein (ähnlich wie HC)	ja	O(d) (wenn nur ein Strahl)	Exist.: 1 Konstr.: d
Bestensuche	ja (VL:nicht immer)	ja	O(b^d)	b^d
Schema S allg	abh. vom Verfahren	ja	O(b^d)	abh. vom Verfahren

17

Fragen?

Danke für eure Aufmerksamkeit.

Benjamin Altmeyer

18