

Seminar Denken

2. Sitzung

Problemlösen I Problemräume und Lösungsstrategien

Gliederung

1. Einführung / Graphen und Bäume
2. State Action Trees
Strategien: Tiefensuche / Breitensuche / *Progressive Deepening*
Beispiel: Turm von Hanoi
3. Unterschiedsreduktion
Strategien: *Hill Climbing* / *Best First* Strategie
Beispiel: Wasserumfüllaufgabe
4. Problemreduktion
Strategien: *Goal Recursion* / *Perceptual Strategy*
Beispiel: Turm von Hanoi

Problemlösen: Einführung

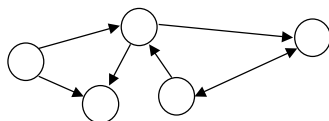
Fragestellung: Welche mentalen Prozesse sind beteiligt, wenn Menschen versuchen, Probleme zu lösen?

Die Forschung hat sich auf bestimmte Problemtypen konzentriert:

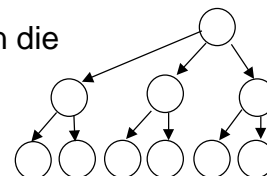
- Generell:
 - Gut-definierte Probleme → Die Vp weisen vergleichbare Ausgangsmodele der Aufgabe auf.
 - „Kleine Probleme“ → Lösungszeit < 1 h
- Probleme ohne Gegner: Semantisch arme Probleme → Alle Vpn haben wenig Vorwissen.
- Probleme mit Gegner: Vergleich Novizen und Experten

Graphen / Bäume

- Die meisten gut-definierten Probleme lassen sich in Graphen darstellen.
- Ein Graph besteht aus Knoten (nodes) und Kanten (arcs).

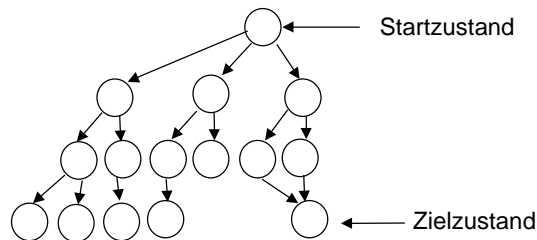


- Bäume sind spezielle Graphen, bei denen die Kanten gerichtet sind und es keine Schleifen gibt.



Graphen / Bäume

- Problemrepräsentation in Bäumen: Knoten repräsentieren die verschiedenen Problemzustände, Kanten repräsentieren die Operationen, mit denen ein Zustand in einen anderen überführt werden kann. → *State Action Trees*



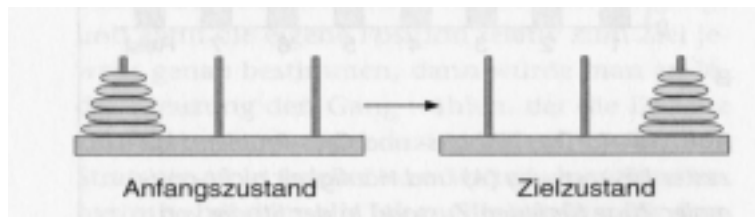
- Problemlösen ist Suche im Baum nach dem Zielzustand.

Beispiel: Turm von Hanoi

Anfangszustand: Scheiben liegen auf Stab A – der Größe nach geordnet (größte Scheibe zu unterst)

Zielzustand: Scheiben liegen auf Stab B – der Größe nach geordnet (größte Scheibe zu unterst)

Operationen: Die Scheiben dürfen umgelegt werden, aber nur so, dass zu keinem Zeitpunkt eine größere Scheibe auf einer kleineren liegt .



Beispiel: Turm von Hanoi



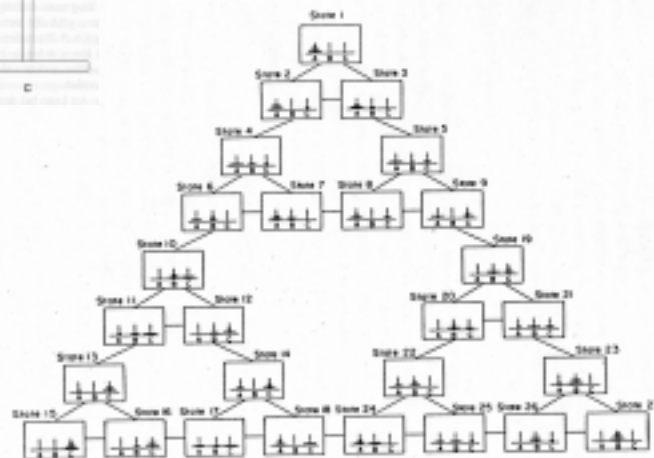
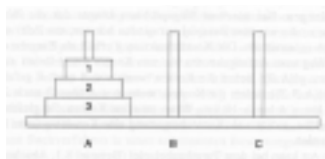
Anzahl notwendiger Züge:

$2^n - 1$, wobei n die Anzahl der Scheiben ist.

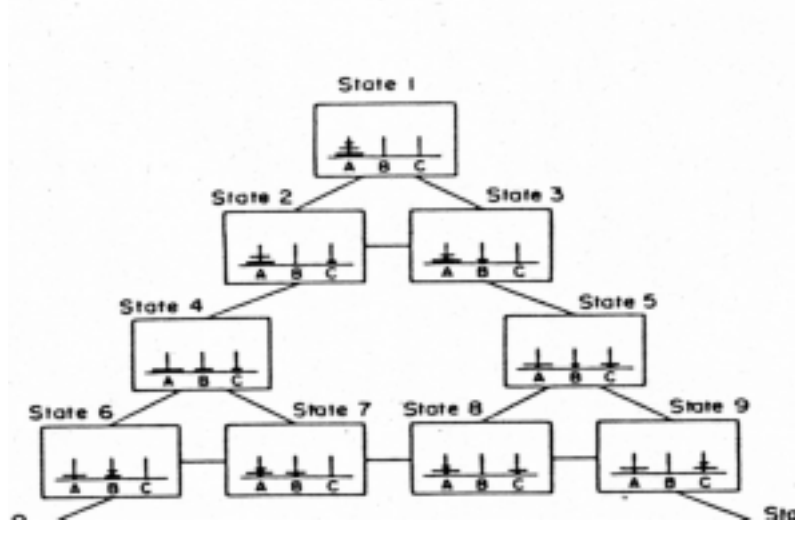
Bei $n=4$ kann das Problem also in 15 Zügen gelöst werden.

Bei $n=64$ werden schon 18446744073709524999 Züge benötigt.
(Bei einem Zug pro Sekunde braucht man für die Lösung fast eine Billionen Jahre)

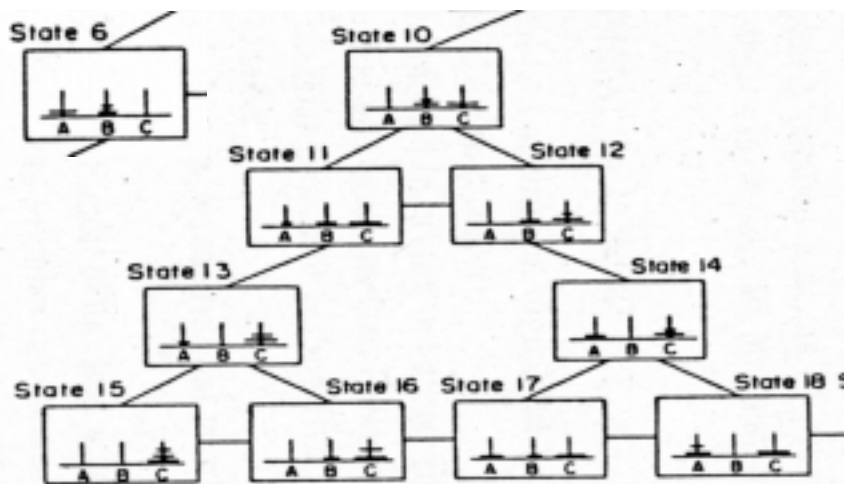
State Action Trees: Turm von Hanoi



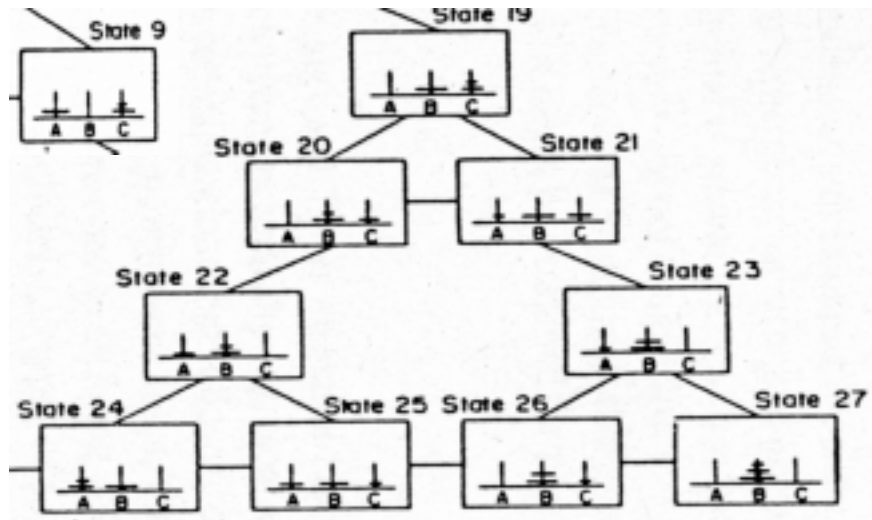
State Action Trees: Turm von Hanoi



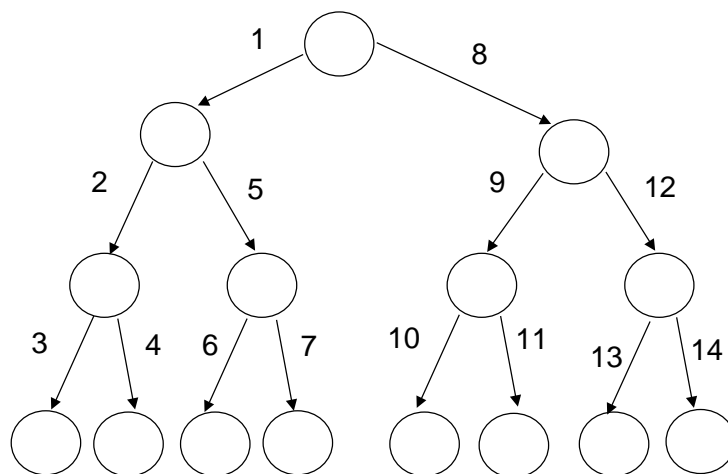
State Action Trees: Turm von Hanoi



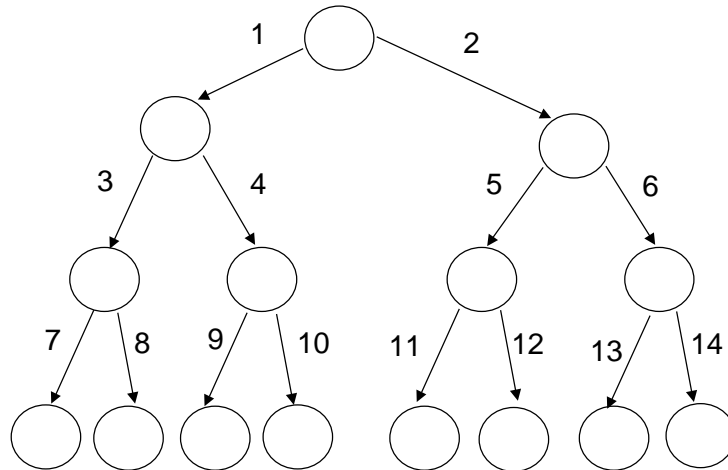
State Action Trees: Turm von Hanoi



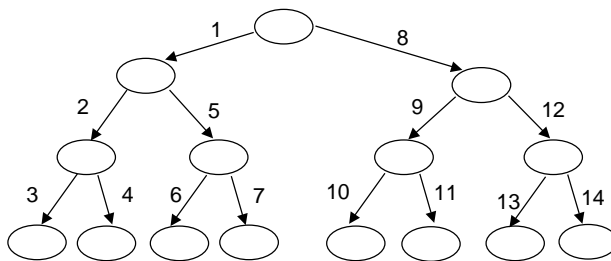
Suchstrategien: Tiefensuche (*depth first*)



Suchstrategien: Breitensuche (*breadth first*)

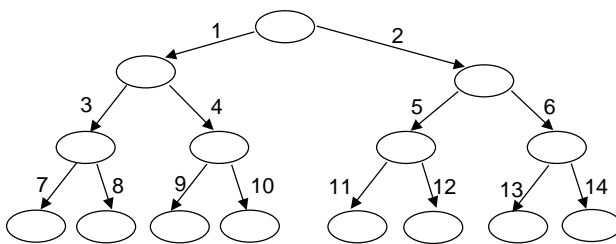


Suchstrategien: Vergleich



Tiefensuche:

- Geringer *Memory Load*
- Lösung wird u.U. nicht gefunden.
- Gefundene Lösung ist u.U. nicht die kürzeste.



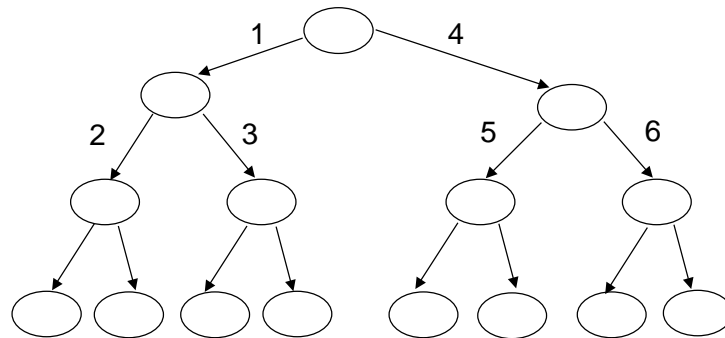
Breitensuche:

- Starker *Memory Load*
- Lösung wird gefunden.
- Gefundene Lösung ist die kürzeste Lösung.

Suchstrategien: *Progressive Deepening*

Tiefensuche bis zu einer festgelegten Tiefe (z.B. 2), wobei alle Äste bis zu dieser Tiefe durchsucht werden.

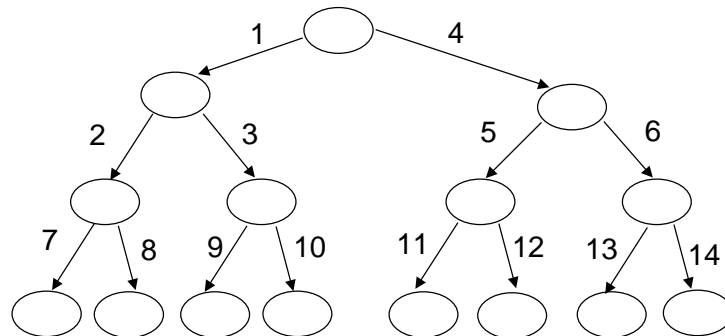
Wenn der Zielzustand dabei nicht gefunden wurde, wird das Kriterium erweitert (neues Tiefenkriterium).



Suchstrategien: *Progressive Deepening*

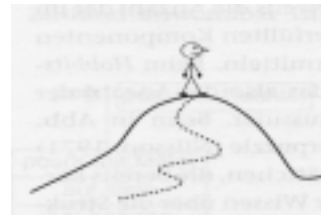
Tiefensuche bis zu einer festgelegten Tiefe (z.B. 2), wobei alle Äste bis zu dieser Tiefe durchsucht werden.

Wenn der Zielzustand dabei nicht gefunden wurde, wird das Kriterium erweitert (neues Tiefenkriterium).



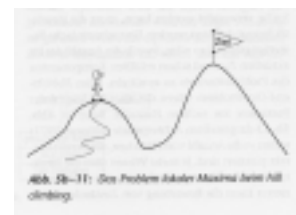
Unterschiedsreduktion

- Die Suche kann effizienter gestaltet werden, wenn die Zustände hinsichtlich ihrer Ähnlichkeit zum Zielzustand bewertet werden.
- Dies setzt eine Evaluationsfunktion voraus.
- Beim *Hill Climbing* versucht der Problemlöser, den besten Zustand im *State Action Tree* dadurch zu finden, dass er immer nur in Zustände wechselt, die dem Zielzustand ähnlicher sind als der vorherige Zustand.
- Metaphorisch: Der Zielzustand ist der höchste Punkt in der Umgebung. Dieser wird erreicht, indem immer nur Wege eingeschlagen werden, die höher hinauf führen.



Unterschiedsreduktion

- Wenn der Zielzustand nicht genau spezifiziert ist, dann ergeben sich beim *Hill climbing* möglicherweise folgende Probleme:
 - Lokales Maximum: Alle Wege führen scheinbar weg vom Ziel.
 - Plateau: Kein Weg führt näher ans Ziel.
- Alternative: Bei der *Best-First* Strategie wird die Suche vom bis dahin besten Zustand aus weiter vorangetrieben.

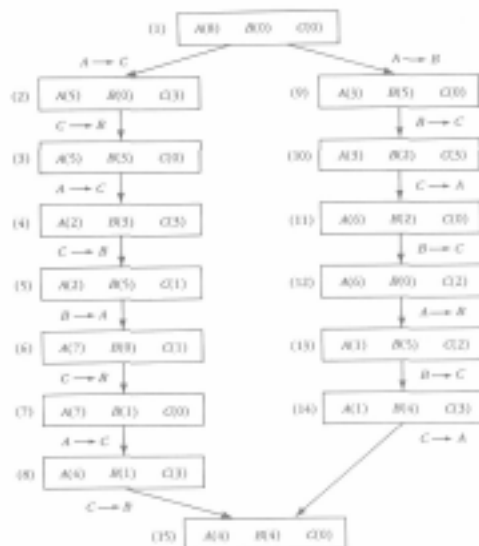


Übung: Wassermüllaufgabe

- Es gibt 3 Krüge A, B und C.
- A fasst 8 Tassen, B fasst 5 Tassen und C fasst 3 Tassen Wasser.
- Anfangszustand: A ist gefüllt, B und C sind leer.
- Zielzustand: A und B enthalten jeweils 4 Tassen Wasser, C ist leer.

Durchgang	Inhalt von			Umfüllen von
	A	B	C	
1.)	8	0	0	A nach
2.)				
3.)				

Wassermüllaufgabe



Mögliche Evaluationsfunktion:

$$E_i = |c_i(A) - G(A)| + |C_i(B) - G(B)|$$

$c_i(X)$: Tassen in Krug X bei Zustand i

$G(X)$: Tassen in X bei Zielzustand

Für welchen Zustand entscheiden sich Problemöser, die die Methode der Unterschiedsreduktion mit dieser E-Funktion anwenden?

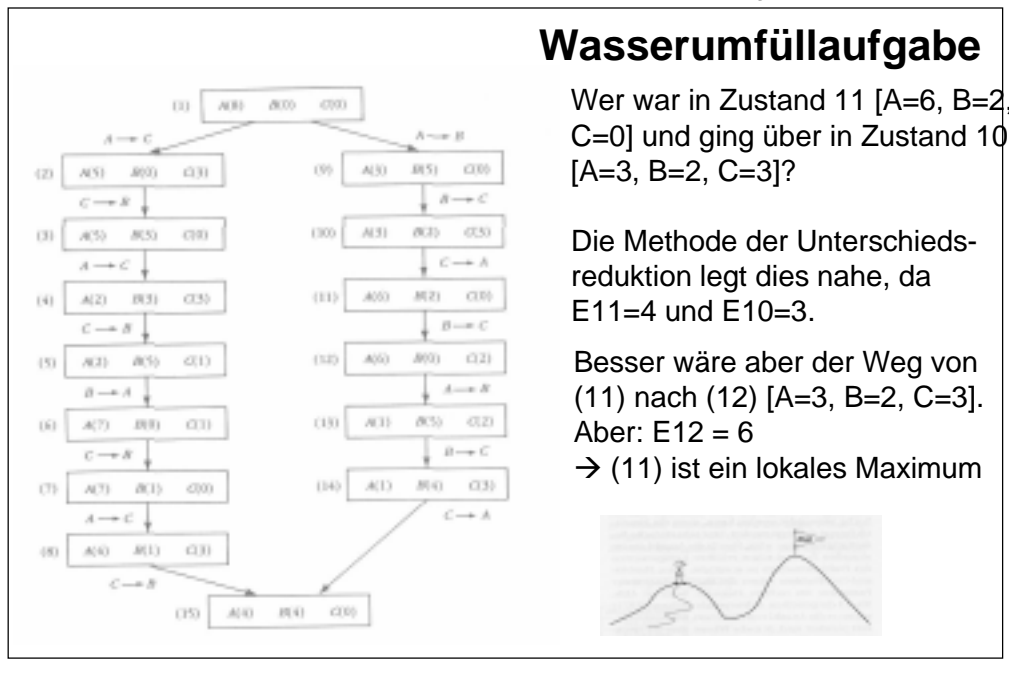
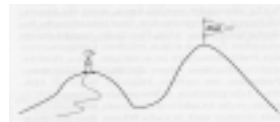
→ Zustand 9, da $E_9=2$ und $E_2=5$

Wasserumfüllaufgabe

Wer war in Zustand 11 [A=6, B=2, C=0] und ging über in Zustand 10 [A=3, B=2, C=3]?

Die Methode der Unterschiedsreduktion legt dies nahe, da $E_{11}=4$ und $E_{10}=3$.

Besser wäre aber der Weg von (11) nach (12) [A=3, B=2, C=3].
Aber: $E_{12} = 6$
→ (11) ist ein lokales Maximum

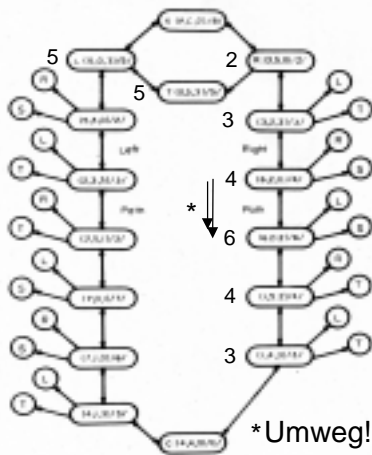


□ Wasserumfüllaufgabe: Altwood & Polson (1976)

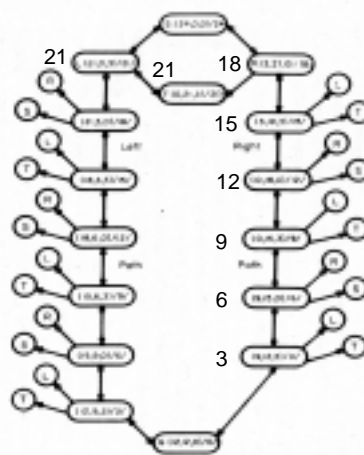
- Vpn lösen eine Reihe von Wasserumfüllaufgaben, die sich z.B. hinsichtlich folgender Faktoren unterscheiden:
 - (A) Anzahl der möglichen Schleifen
 - (B) Notwendigkeit von Umwegen
- Abhängige Variablen:
 - (1) Lösungserfolg: Wird Problem gelöst? In wie vielen Zügen?
 - (2) Strategien: Wird von bestimmten Zuständen aus, bevorzugt in bestimmte andere Zustände gewechselt?
 - (3) Gedächtniskapazität: Werden Züge wiederholt und wenn ja unter welchen Bedingungen?

Wasserumfüllaufgabe: Altwood & Polson (1976)

[8,5,3]



[24,21,3]



Wasserumfüllaufgabe: Altwood & Polson (1976)

- Das (8,5,3) Problem kann in 7 Zügen gelöst werden, Vpn brauchen durchschnittlich 26 Züge.
- Problemschwierigkeit wird bestimmt durch (a) die Menge der Schleifen und (b) die Notwendigkeit, Umwege zu laufen.
- Vpn wenden bei der Wegwahl eine Evaluationsfunktion an, und vermeiden es, in bereits besuchte Zustände zu laufen.
- Bei der Auswahl des „optimalen“ Zuges können bis zu drei mögliche Züge im Arbeitsspeicher gehalten und verglichen werden.
- Die Information über besuchte Zustände wird mit einer Wahrscheinlichkeit von $p=.90$ im LZG gespeichert.
- Vpn schauen nicht mehr als einen Zug voraus.

Simon (1975): Verschiedene Strategien

Goal Recursion:

Im Arbeitsspeicher muss eine Liste von abzuarbeitenden Zielen und Unterzielen verfügbar gehalten werden. Diese Liste wächst dramatisch mit der Anzahl der Scheiben im Turm von Hanoi Problem.

Perceptual Strategy:

- Identifiziere die größte Scheibe, die noch nicht auf Zielstab liegt [k].
- Identifiziere die größte Scheibe, die eine Bewegung von k blockiert [l].
- Wenn es keine blockierende Scheibe gibt, dann bewege Scheibe k
- Wenn es eine blockierende Scheibe l gibt, dann verfolge das Ziel l auf den Stab zu bewegen, auf dem k weder liegt noch liegen soll.
- Rekursives Abarbeiten von Schritten (b) und (c) für die blockierende Scheibe.

→ Nur ein Ziel muss im Arbeitsspeicher gehalten werden

Empirische Befunde zu Strategien

Karat (1982): Erhebung von Zuglatenzen und Zugmustern bei Aufgabenvariation

- Vpn verfolgen das Unterziel, die größte Scheibe auf Zielstab zu bewegen.
- Vpn vermeiden Zustandswiederholungen.

Anzai & Simon (1979): Eine Vp bekam vier Lösungsversuche beim Turm von Hanoi Problem

- Strategien entwickelten sich
 1. *State Action Tree* mit Vermeidung von Zustandswiederholungen
 2. *Perceptual Strategy*
 3. *Goal Recursion*