Sub-optimale Graphensuche für wissensbasierte Bildinterpretation

Wlodzimierz Kasprzak

Bericht über die Forschung im Rahmen des AvH–Stipendiums

Zusammenfassung

Das Ziel war die theoretische und experimentelle Beurteilung der Anwendung sub-optimaler Baumsuche bzw. Graphensuche innerhalb einer sogenannten alternierenden Kontrolle, die als Basisverfahren für wissensbasierte Bildfolgeninterpretation dienen soll. Als erstes wird ein Vergleich unterschiedlicher optimaler Graphen-Suchverfahren unternommen. Der Zusammenhang zwischen der Qualität der Bewertungsfunktion und dem Aufwand der Baumsuche wird als nächstes untersucht. Zwei Techniken der suboptimalen Suche nach Zielkonzepten – das MIN-MAX Ausschliessen und die Zielobjekte-Äquivalenz – werden vorgeschlagen. Ein weiterführendes Ziel wäre die Schaffung einer Kontrolle mit Optionen für verschiedene optimale und sub-optimale Suchverfahren.

Einführung

Der A^*-Algorithmus ist ein optimales Suchverfahren für Graphen mit additiver Kostenfunktion $f = g + h$, unter Annahme sog. *zulässiger Heuristik* h [NIL71]. Ohne der letzten Bedingung wird der A^* als A bezeichnet. Die *Optimalität* von A^* 'bedeutet, daß der A^* keinen Suchgraphknoten mehr expandiert, als irgendein Verfahren, das mit derselben heuristischen Information ausgestattet ist wie A^*.

Die Optimalität von A^* ist bekanntlich bezogen auf *konsistente Heuristik*. Diese Einschränkung kann aber nicht generell von den Anwendern gefordert werden. Vielmehr können wir es mit nicht konsistenten und sogar *unzulässigen* Bewertungen zu tun haben. Dies kommt besonders dann vor, wenn sich ein Modell in der Lernphase befindet. Für diese allgemeinenen Fälle wurden Verfahren vorgeschlagen mit besserem Verhalten als A – A^{**} [DEC88], bzw. B [MAR77], C [BAG88] und D [MAH88]. Der Aufwand der Graphensuche wird bezüglich $\mathcal{N}$ gerechnet – der Anzahl von Knoten im Graphen mit niedrigeren Kosten als die optimale Lösung.

In der Praxis kann kaum mit einer "guten" Heuristik gerechnet werden. Weil bei beschränkter Rechenkapazität größere Probleme bisher nicht gelöst sein konnten, hat man *sub-optimale* [PEA84] bzw. *lokal optimale* Suche [KOR90] vorgeschlagen.

Bei der A^*–Baumsuche muß in Grenzfällen von einem linearen bis zu exponentiellem Aufwand bzgl. der *Pfadlänge* N ausgegangen werden. Der Austausch zwischen Qualität der Heuristik und Komplexität der Suche ist wenig elastisch [PEA84]. Hohe Präzision ist erforderlich um einen polynomischen Aufwand zu erreichen. Wir referieren die Ergebnisse von [HUY80], [PEA83] (Suchbaum mit einem einzigen Zielknoten) und [BAG88] (Suchbaum mit mehreren Zielknoten).

Ein Weg den Aufwand der Baumsuche zu reduzieren besteht in *hybriden* Verfahren, die optimale Suche mit Tiefensuche kombinieren [KOR85], [CHA89].

Welche Schlüsse können wir aus der Übersicht von Graphen- und Baum-Suche für die wissensbasierte Signalinterpretation ziehen ? Jetzt haben wir es mit unterschiedlichen **Zustandsraum** und **Suchbaum** zu tun. Beide sind dazu nur *implizit* gegeben. Damit ist auch die Bewertungsfunktion implizit – die

Kantenkosten $c(v_i, v_j)$ sind sekundär zu den Knotenkosten und sind das Resultat einer Substraktion: $c(v_i, V_j) = g(v_j) - g(v_i)$.
Wir analysieren das Verhalten der alternierenden Kontrolle [KUM91] ohne dabei die Einzelheiten der für Sprachanalyse aufgebauten Bewertungsfunktion [SAG90] theoretisch aufzufassen. Die analysierte Kontrolle vereinigt in sich drei Suchprobleme. Wir befassen uns grundsätzlich mit der Suche nach bester Sequenz von Zielkonzepten, die zum globalen Ziel führt. Das ist ein zu den zwei restlichen übergeordnetes Suchproblem (das wir mit $S1$ bezeichnen).

Bei der Bildfolgeninterpretation können wir die globale Optimalität der Suche zugunsten einer *lokalen* abschwächen. Durch die Anwendung einer schritthaltenden Analysestrategie wird die eventuell nur lokal optimale Interpretation, dank immer neu hinzukommenden Bilddaten, ständig verifiziert und in eine global optimale übergeführt.

Wir beginnen mit dem Entwurf eines *bipartiellen* Bewertungsschemas, daß den Aufbau des Suchraumes über Daten und Modell wiederspiegelt. Als nächstes folgen zwei Suchverfahren.
Die Idee einer suboptimalen Baumsuche liegt darin relativ 'schlechte' Pfade im Vergleich zum aktuell besten Pfad in markanten Punkten der Analyse (bei Übergängen im Suchraum $S1$) zeitweilig zurückzustellen. Das Verfahren garantiert eine eine *eps*–optimale Lösung.
Die zweite Idee betrifft die Umwandlung der Baumsuche in $S1$ in eine quasi-Graphensuche. Es werden *äquivalente* Ziele definiert, die uns erlauben aus äquivalenten Pfaden im Suchbaum nur den besten weiter zu verfolgen. Dies bedeutet, daß ganze Unter-Suchbäume für Suchprobleme (2) und (3) vermieden sein können.

Inhalt:

1 A, A^*

Gemäß Nilsson[NIL71] ist A eine Bestensuche für endliche Graphen mit additiver Kostenfunktion $f = g + h$.

Klassen von Heuristikfunktionen $h(n)$ für Graphen $\mathcal{G}$:

- Eine Heuristik $h(n)$ ist *zulässig* wenn $h(n) \leq h^*(n), \forall n \in \mathcal{G}$.
- Eine Heuristik h_2 ist *mehr informiert* als h_1 wenn beide zulässig sind und $h_2(n) > h_1(n), \forall n \in \mathcal{G}$ (außer für Zielknoten).
- Eine Heuristik $h(n)$ ist *konsistent* wenn $h(n) \leq c(n, n') + h(n')$, $\forall$Pfade $(n, n') \in \mathcal{G}x\mathcal{G}$, wobei $c(n, n')$ sind die Kosten des Pfades von n nach n'.
- Eine Heuristik $h(n)$ ist *monoton* wenn $h(n) \leq c(n, n') + h(n'), \forall$ Kanten $(n, n') \in \mathcal{G}x\mathcal{G}$. *Monotone* und *konsistente* Heuristiken sind äquivalente Begriffe.

In [NIL80] wird definiert, daß A^* ein A-Algorithmus ist mit der Bedingung, daß $h(n) \leq h^*(n)$ für jeden Knoten $n \in \mathcal{G}$ erfüllt ist, d.h. h zulässig ist.
In der neueren Literatur [BAG88], [PEA84], [DEC88] wird folgender Standpunkt vertreten: Ein Algorithmus wird vielmehr dadurch charakterisiert, WIE er die Daten verarbeitet und nicht durch den TYP der verarbeiteten Daten. Somit gäbe es keinen Unterschied zwischen A und A^*.

Bei unzulässiger Heuristik ist bei der A-Suche folgendes wichtig: bei zwei Pfadem zu demselben Knoten wird der bessere Pfad auf der Basis der Funktion g und nicht f ausgewählt (Schritt 7.b). Die Funktion f wird wie immer zur Knotenauswahl aus OFFEN (Schritt 3) verwendet.

A bzw. A^*:

1. bringe s nach OFFEN; setze $g(s) \leftarrow 0, f(s) \leftarrow 0$.
2. IF OFFEN ist leer THEN STOP mit Fehler.
3. Entferne aus OFFEN den Knoten n mit kleinstem f Wert und bringe n nach GESCHLOSSEN. (Löse Schlingen zugunsten eines Zielknoten auf).
4. IF n ist ein Zielknoten THEN STOP mit aktuellem $g(n)$ Wert und Lösungspfad von s nach n als Ergebnis.
5. Expandiere n (falls keine Nachfolger dann $\longrightarrow$ Schritt 2).
6. FOR jeden Nachfolger n_i von n DO: $g_i \leftarrow g(n) + c(n, n_i)$.
7. FOR jeden Nachfolger n_i von n DO:
   a. IF $n_i \notin$ (OFFEN $\cup$ GESCHLOSSEN)
      THEN bringe n_i nach OFFEN, $g(n_i) \leftarrow g_i$, $f(n_i) \leftarrow g_i + h(n_i)$.
   b. IF $n_i \in$ (OFFEN $\cup$ GESCHLOSSEN) $\wedge$ $(g(n_i) > g_i)$
      THEN $g(n_i) \leftarrow g_i$, $f(n_i) \leftarrow g_i + h(n_i)$; entferne den früheren Pfad $(s - n_i)$
      IF $n_i \in$ GESCHLOSSEN THEN bringe n_i nach OFFEN.
8. $\longrightarrow$ Schritt 2.

Eine Wariante von A^*, in der nach Erreichen eines bereits expandierten Knoten n_i (Schritt 7.b), die neue Bewertung auch gleich weiter propagiert wird ohne den Knoten von GESCHLOSSEN nach OFFEN zu bringen, wird in [NIL80] definiert. Bei Baumsuche ist diese Modifikation ohne Bedeutung, da der Knoten n_i immer neu ist und Schritt 7.b dann niemals stattfindet. Bei Graphensuche mit konsistenter Heuristik wird der Schritt 7.b auch niemals ausgeführt weil $g(n_i) \leq g_i$ immer gilt.

2 Die Optimalität von A^* bei zulässiger Heuristik

Charakteristiken von Graphen-Suchverfahren:

- Ein Algorithmus ist **komplett** wenn er immer eine Lösung findet, falls solche existiert.
- Ein Algorithmus ist **zulässig** wenn er immer eine optimale Lösung findet falls solche existiert.
- Zwei Algorithmen sind **gleich informiert** wenn sie Zugang zu derselben Heuristik-Information haben, ganz unabhängig davon wie sie diese Information nutzen.

  Beispiel: Ein Alg. B kann sich durch zwei legitime Wege Information verschaffen, die ihn dazu bringt einen Knoten n für den gilt $g(n) + h(n) \leq C^*$ (C^* - die Kosten des biligsten Lösungspfades) nicht zu expandieren. Zum einen kann B die Eigenschaften des bisher expandierten Suchraumes untersuchen und feststellen, daß n in Wirklichkeit einer viel höheren Schätzung $h(n)$ bedarf, als dies in n geschätzt wird. Zum anderen kann B diese Information aus dem bisher nicht expandiertem Suchgraphen gewinnen. Im Prinzip könnte A^* dies auch tun.
- Algorithmus A **dominiert** über Algorithmus B bezüglich der Menge $\mathcal{I}$ von Problemen wenn für jede Problem-Instanz $I \in \mathcal{I}$, die Menge der durch A expandierten Knoten eine Untermenge der durch B expandierten Knoten ist ($V(\mathrm{A}) \subseteq V(\mathrm{B})$).
- Ein Algorithmus ist **optimal** gegenüber einer Klasse von Algorithmen, wenn er jeden Algorithmus aus dieser Klasse dominiert.

Dechter und Pearl [DEC88] untersuchen die Optimalität von A^* bei zulässiger Heuristik. Am Anfang führen Sie eine weitere Charakteristik der Heuristik ein. **Nichtpathologische** Probleme – es gibt mindestens einen optimalen Lösungspfad P in $\mathcal{G}$ auf dem h nicht voll informiert ist, d.h. $\forall n \in P : h(n) < h^*(n)$ (außer dem Zielknoten).

Vier Mengen von **Problem-Instanzen** (Heuristik-Funktionen für den Suchgraph G):

- $\mathcal{I}_{ZU}$ - zulässige Funktionen $h(n)$
- $\mathcal{I}_{\overline{ZU}}$ - zulässige und nichtpathologische $h(n)$; $\mathcal{I}_{\overline{ZU}} \subseteq \mathcal{I}_{ZU}$
- $\mathcal{I}_{KON}$ - konsistente (monotone) Funktionen $h(n)$; $\mathcal{I}_{KON} \subseteq \mathcal{I}_{ZU}$
- $\mathcal{I}_{\overline{KON}}$ - konsistente nichtpathologische $h(n)$; $\mathcal{I}_{\overline{KON}} \subseteq \mathcal{I}_{KON}$

Die nichtpathologischen Probleme bilden also spezifische Untermengen von $\mathcal{I}_{ZU}$ (und $\mathcal{I}_{KON}$).

Vier **Optimalitäts**–Kategorien [DEC88]:

1. A^* ist **0-optimal** gegenüber $\mathcal{A}$ bezüglich $\mathcal{I}$ wenn für jedes Problem $I \in \mathcal{I}$ und bei jeder Schlingen-Auflösungs-Regel in A^* und bei jedem Algorithmus A aus $\mathcal{A}$, A^* expandiert eine Untermenge der durch A expandierten Knoten.

$$\forall_{I \in \mathcal{I}} \forall_r \forall_{B \in \mathcal{A}} : V_r(A^*) \subseteq V(B)$$

2. A^* ist **1-optimal** gegenüber $\mathcal{A}$ bezüglich $\mathcal{I}$ wenn für jedes Problem $I \in \mathcal{I}$ zumindest eine Schlingen-Auflösungs-Regel existiert, bei der für jeden Algorithmus A aus $\mathcal{A}$ eine Untermenge der durch A expandierten Knoten expandiert wird.

$$\forall_{I \in \mathcal{I}} \exists_r (\forall_{B \in \mathcal{A}} : V_r(A^*) \subseteq V(B))$$

3. A^* ist **2-optimal** gegenüber $\mathcal{A}$ bezüglich $\mathcal{I}$ wenn es kein Problem $I \in \mathcal{I}$ gibt, für das ein A aus $\mathcal{A}$ eine ordentliche Untermenge der durch A^* bei irgendeiner Schlingen-Auflösungs-Regel expandierten Knotemenge expandiert.

$$\sim \exists_{I \in \mathcal{I}} (\exists_{B \in \mathcal{A}} \exists_r : V(B) \subseteq V_r(A^*))$$

4. A^* ist **3-optimal** gegenüber $\mathcal{A}$ bezüglich $\mathcal{I}$ wenn folgendes gilt: falls ein Problem $I_1 \in \mathcal{I}$ existiert, für das ein Algorithmus $B \in \mathcal{A}$ einen Knoten ausläßt, der bei einer Schlingen-Auflösungs-Regel $r1$ durch A^* expandiert wird, dann gibt es auch ein anderes Problem $I_2 \in \mathcal{I}$, in dem A^* mit Regel $r1$ einen Knoten ausläßt, der durch B expandiert wird. (Mit anderen Worten – keine Schlingen-Auflösungs-Regel in A^* wird strikt dominiert durch ein Element aus $\mathcal{A}$.)

$$(\exists_{I1 \in \mathcal{I}} \exists_{B \in \mathcal{A}} \exists_{r1} : V_{r1}(A^*) \not\subseteq V(B)) \Rightarrow (\exists_{I2 \in \mathcal{I}} : V_{r1}(A^*) \subset V(B))$$

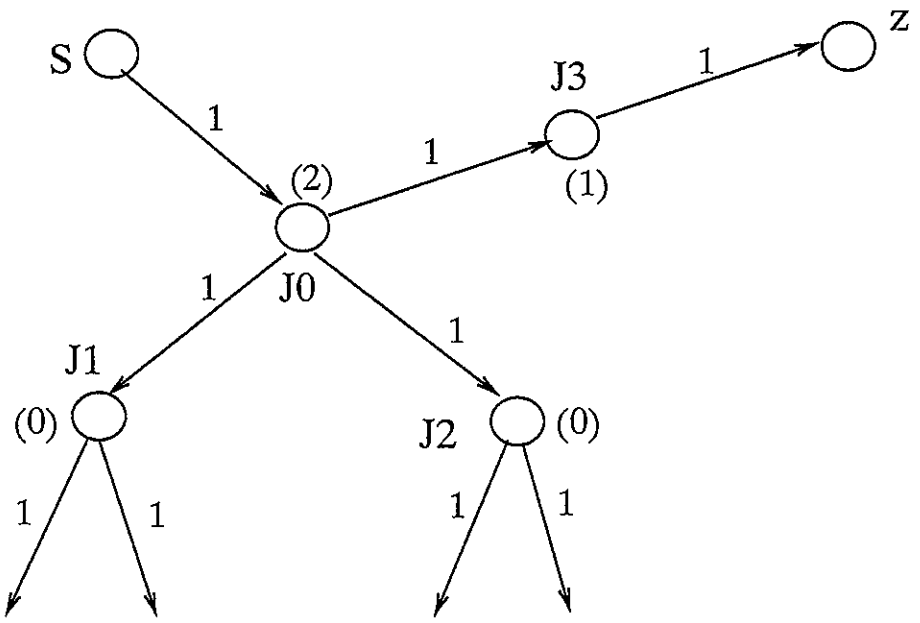
r – eine Schlingen-Auflösungs-Regel im Schritt 3 von A^*

Problemkreis	Klassen von Algorithmen	
	Zulässig wenn $h \leq h^*$ $\mathcal{A}_{zu}$	Global kompatibel mit A^* $\mathcal{A}_{gk}$; Bestensuche $\mathcal{A}_{bs}$
Zulässig $\mathcal{I}_{ZU}$	A^{**} ist 3-optimal kein 2-optimaler Alg.	A^* ist 1-optimal kein 0-optimaler Alg.
Zulässig nichtpathologisch $\mathcal{I}_{\overline{ZU}}$	A^* ist 2-optimal kein 1-optimaler Alg.	A^* ist 0-optimal
Konsistent $\mathcal{I}_{KON}$	A^* ist 1-optimal kein 0-optimaler Alg.	A^* ist 1-optimal kein 0-optimaler Alg.
Konsistent nichtpathologisch $\mathcal{I}_{\overline{KON}}$	A^* ist 0-optimal	A^* ist 0-optimal

Tabelle 1: Die Optimalität von A^* [Dechter & Pearl 88]

Die wichtigsten Ergebnisse von Dechter & Pearl [DEC88] (Tabelle 1):

- A^* ist nicht 1-optimal über zulässige Algorithmen $\mathcal{A}_{zu}$ in jedem Problem mit optimistischer Heuristik. Beispiel: Wir zeigen einen zulässigen Algorithmus bezüglich $\mathcal{I}_{ZU}$, der in manchen Problemen besser



A* : f(J0) = 3, f(J1) = 2, f(J2) = 2, f(J3) = 3
J1 und J2 werden bei jeder Regel expandiert
S -> J0 -> J1 -> J2 -> J3 -> z

A** : f(J0) = 3, f(J1) = 3, f(J2) = 3, f(J3) = 3
Es gibt eine Regel bei der:
S -> J0 -> J3 -> z

Nicht-pathologische Probleme - A** verhalt sich wie A*

Pathologische Probleme - es gibt Regel in A**,
die jede Regel r in A* dominieren

Abb. 2 A** ist 3-optimal bezuglich zulassiger Heuristik

sein kann als A^* mit belibiger Schlingen-Auflösungs-Regel. Sei M ein Algorithmus wie folgt:
Führe eine von rechts-nach-links Tiefensuche durch, aber expandiere n nicht, den linken Nachfolgerknoten von s. Nach systematischer Suche im zentralen und rechten Teil des Graphen gehe zu n über. Stelle fest, ob n auf einem besseren Lösungspfad als bisher festgestellt liegen kann. Wenn ja, dann expandiere n und führe eine komplette Tiefensuche im Unterbaum mit Wurzel n durch.
M ist selbstverständlich zulässig. Er wird die Expandierung von n nur dann auslassen, wenn genügend Information vorliegt um diesen Schritt zu rechtfertigen. Im Graphen in Bild 1.(a) sind als Kantenkosten die Werte $c(v_i, v_j) = g(v_j) - g(v_i)$, und als Knotenkosten in Klammern die Heuristik-Kosten $h(v_i)$ angegeben. M würde viele Knoten auslassen, die durch A^* expandiert werden. A^* expandiert den Knoten J_1 gleich nach s ($f(J_1) = 4$) und dadurch wird auch viele seiner Nachfolger expandieren. M würde J_3 expandieren, dann den Zielknoten γ auswählen, anschliessend J_2 expandieren und anhalten ohne J_1 zu expandieren. Basierend auf der Zulässigkeit von h, M würde feststellen, daß die Schätzung $h(J_1) = 1$ zu sehr optimistisch ist und daß sie mindestens $h(J_2) - 1 = 19$ betragen sollte. Somit könnte J_1 nicht auf einem besseren Lösungspfad als (s, J_3, γ) liegen.
Algorithmus M ist aber selbst auch nicht 1-optimal weil er um J_1 auszulassen, J_2 und J_2 expandieren muß, die bei keiner Schlingen-Auflösungs-Regel in A^* expandiert werden.

Satz 2.1 ([DEC88]): Wenn ein zulässiger Algorithmus M für ein Problem $I \in \mathcal{I}_{\overline{ZU}}$ einen Knoten nich expandiert, den A^* bestimmt expandiert, dann muß M für dieses Problem I einen Knoten expandierer der durch A^* bei jeder Schlingen-Auflösungs-Regel ausgelassen wird.

Satz 2.1 bedeutet, daß kein Algorithmus 1-optimal gegenüber allen zulässigen Algorithmen sein kann bezüglich pathologischer Heuristik $\mathcal{I}_{\overline{ZU}}$.

- A^* is 2-optimal gegenüber $\mathcal{A}_{zu}$ bezüglich $\mathcal{I}_{\overline{ZU}}$.
  Beispiel: Abb. 1.(b) Diese Optimalität gilt nur für nicht-pathologische Probleme. In diesem Beispiel einer pathologischen Heuristik M expandiert den Knoten J_1 nicht, der durch A^* expandiert wird und M expandiert auch keinen Knoten, den A^* auslassen könnte.

- Ein anderer Algorithmus A^{**} dominiert über A^* und ist 3-optimal gegenüber $\mathcal{A}_{zu}$ bezüglich $\mathcal{I}_{ZU}$.
  A^{**} benutzt folgende f-Funktion:

$$f'_{P(s-n)}(n) = \max\{f(n') = g_{P(s-n)}(n') + h(n') \quad | \quad n' \in P_{(s-n)}\} \tag{1}$$

  A^{**} unterscheidet sich von A^* in dem nicht nur der $g + h$ Wert des Knoten n benutzt wird, sondern auch alle $g + h$ Werte für Knoten entlang des Pfades von s nach n berücksichtigt werden. Das Maximum der Funktion $f(n')$ wird dann als $f'_p(n)$ zur Knotenauswahl verwendet.
  z.B. in Abb. 1(a), wenn A^{**} einmal den Knoten J_2 expandiert, dann besitzt J_1 gleich den Wert $f(J_1) = 21$ und die zu ihm führende Kante kommt von J_2.
  A^{**} ist zulässig und in nichtpathologischen Problemen expandiert er dieselbe Menge von Knoten wie A^*. Für pathologische Probleme gibt es Schlingen-Auflösungs-Regel in A^{**}, die jede Schlingen-Auflösungs-Regel in A^* dominieren.

Satz 2.2 ([DEC88]): Für jede Schlingen-Auflösungs-Regel in A^* und jedes Problem $I \in \mathcal{I}_{\overline{ZU}}$, gibt e eine Regel für A^{**} bei der A^{**} eine Untermenge der durch A^* expandierten Knoten expandiert. Es gib ein Problem und eine Regel für A^{**} bei der A^{**} eine richtige Untermenge der durch A^* expandierte Knoten expandiert, unabhängig von der Regel für A^*.

Beispiel: Die Knoten n_1 und n_2 in Abb. 2 werden expandiert bei jeder Schlingen-Auflösungs-Regel in A^*, aber es gibt eine Schlingen-Auflösungs-Regel in A^{**}, die nur den Pfaden P_{s-t} expandiert.

3 Optimale Graphen-Suchverfahren

Die Menge der Knoten V im Graphen G (bei zulässiger Heuristik) wird definiert als:
(a) $s \in V$,
(b) $n \in V$ wenn $\exists$ Pfad P von s nach n : $g(n) + h(n) \leq C^*$ und der Vorgänger von n entlang P ist in V.
Sei $\mathcal{N} = |V|$.

Wenn h **konsistent** ist, dann expandiert A^* höchstens alle Knoten aus V (und nur einmal). Der Suchaufwand ist linear zu $\mathcal{N}$: $O(\mathcal{N})$. Wenn h **nicht konsistent** ist, dann kann jeder Knoten aus V mehrfach expandiert werden (falls G kein Baum ist).

Satz 3.1 ([MAR77]): Für jedes $\mathcal{N}$ existiert ein Suchgraphen $G_{\mathcal{N}}$ mit $\mathcal{N}$ Knoten, positiven Kantenkosten und zulässiger Heuristik ($h(n) \leq h^*(n), \forall n$), für den der Algorithmus A^* einen Aufwand von $O(2^{\mathcal{N}})$ Schritten aufweist.

3.1 Algorithmus B (Martelli)

Die Schritte 1 und 3 von A^* werden ersetzt durch B1 und B3:

B1. bringe s in OFFEN, setze $g(s) \leftarrow 0, f(s) \leftarrow 0, F \leftarrow 0$
B3. wenn Knoten in OFFEN existieren mit $f < F$, wähle aus ihnen den Knoten n mit kleinstem g-Wert. Andernfalls wähle einen Knoten n aus OFFEN mit kleinstem f-Wert und setze $F \leftarrow f(n)$. (Löse Schlingen zugunsten eines Zieles auf). Entferne n aus OFFEN und bringe n nach GESCHLOSSEN.

Der Beweis der Zulassgkeit wäre sehr ähnlich dem Beweis der Zulässigkeit von A^*.

Satz 3.2 ([MAR77]) : Für beliebigen Suchgraphen G mit $\mathcal{N}$ Knoten, mit positiven Kosten und zulässiger Heuristik ($h(n) \leq h^*(n), \forall n$) gilt:

- B führt höchstens $O(\mathcal{N}^2)$ Schritte durch
- wenn A^* und B Schlingen auf dieselbe Weise auflösen, dann expandiert B nicht mehr Knoten als A^*.

Der Satz 3.2 bedeutet, daß der B Algorithmus immer an Stelle von A^* benutzt sein kann mit mindestens derselben Effektivität. B verhält sich wie A^* bei konsistenter Heuristik, weil es dann keinen Knoten gibt mit $f < F$; bei nicht-konsistenter Heuristik B verhält sich im allgemeinen besser als A^*.

3.2 Algorithmus C (Bagchi,Mahanti)

Ähnlich wie A^* nur das Schritte 1 und 3 durch Schritte C1 und C3 ersetzt werden:

C1. bringe s nach OFFEN, setze $g(s) \leftarrow 0, f(s) \leftarrow 0, F \leftarrow 0$
C3. wenn Knoten in OFFEN existieren mit $f \leq F$, wähle aus ihnen den Knoten n mit kleinstem g-Wert. Andernfalls betrachte Knoten in OFFEN mit kleinstem f-Wert, wähle aus ihnen denjenigen Knoten n mit kleinstem g-Wert und setze $F \leftarrow f(n)$. (Löse Schlingen beliebig auf aber immer zugunsten eines Zieles). Entferne n aus OFFEN und bringe ihn nach GESCHLOSSEN.

Bagchi & Mahanti haben das Verhalten von A^*, B und C bei unzulässiger Heuristik untersucht. Die einzige Bedingung ist, daß h nicht-negativ ist.

Zwei Diskriminanten Q und Q_{opt} werden eingeführt:

- Seien P_1, P_2, P_3... Lösungspfade in G. Der **Pfadenmax** von P_i: $Mi = \max_{n \in P_i}\{c(P_i, n) + h(n)\}$, wobei $c(P_i, n)$ die Kosten des Pfades P_i von s nach n sind.

- Der erste **Diskriminant** $Q = \min_{i \geq 1}\{M_i\}$.
- Seien $P_{i1}, P_{i2}, ..., P_{ik}, k \geq 1$ Pfade mit minimalen Kosten in G. Der zweite **Diskriminant** $Q_{opt} = \min_{1 \leq j \leq k}\{M_{ij}\}$.

Die Menge der Knoten V im Graphen G wird bei unzulässiger Heuristik wie folgt definiert:
(a) $s \in V$,
(b) $n \in V$ wenn $\exists$ Pfad P von s nach n : $g(n) + h(n) \leq Q$ und der Vorgänger von n entlang P ist in V.
$\mathcal{N} = |V|$ bedeutet wie früher die Anzahl der Knoten in V.

Für zulässige Heuristik h gilt immer $Q = C^* = f(s)$, aber auch bei unzulässiger kann es vorkommen, daß $Q = C^* = f(s)$.

Wenn $Q = f(s)$, dann finden alle drei Algorithmen A, B und C eine Lösung mit minimalen Kosten. Aber wenn $Q > f(s)$ dann kann solch eine Lösung nicht immer gefunden werden.

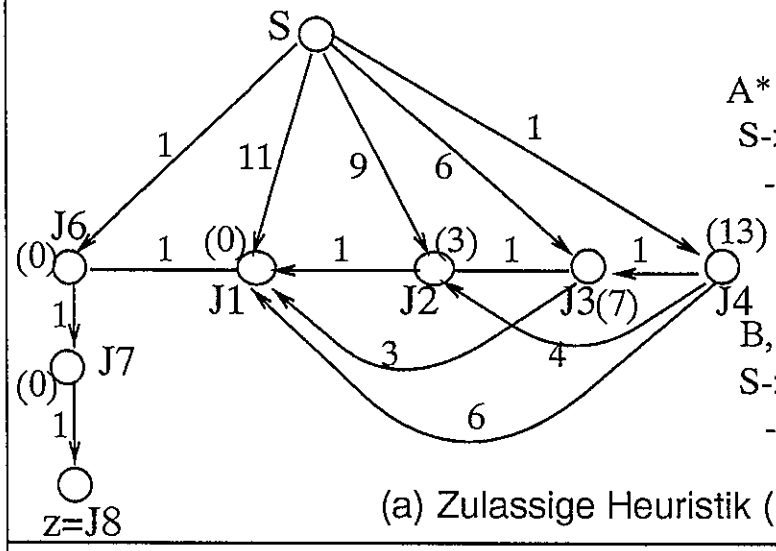
Satz 3.3 ([BAG88]): C expandiert $O(\mathcal{N}^2)$ Knoten im schlechtesten Fall.
B gibt niemals eine bessere Lösung (mit niedrigeren Kosten) als C.

Insbesondere wenn $Q = Q_{opt}$ dann findet C eine Lösung mit minimalen Kosten.

Beispiel:
1. Betrachten wird den Graphen G_8 in Abb. 3(a). Die Heuristik ist zulässig. A führt 18 Knoten-Expandierungen durch, B und C - nur 13:
in A : $s, n_6, n_7, n_1, n_2, n_1, n_3, n_1, n_2, n_1, n_4, n_1, n_2, n_1, n_3, n_1, n_2, n_1$;
in B, C: $s, n_6, n_7, n_1, n_2, n_1, n_3, n_2, n_1, n_4, n_3, n_2, n_1$.
Es gibt also eine Klasse von Graphen für die der Aufwand von A exponentiell wird. B, C bleiben polynomisch.
2. Abb. 3(b) zeigt einen G_9 Graphen. Für diesen Graphen sowohl B, C wie auch A expandieren 8 Knoten: $s, n_7, n_8, n_5, n_4, n_3, n_2, n_1$.
Für einen ähnlichen Graphen mit N Knoten würden B und A nur $(N-1)$ Knoten expandieren. h ist zulässig. Es gibt also Graphen für die A oder B einen linearen Aufwand besitzen.
3. In Abb. 3(c) haben wir eine unzulässige Heuristik. Für diesen Graphen liefern B und C immer die Lösung mit Kosten 3. A kann bei einigen Regeln r die 4 als Ergebnis zeigen. B, C liefern die minimalen Kosten eines Lösungspfades, A demgegenüber nicht immer. Dieses Beispiel zeigt, daß bei unzulässiger Heuristik das Ergebnis von A schlechter sein kann als das von B, C.

C expandiert manchmal mehr Knoten als B. Aber bei folgenden Heuristiken ist C auch in diesem Bezug dem B überlegen.

- Die Heuristik h ist **nicht irreführend** wenn:
  Seien m, n zwei Knoten in G und $m \neq s, n \neq s$. Sei P_1 ein beliebiger Pfad von s nach m und P_2 ein beliebiger Pfad von s nach n. Dann:
  $[c(P_1) + h(m) < c(P_2) + h(n)] \Rightarrow [c(P_1) + h^*(m) < c(P_2) + h^*(n)]$
- Die Heuristik h ist **geeignet** wenn die oben angegebene Bedingung nur durch solche Pfade P_1 und P_2 erfüllt sein muß die zugleich in OFFEN sein können, (d.h. P_1 ist nicht ein Unterpfad von P_2 und umgekehrt).

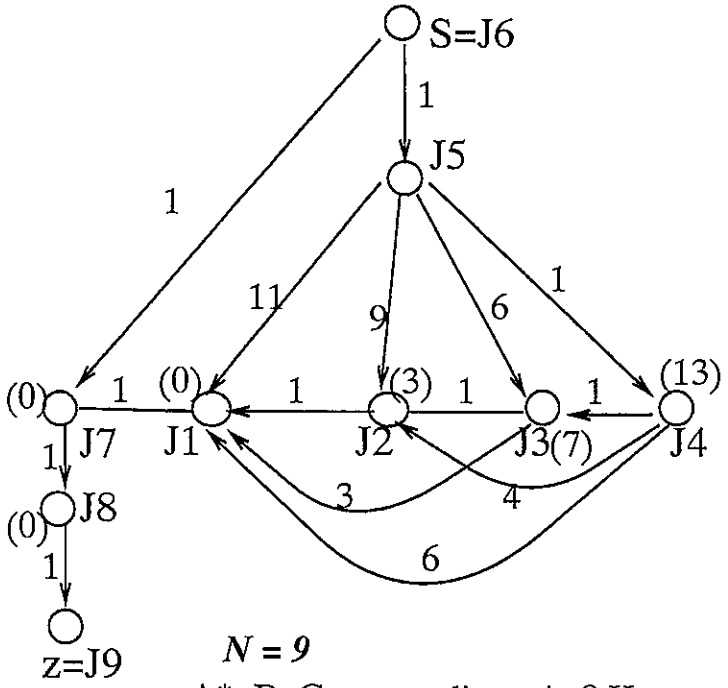


N = 8

A* : expandiert 18 Knoten
S->J6->J7->J1->J2->J1->J3->J1->J2->J1
->J4->J1->J2->J1->J3->J1->J2->J1

B, C : expandieren je 13 Knoten
S->J6->J7->J1->J2->J1->J3->J2->J1->J4
->J3->J2->J1

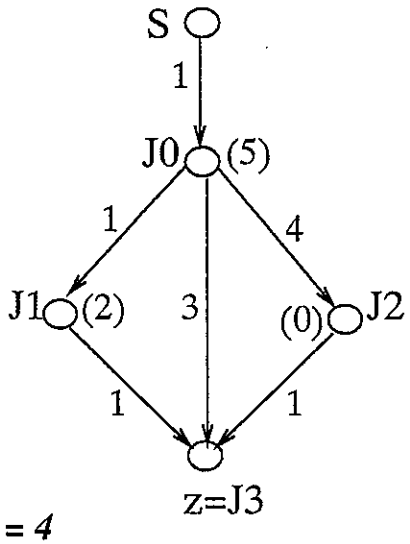
(a) Zulassige Heuristik (nicht konsistent)



N = 9

A*, B, C : expandieren je 8 Knoten
S->J7->J8->J5->J4->J3->J2->J1

(b) Zulassige Heuristik



N = 4

A* : 3 Knoten; nach J0 wird meistens
gleich der Zielknoten gewahlt
S->J0->J3 , aber f(z) = 4

B, C : expandieren je 4 Knoten
S->J0->J1->J3 , f(z) = 3

(c) Unzulassige Heuristik

Abb. 3 Das Verhalten von B, C und A*

Satz 3.4 ([BAG88]):

1. Wenn h *nicht irreführend* ist, dann expandiert Alg. A $O(\mathcal{N})$ Knoten.
2. Wenn die Heuristik h *geeignet* ist, dann:
   - Alg. A expandiert höchstens $O(\mathcal{N}^2)$ Knoten
   - Alg. B expandiert höchstens $O(\mathcal{N})$ Knoten
   - Alg. C expandiert jeden Knoten höchstens einmal und findet immer die Lösung mit minimalen Kosten.

3.3 Algorithmus D (Mahanti,Ray)

Ähnlich wie A^*, nur das Schritte 3 und 6 durch $D3$ und $D6$ ersetzt werden:

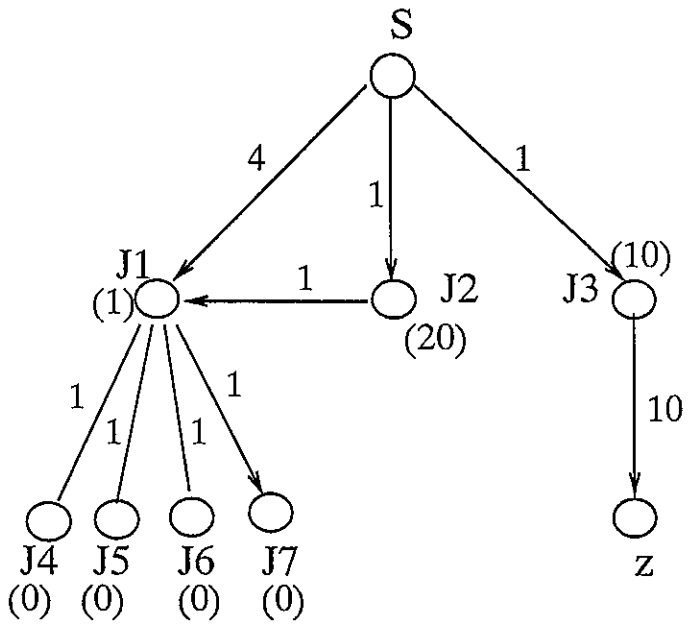
$D3$. betrachte in OFFEN Knoten mit kleinstem f-Wert, wähle aus ihnen den Knoten n mit kleinstem g-Wert. (Löse Schlingen zugunsten eines Zielknoten auf). Entferne n aus OFFEN und bringe ihn nach GESCHLOSSEN.
im Schritt $D5$: ... (wenn n keine Nachfolger hat, setze $h(n) \leftarrow \infty$)
$D6$. FOR jeden Nachfolger n_i von n DO: $g_i \leftarrow g(n) + c(n, n_i)$
$e \leftarrow \min_i\{c(n, n_i) + h(n_i)\}$
IF $h(n) < e$ THEN setze $h(n) \leftarrow e$
ELSE FOR jeden n_i DO: IF $h(n_i) < h(n) - c(n, n_i)$ THEN $h(n_i) \leftarrow h(n) - c(n, n_i)$

Satz 3.5 ([MAH88]): Algorithmus D expandiert jeden Knoten des Graphen $\mathcal{G}$ höchstens sooft wie Alg. C, falls beide Algorithmen dieselbe Schlingen-Auflösungs-Regel anwenden. D findet dieselbe Lösung wie C.

Genauso wie für C kann man Graphen finden, für die D auch mehr Knoten expandiert als B.

Alg.	schlecht. Komplex.	Ergebnis bei zuläs. h	Ergebnis bei unzulässigem h	Bemerkung
A	$O(2^{\mathcal{N}})$	$h(s)$	$f_A \leq Q$	
B	$O(\mathcal{N}^2)$	$h(s)$	$f_B \leq f_A \leq Q$	Letzter F Wert $= Q$
C	$O(\mathcal{N}^2)$	$h(s)$	$f_C = C_{min}$ und $f_C \leq f_B \leq f_A \leq Q$	Letzter F Wert $= Q$ manchmal: $V(C) \supset V(B)$
D	$O(N^2)$	$h(s)$	$f_D = f_C = C_{min}$	$V(D) \subseteq V(C)$

Tabelle 2: Optimale Graphen-Suchverfahren



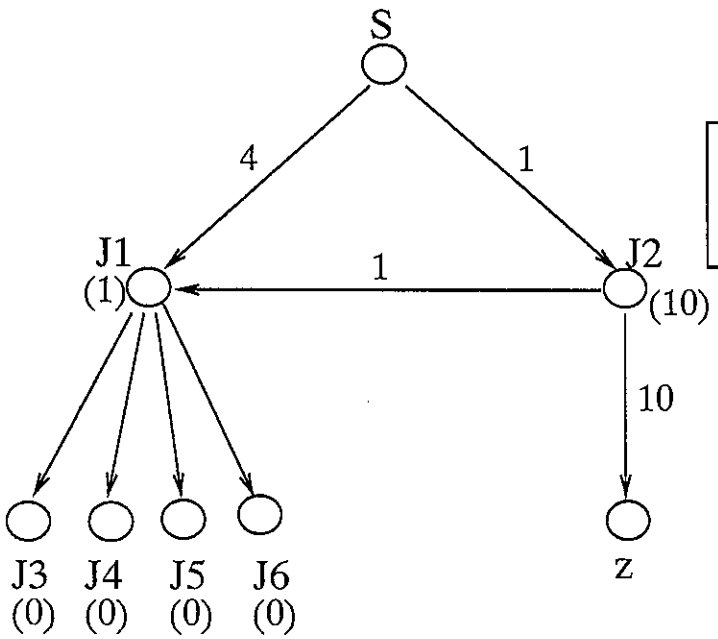
A* :
f(J1)= 4, f(J2)= 21, f(J3)= 11, f(z)= 11
S -> J1 -> J4 ... J7

M :
Tiefensuche ohne linken S-Nachfolger
erst am Ende untersuche J1

S -> J3 -> z -> J2 (Stop)

Da h zulassig sein soll ist
h(J1) = 1 zu optimistisch ,
h(J1) > h(J2) - 1 = 19
Somit f(J1) > 23 und Stop

(a) A* ist nicht 1-optimal bzgl. nichtpathologischer Heuristik



A*: S -> J1 -> J3 ... J6 -> J2 -> z
M: S -> J2 -> z (Stop)

M expandiert J1 nicht und auch
keinen Knoten, den A* auslasst

(b) A* ist nicht 2-optimal bzgl. pathologischer Heuristik

Abb. 1 Ein zulassiger Algorithmus, der manchmal besser ist als A*

4 Sub-optimale Graphensuche

Es ist relativ selten, daß optimale Lösungen für Probleme der realen Welt erwartet werden. Die Komplexität einer Aufgabe kann so hoch sein, daß A^* die optimale Lösung wegen fehlender Ressourcen nicht finden kann. Meistens gibt man sich dann mit **fast-optimalen** Ergebnissen zufrieden.

Wir stellen vier Ansätze zur sub-optimalen Suche vor:

1. Minimierung des *Suchaufwandes* anstelle der *Lösungskosten.*
2. Suche nach einer nur *begrenzt* schlechteren Lösung ?
3. Falls die Heuristik h nur stochastisch modelliert sein kann, Minimierung des *Risikos* anstatt der *Lösungskosten.*
4. *Lokal* optimale Suche.

4.1 Minimierung des Suchaufwandes

Ein Ansatz wäre g und h separiert in die Bewertung einzubeziehen – **gewichtete** Suche:
$f_w(n) = (1-w)g(n) + wh(n)$, wobei für $w = 0, 1/2, 1$ ist f_w kompatibel mit der Funktion f von A^*.

Falls h zulässig ist, dann ist auch f_w zulässig für $0 \leq w \leq 1/2$; kann aber unzulässig werden für $1/2 < w \leq 1$ abhängig davon wie weit h von h^* entfernt ist.

In [BAG88] wurde gezeigt, daß für $0 \leq w \leq 1/2$ die erwartete Anzahl der expandierten Knoten keine polynomische Funktion von N ist, mit Ausnahme besonderer Fälle.
Für Puzzle-Probleme wurde experimentell festgestellt [GAS79], daß f_w am effektivsten beim Erreichen der Zulässigkeits-Grenze $w = 1/2$ ist.

4.2 ϵ-optimale Versionen von A^*

Wir zeigen zwei Verfahren, "Dynamische Gewichte" und A^*_ϵ. Beide garantieren uns, daß die gefundene Lösung t sich von der optimalen Lösung höchstens um Faktor $(1+\epsilon)(\epsilon \geq 0)$ unterscheidet, d.h. $C(t) \leq C^*(1+\epsilon)$.

Dynamische Gewichte ([POH77])

In diesem Vorschlag ist der Wert von w abhängig von der Länge des Pfades:

$$f(n) = g(n) + h(n) + \epsilon \left|1 - \frac{d(n)}{N}\right| h(n)$$

wobei $d(n)$ die Tiefe des Knoten n im Baum bedeutet und N die erwartete Tiefe des Zielknoten ist.

Der Anteil von h vermindert sich mit zunehmender Tiefe: für $d << N$ h besitzt ein Gewicht von $(1+\epsilon)$ (tiefenorientierte Suche); für $d \sim N$ h hat ein Gewicht ~ 1:

Weil für jeden Knoten n' aus OFFEN entlang eines optimalen Pfades gilt $g(n') = g^*(n')$, ergibt sich daraus:

$$\begin{aligned} f(n') &\leq g^*(n') + h^*(n') + \epsilon \left[1 - \frac{d(n')}{N}\right] h^*(n') \\ &\leq C^* + \epsilon h^*(n') \leq (1+\epsilon)C^* \end{aligned}$$

Algorithmus A^*_ϵ ([PEA84])
Zwei Listen werden genutzt - OFFEN und FOKUS.

$$FOKUS = \left\{ n : f(n) \leq (1+\epsilon) \min_{n' \in \text{OFFEN}} f(n') \right\} \tag{2}$$

A^*_ϵ wählt den Knoten aus FOKUS mit dem kleinstem h_F Wert, wobei $h_F(n)$ eine zweite Heuristik ist zur Schätzung des Aufwandes der gebraucht wird um von n ausgehend die Suche abzuschliessen.

4.3 Risiko-zulässige Suche

Algorithmen R^*_δ, $R^*_{\delta,\epsilon}$ - δ-risiko-zulässige Suche ([PEA84])

R^*_δ ist ein Algorithmus, der das Wissen über die Unsicherheit von h ausnutzt um das Risiko einer unzulässigen Lösung zu begrenzen.

Falls e eine Schranke der Überschätzung ist $(h(n) - h^*(n) \leq e)$, dann ist A^* e-optimal $(C(t) - C* \leq e)$.
Es kann vorkommen das eine ansonsten sehr präzise Schätzung, manchmal die h^*-Funktion um ein e überschätzt. Wir wollen Wahrscheinlichkeitsangaben über die Zulässigkeit ermitteln. Nehmen wir an, daß die Unsicherheit der Schätzung die Form einer Wahrscheinlichkeitsdichte $\varrho_{h*}(x)$ besitzt – die relative Wahrscheinlichkeit, daß h^* in der Umgebung von x vorkommt.
Falls g^* bekannt ist (z.B. bei der Baumsuche wo $g^* = g$) führt $\varrho_{h*}(x)$ zu einer Dichtefunktion für $f^* = g^* + h^*$: $\varrho_{f*}(y) = \varrho_{h*}(y - g^*)$.

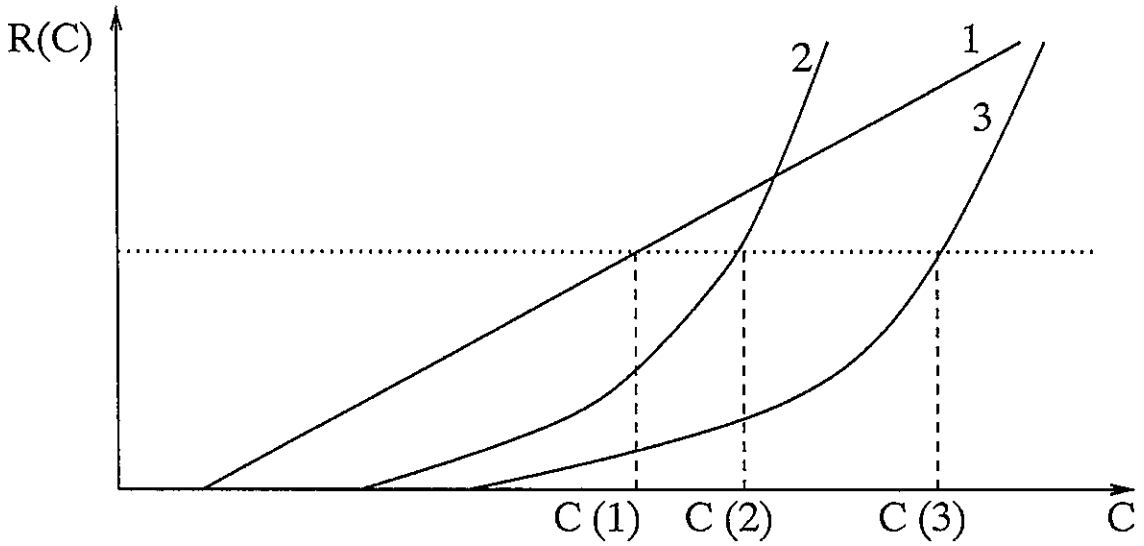
Wenn $g(n) \neq g^*(n)$ und P_{s-n} ist der bisher beste Pfad zwischen s und n dann gilt für einen Lösungspfad der P_{s-n} enthält, $f^+(n) = g(n) + h^*(n)$ mit der Dichte $\varrho_{f+}(y) = \varrho_{h*}(y - g)$.

Wenn die Kosten von Lösungen die P_{s-n} enthalten durch $\varrho_{f+}(y)$ repräsentiert werden, dann kann das Risiko des Nicht-Expandierens eines Knoten durch $R(C)$ charakterisiert werden, das aus $\varrho_{h*}(y)$ berechnet wird. R^*_δ verbindet mit jedem Knoten in OFFEN eine **Kostenschwelle** $C_\delta(n)$, als Lösung der Gleichung $R(C) = \delta$.
$C_\delta(n)$ sind maximal erlaubte Lösungskosten befor das Risiko des Auslassen von n größer wird als δ.

R^*_δ ist identisch mit A^* mit Ausnahme, daß ein Knoten n aus OFFEN mit kleinstem $C_\delta(n)$ zur Expandierung gewählt wird. Die Suche dauert solange bis das Risiko von allen Knoten in OFFEN kleiner wird als δ.

Für $\delta = 0$ ist R^*_δ identisch mit A^*.

Beispiel: Abb. 4 zeigt drei typische $R(C)$-Verläufe für Knoten n_1, n_2 and n_3. Wenn n_1 zur Expansion gewählt wird und es stellt sich heraus, daß es ein Zielknoten ist (mit $h(n_1) = 0$), dann dürfen seine Kosten $g(n_1)$ höchstens $C_\delta(n_1)$ betragen. Das Auslassen von n_2 und n_3 ist erlaubt, weil die mit den Knoten verbundenen Risiken $R_{n_2}[g(n_1)]$ nnd $R_{n_3}[g(n_1)]$ garantiert unter δ liegen. Das ist nicht länger erfüllt, wenn n_2 oder n_3 zur Expandierung ausgewählt werden. Diese haben höchere Kostenschwellen als n_1 auf der Ebene von δ.



Wenn Knoten 1 als Ziel erreicht wurde, dann ist das Auslassen von Knoten 2 und 3 erlaubt, wenn

$h(1) = 0$, $g(1) < C(1)$ und $R2(g(1)) < R3(g(1)) <$

Abb. 4 Beispiele von R(C) Kurven

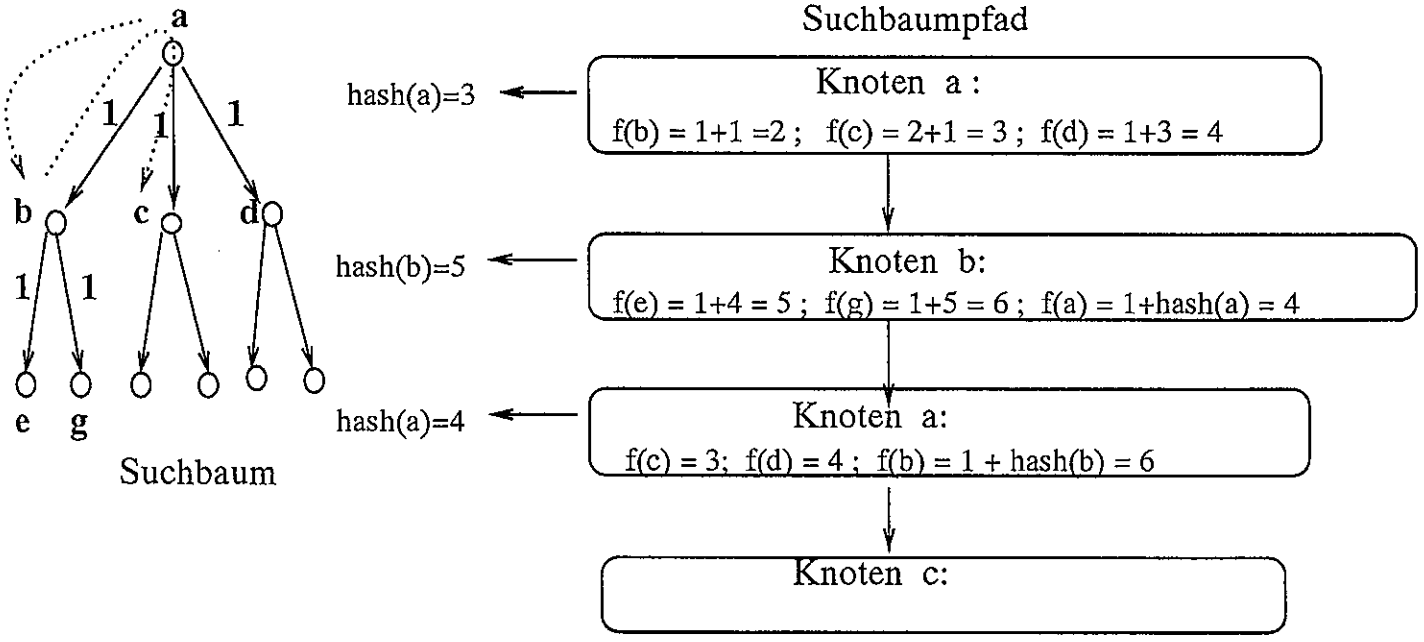


Abb. 5 Ein Beispiel der RTA*-Suche

Satz 4.1 ([PEA84]): Seien R_1, R_2 und R_3 folgende Risiko-Funktionen:

1. R_1 - das Risko des "schlechtesten Falles":

$$R_1(C) \quad = \sup_{y:\varrho_{f^*}(y)>0}\{C-y\} = \quad C-g-h_a$$

   wobei h_a der kleinste h^*-Wert mit positiver Dichte ist.

2. R_2 - Die Wahrscheinlichkeit der sub-optimalen Terminierung:

$$R_2(C) \quad = P\{(C-f^+>0)\} = \int_{y=-\infty}^{C} \varrho_{f^+}(y)dy$$

3. R_3 - das erwartete Risiko:

$$\begin{aligned} R_3(C) \quad &= \quad E[\max\{(C-f^+),0\}] \\ &= \int_{y=-\infty}^{C} (C-y)\varrho_{f^+}(y)dy \end{aligned}$$

R^*_δ ist **δ-risiko-zulässig** bezüglich der Risiko-Funktionen R_1, R_2 oder R_3, d.h. für die Lösungskosten C des Algorithmus gilt immer: $R(C) \leq \delta$ für jeden noch verbleibenden Knoten in OFFEN.

Wir haben bisher die Bedingung gestellt, daß das Risiko unterhalb eines *bestimmten* Wertes δ bleibt. Für die Risiko-Funktionen R_1 und R_2 kann diese Schwelle auch *relativ* zu den Lösungskosten gesetzt werden. Die proportionale Risiko-Schwelle $C'_\delta(n)$ wäre dann die Lösung der Gleichung

$$\frac{R(C')}{C'} \quad = \quad \delta$$

Alle Bedingungen des Satzes 4.1 sind erfüllt, wenn $C'_\delta(n)$ als Auswahlkriterium genutzt wird und $0 \leq \delta \leq 1$.

R^*_δ genauso wie A^* besitzt kein Wissen über den Berechnungsaufwand, der für die Lösung gebraucht wird. Beiden Verfahren haben denselben Nachteil, daß sie viel Zeit brauchen um zwischen vielen Lösungen mit ungefähr gleicher Qualität die beste auszusuchen. Um dem vorzubeugen können wir das Schema aus A^*_ϵ direkt in R^*_δ anwenden. Dabei zahlen wir den Preis, daß das Risiko nur unbeachtlich überschritten sein kann.

Algorithmus $R^*_{\delta,\epsilon}$: A^* mit folgenden Änderungen:
1. Bringe s nach OFFEN. Setze $g(s) \leftarrow 0, f(s) \leftarrow 0, C_\delta(s) \leftarrow 0$.
3a. Ordne Knoten in OFFEN gemäß wachsenden Schwellen $C_\delta(n)$.
3b. Wähle aus OFFEN Knoten n_0 mit kleinstem $C_\delta(n_0)$.
3c. Bilde eine Unterliste FOKUS $= \{n : C_\delta(n) - C_\delta(n_0) \leq \epsilon\}$.
3d. Entferne aus OFFEN Knoten n aus FOKUS der die schnellste Suche verspricht (n besitzt den kleinsten h_F Wert). (Löse Schlingen zugunsten eines Zielknoten auf). Bringe n nach GESCHLOSSEN.

Satz 4.2 ([PEA84]): Für die Risiko-Funktionen R_1 und R_3 ist der Algorithmus $R^*_{\delta,\delta}$ δ-risiko-zulässig (d.h. die Wahl $\epsilon \leftarrow \delta$ führt zum Lösungsrisiko-Wert höchstens δ).

4.4 Lokal optimale Suche

Algorithmus *RTA** ([KOR90])

Es gibt keine Menge OFFEN, der Weg führt immer vom aktuellen Knoten zu seinem besten Nachbarknoten. Dank einer Modifkation der Bewertungsfunktion zählt auch der Vorgängerknoten zu diesen Nachbarn.

Deswegen haben wir es hier mit einer lokal optimalen Suche mit 'Backtracking' zu tun.
Die Bewertungsfunktion für einen Nachbarkoten n ist:
$f(n) = g(n) + h(n)$, wobei $g(n)$ die Kosten des Übergangs vom aktuellen Knoten zum Knoten n sind und $h(n)$ ist die Schätzung der Kosten von n bis zur Lösung (Restkosten für n).
Dabei werden die Restkosten eines bereits visitierten Knoten in einer *Hash* Tabelle gespeichert, weil sie variabel sind und immer beim Verlassen des jeweiligen aktuellen Knoten auf die zweitbeste Bewertung der direkten Nachfolger gesetzt werden. Man verläßt den Knoten in Richtung des besten Nachfolgers und eine Rückkehr wird nur dann sinnvoll, wenn der zweite Nachfolger bessere Lösungsaussichten verspricht, als der erste es zum späteren Zeitpunkt haben wird.

*RTA** braucht einen nur linearen Speicheraufwand.

*RTA** findet eine Lösung wenn:

- der Suchraum endlich ist und
- die Kantenkosten positive Werte besitzen und
- die Heuristikwerte endlich sind und
- ein Zielknoten von jedem Startzustand erreichbar ist (keine 'one-way' Kanten mit 'dead-ends')

Für die Optimalität der Suche gilt folgendes:

- Jede Bewegung von *RTA** in einem *Baum* führt zum Pfad dessen Restkostenschätzung minimal ist bezüglich des jeweiligen kumulierten Suchfrontes (der kumulierte Suchfront besteht aus allen generierten und zugleich nicht-expandierten Knoten seit dem Anfang der Suche).
- *RTA** muß wesentlich modifiziert werden um lokal optimale Entscheidungen für einen generellen *Graphen* mit Zyklen zu treffen.

5 Die Komplexität der A^*-Baumsuche

5.1 Die Komplexität von A^* im schlechtesten Fall

In A^* liegen sowohl das Ziel (Kostenminimierung) wie auch die Heuristik h **quantitativ** vor. Anstatt die Ähnlichkeit zwischen dem Modell und der Realität zu untersuchen, was schwer zu quantisieren ist, können wir den Zusammenhang Modell-Realität operationell spezifizieren, als die Genauigkeit der Kostenschätzung. Je besser das Modell desto besser ist die Schätzung.

Die Differenz $h - h^*$ wird als Fehler (auch Ungenauigkeit) der Heuristik h bezeichnet. Falls h eine präzise Schätzung von h^* ist, dann expandiert A^* ausschlieslich Knoten auf optimalen Lösungspfaden. Ansonsten expandiert A^* jeden Knoten n aus OFFEN für den gilt $g(n) + h(n) < C^*$, wobei C^* die Kosten des besten Pfades sind.
Wenn die Heuristik genau ist ($h = h^*$) dann geht die Suche direkt zum Ziel und nur N Knoten werden expandiert (wobei N die Länge des Pfades ist). Der andere Extremfall ist, wenn keine Heuristik vorhanden ist ($h = 0$) – die Suche wird sehr aufwendig und besitzt eine exponentielle Komplexität.

In [POH77] und [GAS79] wurde festgestellt, daß für den schlechtesten Fall gilt:

$$\begin{aligned} \frac{h(n) - h^*(n)}{h^*(n)} \text{ ist konstant} &\Rightarrow O[exp(cN)] \\ h(n) - h^*(n) \text{ ist konstant} &\Rightarrow O[N] \end{aligned}$$

wobei N die Länge des Lösungspfades ist.

Mit Hilfe der **Wahrscheinlichkeitsanalyse** kann die durchschnittliche Komplexität festgestellt werden :

- Wir wollen herausfinden, wann die durchschnittliche A^*-Suche bedeutend besser wird. Es ist bekannt, daß A^* im Durchschnitt viel besser ist als im schlechtesten Fall.
- Probabilistische Charakteristik der Fehler kann leichter empirisch zu definieren sein als das Bestimmen von Grenzwerten.

5.2 Komplexität bei zulässiger Heuristik

Huyn et al [HUY80] und Pearl [PEA83] nahmen zur Analyse einen homogenen b-arigen Baum T mit bidirektionalen Kanten mit einheitlichen Kosten. Es gibt nur einen Zielknoten γ in der Entfernung N von Startknoten s und ansonsten gibt es nur "außer Kurs" Unterbäume $T_1, ..., T_N$.
Mit $q_{j,k}$ wird die Überlebenswahrscheinlichkeit eines Knoten $n_{j,k}$ der Tiefe k des Baumes T_j bezeichnet. Es wird eine zulässige Heuristik mit positiven Werten angenommen.

In der Wahrscheinlichkeitsanalyse von A^* wird angenommen, daß jede h(n) eine Zufallsvariable im Bereich $[0, h^*(n)]$ ist und wird charakteriziert durch eine Verteilungsfunktion $F_{h(n)}(x) = Prob[h(n) \leq x]$.
Für jede zwei Knoten n_1, n_2 sollen die $h(n_1)$ und $h(n_2)$ unabhängig voneinander sein. Der Fehler $h - h^*$ im Knoten n ist dann nur von $h^*(n)$ und $g(n)$ abhängig und nicht von den Fehlern in anderen Knoten.

Der **relative Fehler** und seine **Verteilungsfunktion** sind gegeben als:

$$\begin{aligned} Y(n) &= \frac{h(n) - h^*(n)}{h^*(n)} \\ F_{Y(n)}(y) &= Prob[Y(n) \leq y] \end{aligned}$$

Es wird angenommen, daß der typische absolute Wert von $h(n) - h^*(n)$ proportional zu der Entfernung $h^*(n)$ wächst, für jeden Knoten n.

Folgendes kann geprüft werden ([HUY80]): für jede Fehler-Verteilung, falls A_2^* **stochastisch mehr informiert** ist als A_1^* (d.h. $P(h_2(n) > x) \geq P(h_1(n) > x)$, $\forall x \in \mathcal{R}$), dann ist A_2^* **stochastisch mehr effizient** als A_1^*.
Ein $F_Y(y)$ für Heuristik h begrenzt von unten die Menge $\{F_{Y(n)}(y)\}$. Wenn die Verteilung eines h' von unten durch $F_Y(y)$ begrenzt wird, dann ist diese Heuristik stochastisch weniger informiert als h und somit, wenn h eine exponentielle Komplexität verursacht, dann tut dieses auch h'.

Als Begrenzung wurde eine stückweise lineare Verteilung genommen (Abb. 6):

$$F_Y(y) = \begin{cases} 1 & y \geq 0 \\ (1-\beta)(1+\frac{y}{\epsilon}) & -\epsilon \leq y \leq 0 \\ 0 & y < -\epsilon \end{cases}$$

mit $0 \leq \beta \leq 1, 0 < \epsilon \leq 1$

Satz 5.1 [PEA84]: Wenn die relativen Fehler in einem einheitlichen m-arigen Baum unabhängig und identisch über $[-\epsilon, 0]$ verteilt sind, dann ist die mittlere Komplexität von A^* wie folgt:

$$E(Z) = \begin{cases} O[exp(cN)] & \text{falls} \quad P(h = h^*) < 1 - 1/b \\ O[N^2] & \text{falls} \quad P(h = h^*) = 1 - 1/b \\ O[N] & \text{falls} \quad P(h = h^*) > 1 - 1/b \end{cases}$$

wobei $\beta = P(h = h^*)$ und c ist eine Konstante ($c > 1$), die nur von $m(1-\beta)$ und ϵ abhängt.

Die Komplexität von A^* reduziert sich zur polynomischen falls $\beta \geq 1 - \frac{1}{b}$. Der Reproduktionsfaktor ist dann $m \leq 1$.

Der Satz 5.1 gilt auch für andere Verteilungen, wenn nur der Fehler proportional mit der Entfernung vom Ziel wächst und eine einzige Verteilungsfunktion alle Knoten beschreibt.

Wann ist die Suche exponentiell wenn Knoten *unterschiedliche* Verteilungsfunktionen besitzen ?

Satz 5.2 ([PEA84]): Falls für jeden Knoten im Baum die Wahrscheinlichkeit, daß der relative Fehler eine fixierten positiven Wert ϵ übersteigt, grösser ist als $1/b$, dann ist die durchschnittliche Komplexität von A^* exponentiell zu N.

Der Beweis basiert auf zwei Beobachtungen:
1. In Reproduktions-Prozessen wo alle Elemente bis zur Tiefe $k < D(j, \epsilon) = \frac{j\epsilon - 1}{2 - \epsilon}$ einen Reproduktionsfaktor $m = bq > 1$ haben, ist die durchschnittliche Familiegröße exponentiell in $D(j, \epsilon)$.
2. Die Tiefe k für die das Argument unter einer gewissen Schwelle $-\epsilon$ liegt ist linear in j.

Der Satz 5.2 impliziert, daß ein exponentielles Wachsen von $E(Z)$ dann vermieden werden kann, wenn der absolute Wert des typischen Fehlers langsamer als linear mit der Entfernung vom Ziel wächst.
Um Fehler zu repräsentieren die belibig steigen (wie $\phi(h^*)$) aber langsamer als linear wird der ϕ-normalisierte Fehler im Knoten n definiert als:

$$Y(n) = \frac{h(n) - h^*(n)}{\phi[h^*(n)]}$$

wobei $\phi(\cdot)$ eine monoton wachsende Normalisierungsfunktion ist und:
$\lim_{x \to \infty} \phi(x) = \infty$, $\lim_{x \to \infty} \phi(x)/x = 0$.

Definition: Wir sagen, daß die Heuristik einen typischen Fehler der Größe $\phi(N)$ induziert, wenn für jeden Knoten im b-arigen Baum es zwei fixierte positive Werte ϵ und λ gibt sowie eine Normalisierungsfunktion

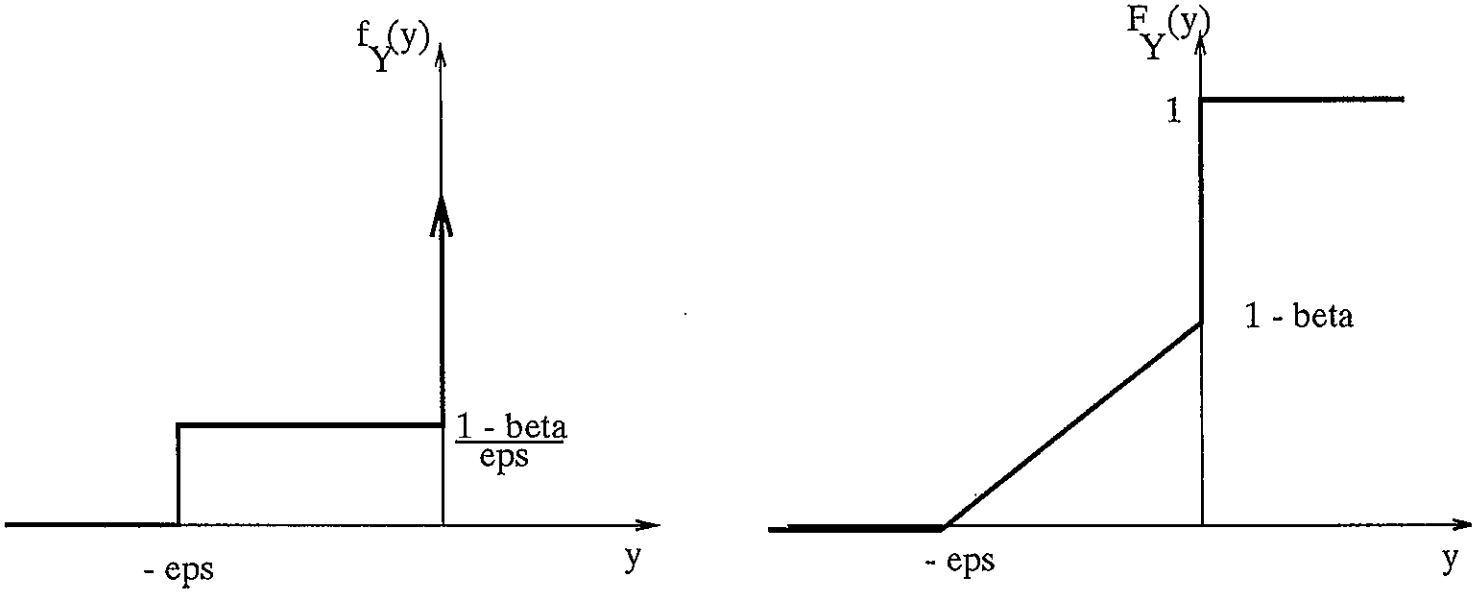


Abb. 6 Eine stuckweise lineare Dichte und Verteilung

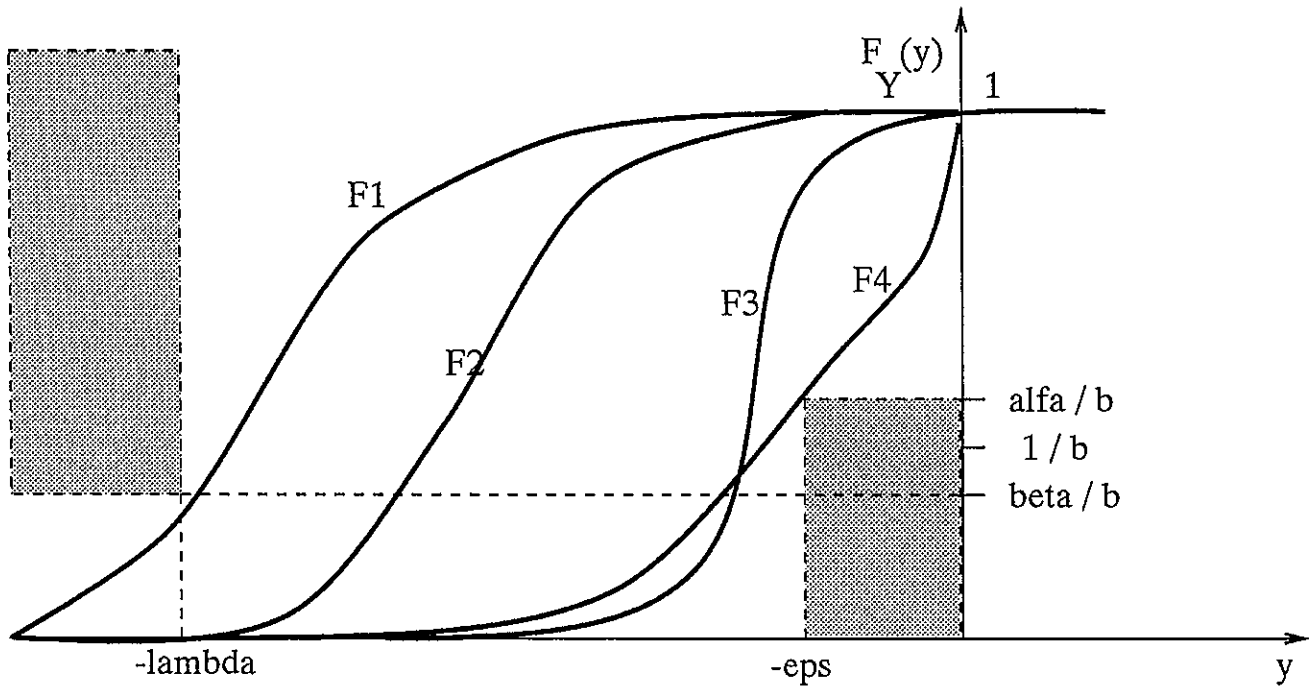


Abb. 7 Beispiele von typischen Fehlern der Ordnung phi(N)

$\phi(\cdot)$ so das:

$$P\left[\left|\frac{h(n)-h^*(n)}{\phi[h^*(n)]}\right| \geq \epsilon\right] \geq \alpha/b, \quad \text{für} \quad \alpha > 1 \tag{3}$$

und gleichzeitig:

$$P\left[\left|\frac{h(n)-h^*(n)}{\phi[h^*(n)]}\right| \geq \lambda\right] \leq \beta/b, \quad \text{für} \quad \beta < 1 \tag{4}$$

Abb. 7 zeigt eine Familie von Verteilungen für typische Fehler.

Satz 5.3 [PEA84]: Wenn für alle Knoten im b-arigen Baum gilt:

- es gibt ein $\epsilon > 0$ und ein$\alpha > 1$, sowie eine Normalisierungsfunktion $\phi(\cdot)$, so das:

$$P\left[\left|\frac{h(n)-h^*(n)}{\phi[h^*(n)]}\right| \geq \epsilon\right] \geq \alpha/b,$$

  dann wird die durchschnittliche Komplexität von A^* von unten begrenzt:

$$E(Z) \geq Nexp\left\{c_1\phi(N)\left[1+O\left(\frac{\phi(\phi(N))}{\phi(N)}\right)\right]\right\}$$

- es gibt ein $\lambda > 0$ und $\beta < 1$, sowie eine Normalisierungsfunktion $\phi(\cdot)$, so das:

$$P\left[\left|\frac{h(n)-h^*(n)}{\phi[h^*(n)]}\right| \geq \lambda\right] \leq \beta/b,$$

  dann wird die durchschnittliche Komlexität von A^* von oben begrenzt:

$$E(Z) \leq Nexp\left\{c_2\phi(N)\left[1+O\left(\frac{\phi(\phi(N))}{\phi(N)}\right)\right]\right\} \tag{5}$$

wobei c_1, c_2 Konstanten sind.

Aus dem obigen Satz ergibt sich, daß wenn der typische Fehler von $h(\cdot)$ der Größe $\phi(N)$ ist und $\lim_{N\to\infty}\frac{\phi(N)}{N} < \infty$, und c eine positive Konstante ist, dann:

$$E(Z) \leq Nexp\{c\phi(N)[1+O(1)]\} \tag{6}$$

Der Zusammenhang zwischen **Präzision** (der Heuristik) und **Komplexität** der A^*-Baumsuche ist "nicht elastisch" – sehr präzise Heuristiken sind erforderlich damit die Komplexität begrenzt bleibt. Wenn der typische Fehler wie $\phi(N) = \sqrt{N}$ wächst, dann wächst die Komplexität von A^* schneller als N^k $(k > 0)$. Eine notwendige Bedingung für polynomischen Aufwand ist eine Heuristik mit logarithmischer Präzision (z.B. $\phi(N) = (logN)^k$). Solche Heuristiken sind schwer zu finden. Die meisten Messwerte unterliegen einem konstanten relativen Fehler und für statistische Inferenzen ist die Fehlergröße $O(N^{1/2})$ (wobei N ist die Anzahl der Zufallswerte).

5.3 Komplexität bei unzulässiger Heuristik – mehrere Zielknoten

Sei $\mathcal{G}$ ein b-ariger Baum mit bi-direktionalen Kanten mit einheitlichen Kosten. Es gibt einen Zielknoten der N Schritte von s entfernt ist und es kann weitere Zielknoten geben in Entfernung $\geq N$.
Die Heuristik brauchtnicht zulässig zu sein. Sei Q der Diskriminant von allen Lösungspfaden.
Für den Baum $\mathcal{G}$ gelten folgende Gleichungen:

- $$F_Q(x) \leq F_Y\left(\frac{x-N}{\phi(N)}\right), \forall x \tag{7}$$

- Sei $a > 0$ eine reale Zahl. Wenn

$$F_Y(x) = \begin{cases} 0, & x \leq -a \\ 1, & x \geq a \end{cases}$$

dann

$$F_Q(x) = 1 \quad \text{für} \quad x \geq N + a\phi(N). \tag{8}$$

Die Komplexität bezüglich drei Formen der Fehler-Normalisierungsfunktion $\phi : \mathcal{R}_+ \to \mathcal{R}_+$ wird untersucht:

1. $\phi(0) = 1$,
2. $\phi(x)$ ist nichtfallend in x,
3. $\phi(x) = \max(1, x)$
   oder $\phi(x) = \max(1, x^\epsilon)$, für gewisses ϵ , $0 < \epsilon < 1$
   oder $\phi(x) = \max(1, \ln px)$ für gewisses $p > 1$

Satz 5.4 [BAG88]: Sei $\phi(x) = \max(1, x^\epsilon)$, $0 < \epsilon \leq 1$. Dann ist $E(Z)$ eine exponentielle Funktion von N wenn $a_1 < \max(0, a_2)$ wobei $\quad a_1 = MIN_{n\in\mathcal{G}}\{x|F_Y(x) \leq 1/b\}, \quad a_2 = MAX_{n\in\mathcal{G}}\{x|F_Y(x) = 1\}$.

Satz 5.4 gilt nicht für $a_1 = a_2 \geq 0$. Wenn $a_1 = a_2 = 0$ dann kann $E(Z)$ von der Strategie der Schlingenauflösung abhängen. Wenn $a_1 = a_2 > 0$ dann ist $E(Z)$ eine lineare Funktion von N.

Sei η_b $(b \geq 0)$ die Anzahl der zyklenfreien Lösungspfade in $\mathcal{G}$ für die $N \leq c(P) \leq N + b\phi(N)$ gilt (d.h. η_0 ist die Anzahl der Lösungspfade von Länge N).
Sei η_G die gesamte Anzahl der zyklenfreien Lösungspfade in $\mathcal{G}$.

Satz 5.5 [BAG88]: Wenn η_G eine polynomische Funktion von N ist und $\phi(x) = \max(1, \ln px)$ $(p > 1)$, dann ist $E(Z)$ eine polynomische Funktion von N.

Seien: $\eta_0 = e^{\beta N}$ $(\beta > 0)$; $\phi(x) = \max(1, \ln px)$ $(p > 1)$. Dann ist $E(Z)$ eine exponentielle Funktion von N wenn $a_1 < \max(0, a_2)$, wobei: $a_1 = MIN_{n\in\mathcal{G}}\{x|F_Y(x) \leq e^{-\beta}\}$, $a_2 = MAX_{n\in\mathcal{G}}\{x|F_Y(x) = 1\}$.

Der Satz gilt auch für Normalisierungsfunktionen der Form $\phi(x) = \max(1, (\ln px)^r)$, wobei $p > 1$ und r ist eine positive ganze Zahl.
Zusammenfassend ist $E(Z)$ eine exponentielle Funktion von N in nicht-degenerierten Fällen, mit Ausnahme wenn ϕ logarithmisch ist und die Anzahl der Zielknoten eine polynomische Funktion von N ist.

5.4 Hybride Verfahren – optimale Suche und Tiefensuche

Die Komplexität von teilweise informiertem "backtracking" wurde auch untersucht in [PEA83]. Solch ein Verfahren unterscheidet sich von gewöhnlicher Tiefensuche dadurch, daß der "beste" Nachfolgerknoten des gerade expandierten Knoten gewählt wird. Die Tiefensuche dauert bis zur gewissen Tiefe, z.B. N, bis zurückgesprungen wird. Die Komplexität dieser Suche beträgt:

$$E(Z_B) \geq \frac{1}{2}(1-\beta^2)b^{N-1}\left[1+O\left(\frac{1}{\phi(N)}\right)\right]$$

$E(Z_B)$ ist immer exponentiell; unabhängig von der Qualität der Heuristik,Sowohl $\phi(\cdot)$ wie auch $F_Y(\cdot)$ haben nur geringen Einfluß auf die Komplexität durch $(1-\beta^2)$.

Backtracking hat aber auch Vorteile gegenüber A^*. Es ist leichter zu implementieren und braucht bedeutend weniger Speicher. Die Nachteile sind nur von Bedeutung bei Optimisierungsproblemen. Wenn es viele gleich wichtige Lösungen gibt, dann kann ein hybrides Verfahren – optimale Suche und Tiefensuche sehr effektiv sein.
Zwei zulässige Algorithmen mit linearem Speicherbedarf:

- **IDA^* (iterative-deepening) Korf [KOR85]**

  IDA^* ist eine Iteration von Tiefensuche mit wachsender Kostenschwelle, bei derer Überschreitung, die dafür verantwortliche Suchbaumkante abgeschnitten wird (Abb. 8). Diese Schwelle entspricht den minimalen Kosten der vorher abgeschnittenen Knoten. In jeder Iteration werden die Knoten von neuem angelegt. IDA^* ist zulässig, expandiert ungefähr dieselbe Anzahl von Knoten wie A^* und hat nur linearen Speicherbedarf.

  Z.B. In einigen Puzzle-Problemen IDA^* löste alle Aufgaben in durchschnittlich 30 CPU Minuten und generierte 1.5 Millionen Knoten/Minute. A^* löste keine Aufgabe, denn nach der Generierung von 30000 Knoten war kein Speicherplatz mehr vorhanden.

- **MA^* Chakrabarti et al. [CHA89]**

  Als Parameter der Suche dient der (über das Minimum) verfügbare Speicherplatz MAX. Es kann gezeigt werden, daßs MA^* niemals mehr Speicher braucht als MAX+$|C^*|$, wobei $|C^*|$ die maximale Pfadlänge über solche Pfade P ist, für die gilt $f(P) \leq C^* = h^*(s)$.
  $h(n,m)$ ist die Restkostenschätzung für Knoten n, berechnet entlang des unmittelbaren Nachfolgerknoten m.
  Bei der Restkostenschätzung für einen Nachfolgerknoten m des Knoten n wird die Idee des **Pfadenmax** angewendet:
  $\forall q \in Nachfolger(m)$: IF $h(m,q) < h(n,m) - c(n,m)$ THEN $h(m,q) \leftarrow h(n,m) - c(n,m)$
  Dadurch wird eine monotone Heuristik erzwingt.

  Die wesentlichen Schritte von MA^* sind (siehe Beispiel in Abb. 9):

  1. Der beste Knoten mit dem minimalen $f(n)$-Wert wird zur Expandierung ausgewählt.
  2. Nur ein einziger Nachfolgerknoten m des aktuellen Knoten n, für den $h(n,m)$ minimal ist, wird generiert. Falls noch weitere Nachfolger zu generieren sind, n bleibt in der Menge OFFEN bis es soweit ist und seine Bewertung $f(n)$ wird modifiziert mit der Restschätzung entlang des besten bisher nicht generierten Nachfolgers:
     $f(n) \leftarrow g(n) + hf(n)$; $hf(n) \leftarrow \min_m\{h(n,m)\}$ für $m \in Nachfolger(n)$ und m noch nicht generiert.
     Die Bewertung $F(n)$ entspricht der in A^* üblichen Kostenfunktion und bleibt nach der Expandierung unverändert:
     $F(n) \leftarrow g(n) + hF(n)$; $hF(n) \leftarrow \min_m\{h(n,m)\}$ für alle $m \in Nachfolger(n)$.
     Für den Nachfolgerknoten m werden die Werte $h(m,q)(\forall q \in Nachfolger(m)), f(m)$ und $F(m)$ berechnet.

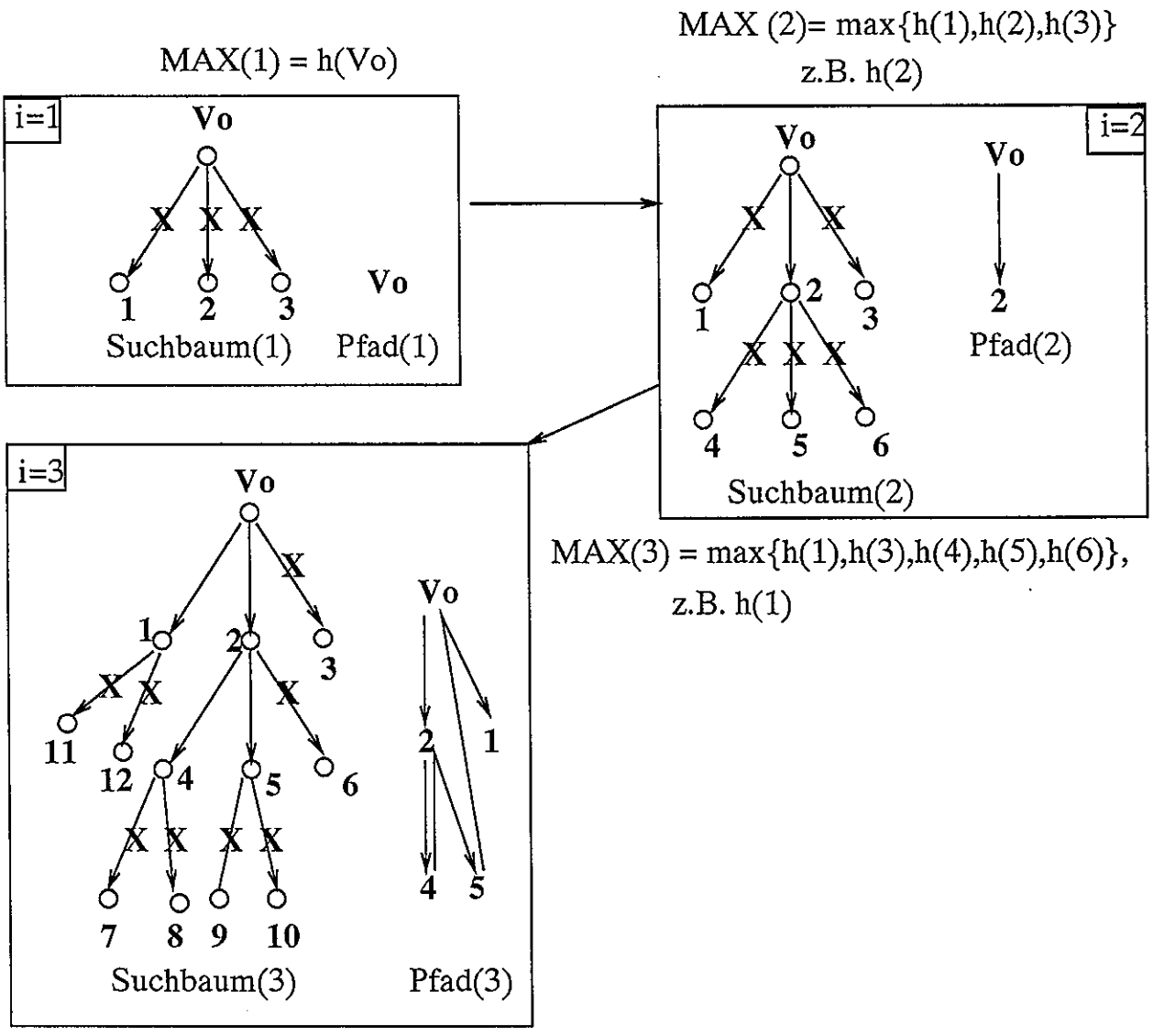


Abb. 8 Drei Iterationsschritte im IDA * -Verfahren

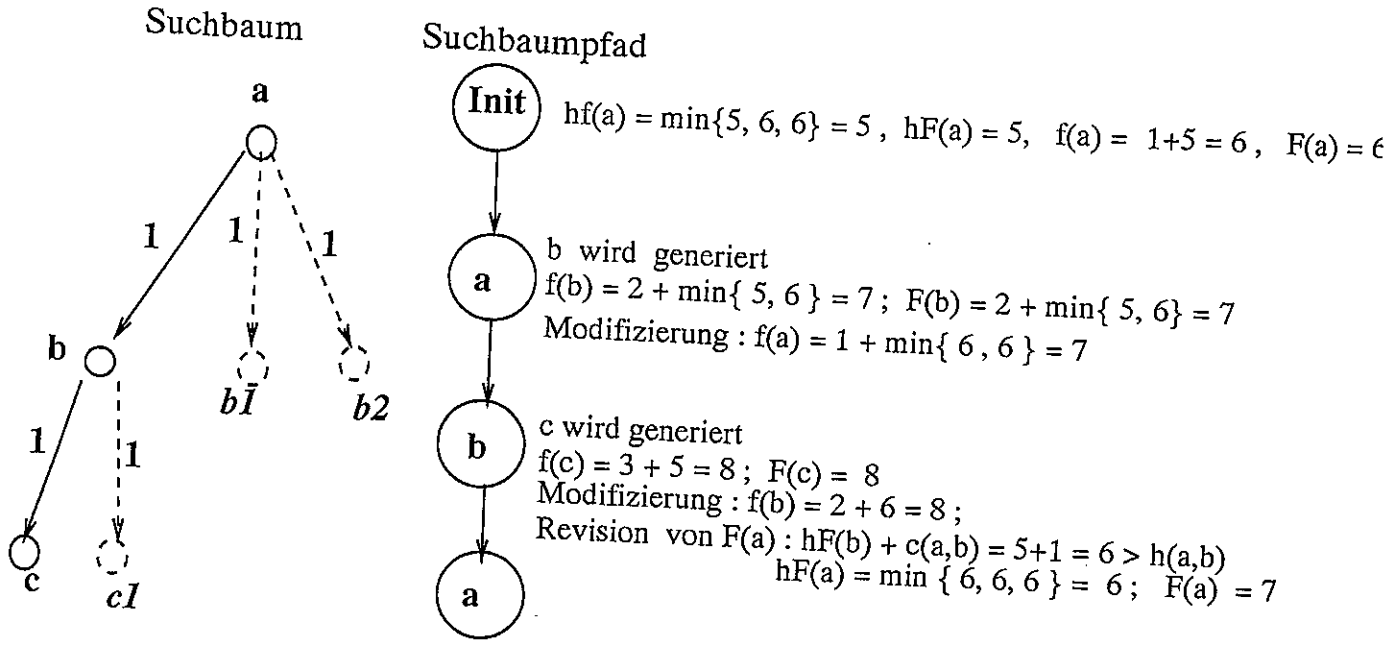


Abb. 9 Das MA*-Suchverfahren

Planungsphase : Systematische Suche mit MINIMIN-Pruning

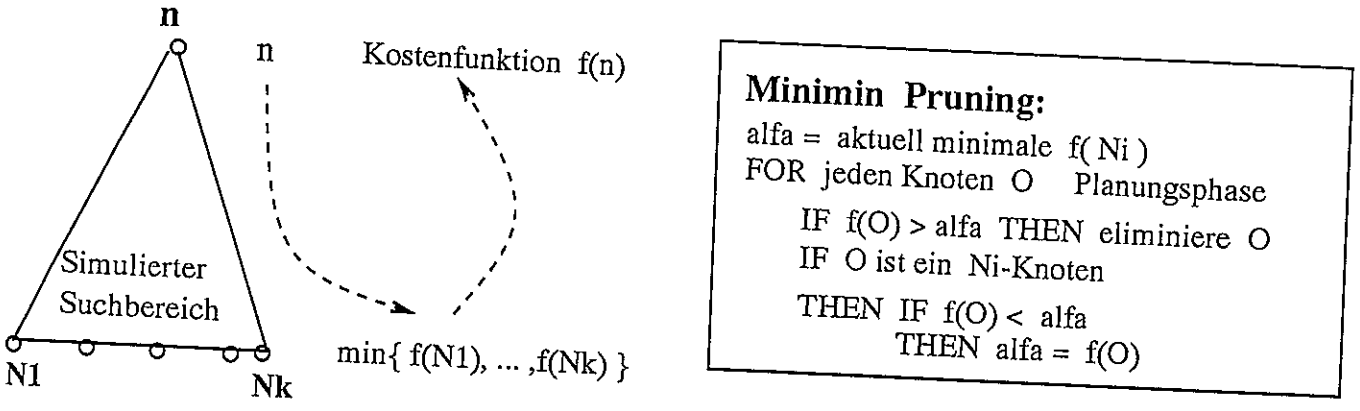


Abb. 10 Planungsphase der Zwei-Phasen-Suche

3. Eine bottom–up Kostenrevision der F–Werte von allen Vorgängern n des aktuellen Knoten m solange diese sich ändern:
   (i) Bringe m in die Menge Z (am Anfang leer)
   (ii) REPEAT bis Z leer ist:
      (a) entferne m aus Z
      (b) sei n der Vorgänger von m
         IF $[hF(m) + c(n,m) > h(n,m)]$
         THEN $h(n,m) \leftarrow hF(m) + c(n,m)$; berechne $hF(n), F(n)$; bringe q in Z

4. Wenn die Anzahl der generierten Knoten eine vorgegebene Grenze MAX überschreitet, wird ein Blatt r mit den größten Kosten $F(r)$ aus OFFEN entfernt. Wenn der Vorgängerknoten in GESCHLOSSEN ist wird er in die Menge OFFEN gebracht.

Anhand von zwei Randexemplaren kann die Komplexität von MA^* beurteilt weden – MAX $= \infty$ und MAX $= 0$. Im ersten Fall arbeitet MA^* ähnlich wie A^* mit der Ausnahme, daß nur ein Nachfolgerknoten auf einmal generiert wird. Mit MAX$= 0$ MA^* wird zu einem "iterative-deepening" Verfahren ähnlich dem IDA^*. In MA^* werden aber die $h(n,m)$-Werte zur Auswahl des besten Nachfolgers herangezogen, anders als in IDA^* wo eine systematische Auswahl erfolgt. Deswegen wird erwartet, daß MA^* mit MAX$= 0$ weniger Knoten expandiert als IDA^*.
Damit können wir feststellen, daß MA^*(MAX) eine Brücke zwischen IDA^* und A^* bildet. Ein Speicher-zur-Komplexität Austausch ("tradeoff") sollte damit möglich sein.

Zur "worst-case" Analyse nehmen wir einen einheitlichen $b-arigen$ Baum mit einheitlichen Kantenkosten. Das Ziel liegt in Entfernung C^* von der Baumwurzel. Es gibt keine Heuristik-Komponente (ständig Null).
Sei $k = \lfloor \log_b MAX \rfloor$. Bis zur Tiefe k findet somit kein Pruning statt und es werden bis dahin $\sum_{i=1}^{k-1} b^1$ Knoten generiert. Auf der Tiefenebene k werden b^k neue Knoten generiert und es kommt zum Pruning. Nur MAX $(= b^k)$ Knoten verbleiben dann. Höchstens b^k Knoten müssen für die nächste Tiefenebene regeneriert werden. Damit werden auf der Ebene $k+1$ $b^k + b^{k+1}$ Knoten generiert anstatt b^{k+1}.

Die gesamte Anzahl von generierten Knoten bis zur Lösung beträgt

$$b + b^2 + ... + b^{k-1} + (C^* - k + 1)b^k + (C^* - k)b^{k+1} + ... + b^{C^*}.$$

MA^* mit MAX extra Speicher spart O(MAX) Knoten gegenüber $M^*(0)$. Die Ersparnis für $k = \lfloor \log_b MAX \rfloor$ beträgt:

$$= \frac{b^k(C^* - k + 1)}{b-1} + \frac{b^2(b^{k-2} - 1)}{(b-1)^2} - \frac{(C^* - 1)b}{b-1} = O\left(\frac{b^k(C^* - k + 1)}{b-1}\right) = O(\text{MAX})$$

$$(C^* - 1)b + (C^* - 2)b^2 + ... + (C^* - k + 1)b^{k-1}$$

Obwohl beide A^* und $MA^*(0)$ $O(b^{C^*})$ Knoten expandieren, spart A^* zusätzlich $O(b^{C^*})$ Knoten gegenüber $MA^*(0)$.

Eine Kombinierung der systematischen Suche mit optimaler Suche schlägt Korf vor in Form der *Zwei-Phasen*-Suche [KOR90]:
A^* muß den ganzen Lösungspfad durchgehen befor er feststellt, daß selbst die erste Bewegung richtig ist. Solange kein optimales Ziel gefunden ist, kann keine Optimalitätsgarantie für bisherige Entscheidungen gegeben werden. Im Gegensatz dazu steht die Zwei-Spieler Suche, wo ein begrenzter Horizont analysiert wird um eine sub-optimale Bewegung zu ermöglichen. Durch Adaptierung dieser Vorhegensweise für die optimale Suche bekommen wir zwei Suchmodi:

1. **Planungsphase** – ein *Minimin–Lookahead* Operator
   In diesem Schritt wird eine Simulation der Nachfolgerknoten bis zu einer vorgegebenen Tiefe im Suchraum vorgenommen um eine genauere, optimistische Restschätzung für den aktuellen Knoten zu gewinnen. Es werden immer das Minimum der Kosten von Knoten aus dem Suchhorizont zurückgeliefert.

2. **Verarbeitungsphase** – Nach einer kompletten Planungsphase wird die aktuell beste Bewegung gemacht z.B. Auswahl des besten Knoten aus OFFEN (A^*). Der Grund warum nicht gleich zum besten Knoten aus dem Suchhorizont übergegangen wird ist die Annahme, daß in nächster Planungsphase, bei Erweiterung des Suchhorizonts ein anderer Pfad als optimaler erscheinen wird.

Die *MiniminLookahead* Planungsphase für A^* wurde vorgeschlagen in [ROS72]. Dadurch können wir nun von einer adaptierbaren Heuristik sprechen. Die Schätzung wird besser. Die Suche selbst bleibt global optimal und auch der Speicheraufwand kann in Praxis exponentiell sein, sollte aber kleiner sein, da die Schätzung besser wird.

Bezüglich der Komplexität solcher Zwei-Phasen-Suche kann man eine Dualität zwischen der Tiefe der Planungsphase und Länge des Lösungspfades feststellen. Die optimale Tiefe hängt von der Relation zwischen dem Aufwand für die Berechnung eines Knoten und für den Übergang im Suchraum (Anlegen der Nachfolger und Auswahl). Diese beiden Aufwände sind problemabhängig. Eine tiefere Simulation während der Planungsphase führt zu kürzeren Lösungspfaden während der Verarbeitungsphase.

Falls die Kostenfunktion f *monotonisch* bzw. h *konsistent* ist, dann kann das α – *Pruning* in der Planungsphase angewendet werden:
während der Planungsphase behalte in der Variabel α den kleinsten f-Wert eines Knoten aus dem Suchhorizont; beende einen Pfad vorzeitig, wenn sein f-Wert nicht kleiner ist als α.

6 Die Basiskontrolle für wissensbasierte Signalinterpretation

6.1 Drei elementare Suchprozesse

Im Gegensatz zur Spiel–Problematik gilt für die Signalinterpretation mit ERNEST folgendes (Abb. 11):

1. Der Zustandsraum ist nicht gleich dem Suchbaum; beide sind dazu nur implizit gegeben.
2. Die Bewertung hängt primär vom Inhalt des Knoten ab und nicht vom Übergang im Suchbaum.
3. Die Kosten der Inferenzen sind hoch im Vergleich zu den Kosten der Suchbaumerweiterung.

Die alternierende Kontrolle [KUM91] enthält eine Bestensuche, die eine spezielle Variante von A^* ist:

- Es genügt, einen Suchbaum $\mathcal{T}$ im Gegensatz zu einem allgemeinen Suchgraphen zu verwenden.
- Die Menge der Zielknoten, sowie der gesamte Suchbaum ist a priori nicht bekannt – die Heuristik ist nur implizit.
- Die Bewertung eines Knotens erfolgt in Form eines Vektors, der die Bedingungen des A*-Algorithmus mit einer statistisch eingrenzbaren „Unzulässigkeit" erfüllt.

Innerhalb des homogenen Suchbaumes werden drei Klassen von Suchproblemen gelöst:

1. Baumsuche nach bester Instanzen-Sequenz von ständig abstrakteren oder spezialisierteren Zielkonzepten im Modell
2. UND/ODER Graphensuche nach bestem Prämissengraph für die Instantiierung des aktuellen Zielkonzeptes
3. Baumsuche nach bester Instanzenmenge für einen Prämissengraph

Das Problem (1) ist auch das generelle Ziel der Analyse: das Finden einer Instanz eines Zielkonzepts, die die Signaldaten ausreichend abdeckt. Der momentane Zustand der Suche in diesem Raum wird durch das aktuelle Zielobjekt repräsentiert. Das Suchproblem (1) ist übergeordnet zu (2) und (3) indem für jede Instanz im Suchraum (1) je ein Suchraum (2) und (3) durchsucht werden muß. Die untergeordneten Suchprobleme (2) und (3) sind ineinander verzweigt und der momentane Zustand wird in Form des expandierten Modells festgehalten.

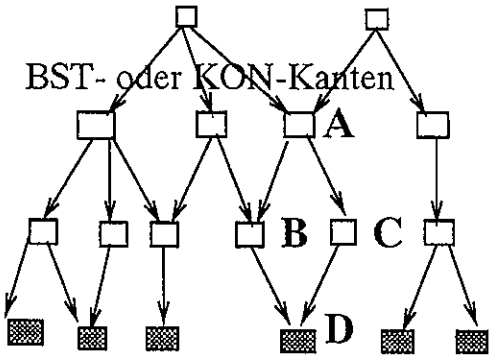
6.2 Komplexitätsanalyse

Die **Länge** N des optimalen Pfades im Suchbaum hängt vor allem von drei Parametern ab:

- Strukturierung des Modells – Anzahl der Hierarchie-Ebenen, Strukturierung der Konzepte
- Anzahl der Daten
- Abbruchbedingung – prozentuale Abdeckung der Daten

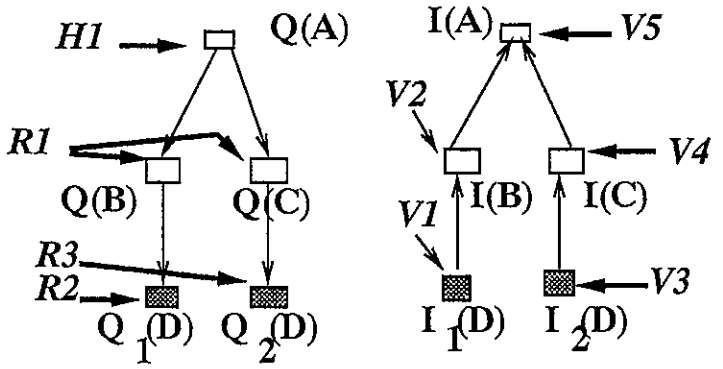
Der **Aufspaltungsfaktor** des Suchbaumes hängt von folgenden Parametern ab:

- Anzahl von übergeordneten Zielkonzepten bei jedem Übergang zu neuem Zielkonzept
- Reihenfolge der Zielkonzepte auf dem Pfad – eine schneller Übergang zur sicherem Zielkonzept auf hoher Abstraktionsebene ermöglicht eine bessere Heuristikfunktion h
- Sicherheit des Inferenzschrittes – umgekehrt proportional zur Anzahl von konkurrierenden Instanzen bzw. modifizierten Konzepten
- Anzahl der Modalitätsmengen



Minimale KONZEPTE

MODELL

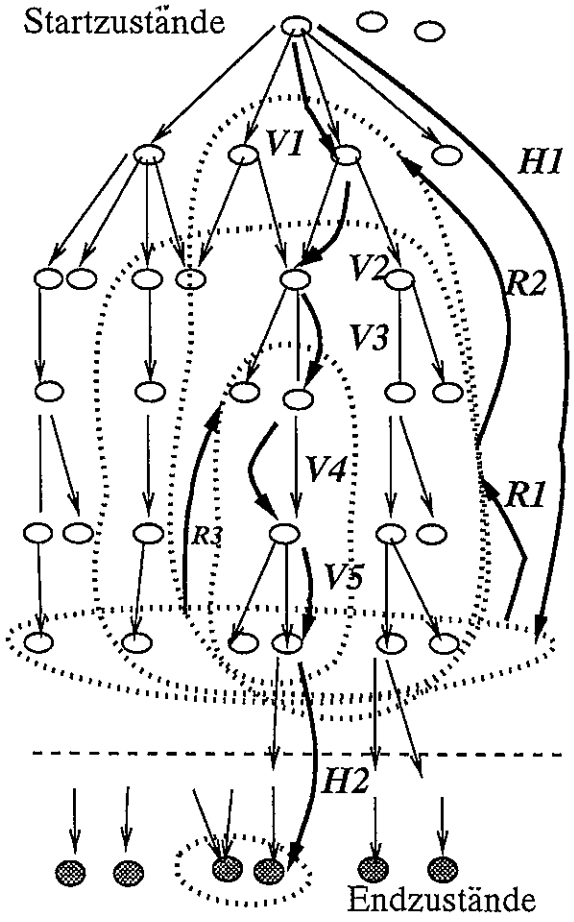


Mod. Konzepte (Q) - in Schritten Hi und Ri

Instanzen (I) - in Schritten Vi

INFERENZEN und ERGEBNISSE

STRATEGIE im Zustandsraum



SUCHE im Entscheidungsbaum

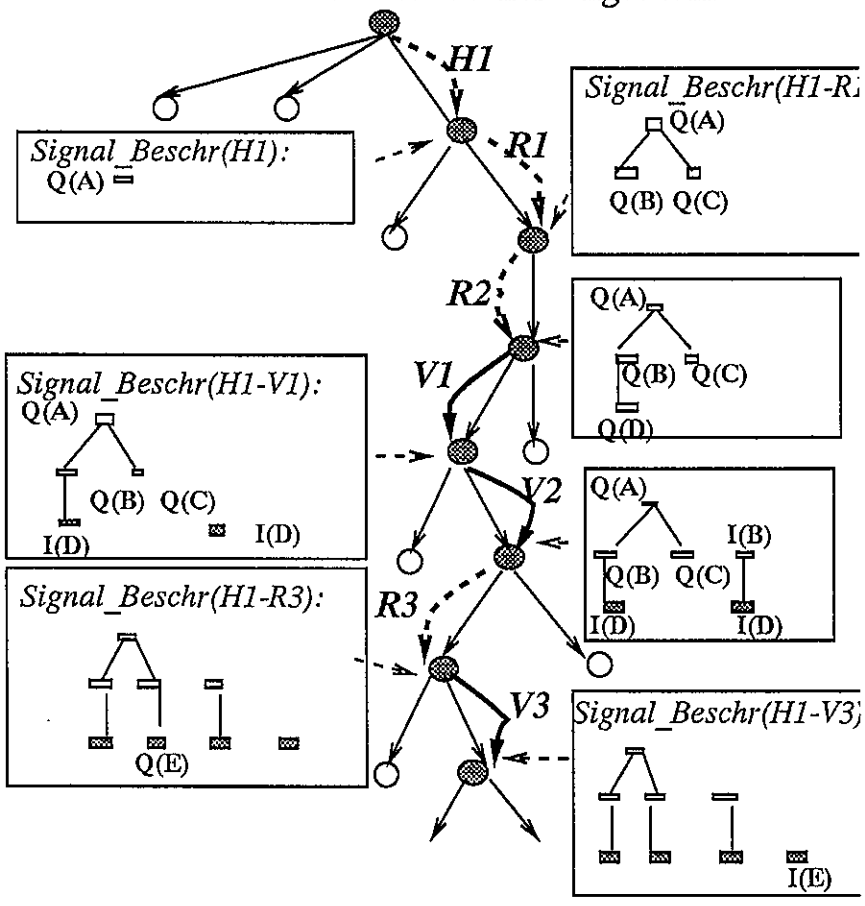


Abb. 11 Zustandsraum und Suchbaum fur Signalinterpretation

Seien:
D – die Anzahl der Daten
ϕ – die Prozentrate für das Abbruchkriterium
$H = H_{kon} + H_{spez}$ – die maximale Anzahl von Hierarchieebenen im Modell
b – die maximale Anzahl von Teilen bzw. Spezialisierungen im Modell
B – die maximale Anzahl der direkten Nachfolgerknoten im Suchbaum

Die Extremwerte für die Länge des Lösungspfades wären (Abb. 12):

- die minimale Anzahl von benötigten Instantiierungen und Modifizierungen:
  $N_{min} \geq 2 * \{\lceil \phi D \rceil + (H-1)\}$, unter der Bedingung, daß $\lceil \phi D \rceil \leq b$. ($\lceil x \rceil$ ist ein Operator, der die ganzzahlige Begrenzung des x von oben liefert.

  Die für den Abbruch zulässige Anzahl von Daten wird in einem Schritt durch eine übergeordnete Instanz abgedeckt. Anschliessend bis zum Erreichen der endgültigen Zielinstanz werden nur Konzepte mit einem Teil instantiiert.

- die maximale Anzahl von Instantiierungen und Modifizierungen:
  $N_{max} \leq 2 * \{\lceil \phi D \rceil * H + (B-1) * (H-1)\}$ bzw.
  $\leq 2 * \{\lceil \phi D \rceil + D * (H-1)\}$ wenn $B - 1 \geq D - \lceil \phi D \rceil$

Für die Wahrscheinlichkeitsanalyse würden wir ϕD fixieren und versuchen sowohl N wie auch B (und die Reproduktionswahrscheinlichkeiten $p_{i,j}$ von Suchraumknoten) als Funktion der Bewertung von Instanzen und mod. Konzepten des Knoten auszudrücken.

6.3 Bewertungsmaße für Signalanalyse

Es wird eine *initiale Bewertung* der Segmentierungsergebnisse vorausgesetzt. Bei vielen Segmentierungsverfahren ist es möglich, ein *Abstands-* oder *Zuverlässigkeitsmaß* als initiale Bewertung zu verwenden. Eine weitere Möglichkeit besteht darin, das Abstandsmaß durch heuristische Funktionen in *normierte Sicherheiten* zu verwandeln. Solche Sicherheiten lassen sich aber nicht nur aufgrund von berechneten *Abständen* bestimmen. Sie können auch aus *Attributwerten* abgeleitet werden.

Beispiele:

1. [SAG90]
   Prinzipiell werden zwei verschiedene Bewertungsparadigmen getrennt – statistische Maße und heuristische Maße. Als einzelne Bewertungsmaße werden betrachtet:

   - Die Zulässigkeit $Z(H)$ einer Hypothese in Bezug auf die sprachliche und inhaltliche Kompetenz des Systems.
   - Die akustische Qualität $q(H)$ einer Hypothese.
   - Die Sicherheit $s(H)$ einer Hypothese.
   - Die pragmatische Priorität $pP(h)$ einer Hypothese.
   - Die semantische Priorität $pS(A_i^M)$ und $pS(A_i^{(I)})$ für spezielle modifizierte Konzepte bzw. Instanzen.

   Bis auf die semantische Priorität genügen all diese Maße zumindest näherungsweise der Forderung des A*-Algorithmus nach optimistischer Schätzung.

   Jede Bewertung $f(v_i)$ eines Suchraumknotens v_i ist durch die Bewertung $\phi(\text{ZIEL}(v_i))$ gegeben:

   $$\phi(O_l) = (Z(O_l), q(O_l), s(O_l), pP(O_l), sP(O_l)) \tag{9}$$

   Der Vergleich zwischen zwei Bewertungsvektoren erfolgt gemäß ihrer lexikographischen Ordnung.

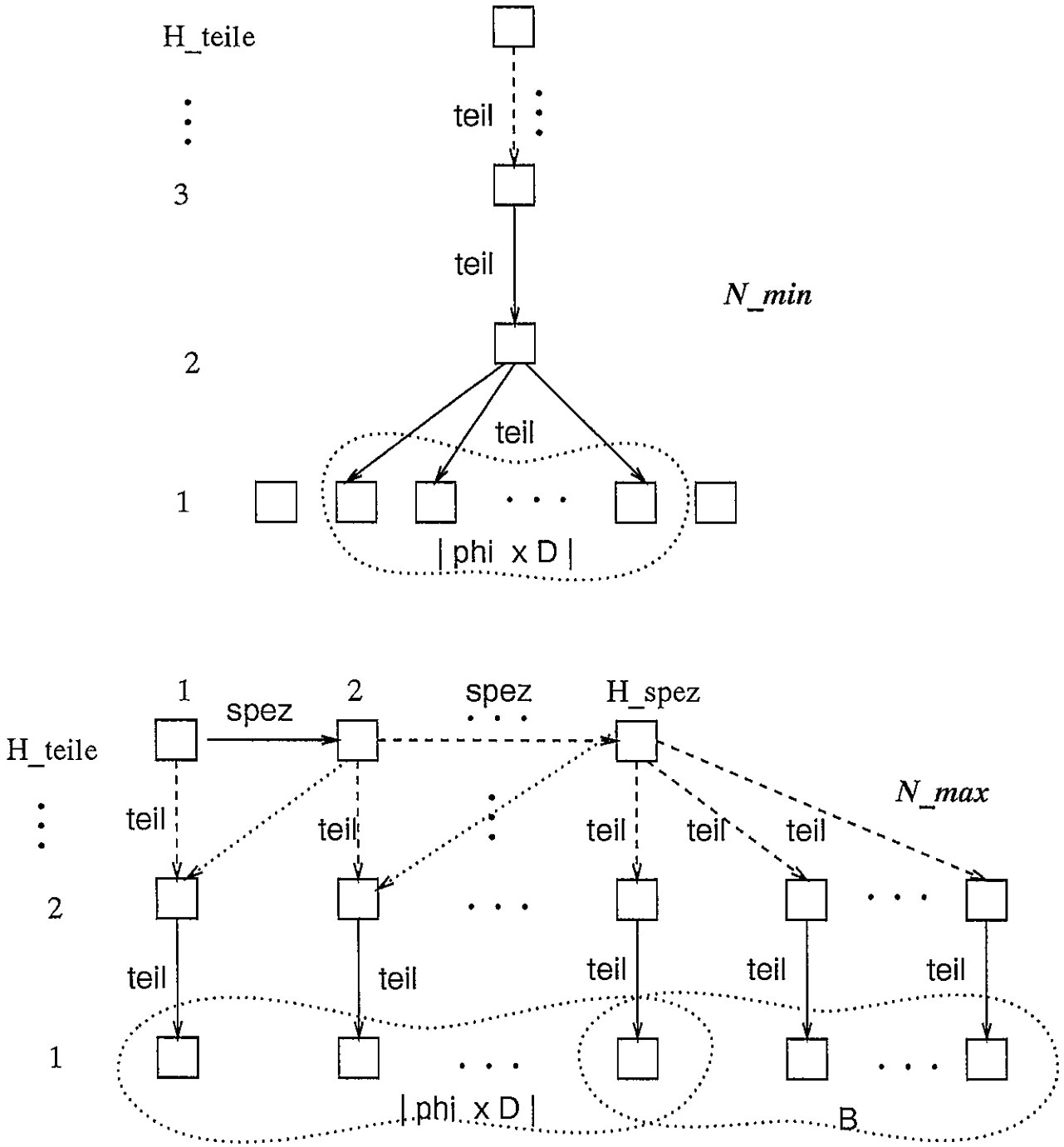


Abb. 12 Grenzwerte für die Länge des Lösungspfades

2. [NIE90]
   Es wird zwischen der *Kompatibilität* und der *Sicherheit* einer Zuordnung unterschieden. Die Kompatibilität gibt die Qualität der Korrespondenz wieder. Die Sicherheit basiert auf dem Bildsegment, z.B. auf der Linienstärke eines Liniensegments. In erster Komponente f_1 des Bewertungsvektors wird die Kompatibilität und in zweiter f_2 die Sicherheit abgespeichert. Beide Maße basieren auf den Attributwerten von Objekten in Bezug auf die Attribute eines Modellobjektes.

   Die dritte Komponente f_3 des Bewertungsvektors beinhaltet das Ergebnis des Relationtests. Sie gibt wieder, wie gut die referierten Attribute die Strukturrelation eines Konzepts erfüllen.

   Die Kombination der Bewertungen f_1, f_2 und f_3 erfolgt durch die Berechnung des Minimum von allen Werten.

   Die ganzzahligen Komponenten i_i des Bewertungsvektors werden benutzt um den strukturellen Unterschied zwischen dem Modellobjekt und den Segmentierungsdaten darzustellen. In der Komponente i_1 wird die gesamte Anzahl der Bestandteile und Konkretisierungen der Instanz gespeichert. Komponente i_2 beinhaltet die Anzahl der Instanzen für Bestandteile und Konkretisierungen, die in den Segmentierungdaten enthalten sind. i_3 ist die Anzahl der Default Instanzen.

   Ein Pfad S_i wird als besser als ein Pfad S_j bezeichnet, wenn die Kompatibilität f_1 von S_i größer ist als die von S_j, wenn die Anzahl der zugeordneten Teile i_2 von S_i größer ist als die von S_j, und wenn die Anzahl der substituierten Instanzen i_3 von S_i kleiner ist als die von S_j.

7 Ein bipartielles Bewertungsschema für Bildinterpretation

Der Problemraum bei der Signalanalyse entsteht durch das Aufeinandertreffen von Modell und Segmentierungsdaten. Die Bewertung wie sie bisher für die alternierende Kontrolle für Sprachanalyse vorgeschlagen wurde trägt diesem kooperativen Entstehen des Problemraumes in mindestens zwei Fällen keine Rechnung:

- Sie ist grundsätzlich auf Qualität ausgerichtet und vernachlässigt den Aufwand. Wir sind aber sowohl an der besten Abdeckung der Daten wie auch an einer schnellsten Lösung interessiert.
- Die Restschätzung bezieht sich nur auf die noch zu instantiierenden Daten. Die Restschätzung für das Modell berücksichtigt die noch zu instantiierenden Konzepte für das aktuelle Zielobjekt und nicht wie gefordert für die Instanz eines globalen Zieles.

7.1 Die Kostenfunktion:

$$\begin{aligned} f = g + h &= (g_a + g_r) + (h_a + h_r) \\ \text{wobei } A &= g_a + h_a - \text{Aufwandkosten (Modell-bedingt)} \\ R &= g_r + h_r - \text{Risikokosten (Daten-bedingt)} \end{aligned}$$

- g_a – Kosten der bisherigen Inferenzen (Modifizierungen und Instantiierungen von Konzepten) auf dem Suchpfad
- g_r – Das zu erwartete Risiko der Instanz des aktuellen Zielkonzepts ($\sim$ Sicherheit der Instanz, Wichtigkeit der abgedeckten Bilddaten).
- h_a – Der minimale Restaufwand für das Erreichen des globalen Zieles
- h_r – Der minimale Qualitätsverlust der Abdeckung von restlich erforderten Segmentierungsdaten.

Es wird darauf hingeweisen, daß:

- die Bewertungen ähnlich wie der ganze Problemraum nur *implizit* gegeben sind. Sie werden erst während der Analyse in Form von Bewertungen eines jeden Suchbaumknoten berechnet.
- meistens ist eine Bewertung unabhängig von der Bewertung des vorangegangenen Suchbaumknoten – jede Bewertung hängt vom Inhalt ab und nicht direkt vom Suchpfad. Die Kantenkosten sind somit sekundär gegeben als die Substraktion zweier Knotenbewertungen ($c(v_i, v_j) = g(v_j) - g(v_i)$).

7.2 Der Bewertungsvektor

Für die Bewertung ist in ERNEST eine Datenstruktur vorgesehen, die aus je 5 Gleitkomma- und Ganzzähligen- Variablen besteht:

```
#define JUDGEMENT_SIZE    5
typedef struct {
          float fbewwert[JUDGEMENT_SIZE];
          long  ibewwert[JUDGEMENT_SIZE];
          } ERNEST_JUDGEMENT;
```

Für die Sprachanalyse wurde die Bewertungsstruktur wie folgt belegt:

```
fbewwert[0] - Qualitaet;
fbewwert[1] - Sicherheit;

ibewwert[0] - Anzahl ueberdeckter Frames;
```

```
ibewwert[1] - Anzahl Ketten;
ibewwert[2] - Zulaessigkeit;
ibewwert[3] - Pragmatische Prioritaet;
ibewwert[4] - Laenge der maximalen Wortkette;
```

Ein für unsere Untersuchungen geeigneter Bewertungsvektor sollte vor allem die Kostenfunktion und ihre Komponenten enthalten. Dadurch werden wir in der Lage sein, die Komplexität der Suche als Funktion der Heuristik bzw. des Qualität-Aufwand Verhältnisses zu untersuchen.

$$[\ f(\text{Gesamtkosten}), g, h, R(\text{Risiko}),\ g_r,\ A(\text{Aufwand}),\ g_a, 1-S,\ 1/W]$$

- $f = g + h = R + A$ – die additive Bewertungsfunktion
- R – das Risiko der globalen Zielinstanz (umgekehrt zur *Qualität*)
- S – die zu erwartete Sicherheit der Instanz des aktuellen Zielkonzepts
- W – Wichtigkeit – drückt den Anteil der Objekte am endgültigen Erfolg der Suche aus

Das Risiko: $R = c(1 - Q)$

- Die **Qualität** Q ist eine Funktion der Sicherheit des Zieles und der Wichtigkeit von abgedeckten bzw. nicht abgedeckten Segmentierungsdaten:

$$\begin{aligned} Q &= \left[S[Ziel(n)] * \tfrac{Daten(n)}{SegmDaten}\right] + \left[\tfrac{RestDaten(n)}{SegmDaten}\right] = [Q_g] + [Q_h] \\ g_r &= c(1 - Q_g) \\ h_r &= c(1 - Q_h) \end{aligned}$$

  wobei $0 \leq Q \leq 1$ und c ist eine Konstante.

$$\begin{aligned} Daten(n) &= \sum_{MinInst \in DATEN(n)} W(MinInst) \\ SegmDaten &= \sum_{MinInst \in SEGM} (S(MinInst) * W(MinInst) \\ RestDaten(n) &= \max[0, (p\%) * SegmDaten - Daten(n)] \end{aligned}$$

- Die **Sicherheit S** eines Objektes O (Instanz, mod. Konzept):

$$S(O) = \left[\min_{i \in \{Kon(O) \cup Bst(O)\}} S(i)\right] * Kompatibilität[(Attr \cup Str\_rel)(O, \tau(O))]$$

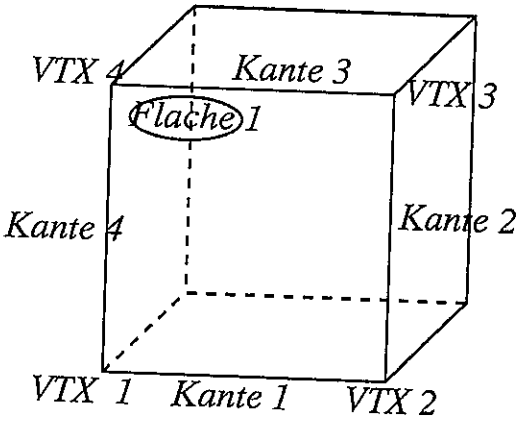
  wobei $\tau(O)$ steht für das Konzept von O.

- Die **Wichtigkeit** des Zielobjekts gibt an wie selten ein Konzept des Modells ist. Sie wird definiert als das Minimum von Wichtigkeiten der Teile bzg. angewendeter Modalitätsmengen im untergeordneten Instantiierungsbaum sowie der Wichtigkeit des Konzepts selbst:

$$W(Ziel(n)) = \min\{W[\tau(Ziel(n))], \{W(i) | \forall i \in Inst\_pfad(Ziel(n))\}\}$$

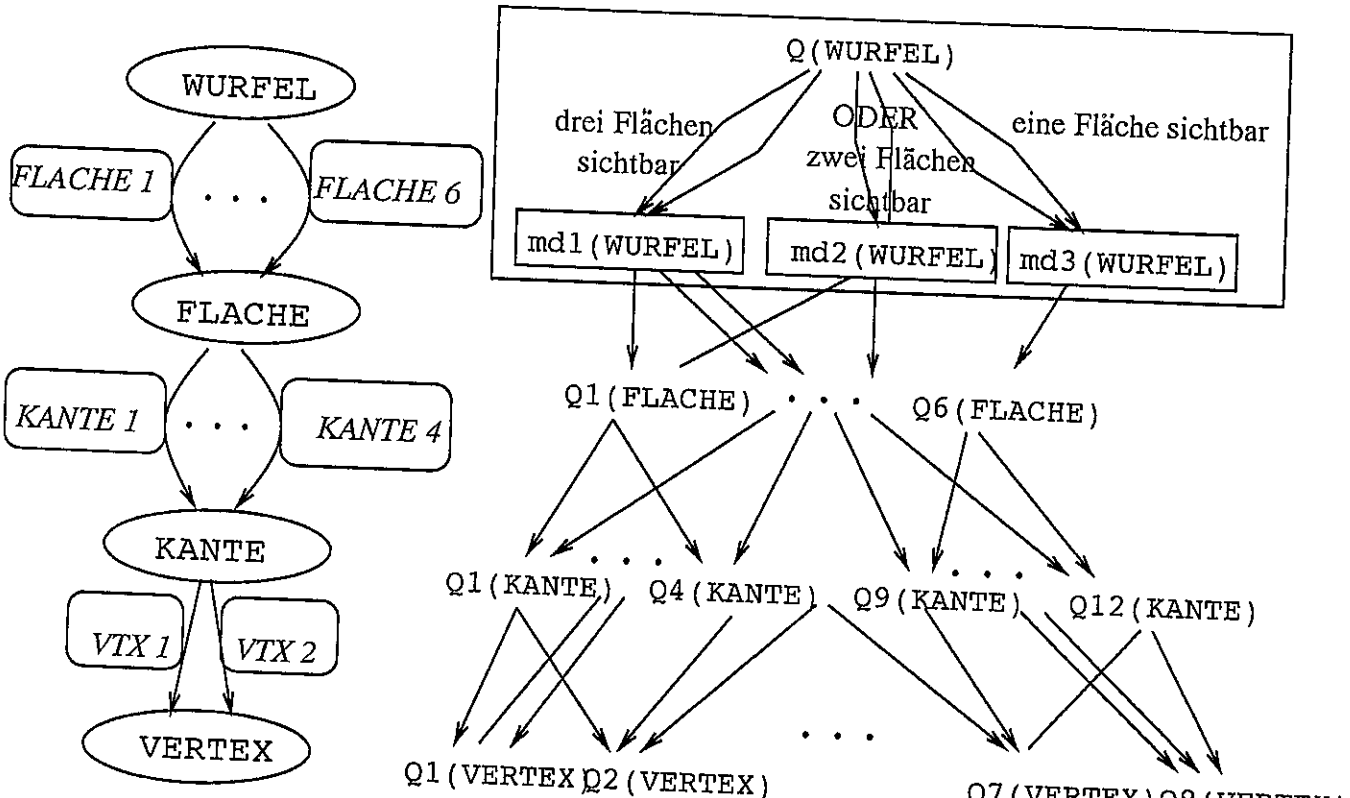
7.3 Syntaktischer Daten–Test

Wir können durch Bewertungen auch sog. "nogoods", d.h. nicht terminierende Pfade feststellen.
Beispiel: Wir spezifizieren für jedes Konzept seine minimalen Erfordernisse an Segmentierungsdaten. Wir können zum Beispiel feststellen, daß im expandiertem Modell eines Konzepts A unabhängig von den Modalitäten und Variationen der Expandierung immer k Objekte für ein minimales Konzept B enthalten sind. Dies bedeutet, daß k Instanzen von B in den Segmentierungsdaten erforderlich sind um eine Instanz von A zu generieren. In Abb. 14 sind für die Instantiierung des Konzepts `WÜRFEL` immer mindestens vier Instanzen des minimalen Konzepts `VERTEX` erforderlich.
In der Bewertungsprozedur kann uberprüft werden ob die Summe der minimalen Erfordernisse für Objekte des Suchbaumknoten durch die noch nicht interpretierten Segmentierungsdaten erfüllt sein kann. Wir testen nicht die Beschränkungen (oder semantische Konsistenz), weil dies einer vollen top–down Expandierung mit Modifizierung bedarf. Die syntaktische Konsistenz wird lediglich überprüft, d.h. ob noch genügend Daten gegebenen Typs vorhanden sind.



Konzept	MIN_Daten
WURFEL	4 x VERTEX
FLACHE	4 x VERTEX
KANTE	2 x VERTEX

MINIMALE Daten-Erfordernisse



MODELL

Expandiertes Modell für WURFEL

Abb. 13 Syntaktischer Daten Test

8 Eine sub-optimale Baumsuche nach Zielinstanzen

Wir befassen uns mit dem Suchproblem (1) der alternierenden Kontrolle, d.h. dem Bestimmen einer besten Sequenz von Zielinstanzen. Eine Voraussetzung für das Verfahren besteht darin, daß zwei Tabellen für **Kostenschwellen** definiert werden:

1. **MIN[i,j]** – enthält die aus Erfahrung resultierenden minimalen Kosten des Übergangs zwischen Instanzen der Zielkonzepte aus Abstraktionsebenen **i** und **j**.

2. **MAX[i,j]** – enthält solche Kosten dieser Übergänge, die von uns als maximal erlaubt definiert sind

Die Prozedur *Schliesse_aus(N)* wird jeweils bei Übergängen im Problemraum (1) aufgerufen, zwischen mehr abstrakten bzw. mehr spezialisierten oder erweiterten Zielkonzepten.

Schliesse_aus(v):	
$I \leftarrow$ *Abstrakt_Ebene*(*Zielinstanz*(v))	
IF $\forall v_i \in Nachfolger(v) : g(v_i) > g(v) + MAX[I, Abstrakt\_Ebene(Ziel(v_i))]$	
THEN RETURN	
$J \leftarrow \max\{j = Abstrakt\_Ebene(Ziel(v_i))\mid\ g(v_i) \leq g(v) + MAX[I,j]\ \}$	
$MAX\_Schwelle = g(v) + MAX[I,J]$	
trage v als Grund in *SUSP.Ident* ein	
FOR alle $n \in$ OFFEN – { v } DO:	
$K = Abstrakt\_Ebene(Ziel(n))$, v_i ist $Min\_Nachfolger(J)$	
IF	$K < J$ AND $g(n) \geq g(v_i)$ - $MIN[K,J]$
THEN	eliminiere n aus OFFEN und referiere in SUSP.Knotenliste
SUSP.Schwelle = *MAX_Schwelle*	

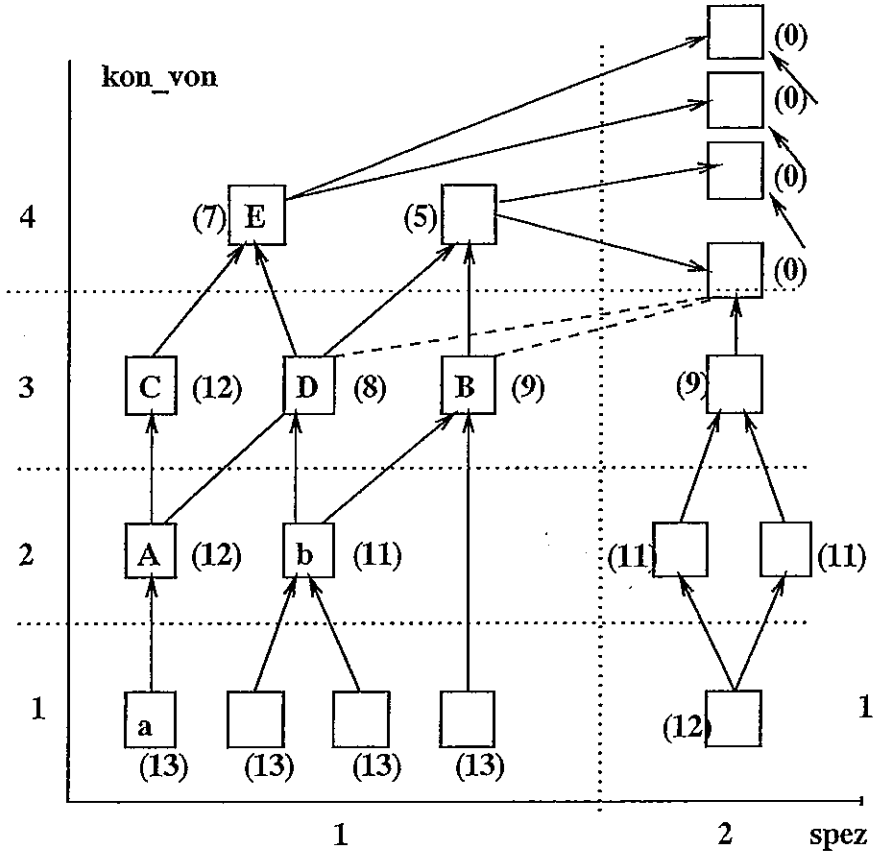
Falls im Laufe der Analyse kein Knoten $n \in$ OFFEN die Bedingung $g(n) \leq MAX\_Schwelle$ erfüllt, dann wird das letzte Ausschliessen zurückgenommen und die durch ihn referierten Knoten wieder freigegeben. Die $MAX\_Schwelle$ wird auf den früheren Wert in $SUSP.Schwelle$ gesetzt. Dies erfolgt unmittelbar vor der Prozedur *test_t_abbruch()*.

Jetzt zeigen wir, daß die Lösung im schlechtesten Fall ϵ-optimal ist.
Beim Übergang der Ebene j können wir uns maximal um $MAX[i,j] - MIN[i,j]$ vom besten Pfad entfernen. Die theoretisch minimalen Kosten eines Lösungspfades, der auf Ebene 0 beginnt, betragen: $f_{opt} \geq MIN[0] + MIN[0,1] + ... + MIN[N-1,N]$, wobei 0 die niedrigste Ebene ist, auf der Zielkonzepte vorkommen und $MIN[0]$ die minimalen Kosten eines ersten Zieles bedeutet.

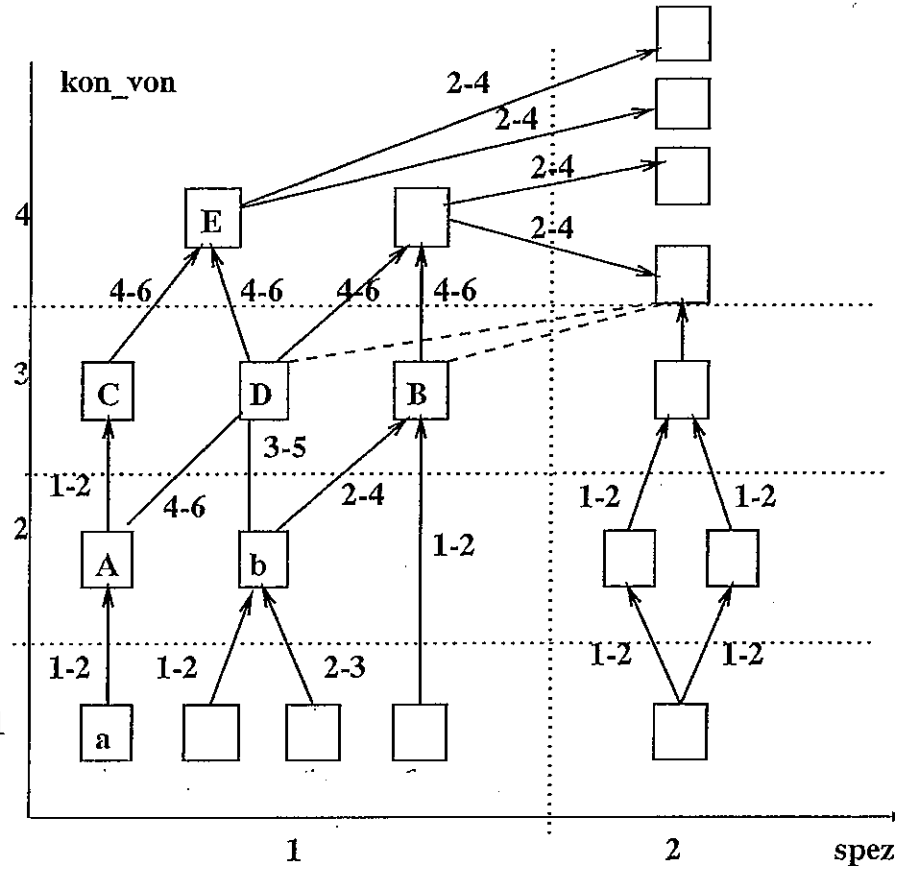
Der denkbar schlechteste Fall für die Optimalität von MIN-MAX Ausschliessen ist, wenn der optimale Pfad mit "niedrigstem" Zielkonzept beginnt und der gefundene Pfad das Ausschliessen auf allen höheren Ebenen (d.h. von 0 bis $N-1$) beinhaltet (N ist die höchste Ebene). In diesem Fall werden die Kosten des Lösungspfades von oben begrenzt:

$$f_{\epsilon-opt} - f_{opt} \leq MAX[1,2] + ... + MAX[N-1,N] - MIN[1,2] - ... - MIN[N-1,N]$$

weil $c_{\epsilon-opt}[0] - c_{opt}[0] < 0$. Wenn es nicht so wäre, dann wäre der gefundene Pfad auf Ebene 0 ausgeschlossen und nicht der optimale Pfad.



Restschätzung der AUFWANDKOSTEN : h_a
(nur zur Illustration, weiterhin wird angenommen,
dass h(v_i) = 0)



RISIKOKOSTEN: g_r (Grenzwerte)
Sekundäre Kantenkosten:
c(K_i,K_j) = g_r(I(K_j)) - g_r(I(K_i))

i \ j	1	2	3	4
1	-	*3*	*6*	*10*
2	x	-	*5*	*11*
3	x	x	-	*6*

KON_VON

i \ j	1	2
1	-	*5*

SPEZ

erlaubte
MAX[i,j]

i \ j	1	2	3	4
1	-	*1*	*1*	*5*
2	x	-	*1*	*5*
3	x	x	-	*4*

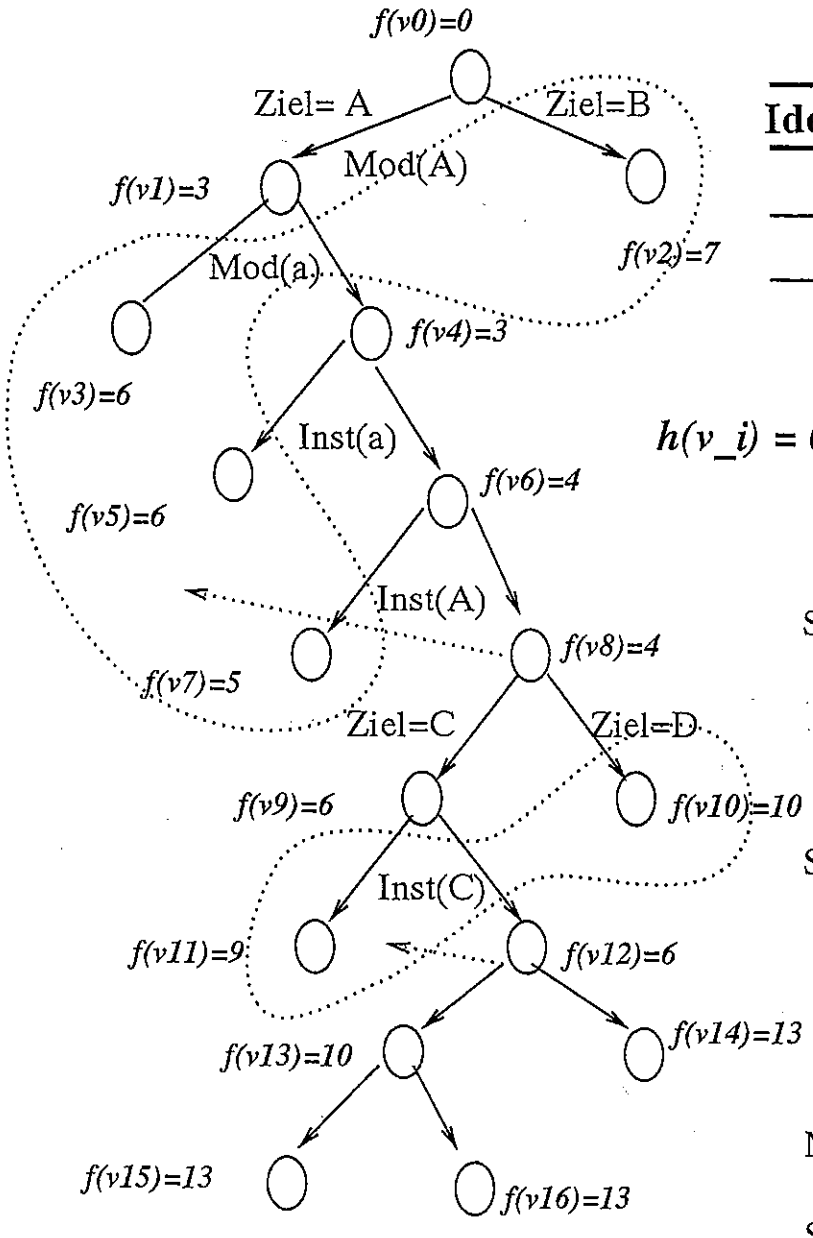
KON_VON

i \ j	1	2
1	-	*2*

SPEZ

MIN[i,j]

Abb. 14(a) MIN, MAX Tabellen



SUSP

Ident.	Schwelle	Knotenliste
v_8	9	v_2,v_3,v_5,v_7
v_12	12	v_10,v_11

$h(v\_i) = 0$

MAX_Schwelle =

SUSP(v8): g(v9) - MIN[2,3] = 6 - 1 = 5
g(v2), g(v3), g(v5) > 5 , g(v7) = 5

MAX_Schwelle = 4 + 5 = 9

SUSP(v12) : g(v13) - MIN[3,4] = 13 - 4 = 9
g(v10) = g(v11) = 10

MAX_Schwelle = 6 + 6 = 12

Nach v_16 : für alle v_i in OFFEN
g(v_i) > MAX_Schwelle
SUSP(v_12) wird zurückgenommen

Abb. 14.(b) Beispiel des MIN-MAX Ausschliessens

9 Eine quasi-Graphensuche nach Zielinstanzen

9.1 Motivation

Die alternierende Kontrolle ist eine Weiterentwicklung der top-down Kontrolle. Sie alterniert zwischen datengesteuerter Zielkonzepte-Generierung und Modell-basierter Instantiierung von Zielkonzepten. Die Effektivität solches Vorhegens basiert auf der Annahme, daß sehr abstrakte Zielkonzepte, die weit entfert von der Segmentierungsebene liegen, sehr schwer direkt aus den Bilddaten zu hypothetisieren sind. Deswegen bedient man sich dazwischen liegenden weniger abstrakten Konzepten, als zwischenzeitlichen Zielen. Das Anliegen ist so schnell wie möglich zu einer guten Hypothese in Form eines modifzierten Konzeptes eines abstrakten Zieles zu gelangen. Gerade diese Etappe wollen wir beschleunigen indem wir äquivalente Zielinstanzen definieren. Diese können sich durch abgedeckte Daten unterscheiden (und gewöhlich ist es so), aber Hauptsache ist, sie führen zumselben globalen Ziel. Zum Beispiel sei das Konzept Auto das globale Ziel der Analyse. Es ist ohne Bedeutung ob wir von einer Instanz des Konzepts Rad über die Kante mit der Rolle Hinterrad oder von einer Instanz von Rad über die Kante mit der Rolle Vorderrad das übergeordnete Ziel Auto erreichen. Haptsache, der Pfad mit der besseren Instanz für Ziel Rad wird gewählt und die schlechteren mit Verweis auf den äquivialenten Pfad aus der Expandierung ausgeschlossen. Nachdem das globale Ziel entlang des besseren Pfades generiert ist, wird nach der zweiten Instanz zu Rad schon innerhalb des neuen expandierten Modells gesucht. Die Inferenzen aus den ausgeschlossenen Pfaden werden jetzt wiederholt, so daß keine mögliche Abdeckung der Daten verfehlt wird. Voraussetung ist jedoch, daß die bisherige Abdeckung von Daten bereits optimal ist.

9.2 Ein Kriterium für äquivalente Pfade

Voraussetzung ist, daß die "Äquivalenz" nur so stark definiert ist, wie es die Qualität der globalen Restschätzung zulässt. Die Wahrscheinlichkeit einer fehlerhaften Entscheidung, welche Instanz von zwei äquivalenten Zielen global optimaler ist, muß statistisch eingrenzbar sein.

Der Äquivalenz-Test findet nur bei der Generierung von neuen Zielen statt.

Es folgt eine noch wage Definition der Äquivalenz von Zielkonzepten.

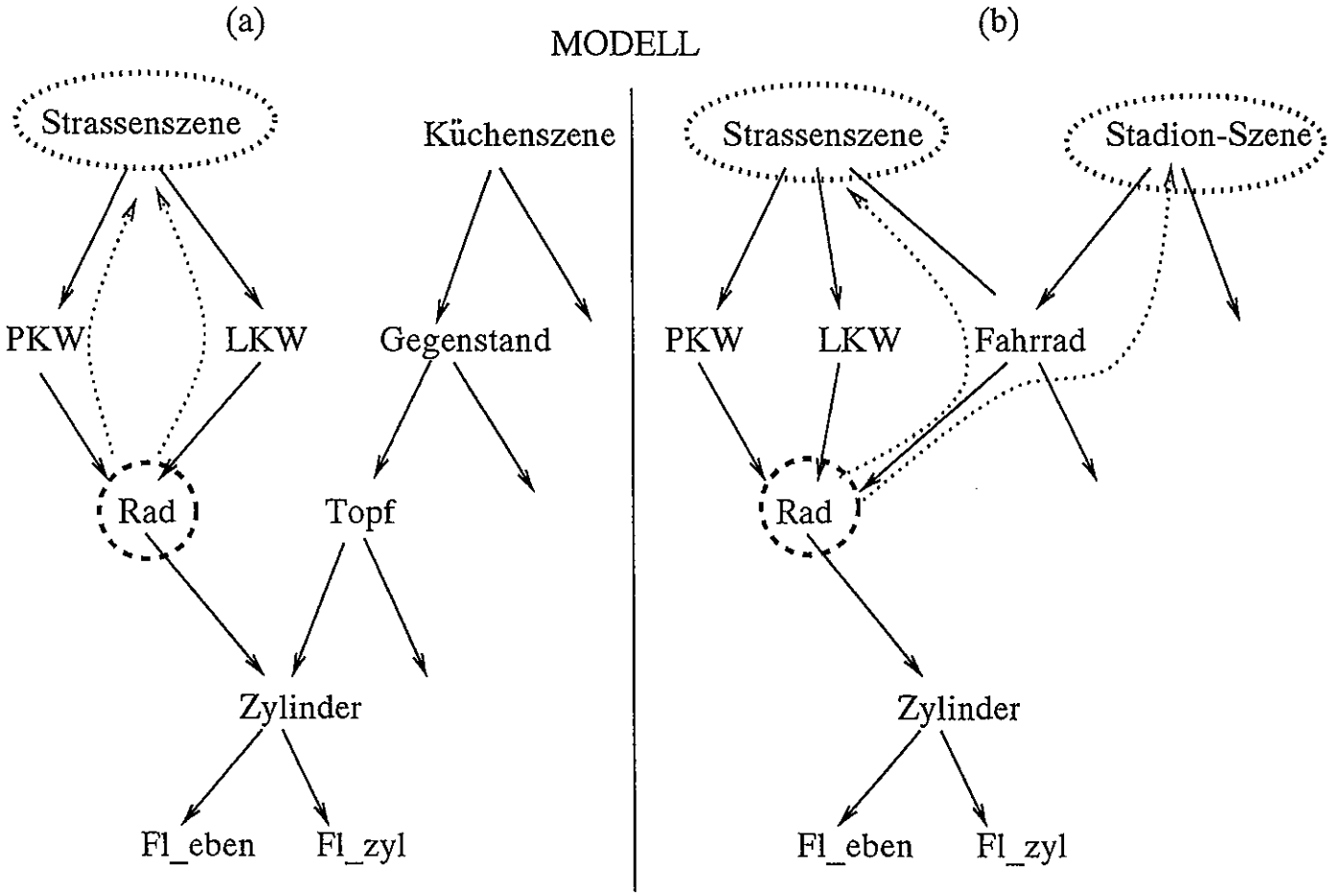
Zwei Zielobjekte desselben Zielkonzepts sind **äquivalent** wenn:

1. die globale Restschätzung für beide zu demselben Ziel führt – entweder gibt es nur ein und dasselbe globale Zielkonzept für beide oder dasselbe Zielkonzept wird bezüglich der Restschätzung durch beide mit Abstand bevorzugt.

2. die in beiden Fällen abgedeckten Segmentierungsdaten voll getrennt sind (keine Instanz wird durch beide Zielobjekte abgedeckt).

Beispiel In Abb. 15 sind schematisch zwei Modellausschnitte dargestellt. Nehmen wir an, daß die Segmentierungsdaten 8 Instanzen enthalten, je vier zu Konzepten `Fl_eben` und `Fl_zyl`. Nehmen wir weiter an, daß während der Analyse vier Instanzen von Konzept `Rad` generiert wurden. Dabei decken die Instanzen $I_{10}(Rad), I_{11}$ dieselben Daten I_1, I_5 ab, sowie die Instanzen $I_{12}(Rad), I_{13}$ dieselben Daten I_4, I_8 ab. Für das Modell im Beispiel (a) können wir gleich aufgrund nur des Kriteriums (2) sagen, daß I_10 äquivalent zu I_{12}, I_{13} ist und I_{11} äquivalent zu I_{12}, I_{13} ist. Andere Kombinationen sind nicht äquivalent, weil dann die abgedeckten Daten nicht getrennt sind. Das Kriterium (1) ist für Rad immer erfüllt, da nur ein übergeordnetes globales Ziel erreicht sein kann Strassenszene. Im Beispiel (b) ist es nicht ohne weiteres ersichtlich, ob jede Äquivalenz aus Beispiel (a) das Kriterium (1) erfüllt. Es muß eine globale Restschätzung für jede Instanz von Rad durchgeführt werden um festzustellen, welches von zwei konkurrierenden globalen Zielen Strassenszene oder Stadion-Szene jeweils mit Abstand bevorzugt wird.

Das Nichtexpandieren von schlechteren Pfaden zu einem Zielkonzept, führt zu einer quasi-Graphensuche



DATEN: I1(Fl_eben), .. . , I4(Fl_eben), I5(Fl_zyl), . . . , I8(Fl_zyl)

Seien I10(Rad) und I11(Rad) gegeben, die I1 und I5 abdecken
Seien I12(Rad) und I13(Rad) gegeben, die I4 und I8 abdecken

Für Modell (a) : I10 ist äquivalent zu I12 oder I13, I11 ist äquivalent zu I12 oder I13,
I10 und I11 sowie I12 und I13 sind nicht äquivalent

Für Modell (b) : die Äquivalenzen für (a) wenn dasselbe globale Zielkonzept bevorzugt wirc

Abb. 15 Beispiele von äquivalenten Zielen

im Raum (1). Wie wir wissen, wird im Schritt 7.b der A^*-Graphensuche darüber entschieden, welcher von zwei Pfaden zumselben Knoten zu wählen ist. Diese Entscheidung ist global optimal – der Pfad mit kleineren g-Kosten ist immer besser. Demgegenüber ist die Optimalität unserer Entscheidung nicht nur von dem g-Wert sondern auch h-Wert, wie auch von der Differenz der abgedeckten Daten abhängig.

Wir können eine potentielle Reduzierung der Suchräume für äquivalente Ziele erreichen und damit des Aufwandes der Baumsuche. Aber durch die Einführung des Schrittes 7 in den A^*-Algorithmus, haben wir ihn zum Graphen-Suchverfahren umgewandelt. Da der Graph nur abstrakt durch die Äquivalenz von Suchbaumpfaden impliziert wird, bezeichnen wir das als quasi-Graphensuche.

Wir sind damit beim Ausgangspunkt des Berichtes angekommen – ist die A^*-Suche optimal für Graphen. Es ist wichtig, die Suche so zu führen, daß die äquivalenten Ziele in bester Reihenfolge generiert werden, d.h. der am schnellsten gefundene Pfad zu einem äquivalenten Ziel soll auch der beste sein. Diese Reihenfolge wird durch A^* nur bei Verwendung einer konsistenten Heuristik garantiert. Wie wir bereits wissen sind für allgemeine Fälle die Verfahren B, C oder D besser geeignet als A^*.

10 Experimente

Beide Verfahren - das MIN-MAX Ausschliessen und die Kontrolle über äquivalenten Pfaden - sollen auf der Basis der alternierenden Kontrolle implementiert werden. Wir überlegen auch welche Änderungen nötig sind, um statt A^* andere Graphen-Suchverfahren als Basissuche der Kontrolle anzuwenden.

10.1 Umfeld

Es ist kein direkter Vergleich mit dem Bewertungsschema, wie er für die Sprachanalyse definiert wurde, möglich. Abgesehen davon, daß zur Zeit kein lauffähiges Modul mit realistischen Bewertungen vorhanden ist, gibt es auch grundsätzliche Unterschiede. Für das Suchproblem (1) wurde bisher keine Heuristik-Komponente spezifiziert. Es wird nicht bezüglich eines globalen Zielkonzeptes geschätzt, sondern höchstens bezu"glich der restlichen Segmentierungsdaten. Dieser Wert wird aber nicht im Bewertungsvektor repräsentiert, so daß wir keine Aussagen treffen können, wie die Suche sich verhält, abhängig von der Qualität der Heuristik.

Ein Modell mit Bewertungsprozeduren, sowie synthetische Daten müssen deshalb auch erstellt werden.

Die Simulationen sollen zeigen, ob die Verfahren zur sub-optimalen Suche überhaupt Vorteile bringen und falls ja, unter welcher Heuristik dies möglich ist.

10.2 Was wird geändert?

Im Anschluß an diese Sektion folgt die Quellendatei der Hauptprozedur der alternierenden Kontrolle (Monitoring-Befehle wurden rausgenommen).

Falls ein anderes Basisverfahren als A^* angewendet sein soll, z.B. B, C, D, dann ist die Knotenauswahlprozedur `get_o_optiknoten()` zu ändern (sie entspricht in etwa dem Schritt in A^* (Sektion 1)). Für A^{**} muß bei der Bewertung von neu generierten Suchbaumknoten die Pfadenmax-Regel berücksichtigt werden. Die dafür verantwortliche Unterprozedur ist `bewerte_t_knoten()`. Für B und C muß während der Initialisierung eine neue Variable F auf Nullwert gesetzt werden.
Wegen seines Schrittes D6 erfordert der D-Algorithmus weiterführende Modifikationen. Wir wir wissen (Sektion 3) kann die Heuristik-Komponente entweder des Vorgängerknoten oder von manchen Nachfolgerknoten nachträglich modifiziert werden. Während der Suchbaumexpandierung (z.B. in Folge der Aufrufe von `instantiiere_k_eintrag()`, `expand_k_knoten()`) findet immer eine Sequenz von Aufrufen `bewerte_t_knoten()` und `insert_o_knoten()` statt (entspricht dem Schritt 6 und dem nach OFFEN bringen in A^*). Für die Modifizierung von Bewertungen muß eine neue Prozedur geschrieben werden, die unmittelbar nach der Suchbaumexpandierung aufzurufen wäre.

Die letzte Bemerkung ist auch passend für die Implementierung einer allgemeinen Graphensuche (statt Baumsuche) (Schritt 7 in A^*).

Die beiden sub-optimalen Verfahren greifen in die Suche nur in markanten Punkten der Analyse ein.

11 Zusammenfassung

Es wurden folgende Arbeitspunkte formuliert:

1. Optimalität von Graphen-Suchverfahren bei unterschiedlicher Heuristik
2. Zusammenhänge zwischen Qualität der Bewertung und Komplexität der Baumsuche
3. Vorschläge von sub-optimalen Suchverfahren für die alternierende Kontrolle
4. Experimente

Bis auf den letzten Punkt wurde der vorgesehene Plan abgearbeitet. Die Experimente konnten nicht auf dem Sprachnetz durchgeführt werden aus zwei Gründen:
1. Es lag keine lauffa"hige Version mit realistischen Bewertungen vor.
2. Das Bewertungsschema ist vom Gesichtspunkt der optimalen Graphensuche sehr spezifisch. Unter anderem besitzt der Bewertungsvektor keine Heuristik-Komponente, so daß keine Aussagen getroffen werden können bezüglich des Zusammenhanges: Suchaufwand–Qualität der Heuristik. Der Einsatz von anderen Suchverfahren als A^* ist für dieses Bewertungsschema fraglich.

Es folgen die wichtigsten Erkenntnisse der Forschung.

1. Der Unterschied zwischen A und A^* wird in letzter Zeit in Frage gestellt (Sektion 1).
2. A^* ist 0-optimal gegenüber zulässigen Algorithmen nur bei konsistenter und nichtpathologischer Heuristik. Im allgemeinen Fall der zulässigen Heuristik, ein anderer Algorithmus A^{**} dominiert über A^* und ist 3-optimal gegenüber zulässigen Algorithmen (Sektion 2).
3. Bezüglich des Aufwandes der Graphensuche sind wir bestrebt das wiederholte Expandieren von Knoten zu vermeiden. Es gibt solche Algorithmen wie B, C und D, die sich in diesem Bezug schon bei zulässiger Heuristik besser verhalten als A^* (Sektion 3).
4. Die im Vergleich zur Problem-Komplexität zu geringe Rechenleistung von bisherigen Komputern führte zur Entwicklung von suboptimalen und lokal optimalen Suchverfahren (Sektion 4), bzw. zu Vorschlägen von hybriden Verfahren (Sektion 5).
5. Sehr präzise Heuristiken (höchstens logarithmisch wachsender Fehler der Schätzung) sind erforderlich damit die Komplexität der A^*-Baumsuche nicht exponentiell mit Pfadlänge N wächst (Sektion 5). Die Anzahl der Zielknoten darf höchstens eine polynomische Funktion von N sein.
6. Die Signalanalyse kennzeichen nur *implizit* gegebene: Suchraum und Kostenwerte (Sektion 6). Die Kantenkosten $c(v_i, v_j)$ sind dazu nur sekundär zu den Knotenkosten $f(v_i), f(v_j)$.
7. Für die Untersuchung der Effektivität von optimalen Suchverfahren brauchen wir die Heuristikwerte der Restschätzung bezüglich des globalen Analysezieles. Der Aufbau des Zustandsraumes für die Signalanalyse über den Segmentierungsdaten und dem Modell sollte sich auch in der Knotenbewertung wiederspiegeln - deswegen wird die Kombination von Qualität (Risiko) und Aufwand vorgeschlagen (Sektion 7).
8. Zwei suboptimale Verfahren sollen zur besseren Suche in die Tiefe der alternierenden Kontrolle führen (Sektion 8, 9). Das Bewertungsschema, wie auch die Lösung des Suchproblems $S1$ sind anwendungsabhängige Elemente der Kontrolle.
9. Um die alternierende Kontrolle mit Optionen für verschiedene Suchverfahren auszustatten sind teilweise geringe, teilweise erhebliche Änderungen notwendig (Sektion 10).

Literatur

[BAG83] Bagchi A., Mahanti A.: *Search Algorithms under Different Kinds of Heuristics - a Comparative Study*, **JACM, 30**(1983), No. 1, 1–21.

[BAG88] Bagchi A., Sen A.K.: *Average–Case Analysis of Heuristic Search in Tree-Like Networks.* In: Kanal L., Kumar V. (Eds.): **Search in Artificial Intelligence**, Springer Series *Symbolic Computation - Artificial Intelligence*, Springer Vg., New York-Berlin-Heidelberg, 1988, 131–165.

[CHA89] Chakrabarti P.P. et al.: *Heuristic Search in Restricted Memory.* **Artificial Intelligence, 41**(1989), 197–221.

[DEC88] Dechter R., Pearl J.: *The Optimality of A**. In: Kanal L., Kumar V. (Eds.): **Search in Artificial Intelligence**, Springer Series *Symbolic Computation - Artificial Intelligence*, Springer Vg., New York-Berlin-Heidelberg, 1988, 166–199.

[GAS79] Gaschnig J.: *Performance measurement and analysis of certain search algorithms*, **Ph.D. dissertation**, Technical Report, CMU-CS-79-124, Computer Science Dept., Carnegie-Mellon University.

[HUY80] Huyn N., Dechter R., Pearl J.: *Probabilistic Analysis of the Complexity of A**. **Artificial Intelligence, 15**(1980), 241–254.

[KOR85] Korf R.E.: *Depth–First Iterative–Deepening: An Optimal Admissible Tree Search.* **Artificial Intelligence, 27**(1985), 97–109.

[KOR90] Korf R.E.: *Real–Time Heuristic Search.* **Artificial Intelligence, 42**(1990), 189–211.

[KUM91] Kummert F.: **Flexible Steuerung eines sprachverstehenden Systems mit homogener Wissensbasis**. Dissertation, Universität Erlangen-Nürnberg, 1991.

[MAH88] Mahanti A., Ray K.: *Network Search Algorithms with Modifiable Heuristics.* In: Kanal L., Kumar V. (Eds.): **Search in Artificial Intelligence**, Springer Series *Symbolic Computation - Artificial Intelligence*, Springer Vg., New York-Berlin-Heidelberg, 1988, 200–222.

[MAR77] Martelli A.: *On the Complexity of Admissible Search Algorithms*, **Artificial Inteligence, 8**(1977), 1–13.

[NIE90] Niemann H., Brünig H., Salzbrunn R., Schröder S.: *A Knowledge–Based Vision System for Industrial Applications*, **Machine Vision and Applications, 3**(1990), No. 4, 210–229.

[NIL71] Nilsson N.J.: **Problem-Solving Methods in Artificial Intelligence**, New York, McGraw–Hill, 1971.

[NIL80] Nilsson N.J.: **Principles of Artificial Intelligence**, Palo Alto, California, Tioga, 1980.

[PEA83] Pearl J.: *Knowledge versus search: A quantitative analysis using A**, **Artificial Intelligence, 20**(1983), 1–13.

[PEA84] Pearl J.: **Heuristics. Intelligent Search Strategies for Computer Problem Solving.** Addison–Wesley, Reading, Mass, 1984.

[PEA88] Pearl J.: **Probabilistic Reasoning in Intelligent Systems**, San Mateo, CA, 1988.

[POH77] Pohl I.: *Practical and theoretical considerations in heuristic search algorithms*, In: [Elcock E.W., Michie D. (ed.): **Machine Intelligence, 8**(1977) New York, Wiley], 55–72.

[ROS72] Rosenberg R.S., Kestner J.: *Look-ahead and one-person games.* **J. Cybern.,2**(1972), No. 4, 27-42.

[SAG90] Sagerer G.: **Automatisches Verstehen gesprochener Sprache**, Bibliographisches Institut, Mannheim, 1990.