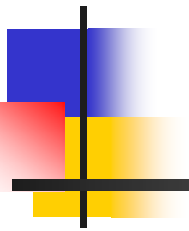
Planen als heuristische Suche



WS 06/07

Hauptseminar

„Problemlösen & Planen“

Yan Yao

Inhalt

- Einführung
- STRIPS-Zustandsmodell
- Heuristik
- HSP 2.0 - ein heuristischer Suchplaner
- Fazit und Ausblick

Grundkonzept

Transformation des Planungssproblems in ein Problem der heuristischen Suche

- Problemrepräsentation in einem Zustandsraum-modell mit einer allgemeinen deklarativen Sprache
- Automatische Gewinnung von Heuristiken direkt aus der Problemrepräsentation
- Steuerung der Suche im Zustandsraum durch die Heuristiken

STRIPS-Zustandsmodell (1)

Zustandsmodell

- formale Definition 6-Tupel **S =** (S, s0, SG, A, f, c)
- Ein übliches Modell für Planungsprobleme als heuristische Suche
- Lösung des Problems: eine Sequenz von Aktionen $a_0, a_1, \ldots a_n$;
  - Mit einer Zustandskurve s0, s1=f(s0),…sn+1=f(sn,an)
  - ai ist anwendbar auf si und sn+1 ist ein Zielzustand
- Optimale Lösung: wenn die gesamten Kosten als Summe der Kosten von einzelnen Aktionen minimal sind, d.h. $\sum c(s_i, a_i)$ mit i=0,…n minimal

STRIPS-Zustandsmodell (2)

Zustandsmodell in STRIPS

- Problem des STRIPS-Zustandsmodells Sp=(S, s_0, SG, A(.), f, c)
  - Ein Zustand s besteht aus Atomen
  - Der Initialzustand s0 besteht aus Atomen
  - Zielzustände SG befinden sich in einem Zustand, d.h. Teilmenge von einem Zustand
  - Eine Aktion anwendbar auf einen Zustand s, wenn deren Vorbedingung eine Teilmenge von dem Zustand ist
  - Übergangsfunktion f führt einen Zustand s in einen neuen Zustand s' über durch eine Aktion anwendbar auf den Zustand s d.h. s'=s-Del(a)+Add(a)
  - Kosten von einer Aktion sind 1 d.h. c(a,s)=1

STRIPS-Zustandsmodell (3)

Suche im STRIPS-Zustandsraum

- Suchkomplexität
  - Komplexe Probleme
  - Suche in sehr großen Problemräumen
  - Zeit- und Speicheraufwand enorm, d.h. exponentielle Steigung mit der Problemgröße
  - eine erschöpfende Durchsuchung eher unrealistisch
- Planungseffizienz und -effektivität
  - Gute Lösung schnell finden
  - Größe des Suchraums reduzieren
  - Suchqualität sichern / verbessern
- Notwendigkeit einer guten Technik, um die Suche zu steuern

=>Einsatz von Heuristiken

Heuristik (1)

Was ist Heuristik?

- Ursprung des Wortes: griechisch *heuriskein* – finden, entdecken
- **Strategien**, die das Auffinden von Lösungen beschleunigen können
- **Faustregel**, Schnelligkeit und Einfachheit gegenüber der Berechnung der exakten Lösung
- **Kriterien, Methoden, Prinzipien** in der Entscheidungsauswahl für eine Alternative, welche die höchste Effektivität verspricht, um ein bestimmtes Ziel zu erreichen. (Judea Pearl)
- **„Kunst des Entdeckens"** Eine Faustregel, die im allgemeinen (aber nicht unbedingt immer) die Effizienz eines Systems verbessert. (Herman Kaindl)

Heuristik (2)

Charakteristiken

- Keine Garantie für eine Lösung, auch wenn eine Lösung existiert, weil sie in der Regel zu einer nicht erschöpfenden Suche führt.
- problemspezifische Informationen nutzen, um eine günstige Verknüpfung zwischen zwei Zuständen zu ermöglichen und letzten Endes den Suchraum zu reduzieren

Heuristik (3)

Heuristische Funktionen steuern die Suche

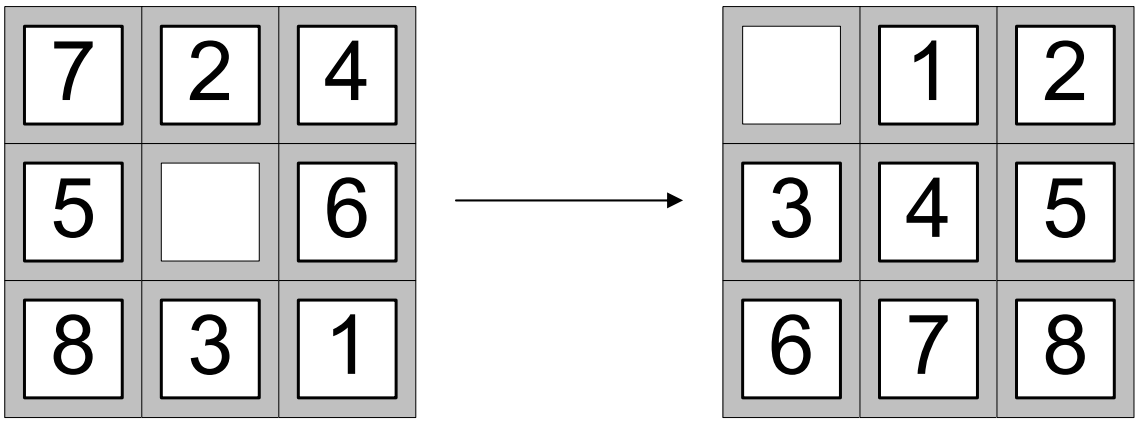
- **Eine heuristische Funktion h(n)** schätzt die Minimalkosten/Distanz von einem Knoten n zum Zielknoten.
- **Bewertungsfunktion f(n):** Knoten-Selektions-Funktion, der Knoten mit den niedrigsten Kosten/Distanz wird ausgewählt für die Expansion
- **Einsatz der heuristischen Funktion in Suchalgorithmen**
  - Greedy-Best-First-Search: f(n)=h(n)
  - A*-Suche: f(n)=g(n)+h(n), wobei g(n) für die bekannte Distanz vom Initialzustand zum Knoten n steht
  - Die Leistung von heuristischen Suchalgorithmen hängt von der Qualität der heuristischen Funktion ab

    ⟶ **Gute Heuristiken**

Heuristik (4)

Heuristische Funktionen steuern die Suche

- **Ein Beispiel von 2 heuristischen Funktionen in 8-Puzzle-Spiel**

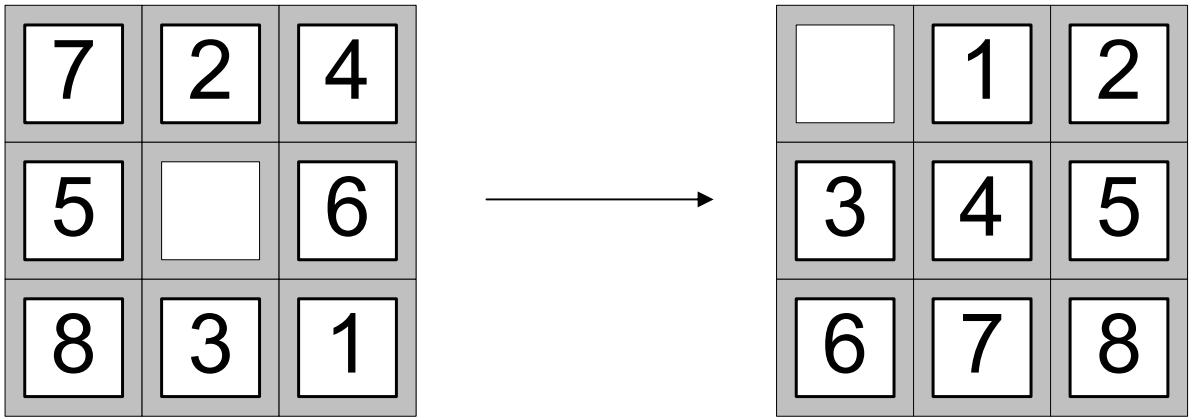


Ausgangszustand Zielzustand

- **Die Heuristik schätzt, wie nah ein Zustand zum Zielzustand liegt**
- **Die Exakte Lösung=26**

Heuristik (5)

Heuristische Funktionen steuern die Suche



Ausgangszustand Zielzustand

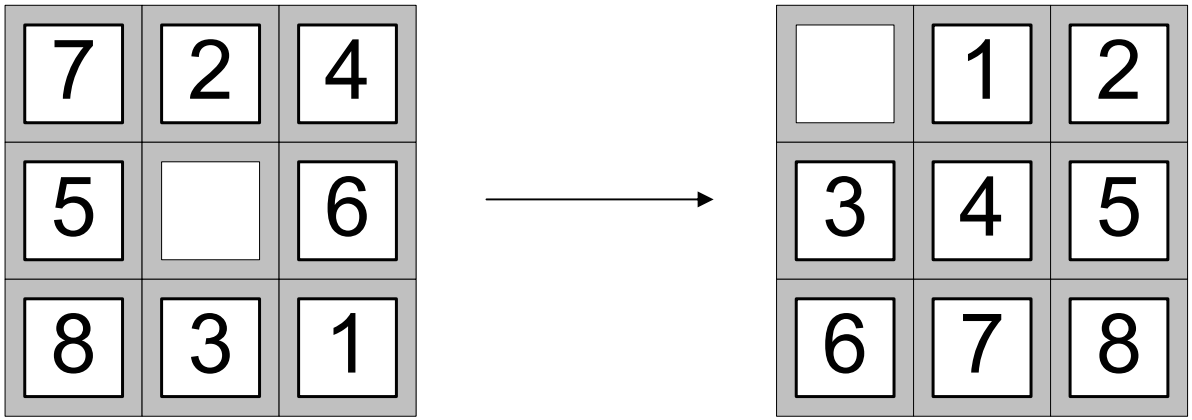
- **h1**:die Zahl der deplazierten Teilchen

  h1=8; 8 Teilchen sind nicht auf dem richtigen Platz

  Eine zulässige heuristische Funktion d.h. keine Überschätzung der optimalen Distanz
- Eine heuristische Funktion ist **unzulässig**, wenn die Schätzung den unteren Grenzwert der optimalen Lösung bildet d.h. Überschätzung

Heuristik (6)

Heuristische Funktionen steuern die Suche



Ausgangszustand Zielzustand

- **h2**: die Summe der (minimale) Distanzen von den einzelnen Teilchen zu den Zielpositionen; Manhattan-Distanz

  für „1" bis „8", die Distanz ist jeweils 3,1,2,2,,2,3,3,2

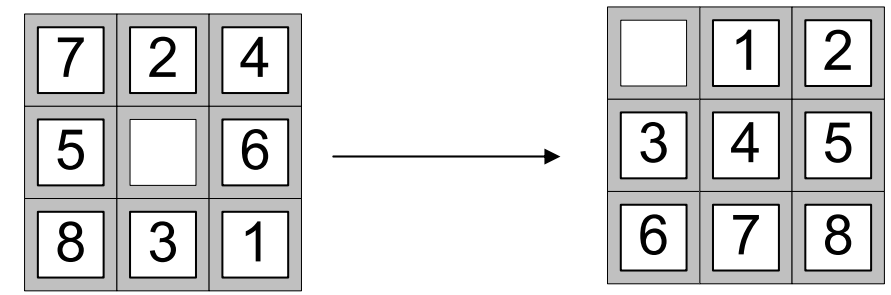
  h2=3+1+2+2+2+3+3+2=18

  eine zulässige heuristische Funktion

Heuristik (7)

Wie gewinnt man Heuristiken?

- **Grundprinzip: „Relaxation Principle“,** d.h. Probleme vereinfachen, Bedingungen/Beschränkungen lockern bzw. entfernen
- **An dem obigen Beispiel:**
  - Gewinnung der ersten Heuristik h1: alle Bedingungen werden ignoriert
  - Gewinnung der zweiten Heuristik h2: man vergisst, dass ein Teilchen erst bewegt werden kann wenn das Nachbarsfeld eine leere Stelle ist.



Ausgangszustand Zielzustand

Heuristik (8)

Wie gewinnt man Heuristiken?

- **Automatische Gewinnung** von Heuristiken direkt aus der Problemrepräsentation
  - Definition eines Problems in einer formalen Sprache
    Möglichkeit für eine automatische Konstruktion dessen Vereinfachung
  - „Relaxation" in STRIPS
    - Vorbedingungen einer Aktion ignorieren
    - Delete-Listen ignorieren
    - Vorbedingungen vereinfachen
  - Trade-off zwischen Vereinfachung und Rechenaufwand
    - Gefährdung der Effektivität der Heuristik durch die Informationsverlust
    - z.B. h1 ist viel einfacher als die h2 aber h2 ist genauer und effektiver
    - Aber der Rechenaufwand von h2 ist wiederum größer als dies von h1

Heuristik (9)

Heuristische Funktionen für Zustandsraumplanen

- „Relaxed Problem: Delete-Listen sind zu ignorieren
- Berechnung der optimalen Lösung des „Relaxed Problems" ist NP hard, deshalb Approximation
- h(s): schätzt die minimalen Kosten zur Erreichung des Ziels G von einem bestimmten Zustand s in dem „relaxed Problem"
- $g_s(p)$: schätzt die minimalen Kosten zur Erreichung des Atoms p von einem bestimmten Zustand s
  - Rekursive Definition

$$g_s(p) = \begin{cases} 0 & \text{if } p \in s \\ \min_{op \in O(p)}[1 + g_s(Prec(op))] & \text{otherwise} \end{cases}$$

Heuristik (10)

Heuristische Funktionen für Zustandsraumplanen

- Rechenalgorithmus
  - Variante des Bellman-Ford-Algorithmus, um die minimale Distanz von einem Knoten zum Startknoten zu finden
  - Initialisierung: $g_s(p)=0$ wenn $p \in s$ anderenfalls $g_s(p)=\infty$
  - Ausführung aller anwendbaren Aktionen
  - Hinzufügen der Atome aus den zugehörigen Add-Listen zum Zustand s
  - Für jede Aktion mit Pre(op) ,die ein p produziert, update $g_s(p)=\min[g_s(p),1+ g_s(\text{Pre}(op))]$
  - Vorgang wiederholen bis sich $g_s(p)$ nicht mehr verändert

Heuristik (11)

Heuristische Funktionen für Zustandsraumplanen

- „Heuristische Funktionen als Kombination von Kosten/Distanz zur Erreichung von Teilzielen
  - $g_s(C)$ schätzt die Kosten zur Erreichung einer Menge Atome C von dem bestimmten Zustand s
  - $g_s(G)$ schätzt die Kosten zur Erreichung des Ziels von dem bestimmten Zustand s
  - Definition: **$h(s)= g_s(G)$**
  - Problem: komplexe Interaktionen zwischen Teilzielen;
    - Negative/positive Abhängigkeiten
    - Unterschätzung/ Überschätzung der optimalen Lösung
  - **Die Additive Heuristik hadd**
    - $g_s(C)=\sum g_s(r)$ mit $r \in C$
    - unzulässig, aber nützlich
    - Annahme: Teilziele sind von einander unabhängig

Heuristik (12)

Heuristische Funktionen für Zustandsraumplanen

- **Max-Heuristik hmax**
  - $g_S(C)=\max g_S(r)$ mit $r \in C$
  - Die Kosten zur Erreichung einer Menge Atome sind gleich den Kosten zur Erreichung eines Atoms, der am meisten kostet
  - Zulässig, aber wenig informativ, konzentriert sich auf die schwierigsten Teilziele und übersieht die anderen Teilziele

- **h^m Higher-Order-Heuristiken**
  - Generale Form von zulässigen Heuristiken
  - Komplexe Interaktionen zwischen den Teilzielen berücksichtigen
  - m=1, Max-Heuristik
  - m=2, Graphplan-Heuristik, Max-2-Heuristik in HSP 2.0

HSP 2.0- ein heuristischer Suchplaner

Eigenschaften von HSP 2.0

- **Domänenunabhängiger Planer**
- **Basiert auf der Arbeit von Bonet und Geffner**
- **Mit h^{add} und Best-First-Search: hervorragende Performanz in AIPS'99**
- **Verbesserung durch Rückwärtssuche und zulässige Heuristiken, h^{max}, h^{2max}**
- **Heuristiken von HSP 2.0**
  - **Automatische Gewinnung der Heuristik aus der STRIP-Kodierung**
  - **Delete-Listen ignorieren**
  - **WA* Algorithmus: f(n)=g(n)+W h(n)**
  - **W mit einem Wertebereich [2,10], anpassbar je nach Domänen und Problemstellungen**

HSP 2.0 ein Block-World-Beispiel (1)

Block World Domain definition

define (domain blocks_world)

```
(:requirements :strips)
```

Strips: Teilmenge von PDDL- Standard

```
(:predicates (on-table ?x)  (on ?x ?y)  (clear ?x))
```

Prädikate

```
(:action MoveToTable
      :parameters (?x ?y)
      :precondition (and (clear ?x)  (on ?x ?y))
      :effect (and (clear ?y)  (on-table ?x)  (not (on ?x ?y))))
```

Aktion MoveToTable

```
(:action MoveToBlock1
      :parameters (?x ?y ?z)
      :precondition (and (clear ?x)  (clear ?z)  (on ?x ?y))
      :effect (and (clear ?y)  (on ?x ?z)  (not (clear ?z))  (not (on ?x ?y))))
```

Aktion MoveToBlock1

```
(:action MoveToBlock2
      :parameters (?x ?y)
      :precondition (and (clear ?x)  (clear ?y)  (on-table ?x))
      :effect (and (on ?x ?y)  (not (clear ?y)  (not (on-table ?x)))))
```

Aktion MoveToBlock2

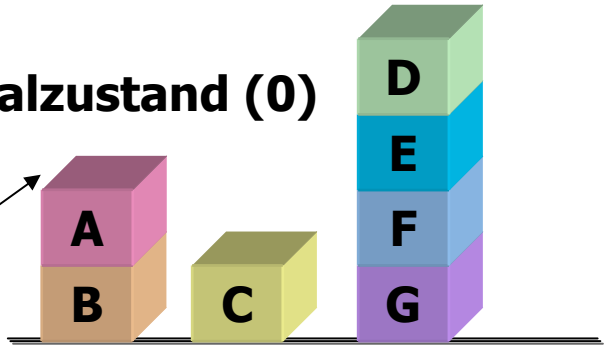
HSP 2.0 ein Blocks-World-Beispiel (2)

Problemdefinition

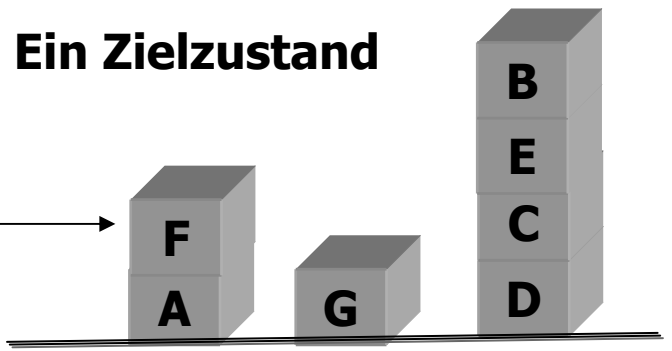
```
(define (problem beispiel)
(:domain blocks-world)
(:objects A B C D E F G)
(:init (on A B)
       (clear A)
       (on-table C)
       (clear C)
       (on-table B)
       (on D E)
       (on E F)
       (on F G))
       (on-table B)
       (clear D))

(:goal (and (on F A) (on-table G) (on C D) (on
      B E))))
```

Initialzustand (0)



Ein Zielzustand



HSP 2.0 ein Blocks-World-Beispiel (3)

Ein Lösungsprotokoll zu dem Beispielproblem

```
yanzi@linux:~> hsp beispiel.pddl hsp-2.0/pddl/blocks-world/domain.pddl
PROBLEM: solving problem: beispiel.pddl hsp-2.0/pddl/blocks-world/domain.pddl
PARAMETERS: -a bfs -d forward -h h1plus -w 2.000000 -v 1
REGISTER: staticAtomCompilation( void ) took 0.005000 secs
REGISTER: operatorCompilation() took 0.006000 secs
GENERAL: node size 72 = 64 (fixed) + 8 (variable)
GENERAL: number of relevant atoms = 56
GENERAL: number of operators = 294
GENERAL: number of buckets = 2048
REGISTER: initialize() took 0.007000 secs
SCHEDULE: forward bfs with h1plus and W = 2.0
SCHEDULE: unconstrained.
HEAPMGMT: allocating memory for 1024 nodes (73728 bytes)... done!
REGISTER: startBFS() took 0.017000 secs
SOLUTION: solution found (length = 6)
+  (MOVETOTABLE D E)
+  (MOVETOBLOCK2 C D)
+  (MOVETOBLOCK1 E F C)
+  (MOVETOTABLE A B)
+  (MOVETOBLOCK2 B E)
+  (MOVETOBLOCK1 F G A)
NODEHASH: nodes in hash table = 47
NODEHASH: diameter of hash table = 1
NODEHASH: average diameter of hash table = 1.000000
STATISTICS: number expanded nodes = 6
STATISTICS: number generated nodes = 46
STATISTICS: average branching factor = 7.666667
REGISTER: main() took 0.017000 secs
BEISPIEL,0.0170,6,(MOVETOTABLE D E),(MOVETOBLOCK2 C D),(MOVETOBLOCK1 E F C),(MOV
ETOTABLE A B),(MOVETOBLOCK2 B E),(MOVETOBLOCK1 F G A)
```

Best-First-Search

Vorwärtssuche

Additive Heuristik

W=2

Default-Modus

Lösung gefunden

6 Schritte

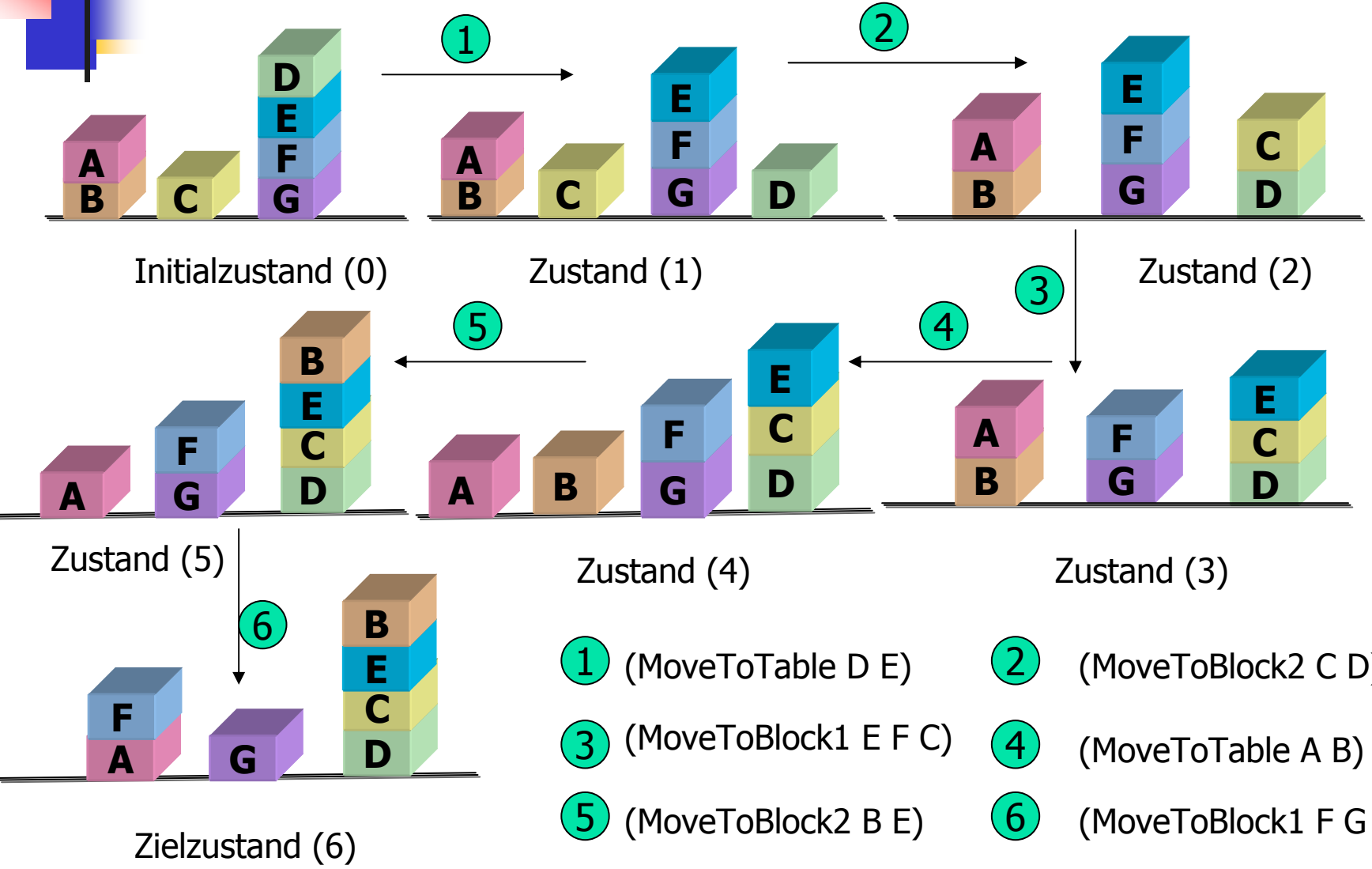
Statistiken:

Expandierte Knoten:6

Generierte Knoten: 46

Lösungsdauer 0,017 Sekunden

HSP 2.0 Blocks-World-Beispiel (4)



Fazit und Ausblick (1)

- Mit effektiven Heuristiken und Suchstrategien kann die einfache Zustandsraumplanung auch effizient sein
- Die Repräsentation des Problems in einer geeigneten formalen Sprache ist von großer Bedeutung und ermöglicht eine leichte Vereinfachung des originalen Problems
- Fast-Forward-Planer von J. Hoffman an der Uni-Freiburg
  - Erfolgreicher Nachfolger von HSP
  - Hervorragende Performanz in AIPS'00
  - „Top-Performer" in STRIPS-Domäne in der AIPS'02

Fazit und Ausblick (2)

- Vorwärtssuche im Zustandsraum
- Basiert auf der gleichen „Relaxation Principle" wie HSP
- Gewinnung eines „Relaxed Plans" (O0,…, Om-1). aus dem „Planning Graph"
  - Distanzschätzung auf dessen Basis: hFF(S):=∑|Oi| i=0,…,m-1
  - Positive Interaktionen zwischen Teilzielen berücksichtigen
  - Identifikation der am wahrscheinlichsten zum Ziel führenden Nachfolger von dem aktuellen Zustand, die durch „**Helpful Actions** "generiert werden
- Modifizierung des Hill-Climbing Algorithmus, („**enforced Hill – Climbing")**
  - Iterative Breadth –First-Search, um Plateaus und lokale Minima zu vermeiden
- Vorteile in Hinsicht auf Runtime und Lösungsschritte