METHODEN DER KÜNSTLICHEN INTELLIGENZ

Ipke Wachsmuth, Universität Bielefeld

Kurztext zu Teil 2: Problemlösen und Suche

Dieser Abschnitt befasst sich mit der Frage: *Wie lassen sich Probleme durch (geschickte) Suche lösen?* In vielen Fällen hat das Lösen von Problemen mit dem Durchsuchen des "Raums der Möglichkeiten" zu tun. Wir stellen uns dazu vor, dass in der Wissensbasis eines intelligenten Agenten ein Problem durch die Beschreibung eines Anfangs- (Start-) und eines Zielzustandes – in einer geeigneten symbolischen Darstellung – gegeben ist, dass der Agent dazu über Regeln oder Operatoren verfügt, mit denen die Problembeschreibung manipuliert werden kann, und ferner über eine Kontrollstrategie, mit der er Entscheidungen über die Auswahl von Operatoren zur schrittweisen Lösung des Problems trifft und das Erreichen des Zielzustandes feststellen kann. In dem folgenden Beispiel

SEND	die Buchstaben sind so durch Ziffern zu ersetzen
+MORE	(gleiche Buchstaben durch gleiche Ziffern),
MONEY	dass eine aufgehende Addition entsteht!

hieße das, dass Operatoren vom Typ `assert:<letter> has_value <digit>` anzuwenden sind und dass der aktuelle Problemlösezustand durch eine Liste solcher Ausdrücke beschrieben ist. Ein Zielzustand ist erreicht, wenn alle Buchstaben durch Ziffern ersetzt sind und eine Funktion `goal_test` feststellt, dass die Spaltensummen (mit Überträgen) passen. Selbst wenn dieses Problem einfach scheint, umfasst der Raum der Möglichkeiten bei diesem Beispiel 40320 (*8!*) Zustände. Die Techniken *heuristischer Suche* befassen sich damit, wie sich geschickt vermeiden lässt, sämtliche möglichen Zustände zu generieren und zu testen.

Formal sind Suchprobleme gegeben durch die Beschreibung eines Anfangszustandes und eines Zielzustandes sowie einer Menge von Operatoren, die jeweils einen Zustand in einen anderen überführen. Problemlösen heißt dann die Suche nach einem Pfad: einer Folge von Operatoren, deren schrittweise Anwendung auf die Problemrepräsentation einen Ausdruck erzeugt, der die Problemlösung darstellt. Zur Terminologie für Suchprobleme gehört die Definition des unmittelbaren Nachfolgers eines Zustandes *S*; dies ist ein Zustand *S'*, der von *S* aus durch Anwendung gerade eines Operators erreicht wird. Entsprechend ist *S'* ein Nachfolger von *S*, wenn er von *S* aus durch eine Folge von Operatoranwendungen erreichbar ist. Der Suchraum ist dann gegeben mit allen durch Operatoranwendungen vom Anfangszustand erreichbaren Zuständen. Dies legt die Darstellung von Suchräumen durch Bäume nahe, mit dem Startzustand als Wurzelknoten; die Knoten für Nachfolgerzustände werden durch Kanten mit ihren Vorgängerknoten verbunden, und die verschiedenen Kanten, die von einem Knoten weiterführen, stehen für die alternativ möglichen Operatoranwendungen (siehe Abb. 2, links).

Es ist eher nebensächlich, ob unter der Lösung eines Suchproblems ein Pfad vom Anfangs- zum Zielzustand oder aber ein konkreter Zustand verstanden wird, der der Beschreibung eines gewünschten Zielzustands genügt. Dagegen kann es unterschiedliche Anforderungen geben, ob es nämlich darum geht, irgendeine Lösung oder alle Lösungen oder eine nach einem gegebenen Kriterium "optimale" Lösung zu finden oder aber zu bestätigen, dass keine Lösung existiert. Vor allem kommt es aber darauf an, nur gerade soviele der aus der formalen

Problembeschreibung implizit resultierenden Zustände des Suchraums explizit zu machen, dass ein Pfad vom Anfangs- zum Zielzustand enthalten ist; hierin unterscheidet sich "blinde" von "geschickter" Suche.

Zu den Basistechniken gehören zunächst die Verfahren der erschöpfenden Suche, bei der ein Suchbaum systematisch in der Tiefe (Tiefensuche) oder Breite (Breitensuche) entfaltet wird. Bei der speichereffizienten Tiefensuche wird der Suchbaum in seinen Ästen bis zu den Blättern (Zustände, von denen kein Operator weiterführt) durchsucht und im Nichterfolgsfall durch Rücksetzen (backtracking) von der letzten Verzweigung aus weiterverfolgt; es besteht aber die Gefahr des Verlierens in unendlicher Tiefe aufgrund zyklisch möglicher Operatoranwendung (Abhilfe bietet das Abbrechen bei vorbestimmter Tiefe: *depth-cutoff* und iterativer Vertiefung der Suche nach Exploration anderer Äste). Bei der Breitensuche werden alle Knoten einer Ebene im Baum vor den Knoten der nächsten Ebene entfaltet; dadurch findet man alle – insbesondere optimale – Lösungen, jedoch ist sie speicherintensiv, zumal die Zahl der Knoten exponentiell mit der Baumtiefe steigen kann (geeignet bei kleinen Suchräumen).

Heuristische Suche

Eine ganze Reihe von Problemen gilt (bei deterministischem Vorgehen) aufgrund einer exponentiellen oder gar kombinatorischen Explosion vom Rechenaufwand her als *intraktabel*. Hierzu gehört das Problem des Handlungsreisenden ("Traveling Salesman"), der auf einer Rundreise *n* Städte aufsuchen soll, wobei ein bestimmter Faktor zu minimieren ist, z.B. die Weglänge. Ein optimaler Weg wird nur dann gefunden, wenn jeder mögliche Weg generiert und derjenige mit geringster Länge ausgewählt wird; das ist ein sogenannter *worst case*. Mit Glück kann bei "blinder" Suche eine Lösung schneller als im *worst case* gefunden werden, jedoch kann das "Glück" nicht garantiert werden, aber doch vielleicht forciert: Die Grundidee von KI-Ansätzen für sog. heuristische Suche liegt darin, dass durch geschicktes RATEN des jeweils als nächstes anzuwendenden Operators sich die Suche in vielen Fällen einschränken lässt; so wird Suchtheorie auch oft als Theorie des geschickten Ratens bezeichnet.

Dazu braucht man allerdings Hinweise, welches jeweils ein guter nächster anzuwendender Operator ist. Zu den Basistechniken gehören hier zwei Typen von Heuristiken: 1. *Operatorordnungsfunktionen*, das sind Algorithmen, welche die in jedem Zustand anwendbaren Operatoren der "Güte" nach ordnen, und 2. *Zustandsbewertungsfunktionen*, die die geschätzte Distanz jedes Zustands vom nächsten Zielzustand (als Zahl) ausgeben. In beiden Fällen lässt sich die Akkuratheit der Funktion nicht garantieren. Sie sollte jedoch "geschicktes Raten" ermöglichen, das besser ist als der Zufall. Ein oft verwendetes heuristisches Verfahren ist die "Bestensuche" mit dem *A*-Algorithmus*, die unter bestimmten Bedingungen (nämlich der Unterschätzung der Restlänge zum Ziel) das Finden optimaler Lösungen garantiert; das erforderliche Geschick bei der Suche besteht hier darin, eine geeignete Schätzfunktion zu finden.

Eine andere wichtige Heuristik ist die im "General Problem Solver"-Programm von Newell, Simon und Shaw verwendete *means-ends-analysis* (Mittel-Ziel-Analyse); hier ist die Frage, wie sich die Anwendung von Operatoren "vorplanen" lässt, auch wenn er im aktuellen Zustand (noch) nicht anwendbar sind. Dazu wird jeweils der aktuell explorierte Zustand mit der Zielzustandsbeschreibung abgeglichen, um eine *Differenz* festzustellen; sie dient als grobes Maß für den Abstand zum Ziel und wird dazu benutzt, erfolgversprechende Operatoren zu raten. Jeder Operator wird dazu in drei Komponenten beschrieben (Vorbedingungen, Überführungsfunktion, Differenzen, die er reduziert); Grund der Auswahl eines nächsten Operators kann sein, dass dadurch ein anderer Operator anwendbar wird.

Abb. 2 Veranschaulichung von Suchbaum (links) und Goal Tree (rechts)

Ein weiterer Schritt, die Suche geschickter zu machen, ist es, das "blinde" mehrfache Explorieren schon "besuchter" Knoten oder Teilbäume zu vermeiden. Grundsätzlich kann es passieren, dass bei einer Suche verschiedene Operatorsequenzen zum gleichen Zustand im Suchraum führen. Zum Beispiel kann bei Tiefensuche mit backtracking die Häufigkeit, mit der ein Knoten besucht wird, exponentiell mit der Baumtiefe wachsen. Um wiederholtes Besuchen von Zuständen im Suchraum zu vermeiden, lassen sich sog. *Diskrimininationsnetze* einsetzen, die einem botanischen Bestimmungsschlüssel ähneln. Dazu wird während der Suche simultan noch ein weiterer Baum aufgebaut, mit dem explorierte Zustände indexiert werden:
(1) in den Blättern werden generierte Zustände eingetragen
(2) in nicht-terminalen Knoten werden Unterscheidungs*gesichtspunkte* (Fragen) eingetragen
(3) die Kanten werden mit den Unterscheidungs*merkmalen* (Antworten) gelabelt
(4) kommt ein neuer Zustand hinzu, der sich von einem Blattknotenzustand in einem Merkmal unterscheidet, wird der Baum gemäß (2) und (3) erweitert. Während der Suche wird für jeden neu generierten Zustand geprüft, ob er schon indexiert ist und, falls ja, von diesem Zustand aus nicht erneut der Suchraum exploriert. So lässt sich der Aufwand für die potenziell exponentielle Suche durch $O(\log n)$ beschränken (n = Anzahl der indexierten Zustände).

***Goal-tree*-Suche**

Bislang sind wir durchgehend davon ausgegangen, dass die Suche nach einer Problemlösung als *Vorwärtssuche*, d.h. ausgehend vom Ausgangszustand auf einen Zielzustand zu erfolgt, wobei das Geschick darin besteht, "auf das Ziel zuzuhalten", und zwar durch den Einsatz von Heuristiken. In vielen Fällen ist es jedoch geschickt, vom Ziel ausgehend in *Rückwärtssuche* einen Lösungspfad zu konstruieren. Das Problem der Zielfindung wird hierbei auf Unter- oder Teilziele zurückgeführt (Problemreduktion), und wenn die fortgesetzte Reduktion des Problems auf ein Teilziel führt, die mit der Beschreibung eines Problems vorgegeben ist, ist man fertig. Jedoch kann es vorkommen, dass bei der Problemreduktion *mehrere* Teilziele entstehen, die in Konjunktion konsistent zu lösen sind.

Zum Zweck der Beschreibung von Suchräumen durch Bäume benutzt man hier zunächst außer den "oder"-Kanten für die alternativ *–rückwärts–* möglichen Operatoranwendungen, welche das Problem auf jeweils ein Unterziel reduzieren, zusätzlich noch "und"-Kanten, nämlich dort, wo Unterziele aufgeworfen werden, die in Konjunktion zu lösen sind: sog. UND-ODER-Bäume. Mit Hilfe von sog. *constraints* können Randbedingungen an die Lösung eines UND-Knotens formuliert werden, die von den in Konjunktion zusammenzusetzenden Teillösungen seiner Unterziele gemeinsam erfüllt werden müssen. Zum Beispiel könnte damit überwacht werden, dass die Summe der Kosten der Teillösungen konjunktiver Knoten durch einen Wert beschränkt ist, oder dass bei einem Kartenfärbungsproblem benachbarte Länder verschieden gefärbt sind; etc. Mit derartigen Zielbäumen (goal trees; Abb. 2 rechts) wird ein impliziter Lösungsraum aufgespannt, der sich – auf dem Umweg über Pläne für Teillösungen – auch wieder als Alternativenraum im üblichen Stil eines Suchbaums rekonstruieren lässt.

Constraint Satisfaction, Planen

Ein spezieller Problemtyp sind *Constraint Satisfaction*-Probleme: Hier sind die Zustände des Suchraums durch Werte einer Menge von Variablen definiert; der *goal-test* spezifiziert eine Menge von Constraints (= Beschränkungen, Randbedingungen), denen die Werte genügen müssen. Mit den Constraints können beliebige Relationen zwischen Variablen repräsentiert werden. Viele Probleme lassen sich als ein Constraint-Netz beschreiben, d.h. als eine Menge von Constraints, die durch gemeinsame Variablen verbunden sind. Ein Constraint-Problem ist dann eine Anfangsbelegung einiger Variablen eines Constraint-Netzes, und seine Lösung besteht darin, möglichst eindeutige Werte für die übrigen Variablen zu finden: Ausgehend von der Anfangsbelegung der Variablen werden alle damit verbundenen Constraints aktiviert; bei deren Auswertung erhalten dann weitere Variablen einen Wert, was zur Aktivierung neuer Constraints führt, bis keine weitere Wertzuweisung an eine Variable mehr möglich ist. Dies Verfahren heißt Propagierung. Während im einfachsten Fall nur feste Werte für eine Variable propagiert werden können, sind leistungsfähigere Constraintsysteme auch in der Lage, Wertemengen, Intervalle oder symbolische Ausdrücke zu propagieren.

Ein wichtiges Einsatzfeld der Suche ist der Bereich des *Planens*. Planungsprobleme sind im Grundansatz zunächst klassische Suchprobleme, jedoch häufig durch einen sehr großen Suchraum gekennzeichnet; einfache Breiten- oder Tiefensuche scheitert dann an der kombinatorischen Explosion. Durch Einbau von heuristischem Wissen lässt sich der Suchaufwand oft der Suchraum so handhabbar machen, dass nur noch sehr wenige Alternativen ausprobiert werden müssen. Mehrstufiges (hierarchisches) Planen beinhaltet zunächst die Planung einer Sequenz von abstrakten Operatoren (Grobplan), die dann schrittweise verfeinert werden. Liegt Erfahrungswissen vor, eignen sich z.B. die Verfahren des Skelettplanens oder der Phasenaufteilung. Für Planungsprobleme mit wenig Erfahrungswissen eignet sich z.B. die Differenzanalyse (means-ends-analyis); im allgemeinen wird auch dabei zunächst mit Groboperatoren geplant, die später verfeinert werden. Dabei können jedoch Nebeneffekte eines Operators die Anwendung anderer Operatoren blockieren. Abhilfe bietet das nichtlineare Planen, bei dem blockierende Interaktionen zwischen Operatoren explizit als Constraints zwischen Operatoren und Objekten repräsentiert werden und die Operator-Reihenfolge durch Überwachen der Constraints festgelegt wird.

Ausblick. Allgemeine Suchverfahren haben sich in der Praxis nicht als durchgehend fruchtbar erwiesen; die meisten Suchheuristiken sind in ihrer Nützlichkeit auf bestimmte Problemklassen beschränkt. Dennoch ist Suche eine wichtige Grundmethode für KI-Programme und wird in der einen oder anderen Form häufig benötigt. Unter zahlreichen Ansätzen der "informierten" Suche ist wissensgestützte Suche die mächtigste; ihre Grundidee besteht darin, dass explizites Wissen über einen Aufgabenbereich die Problemlösesuche extrem verkürzen *kann* (aber nicht *muss*). Ein wissensbasiertes System ist dementsprechend ein Problemlösesystem, in dem Erfahrung und Expertise eines menschlichen Problemlösers eingebettet ist, um die Suchraum-Komplexität zu kontrollieren. Dazu ist solches Wissen allerdings erst explizit zu machen und geeignet zu modellieren. Die Macht von Wissen weiter zu explorieren, sprengt den Rahmen dieses Abschnitts. Führen wir uns jedoch vor Augen, dass ein Safeknacker, der weiß, dass alle Kombinationsschlösser von Safes der Firma X bei Auslieferung mit 25-0-25 oder 50-25-50 voreingestellt sind und dass 20% der Käufer es versäumen, die voreingestellte Kombination zu ändern, im Durchschnitt jeden fünften solchen Safe mit 2 statt 1.000.000 Versuchen knacken könnte (wie im Kapitel "Safeknacker trifft Safeknacker" bei R.P. Feynman: *Sie belieben wohl zu scherzen, Mr. Feynman!* nachzulesen ist) ...