Kapitel 2: Suche

Teil 2

(Dieser Foliensatz basiert auf Material von Mirjam Minor, Humboldt-Universität Berlin, WS 2000/01)

Eigenschaften und Verbesserungen von A*

- vollständig, optimal, Speicherbedarf schlimmstenfalls $O(b^d)$, Zeitbedarf im allgemeinen (z.B. bei $h(n) = 0$ für alle n) auch $O(b^d)$
- Bessere Eigenschaften lassen sich für den allgemeinen Fall nur bei sehr hohen Anforderungen an h zeigen, die häufig nicht zu erfüllen sind. Für ein konkretes Suchproblem und eine gute Heuristik wird A* oft wesentlich besser sein.
- Iterativ Deepening A* (IDA*) führt eine Iterative Tiefensuche durch, wobei als Abbruchkriterium gilt: $g(n) + h(n) > C$. Die Schranke C wird bei jedem Durchlauf erhöht. Speicherbedarf $O(n)$ bei leicht erhöhtem Zeitaufwand. Pro Durchlauf werden nicht mehr Knoten als durch A* expandiert. Das Verfahren ist ebenfalls vollständig und optimal.

Problemzerlegung

Ein weiteres Modell zur Formulierung von Suchproblemen ist die Problemzerlegung. Sie arbeitet auf Mengen von (Teil-)Problemen. Der Übergang zwischen zwei Problemmengen kann auf unterschiedliche Weisen erfolgen:

- Die **Zerlegung in Teilprobleme**, von denen alle zu lösen sind.
- Die **Auswahl von Alternativen**, von denen eine zu lösen ist.

Teilprobleme, die sich nicht weiter zerlegen lassen, sind entweder direkt lösbar (**primitiv / terminal**) oder unlösbar (**nicht-terminal**).

Interpretation als Und-Oder-Baum bzw. -Graph

- Anfangsknoten: stellt das Ausgangsproblem dar
- Und-Verzweigung: repräsentiert Problemzerlegung
- Oder-Verzweigung: repräsentiert Alternativen
- Knoten ohne Nachfolger (Endknoten): entweder primitive oder unlösbare Probleme

Häufig wird die Normierung vorgenommen, daß sich die Verzweigungsarten abwechseln müssen.

Lösung eines Problems im Und-Oder-Graphen

Ein zyklenfreier, endlicher Teilgraph, der folgende Bedingungen erfüllt:

1. der Anfangsknoten ist enthalten
2. alle Endknoten sind primitiv
3. für alle anderen Knoten des Lösungsgraphen gilt:
   - bei einer Und-Verzweigung sind alle Nachfolger enthalten
   - bei einer Oder-Verzweigung ist genau ein Nachfolger enthalten

Beispiele für Problemzerlegungen

Prolog:
Teilprobleme: Subgoals einer Klausel
Problemzerlegung: Anwendung einer Klausel
Alternativen: Klauseln einer Prozedur
primitive Probleme: Fakten

symbolische Integration:
Teilprobleme: Teilintegrale
Problemzerlegung: Teilintegrationen (Summen, Partielle Integration, Substitution) und Umformungen
Alternativen: unterschiedliche Zerlegungen
primitive Probleme: bekannte Standardformen (aus Tabellen)
Lösung ist nicht nur true/false, sondern Ableitung bzw. Vorgehensweise bei Integration

– SAINT - Symbolic Automatic INTegrator, Slagle 1963

– Dauer 11 Min (Rechner von Anfang der 60er Jahre; LISP-Interpreter); heute bei gleichem Programm im Sekundenbereich

– Vorgehen:
1. Integrale mittels Tabelle von Standardformen lösen
2. algorithmenartige Umformungen versuchen (Faktoren herausziehen; in Summe zerlegen)
3. heuristische Umformungen (Umformung des Integranten, Substitution, partielle Intgration) – Analyse des Integranten auf bestimmte Eigenschaften (Zuhilfenahme problemspezifischen Wissens) – z.B. Schwierigkeitsabschätzung mittels maximaler Tiefe der Funktionsverschachtelung

Beispiele für Problemzerlegungen

Spiele:
Teilprobleme: mögliche Spielzustände
Problemzerlegung: gegnerische Züge
Alternativen: eigene Züge
primitive Probleme: Endzustände des Spiels mit Gewinn

Zauberwürfel:
Teilprobleme: zu erledigende Aufgaben
Problemzerlegung: 1.,2.,3. Schicht ordnen; Ecken ordnen, Kanten ordnen
Alternativen: Auswahl bestimmter Schichten und Steine
primitive Probleme: durch bekannte Zugfolge zu lösen

Umformung zwischen Zustandsraumbeschreibung und Problemzerlegung

$\Rightarrow$ Und-Knoten auf den Kanten einfügen (trivial)

$\Leftarrow$

- Zustände: Menge offener Teilprobleme
- Operatoren: Ersetzung eines Problems in der Menge durch seine Teilprobleme (Problemzerlegung);
  Alternative Zerlegungen als Verzweigung;
  Primitive Probleme werden aus der Menge gestrichen
- Anfangszustand: Menge mit Ausgangsproblem
- Zielzustand: Leere Menge

Modellierung der "Türme von Hanoi"

t_i sind Türme; s_i sind Scheiben; Die Scheiben liegen geordnet auf dem ersten Turm und sollen geordnet auf den letzten Turm gebracht werden. Es darf nie eine größere auf einer kleineren Scheibe liegen.

- als Zustandsraum

  Zustände: alle Funktionen $f : \{s_1, ..., s_n\} \rightarrow \{t_1, t_2, t_3\}$ (3^n Zustände)
  Operatoren: $t_i \rightarrow t_j$ für $i, j \in \{1, 2, 3\}$ und $i \neq j$ – oberste Scheibe von t_i nach t_j legen (pro Zustand max. 3 Operatoren anwendbar)
  Anfangszustand: $f(s_i) = t_1$ für $i = 1, \ldots, n$
  Endzustand: $f(s_i) = t_3$ für $i = 1, \ldots, n$

- als Problemzerlegung

  Ausgangsproblem: n Scheiben von t_1 nach t_3
  Problemzerlegung: m Scheiben von t_i nach t_j
  – $(m-1)$ Scheiben von t_i nach t_k
  – 1 Scheibe von t_i nach t_j
  – $(m-1)$ Scheiben von t_k nach t_j
  Primitive Probleme: 1 Scheibe von t_i nach t_j

– Zustandsraumbeschreibung: Funktion ordnet den Scheiben ihre Plätze zu
von 6 möglichen Operatoren max 3 anwendbar, wegen Bedingung, daß nur kleine auf großen Scheiben liegen dürfen

– Problemzerlegung: keine Alternativen -> Lösungsbaum wird direkt konstruiert
– Eine bessere Formulierung des Problems würde den Aufwand verringern

Bewertung der Knoten

lösbare Knoten

- terminale Knoten
- Knoten mit Und-Verzweigungen: alle Nachfolger lösbar
- Knoten mit Oder-Verzeigungen: mindestens ein Nachfolger lösbar

unlösbare Knoten

- nicht-terminale Knoten
- Knoten mit Und-Verzweigungen: mindestens ein Nachfolger unlösbar
- Knoten mit Oder-Verzeigungen: alle Nachfolger unlösbar

Für endliche Bäume ergibt sich bei Festlegung für die Bewertung der Endknoten eine eindeutige Zerlegung in lösbare und unlösbare Knoten.

Bottom-Up Entscheidungsverfahren

(0) LÖSBAR := terminale Knoten
UNLÖSBAR := nicht-terminale Knoten

(1) wähle einen Knoten k dessen Nachfolger alle in LÖSBAR $\cup$ UNLÖSBAR sind

(2) falls bei k Und-Verzweigung und alle Nachfolger in LÖSBAR oder
falls bei k Oder-Verzweigung und ein Nachfolger in LÖSBAR:
LÖSBAR := LÖSBAR $\cup \{k\}$,
sonst UNLÖSBAR := UNLÖSBAR $\cup \{k\}$

(3) Falls Startknoten in LÖSBAR: EXIT(yes)
Falls Startknoten in UNLÖSBAR: EXIT(no)
GOTO 1

Top-Down Verfahren zur Lösungsbaumsuche (1)

(0) OPEN := $\{Startknoten\}$, CLOSED := {}
Falls $Startknoten$ terminal: EXIT(yes)

(1) Sei k der erste Knoten aus OPEN
OPEN := OPEN - $\{k\}$, CLOSED := CLOSED $\cup$ $\{k\}$, SUCC(k) := Nachfolger von k
SUCC(k) an den Anfang von OPEN (Tiefensuche) bzw.
SUCC(k) an das Ende von OPEN (Breitensuche)

(2) Falls SUCC(k) = {}: GOTO (6) /* Markiere k als unlösbar */

(3) Falls in SUCC(k) keine terminalen Knoten sind: GOTO(1)

(4) Markiere die terminalen Knoten in SUCC(k) als lösbar
Falls dadurch k und weitere Vorgänger von k lösbar werden, so markiere auch diese als lösbar

Top-Down Verfahren zur Lösungsbaumsuche (2)

(5) Falls $Startknoten$ als lösbar markiert: EXIT(yes)
Entferne alle Knoten aus OPEN, die einen mit lösbar markierten Vorgänger besitzen
GOTO (1)

(6) Markiere k als unlösbar. Falls dadurch weitere Vorgänger von k unlösbar werden, so markiere auch diese als unlösbar.

(7) Falls $Startknoten$ als unlösbar markiert: EXIT(no)
Entferne alle Knoten aus OPEN, die einen mit unlösbar markierten Vorgänger besitzen
GOTO (1)

Suche in Spielbäumen

Wir betrachten ein 2-Personenspiel. Die Spieler A und B ziehen abwechselnd. Am Ende wird der Gewinn bzw. Verlust von A und B numerisch bewertet.

Beispiele für Bewertungen

– Schach: 1/-1 (A gewinnt), 0/0 (Remis), -1/1 (B gewinnt)

– Backgammon: Werte zwischen -192 und +192

– nicht unbedingt antagonistisch: 10/2 3/-8 4/6 (Absprachen, Koalitionen möglich)

– Gefangenendilemma: 2/2 (beide leugnen) 5/5 (beide gestehen) 0/10 bzw. 10/0 (einer von beiden gesteht)

Warum Spiele?

– Abstrakter Wettbewerb, gut repräsentierbar, oft vollständige Informationen, klare Regeln, begrenzte Anzahl von Zügen/Handlungsweisen

– trotzdem komplex / nicht überschaubar

Erste Schachprogramme Anfang der 50er Jahre (Shannon/Turing):

– Beweis(?) für die Intelligenz(?) eines Programms: "Es kann Schach spielen!"

Suchproblem: günstige Spielzüge finden

Notwendige Komponenten für die Suche

- Anfangszustand
- Operatoren, die die zulässigen Züge beschreiben
- Test ob Endzustand erreicht
- Resultatfunktion (payoff) für die Endknoten, die den Gewinn bzw. Verlust von A und B als numerischen Wert wiedergibt

Darstellung als Baum bzw. Graph möglich

Probleme beim Suchen nach einer Strategie

- Unsicherheit bezüglich gegnerischer Aktionen
- Unüberschaubarkeit durch Komplexität
  (Beispiel Schach: durchschnittlich 35 mögliche Züge, pro Partie ca. 50 Züge (Halbzüge) pro Spieler $\Rightarrow 35^{100}$ Knoten (bei ca. 10^{40} unterschiedlichen zulässigen Stellungen))
- begrenzte Zeit für Entscheidungsfindung
- Zufallselemente

Vereinbarungen für die folgenden Verfahren

Es wird verlangt:
– vollständige Informationen
– Nullsummenspiel (resultat(A) = -resulat(B)) → Angabe für A reicht
– Spielbaum aus der Sicht von A

Ausgangspunkt: A (**Maximierer**) will maximal mögliches Ergebnis erzielen, B (**Minimierer**) wird versuchen, das Ergebnis von A möglichst gering zu halten (optimale Spielweise von B vorausgesetzt).

Ziel: Strategie, die A in jeder Situation sagt, welcher nächste Zug der beste ist, unter Berücksichtigung aller möglichen Züge von B.

Interpretation als Und-Oder-Baum

Zug von A: Oder-Verzweigung (Alternativen)
Zug von B: Und-Verzweigung (man will allen Reaktionen von B begegnen)

Bei Spielen, die nur die Bewertung 1,-1 (Gewinn,Verlust) haben, Lösungssuche analog zur Problemzerlegung:
lösbare Knoten: A gewinnt
unlösbare Knoten: B gewinnt
Strategie: im Lösungsbaum Zug zu einem lösbaren Knoten wählen

Jedes Spiel mit endlicher Baumstruktur und nur Gewinn/Verlust besitzt entweder eine Gewinnstrategie für A oder eine Gewinnstrategie für B. (Bsp.: Tic-Tac-Toe)

Die Minimax-Strategie

Voraussetzung: vollständig entwickelter Baum

S0 Endknoten mit resultat(A) bewerten

S1 Wähle einen Knoten k dessen Nachfolger alle bewertet sind
- Wenn bei k eine Und-Verzweigung beginnt, erhält k die minimale Bewertung seiner Nachfolger
- Wenn bei k eine Oder-Verzweigung beginnt, erhält k die maximale Bewertung seiner Nachfolger

S2 Falls Wurzel bewertet: Stop
GOTO S1

Strategie: A wählt in k den Zug zum maximal bewerteten Nachfolger.

Komplexität von Minimax

Zeitbedarf für vollständige Expansion des Spielbaumes: $O(b^m)$

Möglichkeiten zum Einsparen:

- Die Expansion des Spielbaumes wird vorzeitig abgebrochen; die Qualität des erreichten Zustandes wird abgeschätzt.

- Teilbäume, die Aufgrund der bisher ermittelten Bewertungen ohnehin nicht in Frage kommen, werden nicht entwickelt.

Eine Kombination beider Verfahren ist möglich.

Entscheidung ohne vollständige Entwicklung des Baumes

Folgende Ersetzungen bei den Komponenten werden notwendig:

- Test auf Endzustand $\Rightarrow$
  Test ob Abbruch der Suche sinnvoll (cutoff)
- Resultatfunktion $\Rightarrow$
  heuristische Bewertung (utility) der Nützlichkeit/Qualität beliebiger Spielzustände

Bewertungen werden wie bisher per Minimax nach oben propagiert.

Wahl der Heuristik

Sie soll in den Endzuständen in ihrer Wertung mit der Resultatfunktion übereinstimmen, sie soll die Gewinnchancen von einem Zustand aus widerspiegeln und sie darf nicht zu aufwendig sein.

Häufig werden gewichtete lineare Funktionen verwendet: Es müssen also Kriterien und ihr Einfluß auf die Nützlichkeit der Stellung bestimmt werden. Alternativen: z.B. Neuronale Netze

Beispiel Schach:
Materialwerte Bauer (1), Springer und Läufer (3), Turm (5), Dame (9); gute Verteilung der Bauern und Sicherheit des Königs (0,5); Werte für beide Spieler aufsummieren und gegeneinander verrechnen.

Die Heuristik hilft alleine noch nicht viel, Beispiel Schach:

Annahmen:

- man kann pro Sekunde 1000 Stellungen durchsuchen (1995);
- im Turnierschach 150s Zeit zum Überlegen
- durchschnittlicher Verzweigungsgrad 35

Ergebnis:

- Suchbaum hat Tiefe von 3 (42875 Stellungen) bis 4 (ca. 1,5 Mio Stellungen) Halbzügen

- entspricht der Spielstärke eines Anfängers (durchschnittl. menschliche Schachspieler: 6–8 Halbzüge)

Probleme bei unvollständiger Suche

- Statische Bewertung: Verdichtung aller Bewertungskriterien auf eine Zahl
- Horizont-Effekt: entscheidender Spielzug wird wegen Abbruch der Suche nicht betrachtet

Mögliche Lösungen

- Einsatz von Lernverfahren um gute Gewichtung zu finden
- dynamisches Abbruchkriterium: in unruhigen Situationen (starke Schwankungen in der Bewertung) weiter suchen, bei eindeutig schlechten Situationen vorher abbrechen

Spiele mit Zufallselementen

A und B werten in jedem Schritt zunächst ein Zufallsereignis (z.B. Würfeln bei Backgammon) aus. Das Ergebnis beeinflusst die Zugmöglichkeiten. Im Spielbaum werden zusätzlich Zufallsknoten eingefügt.

Sei c ein Zufallsknoten. Mit Wahrscheinlichkeit $P(d_i)$ tritt das Resultat d_i ein, welches zum Zustand s_i führt. Der Knoten c wird bewertet mit $value(c) = \sum_i P(d_i) value(s_i)$. Die Bewertung der A-(Maximierer) bzw. B-Knoten (Minimierer) erfolgt wie bisher.

Kompexität $O(b^d n^d)$ – dabei ist n Anzahl der möglichen unterschiedlichen Zufallsergebnisse.

Literatur

[1] G. Görz, C.-R. Rollinger, J. Schneeberger (Hrsg.): *Handbuch der Künstlichen Intelligenz.* 3., vollständig überarbeitete Auflage, Oldenbourg, 2000

[2] S. Russel, P. Norvig: *Artificial Intelligence - A Modern Approach.* Prentice-Hall, 1995.

[3] P.H. Winston: *Künstliche Intelligenz.* Addison-Wesley, 1987.

[4] E. Rich, K. Knight: *Artificial Intelligence.* 2. Auflage, McGraw Hill, 1991.

[5] H. Kaindl: *Problemlösen durch heuristische Suche in der Artificial Intelligence.* Springer, 1989.

[6] H.S.M. Coxeter, M. Emmer, R. Penrose, M.L. Teuber (Hrsg.): *M.C. Escher: Art and Science.* North-Holland, 1986.