


# Chapter S:VI

## VI. Relaxed Models

- ❑ Motivation
- ❑  $\varepsilon$ -Admissible Speedup Versions of A\*
- ❑ Using Information about Uncertainty of  $h$
- ❑ Risk Measures
  
- ❑ Nonadditive Evaluation Functions
  
- ❑ Heuristics Provided by Simplified Models
- ❑ Mechanical Generation of Admissible Heuristics
- ❑ Probability-Based Heuristics

# Heuristiken aus Modellvereinfachungen

Beispiel: Modellvereinfachung für das 8-Puzzle

5	2	3
8		4
7	1	6

Zulässige Schätzfunktionen:

$h_1$  Anzahl von fehlplatzierten Puzzleteilen


$h_2$  Summe der Manhattan Distanzen

Warum ist die Zulässigkeit von  $h_1$  und  $h_2$  so einfach zu sehen?

- Die Aussage  $h(n) \leq h^*(n)$  muss für alle Konfigurationen  $n$  des Puzzles, d.h. für alle möglichen Positionen der Puzzleteile auf dem Brett überprüft werden.
- $h^*(n)$  ist unbekannt.

# Heuristiken aus Modellvereinfachungen

Beispiel: Modellvereinfachung für das 8-Puzzle (continued)

5	2	3
8		4
7	1	6

Ursprüngliches Modell:

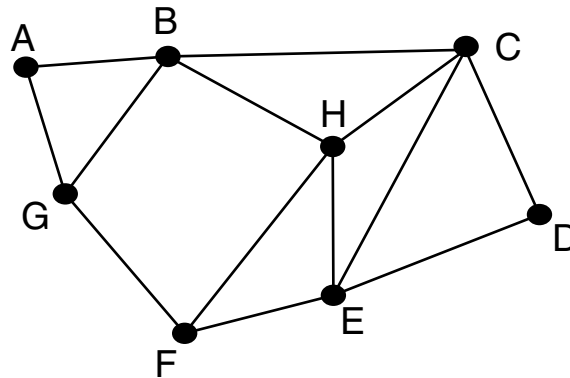
- ❑ Bewege Puzzleteile nur auf Nachbarplätze.
- ❑ Der Zielplatz für das bewegte Puzzleteil muss frei sein

Vereinfachung 1: Ignoriere besetzte Plätze.  $\rightarrow h_2$

Vereinfachung 2: Ignoriere besetzte Plätze und erlaube beliebige Sprünge.  $\rightarrow h_1$

# Heuristiken aus Modellvereinfachungen

## Beispiel: Modellvereinfachung für TSP



Zulässige Schätzfunktionen für den verbleibenden Weg:

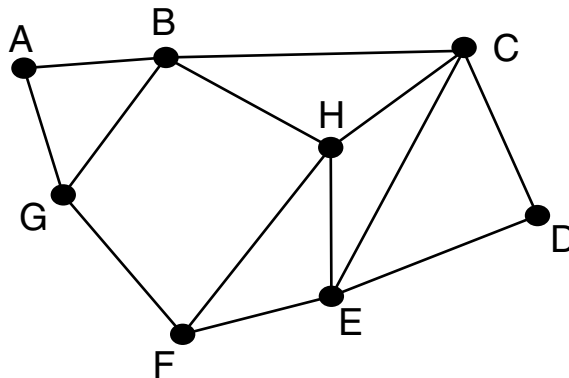
- $h_1$  Summe der Kantengewichte des billigsten Untergraphen mit Grad 2
- $h_2$  Summe der Kantengewichte des minimalen Spannbaums

Ursprüngliches Modell:

- ❑ Die Lösung muss ein zusammenhängender Teilgraph des Problemgraphen sein.
- ❑ In dem durch die Lösung induzierten Teilgraphen muss jeder Knoten den Grad 2 haben.

# Heuristiken aus Modellvereinfachungen

Beispiel: Modellvereinfachung für TSP (continued)



Vereinfachung 1: Ignoriere Zusammenhang des Lösungsgraphen.  $\rightarrow h_1$

Dieses "Optimal Assignment Problem" kann in  $O(k^3)$  Schritten gelöst werden ( $k$  Anzahl restlicher Knoten). Das Ergebnis ist immer eine Menge von Zyklen.

Vereinfachung 2: Ignoriere Knotengrade im Lösungsgraphen.  $\rightarrow h_2$

Dieses "Minimum Spanning Tree Problem" kann in  $O(k^2)$  Schritten gelöst werden, mit  $k$  als Anzahl restlicher Knoten.

# Heuristiken aus Modellvereinfachungen

## Relaxation

Als Relaxierung eines Suchproblems bezeichnet man das Entfernen von Constraints, die die Anwendung von Operatoren verbieten.

Q. Wo liegen die Vorteile der Verwendung relaxierter Modelle?

A. Relaxierte Modelle sind meist einfacher zu handhaben und zu lösen.

→ Verwendung von Heuristiken, die optimale Kosten im relaxierten Modell beschreiben.

# Heuristiken aus Modellvereinfachungen

## Relaxation

Als Relaxierung eines Suchproblems bezeichnet man das Entfernen von Constraints, die die Anwendung von Operatoren verbieten.

Q. Wo liegen die Vorteile der Verwendung relaxierter Modelle?

A. Relaxierte Modelle sind meist einfacher zu handhaben und zu lösen.

→ Verwendung von Heuristiken, die optimale Kosten im relaxierten Modell beschreiben.

Problem:

Durch Relaxierung entstehende Probleme können auch komplexer (bzgl. der Laufzeit) sein.

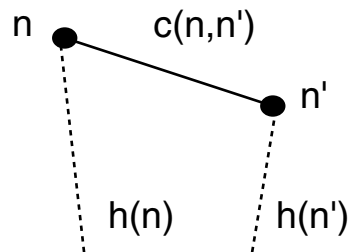
Beispiel:

zusätzliche Operatoren (z.B. diagonales Ziehen im 8er-Puzzle) vereinfachen das Problem nicht zwangsläufig.

# Heuristiken aus Modellvereinfachungen

## Zulässigkeit, Konsistenz und Monotonie

Heuristiken, die die Kosten einer optimalen Lösung in einem vereinfachten Modell beschreiben, sind zulässig und monoton / konsistent.



Die Heuristik  $h$  beschreibe die billigsten Kosten im relaxierten Modell,  $c'(n, n')$  die Kantenkosten im relaxierten Modell. Also gilt wegen der Optimalität von  $h$ :

$$h(n) \leq c'(n, n') + h(n')$$

Für die Kosten  $c'(n, n')$  im relaxierten Modell gilt

$$c'(n, n') \leq c(n, n')$$

Also folgt die Monotonie und damit die Konsistenz von  $h$ . Wegen  $h(\gamma) = 0$  für  $\gamma \in \Gamma$  folgt auch die Zulässigkeit von  $h$ .



# Heuristiken aus Modellvereinfachungen

## Over-Constraining

Als Überspezifizierung eines Suchproblems bezeichnet man das Hinzufügen weiterer Constraints, z.B. durch Vorgeben einer Teillösung.

Problem:

Die durch das Lösen von überspezifizierten Problemen gewonnenen Heuristiken sind meist nicht zulässig.

Beispiel:

Das Problem TSP kann durch die Festlegung einer Anfangstour durch einen Teil der Städte vereinfacht werden. Die durch Lösen des überspezifizierten Problems abgeschätzten Kosten können aber benutzt werden, um Suchpfade abzuschneiden. Warum?

# Automatische Generierung von Heuristiken

## Systematisches Relaxieren von Suchproblemen

Idee:

Zustände werden durch Prädikatenmengen und Operatoren durch Änderung der aktuellen Prädikatenmenge beschrieben.

Operatorbeschreibung:

- ❑ Vorbedingungsliste.  
Eine Menge von Prädikaten, die vor der Operation erfüllt sein müssen.
  - ❑ Additionsliste.  
Eine Menge von Prädikaten, die durch die Operation zu der Zustandsbeschreibung hinzugefügt werden.
  - ❑ Subtraktionsliste.  
Eine Menge von Prädikaten, die durch die Operation aus der Zustandsbeschreibung gelöscht werden.
- jetzt: Relaxation = Entfernen von Prädikaten aus Vorbedingungslisten

# Automatische Generierung von Heuristiken

## Beispiel: 8er-Puzzle

Formale Beschreibung (ähnlich STRIPS):

Prädikat	Bedeutung
$ON(x, y)$	Puzzlestein $x$ liegt auf Position $y$
$CLEAR(y)$	Position $y$ ist leer
$ADJ(y, z)$	Position $z$ ist adjazenz zu Position $y$

mit  $x \in \{X_1, \dots, X_8\}$  und  $y, z \in \{C_1, \dots, C_9\}$ .

Spielfeldbeschreibung:  $ADJ(C_1, C_2); ADJ(C_1, C_4); \dots$

Zustandsbeschreibung z.B.:  $ON(X_1, C_1); \dots; ON(X_8, C_8); CLEAR(C_9)$

Spielzug  $MOVE(x, y, z)$ : Puzzlestein  $x$  von Position  $y$  nach Position  $z$ :

- ❑ Vorbedingungsliste:  $ON(x, y), CLEAR(z), ADJ(y, z)$
- ❑ Additionsliste:  $ON(x, z), CLEAR(y)$
- ❑ Subtraktionsliste:  $ON(x, y), CLEAR(z)$

# Automatische Generierung von Heuristiken

## Beispiel: 8er-Puzzle (continued)

Relaxierung entspricht dem Entfernen von Prädikaten aus der Vorbedingungsliste:

1. Entfernen von  $CLEAR(z)$ ,  $ADJ(y, z)$ .

Ein Puzzlestein kann direkt auf seine Zielposition gelegt werden, egal, ob frei oder nicht.

→ Heuristik  $h_1$  (Anzahl falschplazierter Puzzlesteine)

2. Entfernen von  $CLEAR(z)$ .

Ein Puzzlestein kann zu seiner Zielposition wandern, ohne dass Zwischenpositionen frei sein müssen.

→ Heuristik  $h_2$  (Manhattan-Distanz)

3. Entfernen von  $ADJ(y, z)$ .

Ein Puzzlestein kann direkt auf das freie Feld gelegt werden (Swap-Sort).

→ neue Heuristik  $h_3$  ( $\leq 1.5 \times$  Anzahl falschplazierter Puzzlesteine)

# Automatische Generierung von Heuristiken

Problem:

Relaxierung soll eine vereinfachtes Modell des ursprünglichen Suchproblems liefern. Wann ist ein Problem wirklich einfacher?

Frage:

Kann für ein Suchproblem automatisch ein vereinfachtes Modell erzeugt werden ohne die Vereinfachung zu lösen? – genauer:

Kann für ein Suchproblem der **Grad der Vereinfachung** für die relaxierten Kandidaten bestimmt werden? – einfacher:

Kann für ein Suchproblem zumindest ein **einfaches Modell erkannt** werden, wenn es durch Relaxierung erzeugt wird?

Idee:

Wahl von Klassen einfacher Probleme, für die automatisch die Zugehörigkeit überprüft werden kann.

# Automatische Generierung von Heuristiken

## Einfache Probleme: kompositionale Probleme

In kompositionalen Problemen werden die Zielzustände durch Bedingungen beschrieben, die aus getrennten, unabhängig voneinander lösbaren Teilproblemen bestehen.

### Beispiel 1:

Für das 8er-Puzzle definiert eine Konjunktion von Prädikaten  $ON(X_i, C_j)$  den Zielzustand. Bei Benutzung der Heuristik  $h_1$  (Relaxierung: Entfernen von  $CLEAR(z)$ ,  $ADJ(y, z)$ ) stellt jede Zielbedingung  $ON(X_i, C_j)$  ein unabhängig lösbares Teilproblem – dem unmittelbaren Plazieren des Puzzleteils – dar.

### Beispiel 2:

Für das 8er-Puzzle bilden die einzelnen Zielbedingungen auch bei Benutzung der Heuristik  $h_2$  (Relaxierung: Entfernen von  $CLEAR(z)$ ) unabhängig lösbare Teilprobleme. Hier besteht das Teilproblem aus der Festlegung einer Zugfolge von der aktuellen Position eines Puzzlesteines zu seiner Zielposition.

# Automatische Generierung von Heuristiken

## Einfache Probleme: semikompositionale Probleme

Semikompositionale Probleme zeichnen sich dadurch aus, dass entstehende Teilprobleme nicht völlig unabhängig sind.

- Kommutative Probleme:

Die Reihenfolge der Anwendung einer Menge von Operatoren verändert nicht die Menge der in der Zukunft ausführbaren Operatoren (Greedy-Algorithmen / Hill-Climbing sind anwendbar)

Beispiel: Minimum-Spanning-Tree Heuristik für das TSP

- Partiiell geordnete Probleme:

Teilprobleme können so geordnet werden, dass die Lösung eines Teilproblems kein schon gelöstes Teilproblem mehr beeinflusst.

Beispiel: Heuristik  $h_3$  (Swap-Sort) beim 8er-Puzzle

Aufgabe:

Automatische Erkennung solcher Probleme anhand der formalen Beschreibung.

# Probabilistische Heuristiken

## Probabilistisch definierte Probleme

Idee:

Verfolge die **wahrscheinlich** aussichtsreichste Teillösung zuerst.

Problem:

Es kann nicht garantiert werden, dass die Heuristik zulässig (admissible) ist.

Manche Probleme sind von Natur aus probabilistisch. Für solche Probleme können oft Heuristiken durch statistische Betrachtungen ermittelt werden.



# Probabilistische Heuristiken

## Probabilistisch definierte Probleme (continued)

Beispiel 1: billigste Pfade in Graphen.

Die Kantenkosten seien eine Zufallsgröße mit Mittelwert  $\mu$ . Dann liefert  $f(n) = g(n) + \mu N$ , wobei  $N$  die Anzahl der Kanten von  $n$  zu einer Lösung bezeichnet, eine Kostenschätzung.

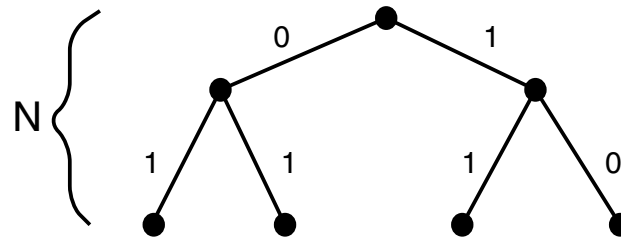
# Probabilistische Heuristiken

## Probabilistisch definierte Probleme (continued)

Beispiel 1: billigste Pfade in Graphen.

Die Kantenkosten seien eine Zufallsgröße mit Mittelwert  $\mu$ . Dann liefert  $f(n) = g(n) + \mu N$ , wobei  $N$  die Anzahl der Kanten von  $n$  zu einer Lösung bezeichnet, eine Kostenschätzung.

Beispiel 2: billigste Pfade in probabilistischen Bäumen.



## Bemerkungen:

- ❑ Ein probabilistischer Baum  $T$  wird aus dem Wahrscheinlichkeitsraum  $T(N, p)$  gezogen, wobei  $N$  die Baumhöhe bezeichnet.  $T$  ist immer binär und vollständig. Die Wahrscheinlichkeit, dass  $T$   $N_0$  Kanten mit Gewicht 0 und  $N_1$  Kanten mit Gewicht 1 hat, beträgt  $p^{N_1}(1 - p)^{N_0}$ .
- ❑ Ein Knoten  $n$  hat Kosten  $c$ , gdw.  $c$  die Summe der Pfadkosten von der Wurzel  $s$  nach  $n$  ist. Die Zufallsvariable  $C(N, p)$  bezeichnet die minimale Anzahl von Kanten mit Gewicht 1 auf einem Pfad von  $s$  zu einer Wurzel in einem Baum aus  $T(N, p)$ . Es gilt:  $p < \frac{1}{2}$ .
- ❑ Such-Algorithmus: In jedem Schritt wird der Knoten mit den billigsten Kosten expandiert. Existieren mehrere solcher Knoten, wird der linkeste gewählt. Das erste Blatt, das expandiert werden soll, wird als Lösung genommen.
- ❑  $t(N, p)$  bezeichnet die Anzahl der während einer Ausführung des Algorithmus expandierten Knoten.

# Probabilistische Heuristiken

## Asymptotische Verteilung von $C(N, p)$

### Theorem 78

Die optimalen Kosten bleiben mit an Sicherheit grenzender Wahrscheinlichkeit beschränkt:

$$P(C(N, p) > k) \leq (2p)^{2^{k+1}-2} \quad k = -1, 0, 1, \dots, N-1 \quad N \rightarrow \infty$$

### Proof (Skizze)

$$F_k^N = P(C(N, p) > k), \quad k = -1, 0, 1, \dots, N-1$$

Um eine Rekurrenzgleichung für  $F_k^N$  aufzustellen, müssen 4 Ausgänge für das Ereignis  $C(N, p) > k$  untersucht werden:

Kosten links	0	1	0	1
Kosten rechts	0	0	1	1
Wahrscheinlichkeit	$(1-p)^2$	$p(1-p)$	$(1-p)p$	$p^2$
$P(C(N, p) > k)$	$(F_k^{N-1})^2$	$F_{k-1}^{N-1} F_k^{N-1}$	$F_k^{N-1} F_{k-1}^{N-1}$	$(F_{k-1}^{N-1})^2$

# Probabilistische Heuristiken

## Asymptotische Verteilung von $C(N, p)$

### Proof (Skizze)

(continued)

$$F_k^N = ((1-p)F_k^{N-1} + pF_{k-1}^{N-1})^2, \quad F_{-1}^N = 1$$

$N \rightarrow \infty$ :

$$F_k = \lim_{N \rightarrow \infty} F_k^N = ((1-p)F_k + pF_{k-1})^2$$

Diese Gleichung hat die nicht-triviale Lösung ( $p < \frac{1}{2}$ ):

$$F_k = \frac{1}{2(1-p)^2} (1 - 2F_{k-1}p(1-p) - \sqrt{1 - 4F_{k-1}p(1-p)})$$

Unter Benutzung der Ungleichung  $\sqrt{1-x} \geq (1-x)(1+\frac{x}{2})$ ,  $0 \leq x \leq 1$  ergibt sich:

$$F_k \leq 4p^2(F_{k-1})^2$$

Da  $F_{-1} = 1$  ist, kann obiger Term wie folgt vereinfacht werden:

$$F_k \leq (4p^2)^{2^k-1} = (2p)^{2^{k+1}-2}$$

# Probabilistische Heuristiken

Asymptotische Verteilung von  $C(N, p)$

## Theorem 79

Eine obere Schranke für die Laufzeit ist:

$$E(t(N, p)) = O(N)$$

# Probabilistische Heuristiken

## Heuristiken aus der Analyse von Stichproben

Beispiel: Wertesuche in Listen. Sei eine Liste  $L$  mit  $n$  verschiedenen, zufälligen reellen Zahlen gegeben:  $(34.6, 11.5, 33.8, 5.4, 9.99, 1.4, 4.0 \dots)$

- ❑ Algorithmus 1.

Durchlaufen der Liste  $\Rightarrow O(n), \Omega(n)$ .

- ❑ Algorithmus 2.

Es sei nun der maximale Wert  $X_m$  bekannt  $\Rightarrow$  der Erwartungswert der Laufzeit liegt nun bei  $\frac{n}{2}$

- ❑ Algorithmus 3.

Nun soll nur ein Wert  $X$  in der  $\epsilon$ -Nachbarschaft um den maximalen Wert  $X_m$  gefunden werden ( $|X - X_m| < \epsilon$ ).  $p(X_m, \epsilon)$  sei die Wahrscheinlichkeit, dass ein Listenelement in  $\epsilon$ -Nachbarschaft um den maximalen Wert  $X_m$  liegt. Also ergibt sich der Erwartungswert der Laufzeit zu  $\frac{1}{p(X_m, \epsilon)}$

# Probabilistische Heuristiken

## Heuristiken aus der Analyse von Stichproben (continued)

Wenn die Listenelemente unabhängig sind und  $X$  eine kontinuierliche Zufallsvariable aus der Verteilung  $F_X(x)$  ist, dann gilt für kleine  $\epsilon$ :

$$\begin{aligned} p(X_m, \epsilon) &= P(X_m - \epsilon \leq X \leq X_m \mid X \leq X_m) \\ &= \frac{F_X(X_m) - F_X(X_m - \epsilon)}{F_X(X_m)} \\ &\approx \frac{F'_X(X_m)}{F_X(X_m)} \epsilon \end{aligned}$$

Die Menge der notwendigen Tests ergibt sich also zu:

$$\approx \frac{F_X(X_m)}{F'_X(X_m)} \frac{1}{\epsilon}$$

Dieses Ergebnis ist unabhängig von  $n$ .