Parallelverarbeitung in einem semantischen Netzwerk für die wissensbasierte Musteranalyse

Der Technischen Fakultät der
Universität Erlangen-Nürnberg

zur Erlangung des Grades

DOKTOR-INGENIEUR

vorgelegt von

Volker Fischer

Erlangen — 1994

Als Dissertation genehmigt von der
Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung:	2. Dezember 1994
Tag der Promotion:	24. März 1995
Dekan:	Prof. Dr. Dr. h.c. F. Durst
Berichterstatter:	Prof. Dr. H. Niemann
	Prof. Dr. G. Sagerer

Zum Geleit

Klassifikation, Analyse und Verstehen von Mustern wie Bildfolgen oder gesprochene Sprache erfordern die Repräsentation und Nutzung von Wissen über den Problemkreis. Hierfür sind in der Vergangenheit verschiedene Ansätze entwickelt worden. In der Regel werden mindestens zwei Verarbeitungsphasen, nämlich eine überwiegend daten– und eine überwiegend modellgetriebene, unterschieden. Diese sollten so gesteuert werden, daß eine Interpretation erzielt wird, die optimal ist bezüglich der beobachteten Muster, des gespeicherten Wissens und der intendierten Anwendung.

Wissensbasierte Verarbeitung und Analyse von Mustern erfordert einen hohen Aufwand an Rechenleistung, der zum Beispiel durch Parallelverarbeitung erbracht werden kann. In der datengetriebenen Verarbeitungsphase überwiegen numerische Rechnungen auf der Ebene der Bildpunkte. Hier gibt es eine Reihe von Techniken zur Parallelisierung. In der modellgetriebenen Phase überwiegen (bisher) symbolische Inferenzen, für die nur wenige Ansätze zur Parallelisierung vorliegen, insbesondere wenn noch die Optimierung einer Gütefunktion gefordert wird.

In der vorliegenden Arbeit wird ein Ansatz entwickelt und experimentell erfolgreich getestet, der wissensbasierte Interpretation von Mustern auf die iterative Optimierung einer vorgegebenen Gütefunktion zurückführt. Wissen wird vom Anwender auf konzeptuell hoher Ebene durch Konzepte eines semantischen Netzes repräsentiert und dann automatisch in ein geschichtetes Netzwerk von Prozeduren transformiert. Alle Prozeduren einer Schicht können parallel aktiviert werden. Es gibt eine Schnittstelle zu den Ergebnissen der datengetriebenen Verarbeitungsphase bzw. der Segmentierung. Die Interpretation erfordert zum einen eine Zuordnung von Segmentierungsergebnissen zu Konzepten der Wissensbasis, zum anderen eine Auswahl unter konkurrierenden Alternativen der Interpretation. Beides sollte so geschehen, daß eine Gütefunktion optimiert wird. Die Lösung erfolgt mit Algorithmen der kombinatorischen Opimierung, von denen sich in der untersuchten Anwendung genetische Algorithmen als besonders leistungsfähig erwiesen haben.

Damit ist an einem konkreten Beispiel gezeigt, daß sich Interpretation von Mustern relativ zu einer symbolisch repräsentierten Wissensbasis als Problem der iterativen numerischen Optimierung formulieren läßt, daß dieses durch bekannte Algorithmen der kombinatorischen Optimierung lösbar ist und daß damit eine effiziente parallele Implementierung der Wissensverarbeitung möglich ist. Zudem haben iterative Algorithmen eine natürliche "Allzeit–bereit (any–time)" Eigenschaft, da in jedem Iterationsschritt eine Lösung des Problems vorliegt, die mit zunehmender Zahl von Iterationen bzw. zunehmender Verarbeitungszeit immer besser wird.

H. Niemann

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die das Entstehen der vorliegenden Arbeit gefördert haben:

- Den Herren Prof. Dr. H. Niemann und Prof. Dr. G. Sagerer danke ich für zahlreiche richtungsweisende Diskussionen und Gespräche sowie für die Übernahme der Gutachten.
- Meinen Kollegen Joachim Hornegger, Ralf Kompe und Elmar Nöth danke ich für die Durchsicht der Arbeit und zahlreiche gleichermaßen kritisch wie aufmunternde Anmerkungen.
- Frau Silke Richter vom Bayerischen Forschungszentrum für Wissensbasierte Systeme (FORWISS München, Forschungsgruppe "Kognitive Systeme", Prof. Dr. B. Radig) danke ich für ihre kollegiale Zusammenarbeit.
- Meiner ehemaligen Kollegin Christine Weighardt schulde ich besonderen Dank für viele anregende Gespräche und ihre Unterstützung bei der Überwindung einiger Anfangsschwierigkeiten.
- Schließlich bedanke ich mich bei der Deutschen Forschungsgemeinschaft (DFG) für die Förderung der Arbeit im Rahmen des Sonderforschungsbereichs 182 "Multiprozessor– und Netzwerkkonfigurationen".

Volker Fischer

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einführung

Nach einer kurzen Einführung in die Themenstellung der Arbeit und der Motivierung von Untersuchungen zur Parallelverarbeitung in der Musteranalyse gibt dieses einleitende Kapitel einen Überblick über ein allgemeines Verarbeitungsmodell für die wissensbasierte Bildanalyse. Darauf aufbauend werden bisherige Ansätze zur parallelen Bildverarbeitung skizziert und Zielsetzung und Beitrag der Untersuchungen zusammengefaßt. Abschließend geben wir einen Überblick über den Aufbau der Arbeit.

1.1 Analyse komplexer Muster

Die Erfassung und Bewältigung neuartiger Situationen in einer komplexen und dynamischen Veränderungen unterliegenden Umgebung ist eine zentrale Leistung menschlicher Intelligenz [Bro89]. Eine Voraussetzung dafür ist die Fähigkeit, Signale aus einer ungeregelten Umwelt, die sich durch eine hohe Datenrate, eine große Dynamik und zahlreiche Störungen auszeichnen, im Bruchteil einer Sekunde zu empfangen und unter wechselnden Randbedingungen so weiterzuverarbeiten, daß eine sinnvolle Reaktion möglich wird. Die Verarbeitung wird dabei nicht allein durch den an einem geeigneten Rezeptor vorliegenden physikalischen Reiz gesteuert, sondern erfolgt unter Ausnutzung eines umfangreichen Alltagswissens, worunter wir hier eine Menge von präsenten Fakten, Erfahrungen, Vorerwartungen und Verhaltensmustern verstehen wollen.

Die wissensbasierte Analyse komplexer Muster ist — neben der Klassifikation einfacher Muster — ein Teilgebiet der Mustererkennung, die sich mit der Verarbeitung, Auswertung und Interpretation von Sinneseindrücken durch digitale Rechenanlagen befaßt [Nie83]. Im hier betrachteten Kontext ist ein Muster eine Funktion $\boldsymbol{f}(\boldsymbol{x})$, die ein aus der Umwelt aufgenommenes und für die digitale Verarbeitung geeignet transformiertes Signal beschreibt. Entsprechend der Wichtigkeit von optischer und akustischer Wahrnehmung für den Menschen konzentrieren sich zahlreiche Aktivitäten im Bereich der Mustererkennung auf die

Klassifikation und Analyse von Bild– und Sprachsignalen; abkürzend sprechen wir auch von den Gebieten der automatischen Bild– bzw. Sprachverarbeitung.

Einsatzgebiete für die in der Mustererkennung entwickelten Verfahren bestehen überall dort, wo die oben umrissenen Fähigkeiten die Grundlage zur maschinellen Verrichtung von Tätigkeiten bilden, die für den Menschen ermüdend oder gefährlich sind, oder die manuell nicht durchzuführen sind. Beispielhaften Charakter besitzen hier die Verarbeitung von Belegen im Zahlungsverkehr zwischen Banken, die Steuerung eines mobilen Roboters im Sicherheitsbereich eines Kernkraftwerkes oder auch die Auswertung großer Datenmengen bei der Erdfernerkundung; umfangreichere Übersichten und weiterführende Literaturhinweise finden sich unter anderem in den einführenden Kapiteln von [Nie83, Nie87, Jäh93].

Zur Eingrenzung und Motivation der in der vorliegenden Arbeit beschriebenen Untersuchungen greifen wir nochmals die bereits erwähnte Unterscheidung der Klassifikation einfacher Muster von der Analyse komplexer Muster auf; dabei folgen wir weitestgehend den Ausführungen in [Nie90a, Kap.1]. Bild 1.1 zeigt einige Beipiele einfacher Muster, die sich dadurch auszeichnen, daß sie als Ganzes behandelt und unabhängig von anderen Mustern klassifiziert werden können.

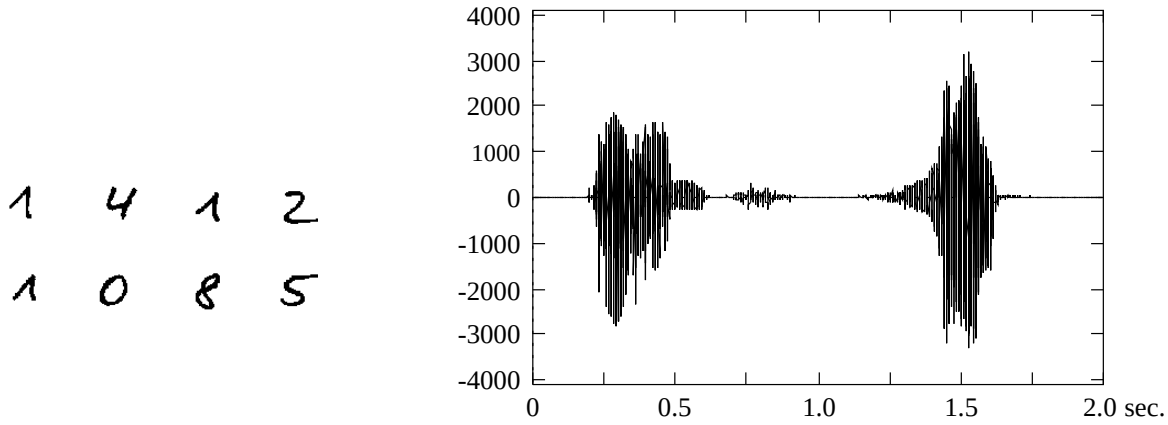


Bild 1.1: Eine Folge handgeschriebener Ziffern und das Sprachsignal (2 sec.) der isoliert gesprochenen Wörter "eins" und "vier" als Beispiele einfacher Muster.

Ziel der Klassifikation ist die Einordnung des Musters in eine von mehreren Musterklassen, die in beiden gezeigten Beispielen die Ziffern 0 bis 9 darstellen könnten. Ist eine Klassifikation nicht mit genügender Zuverlässigkeit möglich, so darf ein Muster einer speziellen Rückweisungklasse zugewiesen werden. Da selbst die Behandlung nur der wichtigsten Methoden zur Klassifikation einfacher Muster den Rahmen dieser Arbeit sprengen würde, müssen wir hierfür auf die zahlreich vorhandene Literatur [Dud73, Dev82, Nie83] verweisen.

Muster, deren Klassifikation als Ganzes nicht möglich ist, oder für welche die Angabe eines Klassennamens nicht ausreicht, werden als komplex bezeichnet. Ziel der Analyse komplexer Muster ist die Generierung einer individuellen symbolischen Beschreibung $\mathcal{B}$,

deren Inhalt von den Erfordernissen der jeweiligen Anwendung abhängt. Im allgemeinen enthält eine symbolische Beschreibung eine Menge der einfacheren Elemente, aus denen ein komplexes Muster aufgebaut ist, und eine Angabe von deren Beziehungen untereinander. Falls erforderlich oder erwünscht, kann die symbolische Beschreibung auch um eine Schlußfolgerung für den Anwendungsbereich ergänzt werden.

Bild 1.2: Zwei im Abstand von 40 Millisekunden aufgenommene Bilder einer Straßenverkehrsszene als Beispiele komplexer Muster.

Die Gegenüberstellung der Bilder 1.1 und 1.2, das als Beispiel eines komplexen Musters einen kurzen Ausschnitt aus einer Bildsequenz [1] zeigt, verdeutlicht, daß die Musteranalyse eine ungleich schwierigere Aufgabe als die Klassifikation ist. Eine symbolische Beschreibung könnte hier wie folgt lauten:

> "Auf der zweispurigen Fahrbahn befinden sich fünf Fahrzeuge, von denen sich vier auf das Kamerafahrzeug zubewegen. Das fünfte Fahrzeug durchfährt mit einer Geschwindigkeit von v km/h und in einer Entfernung von s Metern vor dem Kamerafahrzeug eine Rechtskurve. (Dem Fahrer des Kamerafahrzeugs wird geraten, die Spur beizubehalten und die Geschwindigkeit beim Einfahren in die Kurve nicht zu erhöhen.)"

Einen exemplarischen Überblick über typische Operationen und Objekte in einem Bildanalysesystem und deren Organisation in einem geschichteten Modell des Verarbeitungsprozesses gibt Bild 1.3 auf Seite 5. Eine nähere Charakterisierung der einzelnen Ebenen, die

[1] Für die Überlassung des Bildmaterials dankt der Autor dem Bayerischen Forschungszentrum für Wissensbasierte Systeme (FORWISS), insbesondere Frau C. Weighardt.

— mit anderen Operationen und Objekten versehen — in ähnlicher Form auch in sprachverstehenden Systemen anzutreffen sind (vgl. [Nie87, S. 12]), werden wir in Abschnitt 1.2 vornehmen; für die hier angestrebte Abgrenzung genügt zunächst die Feststellung, daß die Analyse komplexer Muster Methoden der Klassifikation — etwa zur Bestimmung einfacher Objekte einer Szene — enthalten kann. Darüber hinaus ist Bild 1.3 zu entnehmen, daß eine Vielzahl weiterer Verarbeitungsschritte notwendig sind (beispielsweise zur Bestimmung der Beziehungen zwischen Objekten oder zur Generierung einer Handlungsanweisung), welche im Gegensatz zur Klassifikation auch die explizite Formulierung von Wissen aus dem jeweiligen Einsatzbereich des Systems erfordern. Eine umfassende Darstellung dieser Verfahren und ihres Zusammenwirkens in einem Analysesystem muß auch hier der Literatur vorbehalten bleiben; stellvertretend seien [Lie89, Nie90a, Rad93] für den Bereich der Bildanalyse und [Ain88, Nie90a] für die automatische Sprachverarbeitung genannt.

Ein Hauptproblem bei der Entwicklung leistungsfähiger Musteranalysesysteme ist die Behandlung großer Datenströme unter sehr restriktiven Zeitbedingungen. So ergibt sich allein bei der mit einer üblichen Videogeschwindigkeit von 25 Bildern pro Sekunde durchgeführten Aufnahme der in Bild 1.2 gezeigten 512×512–Grauwertbilder eine Datenrate von über 6.5 Megabyte pro Sekunde, die sich bei einer zusätzlichen Ausnutzung der Farbinformation auf circa 20 MB/sec. verdreifachen würde. Aufgrund der früher nur begrenzt verfügbaren Ressourcen zur Aufnahme, Speicherung und Verarbeitung derartiger Datenmengen bildet die Analyse komplexer Muster einen im Vergleich zur Klassifikation relativ jungen Zweig der Mustererkennung. Zusammen mit der Schwierigkeit der Aufgabe erklärt dies, daß erfolgreiche Anwendungen bisher vorrangig im Bereich der Musterklassifikation zu finden sind. So berichtet bereits [Rab85] von Fehlerraten unter 2.5 Prozent für die Erkennung isoliert gesprochener Ziffern, in [Car93] wird eine Fehlerrate von 0.28 Prozent für die Verbundworterkennung von Ziffern erzielt, und auch Diktiersysteme mit relativ großem Wortschatz sind seit kurzem kommerziell erhältlich [Der93, Bak93]. Ebenso werden bei der Klassifikation von gedruckten oder handgeschriebenen Schriftzeichen Ergebnisse erzielt, welche die menschliche Erkennungsleistung bezüglich Fehlerrate und Verarbeitsgeschwindigkeit erreichen oder übertreffen. Bereits der in [Sch78] beschriebene Postanschriftenleser erzielt eine nahezu verschwindende Fehlerrate und liest das maschinengeschriebene Adressfeld von circa 60000 Briefen pro Stunde; bei der Klassifikation von 465 handgeschriebenen Zeichen des mehr als 40000 Schriftzeichen umfassenden chinesischen "Alphabets" werden in [Cho91] eine Erkennungsrate von ca. 95 Prozent und eine mittlere Verarbeitungsdauer von 3.5 Sekunden pro Zeichen auf einer herkömmlichen Workstation angegeben.

Während also Techniken zur Klassifikation einfacher Muster einen relativ hohen Standard erlangt haben und das Laborstadium in zunehmendem Maße verlassen, sind vergleichbare Erfolge für Aufgabenstellungen der automatischen Interpretation komplexer Bild– und Sprachsignale bislang lediglich für ausgewählte Komponenten derartiger Systeme und

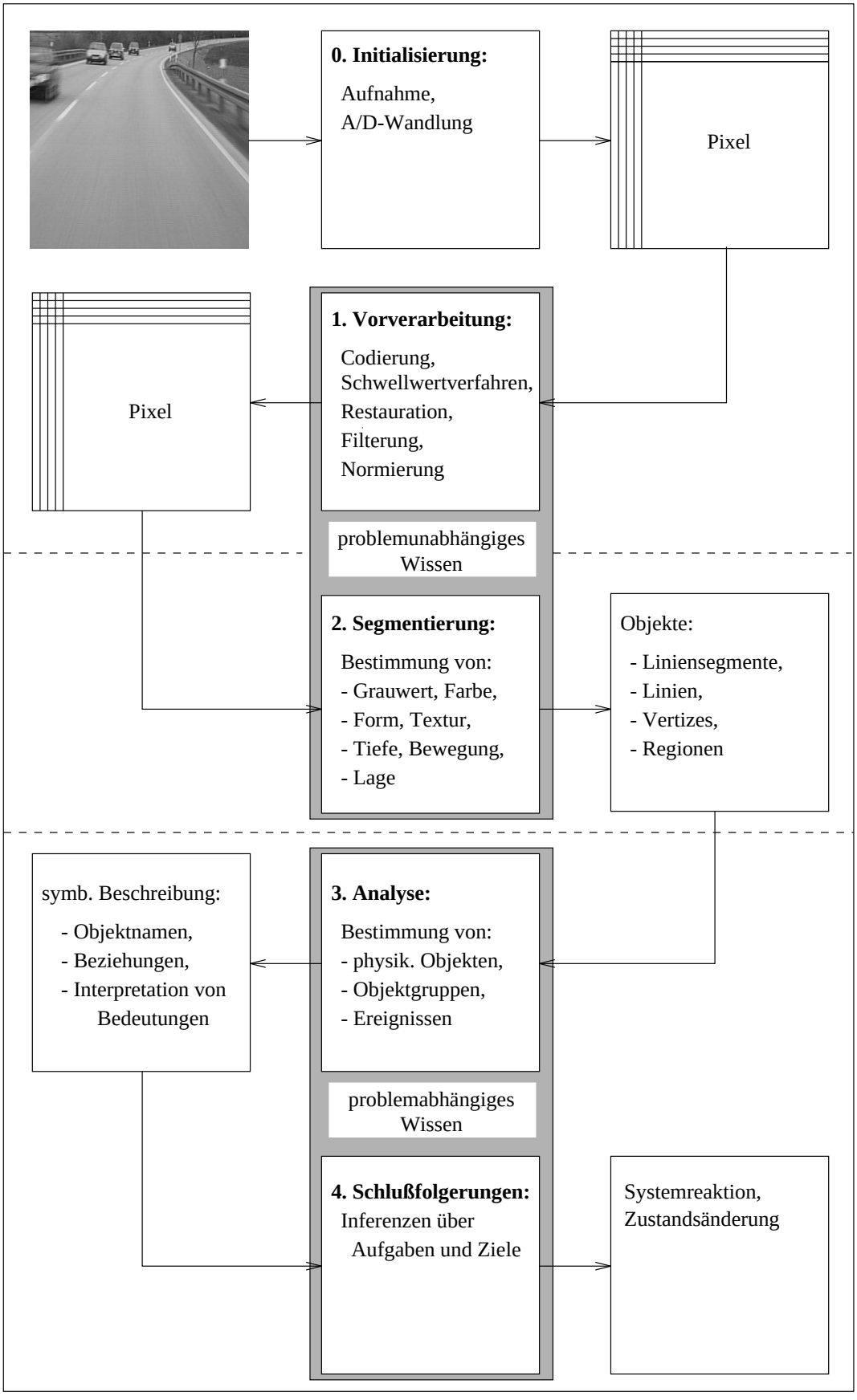


Bild 1.3: Überblick über Operationen, Zwischenergebnisse und Verarbeitungsschichten der Bildanalyse (nach [Nie87]). Erläuterungen finden sich im Text.

nur unter einschränkenden Annahmen zu verzeichnen. So gehört die Echtzeit*erkennung* in einem sprach*verstehenden* System, dessen Ziel die Extraktion des Inhaltes fließend gesprochener Sprache ist, nahezu zum Stand der Technik, wenn ein relativ kleines Vokabular verwendet wird; der Echtzeitfaktor für die Interpretation des Erkennungsergebnisses ist dagegen noch bedeutend größer. Für die Erkennung wird in [Kuh94] ein Echtzeitfaktor von 1.8 bei einem Wortschatz von ca. 1000 Wörtern berichtet, und bei den in [Mas93] beschriebenen Experimenten ergibt sich der Faktor 15 für die Interpretation einer Äußerung.

Die in [Wee91] beschriebenen Ergebnisse des *DARPA integrated image–understanding benchmarks*, einer vergleichenden Untersuchung mehrerer Rechnerarchitekturen anhand typischer Bildverarbeitungsoperationen, zeigen, daß im Bereich der Bildanalyse noch ungünstigere Verhältnisse anzutreffen sind. So erfordert hier bereits ein Teil der zur Interpretation benötigten Erkennungsaufgaben eine deutlich größere Verarbeitungsdauer; für die Segmentierung von 712×568–Grauwertbildern einer Folge von Straßenverkehrsszenen (vergleiche Bild 1.2) wird in [Wet93] ein Echtzeitfaktor von 25 auf einer Workstation mit 125 MIPS (*million instructions per second*) angegeben.

Neben der hohen Datenrate ist eine weitere Ursache für die lange Analysedauer in der großen Zahl durchzuführender Verarbeitungsschritte zu sehen. Kann die benötigte Rechenleistung für die Bildsegmentierung aus den obigen Angaben mit etwa 300 Instruktionen pro Pixel abgeschätzt werden, so ist der Aufwand für die Interpretation eines Bildes im allgemeinen bedeutend größer. In [Lev87] wird dieser mit $10^3 - 10^4$ Instruktionen pro Pixel abgeschätzt, woraus sich ergibt, daß für die Echtzeitinterpretation von Bildfolgen eine Rechenleistung von mehreren GIPS (*giga instructions per second*) erforderlich ist. Zwei Möglichkeiten zum Erreichen der hier beschriebenen Anforderungen bestehen in der Nutzung dedizierter Hardware auf allen Verarbeitungsebenen und im *Einsatz von Parallelrechnern.*

Die Verwendung von Parallelrechnern wird einerseits durch die erkennbaren technologischen Grenzen von konventionellen Computern und Vektorrechnern und eine höhere Wirtschaftlichkeit und Flexibilität motiviert, wirft jedoch andererseits Probleme des Entwurfs und des Betriebs sowie der Entwicklung geeigneter Algorithmen für konkrete Anwendungen auf. Gleichermaßen wie die potentielle Bedeutung wird die Vielzahl der zu lösenden Fragen durch die Tatsache deutlich, daß sich derzeit (1994) gleich zwei von der Deutschen Forschungsgemeinschaft geförderte Projekte mit zusammen über 100 Mitarbeitern mit der Untersuchung von parallelen Rechensystemen befassen. Überblicke über die in den Sonderforschungsbereichen 342 "Methoden und Werkzeuge für die Nutzung paralleler Architekturen" und 182 "Multiprozessor– und Netzwerkkonfigurationen" an der Technische Universität München bzw. der Friedrich-Alexander-Universität Erlangen-Nürnberg [2]

[2] Die vorliegende Arbeit entstand im Rahmen des Teilprojektes "Wissensbasierte Bildanalyse" des SFB 182. Für die Förderung der Arbeit dankt der Autor der Deutschen Forschungsgemeinschaft.

durchgeführten Arbeiten finden sich in [Bod93, Wed94].

Ein Schwerpunkt des Erlanger Sonderforschungsbereiches liegt auf der Überprüfung der realisierten Hard– und Softwarekonzepte anhand rechenintensiver Pilotanwendungen, die von der betriebswirtschaftlichen Produktionssteuerung und der Fertigungsautomatisierung über die Simulation quanten– und strömungsmechanischer Systeme bis zur Musteranalyse reichen. Ziel der im Teilprojekt "Wissensbasierte Bildanalyse" durchgeführten Untersuchungen ist die Entwicklung eines modularen Systems zur automatischen Generierung einer symbolischen Beschreibung dreidimensionaler Szenen. Die Arbeiten folgen der verbreiteten Annahme [Uhr87, Kum90, Pra91, Bur91], daß Möglichkeiten zur Parallelverarbeitung von allen Systemkomponenten genutzt werden müssen, um die durch den Bilddatenstrom vorgegebene hohe Verarbeitungsgeschwindigkeit zu erreichen. Daher werden als ausgewählte Teilaspekte sowohl Methoden zur Tiefengewinnung aus Grauwert– und Farbstereobildern [Pos90, Kol90] untersucht als auch — in der vorliegenden Arbeit — Algorithmen für die parallele Verarbeitung anwendungsabhängigen Wissens entwickelt.

Zur besseren Einordnung der hier verfolgten Thematik beschreiben wir im folgenden Abschnitt zunächst die typischen Anforderungen der einzelnen Teilaufgaben und geben in den Abschnitten 1.3 und 1.4 einen groben Überblick über bisher verfolgte Ansätze zur Parallelverarbeitung in der wissensbasierten Bildanalyse.

1.2 Wissensbasierte Bildanalyse

Bereits das im Anschluß an Bild 1.2 skizzierte Beispiel verdeutlicht, daß das Ziel der wissensbasierten Bildanalyse — die Transformation der Bilddaten (Pixel) in eine angemessene symbolische Beschreibung — keinesfalls durch eine einzelne, in geschlossener Form anzugebende Abbildung geleistet werden kann. Vielmehr wird deutlich, daß eine Reihe von Operationen und Zwischenrepäsentationen notwendig sind, die im allgemeinen in drei Ebenen organisiert werden [Bal82, Nie87]. Ausgehend von Bild 1.3, das ein geschichtetes Verarbeitungsmodell für die Bildanalyse zeigt, werden die Charakteristika der *Ikonik*, der *Ikonik–Symbolik* und der *Symbolik* in diesem Abschnitt näher beschrieben, um anschließend Möglichkeiten und Probleme der Parallelverarbeitung zu diskutieren.

Die *Ikonik*, die auch als *signalnahe Bildverarbeitung* oder *early vision* bezeichnet wird, umfaßt Methoden zur Gewinnung der Bilddaten (Abtastung, Quantisierung) und zur Vorverarbeitung der Abtastwerte, deren Ziel in einer Verbesserung der Bildqualität und der Erleichterung der Analyse besteht. Typische Algorithmen der Vorverarbeitung wie beispielsweise Schwellwertbildung, Filterung oder Normierung sind numerische Operationen auf Bildern; sie nehmen also eine Verknüpfung einer ursprünglichen Zahlenfolge zu einer neuen Folge vor.

Kennzeichnend für die ikonische Bildverarbeitung ist die *Lokalität* und die *Ortsinva-*

rianz der meisten Operationen, die Bild 1.4 anhand eines 3×3 Mittelwertfilters veranschaulicht. Während sich die Lokalität darauf bezieht, daß nur die Pixel aus einer kleinen Nachbarschaft (hier: der hellgrau dargestellten Maske) in die Berechnung des aktuellen Bildpunktes (dunkelgrau unterlegt) eingehen, wird die Ortsinvarianz darin deutlich, daß stets die gleiche, regelmäßige Abhängigkeit von den Nachbarelementen vorliegt.

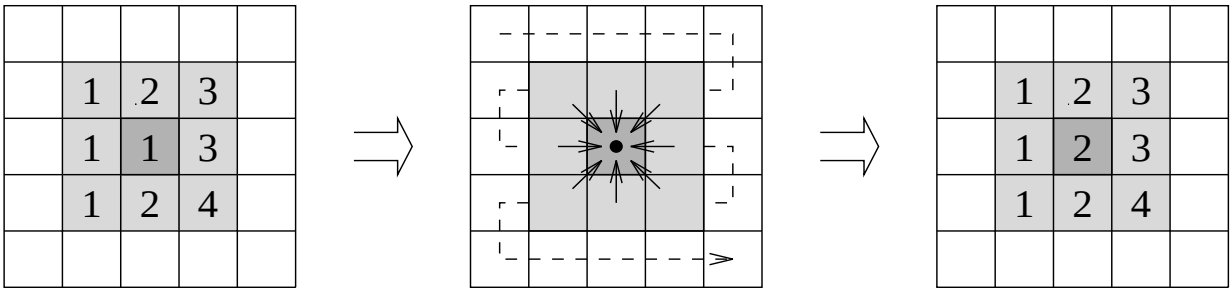


Bild 1.4: Lokalität und Ortsinvarianz ikonischer Bildverarbeitungsoperationen am Beispiel einer Maskenoperation zur Mittelwertfilterung.

Die Ergebnisse der Vorverarbeitung sind Eingangsdaten der *Ikonik–Symbolik*, deren Aufgabe die Bildsegmentierung ist. Den Ausführungen in [Lev87] folgend soll darunter nicht nur die übliche Zerlegung eines Bildes in nichtüberlappende Regionen verstanden werden, sondern allgemein alle Algorithmen, die zur Bestimmung geeigneter Bildprimitiva und einer Reduktion der visuellen Information aus der Ikonik eingesetzt werden. In Bild 1.3 sind einige Segmentierungskriterien und einfachere Bestandteile angegeben, die Elemente einer *initialen symbolischen Beschreibung* $\mathcal{A}$ sein können. In einer geeigneten Repräsentationsform — beispielsweise als Menge von Segmentierungsobjekten [Nie90a, Pau92] — bildet diese die Schnittstelle zur anschließenden wissensbasierten Interpretation. Ein Beispiel zur Segmentierung gibt Bild 1.5, das ein Grauwertbild sowie die Ergebnisse einer Kantendetektion nach [Nev80] und einer anschließenden Linienverfolgung durch ein Hystereseschwellwertverfahren [Brü90] zeigt. Hier könnten die gefundenen Linien in einem späteren Analyseschritt als Autos, Fahrbahnrand oder Verkehrszeichen interpretiert werden.

Wie auch die Algorithmen der ikonischen Bildverarbeitung kommen die in der Ikonik–Symbolik benutzten Verfahren im allgemeinen ohne die Nutzung von Wissen über den Anwendungsbereich aus. Das hier eingesetzte, problem*un*abhängige Wissen betrifft vielmehr die Eigenschaften und Auswirkungen der verwendeten Operationen, und ist in der Regel nur *implizit* im System vorhanden. So erfordert beispielsweise die Bestimmung von Tiefeninformation aus einem Stereobildpaar seitens des Systementwicklers generelles Wissen aus dem Bereich der analytischen Geometrie; dieses wird jedoch nicht explizit im System repräsentiert, sondern ist implizit in den entwickelten (Triangulations-)Algorithmen vorhanden [Nie85a].

Bild 1.5: Ein Grauwertbild und seine initiale symbolische Beschreibung.

Die *Symbolik*, die als Ebene des *Bildverstehens* bezeichnet wird, nutzt die Ergebnisse der initialen Segmentierung zur Erstellung einer symbolischen Beschreibung, die optimal zu den vorliegenden Sensordaten paßt und die für die jeweilige Anwendung relevante Information enthält. Wie bereits erwähnt, ist dazu neben der Klassifikation der (physikalischen) Objekte einer Szene auch die Bestimmung von deren Beziehungen sowie die Ermittlung und Interpretation von Ereignissen und eine entsprechende Reaktion erforderlich.

Kehren wir nochmals zu den Ergebnissen der Segmentierung in Bild 1.5 zurück und betrachten die Aussage

> "Ein *PKW*, befindet sich *rechts vor dem Kamerafahrzeug* und bewegt sich mit *einer Geschwindigkeit von* v *km/h* in die gleiche Richtung."

als mögliche Interpretation, so zeigen sich wesentliche Unterschiede zwischen Ikonik und Symbolik. Ein erster Unterschied betrifft die Art des verwendeten Wissens und dessen Darstellung. Zunächst ist offensichtlich, daß Aussagen im Objektbereich nicht ohne die Nutzung problemabhängigen Wissens möglich sind, das Erfahrungen und Fakten über Aussehen und Auftreten von Objekten des Problemkreises (hier: des PKWs), deren visuell erfaßbaren Eigenschaften (Position und Geschwindigkeit) und deren Bedeutung im Zusammenhang mit der verfolgten Zielsetzung des Systems beinhaltet.

Im Gegensatz zur impliziten Darstellung des eher generellen Wissens in der Ikonik und Ikonik–Symbolik hat sich für das problemabhängige Wissen der symbolischen Bildverarbeitung die *explizite* Repräsentation in einem *(symbolischen) Modell* $\mathcal{M}$ des Anwendungsgebietes, der *Wissensbasis*, als sinnvoll erwiesen. Die so erreichte Modularität vereinfacht die Wartung und Erweiterung des Wissens und eröffnet darüber hinaus die Möglichkeit,

dasselbe System durch den Austausch des Modells in einem anderen Anwendungsgebiet einzusetzen. Die Wahl eines geeigneten Repräsentationsformalismus und dessen effiziente Nutzung besitzen somit große Bedeutung für den Entwurf von wissensbasierten Bildanalysesystemen. Wichtige, in verschiedenen Systemen realisierte Ansätze hierzu sind *Frames* [Min75], *Produktionensysteme* [Buc85] und die auch in der vorliegenden Arbeit verwendeten *semantischen Netze* [Qui68, Fin79]. Frames werden beispielsweise im ACRONYM–System eingesetzt [Bro81], Produktionensysteme finden in [Nag80, McK85] und [Mat90b] Verwendung, und auf der Basis semantischer Netze sind die in [Han78, Tso80] beschriebenen Systeme realisiert. Während wir semantische Netze in Abschnitt 1.4 unter dem Aspekt der Parallelverarbeitung betrachten und das in dieser Arbeit verwendete ERlanger semantische NEtzwerkSysTem ERNEST [Nie90d, Sag90a] in Kapitel 2 ausführlich vorstellen, verweisen wir für eine einführende Darstellung der anderen Formalismen auf [Nie87, Sag90a].

Ein zweiter Unterschied betrifft die Anforderungen an die eingesetzten Algorithmen, die in der Ikonik durch die Ausführung numerischer Operationen auf einer sehr großen Anzahl einfachster Datenelemente — den Bildpunkten oder einer kleinen Nachbarschaft — gekennzeichnet waren. Die Verfahren der symbolischen Bildverarbeitung arbeiten dagegen mit bedeutend weniger, jedoch weitaus komplexeren Elementen, nämlich den Datenstrukturen zur Repräsentation der initialen Segmentierung und der Objekte der Wissensbasis.

Kann der Aufwand für ikonische Operationen aufgrund der Ortsinvarianz im wesentlichen aus der Bildgröße bestimmt werden, so ist eine entsprechende Abschätzung für die seitens der Symbolik erforderliche Zuordnung von Sensor– bzw. Segmentierungsdaten und den Elementen der Wissensbasis im allgemeinen nicht möglich. Neben der Tatsache, daß die Anzahl der zu berücksichtigenden Bildprimitiva wesentlich vom Szeneninhalt abhängt, liegt ein maßgeblicher Grund darin, daß die Ergebnisse der Segmentierung beim heutigen Stand der Technik im allgemeinen mehrdeutig und fehlerbehaftet sind (vgl. Bild 1.5); Ursachen dafür können wechselnde Beleuchtungsverhältnisse in einer Szene, das Auftreten komplizierter, dreidimensionaler Objekte oder Fehler aufgrund des Quantisierungsrauschens sein.

Aufgrund der Mehrdeutigkeit und Unzuverlässigkeit der Segmentierungsergebnisse besteht die Notwendigkeit, den Analyseprozeß durch einen *Kontrollalgorithmus* zu steuern. Dieser nimmt eine Fokussierung auf erfolgsversprechende Zwischenergebnisse und relevante Ausschnitte der Wissensbasis vor und legt fest, welche Algorithmen auszuführen sind, um ein möglichst gutes Analyseergebnis zu erzielen. Die Lösung des hier angedeuteten Optimierungsproblems bedingt die Ausführung unterschiedlicher Algorithmen auf den Zwischenergebnissen der Analyse. So könnten die zur Fahrbahn gehörigen Liniensegmente aus Beispiel 1.5 bereits dem entsprechenden Modell aus der Wissensbasis zugeordnet werden, während etwa im Bereich der weiter entfernten Fahrzeuge mit Hilfe eines anderen Segmentierungsverfahrens zusätzliche Information gewonnen werden könnte, um eine bessere Zuordnung zu ermöglichen. Die Diversität der benötigten Operationen stellt somit neben der Notwendigkeit zur Bearbeitung unterschiedlicher Datenobjekte und der szenenabhängi-

gen Komplexität einen dritten Unterschied zur Ikonik dar, der die variablen Anforderungen der Algorithmen der Symbolik unterstreicht.

Bevor wir uns den Auswirkungen der hier beschriebenen Charakteristika auf die Konzepte zur Parallelverarbeitung in der wissensbasierten Bildanalyse zuwenden, müssen zwei abschließende Bemerkungen der gewählten Darstellung gelten: Zum einen hat sich die Einteilung in drei Verarbeitungsschichten an der zunehmend abstrakteren Repräsentation der Zwischenergebnisse der Analyse (Pixel – Bildprimitiva – physikalische Objekte) orientiert. Die Grenzen zwischen den Ebenen sind jedoch als fließend anzusehen und andere Einteilungen sind möglich. So ist eine Zweiteilung in Ikonik und Symbolik denkbar, wenn als Kriterium die Art des verwendeten Wissens zugrunde gelegt wird; danach würde die Ikonik sämtliche problemunabhängigen Algorithmen umfassen, die Symbolik hingegen alle Methoden, die durch eine Nutzung problemabhängigen Wissens gekennzeichnet sind. Zum anderen sind als Alternativen zu der oben unterstellten rein *datengetriebenen* Verarbeitung, die mit den Abtastwerten des Bildes beginnt und diese schrittweise in abstraktere Repräsentationen transformiert, auch *modellgetriebene* und *bidirektionale* Strategien zu erwähnen. Während bei der modellgetriebenen Analyse versucht wird, Hypothesen über den Szeneninhalt anhand der Bilddaten zu verifizieren, stellt die bidirektionale Strategie eine Mischform dar, bei der die Ergebnisse einer initialen datengetriebenen Verarbeitungsphase zur modellgetriebenen Verifikation von Hypothesen verwendet werden.

1.3 Parallele ikonische Bildverarbeitung

Ziel der Entwicklung von Algorithmen für parallele Rechensysteme ist es, die Bearbeitungsdauer für ein gegebenes Problem durch die Ausnutzung der Rechenleistung mehrerer Prozessoren zu verkürzen. Etablierte Kriterien für die Beurteilung des Erfolgs der Parallelisierung (siehe z.B. [Kum94]) sind der Geschwindigkeitsgewinn (engl.: *Speedup*) s_p und die *Effizienz* e_p. Der Speedup

$$s_p = \frac{t_1}{t_p} \tag{1.1}$$

ist der Quotient aus der Laufzeit t_1 des (schnellsten) sequentiellen Algorithmus und der Zeit t_p, die zur Ausführung des parallelen Algorithmus mit p Prozessoren benötigt wird. Die Effizienz

$$e_p = \frac{s_p}{p} = \frac{t_1}{p \cdot t_p} \tag{1.2}$$

ist ein Maß für die Auslastung der verwendeten Prozessoren; sie wird berechnet, indem der Speedup durch die Prozessoranzahl dividiert wird.

Mit zunehmender Anzahl der Prozessoren nehmen auch die Speedupverluste zu, die

durch *Auslastungs–*, *Synchronisations–* und *Zugriffsverluste* verursacht werden [Bur89]. Auslastungsverluste entstehen durch einen schwankenden Parallelitätsgrad, Synchronisationsverluste treten auf, wenn ein Prozessor auf Daten wartet, die von anderen Prozessoren berechnet werden müssen, und Zugriffsverluste entstehen beim Warten auf gemeinsam benutzte Betriebsmittel. Aufgrund dieser Verluste ist zu beobachten, daß der Speedup im allgemeinen unter dem idealen, *linearen Speedup* $s_p = t_1/p$ liegt und ab einer gewissen Prozessoranzahl gesättigt ist oder sogar absinkt. Speedup und Effizienz sind deshalb nicht unabhängig voneinander zu maximieren; abgeleitete Größen wie die *Wirtschaftlichkeit*

$$w_p = s_p \cdot e_p, \tag{1.3}$$

durch deren Maximierung ein "optimaler Arbeitspunkt" eingestellt werden kann [Fla89], können jedoch zur Ermittlung einer Prozessoranzahl genutzt werden, die akzeptable Werte für *beide* Größen liefert.

Die Ortsinvarianz und überwiegende Lokalität der Operationen auf Bildmatrizen erleichtern deren Parallelisierung erheblich, da sie intuitiv einsichtige Parallelisierungsmöglichkeiten eröffnen und eine gleichmäßige Auslastung der Prozessoren ermöglichen, woraus große Werte für Speedup und Effizienz resultieren. Neben der steigenden Verfügbarkeit paralleler Rechensysteme ist die hier skizzierte konzeptuelle Einfachheit die Hauptursache dafür, daß mittlerweile eine Vielzahl paralleler Implementierungen von ikonischen Algorithmen existieren, die nahezu das gesamte Spektrum der Ikonik, und in zunehmendem Maße auch der Ikonik–Symbolik abdecken. Im vorliegenden Abschnitt soll nicht der Versuch unternommen werden, diese auch nur annähernd zu erläutern — dafür sei auf zahlreiche ausgezeichnete Übersichten [Cyp89, Cha90, Pit93] verwiesen —; vielmehr geht es uns darum, die grundlegende Vorgehensweise gegenüber der Parallelverarbeitung im Bereich der Symbolik (vgl. Abschnitt 1.4) abzugrenzen.

Die räumliche Unabhängigkeit der Pixel bzw. kleiner Bildausschnitte erlaubt eine Zerlegung des Bildes in Teilbilder und deren unabhängige Bearbeitung durch mehrere Prozessoren. Die Bildgröße ($N \times N$ Pixel) bildet dabei eine obere Grenze für die Anzahl der verwendeten Rechnerknoten, die jedoch beim heutigen Stand der Technik nicht wesentlich größer als 128^2 sein kann. Daher wird die Bildmatrix im allgemeinen in Blöcke der Größe M^2 ($M < N$) oder zeilen– bzw. spaltenweise in $M \times N$ Pixel große Streifen eingeteilt, und auf jedem der Teilbilder wird parallel der gleiche Algorithmus ausgeführt; dies ist in Bild 1.6 für $N = 8$ und $M = 4$ schematisch dargestellt.

Die angedeutete überlappende Einteilung ermöglicht es, einzelne Operatoren wie etwa die Mittelwertfilterung aus Bild 1.4 ohne Datentransfer zwischen den Rechnerknoten auszuführen. Der Aufbau komplexerer Verarbeitungsketten erfordert lediglich eine Erweiterung der sequentiellen Algorithmen um den Austausch der Randdaten, der einen konstanten und a priori bekannten Kommunikationsaufwand verursacht: die Betrachtung einer

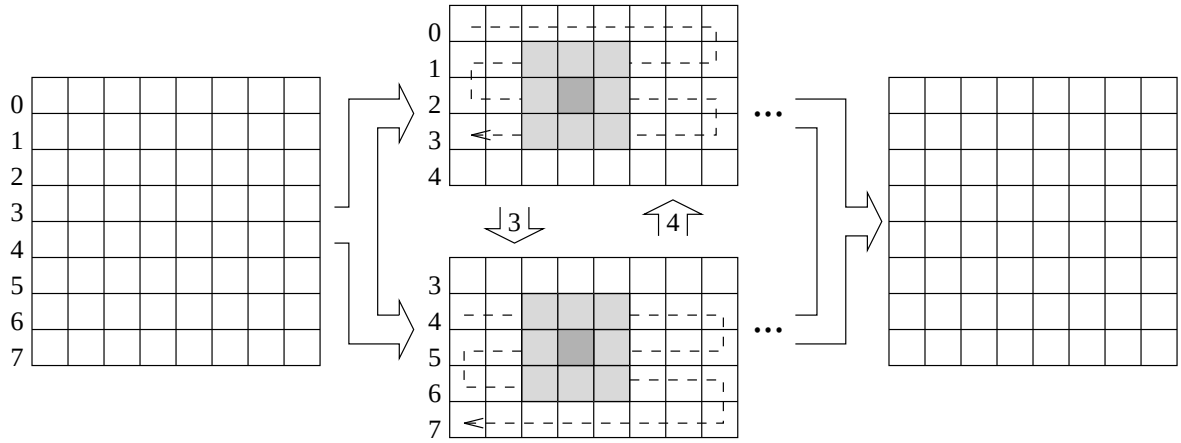


Bild 1.6: Datenparallelität ikonischer Bildverarbeitungsalgorithmen.

Nachbarschaft der Größe $(2m+1)^2$, $m = 1, 2, \ldots$ erfordert den Austausch von $2m$ Spalten oder Zeilen (in Bild 1.6 ist $m = 1$) der Länge N zwischen je zwei Prozessoren.

Überlegungen zur Hardwarearchitektur paralleler Bildanalysesysteme finden sich in [Han80, Uhr87, Pra91]. Als Konsequenz aus der Datenparallelität ikonischer Algorithmen wurden zunächst Architekturen vorgeschlagen, die aus einer großen Anzahl einfacher Verarbeitungselemente (Prozessoren) und einer zentralen Kontrolleinheit aufgebaut sind. Zahlreiche solcher Systeme, die im Sinne der Flynnschen Klassifikation [Fly72] als SIMD–Rechner (*single instruction multiple data*) zu bezeichnen sind, werden in [Cyp89] beschrieben; gegenwärtig ist jedoch ein Trend zur Realisierung auf der Basis zunehmend leistungsfähigerer und kostengünstigerer MIMD–Rechner (*multiple instruction multiple data*) festzustellen [Sch90a, Sou92].

Die wichtigsten Verbindungsstrukturen beider Klassen sind die gitterförmige Anordnung mit einer optionalen Verbindung der Randprozessoren zu einem Torus, die pyramidenartige Anordnung und der n–dimensionale Würfel (*Hypercube*) (vgl. Bild 1.7).

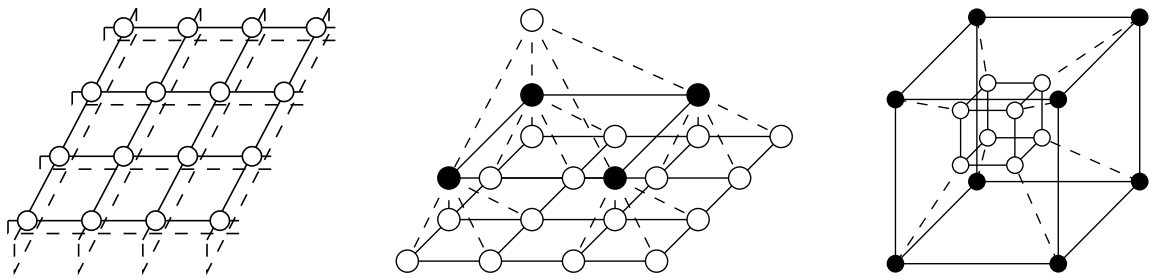


Bild 1.7: Parallelrechnerstrukturen in der Bildverarbeitung: 4×4 Gitter mit Abschluß zum Torus (gestrichelt), Pyramide mit 4×4–Basis und Hypercube der Dimension 4. Knoten entsprechen Prozessoren, Kanten stellen Kommunikationspfade dar.

Während gitterförmige Verbindungen vor allem die lokalen Nachbarschaftsoperationen

der Ikonik unterstützen [Lee90], bieten sich SIMD–Pyramiden auch für die Verwendung einer Auflösungshierarchie an [Tan83] und werden mit MIMD–Rechnern auf den oberen Ebenen auch als Plattform für komplette Bildanalysesysteme vorgeschlagen [Shu90, Sun90b]. Die flexible Topologie des Hypercubes, die auch Verbindungen zwischen "entfernten" Prozessoren vorsieht, erlaubt eine effiziente Parallelisierung ikonischer Algorithmen [Ran89, Pan90] und hat auch zu dessen Einsatz in der Objekterkennung geführt [Tuc91].

Eine weitgehend flexible und kostengünstige Alternative zu den hier vorgestellten Ansätzen bildet die Nutzung von Workstations, die in einem lokalen Netzwerk gekoppelt werden, und sich aufgrund komfortabler Programmierumgebungen und Bedienoberflächen auch als Schnittstellen zu speziellen Architekturen anbieten [Tav95]. Unterstützt das Betriebssystem die Nutzung der parallelen Ressourcen durch die Bereitstellung geeigneter Kommunikations– und Synchronisationsmechanismen, so bietet sich diese Lösung insbesondere bei einem geringen Parallelitätsgrad an [Ree91], und ermöglicht eine einfache funktionale Dekomposition modular aufgebauter Bildanalysesysteme, die für Monoprozessorarchitekturen entwickelt wurden.

1.4 Parallele symbolische Bildverarbeitung

Obwohl die Notwendigkeit zur Parallelverarbeitung auf allen Ebenen der Bildanalyse in den zahlreichen, bereits zitierten Übersichten allgemein anerkannt wird, ist die Parallelisierung der wissensverarbeitenden Algorithmen der Symbolik im Vergleich zur Ikonik weit weniger fortgeschritten. Wesentliche Ursachen dafür liegen zum einen in der Abhängigkeit von den fehlerbehafteten, unsicheren Ergebnissen der Segmentierung und zum anderen in den unregelmäßigen Beziehungen zwischen den Elementen der Wissensbasis, die während der Analyse zu beachten sind. Dadurch variiert sowohl der Aufwand einzelner Verarbeitungsschritte als auch die Anzahl gleichzeitig ausführbarer Operationen, und die gleichmäßige Auslastung der zur Verfügung stehenden Hardware wird durch den schwankenden Parallelitätsgrad erheblich erschwert. Schließlich trägt auch das Fehlen offensichtlicher Parallelisierungsmöglichkeiten, wie sie die Ikonik durch eine Zerlegung der Bildmatrix offeriert, dazu bei, daß bislang zwar einige Vorschläge zur Hardwarearchitektur für die symbolische Bildverarbeitung vorliegen [Sha87a, Lev87, Shu90, Sun90b], Algorithmen jedoch hauptsächlich für ausgewählte Teilprobleme wie beispielsweise parallele Suchverfahren vorliegen [Kum90].

Ist als Konsens der Überlegungen zur Architektur die Forderung nach einer flexiblen Kopplung vergleichsweise weniger, jedoch mächtiger MIMD–Prozessoren hervorzuheben, so kann als Gemeinsamkeit der Entwicklung paralleler Algorithmen lediglich die Notwendigkeit der *Aufgabenverteilung* festgehalten werden. Dies erfordert die Identifizierung gleichzeitig ausführbarer Teilaufgaben in den zur Nutzung des problemabhängigen Wissens verwendeten Inferenzmechanismen. Für die vorliegende Arbeit ist es daher sinnvoll, beste-

hende Ansätze zur Parallelverarbeitung in semantischen Netzen zunächst übersichtsartig zusammenzufassen und auf ihre Eignung für die Musteranalyse zu untersuchen. Parallelisierungsmöglichkeiten in anderen Repräsentations– bzw. Inferenzformalismen werden beispielsweise in [Gup87, For91] diskutiert.

Parallelverarbeitung in semantischen Netzen

Die unter dem Begriff "Semantische Netzwerke" zusammengefaßten Formalismen zur Wissensrepräsentation gehen in ihrer heutigen Form auf Arbeiten zur Modellierung des menschlichen Gedächtnisses zurück [Qui68]; Überblicke über seitdem entwickelte Ansätze und wichtige Zwischenstadien finden sich in [Bra79, Sag90a, Rei92].

Obwohl für semantische Netze bislang keine eindeutige und allgemein anerkannte Definition existiert, ist allen Ansätzen gemeinsam, daß sie Wissen in Form eines gerichteten, markierten Graphen darstellen. Die Knoten des Graphen dienen zur Aufnahme beliebiger Begriffe oder Sachverhalte, und durch die Kanten werden Beziehungen zwischen diesen ausgedrückt. Sowohl die Knoten als auch die Kanten sind mit einem Namen markiert, der ihre Bedeutung angibt. Bild 1.8 zeigt ein einfaches Beispiel, das auch verdeutlicht, daß die Knoten und Kanten zunächst keine feste Bedeutung besitzen. So können Knoten sowohl Individuen ("Charley") als auch Klassen ("Fisch") und Unterklassen ("Goldfisch") repräsentieren und Kanten können Beziehungen zwischen Klassen ("Fisch" und "Flosse"), Klassen und Unterklassen ("Fisch" und "Goldfisch"), oder auch zwischen Klassen und Eigenschaften ("Fisch" und "golden") ausdrücken.

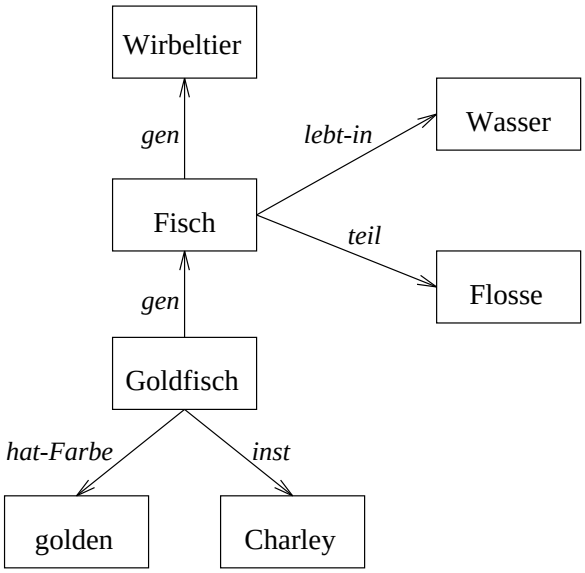


Bild 1.8: Beispiel eines semantischen Netzwerks.

In [Woo75, Bra77] wird dargelegt, daß dieser — insbesondere bei frühen Netzwerkforma-

lismen feststellbare — Mangel an formaler Semantik eine Interpretation genau genommen nicht zuläßt, obwohl die Beschriftungen der Knoten und Kanten eine bestimmte Bedeutung suggerieren. Eine effiziente Nutzung des Wissens wird erst durch die Beschränkung auf eine möglichst kleine, jedoch genügend ausdrucksstarke Menge von *epistemologisch primitiven* Sprachkonstrukten erreicht [Bra79]. Die meisten der seitdem entwickelten Netzwerkformalismen — zu nennen sind hier insbesondere KL-ONE [Bra85a] und seine zahlreichen Derivate (z.B. KRYPTON [Bra85b], NIKL [Kac86] und SB-ONE [Pro90]) sowie PSN [Lev79] und ERNEST [Nie90d, Sag90a] — besitzen daher eine klare syntaktische Definition der Netzwerksprache und beschränken sich auf wenige Knoten– und Kantentypen.

Konzepte und *Instanzen* sind die am weitesten verbreiteten Knotentypen, die eine Unterscheidung zwischen *intensionaler* und *extensionaler* Repräsentation von Begriffen ermöglichen. Während die intensionale Beschreibung die charakteristischen Eigenschaften eines Begriffes definiert, umfaßt dessen Extension alle konkreten Exemplare von Objekten oder Sachverhalten, die diese Definition erfüllen. So ist in Bild 1.8 ein Goldfisch als goldfarbener Fisch definiert, und "Charley" ist eine Instanz des Konzepts "Goldfisch". Letzteres wird durch die *Instanzkante* (*inst*) ausgedrückt. In [Bra79] wird kritisiert, daß Instanzkanten in vielen semantischen Netzwerken zur Darstellung zweier verschiedener Sachverhalte benutzt werden: zum einen zur Bezeichnung von Relationen zwischen Konzepten, und zum anderen zur Darstellung von Beziehungen zwischen intensionaler und extensionaler Beschreibung. Die zwischen den Konzepten "Charley" und "Goldfisch" bestehende Relation wird daher als *Individualisierung* bezeichnet, und der Begriff der *Instantiierung* bleibt allein den Beziehungen zwischen Konzepten und ihren Ausprägungen in der realen Welt vorbehalten. Dieser Unterscheidung folgend wird die Instanzkante auch in ERNEST im Sinne einer Klassifikation von Sensordaten ausschließlich zwischen Konzepten und den zugehörigen Signalausschnitten, den Instanzen, benutzt.

Neben der Instanzkante existieren in den meisten semantischen Netzen zwei weitere Standardrelationen zwischen Knoten: die *Bestandteilskante*, (*teil*, engl.: *has–a, part*) ermöglicht eine Zerlegung komplexer Begriffe in ihre natürlichen Bestandteile (vgl. die Kante zwischen den Konzepten "Fisch" und "Flosse"), und die *Generalisierung* (*gen*, engl.: *is–a*), welche die Zusammenfassung von Konzepten zu (Konzept-)klassen ermöglicht. Die Generalisierung etabliert die *Vererbung* als einzigen Inferenzmechanismus, der in nahezu allen Netzwerkformalismen anzutreffen ist. Die Vererbung der Eigenschaften und Bestandteile erfolgt dabei stets von der Oberklasse auf die Unterklasse; in Bild 1.8 ererbt das speziellere Konzept "Goldfisch" die Eigenschaft, im Wasser zu leben, und das Bestandteil "Flosse" des allgemeineren Konzepts "Fisch".

Die syntaktische Definition von Konzepten als komplexe Datenstrukturen mit *Attributen* und *strukturellen Relationen* [Bra79] wurde wesentlich durch die in [Min75] vorgeschlagenen Frames beeinflußt, deren Grundidee in der Modellierung von Objekten und Ereignissen in einem speziellen situativen Kontext besteht. Frames bestehen aus mehreren

Einträgen (*slots*), für die Vorerwartungen oder Bedingungen angegeben werden können und die während der Verarbeitung mit Werten gefüllt werden müssen; Bild 1.9 zeigt eine einfache Framenotation für einen Ausschnitt des obigen Beispielnetzwerkes.

Frame	: Fisch
ist-ein	: Wirbeltier
Lebensraum	: Ort
Anzahl-Flossen	: integer
Farbe	: ..., golden, ...

Frame	: Goldfisch
ist-ein	: Fisch
Lebensraum	: Wasser
Anzahl-Flossen	: 3
Farbe	: golden

Bild 1.9: Framerepräsentation der Konzepte "Fisch" und "Goldfisch" aus Bild 1.8.

Zur Ermittlung konkreter Werte ist die Anbindung von Prozeduren (*procedural attachment*) an die Einträge der Frames möglich, wodurch framebasierte semantische Netze um eine prozedurale Komponente ergänzt werden. Da hierdurch auch die Berechnung von Attributwerten und Relationen aus Sensordaten unterstützt wird, kann der Formalismus für die Musteranalyse genutzt werden, was beispielsweise durch das mit PSN realisierte Musteranalysesystem ALVEN [Tso80] und die zahlreichen mit der ERNEST–Systemschale realisierten Anwendungen (vgl. die Literaturangaben in Kapitel 2) demonstriert wird.

Als besondere Vorteile semantischer Netze werden im allgemeinen die Anschaulichkeit der Repräsentation und die objektzentrierte Darstellung aller zu einem Begriff vorhandener Fakten in einem Konzeptknoten betont, da dies den Aufbau größerer Wissensbasen erleichtert. In [Rei92, Sha91] wird darüberhinaus die Möglichkeit zur Parallelisierung verschiedener, in semantischen Netzen definierter Inferenzmechanismen hervorgehoben. Als richtungsweisender Beitrag, dessen Einfluß auch in einer Reihe von neueren Ansätzen (siehe unten) deutlich wird, ist hierbei das NETL–System [Fah79] zu erwähnen, das Inferenzprozesse parallelisiert, die auf dem bereits in [Qui68] vorgeschlagenen Prinzip der *Aktivierungsausbreitung* (*spreading activation*) zur Bearbeitung von Anfragen an eine Wissensbasis beruhen.

Das ursprüngliche Ziel der Aktivierungsausbreitung bestand im Auffinden einer Kantenverbindung zwischen zwei beliebigen Knoten eines Netzwerkes, die — umgesetzt in ein "Ich Tarzan, Du Jane"–Englisch [3] — Auskunft über eine semantische Beziehung zwischen den Knoten gibt. Hierzu werden die beiden Ausgangsknoten mit einer Marke belegt, die anschließend über alle von den bereits markierten Knoten ausgehenden Kanten an die Nachbarknoten propagiert wird. Dabei werden jedoch bereits markierte Knoten nicht mehr berücksichtigt. Das Verfahren liefert eine erste Lösung, sobald ein Knoten von zwei verschie-

[3] Dies ist eine Wertung Quillians [Qui68], nicht etwa des Autors der vorliegenden Arbeit

denen Nachbarknoten aktiviert wird, oder sich zwei Marken beim Passieren einer Kante begegnen. Bevor die gefundene Kantenverbindung als Anfrageergebnis ausgegeben wird, kann ihre Zulässigkeit überprüft werden und zahlreiche Modifikationen des beschriebenen Basisalgorithmus (wie beispielsweise die Abschwächung oder Terminierung der Ausbreitung durch bestimmte Knoten– oder Kantentypen und die Beschränkung der Länge des Kantenzuges) sind möglich. Eine Übersicht über weitere Ansätze zur Aktivierungsausbreitung findet sich in [Hen88].

NETL ist ein Softwaresystem, das die Ausführung einfacher Inferenzmechanismus nach dem Prinzip der Aktivierungsausbreitung durch eine massiv–parallele Hardware simuliert. Dazu werden — in Abwandlung des eben beschriebenen Vorgehens — mehrere verschiedene Marken unter der Steuerung einer zentralen Kontrolleinheit verarbeitet. Das System gestattet die simultane Propagierung von Marken sowie die Ausführung einfacher, logischer Operationen auf diesen und ermöglicht dadurch die Beantwortung von solchen Anfragen an ein Netzwerk, die sich auf die Bestimmung gemeinsamer Eigenschaften von Konzepten oder die Ermittlung von Vererbungen in einer Generalisierunghierarchie zurückführen lassen.

Bild 1.10 zeigt an einem Beispiel, daß solche Anfragen durch die parallele Propagierung von Marken mit einem Zeitaufwand beantwortet werden können, der proportional zum Durchmesser des Netzwerkes und unabhängig von dessen Knotenanzahl ist.

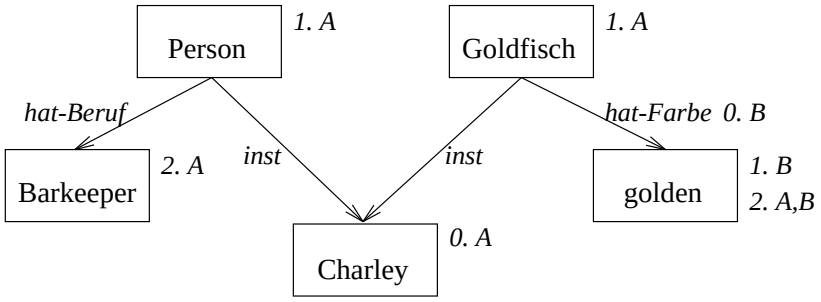


Bild 1.10: Inferenzmechanismen in NETL: Parallele Aktivierungsausbreitung zur Beantwortung von Anfragen an ein Netzwerk.

Um die einfache Frage *"Welche Farbe hat Charley ?"* zu beantworten, werden zunächst zwei Marken A und B im Knoten "Charley" und in der Kante *hat-Farbe* plaziert. Im ersten Schritt wird A über die "*inst*"–Kanten nach *oben* propagiert, wodurch "Person" und "Goldfisch" markiert werden; *zugleich* aktiviert die Marke B den Knoten "golden". Anschließend wandert A über die Kante *hat-Farbe* zum Knoten "golden" und über *hat-Beruf* zum Knoten "Barkeeper" (sowie zu möglichen weiteren, nicht dargestellten Knoten). Ist die Kontrolleinheit so programmiert, daß sie alle Knoten meldet, die zugleich mit A und B markiert sind, wird "golden" als richtige Antwort nach 2 Schritten gefunden, nicht jedoch der nur mit A markierte "Barkeeper"–Knoten.

Vergleichbare Inferenzmechanismen werden auch von Formalismen bereitgestellt, die *Petrinetze* zur Repräsentation und Nutzung von Wissen verwenden [Fid86, Rib88, Rib93]. Petrinetze sind bipartite Graphen, die *Bedingungen* und *Transitionen* als Knotentypen

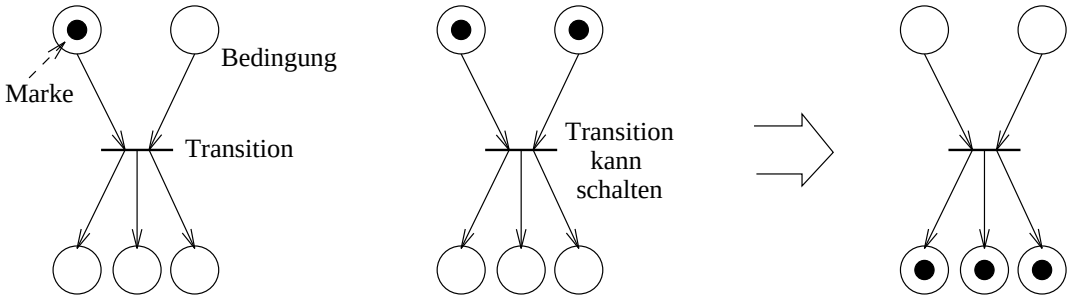


Bild 1.11: Ein Petrinetz (links) und die Propagierung von Marken durch Aktivierung einer Transition.

besitzen (vgl. Bild 1.11). Knoten verschiedenen Typs sind durch gerichtete Kanten verbunden. Bedingungen können Marken aufnehmen und diese über Transitionen an deren Nachbedingungen abgeben. Eine Transition wird aktiv, wenn alle Vorbedingungen mit mindestens einer Marke belegt sind; die Wirkung ist in Bild 1.11 dargestellt. Zahlreiche Varianten des hier skizzierten Basismechanimus, die beispielsweise auch das Vorhandensein inhibitorischer Kanten umfassen, sind in der weiterführenden Literatur zu finden [Rei86].

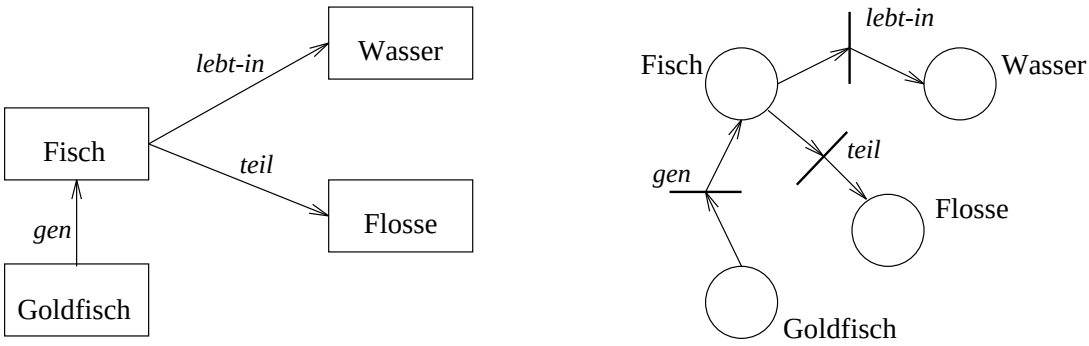


Bild 1.12: Ein semantisches Netzwerk (links) und seine Darstellung als Petrinetz (rechts) nach [Rib93].

In [Rib88, Rib93] werden — wie in Bild 1.12 abgebildet — Konzepte und Kanten eines semantischen Netzes auf Bedingungen und Transitionen eines Petrinetzes abgebildet. Zur Beantwortung von Anfragen werden formal definierte, dynamische Eigenschaften von Petrinetzen ausgenutzt; insbesonders wird die Frage nach gleichzeitig markierten Bedingungen (Konzepten) durch den Vergleich von *Erreichbarkeitsbäumen* entschieden.

Der wesentliche Beitrag des NETL–Systems zur Parallelverarbeitung in semantischen Netzen liegt in der erstmals aufgestellten Fordung nach einer sehr großen Anzahl einfach-

ster Verarbeitungs– und Verbindungselemente, welche die direkte Abbildung der Konzepte und Kanten einer Wissensbasis auf die Hardware eines parallelen Rechensystems erlaubt. Dieser Vorschlag ist von zahlreichen folgenden Ansätzen [Fah83, Bic85, Sha87b, Der87, Sha88, Der90, Eve94] aufgegriffen worden und wird in [Sha89, Sha91] als Grundvoraussetzung für die Entwicklung sehr schneller ("reflektorischer") Inferenzprozesse bezeichnet. In [Mol89] wird mit dem *Semantic Network Array Processor* SNAP eine massiv–parallele, auf die Belange der Aktivierungsausbreitung spezialisierte SIMD–Architektur entworfen; Simulationsergebnisse für verschiedene Aufgaben wie zum Beispiel die Einordnung eines Konzeptes in eine Vererbungshierarchie (Subsumption) und die Bestimmung eines Konzepts, das eine Menge vorgegebener Eigenschaften am besten erfüllt (Klassifikation), finden sich in [Kim90, Mol90].

Die in diesem Abschnitt zusammengefaßten Arbeiten zeigen einerseits, daß semantische Netzwerke ein großes Potential an Möglichkeiten zur Parallelverarbeitung besitzen, beschränken sich andererseits jedoch auf einfach aufgebaute Konzepte und Inferenzmechanismen, die als Mengenoperationen (Durchschnitt, Vereinigung, Komplement oder transitiver Abschluß) auf den diskreten Marken beschrieben werden können. Der "Alles-oder-Nichts"–Charakter dieser Inferenzen — ein bestimmtes Konzept kann entweder markiert werden oder nicht — wird in einigen der oben genannten Ansätze durch den Austausch kontinuierlicher Werte (*"value–passing–systems"*, z.B. [Fah83]) und deren Manipulation durch komplexere arithmetische Operationen aufgehoben [Sha88, Der90]. Dadurch wird zwar der Vergleich *unsicherer* Fakten mit den Einschränkungen und Vorerwartungen einer Wissensbasis möglich; die auf Seite 17 als für die Musteranalyse notwendig erkannte Anbindung beliebiger Analyseprozeduren fehlt jedoch in sämtlichen parallelen Netzwerksystemen. Die Verwendung framebasierter Konzeptdatenstrukturen zur ergonomisch adäquaten Darstellung des Wissens stellt dagegen kein Problem dar: die automatische Generierung einer feingranularen Netzwerkrepräsentation aus den komplexen Konzeptdefinitionen findet in [Sha88, Der90] Verwendung.

Nach diesem Überblick über bisherige Ansätze zur Parallelverarbeitung in semantischen Netzen skizzieren die folgenden Abschnitte die Zielsetzung der durchgeführten Untersuchungen und den Aufbau der Arbeit.

1.5 Zielsetzung und Beitrag der Arbeit

Während bereits handelsübliche, preiswerte Arbeitsplatzrechner dem Menschen bei der Lösung einer Vielzahl von streng formalisierbaren Problemen — wie beispielsweise dem Auffinden von Datensätzen in großen Datenbanken oder der Lösung größerer Gleichungssysteme — zunehmend überlegen sind, können selbst herkömmliche Höchstleistungsrechner die Zeitanforderungen von Aufgaben, die perzeptive Fähigkeiten verlangen, nur selten

erfüllen. Die Tatsache, daß der Mensch derartige Aufgaben sehr gut und offensichtlich mühelos bewältigt, wird häufig der massiv–parallelen Verarbeitung im Gehirn zugeschrieben, die somit eine starke Motivation für den Einsatz von Parallelrechnern darstellt. Trotz der allgemein anerkannten Auffassung, daß der Entwurf leistungsfähiger Musteranalysesysteme zur Nachbildung der menschlichen Perzeptions– und Interpretationsfähigkeit die Nutzung von Parallelisierungsmöglichkeiten auf allen Verarbeitungsebenen erfordert, beschränkt sich der Einsatz von Parallelrechnern in der Bildverarbeitung bislang jedoch vorwiegend auf die Ikonik.

Motiviert durch diese Tatsache untersucht die vorliegende Arbeit Möglichkeiten zur Parallelverarbeitung in ERNEST, einem wissensbasierten Musteranalysesystem auf der Grundlage semantischer Netze. Dabei wird von einer modularen Systemarchitektur ausgegangen, bei der die Sensordaten über mehrere Abstraktionsebenen in eine symbolische Beschreibung (Interpretation) transformiert werden; die hier beschriebenen Untersuchungen konzentrieren sich auf die effiziente Nutzung des problemabhängigen Wissens durch einen parallelen Kontrollalgorithmus. Die in den Gleichungen (1.1 – 1.3) definierten Kriterien werden dabei zur Evaluierung der entwickelten Verfahren herangezogen, die — bedingt durch die Einbettung der Arbeit in das Teilprojekt "Wissensbasierte Bildanalyse" des Sonderforschungsbereichs 182 und motiviert durch die oben skizzierten Rechenzeitanforderungen — an einem Szenario zur Bildfolgenanalyse natürlicher Szenen getestet werden. Die durchgeführten Untersuchungen stellen somit insbesondere einen Beitrag zur parallelen, symbolischen Bildverarbeitung dar.

Als ein Ergebnis der Arbeit zeigen die in einem Workstationnetzwerk erzielten Effizienzen von bis zu 90 Prozent, daß eine erfolgreiche Parallelisierung der wissensbasierten Verarbeitung möglich ist. Die dazu notwendige Neuformulierung der Analyseaufgabe als *Optimierungsproblem* nimmt eine Trennung der Netzwerkebene, auf der die zur Wissensnutzung notwendigen Instantiierungsoperationen parallelisiert werden, von der Parallelisierung der übergeordneten Verfahren zur Optimierung des Analyseergebnisses vor. Dies ermöglicht es einerseits, die erzielten Ergebnisse für die Netzwerkebene mit Parallelisierungsansätzen in anderen Formalismen zu vergleichen, andererseits werden Resultate zur Parallelisierung allgemein einsetzbarer, stochastischer Optimierungsverfahren gewonnen.

In vielen zukünftigen, teilweise auch schon im Laborstadium vorliegenden informationsverarbeitenden Systemen — als mögliche Beispiele seien etwa autonome, mobile Roboter oder echtzeitfähige Dialogsysteme genannt — werden neben der geforderten Verarbeitungsgeschwindigkeit weitere Aspekte an Bedeutung gewinnen:

- Die Interaktion derartiger Systeme mit einer Umwelt, die dynamischen Veränderungen unterliegt, erlaubt es nicht, eine feste Zeitschranke für die Verarbeitungsdauer vorzugeben. Daher ist die Entwicklung von Verfahren erforderlich, die jederzeit zu unterbrechen sind und das frühzeitige Erreichen einer möglichst guten Interpretation

sicherstellen, die an andere Systemkomponenten (beispielsweise zur Einleitung einer raschen Systemreaktion) weitergegeben werden kann. Steht dagegen genügend Zeit zur Verfügung, so soll ein *optimales* Analyseergebnis erzielt werden. Die hier skizzierten Anforderungen werden auch unter dem Begriff der *Any–Time–Eigenschaften* zusammengefaßt.

- Die zunehmend wichtigere Verarbeitung multi–modaler Sensordaten, beispielsweise von Bildfolgen und gesprochener Sprache, innerhalb *eines* Analysesystems erfordert einen flexiblen Formalismus zur Darstellung und Nutzung von Wissen aus unterschiedlichen Anwendungsbereichen und legt die Entwicklung eines problemunabhängigen Kontrollalgorithmus nahe.

- Die potentielle Zunahme der Verarbeitungsgeschwindigkeit und die größere Speicherkapazität von Parallelrechnern eröffnen die Möglichkeit des Entwurfs von komplexen Analysesystemen mit sehr großen Wissensbasen. Da die manuelle Erstellung von Wissensbasen aufgrund bislang nur unzureichender Verfahren zur automatischen Wissensakquisition den momentanen Stand der Technik darstellt, sind auch Aspekte der ergonomischen Repräsentation zu berücksichtigen, um eine einfache Wartung und *Erweiterung* von Wissen zu gewährleisten.

Die hier genannten Anforderungen werden durch die folgenden Überlegungen berücksichtigt:

Optimalität der Interpretation: Die von der symbolischen Verarbeitung zu leistende Abbildung der Segmentierungsergebnisse auf die Begriffe einer Wissensbasis wird als *Optimierungsaufgabe* formuliert. Optimierungstechniken kommen in zahlreichen natur– und ingenieurswissenschaftlichen Aufgabenstellungen erfolgreich zum Einsatz. In Musteranalysesystemen finden derartige Verfahren bislang jedoch hauptsächlich in der Erkennungsphase Anwendung — hervorzuheben sind hier neben dem überragenden Erfolg der Optimierung mit *Hidden Markov Modellen* in der Spracherkennung (z.B. [Rab89, Hua90]) auch statistische Ansätze zur Erkennung und Lokalisierung von dreidimensionalen Objekten in der Bildverarbeitung [Hor94]. In der vorliegenden Arbeit werden probabilistische Verfahren wie beispielsweise das bekannte Simulated Annealing oder genetische Algorithmen zur iterativen Optimierung einer Interpretation eingesetzt.

Verarbeitungsgeschwindigkeit: Die Notwendigkeit zur Steigerung der Verarbeitungsgeschwindigkeit wird durch die Konzeption und Realisierung eines parallelen Kontrollalgorithmus berücksichtigt. Gegenstand der Parallelisierung sind zum einen die eingesetzten Optimierungstechniken, zum anderen werden Methoden zur massiv–parallelen Wissensnutzung in dem verwendeten semantischen Netzwerksystem entwickelt.

Any–Time–Fähigkeit: Die eingesetzten iterativen Lösungsverfahren liefern umso bessere Lösungen, je mehr Iterationen durchgeführt werden können, und garantieren das Auffinden einer optimalen Interpretation, falls genügend Zeit zur Verfügung steht. Die Parallelisierung wirkt hier unterstützend, da sowohl die "Taktfrequenz" für die Generierung von Interpretationen verringert wird, als auch die Möglichkeit zur Erzeugung mehrerer Lösungen pro Iterationsschritt genutzt wird.

Flexibilität und Erweiterbarkeit: Neben der von einer konkreten Anwendung unabhängigen Formulierung der Analyseaufgabe als Optimierungsproblem stellt die Nutzung der ERNEST–Systemschale die Flexibilität und Erweiterbarkeit sicher. Um einerseits die anschauliche und kompakte Darstellung sowie eine relativ einfache Wartung des problemabhängigen Wissens zu gewährleisten, andererseits jedoch dessen massiv–parallele Nutzung zu ermöglichen, werden insbesondere Methoden zur *automatischen Transformation* der begriffszentrierten Repräsentation in ein feingranulares Instantiierungsschema aus subkonzeptuellen Einheiten bereitgestellt.

Über die Lösung einer konkreten Aufgabe aus dem Bereich der symbolischen Bildverarbeitung — der Parallelisierung der wissensbasierten Analyse und einer Anwendung in der Bildfolgenanalyse — hinaus werden somit auch Aspekte künftiger Systeme zur Interpretation multi–modaler Sensordaten beachtet, die sich durch *Echtzeitverarbeitung*, *Robustheit* und die *flexible Interaktion* mit einer ungeregelten Umwelt auszeichnen.

Zum Abschluß dieses Kapitels gibt der folgende Abschnitt einen Überblick über den Aufbau der Arbeit.

1.6 Aufbau der Arbeit

In Abschnitt 1.4 wurde bereits dargelegt, daß die Parallelisierung von Algorithmen der Symbolik nach dem Prinzip der Aufgabenverteilung erfolgen muß. Die Identifizierung gleichzeitig ausführbarer Teilaufgaben wird dabei durch eine Trennung der Operationen im semantischen Netzwerk von der für die Musteranalyse benötigten übergeordneten Kontrollstrategie erleichtert. Unter Ausnutzung dieses Vorgehens ergibt sich für die restlichen Kapitel folgende Gliederung:

Kapitel 2 erläutert die von der ERNEST–Systemschale bereitgestellten Mittel zur Wissensrepäsentation sowie die Inferenzmechanismen und deren Zusammenwirken mit dem übergeordneten Kontrollalgorithmus.

Auf der Grundlage der dort vorgenommenen Beschreibung untersucht Kapitel 3 die Parallelisierung des ERNEST–Kontrollalgorithmus. Dazu wird zunächst die Beschränkung auf eine Teilmenge der Netzwerksprache motiviert, welche jedoch die epistemologischen Primitiva (Konzepte, Attribute, strukturelle Relationen und Kanten) umfaßt und somit

keinen Verlust an Ausdrucksmöglichkeiten bewirkt. Für die eingeschränkte Netzwerksprache wird eine formale Fassung erarbeitet, die sich an der Syntax der Konzeptdefinition orientiert und sowohl zur Abschätzung der Komplexität von Instantiierung und Graphsuche (Kapitel 3) als auch zur Bestimmung geeigneter Elementaroperationen für die parallele Instantiierung (Kapitel 4) genutzt wird. Abschließend erörtert Kapitel 3 die Parallelisierung der Zustandsraumsuche und beschreibt Ergebnisse der Realisierung in einem lokalen Netzwerk. Dabei wird die Notwendigkeit eines Kontrollalgorithmus deutlich, der auf einer Definition des Analysezustandes aufbaut, die eine effiziente Kommunikation zwischen den Prozessoren oder Rechnerknoten ermöglicht.

Kapitel 4 stellt den neu entwickelten parallelen Kontrollalgorithmus vor und beginnt mit einer detaillierten Beschreibung der automatischen Transformation der Wissensbasis in einen Datenflußgraphen für die parallele Instantiierung. Anschließend werden mehrere kombinatorische Optimierungsverfahren vorgestellt und ihr Einsatz bei der iterativen Lösung des Zuordnungsproblems zwischen den Segmentierungsergebnissen und den Modellen der Wissensbasis diskutiert. Die Entwicklung verschiedener Parallelisierungsstrategien beschließt das Kapitel.

Kapitel 5 beschreibt erste Experimente und Erfahrungen mit dem parallelen Kontrollalgorithmus. Nach einer kurzen Einführung in das gewählte Anwendungsszenario wird zuerst die Entscheidung für das feingranulare Instantiierungsschema aus Kapitel 4 validiert. Dabei wird auch deutlich, daß künftig zu entwickelnde Anwendungen die parallele Kontrolle durch eine höhere Modularität des prozeduralen Wissens unterstützen müssen. Abschließend werden die Ergebnisse für die Parallelisierung der entwickelten Optimierungsverfahren beschrieben. Kapitel 6 erörtert mögliche Erweiterungen und beschließt die Arbeit mit einer Zusammenfassung.

Kapitel 2

ERNEST: Eine Systemschale für die wissensbasierte Musteranalyse

Die automatische Interpretation komplexer Sensordaten, wie sie zum Beispiel Bilder und Bildfolgen natürlicher Szenen oder fließend gesprochene Sprache darstellen, hat die Generierung einer symbolischen Beschreibung zum Ziel. Die allgemein anerkannte Annahme, daß dazu Verfahren zur Akquisition, Repräsentation, und Nutzung von Wissen über den zu bearbeitenden Problemkreis benötigt werden, wird durch eine Vielzahl existierender Musteranalysesysteme unterstützt (vgl. die Literaturangaben auf Seite 10).

Grundlage und Untersuchungsgegenstand der vorliegenden Arbeit ist ERNEST, ein System für die wissensbasierte Musteranalyse, das auf dem Formalismus der semantischen Netzwerke aufbaut. Nach einer kurzen Übersicht befaßt sich dieses Kapitel mit den von der Systemschale zur Verfügung gestellten Mitteln zur Wissensrepräsentation und –nutzung.

2.1 Allgemeiner Aufbau

Das ERlanger semantische NEtzwerkSysTem ERNEST [Nie90d, Kum93b] ist eine Systemschale für die Analyse komplexer Sensordaten, die am Lehrstuhl für Mustererkennung (Informatik 5) der Friedrich-Alexander-Universität Erlangen-Nürnberg entwickelt wurde. Das System wird sowohl im Bereich des Verstehens fließend gesprochener Sprache [Eck92, Nie92, Mas94] als auch in unterschiedlichen Anwendungen der medizinischen Diagnostik [Nie85b, Sag88, Sag90b, Wei93] und der industriellen Bildanalyse [Nie90b, Nie90c] eingesetzt.

Der Aufbau des Systems folgt der allgemeinen Struktur eines wissensbasierten Musteranalysesystems nach [Nie90a] und ist in Bild 2.1 dargestellt. Die Basis des Systems bilden die vier Module *Methoden, Wissen, Kontrolle* und *Ergebnisse* [Nie81, Sag90a, Kum92a]. Ergänzenden Charakter besitzen die Komponenten zur automatischen Wissens*akquisition*

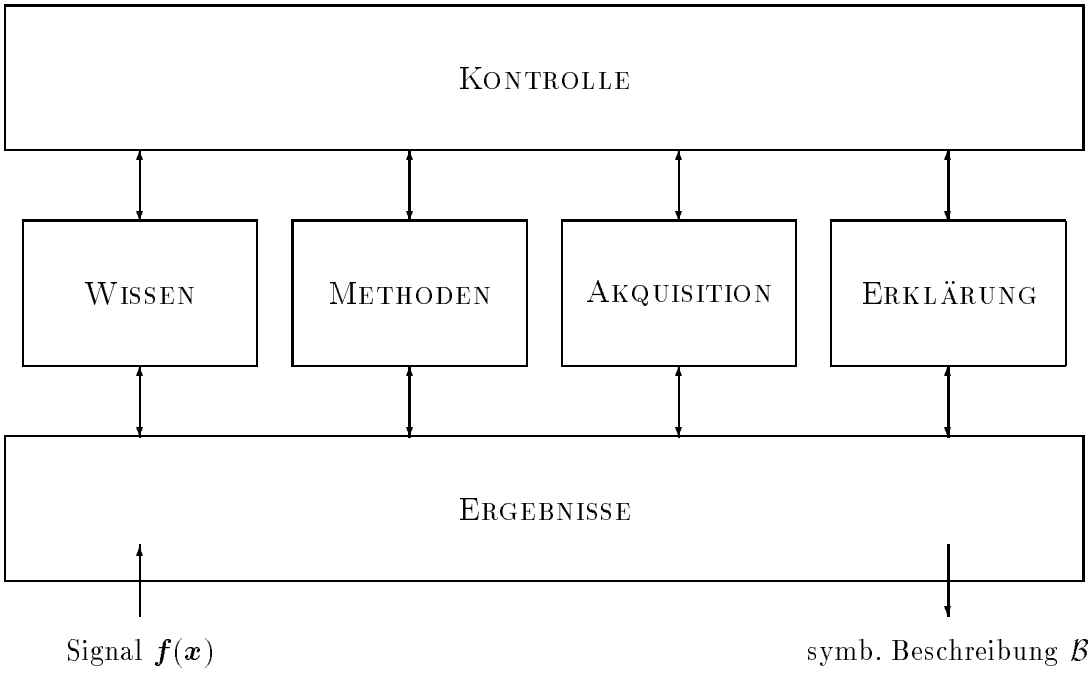


Bild 2.1: Funktionale Organisation eines Musteranalysesystems nach [Nie90a].

[Sch90b] und zur *Erklärung* von Analyseergebnissen [Pre91, Pre92].

Das Modul *Methoden* umfaßt die Algorithmen der Ikonik und Ikonik–Symbolik, deren Aufgabe in der Erstellung einer initialen symbolischen Beschreibung besteht, und die im allgemeinen problemunabhängig sind und ohne die explizite Repräsentation von Wissen auskommen. Ausnahmen werden beispielsweise in [Naz84, Sak85, Mat90b] beschrieben, wo Produktionensysteme zur Steuerung der Segmentierung (Auswahl und Anwendung ikonischer Operatoren) eingesetzt werden.

Im Modul *Wissen* ist aufgabenspezifisches Wissen abgelegt, das zur Interpretation des Signals bzw. der Segmentierungsergebnisse benötigt wird. Dazu gehört sowohl deklaratives Wissen zur Beschreibung der relevanten Aspekte eines Problemkreises als auch prozedurales Wissen in Form von problemabhängigen Algorithmen zur Nutzung der dargestellten Begriffe während der Analyse.

Die *Kontrolle* steuert die Analyse, indem sie Strategien zur Fokussierung auf relevante Daten und Zwischenergebnisse bereitstellt und für die Aktivierung der Methoden sorgt. Das Modul *Ergebnisse* nimmt (Zwischen-)Ergebnisse auf und ermöglicht den verschiedenen Komponenten des Systems den Zugriff darauf.

Die in der vorliegenden Arbeit entwickelten Verfahren basieren auf einer Darstellung

problemabhängigen Wissens in dem im Modul *Wissen* verwendeten Formalismus und den in der *Kontrolle* definierten Mechanismen zu dessen Nutzung während der Analyse. Daher werden beide Systemkomponenten im folgenden ausführlich vorgestellt.

2.2 Die Wissensbasis

Wie bereits erwähnt, werden Wissensbasen in der ERNEST–Systemschale als semantische Netze organisiert, deren Knoten Objekte, Sachverhalte oder Ereignisse repräsentieren und deren Kanten Beziehungen zwischen diesen darstellen; Kanten und Knoten sind mit einer *Rolle*, die ihre jeweilige Bedeutung angibt, markiert. Die ERNEST–Netzwerksprache unterscheidet drei Knotentypen und sechs verschiedene Kanten. Einziger Knotentyp der Wissensbasis ist das *Konzept*. Knoten des Typs *Instanz* und *modifiziertes Konzept* werden während der Analyse erzeugt und im Ergebnisspeicher abgelegt, der ebenfalls als semantisches Netz organisiert ist, wodurch z.B. eine Erklärung der Systemaktivitäten erleichtert wird. Alle Knoten werden durch eine komplexe, framebasierte Datenstruktur realisiert; die Knotentypen unterscheiden sich lediglich dadurch voneinander, daß die in Konzepten vorhandenen Verweise auf Analyseprozeduren zur Instantiierung (s.u.) durch berechnete Werte oder Restriktionen in Instanzen bzw. modifizierten Konzepten ersetzt werden.

Im Gegensatz zu *Konzepten*, welche die intensionalen Beschreibungen beliebiger Begriffe aufnehmen, stellen *Instanzen* extensionale Beschreibungen dar, d.h. sie repräsentieren Ausprägungen von Konzepten in den vorliegenden Sensordaten. Die Einführung *modifizierter Konzepte* ist durch die Notwendigkeit der Verarbeitung unsicherer Daten (Hypothesenmengen) in einem Musteranalysesystem begründet. Sie bieten die Möglichkeit, aufgrund bereits berechneter Zwischenergebnisse oder durch in der Wissensbasis abgelegte Vorerwartungen in einer konkreten Analysesituation Restriktionen für Konzepte zu propagieren. Durch die so vorgenommene Adaption der Wissensbasis an die aktuellen Sensordaten kann die Effizienz der Analyse wesentlich gesteigert werden; ein Beispiel diskutieren wir im Rahmen der Überlegungen zur Parallelisierung in Kapitel 3 auf Seite 69.

Die in der Netzwerksprache definierten Kantentypen sind die *Konkretisierungs–*, *Bestandteils–*, *Referenz–* und *Spezialisierungskante* in der Wissensbasis, sowie die *Instanz–* und *Modellkante*. Von diesen modellieren die drei erstgenannten komplexe Beziehungen zwischen Konzepten, die in einer eigenen Datenstruktur, der KANTENBESCHREIBUNG, näher spezifiziert werden.

Die Interpretation komplexer Muster erfordert die Darstellung und Nutzung von Wissen aus unterschiedlichen begrifflichen Welten, welche im allgemeinen in einem hierarchischen Modell angeordnet werden; Vorschläge zu einer geschichteten Organisation bild– und sprachverstehender Systeme finden sich u.a. in [Nie87]. Die *Konkretisierungskante* verbindet Konzepte unterschiedlicher Abstraktionsebenen und erlaubt somit die homogene

Repräsentation sämtlicher begrifflicher Systeme, die bei der Analyse zu durchlaufen sind, in einer Wissensbasis. Dabei ist das Signal als konkreteste Abstraktionsebene vereinbart und die abstrakteste wird durch das Analyseziel bestimmt. Zwischen diesen beiden Ebenen können — entsprechend den Erfordernissen des jeweiligen Problemkreises — beliebig viele weitere Abstraktionsebenen definiert werden. Als exemplarisch für die wissensbasierte Bildanalyse kann die Einteilung in die zunehmend abstrakteren konzeptuellen Systeme "geometrische Objekte — Objekte des Problemkreises — Schlußfolgerungen" angesehen werden, als Konkretisierung des Konzepts "Motorteil" aus der Begriffswelt der industriellen Fertigung könnte beispielsweise das Konzept "Liniensegment" als seine Manifestation auf der Ebene geometrischer Objekte angegeben werden (vgl. Bild 2.2).

Während die Konkretisierungskante eine Besonderheit der ERNEST–Netzwerksprache darstellt, ist die *Bestandteilskante* unter verschiedenen Bezeichnungen (*"has–a", "part–of"*) in vielen Systemen etabliert; sie realisiert die Dekomposition komplexer Konzepte in einfachere Teile. Ein Unterschied zu anderen Netzwerkformalismen besteht darin, daß Bestandteile als *kontextabhängig* markiert werden können. Dadurch kann berücksichtigt werden, daß die Interpretation bestimmter Konzepte nur in Kenntnis eines übergeordneten Begriffes, des *Kontexts*, möglich ist. Ein Beispiel für kontextabhängige Modellierungen findet sich im System zur automatischen Diagnose von Sequenzszintigrammen des menschlichen Herzens [Sag85, Sag90b], wo der — für die Diagnose wichtige — linke Ventrikel nur im Kontext des Herzens zu erkennen ist ("Herz" $\xrightarrow{kbst}$ "Li-Ventrikel"). Auch in der linguistischen Wissensbasis des Spracherkennungs– und Dialogsystems EVAR [Kum92a, Mas94], dessen Ziel das Führen eines spontansprachlichen Auskunftsdialogs über den Intercity–Fahrplan der Deutschen Bundesbahn ist, werden kontextabhängige Bestandteile intensiv genutzt. Das semantische Wissen des Systems baut auf der Valenztheorie [Tes66] und der Fillmore'schen Kasustheorie [Fil68] auf, nach der die Bildung korrekter Sätze die Belegung bestimmter Leerstellen (Tiefenkasus) erfordert, die durch ein Verb eröffnet werden. Tiefenkasus werden als Bestandteile der sogenannten Verbrahmen modelliert. Die Kontextabhängigkeit ergibt sich nun aus der Tatsache, daß die Interpretation dieser Bestandteile nur in Kenntnis des Verbrahmens möglich ist: so besitzt die Ortsangabe *" in Nürnberg"* im Verbrahmen *"ankommen"* die Bedeutung eines Ankunftsortes, während sie im Kontext von *"abfahren"* als Abfahrtsort zu interpretieren ist.

Zur Modellierung referentieller Bezüge zwischen Konzepten wird in [Kum92a] die *Referenzkante* eingeführt. Eine Referenz liegt dann vor, wenn ein Konzept zwar im Signal detektierbar ist, aber nicht eigenständig interpretiert werden kann. Ein Beispiel aus dem Bereich des automatischen Sprachverstehens gibt [Kum92a, S. 46]: im Satz *"Der Zug, der um fünfzehn Uhr in Nürnberg ankommt"* ist das Pronomen (*"..., der ..."*) zwar im Signal zu finden, kann jedoch nicht ohne die Nominalgruppe, auf die es sich bezieht (*"Der Zug"*), interpretiert werden.

Die — ebenfalls in anderen Netzwerksystemen anzutreffende — *Spezialisierungskante*

zeigt von einem allgemeinen zu einem spezielleren Konzept und ermöglicht den Aufbau von Vererbungshierarchien. Dabei werden sämtliche Kanten–, Attribut– und Relationsbeschreibungen (s.u.) entlang dieser Kante vererbt, sofern dies nicht explizit unterbunden wird.

Instanz– und *Modellkante* sind nicht Teil der Wissensbasis, sondern werden während der Analyse bzw. der automatischen Wissensakquisition erzeugt. Die Instanzkante verbindet ein Konzept mit seinen Instanzen im Sinne einer Klassifikation, und stellt damit die Verbindung von Wissensbasis und Ergebnisspeicher bzw. Modell und Sensordaten her. Die Modellkante besitzt eine ähnliche Bedeutung in der Wissensakquisition. Sie verbindet die Konzepte eines Modells mit denen eines allgemeineren Modellschemas, aus dem sie per Akquisition automatisch erzeugt wurden.

Um die Entwicklung problemunabhängiger Kontrollalgorithmen zu ermöglichen, werden die möglichen Spezialisierungs–, Bestandteils– und Konkretisierungsbeziehungen auf der Menge $\mathcal{K} = \{A, B, C \ldots\}$ der Konzepte eines Netzwerkes eingeschränkt. Dazu wird zunächst jedem Konzept ein Tupel $\delta = (\delta_{spez}, \delta_{kon}, \delta_{bst}, \delta_{mod})$ zugeordnet, das die Lage des Konzeptes bezüglich der vier Kantentypen Spezialisierung, Konkretisierung, Bestandteil und Modell beschreibt und im Eintrag Grade abgelegt wird. Der *Modellgrad* δ_{mod} ist nur für die Belange der automatischen Wissensakquisition relevant (siehe [Sch90b]) und wird hier nicht weiter betrachtet. Die Gleichungen (2.1 – 2.3) definieren den *Spezialisierungsgrad* δ_{spez}, den *Konkretisierungsgrad* δ_{kon} und den *Bestandteilsgrad* δ_{bst} eines Konzeptes über die Länge eines Kantenzugs des entsprechenden Typs zu bestimmten Konzepten.

$$\begin{aligned} \delta_{spez}(A) &= \begin{cases} 0 & \text{falls } \mathbf{SPEZV} = \emptyset \\ \max\limits_{B \in \mathbf{SPEZV}} \{\delta_{spez}(B)\} + 1 & \text{sonst} \end{cases} \\ \mathbf{SPEZV} &= \{B \mid B \xrightarrow{spez} A\} \end{aligned} \tag{2.1}$$

$$\begin{aligned} \delta_{kon}(A) &= \begin{cases} 0 & \text{falls } \mathbf{KON} = \emptyset \\ \max\limits_{B \in \mathbf{KON}} \{\delta_{kon}(B)\} + 1 & \text{sonst} \end{cases} \\ \mathbf{KON} &= \{B \mid A \xrightarrow{kon} B\} \cup \{B \mid C \xrightarrow{kon} B \wedge (C \xrightarrow{spez} A \vee C \xrightarrow{spez} \ldots \xrightarrow{spez} A)\} \end{aligned} \tag{2.2}$$

$$\begin{aligned} \delta_{bst}(A) &= \begin{cases} 0 & \text{falls } \mathbf{BST} = \emptyset \\ \max\limits_{B \in \mathbf{BST}} \{\delta_{bst}(B)\} + 1 & \text{sonst} \end{cases} \\ \mathbf{BST} &= \{B \mid A \xrightarrow{bst} B\} \cup \{B \mid C \xrightarrow{bst} B \wedge (C \xrightarrow{spez} A \vee C \xrightarrow{spez} \ldots \xrightarrow{spez} A)\} \end{aligned} \tag{2.3}$$

Der Spezialisierungsgrad mißt den Abstand eines Konzeptes zu einem (dem) allge-

mein(st)en Konzept; Konkretisierungs– und Bestandteilsgrad berechnen die Pfadlänge zu einem *minimalen Konzept*, für das $\delta_{kon} = \delta_{bst} = 0$ gilt. Minimale Konzepte besitzen für den Analyseprozeß eine besondere Bedeutung, da sie im allgemeinen die Instantiierung initialisieren. Konzepte, die nicht minimal sind, aber ebenfalls direkt auf die Ergebnisse der Segmentierung zugreifen, werden auch als *Schnittstellenkonzepte* bezeichnet.

Mit Hilfe der Graddefinitionen werden die erlaubten Beziehungen zwischen den Konzepten durch die Bedingungen

$$A \xrightarrow{bst} B \quad \Rightarrow \quad \delta_{spez}(A) \geq \delta_{spez}(B) \wedge \neg(B \xrightarrow{spez} A) \tag{2.4}$$

$$A \xrightarrow{spez} B \quad \Rightarrow \quad \delta_{kon}(A) = \delta_{kon}(B) \tag{2.5}$$

$$A \xrightarrow{bst} B \quad \Rightarrow \quad \delta_{kon}(A) = \delta_{kon}(B) \tag{2.6}$$

eingeschränkt. Bedingung (2.4) fordert, daß Bestandteile höchstens so speziell wie ihre gemeinsamen Komposita sein können. Die Bedingungen (2.5 und 2.6) beschränken Spezialisierungs– und Bestandteilsbeziehungen auf Konzepte des gleichen Konkretisierungsgrades, wodurch sich eine klare Gliederung einer Wissensbasis in unterschiedliche Abstraktionsebenen ergibt. Somit garantieren die Restriktionen nicht nur die Zyklenfreiheit des Netzwerkes, sondern dienen auch zur Unterstützung der inhaltlichen Konsistenz und Widerspruchsfreiheit des dargestellten Wissens. Darüber hinaus lassen die definierten Grade die Berechnung von *Prioritäten* zu, welche allgemeine, problemunabhängige Kriterien für die Verarbeitungsreihenfolge der Konzepte während der Analyse ausdrücken (vgl. [Sag90a, S. 246]). Im Rahmen dieser Arbeit werden die Grade in Kapitel 3 dazu genutzt, den Instantiierungsaufwand eines Konzeptes abzuschätzen.

In den meisten Anwendungsgebieten des Systems zeichnen sich die zu repräsentierenden Begriffe durch eine gewisse Variabilität aus. Beispielsweise können in einer Straßenverkehrsszene eine variable Anzahl von Fahrzeugen auftreten, von einem Fahrzeug können verschiedene Teile aufgrund von Verdeckungen oder aufgrund des Blickwinkels nicht sichtbar sein, oder die Objekte unterscheiden sich geringfügig durch Bestandteile bzw. Merkmale, die für ihre Erkennung und Interpretation nicht unbedingt erforderlich sind (z.B. Antenne oder Spoiler eines Autos).

Wird für jede Variante eines Begriffes ein eigenes Konzept angelegt, so werden die entstehenden Wissensbasen sehr groß, wodurch die Übersichtlichkeit der Darstellung abnimmt. Gleichzeitig wächst dadurch die Möglichkeit von Fehlern oder Inkonsistenzen und die Wartung und Erweiterung des Wissens wird erschwert. Aus diesen Gründen wurde bei der Entwicklung des Systems ein Schwerpunkt auf die Integration von Elementen zur objektzentrierten Realisierung großer Wissensbasen gelegt. Um die Anzahl der Kanten und

Knoten möglichst gering zu halten, erlaubt die Netzwerksprache sowohl die kompakte Beschreibung komplexer Relationen zwischen Konzepten als auch die Definition verschiedener intensionaler Ausprägungen eines Begriffes in einem Konzept. Eine diesbezüglich weitgehend flexible Modellierung wird durch die Bereitstellung verschiedener Sprachkonstrukte gewährleistet:

- Die Kantenbeschreibung erlaubt die Angabe einer minimalen und maximalen Anzahl für jedes Bestandteil bzw. jede Konkretisierung eines Konzepts, wodurch deren mehrfaches Vorkommen mit nur einer Kantenbeschreibung zu modellieren ist. Typische Anwendungsbeispiele sind hier das Auftreten mehrerer gleicher Objekte in einer Szene, die Zerlegung von Polygonen in Liniensegmente oder die nähere Beschreibung eines Nomens durch mehrere Adjektive.

- Besitzen mehrere Konkretisierungen oder Bestandteile die gleiche funktionale Rolle, so werden diese in einer Kantenbeschreibung zusammengefaßt und dort als *alternative Zielknoten* vermerkt. Zur Illustration verweisen wir auf Bild 3.11, Seite 68: Als Konkretisierungen des Konzepts "Dach" kommen auf der Ebene geometrischer Objekte für verschiedene Ansichten die Konzepte "Dreieck" oder "Rechteck" infrage. Da beide in der gleichen Relation zu "Dach" stehen, werden sie gemeinsam in *einer* Kantenbeschreibung modelliert; in einer bestimmten Analysesituation kann jedoch nur genau eine der so dargestellten Beziehungen gültig sein.

- Die in der Konzeptdatenstruktur verankerten MODALITÄTSBESCHREIBUNGEN sind das wichtigste Strukturierungsmittel der Wissensbasis. Sie gestatten die Modellierung von unterschiedlichen intensionalen Ausprägungen eines Begriffes durch die Angabe von *Modalitätsmengen*, in denen Bestandteile und Konktretisierungen eines Konzeptes zu zulässigen Begriffsvarianten gruppiert werden können. Innerhalb einer Modalitätsmenge können Kanten als *obligatorisch*, *optional* oder *inhärent* gekennzeichnet werden.

  Während obligatorische Bestandteile und Konkretisierungen für die Instantiierung eines Konzeptes unbedingt erforderlich sind, können optionale Teile vorhanden sein oder auch nicht. Als inhärent werden solche Teile markiert, die zur Instantiierung eines Konzeptes erforderlich sind, aber nicht unbedingt eine Ausprägung in den Sensordaten besitzen müssen. Sie müssen während der Analyse aus anderen Instanzen erschlossen oder durch sinnvolle Vorerwartungen (Defaultwerte) belegt werden; [Sag90a] diskutiert Beispiele aus der Wissensbasis des Spracherkennungssystems EVAR.

Die Modalitätsbeschreibung wird durch eine Substruktur zur Darstellung struktureller Beziehungen zwischen den Teilen eines Konzeptes, der ADJAZENZ, komplettiert. Sie erlaubt die effiziente Repräsentation von zeitlichen und/oder räumlichen Beziehungen zwischen

den Elementen (Kanten) einer Modalitätsmenge in Form einer Bitmatrix. Der Kohärent–Eintrag der Adjazenz ermöglicht es, die Forderung einer kompakten Überdeckung eines Signalabschnitts durch ein Konzept auszudrücken. Für die ausführliche Diskussion von Beispielen aus der Bild– und Sprachanalyse sei auf [Sch90b, Sag90a] verwiesen.

Während Kanten die Beziehungen zwischen Konzepten bzw. Konzepten und Instanzen definieren, dienen *Attribute* und *Relationen* zur näheren Charakterisierung der darzustellenden Begriffe. Dabei geben Attribute die meßbaren Eigenschaften eines Konzeptes an, und können sowohl numerische (z.B. die Höhe eines Objektes in Metern) als auch symbolische Werte (z.B. den Kasus eines Wortes) annehmen. Attribute können für ein Konzept als *lokal* vereinbart werden, wodurch sie nicht der Vererbung entlang der Spezialisierungskanten unterliegen. Aus ergonomischen Gründen werden Attribute, die zur intensionalen Beschreibung eines Begriffes beitragen, von solchen getrennt, die lediglich für die Musteranalyse relevant sind und als *Analyseparameter* bezeichnet werden.

Eine analoge Unterscheidung wird für Relationen getroffen. *Strukturrelationen* definieren Beziehungen zwischen Attributen, *Analyserelationen* solche zwischen Analyseparametern. Da Attribute und Analyseparameter sowie Stuktur– und Analyserelationen nur bei der Erstellung einer Wissensbasis unterschieden, ansonsten aber vollkommen einheitlich behandelt werden, sprechen wir im folgenden nur noch von *Attributen* und *Relationen*; auch sie werden in eigenen Datenstrukturen (ATTRIBUT– und RELATIONSBESCHREIBUNG) genauer definiert.

Bild 2.2 illustriert die bisher eingeführten Mittel zur Repräsentation *deklarativen* Wissens, die eine vollständige Beschreibung der Begriffe eines Problemkreises und ihrer Beziehungen untereinander erlauben.

Die Nutzung des Wissens bei der Interpretation von Sensordaten erfordert zusätzlich zu den deklarativen Elementen die Integration unterschiedlicher Arten *prozeduralen* Wissens. Zunächst wird ein Inferenzmechanismus benötigt, der angibt, wie modifizierte Konzepte und Instanzen zu einem Konzept gebildet werden. Der Einsatz des Systems in verschiedenen Anwendungsbereichen der wissensbasierten Signalinterpretation motiviert hierbei die Einführung einer problemunabhängigen prozeduralen Semantik, die nur von der Syntax des Netzwerkes, nicht aber von den dargestellten Begriffen abhängig ist. Sie wird zusammen mit dem übergeordneten Kontrollalgorithmus des Systems in Abschnitt 2.3 vorgestellt.

Zur Erzeugung von Instanzen und modifizierten Konzepten müssen Attribut– und Relationswerte aus den Sensordaten extrahiert und mit der Aussage eines Konzeptes verglichen werden. Dies erfordert die Anbindung problemabhängiger Routinen an mehrere Einträge eines Konzeptes bzw. seiner Substrukturen und wird von der Netzwerksprache durch die Bereitstellung einer einheitlichen Prozedurschnittstelle, der FUNKTIONSBESCHREIBUNG, unterstützt. Sie ermöglicht die Angabe folgender Analyseprozeduren:

- Die Bewertung eines Konzeptes gestattet es, die Übereinstimmung von Instanzen und

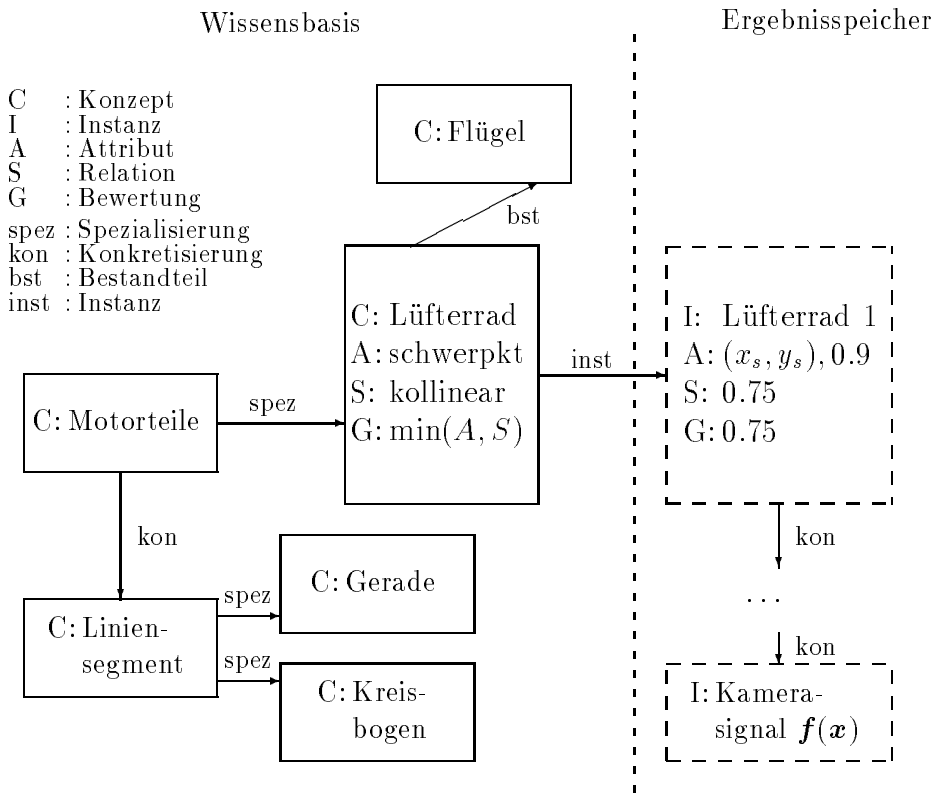


Bild 2.2: Wissensrepräsentation mit ERNEST in einem Beispiel aus der Analyse industrieller Szenen (nach [Nie90b]).

modifizierten Konzepten, die aus den Sensordaten berechnet wurden, mit den in der Wissensbasis abgelegten Begriffen zu ermitteln. Dies ist erforderlich, da die Ergebnisse der initialen Segmentierung aufgrund verrauschter Sensordaten im allgemeinen fehlerbehaftet oder mehrdeutig sind, und somit unsichere Daten verarbeitet werden müssen. Bewertungen sind in einem beliebigen Kalkül formulierbar (z.B. Wahrscheinlichkeiten, Fuzzy–Bewertungen) und können beispielsweise die Qualität, die Sicherheit und die Priorität einer Hypothese bzw. einer Teilinterpretation ausdrücken.

- In der Kantenbeschreibung sind Funktionen zur Kantenbewertung, zur inversen Kantenbewertung und zur Referenzauswahl verankert.

  Die Kantenbewertung ermöglicht die Beurteilung der Instanzen von Bestandteilen und Konkretisierungen eines Konzeptes bezüglich des Begriffes, der durch das übergeordnete Konzept repräsentiert wird. Umgekehrt können Werte und Bewertungen eines Konzeptes mit Hilfe der inversen Kantenbewertung an die Subkonzepte

übergeben werden, wodurch die Behandlung eines Bestandteiles oder einer Konkretisierung unter Berücksichtigung des übergeordneten Begriffes ermöglicht wird.

Die Angabe einer Funktion zur Referenzauswahl ist nur für Referenzkanten möglich bzw. erforderlich. Sie wird zur sinnvollen Auswahl von Instanzen der Konzepte eingesetzt, die zur vollständigen Interpretation des übergeordneten Konzeptes benötigt werden (vgl. auch [Kum92a, S. 52]).

- In der Attributbeschreibung werden Prozeduren zur Attributwerteberechnung und zur Attributbewertung angegeben. Während der Instantiierung werden mit der Werteberechnung die Werte von Attributen aus den Sensordaten gemessen oder berechnet. Wird ein modifiziertes Konzept datengetrieben generiert (siehe Abschnitt 2.3), so liefert die Werteberechnung eine Einschränkung (*Restriktion*) für den Wertebereich des Attributes, falls noch nicht alle Argumente der Funktion bekannt sind. Die Angabe einer inversen Attributwerteberechnung kann bei der modellgetriebenen Erzeugung modifizierter Konzepte zur Propagierung von Einschränkungen für noch nicht berechnete Argumente benutzt werden.

  Die Attributbewertung erlaubt es, die berechneten Werte oder Restriktionen mit den in der Wissensbasis abgelegten Vorerwartungen bzw. Beschreibungen zu vergleichen, und die Übereinstimmung in einem beliebigen Bewertungskalkül auszudrücken.

- Die Relationsbewertung in der Relationsbeschreibung dient zur Überprüfung der dargestellten Beziehung zwischen Attributen und liefert ein Maß für deren Erfülltheit. Die inverse Relationsbewertung dient in Analogie zur inversen Attributwerteberechnung der Einschränkung noch nicht berechneter Argumente.

Die Argumente der Analyseprozeduren werden in Abhängigkeit von der jeweiligen Substruktur auf eine epistemologisch sinnvolle Menge von Netzwerkelementen eingeschränkt (vgl. Bild 4.4, S. 89). Neben den genannten Prozeduren werden auch solche für die Wissensakquisition und zur Erklärung an die verschiedenen Netzwerkstrukturen gebunden. Sie werden in [Sch90b, Pre91, Pre92] ausführlich erläutert.

Die Forderung nach einem problemunabhängigen Kontrollalgorithmus steht zunächst im Widerspruch zu einer möglichst hohen Effizienz der Analyse. Neben den bereits erwähnten Prioritäten und der Möglichkeit, die Bearbeitungsreihenfolge durch die Belegung des Eintrags Vorrang einer Kantenbeschreibung zu bestimmen, stellt die Netzwerksprache daher eine weitere Art prozeduralen Wissens, die Elemente zur *lokalen Steuerung* der Analyse zur Verfügung. Um auch hier eine durchgängige objektzentrierte Modellierung zu unterstützen, wird die Substruktur IDENTIFIKATION in der Konzeptdatenstruktur verankert.

Identifikationen ermöglichen die Formulierung von problemabhängigen Einschränkungen der Instantiierungsmöglichkeiten eines Konzeptes, und werden hier in Anlehnung an [Nie90d, Sag90a] zunächst an Beispielen erläutert. Der Einfluß auf die Komplexität der

Instantiierung wird in Kapitel 3 untersucht. Bild 2.3 zeigt ein Beispielnetzwerk mit fünf

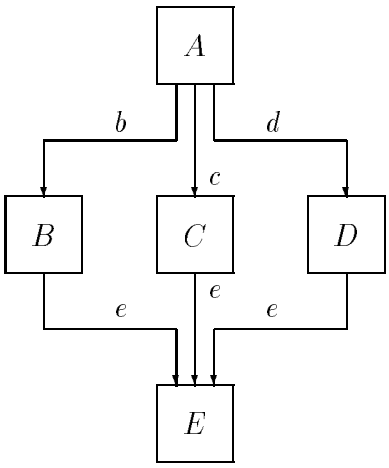


Bild 2.3: Ein Beispielnetzwerk zur Erläuterung der problemabhängigen Instanzbindung.

Konzepten, für das in unterschiedlichen Anwendungen folgende Interpretationen möglich sind:

- In der Wissensbasis für die linguistischen Analyse im Spracherkennungssystem EVAR [Nie92, Kum92a] kann die dargestellte Situation beispielsweise auf der Ebene der syntaktischen Analyse gefunden werden. Hier könnte A eine Nominalgruppe bezeichnen, welche die Bestandteile Artikel (Konzept B), Adjektiv (C) und Nomen (D) besitzt, die durch je eine Worthypothese (Konzept E) instantiiert werden. Der resultierende *Instantiierungspfad*, dessen Berechnung in Abschnitt 2.3 erläutert wird, besitzt demnach drei Instanzen zum Konzept E:

$$\mathcal{I} \;=\; I_1(A)\{I_1(B)[I_1(E)], I_1(C)[I_2(E)], I_1(D)[I_3(E)]\}$$

- Im System zur Analyse von Sequenzszintigrammen des menschlichen Herzens [Sag85] könnte das Konzept A einen Herzzyklus bezeichnen, der aus den drei Phasen Kontraktion, Stagnation und Expansion (Konzepte B, C und D) besteht. Für deren Instantiierung wird die jeweils gleiche Instanz des Herzens (Konzept E) benötigt, wodurch sich folgender Instantiierungspfad ergibt:

$$\mathcal{I} \;=\; I_1(A)\{I_1(B)[I_1(E)], I_1(C)[I_1(E)], I_1(D)[I_1(E)]\}$$

- Bei der automatischen Erkennung und Lokalisierung von Werkstücken in einer industriellen Umgebung [Sal92] könnte A auf der Ebene der geometrischen Objekte ein

Dreieck bezeichnen, zu dessen Instantiierung drei Seiten (B,C und D) benötigt werden. Jede der Seiten ist durch zwei Ecken E bestimmt. Dabei haben je zwei Seiten eine gemeinsame Ecke, müssen also mit der gleichen Instanz gebildet werden:

$$\mathcal{I} \;=\; I_1(A)\{I_1(B)[I_1(E), I_2(E)], I_1(C)[I_1(E), I_3(E)], I_1(D)[I_2(E), I_3(E)]\}$$

Die drei Fälle können durch die Belegung der IDENTIFIKATION explizit ausgedrückt werden. Die im ersten Beispiel benötigten drei verschiedenen Instanzen von E (einer Worthypothese) werden dann erzeugt, wenn die Substruktur im Konzept A nicht belegt ist. Der zweite Fall, die einmalige Instantiierung von E (des Herzens) auf einem Suchbaumpfad, kann durch die Angabe der Konstante `UNIQUE` in der Identifikationsbeschreibung des Konzepts E erzwungen werden. Im dritten Beispiel werden im Konzept A die Pfade (Kantenzüge) $b.e$ und $c.e$ miteinander identifiziert. Dadurch wird erzwungen, daß eine Instanz von E, die ein Bestandteil von B ist, auch zur Instantiierung von C verwendet wird. Desweiteren werden die Pfade $b.e$ und $d.e$ sowie $c.e$ und $d.e$ angegeben, wodurch die beschriebene Forderung nach einer gemeinsamen Ecke zweier Seiten modelliert ist.

2.3 Kontrolle der Analyse

Zu Beginn dieses Kapitels haben wir die automatische Generierung einer optimalen symbolischen Beschreibung (unter Verwendung der Begriffe eines gegebenen Problemkreises) als Ziel der Analyse komplexer Muster definiert. Die Interpretation soll optimal zu den von der Vorverarbeitung gelieferten Daten passen, kompatibel zur Wissensbasis sein, und die Information enthalten, die eine sinnvolle Systemreaktion ermöglicht. Im Sprachgebrauch der ERNEST–Systemschale werden Konzepte, die — im Sinne des jeweiligen Benutzers — hinreichend relevante Aussagen über den Applikationsbereich repräsentieren als *Zielkonzepte* (oder auch: Analyseziele) bezeichnet; ihre Instanzen beinhalten die geforderte symbolische Beschreibung des Signals, und es ist Aufgabe des Kontrollalgorithmus, ihre Instantiierung zu steuern.

In [Sch86, Sag90a] werden die wesentlichen, zum Teil konträren Anforderungen und Ziele beim Entwurf eines Wissensrepräsentationssystems diskutiert. Hervorzuheben sind die Problemunabhängigkeit des Inferenzmechanismus und der Kontrollstrategie sowie die Möglichkeit zur Bewertung von Zuständen des Analyseprozesses. Während die erste Forderung die Repräsentation und Nutzung von Wissen in unterschiedlichen Anwendungsbereichen ermöglicht, zielt die zweite auf die Entwicklung von Systemen zur wissensbasierten Signalinterpretation, die beim heutigen Stand der Technik auf allen Verarbeitungsebenen (ikonisch, symbolisch) mit unsicheren und fehlerbehafteten Daten umgehen müssen.

Im ERNEST–Kontrollalgorithmus münden diese Überlegungen in einer aus sechs Inferenzregeln bestehenden prozeduralen Semantik und der Verwendung eines heuristischen

Graphsuchverfahrens, des A^*–Algorithmus, zur Steuerung der Analyse. Aufgrund der weiten Verbreitung des Verfahrens in der Literatur (vergleiche z.B. [Ric83, Win84, Nie87] und insbesondere [Nil82] für einen Beweis der Zulässigkeit) unterziehen wir zunächst nur die Inferenzregeln einer näheren Betrachtung. Eine Diskussion der Komplexität des Algorithmus und weiterer, für die Parallelisierung relevanter Eigenschaften führen wir in Kapitel 3, um die Entwicklung einer neuen Kontrollstrategie zu motivieren.

Grundlage des Instantiierungsprozesses sind zwei Inferenzregeln zur datengetriebenen Erzeugung von Instanzen, die wir nach [Sag90a] zitieren. Eine ausführliche Darstellung der zur Behandlung referentieller Bezüge und inhärenter Kanten notwendigen Erweiterungen findet sich in [Kum92a]; sie ist im Rahmen dieser Arbeit jedoch nicht erforderlich, da wir uns bei der Entwicklung eines parallelen Kontrollalgorithmus auf eine Teilmenge der Netzwerksprache konzentrieren werden (siehe Kapitel 3.1).

Regel 1: *Generierung partieller Instanzen*
WENN für ein Konzept A oder ein modifiziertes Konzept $Q_i(A)$
und eine Modalitätsbeschreibung von A
Instanzen existieren für
- alle als obligatorisch bezeichnete Konkretisierungen und
- alle als obligatorisch bezeichnete Bestandteile,
  die nicht kontextabhängig von A sind

und eine partielle Instanz eines Konzepts existiert,
das einen Kontext für A definiert
DANN generiere partielle Instanzen $I_j^p(A)$ wie folgt:
- generiere für $I^p(A)$ eine leere Instanz,
- trage die verwendete Modalitätsbeschreibung ein,
- verbinde $I^p(A)$ mit den Instanzen, die in der Prämisse genannt sind,
- übertrage die restlichen Kanten von A bzw. $I_j(A)$ nach $I^p(A)$,
- aktiviere die an A gebundenen Funktionen für $I^p(A)$ in der Reihenfolge
  - Kantenbewertungen
  - Attributwerteberechnungen
  - Attributbewertungen
  - Relationsbewertung
  - Bewertung

(2.7)

Regel 2: *Generierung von Instanzen*
WENN für eine partielle Instanz $I_i^p(A)$
Instanzen existieren für alle als obligatorisch bezeichneten Bestandteile
DANN generiere Instanzen $I_j(A)$ aus $I_i^p(A)$ wie folgt:
- erzeuge eine leere Instanz für ein $I(A)$
- übernehme die Kantenbeschreibungen für Konkretisierungen aus $I_i^p(A)$
- verbinde die Instanzen mit den in der Prämisse referierten Instanzen
- aktiviere die an A gebundenen Funktionen für die $I_j(A)$ in der
  Reihenfolge wie in Regel 1

(2.8)

Regel 1 dient der Generierung *partieller* Instanzen, die erforderlich sind, um kontextsensitive Bezüge aufzulösen. Nach der – gegebenenfalls mehrfachen – Anwendung von Regel 1 werden die partiellen Instanzen mit Hilfe von Regel 2 zu kompletten Instanzen vervollständigt. Besitzt ein Konzept keine kontextabhängigen Bestandteile, so können beide Regeln unmittelbar hintereinander angewendet werden.

Regel 3 erlaubt die Erweiterung von Instanzen durch die Berücksichtigung optionaler Bestandteile und Konkretisierungen. Sie wird im allgemeinen nur dann aktiviert, wenn eine berechnete Interpretation das vorliegende Signal nicht genügend überdeckt. Als optional im Sinne der Regelprämisse gelten dabei — neben den in Modalitätsmengen markierten Kanten — auch alle Bestandteile und Konkretisierungen, die häufiger auftreten, als es die minimale Anzahl (vgl. Seite 31) in der jeweiligen Kantenbeschreibung erfordert.

Regel 3: *Erweiterung von Instanzen*
WENN für eine Instanz $I_j(A)$ eine Instanz zu einem Konzept existiert,
das gemäß der Modalitätsbeschreibung von $I_j(A)$ optional ist
DANN generiere erweiterte Instanzen $I_k(A)$ aus $I_j(A)$ wie folgt:
- erzeuge eine leere Instanz für ein $I(A)$
- übernehme die Kantenbeschreibungen der obligatorischen Kanten aus $A_i^{(I)}$
- verbinde die Instanzen mit den in der Prämisse referierten Instanzen
- aktiviere die an A gebundenen Funktionen für die $A_j^{(I)}$ in der Reihenfolge wie in Regel 1

(2.9)

Regel 4: *Datengetriebene Generierung modifizierter Konzepte*
WENN für ein Konzept A oder ein modifiziertes Konzept $Q_i(A)$
eine Instanz oder ein modifiziertes Konzept existiert für
- eine Konkretisierung oder
- ein Bestandteil oder
- einen Kontext

DANN generiere modifizierte Konzepte $Q_j(A)$ wie folgt:
- generiere für $Q_j(A)$ ein leeres modifiziertes Konzept,
- verbinde $Q_j(A)$ mit dem in der Prämisse genannten Knoten,
- übertrage die restlichen Kanten von A bzw. $Q_i(A)$ nach $Q_j(A)$,
- aktiviere die an A gebundenen Funktionen für $Q_j(A)$ in der Reihenfolge
  - Kantenbewertungen
  - Attributwerteberechnungen
  - Attributbewertungen
  - Relationsbewertung
  - Bewertung

(2.10)

Sofern ein Konzept kein Analyseziel repräsentiert, stellen seine Instanzen oder Modifikationen Teilergebnisse dar, die dazu verwendet werden können, die Effizienz der Analyse durch eine Adaption der Wissensbasis an die erzielte Teilinterpretation zu steigern. Die Regeln 4 und 5 etablieren die Generierung modifizierter Konzepte, die Einschränkungen

(*constraints*) für noch nicht instantiierte Konzepte darstellen. Während die Erzeugung modifizierter Konzepte durch Regel 4 *datengetrieben*, also aufgrund berechneter Instanzen für Bestandteile, Konkretisierungen oder einen Kontext erfolgt, ermöglicht Regel 5 eine *modellgetriebene* Generierung. Letztere macht Vorerwartungen für den Analyseprozeß verfügbar, indem Einschränkungen aus übergeordneten Konzepten berücksichtigt werden.

Regel 5: *Modellgetriebene Generierung modifizierter Konzepte*
WENN für ein Konzept A oder ein modifiziertes Konzept $Q_i(A)$
eine Instanz oder ein modifiziertes Konzept existiert für ein
Konzept B mit $B \xrightarrow{kon} A$ oder $B \xrightarrow{bst} A$
DANN generiere modifizierte Konzepte $Q_j(A)$ wie folgt:
- generiere für $Q_j(A)$ ein leeres modifiziertes Konzept,
- verbinde $Q_j(A)$ mit dem in der Prämisse genannten Knoten,
- übertrage die restlichen Kanten von A bzw. $Q_i(A)$ nach $Q_j(A)$,
- aktiviere an A bzw. B gebundene Funktionen für $Q_j(A)$ wie folgt:
  - inverse Kantenbewertung aus B der Kantenbeschreibungen die A als Zielknoten besitzen
  - inverse Attributwerteberechnung aus B, sofern ein Attribut aus A Argument ist
  - inverse Relationsbewertung aus B, sofern ein Attribut aus A Argument ist
  - Kantenbewertungen aus A
  - Attributwerteberechnungen aus A
  - Attributbewertungen aus A
  - Relationsbewertungen aus A
  - Bewertung aus A

(2.11)

Regel 6: *Konzeptschätzung*
WENN für ein Konzept A
Werte für Attribute oder Analyseparameter bekannt sind
DANN generiere modifizierte Konzepte $Q_i(A)$ wie folgt:
- erzeuge ein leeres modifiziertes Konzept $Q_i(A)$,
- trage die bekannten Werte ein,
- aktiviere für $Q_i(A)$ die an A gebundenen Funktionen in der Reihenfolge
  - Attributwerteberechnungen (ohne die bereits gesetzten)
  - Attributbewertungen
  - Relationsbewertung
  - Bewertung

(2.12)

Die letzte Inferenzregel dient zur Schätzung von Zielkonzepten. Liegen Segmentierungsergebnisse vor, aus denen einige Attribute und ihre Werte bekannt sind, so lassen sich mit Hilfe automatisch generierter Attribut–Konzept–Tabellen mögliche Analyseziele bestimmen. Die dadurch vorgenommene Fokussierung der Analyse führt insbesondere bei einer großen Anzahl potentieller Zielkonzepte zu einer erheblichen Effizienzsteigerung.

Bevor wir das Zusammenwirken der Inferenzregeln mit der übergeordneten Kontroll-

strategie skizzieren, wollen wir die Bedeutung der Regeln 1 und 2 für den — besonders im Hinblick auf die angestrebte Parallelverarbeitung interessanten — Datenfluß im Netzwerk hervorheben. Die rekursive Untersuchung der Prämissen beider Regeln liefert einen Instantiierungspfad, der Instanzen zu allen Konzepten enthält, die zur symbolischen Beschreibung eines gegebenen Analyseziels notwendig sind; Bild 2.4 zeigt einen Algorithmus, der auch Basis der Analysevorbereitung für den in dieser Arbeit entwickelten parallelen Kontrollalgorithmus ist.

/* *Bestimmung eines Instantiierungspfades* $\mathcal{I}$ */			
gegeben: Zielkonzept C_g			
initialisiere zwei Listen: PATH := $\emptyset$, REST := $\{C_g\}$			
WHILE REST $\neq \emptyset$			
	entferne das letzte Element A aus REST		
	IF	A ist ein Konzept	
	THEN	bestimme PREM(A) als Liste aller Bestandteile und Konkretisierungen von A aus einer Prämisse von Regel 1	
		IF	A hat kontextabhängige Bestandteile
		THEN	bringe eine partielle Instanz $I^p(A)$ an das Ende von PATH
			bringe eine Instanz $I(A)$ an das Ende von REST
		ELSE	bringe eine Instanz $I(A)$ an das Ende von PATH
		bringe alle Konzepte $C \in$ PREM(A) an das Ende von REST	
	ELSE	/* *A ist eine Instanz* $I(B)$ */	
		bringe $I(B)$ an das Ende von PATH	
Ergebnis: (partielle) Instanzen des Instantiierungspfad $\mathcal{I}$ in PATH			

Bild 2.4: Algorithmus zur Berechnung eines Instantiierungspfades zu einem gegebenen Analyseziel.

Das in Bild 2.5 gezeigte Beispielnetzwerk (nach [Kum93b]) dient zur Erläuterung des Algorithmus. Betrachten wir die weiß eingezeichneten Teile der beiden Jeeps (links) als optional — im Falle der oberen Variante sei dies lediglich der Abweiser, bei der unteren Variante zusätzlich der Überrollbügel und die Windschutzscheibe —, so könnte das Konzept "Jeep" die zwei Modalitätsmengen

- Modalitätsmenge 1:
  obligatorisch : *vorderrad, hinterrad, karosserie, fenster, reserverad,*
  optional : *abweiser*
- Modalitätsmenge 2:
  obligatorisch : *vorderrad, hinterrad, karosserie, reserverad,*
  optional : *abweiser, fenster, ueberrollbuegel*

besitzen. Die *kursiv* gedruckten Kantenrollen sind hierbei — soweit nicht anders eingezeichnet — nach den Zielknoten der jeweiligen Kante benannt, die aus dem rechts abgebildeten Netzwerk zu entnehmen sind.

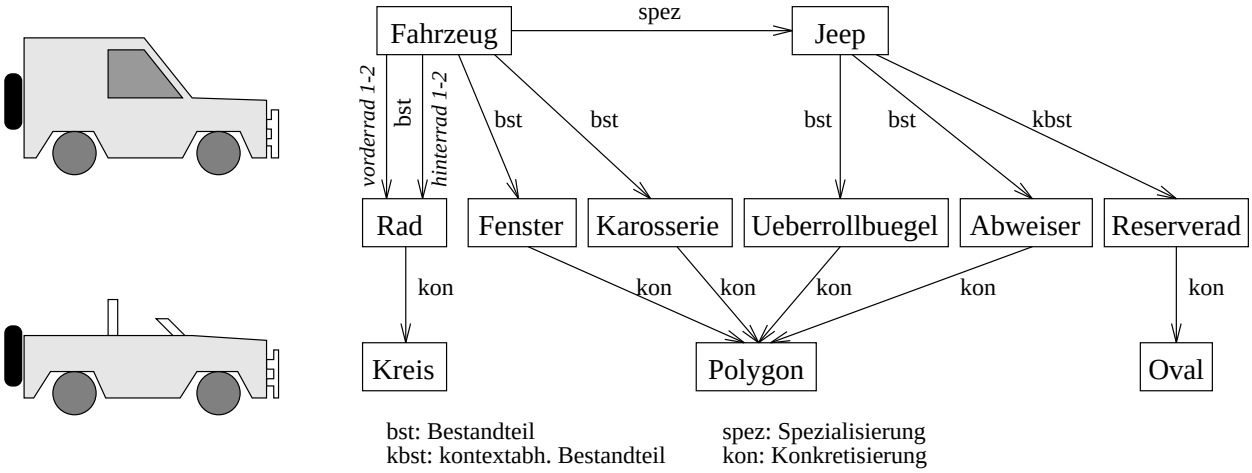


Bild 2.5: Ein Beispielnetzwerk zur Berechnung des Instantiierungspfades (nach [Kum93b]).

Die Anwendung von Algorithmus 2.4 auf das Konzept "Jeep" liefert zwei Instantiierungspfade

$$\mathcal{I}_1 = I^p(\text{Jeep}),\ I(\text{Reserverad}),\ I(\text{Oval}),\ I(\text{Fenster}),\ I_1(\text{Polygon}),$$
$$I(\text{Karosserie}),\ I_2(\text{Polygon}),\ I_1(\text{Rad}),\ I_1(\text{Kreis}),\ I_2(\text{Rad}),\ I_2(\text{Kreis}),\ I(\text{Jeep})$$

und

$$\mathcal{I}_2 = I^p(\text{Jeep}),\ I(\text{Reserverad}),\ I(\text{Oval}),\ I(\text{Karosserie}),\ I(\text{Polygon}),$$
$$I_1(\text{Rad}),\ I_1(\text{Kreis}),\ I_2(\text{Rad}),\ I_2(\text{Kreis}),\ I(\text{Jeep}),$$

welche (partielle) Instanzen zu allen obligatorischen Bestandteilen und Konkretisierungen enthalten. Der Instantiierungspfad ist also aufgrund der Repräsentation von Begriffsvarianten in einem Konzeptknoten keineswegs eindeutig, und läßt im allgemeinen auch keine Aussage über die Reihenfolge der Instantiierung zu, da die Einträge zur lokalen Steuerung der Analyse nicht beachtet werden.

Während die Regelprämissen den netzweiten Informationsfluß einer rein datengetriebenen Analyse definieren, ist der konzeptinterne Datenfluß durch die Aktivierungsreihenfolge der Analyseprozeduren im Aktionsteil der Regeln weitestgehend festgelegt. Aus Gründen der Vollständigkeit sei erwähnt, daß die Berechnungsreihenfolge der Konzepte und Substrukturen in ERNEST automatisch in einer *Analysevorbereitungsphase* ermittelt wird. Diese nimmt auch eine Auflösung der Vererbungen und die Überprüfung der formalen Konsistenzkriterien (2.4 – 2.6) vor und sichert die Zyklenfreiheit der Argumentbeziehungen unter den Analyseprozeduren.

/* *Basiskontrolle in* ERNEST */				
gegeben : Menge von Zielkonzepten C_g, $g = 1, \ldots, n$, OPEN := $\emptyset$				
/* *initiale Schätzung von Analysezielen* */				
Bilde modifizierte Konzepte $Q_j(C_g)$ durch Anwendung von Regel 6, generiere und bewerte Suchbaumknoten v_{gj}, bringe Knoten nach OPEN				
/* *Kontrolle durch Zustandsraumsuche* */				
WHILE OPEN$\neq \emptyset$				
	entferne den besten Knoten v_b aus OPEN			
	IF	Die Objekte in v_b erfüllen ein Analyseziel		
	THEN	STOP mit ERFOLG /* *Interpretation im Knoten* v_b */		
	IF	es können neue Analyseziele G_i etabliert werden		
	THEN	benutze Regel 1 - 4 und die Objekte aus v_b zur Einschränkung der G_i, bilde neue Suchbaumknoten		
	ELSE	IF	ein Objekt aus v_b kann instantiiert werden	
		THEN	instantiiere und modifiziere die Objekte aus v_b soweit möglich mit den Regeln 1 - 5, bilde neue Suchbaumknoten	
		ELSE	IF	in v_b ist ein Objekt mit nichterfüllter Prämisse gemäß Regel 1 und 2
			THEN	expandiere das Modell und modifiziere soweit möglich mit den Regeln 4 und 5, bilde neue Suchbaumknoten
			ELSE	suche nach optionalen Teilen und Spezialisierungen
	bringe neue Suchbaumknoten nach OPEN			
STOP mit FEHLER; /* *Daten nicht interpretiert* */				

Bild 2.6: Überblick über die Basiskontrolle der ERNEST–Systemschale (nach [Nie90b]).

Die Einbettung der Inferenzregeln in den A^*–Algorithmus ist in Anlehnung an [Nie90b, Kum92b] in Bild 2.6 dargestellt, das einen groben Überblick über die Basiskontrolle gibt. Eine ausführliche Beschreibung, die auch die Nutzung der Informationen zur lokalen Steuerung (Vorrang, Prioritäten, Identifikationen) umfaßt, und eine Charakterisierung des Kontrollalgorithmus als *sequentieller Entscheidungsprozeß* findet sich in [Kum92a, Kap. 4.3 – 4.8].

In dem während der Analyse generierten Suchbaum dienen die Knoten zur Aufnahme von Analysezuständen, die jeweils aus einer konsistenten Menge von Objekten (Konzepte, Instanzen, modifizierte Konzepte) und den zugehörigen Sensordaten bzw. Segmentierungsergebnissen bestehen. Die Modifikation möglicher Zielkonzepte durch die Anwendung der Regel 6 sorgt für die Initialisierung der Analyse. Mit den dabei generierten Suchbaumknoten beginnt der eigentliche Suchvorgang. Dazu wird zunächst ein anwendungsspezifisches Abbruchkriterium überprüft. Kann die Analyse noch nicht beendet werden (beispielsweise

weil der betrachtete Knoten keine Instanz enthält, die eine ausreichende Interpretation gewährleistet) wird — ebenfalls anwendungsabhängig — versucht, aufgrund der aktuellen Daten neue Analyseziele zu etablieren. Durch die Aktivierung der Regeln 1 – 4 werden die vorliegenden Daten zur Propagierung von Einschränkungen der neuen Ziele genutzt.

Können keine neuen Analyseziele bestimmt werden, so wird zunächst versucht, unter Berücksichtigung der lokalen Steuerungsinformation ein Objekt zu finden, das instantiiert werden kann. Ist ein solches vorhanden, so werden die Objekte des Knotens so weit wie möglich instantiiert und modifiziert (Regeln 1 – 5), wobei Nachfolger des aktuellen Knotens zur Aufnahme konkurrierender Ergebnisse erzeugt werden. Ist keine Instantiierung möglich, wird der Knoten durch die Hinzunahme von Objekten gemäß den Prämissen der Regeln 1 und 2 expandiert; neue Suchbaumknoten entstehen in diesem Fall durch die Bildung modifizierter Konzepte unter Verwendung der Regeln 4 und 5.

Die Behandlung optionaler Kanten und speziellerer Konzepte wird nur angestoßen, wenn ein Analyseziel instantiiert ist, die Sensordaten jedoch nicht ausreichend interpretiert (überdeckt) werden konnten.

2.4 Zusammenfassung

Das Erlanger Semantische Netzwerksystem ERNEST ist eine Systemschale für die Interpretation komplexer Sensordaten, deren modulare Architektur sich an einem allgemeinen Aufbau wissensbasierter Musteranalysesysteme orientiert und die in mehreren Anwendungsgebieten des Bild– und Sprachverstehens eingesetzt wird. Als Grundlage der folgenden Untersuchungen zur Parallelverarbeitung in semantischen Netzen wurden in diesem Kapitel die Module zur Wissensrepräsentation und –nutzung vorgestellt.

Die Netzwerksprache ermöglicht die homogene Repräsentation von Begriffen unterschiedlicher Abstraktionsebenen in einer Wissensbasis. Die Konzepte eines semantischen Netzwerkes werden durch Attribute und Relationen näher beschrieben und können durch Spezialisierungs–, Bestandteils– und Konkretisierungskanten zueinander in Beziehung gesetzt werden. Bestandteile und Konkretisierungen können als obligatorisch oder optional markiert und in Modalitätsmengen zu zulässigen Kombinationen zusammengefaßt werden, wodurch die Modellierung unterschiedlicher Varianten eines Begriffes in einem Netzwerkknoten ermöglicht wird.

Durch eine einheitliche Schnittstelle zu den Analyseprozeduren unterstützen Konzepte und ihre Substrukturen die Anbindung prozeduralen Wissens zum Abgleich der dargestellten Begriffe mit den vorliegenden Sensordaten. Zusätzliche Einträge erlauben die Formulierung problemabhängigen strategischen Wissens, das zur Steigerung der Effizienz des Analyseprozesses eingesetzt wird.

Ziel der Analyse ist die Generierung einer Instanz zu einem Zielkonzept, die kompati-

bel zur Wissensbasis ist und die Sensordaten möglichst gut ausnutzt. Die Erzeugung von Instanzen und modifizierten Konzepten, die Einschränkungen der Wissensbasis aufgrund von Teilergebnissen der Analyse darstellen, beruht auf sechs problemunabhängigen Inferenzregeln. Diese definieren die prozedurale Semantik der Netzwerksprache und legen die Reihenfolge der Instantiierung und der dazu notwendigen Aktivierung der Analyseprozeduren fest. Als übergeordnete Kontrollstrategie dient der A^*–Algorithmus zur Bestimmung eines optimalen Pfades in einem Suchraum, der durch die Verfolgung alternativer Modellierungen und konkurrierender Instantiierungen entsteht.

Kapitel 3

Parallelverarbeitung in ERNEST

Aufbauend auf der Vorstellung der ERNEST–Systemschale im vorigen Kapitel diskutieren die folgenden Abschnitte Möglichkeiten der Parallelverarbeitung in dem durch die Inferenzregeln und den A^*-Algorithmus vorgezeichneten Analyseprozeß.

Zunächst motiviert Abschnitt 3.1 die Auswahl einer Teilmenge der Netzwerksprache, welche die Elemente zur Wissensrepräsentation und –nutzung umfaßt, die sowohl bei unseren Überlegungen zur Parallelisierung der ERNEST–Kontrolle als auch in der Entwicklung eines neuen Kontrollalgorithmus berücksichtigt werden. Die dabei erarbeitete, an der Netzwerksyntax orientierte Beschreibung der Sprachkonstrukte nutzen wir in Abschnitt 3.2 zu einer (groben) Abschätzung der Komplexität des Instantiierungsprozesses; sie ist darüber hinaus Ausgangspunkt für die Bestimmung von Elementaroperationen für das in Kapitel 4 vorgestellte iterativ–optimierende Verfahren. Daran anschließend erörtert Kapitel 3.3 Aspekte der Parallelverarbeitung auf der Ebene einzelner Konzepte und bei der Abarbeitung der Konzepte eines Instantiierungspfades sowie Parallelisierungsmöglichkeiten für die übergeordnete heuristische Suche.

3.1 Komponenten für die Parallelisierung

Die Parallelisierung der im Netzwerk ablaufenden Operationen wird durch mehrere Randbedingungen kompliziert:

- Das Volumen der zu verarbeitenden Daten hängt von den Ergebnissen der initialen Segmentierung ab und kann selbst für Muster eines Problemkreises erheblich variieren. Dies erschwert sowohl die theoretische Abschätzung der Komplexität der Analyse als auch die Beseitigung von Engpässen durch den Einsatz mehrerer Prozessoren oder Rechnerknoten.

- In ihrem dargestellten Umfang sind sowohl die Netzwerksprache als auch die Inferenzmechanismen das Ergebnis eines mehrjährigen Entwicklungsprozesses, in den

— neben generellen Aspekten der wissensbasierten Musteranalyse — Anforderungen und Erfahrungen aus unterschiedlichen Anwendungen eingang fanden. Als Beispiel mögen die effiziente Darstellung räumlicher und zeitlicher Bezüge durch Adjazenzen und die Einführung des Vorranges für die Instantiierung von Konzepten dienen; seitens des Kontrollalgorithmus können die Bereitstellung der Inferenzregel zur Zielkonzeptschätzung und zur Generierung modifizierter Konzepte genannt werden. Im Vergleich zu Systemen, die keine unsicheren bzw. mehrdeutigen Daten verarbeiten müssen oder die auf einzelne Anwendungen spezialisierte Inferenzmechanismen besitzen, wird dadurch die Isolation parallel ausführbarer Operationen und deren Verteilung erschwert.

- Es ist a priori nicht vorherzusagen, ob sich die zur Effiziensteigerung der *sequentiellen* Analyse eingeführten Techniken auch im Falle einer *parallelen* Verarbeitung als sinnvoll erweisen. Beispielsweise ist es in bestimmten Analysesituationen möglich, bei der Bearbeitung von Objekten des aktuellen Suchbaumknotens Instanzen aus anderen Knoten zu kopieren. Dadurch wird eine aufwendige Instantiierung auf Kosten zusätzlich durchzuführender, einfacherer Tests eingespart; es ist jedoch fraglich, ob dieses Vorgehen bei einer parallelen Verfolgung alternativer Pfade im Suchbaum aufgrund der zusätzlich erforderlichen Kommunikation effizient ist.

Während die Verarbeitung unsicherer und mehrdeutiger Daten beim heutigen Stand der Technik eine nicht zu umgehende Notwendigkeit für die wissensbasierte Musteranalyse darstellt, können wir die letztgenannte Problematik dadurch abmindern, daß wir uns auf die wichtigsten Modellierungsmöglichkeiten und die Basisinferenzmechanismen konzentrieren. Neben den Mitteln zur Wissensakquisition und zur Erklärung der Analyseergebnisse, deren eher ergänzender Charakter für die Systemfunktion bereits in Kapitel 2 erwähnt wurde, schließen wir daher die folgenden Netzwerkelemente von unseren Überlegungen aus, ohne die Ausdrucksmöglichkeiten der Netzwerksprache *im wesentlichen* zu verringern:

- Referentielle und inhärente Kanten: Diese kommen gegenwärtig nur bei der Modellierung linguistischen Wissens im Spracherkennungs– und Dialogsystem EVAR [Mas94] zum Einsatz. (Vorschläge zu ihrer Behandlung im Rahmen eines parallelen Kontrollalgorithmus werden in Kapitel 6 skizziert.)

- Adjazenzen: Räumliche und zeitliche Bezüge zwischen Konzepten sowie die Forderung nach Kohärenz können ohne weiteres durch Attribute und Relationen modelliert werden.

- Lokale Attribute: Eigenschaften eines Konzeptes können auch in der zugehörigen Attributbeschreibung von der Vererbung an speziellere Konzepte ausgeschlossen werden.

Modalitätsmengen und die in der Kantenbeschreibung verankerten Mittel zur Unterstützung der objektzentrierten Repräsentation könnten ebenfalls unberücksichtigt bleiben, ohne die in Konzepten darstellbaren Begriffe und Sachverhalte einzuschränken; sie werden jedoch im folgenden aufgrund ihrer Bedeutung für die Komplexität der Analyse (siehe Abschnitt 3.2) weiterhin zugelassen.

Bezeichnen wir mit T_C, T_A, T_S und T_E die Rollennamen, welche Konzepte, Attribute, Relationen und Kanten netz– bzw. konzeptweit eindeutig identifizieren, und mit $\mathcal{T}_C$, $\mathcal{T}_A$, $\mathcal{T}_S$ und $\mathcal{T}_E$ die Mengen sämtlicher dieser Rollen, so ist ein Konzept unter den getroffenen Vereinbarungen durch

- die Angabe seiner Attributbeschreibungen A, auch zur Aufnahme der Analyseparameter (vgl. Abschnitt 2.2),
- seiner Relationsbeschreibungen S (ebenfalls ohne Unterscheidung zwischen Struktur– und Analyserelationen),
- die Bestimmung seiner Spezialisierungen V,
- die Zusammenfassung der Kantenbeschreibungen für Bestandteile P_{ci}, kontextabhängige Bestandteile P_{cd} und Konkretisierungen K in Modalitätsmengen H,
- sowie — durch den Verwendungszweck Musteranalyse motiviert — die Angabe einer Funktion G_C zur Bewertung von Instanzen und modifizierten Konzepten

vollständig definiert:

$$
\begin{aligned}
C \;=\; (\;& T_C, \\
& (A : T_A)^*, (S : T_S)^*, \\
& (V : T_{C'})^*, \\
& H^*, \\
& (G_C : 2^{\mathcal{T}_A \cup \mathcal{T}_S \cup \mathcal{T}_E} \mapsto J) \quad), \\
H \;=\; (\;& ((P_{cd} : T_E)^*, (P_{ci} : T_E)^*, (K : T_E)^*)_{obl}, \\
& ((P_{cd} : T_E)^*, (P_{ci} : T_E)^*, (K : T_E)^*)_{opt} \quad).
\end{aligned}
\tag{3.1}
$$

Hierbei wird der Sternoperator als abkürzende Schreibweise benutzt; wir schreiben $(V : T_C)^*$ für die Menge der in einem Konzept angegebenen Spezialisierungen, die natürlich auch leer sein kann. Die Bewertungsfunktion G_C berücksichtigt die durch ihre Rollen spezifizierten Attribute, Relationen und Kanten eines Konzeptes und bewertet sie in einem beliebigen Kalkül J. Im Konzept "Lüfterrad" in Bild 2.2 auf Seite 33 sind die hier eingeführte Notation und die Verknüpfung von Attributen ("schwerpkt") und Relationen ("kollinear") durch die Bewertungsfunktion illustriert.

Die Attributbeschreibung

$$\begin{aligned} A \;=\; (\;\; & T_A, \\ & (m : \{\texttt{JA}, \texttt{NEIN}, \texttt{DELETED}, T_{A'}\}), \\ & (W : \{\texttt{INTEGER}, \texttt{REAL}, \ldots\}), \\ & (B \subseteq W),\ (n \in \mathbb{N}), \\ & (\varphi_A : 2^{\mathcal{T}_A} \mapsto W^n), \\ & (G_A : \{T_A\} \mapsto J)\ ). \end{aligned} \tag{3.2}$$

dient zur genaueren Definition von Attributen und Analyseparametern und stellt Möglichkeiten zur Anbindung der Analyseprozeduren zur Attributwerteberechnung und –bewertung bereit. Der Modifiziert–Eintrag m ermöglicht die explizite Beeinflussung der Vererbung entlang der Spezialisierungskanten und wird mit `NEIN`, `DELETED` oder `JA` oder der Rolle $T_{A'}$ eines Attributes aus einem allgemeineren Konzept belegt. Soll ein Attribut eines allgemeinen Konzeptes in einer Spezialisierung gelöscht werden, so ist es dort mit identischer Rolle und der Konstante `DELETED` anzugeben. Bei der Änderung einzelner Einträge in der Attributbeschreibung, beispielsweise der Angabe einer engeren Restriktion für das Attribut des spezielleren Konzeptes, wird der Eintrag auf `JA` gesetzt, wenn die funktionale Rolle erhalten bleibt, andernfalls ist die Rolle des zu modifizierenden Attributes anzugeben.

Die Einträge Werttyp W, Wertanzahl n und Restriktion B definieren den Datentyp eines Attributwertes. Neben den in Gleichung (3.2) angedeuteten programmiersprachlichen Basistypen sind als Werttypen auch komplexe, an die jeweilige Anwendung angepaßte Datenstrukturen zulässig, die als `RECORD`s bezeichnet werden. Die Wertanzahl n gibt an, wieviele nicht konkurrierende Werte für ein Attribut benötigt werden, und die Restriktion dient — falls möglich — zur Einschränkung des Wertebereiches auf ein bestimmtes Intervall oder einzelne Werte aus W.

Die Notwendigkeit der Anbindung prozeduralen Wissens an die Netzwerkelemente wurde bereits in Kapitel 2 begründet. In der Attributbeschreibung sind je eine Funktion zur Werteberechnung φ_A und zur Bewertung G_A anzugeben. Während erstere aus Attributen oder Sensordaten (bei nicht belegtem Argument–Eintrag) Werte des Attributes berechnet, nimmt letztgenannte — sofern vorhanden mit Hilfe der Restriktion — eine lokale Bewertung der Attributwerte für die erzeugten Instanzen oder modifizierten Konzepte vor.

Ein wichtiges Merkmal der Netzwerksprache, das wir in Kapitel 4 zur feingranularen Parallelisierung der Instantiierungsoperationen nutzen werden, besteht in der Verknüpfung von Attributen, Relationen und den Bewertungen für Kanten und Konzepte durch die Analyseprozeduren. Neben den Abhängigkeiten zwischen Konzepten, die in Gleichung (3.1) durch die Modalitätsbeschreibungen H und Spezialisierungen V gegeben sind, können somit Beziehungen zwischen Elementen der Wissensbasis auf einer subkonzeptuellen Ebene explizit angegeben werden. Diese, für die automatische Zerlegung komplexer Konzepte ele-

mentare Eigenschaft wird durch die Angabe von Rollennamen im Definitionsbereich der Analyseprozeduren hervorgehoben. Ist $\mathcal{T}'_A = \{T_{A'}, T_{A''}, \ldots, T_{A^{(n)}}\}$ eine Teilmenge von $\mathcal{T}_A$ so bezeichnet $\varphi_A : \mathcal{T}'_A \mapsto W$ eine Funktion

$$\varphi_A \;:\; W_{T_{A'}} \times W_{T_{A''}} \times \ldots \times W_{T_{A^{(n)}}} \mapsto W_{T_A}$$

und analoge Bezeichnungen gelten für die Bewertungsfunktionen G in den Gleichungen (3.1 – 3.4). In Bild 3.1 sind die Komponenten der Attributbeschreibung und die eingeführten Bezeichnungen für das Attribut "steigung" des Konzepts "V1" aus dem Netzwerkausschnitt auf Seite 65 illustriert. Da es sich um eine Spezialisierung eines Attributes mit gleicher Rolle in einem allgemeineren Konzept "Linie" handelt, ist der Modifiziert–Eintrag mit der Konstanten `JA` belegt. Die Berechnung der Attributwerte aus der initialen Segmentierung $\mathcal{A}$ und deren Bewertung unter Verwendung der vorhandenen Restriktion sind im mittleren und rechten Teil des Bildes schematisch dargestellt.

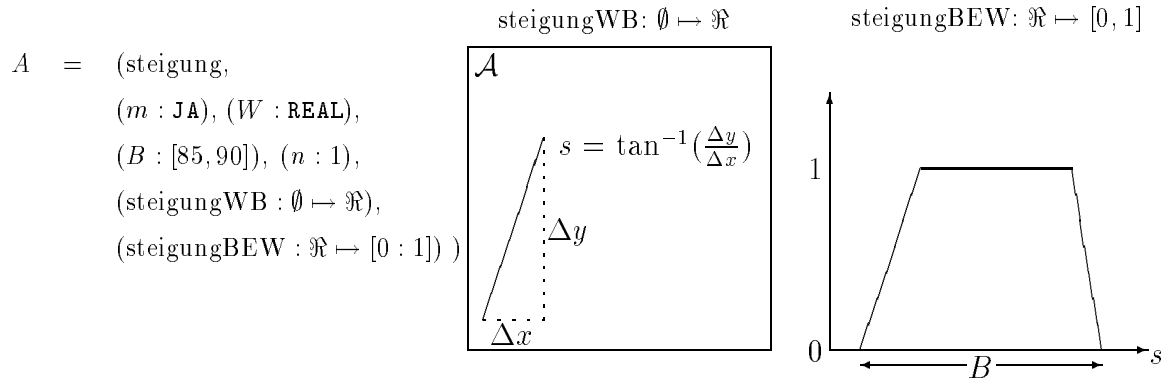


Bild 3.1: Die Attributbeschreibung für das Attribut "steigung" aus Bild 3.8 (links) und eine schematische Darstellung von Attributwerteberechnung (mitte) und Attributbewertung (rechts).

Relationsbeschreibungen unterliegen den gleichen, im Modifiziert–Eintrag zu beeinflussenden Vererbungsregeln und besitzen als einzige weitere Komponente einen Eintrag zur Angabe einer Bewertung G_S, die den Grad der Erfülltheit der Relation mißt, und deren Argumente mit den Rollen der beteiligten Attribute belegt werden. Als Beispiel ist in Bild 3.2 die Relationsbeschreibung

$$\begin{aligned} S \;=\; (\;& T_S, \\ & (m : \{\texttt{JA}, \texttt{NEIN}, \texttt{DELETED}, T_{S'}\}), \\ & (G_S : 2^{\mathcal{T}_A} \mapsto J)\;). \end{aligned} \tag{3.3}$$

der Relation "v-parallel" zur Beurteilung der Parallelität vertikaler Liniensegmente aus Bild 3.8 (vgl. Seite 65) dargestellt. Bei der Bewertung wird vereinfachend davon ausgegangen, daß zwei Linien bereits dann als parallel betrachtet werden dürfen, wenn sie die gleiche Steigung besitzen. Geringe Unterschiede werden dabei durch die Abstufung der Bewertung G_S toleriert; ist eines der Segmente jedoch keine vertikale Linie, so gilt $G_S = 0$.

$$
\begin{aligned}
S \quad = \quad & (\text{v-parallel}, \\
& (m : \texttt{NEIN}), \\
& (\text{v-parallelBEW}: \Re^2 \mapsto [0:1]) \ ) \\
& \qquad (\text{vertikale1.steigung}, \\
& \qquad \ \text{vertikale2.steigung}) \mapsto [0:1]
\end{aligned}
$$

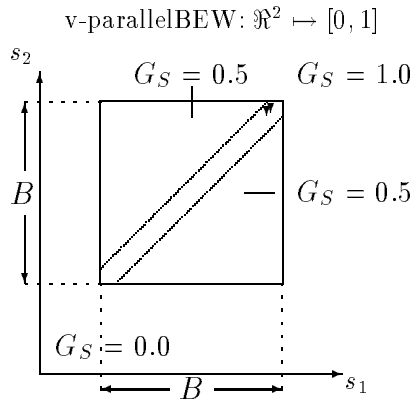


Bild 3.2: Die Relationsbeschreibung für die Relation "v-parallel" aus Bild 3.8 (links) und eine schematische Darstellung der Relationsbewertung (rechts).

Bestandteils– und Konkretisierungsbeziehungen von Konzepten werden durch die Kantenbeschreibung

$$
\begin{aligned}
E \quad = \quad (\quad & T_E, \\
& (m : \{\texttt{JA}, \texttt{NEIN}, \texttt{DELETED}, T_{E'}\}), \\
& (g : T_C)^+, (d : (d_{min}, d_{max}) \in \mathbb{N} \times \mathbb{N}), \\
& (c : \{\texttt{JA}, \texttt{NEIN}\}), (b : \{\texttt{JA}, \texttt{NEIN}\}), \\
& (G_E : 2^{T_C} \mapsto J) \ )
\end{aligned}
\tag{3.4}
$$

modelliert. Neben dem Modifiziert–Eintrag finden sich hier zunächst die Zielknotenliste g und der Dimension–Eintrag d als weitere Möglichkeit zur Beschreibung unterschiedlicher intensionaler Varianten eines Begriffes. In der Zielknotenliste werden alle Konzepte angegeben, welche die durch die Kante repräsentierte Bestandteils– oder Konkretisierungsrelation erfüllen können. Der Plusoperator zeigt an, daß mindestens ein Konzept angegeben werden muß. Die *minimale* Kantendimension d_{min} gibt an, wie oft eine Instanz des Subkonzeptes zur Instantiierung eines Konzeptes in den Sensordaten vorhanden sein muß, die *maximale* Dimension d_{max} beschränkt die Anzahl mit der ein Bestandteil oder eine Konkretisierung höchstens vorhanden sein darf.

In einer konkreten Analysesituation ist genau eine Instanz oder Modifikation eines Konzeptes aus der Zielknotenliste (als Bestandteil oder Konkretisierung) an die Kante ge-

bunden. Dieses Objekt ist implizites Argument der Kantenbewertung G_E, die seine Aussage bezüglich des übergeordneten Konzeptes überprüft.

Während die bisher erwähnten Elemente der Kantenbeschreibung für beide Kantentypen identisch sind, kann der Kontext–Eintrag c nur für Bestandteilskanten belegt werden. Bestandteile P_{cd}, die nur im Zusammenhang des übergeordneten Konzeptes (eines größeren Kontextes) definiert und analysiert werden können, sind durch die Belegung des Eintrags mit der Konstanten `JA` als kontextabhängig zu kennzeichnen.

Der — ebenfalls nur für Bestandteilskanten definierte — Komplementeintrag b, der die Modellierung "negativer" Bestandteile ("B ist *nicht* Teil von A") erleichtert, vervollständigt die Kantenbeschreibung. Die identische Behandlung "positiver" und "negativer" Bestandteile durch die Inferenzregeln — ihre Instanzen oder modifizierten Konzepte sind während der Analyse lediglich durch ihren Beitrag zur Bewertung des gemeinsamen Ganzen zu unterscheiden — erlaubt es, diesen Eintrag im folgenden zu vernachlässigen.

3.2 Komplexität der Basiskontrolle

Vor der Beschreibung unserer Überlegungen zur Parallelisierung der ERNEST-Kontrolle in Kapitel 3.3 untersuchen wir in diesem Abschnitt die Komplexität des in Bild 2.6 dargestellten Basisalgorithmus, um einen Eindruck davon zu gewinnen, welchen Erfolg der Einsatz mehrerer Prozessoren verspricht.

Während die Komplexität der Instantiierung eines gegebenen Zielkonzepts allein von den problemunabhängigen Inferenzregeln bestimmt wird, zeigen sich für die übergeordnete Graphsuche auch anwendungsabhängige Einflußgrößen. Hier ist insbesondere die Qualität der initialen symbolischen Beschreibung zu erwähnen: die Ausführungen in Kapitel 2.3 verdeutlichen, daß eine Vielzahl konkurrierender Segmentierungsergebnisse nicht die Anzahl zu instantiierender Subkonzepte, wohl jedoch den Suchaufwand vergrößert. Trotz der engen Verzahnung von Instantiierungsoperationen und Kontrollstrategie, die beispielsweise bei der Behandlung optionaler Kanten deutlich wird, nehmen wir daher im folgenden eine getrennte Betrachtung beider Verarbeitungsebenen vor. Diese Separierung wird sich auch bei der Konzeption eines massiv–parallelen Kontrollalgorithmus in Kapitel 4 als sinnvoll erweisen.

3.2.1 Komplexität der Instantiierung

Als Maß für den Instantiierungsaufwand eines Zielkonzeptes bietet sich die Länge des von Algorithmus 2.4 berechneten Instantiierungspfades an, die proportional zur Anzahl der zu behandelnden (Sub-)konzepte ist. Da die Instantiierung und Modifikation von Konzepten an die Expansion von Suchbaumknoten geknüpft ist (vgl. Algorithmus 2.6, S. 42), bestimmt die Pfadlänge auch die Tiefe des vom Kontrollalgorithmus generierten Zustandsraumes, die

kennzeichnend für die Größe des Suchproblems ist.

Reine Spezialisierungshierarchien, in denen keine Bestandteils– und Konkretisierungsrelationen zugelassen sind, stellen einen Spezialfall dar. Da die Prämisse von Regel 1 in diesem Fall stets leer (erfüllt) ist, kann jedes Konzept sofort instantiiert werden und besitzt einen Instantiierungspfad der Länge 1. Die Annahme eines identischen Aufwandes für die Instantiierung aller Konzepte trifft jedoch sicherlich nicht zu, da die *lokale Komplexität* L_C eines Konzeptes, unter der die Anzahl zu berechnender Attribute und Relationen zu verstehen ist, von dessen Lage in der Spezialisierungshierarchie abhängig ist. Letztere wird durch den Spezialisierungsgrad δ_{spez} aus Gleichung (2.1) beschrieben, der die Entfernung in Spezialisierungskanten von einem allgemeinsten Konzept angibt. Da die Vererbung der Substrukturen von einem allgemeinen auf ein spezielleres Konzept erfolgt, ist die lokale Komplexität eines Konzeptes proportional zu seinem Spezialisierungsgrad.

In baumartigen Taxonomien, die sich in einigen Anwendungsgebieten als ausreichend erweisen (z.B. [Sag90b]), ergibt sich die Anzahl der Konzepte, die Attribute und Relationen an das betrachtete Zielkonzept vererben, direkt aus dessen Spezialisierungsgrad. Durch

$$L_C(C_g) \quad = \quad a \cdot \delta_{spez}(C_g) \tag{3.5}$$

kann somit eine *lineare* Abhängigleit des Instantiierungsaufwands vom Spezialisierungsgrad des Analyseziels festgehalten werden. Allgemeinere Taxonomien erlauben, daß ein Konzpet Substrukturen von mehreren direkten Generalisierungen erbt, deren (mittlere) Anzahl hier mit $(\#gen)$ bezeichnet sei. Die Anzahl aller Konzepte, die Substrukturen an C_g vererben, ist im schlechtesten Fall *exponentiell* abhängig vom Spezialisierungsgrad $\delta_{spez}(C_g)$, und die Anzahl der zu berechnenden Attribute und Relationen von C_g ergibt sich zu

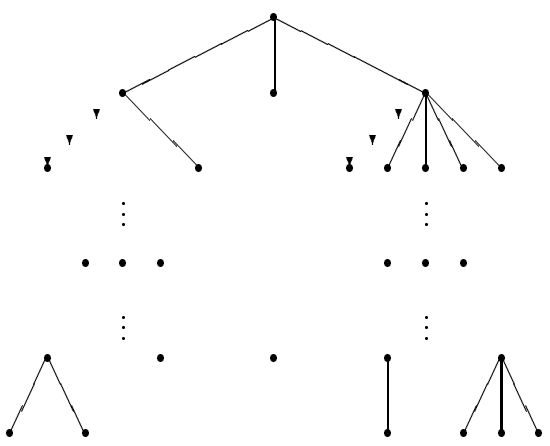
$$L_C(C_g) \quad = \quad a \cdot (\#gen)^{\delta_{spez}(C_g)} \tag{3.6}$$

Da die Netzwerksprache die explizite Löschung bzw. eine Überlagerung der Substrukturen eines allgemeineren Konzeptes in seinen Spezialisierungen erlaubt, liefern die Gleichungen (3.5) und (3.6) jeweils obere Schranken für den Instantiierungsaufwand in reinen Spezialisierungshierarchien, wenn für a die maximale Anzahl der in einer Generalisierung von C_g definierten Attribut– und Relationsbeschreibungen eingesetzt wird. Bei der Bestimmung von a dürfen dabei nur die Substrukturen berücksichtigt werden, deren Modifiziert–Eintrag nicht mit der Konstante `DELETED` belegt ist.

Netzwerke, die nur Spezialisierungskanten enthalten, sind für die wissensbasierte Musteranalyse von geringer praktischer Bedeutung, da sie weder die Dekomposition von komplexen Begriffen zur Vereinfachung der Analyse ermöglichen noch die Repräsentation unterschiedlicher konzeptueller Systeme erlauben. Für Wissensbasen, die zusätzlich Bestandteils– und Konkretisierungsbeziehungen zwischen Konzepten zulassen, kann jedoch aus obigen Überlegungen neben einer Zunahme der lokal durchzuführenden Berechnungen auch ge-

folgert werden, daß die Anzahl der (direkten) Subkonzepte eines gegebenen Analyseziels ebenfalls mit seinem Spezialisierungsgrad anwächst, da Kantenbeschreibungen den gleichen Vererbungsmechanismen unterliegen wie die bisher betrachteten Attribut– und Relationsbeschreibungen. Die so bedingte Zunahme der Länge des Instantiierungspfades mit der Spezialität der gesuchten Interpretation wird von der ERNEST–Kontrolle dadurch berücksichtigt, daß allgemeine Konzepte bevorzugte Analyseziele bilden, und die Instantiierung von Spezialisierungen erst dann versucht wird, wenn auf einem allgemeineren Niveau keine Interpretation gefunden werden konnte (vgl. Algorithmus 2.6).

Im folgenden wird nun untersucht, welche Netzwerkelemente den Instantiierungsaufwand für ein Analyseziel eines festen Spezialisierungsgrades beeinflussen. Dabei ist zunächst festzuhalten, daß als Folge der objektzentrierten Modellierung unterschiedlicher Begriffsvarianten in einem Knoten der Wissensbasis in Netzwerken mit Bestandteils– und Konkretisierungskanten zu einem Konzept im allgemeinen mehrere Instantiierungspfade unterschiedlicher Länge existieren. Neben der (generellen) Komplexität ist daher auch die Ermittlung einer mittleren Pfadlänge und der Beitrag einzelner Strukturierungsmittel (wie z.B. der Modalitätsmengen) von praktischem Interesse; bei deren Bestimmung dient uns die Betrachtung von — zunächst baumartigen — Graphen als Resultat eines *inhomogenen Reproduktionsprozesses* (z.B. [Fel50, Pea84]) als Hilfsmittel.



Ebene	Größe	Erwartungswert
0	k_0	$\mathcal{E}(k_0) = 1$
1	k_1	$\mathcal{E}(k_1) = b \cdot q_1 \cdot \mathcal{E}(k_0)$
2	k_2	$\mathcal{E}(k_2) = b \cdot q_2 \cdot \mathcal{E}(k_1)$
l	k_l	$\mathcal{E}(k_l) = b \cdot q_l \cdot \mathcal{E}(k_{l-1})$
$d-1$	k_{d-1}	
d	k_d	$\mathcal{E}(k_d) = b \cdot q_d \cdot \mathcal{E}(k_{d-1})$

Bild 3.3: Veranschaulichung eines inhomogenen Reproduktionsprozesses (nach [Pea84]).

Ausgangspunkt eines Reproduktionsprozesses (vgl. Bild 3.3) ist die Wurzel des Baumes in der 0–ten Generation (Ebene), die eine feste Anzahl b von Nachfolgern, die erste Generation, erzeugt. Diese "überleben" mit einer Wahrscheinlichkeit q_1, und erzeugen ihrerseits die Knoten der nächsten Ebene. Der so fortgesetzte Prozeß bricht ab, wenn eine Generati-

on "ausstirbt", d.h. die Überlebenswahrscheinlichkeit einer Ebene gleich Null ist. Während man bei einfachen Reproduktionsprozessen von einer festen Überlebenswahrscheinlichkeit auf jeder Ebene ausgeht, kann sich diese bei inhomogenen Prozessen für jede Generation ändern; es wird jedoch angenommen, daß sie für alle Knoten einer Generation identisch und unabhängig von der Knotenanzahl der Ebene ist. Die Anzahl

$$k_l = \begin{cases} \sum_{i=1}^{k_{l-1}} K_{i,l-1} & \text{falls} \quad k_{l-1} \geq 1 \\ 0 & \text{falls} \quad k_{l-1} = 0 \end{cases} \tag{3.7}$$

der Knoten auf der Ebene l wird als Summe von Zufallsvariablen $K_{i,l-1}$ aufgefaßt, die angeben, wieviele Nachfolger der i–te Knoten auf der Ebene $l-1$ erzeugt. Der Erwartungswert

$$\mathcal{E}(k_l) = \sum_{i=1}^{\mathcal{E}(k_{l-1})} \mathcal{E}(K_{i,l-1}) \tag{3.8}$$

darf wegen der Unabhängigkeit der Überlebenswahrscheinlichkeiten vom Knotenindex i zu

$$\mathcal{E}(k_l) = \mathcal{E}(k_{l-1}) \cdot \mathcal{E}(K_{l-1}) \tag{3.9}$$

vereinfacht werden. Erzeugt jeder Knoten b Nachfolger, so ist K_{l-1} unter den genannten Voraussetzungen binomialverteilt und besitzt

$$\mathcal{E}(K_{l-1}) = b \cdot q_l \tag{3.10}$$

als Erwartungswert. Da auf der Ebene 0 genau ein Knoten existiert, ist $\mathcal{E}(k_0) = 1$ und aus den Gleichungen (3.9) und (3.10) erhält man

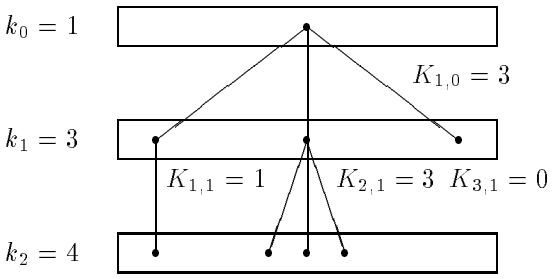
$$\mathcal{E}(k_l) = b^l \cdot q_l \cdot q_{l-1} \cdot \ldots \cdot q_1, \quad l \geq 1 \tag{3.11}$$

als Erwartungswert für die Anzahl der Knoten auf der Ebene l. Schließlich gewinnt man durch Summation über die d Ebenen einen Erwartungswert

$$\begin{aligned} \mathcal{E}(\eta) &= \sum_{i=0}^{d} \mathcal{E}(k_i) \\ &= \sum_{i=0}^{d} (b^i \cdot \prod_{j=1}^{i} q_j) \end{aligned} \tag{3.12}$$

für die Anzahl η der Knoten des gesamten Baumes. In Bild 3.4 sind die Bezeichnungen aus den Gleichungen (3.8 – 3.12) und der dadurch beschriebene Reproduktionsprozeß an einem kleinen Beispiel verdeutlicht.

Die Nutzung von Gleichung (3.12) zur Bestimmung der mittleren Länge des Instan-



$$\mathcal{E}(k_0) = 1, q_1 = 3/b, q_2 = 4/(3 \cdot b)$$

$$\mathcal{E}(k_1) \overset{(3.9)}{=} \mathcal{E}(k_0) \cdot \mathcal{E}(K_0) \overset{(3.10)}{=} \mathcal{E}(k_0) \cdot \ b \cdot q_1 = 3$$

$$\mathcal{E}(k_2) \overset{(3.9)}{=} \mathcal{E}(k_1) \cdot \mathcal{E}(K_1) \overset{(3.10)}{=} \mathcal{E}(k_1) \cdot \ b \cdot q_2 = 4$$

$$\mathcal{E}(\eta) \overset{(3.12)}{=} \textstyle\sum_{i=0}^{2} \mathcal{E}(k_i) = 8$$

Bild 3.4: Beispiel eines Reproduktionsprozesses zur Erläuterung der Gleichungen (3.8) – (3.12).

tiierungspfades ist zunächst nicht offensichtlich, da ERNEST–Netzwerke im allgemeinen keine Bäume sondern Graphen sind; beispielsweise weil ein *Konzept* Konkretisierung mehrerer abstrakter Konzepte sein kann. Werden jedoch, wie im vorigen Abschnitt eingeräumt, die durch Identifikationsbeschreibungen formulierbaren Instanzbindungen ausgeschlossen, so bilden die generierten *Instanzen* mit ihren zugehörigen Bestandteils– und Konkretisierungsbeziehungen einen Baum, dessen Knotenanzahl der Anzahl (vollständiger) Instanzen im Instantiierungspfad entspricht und mit Gleichung (3.12) abgeschätzt werden kann. Diese zeigt einerseits die bekannte Tatsache eines *exponentiellen* Zusammenhangs zwischen der Knotenanzahl und der Tiefe eines Baumes, verdeutlicht andererseits aber auch den *worst-case*–Charakter dieser Abhängigkeit, da der Einfluß der Überlebenswahrscheinlichkeiten zu einer lediglich linear wachsenden Knotenzahl führt, wenn auf jeder Ebene des Baumes $q \leq 1/b$ gilt.

Die Tiefe d_I und der Verzweigungsfaktor b_I des "Instanzenbaumes" können für ein gegebenes Analyseziel C_g aus dessen Lage in der Wissensbasis abgeschätzt werden. Dabei müssen nur die Konzepte des Teilnetzes, das alle von C_g aus über Bestandteils– und Konkretisierungskanten (inklusive ererbter) erreichbaren Konzepte enthält, berücksichtigt werden. Fassen wir die Bestandteile und Konkretisierungen in der Menge

$$\begin{aligned} \mathbf{TEIL} \ &= \ \{B \mid C_g \xrightarrow{kon} B\} \cup \{B \mid C_g \xrightarrow{bst} B\} \cup \\ &\quad \{B \mid C \xrightarrow{kon} B \wedge (C \xrightarrow{spez} C_g \vee C \xrightarrow{spez} \ldots \xrightarrow{spez} C_g)\} \cup \\ &\quad \{B \mid C \xrightarrow{bst} B \wedge (C \xrightarrow{spez} A \vee C \xrightarrow{spez} \ldots \xrightarrow{spez} C_g)\} \\ &= \ \mathbf{BST} \cup \mathbf{KON} \end{aligned} \tag{3.13}$$

zusammen, so ist der zu untersuchende Ausschnitt der Wissensbasis durch

$$netz(C_g) \;=\; \begin{cases} C_g & \text{falls} \quad \mathbf{TEIL} = \emptyset \\ C_g \cup \bigcup\limits_{A \in \mathbf{TEIL}} netz(A) & \text{sonst} \end{cases} \tag{3.14}$$

gegeben. Für die Konzepte "Fahrzeug" und "Jeep" des Beispielnetzwerkes aus Bild 2.5 sind die für die Instantiierung relevanten Teilnetze

$$\begin{aligned} netz(\text{Fahrzeug}) &= \{\ \text{Fahrzeug}, \text{Rad}, \text{Fenster}, \text{Karosserie}, \text{Kreis}, \text{Polygon}\ \} \\ \text{und} & \\ netz(\text{Jeep}) &= \{\ \text{Jeep}, \text{Rad}, \text{Fenster}, \text{Karosserie}, \text{Ueberrollbuegel}, \text{Abweiser}, \\ &\quad\ \ \text{Reserverad}, \text{Kreis}, \text{Polygon}, \text{Oval}\ \} \end{aligned}$$

in Bild 3.5 hervorgehoben. Die Länge des Instantiierungspfades von C_g entspricht jedoch

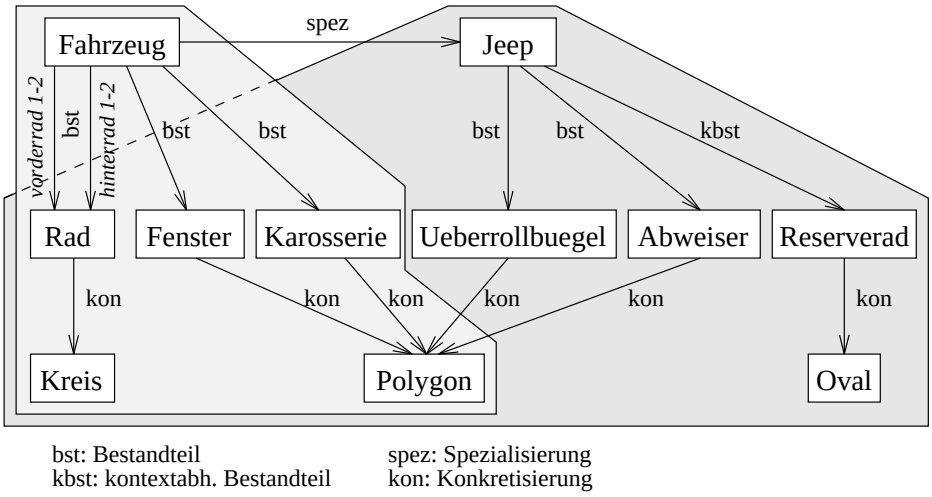


Bild 3.5: Teilnetze (Gleichung (3.14)) für die Instantiierung der Konzepte "Fahrzeug" und "Jeep" aus Bild 2.5.

nicht zwingend der Mächtigkeit von $netz(C_g)$, weil Instanzen zu Konzepten aus der Menge während der Analyse nicht (z.B. weil sie optional sind) oder aber mehrfach (z.B. bei minimaler Kantendimension $d_{min} > 1$) benötigt werden.

Da die Prämissen der Instantiierungsregeln bis auf die Ebene der minimalen Konzepte aufgelöst werden müssen, ist die Tiefe d_I abhängig von der Entfernung des Zielkonzeptes zu einem minimalen Konzept. In Netzwerken, die zusätzlich zur Spezialisierung entweder nur Konkretisierungen oder Bestandteile zulassen, entspricht diese Entfernung dem Konkretisierungs– bzw. Bestandteilsgrad (δ_{kon} bzw. δ_{bst}, siehe (2.2 – 2.3)) von C_g. In Wissensbasen, die die volle Netzwerksprache nutzen, muß die Rückwärtsverfolgung der

Regelprämissen über beide Kantentypen vorgenommen werden, so daß die Tiefe d_I des Instanzenbaumes dem *Teilegrad*

$$\delta_{teil}(C_g) \quad = \quad d_I = \begin{cases} 0 & \text{falls } \mathbf{TEIL} = \emptyset \\ \max\limits_{B \in \mathbf{TEIL}} \{\delta_{teil}\} + 1 & \text{sonst} \end{cases} \tag{3.15}$$

des Zielkonzeptes entspricht.

Für die Bestimmung des Verzweigungsfaktors b_I eines Konzeptes ist die Aufspaltung der Konzepte in $netz(C_g)$, die sowohl in den Modalitätsmengen H als auch in den Kantenbeschreibungen E modelliert wird, ausschlaggebend. Durch die daten– und problemabhängige Behandlung optionaler Teile ist es nicht möglich, allein aufgrund der Netzwerksyntax eine exakte Beziehung zwischen den zur Instantiierung notwendigen Subkonzepten von C_g und dem Verzweigungsfaktor im Instanzenbaum zu etablieren. Eine Beschränkung auf den obligatorischen Teil

$$H_{obl} \quad = \quad ((P_{cd} : T_E)^*, (P_{ci} : T_E)^*, (K : T_E)^*)_{obl} \tag{3.16}$$

der Modalitätsmengen ermöglicht es jedoch, die Elemente der Netzwerksprache zu ermitteln, die wesentlichen Einfluß auf die Kanten*anforderung* h eines Konzeptes besitzen. Aus den Kantenbeschreibungen, deren Rollen in H_{obl} auftreten, kann die Anzahl an Instanzen abgelesen werden, die zur Instantiierung eines Konzeptes unter Verwendung einer bestimmten Modalitätsmenge existieren müssen:

$$h \quad = \quad \sum_{\{E | T_E \in H_{obl}\}} d_{min}. \tag{3.17}$$

Dabei ist nur über die minimalen Kantendimensionen zu summieren, weil die restlichen $(d_{max} - d_{min})$ Bestandteile oder Konkretisierungen als optional aufgefaßt werden (vgl. S. 31). Die Maximierung über die Modalitätsmengen aller Konzepte in $netz(C_g)$ liefert einen modellbedingten Schätzwert für den Verzweigungsfaktor des Instanzenbaumes:

$$b_I \quad = \quad \max_{C \in netz(C_g)} \{\max_{H_C} \{h\}\}. \tag{3.18}$$

Identifizieren wir die Länge des Instantiierungspfades mit der Anzahl der Knoten im "Instanzenbaum", so erhalten wir durch Einsetzen von b_I und d_I in die Formel (3.12)

$$\mathcal{E}(|\mathcal{I}|) \quad = \quad \sum_{i=0}^{d_I} b_I^i \cdot \prod_{j=1}^{i} q_i \tag{3.19}$$

als Erwartungswert für die Länge des Instantiierungspfades, und können eine exponentielle Abhängigkeit von der Tiefe der Wissensbasis bzw. des vom Analyseziel aufgespannten Ausschnitts festhalten.

Für das Konzept "Jeep" aus Bild 2.5 sind die bislang eingeführten Kenngrößen der Wissensbasis im oberen Teil von Bild 3.6 auf Seite 59 eingetragen. Während wir die Tiefe $d_I = 2$ gemäß Gleichung (3.15) aus der Verfolgung eines Kantenzuges zu einem minimalen Konzept gewinnen, sind zur Bestimmung von b_I die Modalitätsmenge des Konzepts zu betrachten, und wir erhalten $b_I = 5$ aus der ersten Modalitätsmenge (vgl. Seite 40). Setzen wir $q_i = 1$, so liefert Gleichung (3.19) den Erwartungswert $\mathcal{E} = 31$ und ein Vergleich mit der tatsächlichen Länge der Instantiierungspfade $|\mathcal{I}_1| = 12$ und $|\mathcal{I}_2| = 10$ (vgl. Seite 41) verdeutlicht erneut den worst-case–Charakter der Abschätzung.

3.2.2 Komplexität der Graphsuche

Wie bereits in Abschnitt 2.3 erwähnt, ist der A^*–Algorithmus, der die übergeordnete Kontrollstrategie der ERNEST–Systemschale bildet, ein heuristisches Graphsuchverfahren, bei dem stets der bezüglich einer gewählten Kostenfunktion φ beste Knoten zur Expansion (Weiterverarbeitung) ausgewählt wird. Gegenüber anderen *Best-First*–Suchverfahren zeichnet sich der A^*-Algorithmus durch die Verwendung einer speziellen Kostenfunktion aus, welche die Kosten φ eines Knotens v_i gemäß

$$\varphi(v_i) \quad = \quad \psi(v_i) + \hat{\chi}(v_i) \tag{3.20}$$

aus den bis zu seiner Erzeugung angefallenen Kosten $\psi(v_i)$ und einem heuristischen Schätzwert $\hat{\chi}(v_i)$ für die Restkosten ermittelt. Um die Zulässigkeit des Verfahrens zu garantieren, muß $\hat{\chi}(v_i)$ *optimistisch* sein, darf also höchstens so groß sein wie die tatsächlichen Kosten $\chi(v_i)$, die auf dem optimalen Pfad von v_i zu einem Zielknoten anfallen.

Die Betrachtung zweier spezieller Restkostenschätzungen verdeutlicht deren wesentlichen Einfluß auf die Effizienz der Suche: während bei einer *perfekten* Schätzung ($\hat{\chi}(v_i) = \chi(v_i)$) nur die Knoten des Lösungspfades entwickelt werden, führt deren Verschwinden ($\hat{\chi}(v_i) = 0$) zu einer vollständigen Suche mit einem zur Suchbaumtiefe exponentiellem Aufwand. Diese Beobachtung motivierte zahlreiche Arbeiten (z.B. [Mar77, Gas79, Pea83]), welche die Komplexität des A^*-Algorithmus in Abhängigkeit von der Güte der Restkostenschätzung untersuchen. Eine umfassende Darstellung dieser Untersuchungen findet sich in [Pea84]; wir zitieren im folgenden die wesentlichen Ergebnisse und diskutieren anschließend die Konsequenzen für den ERNEST–Kontrollalgorithmus.

Mit Hilfe des im vorigen Abschnitt vorgestellten Modells eines inhomogenen Reproduktionsprozesses wird in [Huy80, Pea84] ein Erwartungswert $\mathcal{E}(Z)$ für die Anzahl der zu expandierenden Knoten berechnet. Unter der Annahme eines uniformen, b_S–stelligen Baumes mit einheitlichen Kantenkosten und genau einem Zielknoten in der Entfernung d_S von

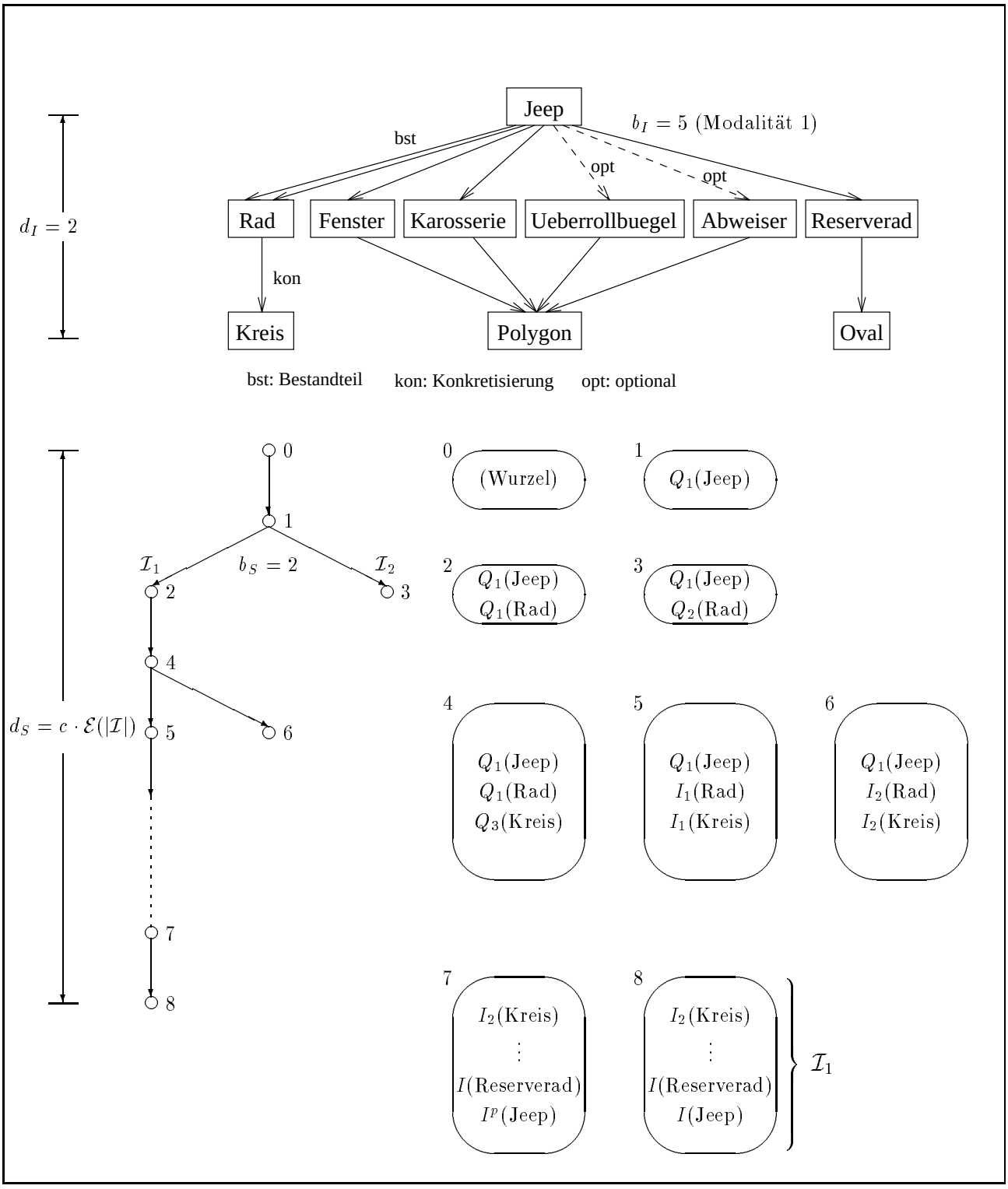


Bild 3.6: Beispielnetzwerk (oben) und schematische Darstellung des Suchbaums (unten) zur Erläuterung der Komplexitätsabschätzungen für Instantiierung und übergeordnete Graphsuche.

der Wurzel setzt sich der Erwartungswert

$$\mathcal{E}(Z) \;=\; d_S + \frac{b_S - 1}{b_S} \sum_{j=1}^{d_S} \sum_{d=1}^{j} (b_S^d \prod_{k=1}^{d} q_{j,k}) \tag{3.21}$$

aus der Anzahl der Knoten auf dem Lösungspfad (erster Summand) und einer Anzahl von Knoten (Doppelsumme) zusammen, die aufgrund einer ungenauen Restkostenschätzung expandiert werden. Bild 3.7 dient zur Erläuterung von Gleichung (3.21). Die $d_S + 1$ Kno-

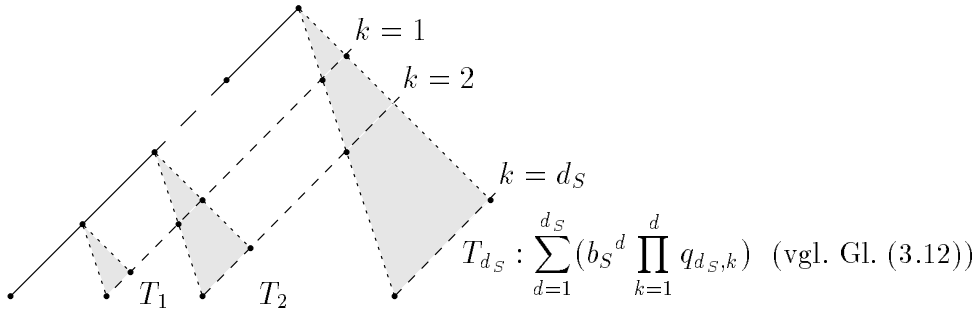


Bild 3.7: Erläuterung des Erwartungswerts (3.21) für die Anzahl der vom A^*–Algorithmus expandierten Knoten.

ten auf dem Lösungspfad, von denen der Zielknoten nicht expandiert werden muß, sind nach links eingezeichnet. Jeder dieser Knoten erzeugt einen der nach rechts eingetragenen Teilbäume T_j, $1 \leq j \leq d_S$, die den Verzweigungsfaktor b_S besitzen. Für jeden der d_S Teilbäume wird der Erwartungswert (3.12) gebildet, und die Erwartungswerte werden zur Anzahl der Knoten addiert, die abseits des Lösungspfades entwickelt werden. Die Faktoren $q_{j,k}$ geben an, mit welcher Wahrscheinlichkeit die Knoten auf der k-ten Ebene des Teilbaumes T_j überleben; sie sind — man betrachte die oben erwähnten Spezialfälle — von der Heuristik $\hat{\chi}$ abhängig. Da bei der Expansion jeder Teilbaumwurzel ein Knoten auf dem Lösungspfad erzeugt wird, muß gegenüber Gleichung (3.12) der Korrekturfaktor $(b_S - 1)/b_S$ eingeführt werden.

Um den Einfluß von $\hat{\chi}$ auf den Erwartungswertes $\mathcal{E}(Z)$ zu untersuchen, wird der durch

$$y(v_i) \;=\; \frac{\chi(v_i) - \hat{\chi}(v_i)}{\chi(v_i)} \tag{3.22}$$

definierte relative Fehler der Restkostenschätzung als unabhängige Zufallsvariablen aufgefaßt und durch eine stückweise lineare Verteilungsfunktion [Huy80, Pea83] beschrieben.

Damit wird

$$\mathcal{E}(Z) \quad = \quad \begin{cases} \mathcal{O}(\exp(d_S)) & \text{falls} \quad Pr(y(v_i)=0) < 1-1/b_S \\ \mathcal{O}(d_S^2) & \text{falls} \quad Pr(y(v_i)=0) = 1-1/b_S \\ \mathcal{O}(d_S) & \text{falls} \quad Pr(y(v_i)=0) > 1-1/b_S \end{cases} \tag{3.23}$$

als mittlere Komplexität des A^*–Algorithmus ermittelt. Die Analyse allgemeiner Fehlerverteilungen zeigt, daß ein exponentieller Suchaufwand für alle Heuristiken zu erwarten ist, deren Fehler stärker als logarithmisch mit der Entfernung des zu bewertenden Knoten zum Zielknoten anwächst.

Zur Abschätzung der Größe der vom ERNEST–Kontrollalgorithmus zu durchsuchenden Zustandsräume mit Hilfe von Gleichung (3.23) sind die vorausgesetzten Modellannahmen für jede konkrete Anwendung zu überprüfen (bzw. zu modifizieren). Aufgrund der allgemein festzustellenden Abhängigkeit zwischen dem Suchaufwand und der — von den Suchbaumparametern abhängigen — Fehlerwahrscheinlichkeit der Heuristik, ist es dennoch sinnvoll, den Verzweigungsfaktor und die Tiefe des Suchbaums bei gegebenem Zielkonzept aus einem Netzwerk abzuschätzen.

Zusätzlich zur konkurrierenden Instantiierung von Konzepten aufgrund mehrdeutiger Segmentierungsergebnisse führt die Auflösung verschiedener intensionaler Beschreibungen eines Begriffes zu alternativen Pfaden im Suchraum, dessen Struktur somit durch die Wissensbasis bereits im wesentlichen festgelegt ist. Der Verzweigungsfaktor b_S muß aus der Anzahl unterschiedlicher Modellierungen der Konzepte in $netz(C_g)$ bestimmt werden, zu der neben den Modalitätsmengen auch die in den Kantenbeschreibungen abgelegten alternativen Zielknoten beitragen. Mit der oben begründeten Beschränkung auf obligatorische Kanten erhalten wir

$$b_S \quad = \quad \max_{C \in netz(C_g)} \{ \sum_{H_C} (\max_{E | T_E \in H_{obl}} |g_E|) \} \tag{3.24}$$

und können diesen Wert verwenden, wenn b_S größer als die zugelassene Zahl konkurrierender Instanzen ist.

Da die Generierung der Instanzen des Instantiierungspfades an die Expansion der Suchbaumknoten gebunden ist, ist die Tiefe

$$d_S \quad = \quad c \cdot \mathcal{E}(|\mathcal{I}|) \tag{3.25}$$

des Suchbaums proportional zur Länge des Instantiierungspfades. Zusammen mit Gleichung (3.15) ergibt sich somit eine — im schlechtesten Fall exponentielle — Abhängigkeit der Suchbaumtiefe d_S von der Tiefe δ_{teil} der Wissensbasis. Der Faktor c, $1 \leq c \leq 2$, wird durch eine problemabhängige Strategie beeinflußt, welche die Anwendung der Inferenzregeln zur Generierung modifizierter Konzepte steuert.

Der hier hergestellte Zusammenhang zwischen den Suchbaumparametern (b_S, d_S) und den Kenngrößen der Wissensbasis (b_I, d_I) ist im unteren Teil von Bild 3.6 auf Seite 59 verdeutlicht. Aus der Betrachtung der Konzepte in *netz*(Jeep) erhalten wir mit Gleichung (3.24) $b_S = 2$, da das Konzept "Jeep" zwei Modalitätsmengen besitzt, und jede der darin als obligatorisch markierten Kanten (vgl. Seite 40) auf genau einen Zielknoten verweist ($|g_E| = 1$). Die aufgrund der unterschiedlichen Modalitätsmengen vorgenommene Aufspaltung des Suchbaums erfolgt bei der Expansion von Knoten 1; die Knoteninhalte der Nachfolger unterscheiden sich hier lediglich durch die modellgetrieben generierte Einschränkung für das Konzept "Rad". Nehmen wir an, daß Knoten 2 zur Weiterverfolgung ausgewählt wurde, so kann nach einer erneuten modellgetriebenen Modifikation der Wissensbasis (Knoten 4) das minimale Konzept "Kreis" instantiiert werden. In den Knoten 5 und 6 ist eine konkurrierende Instantiierung des Konzeptes angedeutet, bei der nicht mehr als zwei alternative Instanzen gebildet werden dürfen, wenn die Abschätzung des Verzweigungsfaktors (3.24) verwendet werden soll. Der hier skizzierte Instantiierungsprozeß wird nun in gleicher Weise für die restlichen Bestandteile des Zielkonzepts solange fortgesetzt, bis wir einen Zielknoten (Knoten 8) erhalten, der sämtliche Instanzen des Instantiierunsgpfades $\mathcal{I}_1$ (vgl. Seite 41) enthält. Die resultierende Suchbaumtiefe d_S aus Gleichung (3.25) ist im linken Teil von Bild 3.6 eingezeichnet.

3.2.3 Diskussion

In den beiden vorigen Abschnitten haben wir zunächst die Komplexität der Instantiierung untersucht und anschließend die wichtigsten Aussagen zur Komplexität des A^*–Algorithmus zusammengestellt. Für den Instantiierungsaufwand erhielten wir als ein *worst-case*-Ergebnis eine exponentielle Zunahme der Länge des Instantiierungspfades mit der Tiefe der Wissensbasis. Für die heuristische Suche konnten wir zeigen, daß neben den konkurrierenden Instantiierungen auch die alternativen Modellierungen in der Wissensbasis den Verzweigungsfaktor des Suchbaums beeinflussen und präzise Restkostenschätzungen erforderlich sind, um ein exponentielles Anwachsen der Suchbaumgröße zu verhindern.

Algorithmen, deren Rechenzeitbedarf exponentiell mit der Problemgröße ansteigt, sind auch auf Parallelrechnern nicht in polynomialer Zeit zu lösen, da hierzu eine — aus technologischen Gründen unrealistische — exponentiell wachsende Prozessoranzahl erforderlich wäre [Gar79]. Dies zeigt einerseits die Grenzen des Einsatzes von parallelen Rechensystemen und motiviert die in [Wah85] geäußerte, zunächst trivial anmutende Empfehlung, auf Parallelrechnern zu bearbeitende Probleme sollten auch auf herkömmlichen Monoprozessorsystemen in polynomialer Zeit lösbar sein. Andererseits erscheint der Einsatz von Parallelrechnern um so attraktiver, je größer der (sequentielle) Rechenzeitbedarf eines Problems ist und wir werden uns daher in Abschnitt 3.3 auf die Frage nach praktisch erzielbaren Erfolgen der Parallelisierung konzentrieren.

Kehren wir nochmals zur "terminologischen" Komplexität der Netzwerksprache zurück, so können wir vergleichend feststellen, daß ähnliche Resultate auch für die Subsumptions– und Klassifikationsalgorithmen vieler KL-ONE–basierter Sprachen berichtet werden [Sch83, Bra84, Neb88, Pro90]; einen zusammenfassenden Überblick gibt auch [Neb90, S. 232]. Es zeigt sich, daß ein polynomialer Aufwand nur noch für sehr einfache Sprachen (wie beispielsweise $\mathcal{FL}^-$ [Bra84]) zu beobachten ist, die Einschränkung komplexerer Formalismen jedoch leicht zu deren praktischer Unbrauchbarkeit führen kann. Als Konsequenz folgen wir daher der häufig vorgeschlagenen Methode, ein sehr selten auftretendes nicht–polynomiales *worst-case*–Verhalten zugunsten der Realisierung anspruchsvoller Anwendungen zu tolerieren [Neb90, Pat90].

Als wesentliches Ergebnis unserer Untersuchungen zur Komplexität der Zustandsraumsuche ist der durch die Gleichungen (3.23 – 3.25) hergestellte Zusammenhang zwischen der erforderlichen Präzision der Restkostenschätzung und den Kenngrößen einer Wissensbasis (hier: Tiefe und Anzahl alternativer Modellierungen) hervorzuheben. Heuristiken der geforderten logarithmischen Genauigkeit sind in vielen Bereichen der wissensbasierten Musteranalyse nur schwer zu finden; die Ursachen dafür reichen von numerischen Problemen bis hin zu allgemein unbefriedigend beantworteten Fragestellungen wie etwa der Kombination von Bewertungskalkülen verschiedener Wissensquellen (Abstraktionsebenen) oder der Bewertung unterschiedlicher Hypothesen.

Nicht nur im Hinblick auf die Entwicklung größerer Systeme, die neben der vorrangig angestrebten Verkürzung der Rechenzeit ein weiteres Ziel des Einsatzes von Parallelrechnern sein kann, muß dieser Mangel als Hauptproblem eingeschätzt werden. So zeigen die unter ERNEST realisierten Anwendungen mittlerer Größe (vgl. die Literaturangaben in Kapitel 2, S. 25) zwar ein akzeptables Speicher– und Laufzeitverhalten, als Preis dafür ist jedoch eine zunehmende Mächtigkeit des prozeduralen Wissens zur Beschränkung des Suchaufwandes festzustellen. Unter diesem Aspekt ist beispielsweise die Einführung der Inferenzregeln zur Generierung modifizierter Konzepte zu sehen; darüberhinaus sind auch komplexe problemabhängige Analyseprozeduren zu erwähnen. Vor dem Hintergrund des Entwurfs massiv–paralleler Verarbeitungsstrategien für wissensbasierte Systeme wird die steigende Bedeutung problemabhängiger Strategien bereits in [Fah79, Kap. 1] kritisiert. Inwieweit sich die hier geforderte Einfachheit der Inferenzmechanismen auf die Mächtigkeit der Kontrollstrategie auswirkt, wird bei den im folgenden Abschnitt beschriebenen Untersuchungen zur Parallelisierung des ERNEST–Kontrollalgorithmus zu prüfen sein.

3.3 Parallelisierungsmöglichkeiten

Die Beschreibung des Kontrollalgorithmus in Abschnitt 2.3 zeigt die Aktivierung der Analyseprozeduren, die sukzessive Anwendung der Inferenzregeln und die Zustandsraumsuche

als wesentliche Aktivität des Analyseprozesses. Es bietet sich daher an, zunächst drei Ebenen der Kontrolle — Konzept–, Netzwerk– und Suchbaumoperationen — zu unterscheiden, und diese getrennt auf Möglichkeiten zur Parallelverarbeitung zu untersuchen.

3.3.1 Paralleler Konzeptfluß

Auf der *Konzeptebene* bilden die im Aktionsteil der Inferenzregeln genannten Attributberechnungen und –bewertungen, die Relationstests sowie die Bewertung der Kanten und der erzeugten Objekte (Instanzen oder modifizierte Konzepte) die Elementaroperationen einer parallelen Instantiierung. Die konfliktfreie Generierung von Instanzen erfordert es, bestehende Datenabhängigkeiten zwischen den Analyseprozeduren eines Konzeptes zu berücksichtigen. Ergebnisse und Bewertungen aus Instanzen und modifizierten Konzepten von Bestandteilen, Konkretisierungen oder Kontexten können bei der Ermittlung dieser Abhängigkeiten unberücksichtigt bleiben, da ihre Existenz aufgrund der Instantiierungsreihenfolge vorausgesetzt werden darf.

Die Reihenfolge der Berechnungen ist nicht von den Zwischenergebnissen der Analyse abhängig und kann daher vorab in einer Analysevorbereitungsphase ermittelt werden. Der *Konzeptflußgraph* (vgl. [Kum92a]) hält die Argumentbeziehungen zwischen den Attributberechnungen eines Konzeptes fest. Nach der Vererbung der Attribute aus den allgemeineren Konzepten wird jeder Attributbeschreibung der Knoten eines gerichteten Graphen zugeordnet. Knoten werden verzeigert, wenn zwischen den zugehörigen Attributen eine — in der Attributwerteberechnung festgelegte — Argumentbeziehung besteht. Da zyklische Abhängigkeiten zwischend den Attributen eines Konzeptes untersagt sind, ist eine Einteilung des Graphen in Ebenen möglich, die den (frühstmöglichen) Berechnungszeitpunkt eines Knotens angeben. Beginnend mit den Knoten der untersten Ebene, welche Attribute repräsentieren, die nur auf die initiale Segmentierung oder ausschließlich auf Attribute aus anderen Konzepten zugreifen, können die Knoten einer Ebene parallel berechnet werden.

Die Bilder 3.8 und 3.9 verdeutlichen die Einordnung der Attributberechnung in den Konzeptflußgraph. Das Konzept "Quadrat" (vgl. Bild 3.8) besitzt die Attribute "flaeche", "umfang" und "formfaktor" sowie die Relationen "v-parallel" und "h-parallel". Die vier Bestandteile, von denen stellvertretend das Konzept "V1" abgebildet ist, sind identisch aufgebaut. Sie werden durch die Attribute "laenge" und "steigung" beschrieben, welche aus den gleichnamigen Attributen des allgemeineren Konzepts "Linie" durch geeignete Modifikationen hervorgehen.

Die Ebene 0 des Konzeptflußgraphen für das Konzept "Quadrat" (Bild 3.9) bilden die Attribute "flaeche" und "umfang", die unabhängig von den restlichen Attributen des Konzeptes sind, und zu deren Berechnung nur die bei der Instantiierung der Bestandteile ermittelten Werte von "laenge" benötigt werden. Der hier auftretende Zugriff auf Attributwerte eines Bestandteiles ist dadurch gekennzeichnet, daß als erster Teil des Arguments

```
KONZEPT: Quadrat

ATTRIBUT: umfang                        ATTRIBUT: formfaktor
  WERTEBERECHNUNG: umfangWB               WERTEBERECHNUNG: ffaktorWB
    ARGUMENT: vertikale1.laenge             ARGUMENT: umfang
              vertikale2.laenge                       flaeche
              horizontale1.laenge           BEWERTUNG: ffaktorBEW
              horizontale2.laenge
    INVERSE: umfangIWB
  RESTRIKTION: 8 - 12
  BEWERTUNG: umfangBEW

ATTRIBUT: flaeche
  WERTEBERECHNUNG: flaecheWB
    ARGUMENT: vertikale1.laenge
              horizontale1.laenge
  BEWERTUNG: flaecheBEW

STRUKTURRELATION: h-parallel            STRUKTURRELATION: v-parallel
  BEWERTUNG: h-parallelBEW                BEWERTUNG: v-parallelBEW
    ARGUMENT: horizontale1.steigung         ARGUMENT: vertikale1.steigung
              horizontale2.steigung                   vertikale2.steigung

BESTANDTEIL: vertikale1                 BESTANDTEIL: vertikale2
  ZIELKNOTEN: V1                          ZIELKNOTEN: V2
  BEWERTUNG: vertikaleBEW                 BEWERTUNG: vertikaleBEW

BESTANDTEIL: horizontale1               BESTANDTEIL: horizontale2
  ZIELKNOTEN: H1                          ZIELKNOTEN: H2
  BEWERTUNG: horizontaleBEW               BEWERTUNG: horizontaleBEW

BEWERTUNG: quadratBEW
  ARGUMENT: formfaktor, h-parallel, v-parallel



KONZEPT: V1
SPEZIALISIERUNG_VON: Linie
ATTRIBUT: laenge                        ATTRIBUT: steigung
  MODIFIZIERT: JA                         MODIFIZIERT: JA
  WERTEBERECHNUNG: laengeWB               WERTEBERECHNUNG: steigungWB
    ARGUMENT:                               ARGUMENT:
  RESTRIKTION: >1                         RESTRIKTION: 85 - 90
  BEWERTUNG: laengeBEW                    BEWERTUNG: steigungBEW

BEWERTUNG: linieBEW
  ARGUMENT: laenge
```

Bild 3.8: Das Konzept "Quadrat"

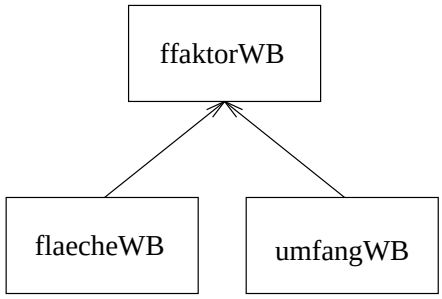


Bild 3.9: Konzeptfluß für das Konzept "Quadrat".

die Rolle der entsprechenden Kante angegeben wird (vergleiche auch Bild 4.4, Seite 89). Die Werte von "flaeche" und "umfang" dienen als Eingangsgrößen für die Berechnung von "formfaktor" auf der nächsten Ebene.

Eine naheliegende Erweiterung des Konzeptflusses besteht darin, die notwendigen Bewertungen der Kanten, Attribute, Relationen und des Konzeptes — abweichend von der im Aktionsteil von Regel 1 festgelegten Reihenfolge — in den Datenfluß zu integrieren. Dazu sind zunächst die gegenseitigen Abhängigkeiten der Argumente *aller* Analyseprozeduren eines Konzeptes festzustellen, und die zusätzlich generierten Flußgraphknoten werden anschließend analog zum oben beschriebenen Vorgehen verzeigert. In Abschnitt 4.2 wird gezeigt, wie der hier vorgeschlagene Ansatz, der sich am Datenfluß zwischen den Elementareinheiten der Wissensbasis orientiert, auf sämtliche Konzepte einer Wissensbasis ausgedehnt werden kann, um eine massiv–parallele Verarbeitungsstrategie zu etablieren. Bild 3.10 zeigt den *erweiterten* Konzeptflußgraphen des Konzepts "Quadrat", in welchem die Berechnung und Bewertung von Attributen zugunsten einer adäquaten Repräsentation in einem Knoten zusammengefaßt wurden.

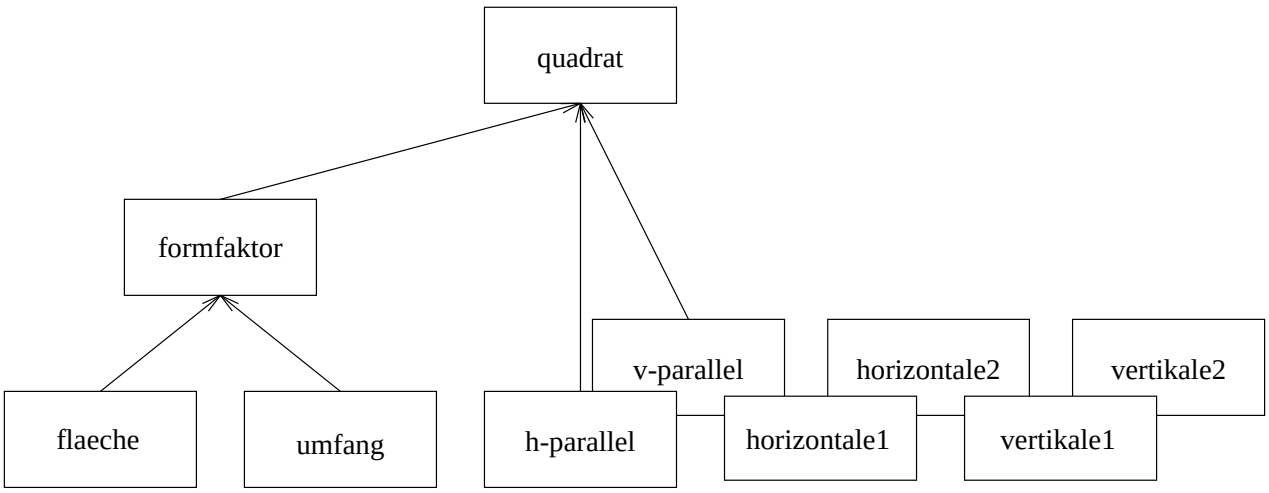


Bild 3.10: Erweiterter Konzeptfluß für das Konzept "Quadrat".

Verglichen mit der parallelen Berechnung und Bewertung von Attributen, Relationen und Kanten in der durch die Inferenzregeln festgelegten Reihenfolge offeriert der erweiterte Konzeptfluß die größtmögliche Parallelität bei der Instantiierung eines einzelnen Konzepts. Daraus resultiert im allgemeinen ein Geschwindigkeitsgewinn, der aufgrund der unregelmäßigen Struktur des Datenflußgraphen jedoch häufig mit einer geringen Effizienz verbunden ist. Es bietet sich daher an, netzwerkweite Parallelisierungsmöglichkeiten zu untersuchen, um auf freien Prozessoren Berechnungen aus anderen Konzepten auszuführen. Diesem Thema widmet sich der nächste Abschnitt.

3.3.2 Paralleler Netzfluß

Auf der *Netzwerkebene* untersuchen wir die parallele Instantiierung von Konzepten einschließlich der als Voraussetzung dafür erforderlichen Berechnung der Bestandteile und Konkretisierungen bis hin zu den minimalen Konzepten; der atomare Verarbeitungsschritt ist hier die Berechnung einer einzelnen Instanz oder eines modifizierten Konzeptes.

Analog zur Bestimmung des Konzeptflusses ist auch die Ermittlung der Abhängigkeiten der Konzepte Gegenstand der Analysevorbereitung in ERNEST, da die Generierung des *Netzflußgraphen* eine schnellere Überprüfung der Anwendbarkeit von Inferenzregeln zur Analysezeit ermöglicht. Dazu wird jedem Konzept ein Knoten mit den verschiedenen Instantiierungsmöglichkeiten, den *Netzflußprämissen*, zugeordnet. Diese werden aus je einer Modalitätsmenge sowie einem eventuellen Kontext des Konzeptes gebildet und berücksichtigen auch die Einschränkungen, die sich aus den problemabhängigen Instanzbindungen (durch Identifikationen) ergeben; letzteres ist ausführlich in [Kum92a, S. 74 – 77] beschrieben.

Bild 3.11 illustriert den Netzfluß an einem einfachen Beispiel, das im oberen Teil ein "Haus" und zwei mögliche Ansichten zeigt, die durch Modalitätsbeschreibungen modelliert werden. Die obligatorischen Bestandteile beider Modalitätsmengen sind in der Darstellung des Netzwerks (unten links) durch verschieden strichlierte Kästen zusammengefaßt. Der Netzflußknoten (unten rechts) enthält zwei Prämissen und läßt mehrere Möglichkeiten der Parallelverarbeitung zu: die gleichzeitige Instantiierung der Konzepte einer Prämisse — beispielsweise von "Dach", "Mauer" und "Kamin" und "Fenster", die obligatorische Bestandteile der Modalitätsmenge zu Prämisse 1 sind — und die parallele Behandlung konkurrierender Netzflußprämissen — ist "Haus" durch ("Dach", "Mauer", "Kamin" und "Fenster") oder ("Mauer", "Tür", "Kamin" und "Dach") zu instantiieren ?

Bereits in Abschnitt 3.2.2 wurde angesprochen, daß die Modalitätsmengen eines Konzeptes während der Analyse durch die Aufspaltung des Suchbaumes aufgelöst werden. Da für die Objekte eines Suchbaumknotens zudem ein eindeutiger Kontext gültig ist, liegt den Instanzen und modifizierten Konzepten eines konkreten Analysezustandes stets genau eine Netzflußprämisse zugrunde (vgl. Bild 3.12). Die gleichzeitige Verfolgung konkurrierender

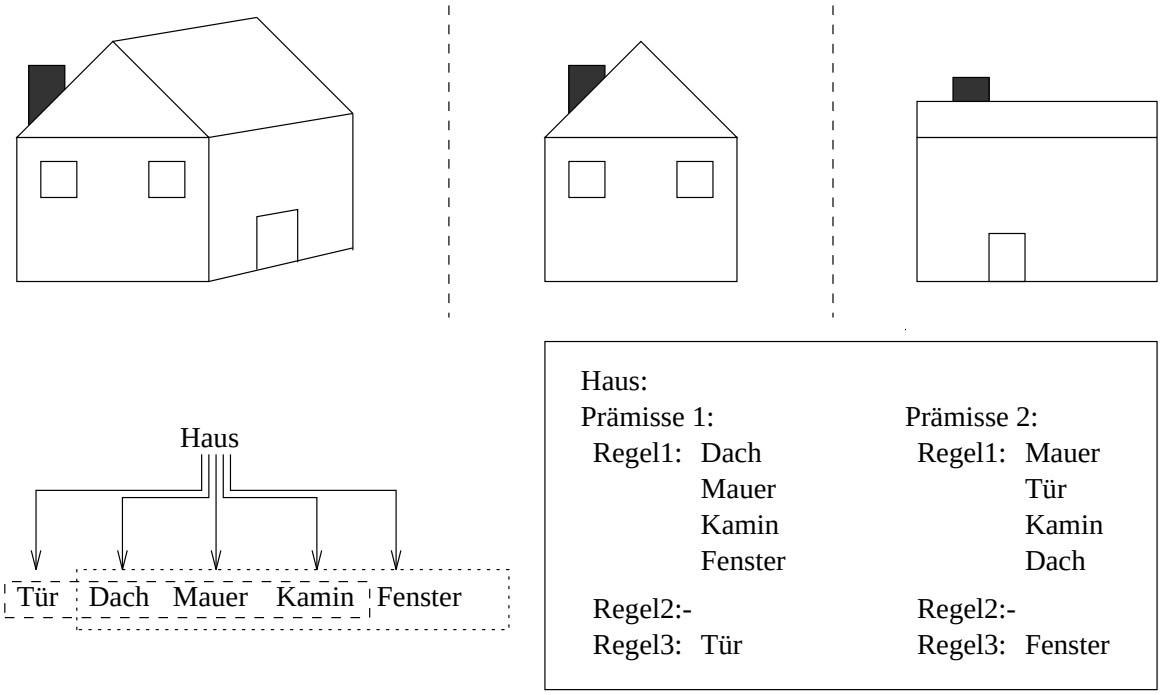


Bild 3.11: Graphische Darstellung des Konzeptes "Haus" und zweier Ansichten (oben) sowie das zugehörige Netzwerk und der Netzflußknoten des Konzeptes.

Prämissen kann somit auf eine parallele Suche im Zustandsraum zurückgeführt werden, die im nächsten Abschnitt untersucht wird.

Die parallele Bearbeitung der Konzepte einer Prämisse ermöglicht die gleichzeitige Instantiierung von obligatorischen und optionalen Bestandteilen und Konkretisierungen, die im Netzflußknoten in Bild 3.11 unter `Regel 1` bzw. `Regel 3` referiert werden. Zu beach-

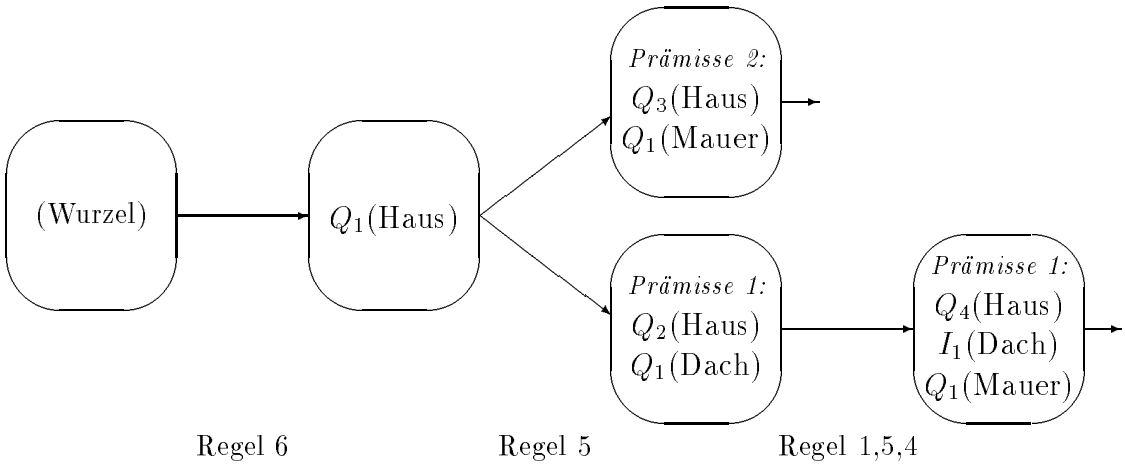
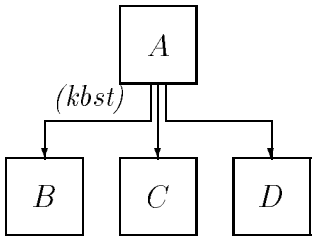


Bild 3.12: Behandlung konkurrierender Netzflußprämissen durch Aufspaltung des Suchbaums.

ten ist, daß das Auftreten kontextabhängiger Bezüge, deren Auflösung durch die Bildung partieller Instanzen erfolgt, den Parallelitätsgrad einschränkt. So ist in Bild 3.13 die gleichzeitige Instantiierung von B, C und D mit drei Prozessoren nur möglich, wenn keines der Konzepte kontextabhängiges Bestandteil von A ist. Stellt dagegen A einen Kontext zu B dar, so können nur zwei Prozessoren zur parallelen Erfüllung der Prämisse von A verwendet werden, da zunächst eine partielle Instanz $I^p(A)$ zu generieren ist. Mit dieser müssen in zwei weiteren sequentiellen Schritten zunächst $I(B)$ und anschließend $I(A)$ erzeugt werden.



t	Proz.1	Proz.2	Proz.3
1.	$I(B)$	$I(C)$	$I(D)$
2.	$I(A)$	–	–

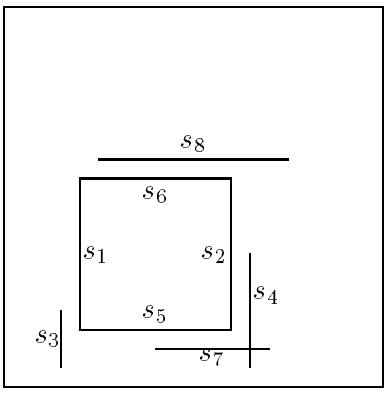
t	Proz.1	Proz.2	Proz.3
1.	$I(C)$	$I(D)$	–
2.	$I^p(A)$	–	–
3.	$I(B)$	–	–
4.	$I(A)$	–	–

Bild 3.13: Parallele Instantiierung einer Netzflußprämisse ohne (mitte) und mit kontextabhängigen Bestandteilen (rechts).

Die bis hierher beschriebenen Untersuchungen zur Parallelisierung der Netzwerkoperationen umfassen nur die rein *datengetriebene* Erzeugung von Instanzen durch die Regeln 1 bis 3. Sie mögen sich als ausreichend erweisen, wenn eine annähernd ideale Segmentierung vorliegt, welche die kombinatorische Vielfalt potentieller Zuordnungen zwischen Konzepten und Sensordaten stark einschränkt. Die weitaus realistischere Annahme einer imperfekten (fehlerbehafteten) initialen Beschreibung macht es jedoch erforderlich, auch die Generierung modifizierter Konzepte, durch die eine erwartungsgesteuerte Analyse etabliert wird, zu berücksichtigen. Dazu nehmen wir anhand der schematischen Darstellung realer Segmentierungsergebnisse in Bild 3.14 und des Netzwerkausschnittes 3.8 eine exemplarische Gegenüberstellung der Analyseschritte und Zwischenergebnisse einer datengetriebenen und einer erwartungsgesteuerten Instantiierung vor.

Die datengetriebene Analyse verwendet im ersten Schritt die Liniensegmente s_1,s_2 und s_4 zur Erzeugung von drei Instanzen des Konzepts "V1"; da die restlichen Segmente s_3 und $s_5, \ldots, s_8$ die Restriktionen der Attribute "laenge" bzw. "steigung" aus Bild 3.8 nicht erfüllen, werden für diese keine weiteren Instanzen generiert. Nachdem in den Schritten 2 – 4 die restlichen Bestandteile in analoger Vorgehensweise instantiiert wurden, kann das Konzept "Quadrat" im fünften Schritt instantiiert werden: mit je drei Instanzen für die Vertikalen bzw. Horizontalen müssen $3^2 \cdot 2^2 = 36$ Instanzen generiert und bewertet werden.

Im Falle einer erwartungsgesteuerten Analyse werden die Restriktionen und die inverse Attributwerteberechnung des Attributs "umfang" zur Propagierung von Einschränkungen genutzt. Kann "Quadrat" mit Regel 6 als Zielkonzept etabliert werden, so wird da-



datengetriebene Analyse:

1. $I_{1,\ldots,3}$(V1): s_1, s_2, s_4
2. $I_{1,\ldots,3}$(V2): s_1, s_2, s_4
3. $I_{1,\ldots,3}$(H1): s_5, s_6, s_8
4. $I_{1,\ldots,3}$(H2): s_5, s_6, s_8
5. I_1(Quadrat): $s_1 s_2 s_5 s_6$

$\vdots$

I_{36}(Quadrat): $s_4 s_2 s_8 s_6$

erwartungsgesteuerte Analyse:

1. Q_1(Quadrat), 2. Q_1(V1)
3. $I_{1,\ldots,2}$(V1): s_1, s_2
4. Q_2(Quadrat), 5. Q_1(V2)
6. $I_{1,\ldots,2}$(V2): s_1, s_2
7. Q_3(Quadrat), 8. Q_1(H1)
9. $I_{1,\ldots,2}$(H1): s_5, s_6
10. Q_4(Quadrat), 11. Q_1(H2)
12. $I_{1,\ldots,2}$(H2): s_5, s_6
13. I_1(Quadrat): $s_1 s_2 s_5 s_6$

$\vdots$

I_4(Quadrat): $s_2 s_1 s_6 s_5$

Bild 3.14: Ablauf der Instantiierung bei daten– und erwartungsgesteuerter Analysestrategie.

bei zunächst ein modifiziertes Konzept Q_1(Quadrat) generiert, in welches die Restriktion $B_{\text{umfang}} = [8, 12]$ eingetragen wird. Die nun mögliche Anwendung von Regel 5 sorgt für die *modellgetriebene* Modifikation von "V1", wobei — man vergleiche Seite 39 — die inversen Attributberechnungen der Attribute aus "Quadrat" ausgeführt werden, sofern diese ein Attribut aus "V1" als Argument referieren. Im vorliegenden Beispiel gilt dies offensichtlich für "umfang", und durch die Inverse $\varphi^{-1}_{\text{umfang}}$ kann eine Restriktion für die Seitenlänge berechnet werden.

Eine geeignete Definition der inversen Attributberechnung, die auch verdeutlicht, daß es sich nicht zwangsläufig um die Umkehrfunktion im mathematischen Sinne handeln muß, ist in diesem Beispiel durch

$$\begin{aligned} \varphi^{-1}_{\text{umfang}} &: B_{\text{umfang}} \mapsto B_{\text{laenge}} \\ \varphi^{-1}_{\text{umfang}}([a,b]) &= [a/4, b/4] \end{aligned}$$

gegeben. Nachdem der Wertebereich für "laenge" durch die Anwendung der Inversen auf das Intervall $[2,3]$ eingeschränkt wurde, wird "V1" durch die Aktivierung von Regel 1 im dritten Schritt instantiiert. Aufgrund der neu berechneten Restriktion in Q_1(V1) werden die Instanzen I_1(V1) und I_2(V1) zu den Segmenten s_1 und s_2 berechnet; für beide liefert die Attributberechnung von "laenge" den Wert 2. Die nun folgende *datengetriebene* Modifikation von "Quadrat" mittels Regel 4 ermöglicht es, eine engere Restriktion $B_{\text{umfang}} = [8, 8]$ zu berechnen, und das neu generierte modifizierte Konzept Q_2(Quadrat) wird in Schritt 5 zur erneut modellgetriebenen Modifikation von "V2" verwendet. In den Schritten 7 – 12 wiederholt sich das Vorgehen für die restlichen Bestandteilen, nach deren Instantiierung

schließlich nur noch vier Instanzen des Zielkonzeptes gebildet werden können.

Da die Behandlung der Subkonzepte bei einer strikt datengetriebenen Analyse unabhängig voneinander, d.h. ohne die Ausnutzung bereits berechneter Zwischenergebnisse erfolgt, sind die Schritte 1 – 4 in diesem Falle parallel ausführbar. Im Gegensatz dazu führt die erwartungsgesteuerte Abarbeitung der Netzflußprämisse zu einem rein sequentiellen Datenfluß, wodurch ein *Tradeoff* zwischen der Mächtigkeit der Kontrollstrategie und den Parallelisierungsmöglichkeiten deutlich wird: soll beispielsweise die bei der Instantiierung von "V1" berechnete Länge der Liniensegmente zur Einschränkung möglicher Segmentkandidaten für "V2" verwendet werden, so kann "V2" nicht gleichzeitig mit (unabhängig von) "V1" instantiiert werden. Da jedoch die Beachtung sukzessiv ermittelter Restriktionen dazu führt, daß die erwartungsgesteuerte Analyse deutlich weniger Objekte erzeugt, ist keine der beiden Strategien a priori zu bevorzugen; als entscheidende Einflußgrößen stellen sich hier problemabhängige Kriterien heraus wie beispielsweise die Qualität der Segmentierung und die Wirksamkeit der Modifikation von Konzepten zur Begrenzung potentieller Instantiierungen.

Auf der Suchbaumebene schlägt sich die vermehrte Generierung konkurrierender Objekte in zusätzlichen alternativen Pfaden nieder, weshalb die Parallelisierung des Netzflusses keinesfalls isoliert von der übergeordneten Suche betrachtet werden darf. Sie erweist sich vielmehr nur dann als sinnvoll, wenn entweder sehr genaue Kostenfunktionen zur Verfügung stehen, oder der erhöhte Suchaufwand durch eine effiziente Parallelisierung des A^*–Algorithmus zumindest kompensiert werden kann. Dieser Zusammenhang motiviert die im folgenden Abschnitt beschriebenen Untersuchungen zur parallelen Suche.

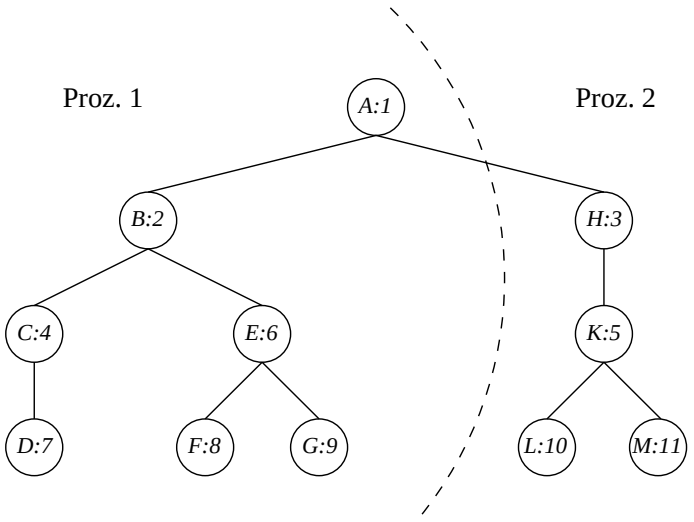
3.3.3 Parallele Graphsuche

Kombinatorische Suchverfahren werden in vielen Bereichen der Ingenieurswissenschaften, des Operation Research oder der Künstlichen Intelligenz zur Lösung unterschiedlichster Optimierungsprobleme eingesetzt. Aufgrund ihres hohen Rechenzeitbedarfs — viele praktisch relevante Aufgabenstellungen sind *NP–schwer* [Gar79, Pap94] — ist die Parallelisierung derartiger Verfahren Thema zahlreicher Arbeiten [Mon87, Ros88, Rou89, Höp90], die auch zur Entwicklung spezieller Hardwareplattformen geführt haben (z.B. [Wah84]). Eine Diskussion der wichtigsten Aspekte bei der Parallelisierung von Suchverfahren unterschiedlicher Klassen (UND–Bäume, ODER–Bäume sowie UND/ODER–Bäume) und eine ausführliche Bibliographie finden sich in [Wah90]); wir beschränken uns daher auf eine Skizzierung der wichtigsten Methoden und die Beschreibung erster praktischer Erfahrungen bei der Parallelisierung der A^*–Suche aus Bild 2.6. Hierbei besteht die Aufgabe einer parallelen Suche in der gleichzeitigen Verfolgung daten– oder modellbedingter konkurrierender Analysezustände, die als bewertete Knoten in der OPEN–Liste vorliegen.

In [Höp90] werden zwei Ansätze zur Parallelisierung kombinatorischer Suchverfahren

vorgestellt und am Beispiel des bekannten 3×3–Verschiebepuzzle evaluiert. Bild 3.15 zeigt zunächst das *Prinzip der Zustandsraumzerlegung*, bei der jeder Prozessor einen Teil des Suchraumes unabhängig von allen anderen Prozessoren durchsucht. Vorteile des Verfahrens liegen in seinem günstigen Kommunikationsverhalten, da nur Zustände, die außerhalb des eigenen Arbeitsbereiches liegen, von einem Rechnerknoten an einen anderen übergeben werden müssen. Nachteilig wirkt sich im allgemeinen aus, daß nur wenige Prozessoren zur Entwicklung des Lösungspfades beitragen, auch wenn durch die *lokale* Suche innerhalb einer Partition bei bestimmten Eigenschaften der Kostenfunktion superlineare Speedups zu beobachten sind (vgl. auch [Wah90]).

Im ERNEST–Kontrollalgorithmus sind die als Ergebnisse der Zielkonzeptschätzung vorliegenden konkurrierenden Analysezustände natürliche Kandidaten für die Bearbeitung durch einen eigenen Prozessor, da bei einer statischen Zuordnung zusätzliche Berechnungen der Partitionszugehörigkeit entfallen können. Allerdings wird der Lösungspfad in diesem Fall nur von einem Prozessor ermittelt, während die übrigen in Teilbäumen suchen, die nur bei einer geringen Penetranz der (sequentiellen) Suche bearbeitet würden.

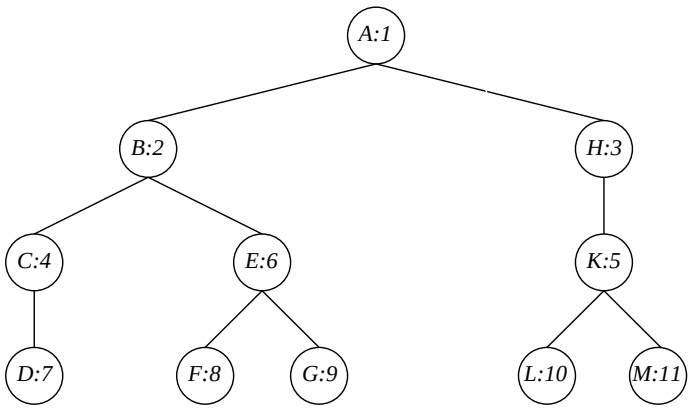


Iter.	seq.	ZR-Zerlegung Proz. 1	Proz. 2
1.	A	A	–
2.	B	B	H
3.	H	C	K
4.	C	E	L
5.	K	D	M
6.	E		
7.	D		

Bild 3.15: Parallele Suche durch Zustandsraumzerlegung. In den Knoten des zu durchsuchenden Graphen (links) sind Kosten gemäß Gleichung (3.20) eingetragen. Tabellarisch gegenübergestellt sind die sequentielle Entwicklung des Zustandsraumes und die Entwicklung durch zwei Prozessoren bei der gestrichelt eingezeichneten Zerlegung (rechts).

Bei der *Parallelisierung durch Aufgabenverteilung*, vergleiche Bild 3.16, werden die in der OPEN–Liste vorhandenen Knoten als alternativ zu lösende Teilaufgaben betrachtet. Die p ersten Knoten der Liste werden auf die p zur Verfügung stehenden Prozessoren verteilt und dort expandiert; die erzeugten Nachfolger werden wiederum in die OPEN–Liste eingeordnet. Da somit in jedem Iterationsschritt die *global besten* Knoten expandiert

werden, trägt im allgemeinen jeder Prozessor zur Entwicklung des Lösungspfades bei und für gut informierte Suchverfahren ist bereits bei wenigen Prozessoren ein gesättigter, nahezu linearer Speedup zu beobachten [Höp90].



Iter.	seq.	Aufgabenverteilung Proz. 1	Proz. 2
1.	A	A	–
2.	B	B	H
3.	H	C	K
4.	C	E	D
5.	K		
6.	E		
7.	D		

Bild 3.16: Parallele Suche durch Aufgabenverteilung. Tabellarisch gegenübergestellt sind die sequentielle Entwicklung des Zustandsraumes und die Entwicklung bei Entnahme von je zwei Knoten aus der OPEN–Liste.

Der Schwachpunkt der Aufgabenverteilung liegt in dem relativ hohen Kommunikations– und Synchronisationsaufwand des Verfahrens. Dieser ist zum einen erforderlich, um die erzeugten Knoten nach jeder Iteration in die zentral gehaltene OPEN–Liste einzusortieren; zum anderen entsteht bei Suchproblemen, die eine allgemeine Graphstruktur besitzen, zusätzlicher Verwaltungsaufwand zur Vermeidung der mehrfachen Expansion eines Knotens. Abhilfe bietet hier eine Variante der Aufgabenverteilung die eine *dezentrale*, d.h. prozessorlokale Verwaltung der ausgeführten Expansionen vorsieht und somit die mögliche Mehrfachexpansion von Knoten zugunsten eines deutlich besseren Kommunikationsverhaltens in Kauf nimmt.

Eine prototypische Realisierung der Aufgabenverteilung für den ERNEST–Kontrollalgorithmus ist in [Hua93] beschrieben. Auf der Basis eines *Client/Server–Modells* kommunizieren mehrere, in einem lokalen Netzwerk gekoppelte Workstations unter Verwendung des *RPC–Mechanismus* (*Remote Procedure Call*, z.B. [Sch92a, Sch92b]) und des *XDR–Standards* (*eXternal Data Representation*, z.B. [Cor91]) zum Datenaustausch zwischen Client und Servern. Letzterer beschränkt sich auf die Übergabe von Suchbaumknoten, die von den Servern zu expandieren sind, und auf die Rückgabe der erzeugten Nachfolgerknoten an den Client. Durch die Baumstruktur des Suchraumes müssen keine weiteren Vorkehrungen zur Vermeidung von Mehrfachexpansionen getroffen werden.

Um aufwendigere Implementierungsarbeiten zunächst zu vermeiden, wurden erste Un-

tersuchungen der Möglichkeiten und Grenzen des Ansatzes an einem (synthetischen) Beispielnetzwerk durchgeführt. Die Analyse unsicherer Eingangsdaten wurde dabei durch die systematische Variation des relativen Fehlers y der Restkostenschätzung (vgl. Gl. 3.22) simuliert.

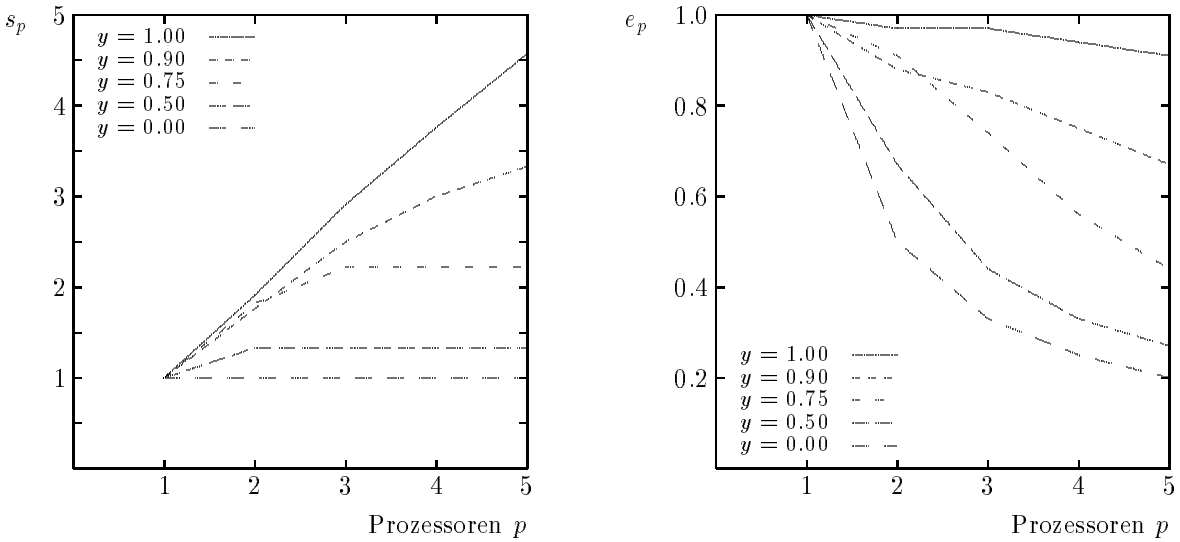


Bild 3.17: Parallelisierung des A^*-Algorithmus: Speedup und Effizienz für die Anzahl durchgeführter Iterationen.

Die Bilder 3.17 und 3.18 zeigen Speedup und Effizienz für die Anzahl der durchgeführten Iterationsschritte und die gemessenen CPU-Zeiten bei der Verwendung von bis zu fünf Servern. Der für unterschiedliche Restkostenschätzungen erzielte Parallelisierungserfolg bei den benötigten Iterationen soll hier nur kurz diskutiert werden, da diese Versuchsreihen lediglich als Vergleichsgrundlage für die Auswertung der Aufgabenverteilung bezüglich der benötigten CPU-Zeiten konzipiert waren. Die erzielten Ergebnisse liegen für alle Variationen der Restkostenschätzung innerhalb der in [Wah90, S. 118] angegebenen Grenzen für die Iterationsschritte einer parallelen *Best-First*–Suche. Desweiteren zeigt sich das bekannte Phänomen, daß Speedup und Effizienz mit zunehmender Güte der Schätzfunktion sinken; bei einer perfekt informierten Schätzung ($y = 0$) ist eine Parallelisierung schließlich überflüssig.

Grenzen und Anforderungen der Aufgabenverteilung im ERNEST–Kontrollalgorithmus werden bei einer näheren Betrachtung der Ergebnisse der Zeitmessungen in Bild 3.18 sichtbar. Hier zeigen sich halbwegs akzeptable Werte für Speedup und Effizienz nur noch für wenig oder gar nicht informierte Restkostenschätzungen und bei geringer Prozessoranzahl. Die Abweichungen von den idealen Werten einer Versuchsreihe — diese entsprechen bei einer verlustfreien Kommunikation den Ergebnissen in Bild 3.17 — betragen im Mittel 35 Prozent und sind auf Synchronisationsverluste und den hohen Kommunikationsauf-

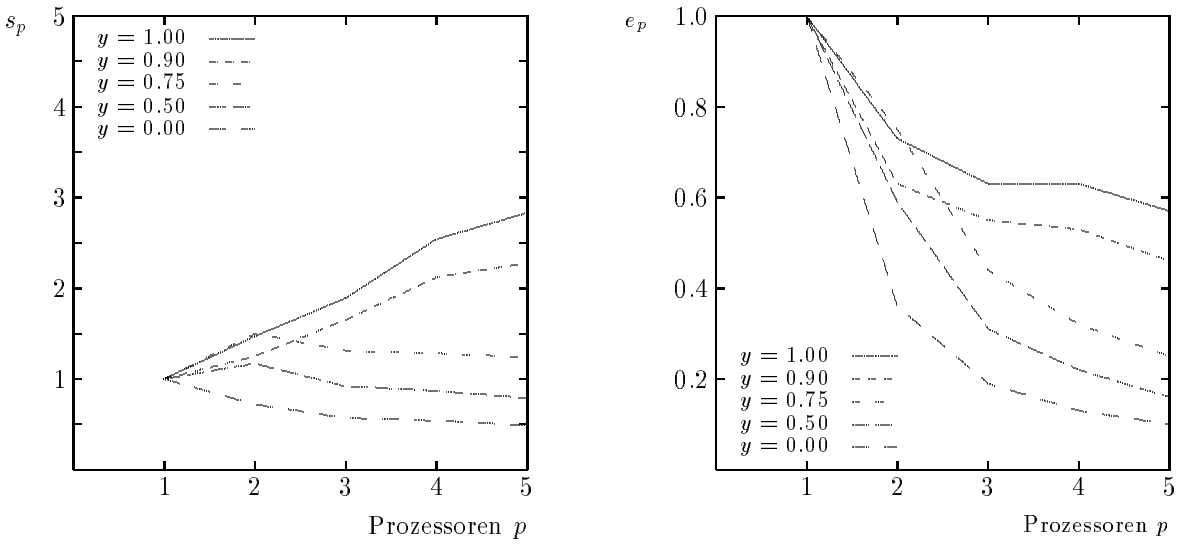


Bild 3.18: Parallelisierung des A^*-Algorithmus: Speedup und Effizienz für die gemessenen CPU-Zeiten.

wand zurückzuführen. Während sich erstere in den durchgeführten Experimenten als vernachlässigbar erweisen, ist bereits bei kleinen Netzwerken ein nahezu gleicher Zeitbedarf für Knotentransfer und –expansionen zu beobachten. Gründe dafür sind in der groben Granularität der Parallelisierung sowie in der Art des auszutauschenden Datenvolumens zu sehen. So müssen in einem Iterationsschritt mehrere Analysezustände (ein ausgewählter Knoten der OPEN–Liste und seine Nachfolger) zwischen Client und jedem Server ausgetauscht werden. Zusätzlich zu sämtlichen Instanzen und modifizierten Konzepten der Knoten ist dabei auch der Austausch von Kontrollinformation zur Rekonstruktion ihrer internen Struktur (des Modellausschnittes) erforderlich. Da die Definition des Remote Procedure Calls einen "schmalen Kanal zur Parameterübergabe" vorsieht [Sch92a], können wir aus den berichteten Ergebnissen auch auf die Notwendigkeit eines leistungsfähigeren Kommunikationsmechanismus schließen, dessen Auslegung bzw. Auswahl jedoch durch das dynamische Anwachsen der Suchbaumknoten um weitere Objekte erschwert wird. In Kapitel 4 werden wir darstellen, wie diese Problematik durch eine für die Parallelverarbeitung geeignetere Definition des Analysezustandes umgangen werden kann.

3.4 Zusammenfassung

Ausgehend von einer Teilmenge der in Kapitel 2 vorgestellten Netzwerksprache wurden im vorliegenden Kapitel Möglichkeiten zur Parallelisierung des ERNEST–Kontrollalgorithmus untersucht. Neben den epistemologisch primitiven Elementen (Konzepte, Kanten, Attribute, Relationen) wurden als zusätzliches Repräsentationsmittel lediglich Modalitätsmengen

zugelassen, da diese die Handhabbarkeit größerer Wissensbasen erheblich erleichtern.

Die zunächst vorgenommene, an der Netzwerksyntax orientierte formale Erfassung der Sprachkonstrukte ist durch zwei Zielsetzungen motiviert: sie soll zum einen die Isolation von Elementaroperationen für das im nächsten Kapitel vorzustellende massiv–parallele Instantiierungsverfahren erleichtern; zum anderen dient sie als Hilfsmittel bei Untersuchungen zur Komplexität des ERNEST–Kontrollalgorithmus. Um die Erfolgschancen eines parallelen Kontrollalgorithmus zu erörtern, wurden letztere den Ausführungen zur Parallelisierung vorangestellt und sowohl auf der Ebene der Netzwerksprache geführt als auch auf die verwendete Graphsuche ausgedehnt.

Für den Instantiierungsaufwand eines gegebenen Zielkonzeptes zeigt sich im Rahmen einer *worst-case*–Analyse eine exponentielle Abhängigkeit von dessen Entfernung zu einem minimalen Konzept. Die Untersuchungen zur Komplexität der Graphsuche stellen einen Zusammenhang zwischen Tiefe und Verzweigungsfaktoren von Wissensbasis und Suchbaum her und verdeutlichen die Notwendigkeit sehr präziser Restkostenschätzungen zur Verhinderung eines exponentiell wachsenden Suchaufwandes.

Die sich anschließenden Überlegungen und Experimente zur Parallelverarbeitung auf Konzept–, Netzwerk– und Suchraumebene werden durch die praktisch zu bewältigende Komplexität mehrerer unter ERNEST realisierter Anwendungssysteme motiviert. Die parallele Abarbeitung des erweiterten Konzeptflusses erweist sich dabei als problemlos; eine Beschränkung auf die konzeptlokalen Parallelisierungsmöglichkeiten verzögert jedoch die Durchführung von unabhängigen Berechnungen in anderen Konzepten und begründet die im nächsten Kapitel vorgenommene Integration von Konzept– und Netzwerkebene in einem Instantiierungsschema.

Auf der Netzwerkebene sind die gleichzeitige Verarbeitung alternativer Begriffsvarianten und die parallele Instantiierung der Konzepte einer Regelprämisse zu unterscheiden. Während sich erstere auf die Verfolgung konkurrierender Suchbaumpfade zurückführen läßt, verdeutlicht letztere die konträre Natur von Parallelität und Kontrolle. Die parallele Instantiierung von Konzepten führt im allgemeinen zu einer starken Vergrößerung des zu durchsuchenden Zustandsraumes, die nur im sequentiellen Falle durch die schrittweise Nutzung von bereits berechneten Zwischenergebnissen (Generierung modifizierter Konzepte) verhindert, d.h. *"kontrolliert"* werden kann.

Die abschließend durchgeführten Experimente zur Parallelisierung der A^*–Suche verwenden das Prinzip der Aufgabenverteilung, bei dem die besten Analysezustände gleichzeitig weiterverarbeitet (expandiert) werden. Da hierbei sämtliche Objekte der Suchbaumknoten und zusätzliche Kontrollinformation zwischen den beteiligten Rechnerknoten ausgetauscht werden müssen, zeigen sich nur bei schlecht informierten Restkostenschätzungen und der Verwendung weniger Prozessoren akzeptable Geschwindigkeitsgewinne. Es ist somit zu erwarten, daß die durch die Parallelisierung der Instantiierung generierte Hypothesenflut durch eine Beschleunigung der Graphsuche nicht kompensiert werden kann. Darin

ist die wesentliche Motivation für den Entwurf eines neuen Kontrollalgorithmus zu sehen, der im folgenden Kapitel vorgestellt wird.

Kapitel 4

Iterativ-optimierende und parallele Kontrolle

Die im vorigen Kapitel dargestellten Überlegungen zur Parallelisierung des ERNEST–Kontrollalgorithmus haben zum einen verschiedene Möglichkeiten der Parallelverarbeitung aufgezeigt, zum anderen jedoch auch deren Probleme verdeutlicht. Diese reichten von einer begrenzten Parallelität auf der Konzeptebene über eingeschränkte Kontrollmechanismen auf der Netzwerkebene bis zu hohen Kommunikationsanforderungen bei der Parallelisierung der Graphsuche und motivieren den Entwurf eines neuen, für die Parallelverarbeitung geeigneteren Kontrollalgorithmus. Die Entwicklung eines solchen Verfahrens ist Gegenstand dieses Kapitels.

Nach einer Zusammenfassung der wesentlichen Anforderungen und Entwurfsziele in Abschnitt 4.1 befaßt sich Abschnitt 4.2 mit der Wahl geeigneter Elementaroperationen für die parallele Kontrolle und der automatischen Bestimmung des Datenflusses zwischen diesen. Daran anschließend stellt Abschnitt 4.3 den neu entwickelten, iterativ–optimierenden Kontrollalgorithmus vor und abschließend werden Möglichkeiten zur Parallelisierung diskutiert, die in Kapitel 5 an einer realen Anwendung beispielhaft evaluiert werden.

4.1 Entwurfsziele

Als Grundvoraussetzung für sehr schnelle ("reflektorische") Analyseprozesse wird in [Sha89, Sha91] eine direkte Übereinstimmung der syntaktischen Struktur des benötigten Wissens mit den verwendeten Inferenzmechanismen genannt. Es wird ausgeführt, daß solche Formalismen diese Forderung erfüllen, die Wissen durch gerichtete, azyklische Graphen repräsentieren, in denen die Knoten geeigneten "Informationsstücken" entsprechen und deren Kanten die inferentiellen Abhängigkeiten zwischen diesen darstellen.

Semantische Netze besitzen eben diese Eigenschaft, die jedoch in unterschiedlichen

Ansätzen verschieden stark ausgeprägt ist. Sie tritt am deutlichsten in Systemen auf, die einfach aufgebaute Konzepte und simple Inferenzmechanismen besitzen; zu nennen sind hier in erster Linie Formalismen, die nach dem Prinzip der Aktivierungsausbreitung *(spreading activation)* arbeiten [Fah79, Fah83, Hen88, Kim90, Mol90].

Bei diesem, erstmals von [Qui68] vorgeschlagenen Prinzip werden kurze, zumeist aus wenigen Bits oder einer skalaren Größe bestehende Nachrichten *(Marken)* zwischen den Knoten ausgetauscht und dort einfachen logischen Operationen unterzogen. Die parallele Ausbreitung und Manipulation der Marken verspricht hierbei eine Verarbeitungsdauer, die nicht von der Anzahl der Knoten, sondern von der Tiefe der Wissensbasis (d.h. dem längsten Kantenzug zwischen zwei Knoten) abhängt; ein Beispiel hierzu wurde bereits auf Seite 18 diskutiert. Während die potentielle Schnelligkeit des Inferenzprozesses als Vorteil des Ansatzes gewertet werden muß, ergeben sich Probleme bezüglich der Repräsentation des deklarativen Wissens. Da ein Netzwerkknoten einen relativ kleinen Ausschnitt des problemspezifischen Wissens (beispielsweise ein einzelnes Attribut) beinhaltet, wird die Wissensbasis leicht unübersichtlich und der Aufbau größerer Wissensbasen ist umständlich, sofern dies nicht automatisch geschehen kann.

Ein Rückblick auf Kapitel 2 zeigt, daß die hier skizzierten Überlegungen zur Verarbeitungsstrategie bei der Entwicklung der ERNEST–Netzwerksprache zugunsten einer ergonomischeren Wissensrepräsentation zurückgestellt wurden, die sich im komplexen Aufbau der Konzepte und zahlreichen Strukturierungsmöglichkeiten für große Wissensbasen äußert. Darüber hinaus wurde ersichtlich, daß die speziellen Probleme der Musteranalyse relativ komplizierte Inferenz– und Kontrollmechanismen erforderlich machen. Beim Entwurf eines neuen Kontrollalgorithmus befinden wir uns daher in einem Spannungsfeld, daß einerseits von den bestehenden Zeitschranken und den Grundlagen schneller Inferenzprozesse und andererseits von der Forderung nach geeigneten Möglichkeiten zur Wissensdeklaration und der Notwendigkeit der Integration komplexen prozeduralen Wissens gebildet wird.

Um beiden Positionen gleichermaßen gerecht zu werden, formulieren wir die folgenden Entwurfskriterien:

> ***Ergonomie der Repräsentation:*** Der Einsatz der ERNEST–Systemschale in mehreren Bereichen der wissensbasierten Musteranalyse zeigt die Vorteilhaftigkeit der objektzentrierten Darstellung von Begriffen und ihrer abgeschlossenen, "internen" Beschreibung durch Attribute und Relationen. Die zusätzliche Beschränkung auf wenige Kantentypen ermöglicht die relativ rasche Entwicklung von Systemen mit einigen hundert Konzepten und Kanten. Die hier deutlich werdende ergonomische Adäquatheit des framebasierten Repräsentationsformalismus sollte für den Benutzer erhalten bleiben.
>
> ***Massive Parallelität:*** Die bereits in Kapitel 1 skizzierten engen Zeitschranken für die Verarbeitungsdauer und die große Anzahl auszuführender Operationen be-

gründen die Forderung nach einem massiv–parallelen Vorgehen. Das zu entwickelnde Verfahren sollte daher sowohl alle Möglichkeiten der Parallelisierung des Instantiierungsprozesses nutzen als auch die Parallelisierung der übergeordneten Kontrolle gestatten.

Problemunabhängigkeit: Die Notwendigkeit eines allgemein anwendbaren Verfahrens wurde bereits in Kapitel 2 begründet. Als Konsequenz daraus ist zu fordern, daß auch die Wissensnutzung durch einen parallelen Kontrollalgorithmus nicht von den in der Wissensbasis dargestellten Begriffen abhängt, sondern allein auf der Netzwerksyntax beruht.

Kommunikationsverhalten: Die Experimente zur Parallelisierung des A^*–Algorithmus haben den Einfluß der zwischen den Prozessoren auszutauschenden Datenmengen auf den zu erzielenden Speedup verdeutlicht. Sowohl bei der Bestimmung der atomaren Schritte des Instantiierungsprozesses als auch bei der Definition von Analysezuständen für die Kontrollstrategie ist daher ein konstanter, möglichst geringer Datenaustausch anzustreben.

Speichereffizienz: Auch im Hinblick auf die Realisierung größerer Systeme sind Begrenzungen des (prozessor-)lokalen Speicherplatzes zu berücksichtigen. Obwohl die wachsende Anzahl konkurrierender Hypothesen und eingeschränkte Möglichkeiten zu deren Unterdrückung den Entwurf eines speichereffizienten Algorithmus erschweren, sollte das Verfahren nicht die zumindest theoretisch unbegrenzte Speicherkapazität *skalierbarer* Multiprozessorsysteme — wie beispielsweise der Erlanger MEMSY–Architektur (*Modular Expandable Multiprocessor SYstem* [Fri89, Hof93, Lin94]) — beanspruchen. Vielmehr ist anzustreben, daß der Algorithmus unter entsprechenden Geschwindigkeitsverlusten auch auf einer herkömmlichen Workstation mittlerer Kapazität ablaufen kann.

Any–Time–Eigenschaft: Aufgrund nur begrenzt zur Verfügung stehender Zeit– und Speicherplatzressourcen sind Vorkehrungen zu treffen, die ein frühzeitiges Erreichen einer möglichst optimalen (Teil-)interpretation sichern, die gegebenenfalls an andere Komponenten des Analysesystems (beispielsweise zur Generierung einer geeigneten Systemreaktion) weitergegeben werden kann. Diese Forderung legt ein iteratives Vorgehen nahe, das jederzeit zu unterbrechen ist und mit einer möglichst geringen "Taktfrequenz" eine sukzessive Verbesserung der erzielten Interpretation vornimmt.

Um die Ergonomie der Darstellung zu gewährleisten und gleichzeitig die Realisierung eines schnellen Kontrollalgorithmus zu ermöglichen, lassen wir die Netzwerksprache unverändert und entwerfen zunächst eine automatische Transformation der Wissensbasis,

welche die oben geforderte Übereinstimmung zwischen syntaktischer Struktur und Nutzung des Wissens sicherstellt. Da die Transformation unabhängig von den Zwischenergebnissen der Interpretation ist, muß sie lediglich einmal während der im folgenden beschriebenen *Analysevorbereitung* durchgeführt werden. Die Instantiierung des so erzeugten Datenflußgraphen und eine Neuformulierung des Kontrollalgorithmus unter Berücksichtigung der übrigen Entwurfsziele werden anschließend erörtert.

4.2 Analysevorbereitung

Bereits bei der Diskussion der Inferenzregeln in Abschnitt 2.3 und der Parallelisierungsmöglichkeiten auf Konzept– und Netzwerkebene (Abschnitt 3.3) wurde erwähnt, daß die bei der Analyse zu berücksichtigenden Abhängigkeiten zwischen den Elementen der Wissensbasis vorab ermittelt werden können, um die Verarbeitung zu beschleunigen. Neben der Überprüfung der formalen Konsistenzkriterien und der Zyklenfreiheit gehört dazu die explizite Vererbung der Substrukturen, die Feststellung der Reihenfolge der Attributberechnungen und die Ermittlung der zulässigen Prämissenbelegungen für die Instantiierung.

Die Bestimmung von Konzept– und Netzfluß ist für die sukzessive Erzeugung von Instanzen und modifizierten Konzepten durch den ERNEST–Kontrollalgorithmus ausreichend; die Analysevorbereitung für ein massiv–paralleles Instantiierungsschema muß jedoch darüber hinaus auch die Abbildung der Wissensbasis auf die zur Verfügung stehenden Prozessoren vornehmen. Die Verteilung bedingt zunächst die Bestimmung geeigneter Elementaroperationen, die unabhängig voneinander auf den Prozessoren ablaufen können und vor der Realisierung der Analysevorbereitung festgelegt werden müssen.

4.2.1 Elementaroperationen für die Parallelisierung

Die Wahl von atomaren Verarbeitungsschritten ist ein zentrales Problem des Entwurfs paralleler Algorithmen, da hierdurch wichtige Einflußgrößen auf den zu erwartenden Speedup festgelegt werden, von denen stellvertretend der Umfang des Datenaustausches zwischen den Rechnerknoten genannt sei. Geeignete Elementaroperationen können daher im allgemeinen nicht ohne die Berücksichtigung der Architekturmerkmale des jeweiligen Zielrechners ermittelt werden. Dies ist für die regulär strukturierten Algorithmen der ikonischen Bildverarbeitung relativ einfach (vgl. Kapitel 1) und gehört bei rein numerischen Verfahren, die nach dem Prinzip der Datenpartitionierung arbeiten, auch in anderen Einsatzgebieten von Parallelrechnern zum Stand der Technik (siehe z.B. [Per93, Fri94]).

Für die Entwicklung eines parallelen Kontrollalgorithmus sind die Voraussetzungen dagegen bedeutend ungünstiger. Die unregelmäßigen Abhängigkeiten der Algorithmen aus der Symbolik, eine Vielzahl unterschiedlicher Operationen, und die Beeinflussung durch die Ergebnisse der initialen Segmentierung sind Faktoren, welche die Abstimmung der Ele-

mentaroperationen auf die zur Verfügung stehende Hardware erheblich erschweren. Das Erreichen einer möglichst geringen Gesamtlaufzeit erfordert darüber hinaus die Analyse des Rechenzeitbedarfs der Abschnitte eines zu parallelisierenden Algorithmus und müßte in unserem Falle auch eine Untersuchung der anwendungsspezifischen Analyseprozeduren einschließen. Die für einen universell einsetzbaren Kontrollalgorithmus zu fordernde Problemunabhängigkeit macht es jedoch notwendig, die Parallelisierung einzelner Attributberechnungen und Bewertungen von unseren Überlegungen auszuklammern, und uns auf die Elemente der Wissensbasis als Kandidaten für atomare Berechnungen zu konzentrieren. In einem Bildanalysesystem, dessen Aufbau dem in Kapitel 1 beschriebenen Schichtenmodell folgt, stellt der so erzielbare Speedup (vgl. die experimentellen Untersuchungen in Kapitel 5) daher eine untere Schranke für die Beschleunigung der Analyse dar.

Eine oft verfolgte Strategie bei der Parallelisierung von Inferenzprozessen in semantischen Netzen ist die Zuordnung von Konzepten zu Prozessoren und die Realisierung der Kanten als Verbindungen [Dix87, Mol89, Mol90]. Dieses Vorgehen — dem wir rückblickend die Parallelisierung des Netzflusses in Abschnitt 3.3 zuordnen können — wird für die einfachen Konzepte der einleitend zitierten Ansätze zur Aktivierungsausbreitung als problemlos beschrieben. Bei einem komplexeren Knotenaufbau ist jedoch zu erwarten, daß Konzepte mit zahlreichen Attributen und Relationen den "Flaschenhals" der Analyse bilden. Daher wird in einigen parallelen Netzwerksystemen [Tou87, Sha88, Der90] eine automatische Auflösung der framebasierten Repräsentation und eine Abbildung der einfacheren Substrukuren auf die Prozessoren vorgenommen. Die Ausnutzung der größeren Parallelität zwischen den Untereinheiten verspricht dabei eine Verkürzung der Verarbeitungsdauer und darüber hinaus ein effizienteres Kommunikationsverhalten, da keine großen Objekte zwischen den Rechnerknoten auszutauschen sind. Da die ERNEST–Netzwerksprache zum einen die Definition von Konzepten mit beliebig vielen Attributen, Relationen und Kanten erlaubt, und zum anderen auch die automatische Bestimmung von deren Abhängigkeiten zuläßt, ist es sinnvoll und möglich, dieses Vorgehen zur Erzielung größerer Parallelisierungserfolge aufzugreifen.

Betrachten wir nochmals die Inferenzregeln aus Abschnitt 2.3, so ist zu erkennen, daß deren Aktionsteil im wesentlichen aus der Aktivierung der problemabhängigen Analyseprozeduren besteht. Es ist daher naheliegend, alle zur Instantiierung eines Konzeptes notwendigen Inferenzen bis auf die Ebene dieser — im Sinne der Problemunabhängigkeit elementaren — Berechnungen aufzulösen. Um dabei auch die Abhängigkeiten zwischen Substrukturen aus verschiedenen Konzepten zu erfassen, kombinieren wir den erweiterten Konzeptfluß und den Netzfluß in einem Datenflußgraphen, dessen Knoten die Berechnung bzw. Bewertung eines einzelnen Attributes, einer Relation, Kante oder einer Instanz repräsentieren. Aufgrund der grundlegenden Bedeutung der Attributbeschreibungen für die Analyse — sie bilden die Schnittstelle zu den Segmentierungsergebnissen und initialisieren die Instantiierung — wird der Datenfluß zwischen den Analyseprozeduren im folgenden

abkürzend als *Attributfluß* bezeichnet; die Transformation einer gegebenen Wissensbasis in einen Attributflußgraphen beschreibt der nächste Abschnitt.

4.2.2 Konstruktion des Attributflußgraphen

Der Attributflußgraph ist ein problemunabhängiges, feingranulares Instantiierungsschema, dessen Vorteile in der größtmöglichen Parallelität und dem Transport geringer Datenmengen — einzelne Werte bzw. Bewertungen anstelle komplex aufgebauter Instanzen — zwischen den Prozessoren zu sehen sind. Der Wegfall der Gliederungsmöglichkeiten durch die Kantentypen und die aus der Aufgabe der begriffszentrierten Repräsentation resultierende größere Knotenanzahl erschweren es jedoch erheblich, aufgabenspezifisches Wissen direkt als Attributflußgraph darzustellen. Daher ist eine automatische Konstruktion aus der strukturierten und übersichtlicheren Darstellung der ERNEST–Wissensbasis, für die überdies geeignete Entwicklungswerkzeuge existieren, erforderlich; diese Konstruktion wird nun ausführlich beschrieben.

Die Kombination von netz– und konzeptweitem Datenfluß erfolgt in einem mehrstufigen Prozess, der zunächst alle Instantiierungspfade zu den Zielkonzepten eines Netzwerkes ermittelt. Anschließend wird die dabei gewonnene Zwischendarstellung der Wissensbasis durch die Berücksichtigung der Abhängigkeiten zwischen den Substrukturen zum Attributflußgraphen verfeinert. In einem optionalen, nur auf Wunsch des Benutzers durchgeführten dritten Schritt werden schließlich alle Knoten des Graphen eliminiert, die nicht zur Unterscheidung zwischen den verschiedenen Analysezielen beitragen.

Im ersten Schritt der Analysevorbereitung, der *Netzwerkexpansion*, sind die verschiedenen Begriffsvarianten in den Konzepten aufzulösen. Um *alle* Inferenzen des Analyseprozesses zu beachten, ist auch zu überlegen, wie optionale und/oder mehrfach auftretende Teile in den Datenfluß integriert werden können. Bereits in Kapitel 2 wurde ein Algorithmus zur Bestimmung eines Instantiierungspfades für ein vorgegebenes Zielkonzept angegeben, der auf der sukzessiven Ermittlung der Prämissenkonzepte für die Inferenzregeln zur Generierung von Instanzen (Regeln 1 und 2) beruhte. Dabei wurde bemerkt, daß der berechnete Pfad nicht eindeutig bestimmt war, da für jedes Konzept nur jeweils eine von mehreren möglichen Prämissen betrachtet wurde. Die Anzahl sämtlicher Pfade, die den gleichen Einflußgrößen unterliegt wie die Anzahl modellbedingter Wege durch den Suchbaum (vgl. Abschnitt 3.2.2), erlaubt eine explizite Entwicklung aller Pfade nur für relativ kleine Netzwerke. Für komplexere Wissensbasen, die starken Gebrauch von den Strukturierungsmöglichkeiten der Netzwerksprache machen, bietet sich daher eine implizite Repräsentation aller Pfade an, aus der der Kontrollalgorithmus die optimal zu den Sensordaten passenden Instanzen ermittelt. Dies widerspricht zwar zunächst der Forderung nach einer Abbildung sämtlicher inferentieller Abhängigkeiten auf die Hardware, stellt aber einen sinnvollen Kompromiß zwischen der benötigten Knotenanzahl und einer Vergrößerung des

```
/* Netzwerkexpansion */
gegeben: Liste von Zielkonzepten GOAL
initialisiere einen leeren Graphen ℰ𝒩ℰ𝒯 = (V_ℰ, E_ℰ) = (∅, ∅)
∀ C_g ∈ GOAL
    initialisiere eine Liste REST := {C_g}
    WHILE REST ≠ ∅
        entferne das letzte Element A aus REST
        i = index(A)
        V_ℰ = V_ℰ ∪ {A_i}
        bestimme die Mengen OBLPREM_k(A) aus der Prämisse von Regel 2
        bestimme die Mengen OPTPREM_l(A) aus der Prämisse von Regel 3
        setze 𝒫(A) = ⋃_k OBLPREM_k(A) ∪ ⋃_l OPTPREM_l(A)
        ∀ B ∈ 𝒫(A)
            bestimme d_max aus der Kante A --T_E--> B
            FOR 1 ≤ n ≤ d_max
                j = index(B)
                bringe B an das Ende von REST
                E_ℰ = E_ℰ ∪ {(A_i, B_j)}
Ergebnis: expandiertes Netzwerk in ℰ𝒩ℰ𝒯 = (V_ℰ, E_ℰ)
```

Bild 4.1: Der erste Schritt der Analysevorbereitung: Algorithmus zur Netzwerkexpansion.

Aufwands für die übergeordnete Kontrolle dar, die durch die konkurrierenden Hypothesen ohnehin notwendig ist.

Bild 4.1 zeigt den Algorithmus zur Netzwerkexpansion, dessen Eingabe eine Liste mit den möglichen Analysezielen ist. Das Verfahren greift die aus Abschnitt 2.3 bekannte Untersuchung der Regelprämissen auf, wobei sich jedoch Unterschiede in der Behandlung von Modalitätsmengen, optionalen Teilen und Kontexten ergeben. Während in Algorithmus 2.4 die Auswahl einer Modalitätsmenge zur Erfüllung genau einer Regelprämisse ausreichte, werden nunmehr sämtliche Prämissen simultan erfaßt, indem alle Bestandteile und Konkretisierungen unabhängig von ihrer Gruppierung in Modalitätsmengen als Prämissenkonzepte des aktuellen Konzeptes aufgefaßt werden. Zwischen obligatorischen und optionalen Teilen, von denen letztere vorher unberücksichtigt blieben und von der ERNEST–Kontrolle erst bei Bedarf behandelt wurden, wird nunmehr nicht mehr unterschieden. Die Grundidee dabei ist, daß die Instantiierung optionaler Teile durch die parallele Bearbeitung der Konzepte ohne Zeitverlust möglich ist, wenn die Kontrollstrategie einen effizienten Umgang mit den zusätzlich erzeugten Hypothesen gestattet. Dieser Idee folgt auch die Behandlung mehrfach auftretender Teile, die gemäß der maximalen Kantendimension d_{max} dupliziert

werden, und die Auflösung der Liste alternativer Zielknoten einer Kante vor Beginn der Analyse.

Kontextabhängigkeiten können bei der Netzwerkexpansion vollständig unberücksichtigt bleiben, da sich diese — wie im folgenden (vgl. Seite 90) erläutert — aufgrund der eingeschränkten Netzwerksprache nur bei der Ermittlung der Abhängigkeiten zwischen Attributen und Relationen niederschlagen. In Algorithmus 4.1 können die obligatorischen Bestandteile und Konkretisierungen daher direkt aus der Prämisse von Regel 2 bestimmt werden. Aus Gründen der Vollständigkeit sei noch die Funktion index erwähnt, die durch

$$\mathsf{index}(A) \;=\; \begin{cases} i+1 & \text{falls} \quad \exists i : A_i \in V_{\mathcal{E}} \\ 1 & \text{sonst} \end{cases} \tag{4.1}$$

definiert ist, und für die korrekte Indizierung und Verwaltung der mehrfach benötigten Konzepte sorgt.

Anhand des in Bild 4.2 dargestellten Beispielnetzwerks und der aus der Expansion resultierenden Zwischendarstellung soll die Arbeitsweise von Algorithmus 4.1 im folgenden kurz skizziert werden. Nehmen wir an, daß zunächst das Zielkonzept A behandelt wird, so wird dieses nach der Entfernung aus der Liste REST mit dem Index 1 versehen, da noch kein Exemplar von A in der Knotenmenge $V_{\mathcal{E}}$ vorhanden ist; anschließend wird $V_{\mathcal{E}}$ um den Knoten A_1 erweitert. Die Bestimmung der Mengen der obligatorischen und optionalen Bestandteile und Konkretisierungen von A aus den Prämissen der Regeln 2 und 3 liefert zwei obligatorische Mengen $\mathsf{OBLPREM}_1(A) = \{C\}$ und $\mathsf{OBLPREM}_2(A) = \{D\}$ — im Beispiel besitzt die obligatorische Kante zwei alternative Zielknoten C und D — sowie die optionale Menge $\mathsf{OPTPREM}_1(A) = \{E\}$, welche zur Menge $\mathcal{P}(A) = \{C, D, E\}$ der Prämissenkonzepte vereinigt werden.

Da die obligatorische Kante die maximale Dimension $d_{max} = 2$ besitzt, werden bei der Abarbeitung der Prämissenkonzepte je zwei Exemplare von C und D an das Ende

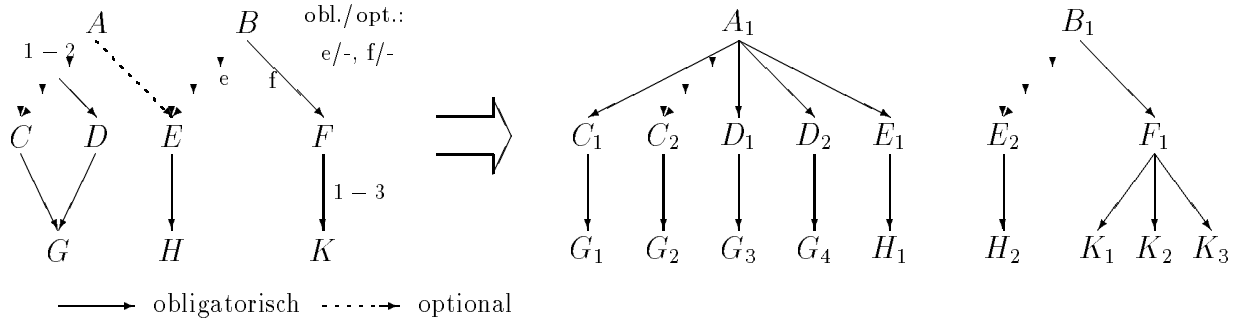


Bild 4.2: Der erste Schritt der Analysevorbereitung: Beispiel zur Netzwerkexpansion.

der Liste REST gebracht und die vier Kanten $(A_1, C_1), (A_1, C_2), (A_1, D_1)$ und (A_1, D_2) zur Kantenmenge $E_{\mathcal{E}}$ hinzugefügt. Nach einer analogen Behandlung der Kante $A \longrightarrow E$ erhalten wir schließlich REST $= (C, C, D, D, E)$ und $E_{\mathcal{E}} = \{(A_1, C_1), \ldots, (A_1, E_1)\}$. Im nächsten Schleifendurchlauf wird die Knotenmenge $V_{\mathcal{E}}$ um den Knoten E_1 ergänzt, und das hier skizzierte Verfahren wird für die obligatorische Prämisse OBLPREM$_1(E) = \{H\}$ wiederholt. Die Abarbeitung der minimalen Konzepte H (und später G) führt aufgrund der leeren Regelprämissen schließlich dazu, daß keine neuen Konzepte in die Liste REST aufgenommen werden, und nach der Erzeugung des linken Teils des expandierten Netzwerkes weitere Analyseziele (hier: B) behandelt werden können.

Ein Konzept, das zur Instantiierung mehrerer Analyseziele benötigt wird (wie etwa das Konzept E in Beispiel 4.2), ist für jedes Zielkonzept durch mindestens eine eigene Kopie vertreten, welche aus Effizienzgründen durch einen Verweis auf die Wissensbasis realisiert ist. Dadurch eröffnet sich eine einfache Möglichkeit zur Verteilung der Analyse, da die Netzwerkausschnitte zu verschiedenen Zielkonzepten disjunkt sind und somit von *unabhängigen* Prozessoren bzw. Prozessorclustern bearbeitet werden können.

Zur Generierung des Attributflusses sind in einem zweiten Schritt die Knoten des expandierten Netzwerkes in ihre Substrukturen aufzuspalten und anschließend die Datenabhängigkeiten zwischen den dabei erzeugten Attributflußknoten zu ermitteln. Das Vorgehen ist in Algorithmus 4.3 zusammengefaßt. Da die Aufgabe der Knoten in der Aktivierung der zugehörigen Analyseprozeduren besteht, ergibt sich der Datenfluß aus den Argumentbeziehungen der Werteberechnungs– und Bewertungsfunktionen.

Aus den Gleichungen (3.1 – 3.4) ist ersichtlich, daß die Argumente der Analyseprozeduren in ERNEST durch die Angabe von eindeutigen Rollennamen spezifiziert werden. Die Syntax der Netzwerksprache gestattet somit die automatische Bestimmung der Substrukturen, die bereits instantiiert sein müssen, wenn eine Prozedur aktiviert werden soll. Die verschiedenen Belegungsmöglichkeiten der Argumente erlauben die Definition folgender Abhängigkeiten zwischen den Substrukturen:

- Wird für eine Prozedur kein Argument spezifiziert, so stellt die zugehörige Substruktur (i.a. eine Attributbeschreibung mit einer Werteberechnung $\varphi_A : \emptyset \mapsto W^n$) eine Schnittstelle zu den initialen Segmentierungsergebnissen dar, oder ihre Berechnung erfordert eine Interaktion mit dem Benutzer.

- Mit einem *einstelligen* Rollennamen wird eine (ererbte) Substruktur des gleichen Konzeptes als Argument der Berechnung deklariert.

- Bei der Angabe eines *zweistelligen* Rollennamens *rolle1.rolle2* wird *rolle1* als Kantenrolle interpretiert. Damit ist der Zugriff auf Attribute aus Bestandteilen, Konkretisierungen oder — falls *rolle1* die Konstante SUPER ist — aus einem übergeordneten Kontext des Konzeptes möglich.

/* *Verfeinerung des expandierten Netzwerks zum Attributflußgraphen* */
gegeben: expandiertes Netzwerk $\mathcal{ENET} = (V_\mathcal{E}, E_\mathcal{E})$
initialisiere einen Graphen: $\mathcal{AFG} := (V_\mathcal{A}, E_\mathcal{A}) = (\emptyset, \emptyset)$
initialisiere $V_{conc} := \emptyset$, $V_{att} := \emptyset$, $V_{rel} := \emptyset$, $V_{edge} := \emptyset$
/* *expandiertes Netzwerk aufspalten* */
$\forall\ v_\mathcal{E} = (C_i) \in V_\mathcal{E}$
bringe einen Bewertungsknoten $v_\mathcal{A} = (C_i, T_C, G_C)$ nach V_{conc}
$\forall\ T_A \in C$: bringe einen Attributknoten $v_\mathcal{A} = (C_i, T_A, \varphi_A, G_A)$ nach V_{att}
$\forall\ T_S \in C$: bringe einen Relationsknoten $v_\mathcal{A} = (C_i, T_S, G_S)$ nach V_{rel}
$\forall\ e_\mathcal{E} = (C_i, D_j) \in E_\mathcal{E}$
bringe einen Kantenknoten $v_\mathcal{A} = (C_i, T_E, D_j, G_E)$ nach V_{edge}
$V_\mathcal{A} := V_{conc} \cup V_{att} \cup V_{rel} \cup V_{edge}$
/* *Knoten aus* $V_\mathcal{A}$ *gemäß Tabelle 4.4 verzeigern* */
$\forall\ v_\mathcal{A} \in V_\mathcal{A}$
initialisiere die Menge der Vorgänger $V_{pred} := \emptyset$
IF $v_\mathcal{A} = (C_i, T_A, \varphi_A, G_A) \in V_{att}$ /* *Attributknoten* */
THEN $R := \{r = (rolle1.rolle2) \mid \varphi_A(\ldots, r, \ldots)\}$
$\forall\ r \in R$
IF *rolle1* = `LEER`
THEN $V_{pred} := \{u \in V_{att} \mid u = (C_i, rolle2, ., .)\}$
ELSE IF *rolle1* = `SUPER`
THEN $V_{pred} := \{u \in V_{att} \mid u = (B_j, rolle2, ., .) \wedge B_j \stackrel{kbst}{\longrightarrow} C_i\}$
ELSE $V_{pred} := \{u \in V_{att} \mid u = (B_j, rolle2, ., .) \wedge C_i \stackrel{rolle1}{\longrightarrow} B_j\}$
IF $v_\mathcal{A} = (C_i, T_S, G_S) \in V_{rel}$ /* *Relationsknoten* */
THEN $R := \{r = (rolle1.rolle2) \mid G_S(\ldots, r, \ldots)\}$
bestimme die Vorgängerknoten analog zu den Attributknoten
IF $v_\mathcal{A} = (C_i, T_E, D_j, G_E) \in V_{edge}$ /* *Kantenknoten* */
THEN $V_{pred} := \{u \in V_{conc} \mid u = (D_j, ., .)\}$
IF $v_\mathcal{A} = (C_i, T_C, G_C) \in V_{conc}$ /* *Bewertungsknoten* */
THEN $R := \{r = (\texttt{LEER}.rolle2) \mid G_C(\ldots, r, \ldots)\}$
$\forall\ r \in R : V_{pred} := V_{pred} \cup \{u_\mathcal{A} \in V_{att} \cup V_{rel} \cup V_{edge} \mid u_\mathcal{A} = (C_i, rolle2, ., .)\}$
$\forall\ u_\mathcal{A} \in V_{pred}$
$E_\mathcal{A} := E_\mathcal{A} \cup \{(u_\mathcal{A}, v_\mathcal{A})\}$
Ergebnis: Attributflußgraph $\mathcal{AFG} = (V_\mathcal{A}, E_\mathcal{A})$

Bild 4.3: Der zweite Schritt der Analysevorbereitung: Verfeinerung des expandierten Netzwerks zum Attributflußgraphen.

In Abhängigkeit von der Aufgabe einer Prozedur werden die möglichen Argumente der Analyseprozeduren auf eine sinnvolle Teilmenge aller Substrukturen im Netzwerk eingeschränkt. Bild 4.4 faßt die zulässigen Abhängigkeiten übersichtsartig zusammen; Beispiele können dem Netzwerkausschnitt auf Seite 65 entnommen werden.

Analyseprozedur	Argumente	Bedeutung
Attributwerteberechnung	T_A	Das Attribut mit der Rolle T_A aus dem gleichen Konzept
	$T_E.T_A$	Das Attribut mit der Rolle T_A aus dem Bestandteil oder der Konkretisierung mit der Rolle T_E
	SUPER.T_A	Das Attribut mit der Rolle T_A aus einem Kontextkonzept
Attributbewertung	–	Keine explizit anzugebenden Argumente. Bewertet wird das zugehörige Attribut
Relationsbewertung	T_A $T_E.T_A$ SUPER.T_A	Analog zur Attributwerteberechnung
Kantenbewertung	–	Keine explizit anzugebenden Argumente. Bewertet wird das Objekt, das an die Kante gebunden ist.
Bewertung	T_A T_S T_E	Das Attribut mit der Rolle T_A Die Relation mit der Rolle T_S Die Kante mit der Rolle T_E

Bild 4.4: Übersicht über die Analyseprozeduren in ERNEST. Nicht berücksichtigt ist die Referenzkante mit der zugehörigen Funktion zur Referenzauswahl (zur Begründung siehe Seite 46).

Werden zyklische Kanten zwischen den Konzepten des Netzwerkes bereits durch die ERNEST–Analysevorbereitung verhindert, so sind bei der Berechnung des Attributflußgraphen Zyklen zwischen Attributen zu berücksichtigen. Diese können bei der Angabe zweistelliger Rollennamen entstehen, wenn von einem kontextabhängigen Bestandteil auf Attribute des Kontexts, und von diesen wiederum auf die Attribute des Bestandteils zugegriffen wird (vgl. Bild 4.5). Wie bereits erwähnt, ist der ERNEST–Kontrollalgorithmus in der Lage, solche Zyklen durch die Generierung partieller Instanzen aufzulösen. Dazu muß

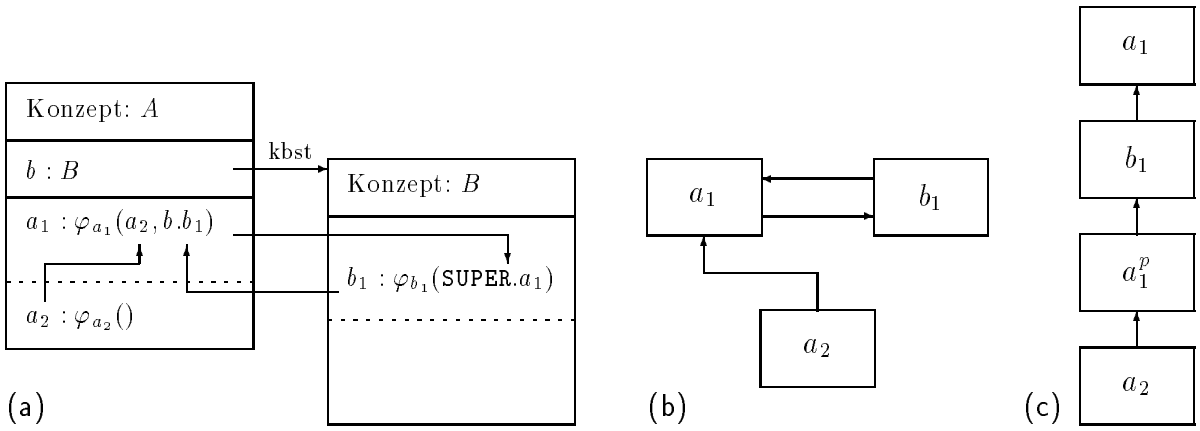


Bild 4.5: Zyklische Abhängigkeit von Attributen der Wissensbasis (a), (b) und ihre Auflösung im Attributflußgraphen (c).

im Beispiel 4.5.a zunächst eine partielle Instanz $I^p(A)$ generiert werden, die lediglich eine Restriktion für das Attribut a_1 enthält. Unter Verwendung der erzeugten Einschränkung für a_1 kann B instantiiert, und mit dem dabei berechneten Wert für b_1 schließlich a_1 berechnet und $I(A)$ generiert werden.

Bild 4.5.c zeigt, wie die hier skizzierte Strategie durch die Einführung *partieller* Attributknoten zur Berechnung von Restriktionen im Attributfluß nachgebildet werden kann. Die Tatsache, daß zyklische Abhängigkeiten zwischen Attributen in den bisher realisierten Anwendungen nicht zu beobachten sind, ist jedoch ein starkes Indiz dafür, daß generell auf diese Modellierungsmöglichkeit verzichtet werden kann. Auch im Sinne der oben erhobenen Forderung nach einem *azyklischen* Inferenzgraphen geht die Analysevorbereitung daher davon aus, daß bei der Transformation einer gegebenen Wissensbasis keine derartigen Zyklen vorliegen.

Am Beispiel des Konzepts "Quadrat" (vgl. Seite 65) soll im folgenden die Verfeinerung des expandierten Netzwerks zum Attributflußgraphen verdeutlicht werden. Die Anwendung von Algorithmus 4.1 liefert zunächst das expandierte Netzwerk $\mathcal{ENET} = (V_\mathcal{E}, E_\mathcal{E})$ mit

$$\begin{aligned} V_\mathcal{E} &= \{\mathrm{Quadrat}_1, \mathrm{V1}_1, \mathrm{V2}_1, \mathrm{H1}_1, \mathrm{H2}_1\} \\ &\quad \text{und} \\ E_\mathcal{E} &= \{(\mathrm{Quadrat}_1, \mathrm{V1}_1), \ldots, (\mathrm{Quadrat}_1, \mathrm{H2}_1)\}, \end{aligned}$$

welches im ersten Schritt der Verfeinerung aufzuspalten ist. Dazu werden zunächst Attributflußknoten für sämtliche Attribute und Relationen sowie für die Bewertung des in $v_\mathcal{E}$ abgelegten Konzepts erzeugt, und mit den zugehörigen Analyseprozeduren aus der Wissensbasis versehen. Während an die Flußgraphknoten zur Bewertung von Relationen und

Konzepten lediglich eine Prozedur G_S bzw. G_C gebunden wird, werden Attributknoten sowohl mit der Attributwerteberechnung φ_A als auch mit der Attributbewertung G_A versehen; beispielsweise erhalten wir die Flußgraphknoten

$$\begin{aligned}
v_1 &= (\text{Quadrat}_1, \text{Quadrat}, \text{quadratBEW}), && /* \textit{ Bewertungsknoten } */\\
v_2 &= (\text{Quadrat}_1, \text{umfang}, \text{umfangWB}, \text{umfangBEW}), && /* \textit{ Attributknoten } */\\
v_3 &= (\text{Quadrat}_1, \text{flaeche}, \text{flaecheWB}, \text{flaecheBEW}), && /* \textit{ Attributknoten } */\\
v_4 &= (\text{Quadrat}_1, \text{formfaktor}, \text{ffaktorWB}, \text{ffaktorBEW}), && /* \textit{ Attributknoten } */\\
v_5 &= (\text{Quadrat}_1, \text{h-parallel}, \text{h-parallelBEW}), && /* \textit{ Relationsknoten } */\\
v_6 &= (\text{Quadrat}_1, \text{v-parallel}, \text{v-parallelBEW}) && /* \textit{ Relationsknoten } */
\end{aligned}$$

durch die Aufspaltung des Knotens $v_{\mathcal{E}} = \text{Quadrat}_1$, und aus der Bearbeitung von $v_{\mathcal{E}} = \text{V1}_1$ ergeben sich die Flußgraphknoten

$$\begin{aligned}
v_7 &= (\text{V1}_1, \text{V1}, \text{linieBEW}), && /* \textit{ Bewertungsknoten } */\\
v_8 &= (\text{V1}_1, \text{laenge}, \text{laengeWB}, \text{laengeBEW}), && /* \textit{ Attributknoten } */\\
v_9 &= (\text{V1}_1, \text{steigung}, \text{steigungWB}, \text{steigungBEW}). && /* \textit{ Attributknoten } */
\end{aligned}$$

Da Abhängigkeiten zwischen den während der Analyse generierten Instanzen bei der Netzwerkexpansion in der Kantenmenge $E_{\mathcal{E}}$ festgehalten werden, können die Attributflußknoten zur Bewertung sämtlicher Bestandteils– und Konkretisierungsbeziehungen direkt mit Hilfe dieser Menge bestimmt werden. Hierbei wird für jede Kante $e_{\mathcal{E}} = (C_i, D_j)$ ein Flußgraphknoten erzeugt, in dem die beteiligten *Instanzen*, die Kantenrolle T_E, und die Kantenbewertung G_E aus der Wissensbasis vermerkt werden. So wird etwa für die Kante $e_{\mathcal{E}} = (\text{Quadrat}_1, \text{V1}_1)$ die Bewertungsprozedur "vertikaleBEW" bestimmt und der Flußgraphknoten

$$v_{10} = (\text{Quadrat}_1, \text{vertikale}_1, \text{V1}_1, \text{vertikaleBEW}) \qquad /* \textit{ Kantenknoten } */$$

generiert.

Die Bestimmung des Datenflusses zwischen den erzeugten Knoten bildet den zweiten Schritt von Algorithmus 4.3, der hier exemplarisch anhand einiger Knoten erläutert werden soll. Aus der Werteberechnung "ffaktor" des Knotens v_4 entnehmen wir zunächst die beiden *einstelligen* Rollennamen "umfang" und "flaeche". Per conventionem gilt hier *rolle1* = `LEER`, weshalb die Vorgängerknoten im Attributflußgraphen aus dem selben Knoten $V1_1$ des expandierten Netzwerkes hervorgegangen sein müssen; ein Vergleich der Attributrollen liefert für v_4 die Knoten v_2 ("umfang") und v_3 ("flaeche"), und dementsprechend werden die Kanten (v_2, v_4) und (v_3, v_4) in die Kantenmenge $E_{\mathcal{A}}$ aufgenommen. Im Gegensatz dazu sind die Argumente der Attributwerteberechnung "flaecheWB" im Knoten v_3

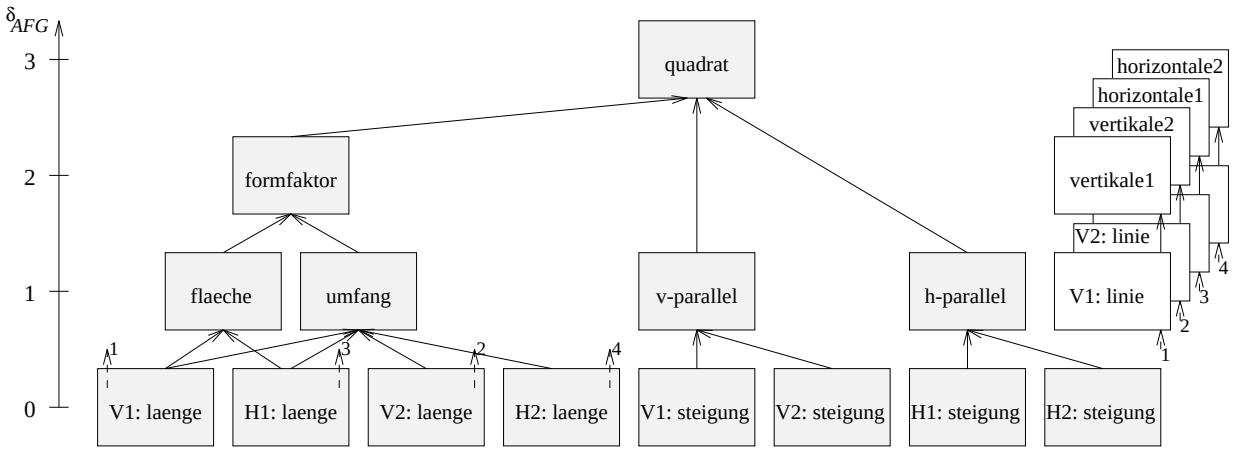


Bild 4.6: Der vollständige Attributflußgraph für das Konzept "Quadrat" aus Beispiel 3.8, S. 65. Die Konstruktion des Graphen durch Algorithmus 4.3 und der am linken Rand angetragene Attributflußgrad werden im Text erläutert.

über *zweistellige* Rollennamen ("vertikale1.laenge","horizontale1.laenge") spezifiziert und *rolle1* muß als Kantenrolle interpretiert werden. Durch die Verfolgung der Kante (hier: "vertikale1") und die Überprüfung der Attributrollen im Zielknoten $V1_1$ erhalten wir schließlich den Knoten v_8 als Vorgänger von v_3. Da in der Attributwerteberechnung des Knotens v_8 keine Argumente referiert werden, besitzt dieser selbst keine Vorgänger und stellt somit einen initialen Knoten des Flußgraphen dar. Zur abschließenden Illustration der Analysevorbereitung zeigt Bild 4.6 den vollständigen Attributflußgraphen für das Konzept "Quadrat", wobei jedoch aus Darstellungsgründen nicht alle Kanten explizit eingetragen, sondern teilweise nur gestrichelt angedeutet und mit einem (numerischen) Verweis versehen sind. Die in Abschnitt 1.4 auf Seite 14 erwähnten unregelmäßigen Abhängigkeiten in der symbolischen Verarbeitung werden hier durch die unterschiedliche Anzahl an Vorgängerknoten deutlich, die in unserem Beispiel zwischen eins (z.B. für den Knoten "V1: linie") und vier (für den Knoten "umfang") beträgt.

Eine nähere Betrachtung von Bild 4.6 macht deutlich, daß lediglich die grau unterlegten Knoten zugleich zur vollständigen Beschreibung der Sensordaten und zur Bewertung des Zielkonzeptes beitragen; die übrigen Knoten besitzen dagegen keinen Einfluß auf die Entscheidung, ob ein Analyseziel (hier: "Quadrat") in den Sensordaten vorliegt. Wenn eine vollständige Beschreibung entweder aus Zeitgründen nicht erstellt werden kann oder aber nicht benötigt wird — Anwendungen dieser Art sind insbesondere in der Bildfolgenanalyse denkbar — so kann auf die Berechnung dieser Knoten verzichtet werden. Der in Bild 4.7 dargestellte, nur auf Wunsch des Benutzers durchgeführte letzte Schritt der Analysevorbereitung sorgt daher für eine "Ausdünnung" des Attributflußgraphen. Ausgehend von den Knoten die keine Analyseziele repräsentieren und keine Nachfolger besitzen — in unserem

/* *Optionale Ausdünnung des Attributflußgraphen* */
gegeben: Attributflußgraph $\mathcal{AFG} = (V_{\mathcal{A}}, E_{\mathcal{A}})$ Menge der zu löschenden Knoten $W_{\mathcal{A}} \subset V_{\mathcal{A}}$
WHILE $W_{\mathcal{A}} \neq \emptyset$
wähle einen Knoten w aus $W_{\mathcal{A}}$
$W_{\mathcal{A}} := W_{\mathcal{A}} \setminus \{w\}$, $V_{\mathcal{A}} := V_{\mathcal{A}} \setminus \{w\}$
$\forall \; v \in V_{\mathcal{A}}$
IF $\exists\ (v,w) \in E_{\mathcal{A}}$
THEN $E_{\mathcal{A}} := E_{\mathcal{A}} \setminus \{(v,w)\}$
IF $\nexists\ v' \in V_{\mathcal{A}} : (v,v') \in E_{\mathcal{A}}$
THEN $W_{\mathcal{A}} := W_{\mathcal{A}} \cup \{v\}$
Ergebnis: reduzierter Attributflußgraph $\mathcal{AFG} = (V_{\mathcal{A}}, E_{\mathcal{A}})$

Bild 4.7: Der letzte Schritt der Analysevorbereitung: Ausdünnung des Attributflußgraphen.

Beispiel sind dies die am rechten Rand von Bild 4.6 eingezeichneten Kantenknoten mit den Rollen "vertikale1",..., horizontale2" — werden die Knoten und Kanten des Datenflußgraphen sukzessive eliminiert. Ziel dieses Vorgehens ist es, durch die Verringerung der Knotenanzahl die Dauer der datengetriebenen Instantiierung und dadurch die "Taktfrequenz" des iterativen Kontrollalgorithmus zu senken.

4.3 Iterativ-optimierende Kontrolle

Nachdem mit dem Attributflußgraphen nunmehr eine für die massiv–parallele Instantiierung geeignete Repräsentation der Wissensbasis vorliegt, beschreiben wir im folgenden dessen Nutzung bei der Analyse. Der bisher vorgenommenen Unterscheidung der Netzwerkebene von der übergeordneten Kontrolle folgend, konzentrieren wir uns zunächst auf die datengetriebene Abarbeitung des Attributflusses. Diese dient zur Berechnung einer (anfänglichen) symbolischen Beschreibung, und ist in den anschließend beschriebenen Iterationsprozeß zur Optimierung der Interpretation eingebettet.

4.3.1 Datengetriebene Instantiierung

Durch die Konstruktion des Attributflusses als Graph, der den Informationsfluß zwischen den Analyseprozeduren der Wissensbasis beschreibt, ist der Ablauf der datengetriebenen Instantiierung bereits weitestgehend festgelegt: Die Verarbeitung beginnt mit den Knoten, die keine Vorgänger besitzen, und endet bei den Knoten ohne Nachfolger. Zur Instantiierung der Knoten werden die Analyseprozeduren der zugehörigen Netzwerkelemente aktiviert,

sobald die Berechnung der Vorgängerknoten beendet ist. In Abhängigkeit von der Aufgabe eines Knotens sind folgende Besonderheiten zu beachten:

- Zur Reduktion der Knotenanzahl sind beide Analyseprozeduren aus der Attributbeschreibung (Attributwerteberechnung und –bewertung) an einen einzigen Flußgraphknoten gebunden. Attributknoten werden daher in zwei Schritten instantiiert, wobei zunächst die Attributwerte, und anschließend deren Bewertungen zu berechnen sind.

- Im Gegensatz zum ERNEST–Kontrollalgorithmus, der ein *komplettes Objekt* (eine Instanz oder ein modifiziertes Konzept inclusive der berechneten Attribute und Relationen, vgl. Gleichung (3.4) und Bild 4.4) an die zuständige Analyseprozedur übergibt, sind in Algorithmus 4.3 als Argumente der Kantenbewertung im Attributfluß lediglich die *Bewertungen* von Instanzen zugelassen.

- Die Bewertung von Konzeptknoten muß für jede der durch eine Modalitätsmenge beschriebenen Begriffsvarianten durchgeführt werden.

Während die Zusammenfassung von Attributberechnung und –bewertung in einem Knoten als eher technisches Detail zu verstehen ist, stellt die Modifizierung der Schnittstelle zur Kantenbewertung einen — bei der automatischen Generierung des Attributflußgraphen unumgänglichen — Verlust an Kontrollmöglichkeiten dar, da eine dedizierte Bewertung von Kanten anhand einzelner Teilergebnisse nicht mehr möglich ist. So gestattet es der ERNEST–Kontrollalgorithmus, in einer Analyseprozedur zur Kantenbewertung über entsprechende Zugriffsfunktionen die Attributwerte oder Relationsbewertungen des übergebenen Objekts zu erfragen, um eine zusätzliche Überprüfung der berechneten Werte bezüglich des übergeordneten Konzepts vorzunehmen. Da sich ein derartiger Zugriff jedoch nicht in der Netzwerksyntax manifestiert, können die resultierenden Abhängigkeiten der Flußgraphknoten nicht automatisch bestimmt werden. Bei der Konstruktion des Attributflußgraphen kann daher lediglich eine Standardaufgabe der Kantenbewertung im Analyseprozeß — die Weitergabe der Bewertungen von Subkonzepten zur Beurteilung der Gesamtaussage des übergeordneten Konzepts — berücksichtigt werden.

Die mehrfache Bewertung von Konzeptknoten ist neben der Verarbeitung mehrdeutiger Segmentierungsergebnisse Ursache dafür, daß im allgemeinen mehrere Werte und Bewertungen zwischen den Knoten ausgetauscht werden; auf die somit notwendige Verarbeitung konkurrierender Hypothesen (Zwischenergebnisse) werden wir weiter unten eingehen, wollen uns aber vorher den Möglichkeiten der Parallelverarbeitung zuwenden.

Zur Parallelisierung der Bottom-Up–Analyse wird mittels Gleichung (4.2) für jeden Knoten v des Graphen der *Attributflußgrad*

$$\begin{aligned} \delta_{AFG}(v) &= \begin{cases} 0 & \text{falls } V_{pred} = \emptyset \\ \max\limits_{u \in V_{pred}} \{\delta_{AFG}(u)\} + 1 & \text{sonst} \end{cases} \\ V_{pred}(v) &= \{u \in V_{\mathcal{A}} | (u,v) \in E_{\mathcal{A}}\} \end{aligned} \tag{4.2}$$

berechnet, der die Länge des maximalen Kantenzuges zu einem Knoten ohne Vorgänger angibt und in Bild 4.6 am linken Rand eingetragen ist. Die Mengen $V_{\mathcal{A}}$ und $E_{\mathcal{A}}$ bezeichnen wie in Algorithmus 4.3 die Knoten– und Kantenmenge des Attributflußgraphen, dessen Tiefe d_A durch

$$d_A = \max_{v \in V_{\mathcal{A}}} \{\delta_{\mathcal{AFG}}(v)\} \tag{4.3}$$

definiert ist. Knoten gleichen Grades sind per constructionem voneinander unabhängig und können zu Ebenen parallel instantiierbarer Knoten zusammengefaßt werden. Die unterste Ebene des Graphen wird von den Knoten mit Grad 0 gebildet, welche Attribute repräsentieren, die direkt auf die Segmentierungsergebnisse zugreifen. Auf der obersten Ebene befinden sich im allgemeinen die Knoten zur Bewertung der Analyseziele.

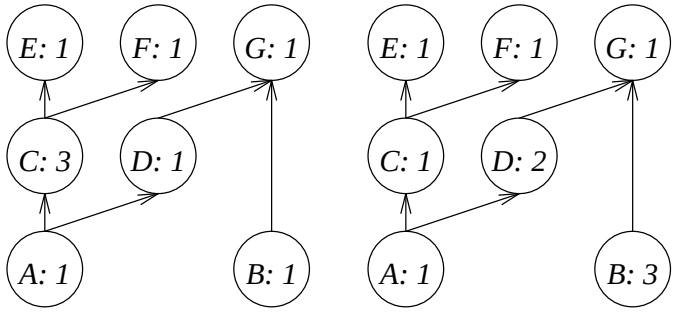
```
/* Bottom-Up-Instantiierung des Attributflußgraphen */
gegeben: ein Attributflußgraph AFG = (V_A, E_A),
         eine Menge Ξ = {ξ_v | v ∈ V_A} von Semaphoren
sei V_pred(v) := {u|(u,v) ∈ E_A} wie in Gleichung (4.2),
sei V_succ(v) := {w|(v,w) ∈ E_A}
∀ v ∈ V_A
    initialisiere ξ_v := −|V_pred(v)|
∀ v ∈ V_A DOPARALLEL
    wait(ξ_v)
    instantiiere_knoten(v)
    ∀ w ∈ V_succ(v)
        increment(ξ_w)
Ergebnis: instantiierter Attributflußgraph
```

Bild 4.8: Algorithmus zur parallelen Instantiierung des Attributflußgraphen.

Bild 4.8 zeigt den Algorithmus zur parallelen Instantiierung des Attributflußgraphen, welche mit Hilfe der auf der Menge Ξ der Semaphoren definierten unteilbaren Operationen wait und increment gesteuert wird. Während die wait–Operation die Instantiierung eines Knotens v solange blockiert, bis die zugehörige Semaphorvariable ξ_v den Wert $\xi_v = 0$ annimmt, sorgt die increment–Operation für eine Erhöhung der Variablen ($\xi_v = \xi_v + 1$), wodurch die Beendigung der Instantiierung eines Vorgängerknotens und somit die Erfüllung einer notwendigen Vorbedingung angezeigt wird. Die Prozedur instantiiere_knoten, die in

Algorithmus 4.10 näher spezifiziert wird, aktiviert die an einen Knoten gebundenen Analyseprozeduren aus der Wissensbasis und dient somit der eigentlichen Instantiierung eines Knotens; die für die Knoten der höheren Ebenen ($\delta_{AFG}(v) \neq 0$) festgelegte Strategie zur Behandlung alternativer Teilinterpretationen wird weiter unten diskutiert.

Die Dauer der parallelen, datengetriebenen Instantiierung ist proportional zur Tiefe d_A des Attributflußgraphen, sofern zum frühestmöglichen Berechnungszeitpunkt für jeden Knoten ein freier Prozessor zur Verfügung steht. Die Anzahl tatsächlich benötigter Prozessoren entspricht jedoch *nicht* der maximalen Anzahl von Knoten einer Ebene, wie dies etwa in Bild 4.6 suggeriert wird, sondern ist von den Knotenlaufzeiten abhängig. Bild 4.9 illustriert den Sachverhalt anhand zweier Flußgraphen mit identischer Struktur, für deren Knoten jedoch ein unterschiedlicher Zeitbedarf angenommen wird. Die parallele Abarbeitung beider Graphen ist tabellarisch (rechts) dargestellt. Im Falle des linken Graphen ist die



Graph 1 Graph 2

Zeit	Graph 1	Graph 2
0	A, B	A, B
1	C, D	C, D, B
2	C, G	E, F, D, B
3	C	G
4	E, F	$\emptyset$
5	$\emptyset$	$\emptyset$

Bild 4.9: Zur Erläuterung der massiv–parallelen Instantiierung: Zwei Attributflußgraphen mit identischer Struktur und verschiedenen Laufzeiten als Ursache für einen wechselnden Parallelitätsgrad.

tatsächlich Parallelität kleiner ($p = 2$), für den rechten Graphen dagegen größer ($p = 4$) als die maximale Knotenanzahl einer Ebene. Die hier verwendete *Zustandsraumanalyse* zur Bestimmung der parallelen Laufzeit wird bei der Evaluierung der Bottom-Up-Instantiierung in Kapitel 5 erläutert.

Anhand von Bild 4.11, welches das Prinzip der datengetriebenen Instantiierung veranschaulicht, wird im folgenden die Verarbeitung mehrdeutiger Segmentierungsergebnisse durch die Prozedur instantiiere_knoten aus Bild 4.10 diskutiert und die im nächsten Abschnitt beschriebene iterative Vorgehensweise motiviert. Im oberen Teil des Bildes ist nochmals der Attributflußgraph aus Bild 4.6 dargestellt, und im unteren linken Teil die Ergebnisse $s_1, \ldots, s_n$ einer initialen Segmentierung (hier: die Segmente $s_1, \ldots, s_8$ aus Beispiel 3.14). Zuordnungen zwischen Segmenten und den initialen Attributflußknoten auf der Ebene 0 des Graphen entstehen durch die Aktivierung der Prozeduren zur Attributwerteberechnung φ_A und zur –bewertung G_A und sind durch Punkte in der matrixförmigen Anordnung gekennzeichnet; der unterschiedliche Schwärzungsgrad symbolisiert die Bewer-

/* *Instantiierung eines Attributflußknoten* */			
gegeben: Attributflußknoten $v \in V_{\mathcal{A}} = V_{conc} \cup V_{att} \cup V_{rel} \cup V_{edge}$ $V_{pred} = \{u_1, u_2, \ldots, u_n\}$ wie in Gl. (4.2)			
initialisiere eine Ergebnismenge $R(v) := \emptyset$			
IF	$\delta_{AFG}(v) = 0$ /* *initialer Knoten* */		
THEN	/* *instantiiere v mit den Segmentierungsergebnissen* */		
	IF	$v \in V_{att}$ /* *Attributknoten* */	
	THEN	bestimme die Prozeduren zur Werteberechnung φ_A und zur Bewertung G_A aus v	
		berechne $R(v) = \{(\varphi_A(), G_A(\varphi_A()))_i \mid 1 \leq i \leq k\}$	
	ELSE	bestimme die Bewertung G_X aus v, $X \in \{C, S, E\}$, vgl. Alg. 4.3	
		berechne $R(v) = \{(G_X())_i \mid 1 \leq i \leq k\}$	
ELSE	/* *instantiiere v mit den Hypothesen der Vorgängerknoten* */		
	setze $Q(v) := R(u_1) \times \ldots \times R(u_n)$ $= \{(r_1, \ldots, r_n) \mid r_1 \in R(u_1), \ldots r_n \in R(u_n)\}$		
	$\forall \quad q \in Q(v)$		
	IF	$v \in V_{att}$ /* *Attributknoten* */	
	THEN	bestimme die Prozeduren zur Werteberechnung φ_A und zur Bewertung G_A aus v	
		$R(v) := R(v) \cup \{(\varphi_A(q), G_A(\varphi_A(q)))_i \mid 1 \leq i \leq k\}$	
	ELSE	bestimme die Bewertung G_X aus v, $X \in \{C, S, E\}$	
		$R(v) := R(v) \cup \{(G_X(q))_i \mid 1 \leq i \leq k\}$	
Ergebnis: Menge $R(v)$ mit konkurrierenden Hypothesen für den Knoten v			

Bild 4.10: Algorithmus zur Instantiierung eines Attributflußknoten. Erläuterungen finden sich im Text.

tungen der so erzeugten Hypothesen, welche die Mengen $R(v)$ in Algorithmus 4.10 bilden. Unter Beachtung der in der Attributbeschreibung eingetragenen Restriktionen werden hier beispielsweise drei Hypothesen für den Knoten "V1:laenge" generiert, wozu die Segmente s_1, s_2, und s_4 verwendet werden. Da das Segment s_3 zwar die Einschränkung für die Steigung, nicht jedoch für die Länge einer vertikalen Linie erfüllt (vergleiche dazu auch den auf Seite 69 skizzierten Instantiierungsprozeß in ERNEST), muß für dieses Segment keine Längenhypothese generiert werden. Die — im vorliegenden Beispiel nicht notwendige — Möglichkeit, Segmentierungsergebnisse bei der Instantiierung der Knoten zusammenzufassen, ist für den linken Knoten durch eine gestrichelte Ellipse angedeutet.

Nach der parallelen Instantiierung der initialen Knoten können deren Nachfolger berechnet werden. Konkurrierende Hypothesen aus der initialen Zuordnung werden in Algorithmus 4.10 auf den nachfolgenden Ebenen des Attributflußgraphen durch die wiederholte Instantiierung der Knoten behandelt, wozu alle möglichen Kombinationen von Eingangswerten der Berechnung in der Menge $Q(v)$ abgelegt werden. Im Falle der Attribut– oder

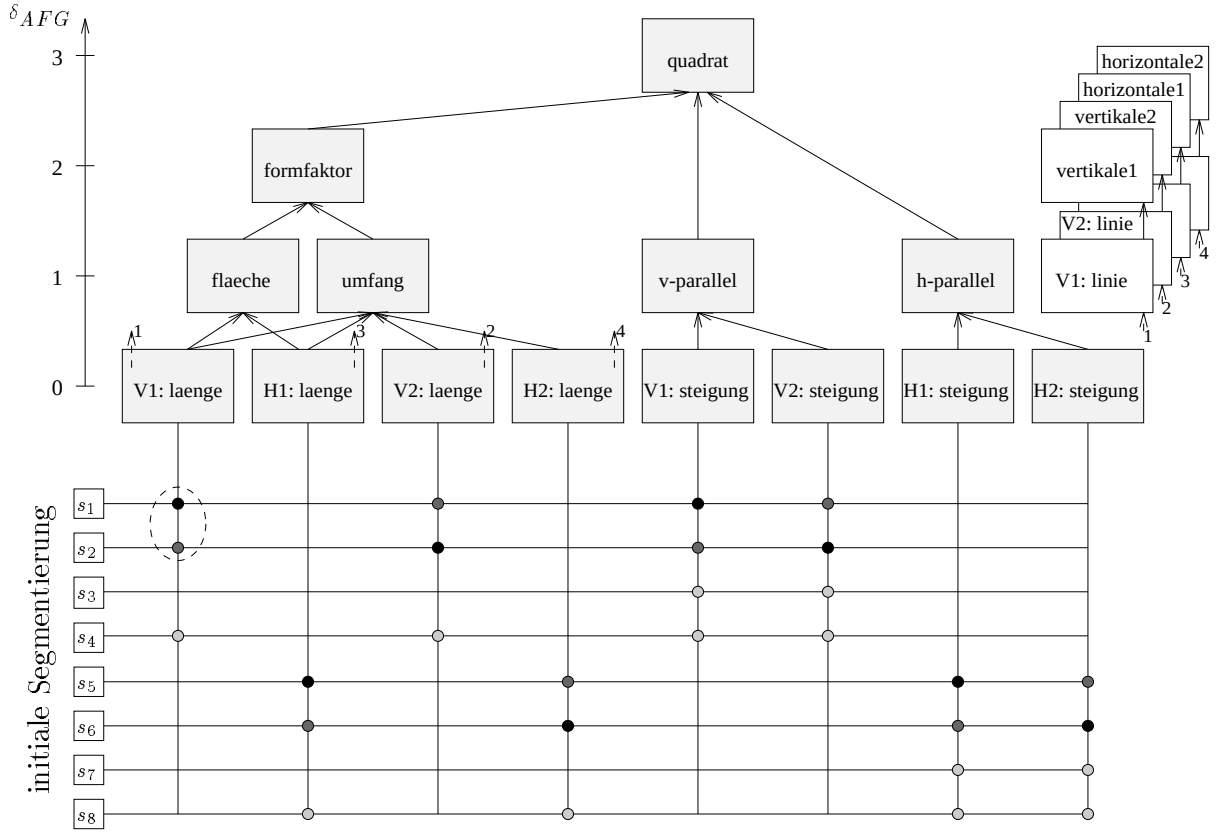


Bild 4.11: Das Prinzip der datengetriebenen Instantiierung des Attributflußgraphen.

Relationsknoten werden hierzu die Attributwerte aus $q \in Q$ an die Werteberechnung φ_A oder die Relationsbewertung G_S übergeben; bei einer Konzept– oder Kantenbewertung besteht q dagegen ausschließlich aus Bewertungen von Substrukturen oder Instanzen. Die Anwendung von Algorithmus 4.10 auf sämtliche Knoten liefert schließlich die konkurrierenden Bewertungen der im Attributflußgraph vorliegenden Zielkonzepte.

Es ist offensichtlich, daß die Instantiierung eines Knotens um so länger dauert, je mehr Hypothesen von den Vorgängerknoten erzeugt wurden. Da die Knoten der Ebene 0 unabhängig voneinander instantiiert und somit keine Zwischenergebnisse zur Einschränkung der initialen Zuordnungen verwendet werden können, entstehen zusätzliche Hypothesen, die erst im Verlauf der Analyse als inkonsistent verworfen werden können, und es besteht die Gefahr der kombinatorischen Explosion von Zwischenergebnissen auf den oberen Ebenen des Graphen. So müßten in Bild 4.11 aus den je drei Hypothesen für die Linienlängen bereits $3 \cdot 3 = 9$ Kombinationen von Hypothesen zur Berechnung des Knotens "flaeche" und — aufgrund der größeren Anzahl an Vorgängerknoten — $3 \cdot 3 \cdot 3 \cdot 3 = 81$ Kombinationen für "umfang" berücksichtigt werden. Von letzteren wären $3 \cdot 2 \cdot 3 \cdot 2 = 36$ weiterzuverfolgen; die restlichen dürften als inkosistent ausgeschlossen werden, da sie ein und dasselbe

Liniensegment zur Instantiierung von zwei Flußgraphknoten verwenden.

Eine erste Realisierung der Instantiierungsphase [Fis93a] in der eine konkurrierende Verarbeitung alternativer Hypothesen gemäß Algorithmus 4.10 vorgesehen war, hat gezeigt, daß die hier beschriebenen Nachteile durch Strategien zur Beschneidung von Teilinterpretationen (*Pruning*) nur teilweise kompensiert werden können. Eine ausführlichere Diskussion der erzielten Ergebnisse erfolgt nach der Vorstellung des Anwendungsszenarios in Kapitel 5.

Die in Abschnitt 4.1 festgelegten Entwurfsziele (günstiges Kommunikationsverhalten, Speichereffizienz und Any–Time–Eigenschaft) sind somit nur für Attributflußgraphen mit geringer Tiefe und Knotenanzahl und — ähnlich wie die Parallelisierung des Netzflusses (vgl. Seite 71) — nur im Falle einer qualitativ hochwertigen Segmentierung zu erreichen. Im folgenden Abschnitt wird daher ein iterativer Kontrollalgorithmus entwickelt, welcher in jedem Iterationsschritt lediglich eine Hypothese aus jedem Knoten verarbeitet, dadurch einen geringen Kommunikationsaufwand verursacht und ein effzientes Verhalten bezüglich der Speicheranforderung zeigt.

4.3.2 Iterative Optimierung

Als Ziel der wissensbasierten Interpretation komplexer Muster wurde bereits in Abschnitt 1.2 die Erstellung einer symbolischen Beschreibung der Sensordaten genannt, welche

- die für den Anwender relevante Information enthält,
- eine maximale Verträglichkeit mit dem in der Wissensbasis abgelegten Wissen aufweist, und
- optimal zu den Ergebnissen der ikonischen Verarbeitung paßt.

Die Gleichung

$$\begin{aligned} \mathcal{B}(\boldsymbol{f}(\boldsymbol{x})) &= I^*(C_{g_i}) \\ I^*(C_{g_i}) &= \underset{\{I(C_{g_i})\}}{\operatorname{argmax}}\{G(I(C_{g_i}))|\mathcal{M},\mathcal{A}\} \end{aligned} \tag{4.4}$$

charakterisiert die gestellten Anforderungen im Vokabular des ERNEST–Netzwerkformalismus [Fis93b, Sag93]: *unter Verwendung des Modells $\mathcal{M}$ und der initialen symbolischen Beschreibung $\mathcal{A}$* ist eine (die) *Instanz I^* zu einem Zielkonzept C_g* zu generieren, die eine *maximale Bewertung G* besitzt. Methoden zur Lösung der hier vorliegenden Optimierungsaufgabe werden durch Gleichung (4.4) nicht festgelegt; als eine Möglichkeit hatten wir in Abschnitt 2.3 die heuristische Graphsuche auf der Basis des A^*-Algorithmus kennengelernt.

Konkurrierende Instanzen zu den Zielkonzepten werden während der datengetriebenen Instantiierung des Attributflußgraphen berechnet. Die im letzten Abschnitt geführte Diskussion hat gezeigt, daß jeder Instanz genau eine Zuordnung zwischen initialen Knoten

und Segmentierungsergebnissen sowie die Bewertung der Konzeptknoten bezüglich genau einer Modalitätsmenge zugrundeliegt. Dies ermöglicht uns, für jeden initialen Knoten eine Hypothese $r_n^{(i_n)}, 1 \leq n \leq l$, und für jedes Konzept eine Modalitätsmenge $h_m^{(j_m)}, 1 \leq m \leq k$, zu wählen, und den aktuellen *Analysezustand* als Vektor

$$\boldsymbol{r} \quad := \quad (r_1^{(i_1)}, \ldots, r_l^{(i_l)}, h_1^{(j_1)}, \ldots, h_k^{(j_k)}) \tag{4.5}$$

zu definieren, wobei l die Anzahl der Knoten v mit Grad 0 ($\delta_{AFG}(v) = 0$) ist, und k die Anzahl der Konzeptknoten im Attributflußgraphen angibt. Der Graphaus Beispiel 4.10 besitzt acht Knoten auf der Ebene 0, jedoch keine Konzeptknoten, die bezüglich mehrerer Modalitätsmengen zu bewerten sind. Legen wir — wie weiter oben bereits geschehen — die Restriktionen aus Bild 3.8 und die Segmentierungsergebnisse aus Bild 3.14 zugrunde, so werden je drei konkurrierende Hypothesen für die vier Knoten zur Repräsentation der Linienlänge und je vier Hypothesen für die Steigungen erzeugt; die Menge der Analysezustände besitzt somit $3^4 \cdot 4^4 = 20736$ Zustände.

Werden bei der datengetriebenen Instantiierung mit Hilfe von Algorithmus 4.8 bzw. 4.10 lediglich die in $\boldsymbol{r}$ angegebenen Hypothesen und Modalitätsmengen berücksichtigt, so ist bei der Berechnung jedes Knotens v mit $\delta_{AFG} > 0$ genau eine Belegung der Argumente der Analyseprozedur(en) zu beachten ($|Q(v) = 1$, vgl. Bild 4.10). Als Konsequemz daraus ist zunächst festzuhalten, daß die auf Seite 98 für die simultane Behandlung konkurrierender Hypothesen skizzierte kombinatorische Explosion von Zwischenergebnissen vermieden wird; als Resultat der Instantiierung eines Knotens erhalten wir genau eine Bewertung (im Falle eines Relations–, Kanten– oder Konzeptknoten) oder einen bewerteten Attributwert, falls ein Attributknoten instantiiert wurde.

Insbesondere für die Knoten, welche die Analyseziele C_{g_i} repräsentieren, erhalten wir somit je eine Instanzbewertung $G(I(C_{g_i}))$, und gemäß Gleichung (4.4) ist die Instanz mit der maximalen Bewertung als Interpretation der Sensordaten auszuwählen. Da die Bewertungen offensichtlich ihrerseits von dem durch $\boldsymbol{r}$ zusammengefaßten Analysezustand abhängen, ist es naheliegend, letzteren mit einem Bewertungsvektor

$$\boldsymbol{g}(\boldsymbol{r}) := (G(I(C_{g_1})), \ldots, G(I(C_{g_n}))) \tag{4.6}$$

zu versehen, und $\boldsymbol{r}$ so zu bestimmen, daß eine Bewertung $G(I(C_{g_i})$ maximiert wird. Die folgende Definition [Aar89] erlaubt es, diese Aufgabe als ein *kombinatorisches Optimierungsproblem* zu identifizieren:

Ein kombinatorisches Optimierungsproblem ist ein Tupel

$$COP \quad := \quad (Z, \phi, Z_{min}) \tag{4.7}$$

das aus einer *Zustandsmenge* Z, einer *Kostenfunktion* $\phi : Z \mapsto \Re$ und einer *Lösungsmenge* Z_{min} besteht. Die Lösungsmenge Z_{min} ist durch

$$Z_{min} \subset Z \quad \wedge \quad z_i \in Z_{min} :\Leftrightarrow \forall z_j \in Z \ : \ \phi(z_i) \leq \phi(z_j) \tag{4.8}$$

gegeben.

Die hier vorgenommene Definition als Minimierungsaufgabe erfolgt ohne Beschränkung der Allgemeinheit und entspricht dem weiter unten tatsächlich eingeschlagenen Vorgehen. Zur Überführung in ein Maximierungsproblem ist der Begriff der *Kosten* durch den der *Bewertung* zu ersetzen, und eine Lösungsmenge Z_{max} ist durch

$$Z_{max} \subset Z \quad \wedge \quad z_i \in Z_{max} :\Leftrightarrow \forall z_j \in Z \ : \ \phi(z_i) \geq \phi(z_j) \tag{4.9}$$

zu definieren. Eine Analogie zur graphbasierten Suche ergibt sich, wenn wir die Kostenfunktionen $\phi(z)$ und $\varphi(v)$ aus den Gleichungen (4.8) und (3.20) identifizieren. Beide Funktionen evaluieren einen wohldefinierten Analysezustand — zum einen die im Vektor $\boldsymbol{r}$ zusammengefaßten Entscheidungen bei der Instantiierung des Attributflußgraphen, zum anderen die aus Instanzen und modifizierten Konzepten bestehenden Suchbaumknoten — und stellen somit Alternativen zur Bestimmung einer Instanz gemäß (4.4) dar.

Bevor wir uns den Verfahren zur Lösung der durch (4.7 – 4.8) definierten Optimierungsaufgaben zuwenden, soll zunächst die im weiteren Verlauf der Arbeit verwendete Kostenfunktion ϕ eingeführt werden. Wie bereits durch Gleichung (4.6) beschrieben, ist das Ergebnis der datengetriebenen Instantiierung ein Vektor $\boldsymbol{g}$, der für jedes Zielkonzept eine Instanzbewertung $G(I(C_{g_i}))$ enthält. Um anhand dieser Bewertungen zu entscheiden, welches Analyseziel in den Sensordaten vorliegt, nehmen wir an, daß eine fehlerfreie Segmentierung eine eindeutige Interpretation zulassen würde. In diesem Fall wird der Vektor $\boldsymbol{g}$ in derjenigen Komponente einen maximalen Wert annehmen, die dem tatsächlich vorhandenen Zielkonzept entspricht; Instanzen zu den übrigen Zielkonzepten werden hingegen schlecht bewertet, und die entsprechenden Komponenten von $\boldsymbol{g}$ nehmen kleine Werte an. Normieren wir die Bewertungen auf Werte aus dem Intervall $0 \leq G(I(C_{g_i})) \leq 1$, so erhalten wir als Ergebnis der Instantiierung im Idealfall den i-ten Einheitsvektor $\boldsymbol{e}_i$, falls das i-te Zielkonzept die Sensordaten interpretiert. Die Approximation dieser idealen "Trennfunktion" legt nun die Einführung der Kostenfunktion

$$\begin{aligned} \tilde{\boldsymbol{g}}(\boldsymbol{r}) &= \frac{\boldsymbol{g}(\boldsymbol{r})}{||\boldsymbol{g}(\boldsymbol{r})||} \\ \phi(\boldsymbol{r}) &= \min_{1 \leq i \leq n} ((\boldsymbol{e}_i - \tilde{\boldsymbol{g}}(\boldsymbol{r}))^2) \end{aligned} \tag{4.10}$$

nahe, welche die Kosten der Instantiierung des Attributflußgraphen unter Verwendung der

im Analysezustand $\boldsymbol{r}$ zusammengefaßten Entscheidungen angibt. In Beispiel 4.6 ist genau ein Zielkonzept ("Quadrat") vorhanden, wodurch sich die Kostenminimierung gemäß (4.10) auf die Bestimmung desjenigen Zustandsvektors $\boldsymbol{r}$ reduziert, der eine maximal bewertete Instanz $I(\text{Quadrat})$ liefert.

Nachdem nunmehr eine *problemunabhängige* Kostenfunktion für die kombinatorische Optimierung eingeführt ist, können wir uns im folgenden mit den Verfahren zu deren Minimierung befassen.

Lösung kombinatorischer Optimierungsprobleme

Für die meisten "klassischen" Aufgabenstellungen der kombinatorischen Optimierung, wie zum Beispiel das Traveling-Salesman–Problem (vgl. [Pap82]), existieren keine deterministischen Verfahren, die in polynomialer Zeit eine Lösung liefern; sie gehören zur Klasse der NP–schweren Probleme, zu deren Bearbeitung sich — in Abhängigkeit von den Erfordernissen der konkreten Anwendung — zwei Vorgehensweisen anbieten:

- Man besteht auf einer optimalen Lösung und nimmt dafür sehr lange Rechenzeiten in Kauf. Exakte Lösungen von Optimierungsproblemen liefern beispielsweise Verfahren wie die Dynamische Programmierung, der A^*–Algorithmus oder auch eine – aus Aufwandsgründen in der Regel nicht durchführbare – blinde Suche.

- Man interessiert sich für ein möglichst gutes, suboptimales Ergebnis, das überdies schnell zur Verfügung stehen soll. Hierzu bieten sich die im folgenden beschriebenen iterativen Verfahren an, die attraktiv sind, weil sie – unter gewissen Voraussetzungen – die in Abschnitt 4.1 geforderte Any–Time–Eigenschaft besitzen: die Qualität der gefundenen Lösung steigt mit der Anzahl der Iterationen, und falls genügend viele Iterationen durchgeführt werden können, ist das Erreichen einer optimalen Lösung garantiert.

Für die Optimierung einer Interpretation durch den Kontrollalgorithmus eines Musteranalysesystems bietet sich die letztgenannte Alternative an, da im allgemeinen innerhalb einer vorgegebenen Zeitschranke eine (möglichst gute) Lösung erzielt werden muß. Allgemeine Algorithmen sind dabei speziellen, problemabhängigen Verfahren vorzuziehen, um den Einsatz in unterschiedlichen Anwendungsbereichen zu ermöglichen. Einen wichtigen Schritt in diese Richtung stellt die oben vorgestellte, problemunabhängige Definition der Kostenfunktion dar, die keinerlei Annahmen über den Inhalt der Wissensbasis macht.

Bild 4.12 zeigt einen Algorithmus zur Minimierung der Kostenfunktion eines kombinatorischen Optimierungsproblems, der zugleich das Gerüst des iterativ–optimierenden Kontrollalgorithmus ist:

- Die Zustände z_i entsprechen den Analysezuständen $\boldsymbol{r}$ aus Gleichung (4.5).

- Die Berechnung der Kosten $\phi_n = \phi(z_n)$ erfolgt durch die datengetriebene Instantiierung des Attributflußgraphen unter Verwendung eines Zustandes $z = \boldsymbol{r}$ und der anschließenden Anwendung von Gleichung (4.10).

/* *Algorithmus zur Minimierung einer Kostenfunktion* */		
gegeben: Zustandsmenge $Z = \{z_0, z_1, z_2, \ldots\}$, Kostenfunktion $\phi : Z \mapsto \Re$		
bestimme eine Kontrollsequenz (T_k) mit den folgenden Eigenschaften: $T_k > 0 \wedge \forall k \geq 0 : T_k \geq T_{k+1} \wedge \lim_{k\to\infty} T_k = 0$		
bestimme einen initialen Zustand z_0 und berechne $\phi_0 := \phi(z_0)$		
$z_c := z_0$; $\phi_c := \phi_0$		
$\forall$ T_k		
	$z_n :=$ erzeuge_zustand(z_c)	
	$\phi_n := \phi(z_n)$ /* *Berechnung der Kosten* */	
	accept := akzeptiere_zustand(ϕ_c, ϕ_n, T_k)	
	IF	*accept* = TRUE
	THEN	/* *übernehme neuen Zustand und Kosten* */
		$z_c := z_n$, $\phi_c := \phi_n$
	ELSE	/* z_n *wird zurückgewiesen* */
Ergebnis: Zustand z_c ist Lösung mit den Kosten ϕ_c		

Bild 4.12: Algorithmus zur Lösung kombinatorischer Optimierungsprobleme.

Die Bedeutung der hervorgehobenen Anweisungen des Algorithmus wird vor deren Spezifizierung in den folgenden Abschnitten anhand von Bild 4.13 diskutiert, das eine Kostenfunktion und vier ausgewählte Zustände, den aktuellen Zustand z_c, die Folgezustände z_n und z_m, sowie einen Zustand z_{min} aus der Lösungsmenge zeigt. Besitzt der aktuelle Zustand z_c die Kosten ϕ_c, so erscheint es zunächst sinnvoll, als Folgezustand stets einen Zustand zu wählen, der geringere Kosten verursacht; falls ϕ differenzierbar ist, bietet sich der Abstieg entlang des Gradienten an. Ausgehend von z_c wird bei diesem Vorgehen der Zustand z_n erreicht, in dem die Kostenfunktion ein *lokales* Minimum annimmt. Als Konsequenz aus der Strategie, nur Zustände mit geringeren Kosten zuzulassen, bleibt der Gradientenabstieg hier jedoch stecken, und das globale Minimum z_{min} wird nicht erreicht. Um die "Kostenbarriere" zwischen z_n und z_{min} zu überwinden, ist es daher notwendig, gelegentlich auch Zustände mit größeren Kosten einzunehmen; in Bild 4.13 beispielsweise den Zustand z_m. Dieses Vorgehen gestattet es zwar, das lokale Minimum zu verlassen, doch da dies gleichermaßen für das globale Minimum gilt, ist apriori nicht klar, ob die Suche in einem Zustand der Lösungsmenge terminiert.

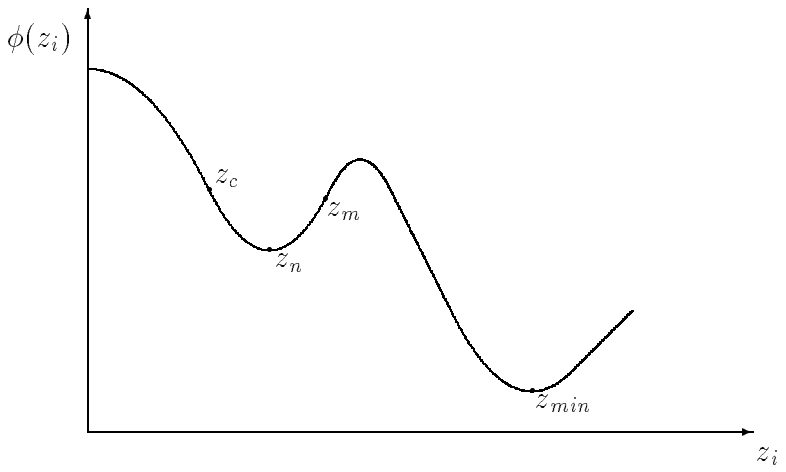


Bild 4.13: Kostenfunktion zur Erläuterung von Algorithmus 4.12.

Die im folgenden vorgestellten Verfahren ermöglichen die Konvergenz in einem globalen Optimum durch die Verwendung einer *Kontrollsequenz*, deren aktueller Wert die Annahme eines *zufällig generierten* Folgezustandes mit größeren Kosten steuert. Die hier skizzierte Grundidee werden wir zunächst anhand des bekannten *Simulated-Annealing*–Algorithmus erläutern; im Anschluß daran beschreiben wir verschiedene Varianten des Verfahrens, und diskutieren den Einsatz *genetischer Algorithmen* als weitere Möglichkeit zur Lösung der durch Gleichung (4.4) definierten Optimierungsaufgabe.

Simulated Annealing

Unter dem Oberbegriff "Simulated Annealing" zusammengefaßte Algorithmen zur Lösung kombinatorischer Optimierungsprobleme gehen auf eine Arbeit von Metropolis zurück, die sich mit der Simulation des Übergangs eines Festkörpers ins thermische Gleichgewicht befaßt [Met53].

Um einen Festkörper in einen Zustand minimaler Energie zu zwingen, wird dieser zunächst durch starkes Erhitzen in den flüssigen Aggregatzustand gebracht. Im Zustand hoher Temperatur können sich die Atome des Körpers frei bewegen, da sie an keine Gitterplätze gebunden sind. Aufgrund ihrer freien Beweglichkeit nehmen die Teilchen eine zufällige Struktur ein; sie besitzen eine hohe (Bewegungs-)energie, die zu einer großen Energie des Gesamtsystems führt. Wird der Körper anschließend abgekühlt, so kann man beobachten, daß die Atome feste Positionen in einem hochstrukturierten Gitter einnehmen. Aufgrund ihrer Unbeweglichkeit besitzen die Teilchen — und somit das Gesamtsystem — eine sehr geringe Energie. Essentiell für das Erreichen eines Zustandes minimaler Energie (Grundzustand) ist eine sehr hohe Starttemperatur, die die zufällige Position der Teilchen bewirkt, und ein sehr langsames Abkühlen. Bei einem zu schnellen Absenken der Tem-

peratur würde der Körper schließlich in einem ungeordneten Zustand verharren, der kein Energieminimum darstellt.

Der Metropolis–Algorithmus simuliert das skizzierte Abkühlen eines Festkörpers. Die Wechselwirkung der Atome wird durch eine *Energiefunktion* u beschrieben, die ihr absolutes Minimum annimmt, wenn sich der Körper im geordneten Grundzustand befindet. Ausgehend von einem Zustand z_i mit hoher Energie $U_i = u(z_i)$ wird ein Folgezustand z_j mit Energie U_j durch eine geringfügige Änderung eines *zufällig ausgewählten* Parameters des Systems erzeugt. Der neue Zustand z_j wird mit einer Wahrscheinlichkeit

$$P(T) = \begin{cases} 1 & \text{falls} \quad U_j - U_i \leq 0 \\ \exp(-\frac{U_j - U_i}{T}) & \text{sonst} \end{cases} \tag{4.11}$$

beibehalten. Gleichung (4.11) beschreibt das bekannte *Metropolis–Kriterium*, das besagt, daß Zustände mit geringerer Energie ($\Delta U = U_j - U_i \leq 0$) immer akzeptiert werden, und bei Folgezuständen mit größerer Energie die künstliche Temperatur T und die Energiedifferenz zur Beeinflussung der Wahrscheinlichkeit verwendet werden. Das Abkühlen des Körpers, das durch eine schrittweise Verringerung von T simuliert wird, führt dazu, daß $P(T)$ gegen 0 strebt, wenn der Körper sich bereits nahe am Grundzustand befindet.

In [Kir82, Kir83] wird der Metropolis-Algorithmus auf die Lösung von Kostenoptimierungsproblemen übertragen:

- Die Zustände des Festkörpers entsprechen der Zustandsmenge Z des Optimierungsproblems aus Gleichung (4.7), und die Zustände minimaler Energie stellen die Lösungsmenge Z_{min} dar.
- Die Energie U des Festkörpers entspricht den Kosten ϕ der Zustände des Optimierungsproblems.
- Die Temperatur dient als Kontrollparameter, der die Annahme eines Zustandes beeinflußt.

Übersichten über die zahlreichen Anwendungen des Algorithmus, die einen Schwerpunkt im Bereich des rechnergestützen Entwurfs hochintegrierter Schaltungen besitzen (z.B. [Rut86, Kra87]), finden sich in [Laa87, Aar87, Aar89]. Im Bereich der Bildverarbeitung existieren Ansätze zur Bildrestauration [Gem84], zur Segmentierung [Son85] und zur Bestimmung korrespondierender Bildpunkte in Stereobildpaaren [Kol90]; Probleme der Wissensrepräsentation und –nutzung mit einem eng verwandten Ansatz, der *Boltzmann–Maschine*, werden in [Hin86, Der90] behandelt.

Durch die aufgezeigten Parallelen können nun das Annahmekriterium (akzeptiere_zustand), die Bestimmung der Kontrollsequenz (T_k) und die Erzeugung von Folgezuständen (erzeuge_zustand) des allgemeinen Lösungsverfahrens aus Bild 4.12 für eine Kostenoptimierung

mittels Simulated Annealing spezifiert werden. Dabei liegen sowohl den Verfahren zur Zustandsänderung und –annahme als auch der Bestimmung einer geeigneten Kontrollsequenz tiefgreifende Erkenntnisse aus der Theorie der *Markovketten* zugrunde, die den mathematischen Apparat zur Beschreibung des Algorithmus bilden (z.B. [Sen81, Mat90a]). Insbesondere ist hervorzuheben, daß die oben aufgeworfene Frage nach der Terminierung des Verfahrens in einem globalen Minimum positiv beantwortet werden kann; Konvergenzbeweise sind in der Literatur weit verbreitet [Gem84, Laa87, Aar87, Aar89] , sollen in dieser Arbeit jedoch aus Gründen der Lesbarkeit nicht nachvollzogen werden.

Eine Zustandsannahme gemäß Gleichung (4.11) wird durch das Annahmekriterium

$$accept \quad = \quad \begin{cases} \texttt{TRUE} & \text{falls} \quad \Delta\phi \leq 0 \\ \texttt{TRUE} & \text{falls} \quad \Delta\phi > 0 \wedge q \leq \exp(-\Delta\phi/T_k) \\ \texttt{FALSE} & \text{sonst} \quad \Delta\phi > 0 \wedge q > \exp(-\Delta\phi/T_k) \end{cases} \tag{4.12}$$

realisiert. Die zufällige Entscheidung im Falle einer positiven Kostendifferenz $\Delta\phi = \phi_n - \phi_c$ wird dabei durch den Vergleich von $exp(-\Delta\phi/T_k)$ mit einer Zufallszahl q aus einer im Intervall $[0,1]$ uniform verteilten Grundgesamtheit herbeigeführt.

Zustandsänderungen werden durch kleine, stochastische Verzerrungen des aktuellen Zustands erreicht, und durch eine *stochastische Übergangsmatrix* $\boldsymbol{Q} = (q_{i,j})$ auf $Z \times Z$ definiert. $\boldsymbol{Q}$ muß den Bedingungen

$$q_{i,j} \quad = \quad q_{j,i} \tag{4.13}$$

$$\begin{aligned} \forall z_i, z_j \in Z \quad : \quad & \exists\ (n \geq 1)\ \exists\ z_0, \ldots, z_n \in Z (z_i = z_0 \wedge z_n = z_j) : \\ & q_{k,k+1} > 0, \quad k = 0, 1, \ldots, n-1 \end{aligned} \tag{4.14}$$

genügen, also *symmetrisch* (4.13) und *irreduzibel* (4.14) sein, wodurch jeder Zustand von jedem anderen durch eine endliche Zahl von Übergängen erreichbar ist. Zur Bestimmung einer geeigneten Übergangsmatrix legt man in der Praxis durch

$$(z_j \in \mathcal{V}_i \Leftrightarrow z_i \in \mathcal{V}_j) \wedge z_i \notin \mathcal{V}_i \tag{4.15}$$

zuerst eine geeignete Nachbarschaft $\mathcal{V}_i$ zum Zustand z_i fest, und definiert $\boldsymbol{Q}$ anschließend durch

$$q_{i,j} \quad = \quad \begin{cases} \frac{1}{|\mathcal{V}_i|} & \text{falls} \quad z_j \in \mathcal{V}_i \\ 0 & \text{sonst} \end{cases} . \tag{4.16}$$

Zur stochastischen Verzerrung des durch Gleichung (4.5) definierten Analysezustandes $z_c = \boldsymbol{r}$ greifen wir dieses Vorgehen auf. Bezeichnet $r_i^{(j)}$, $1 \leq i \leq l$, die j–te konkurrierende Hypothese zum i-ten Knoten der Ebene 0 des Attributflußgraphen, und analog dazu $h_i^{(j)}$,

$1 \leq i \leq k$, die j–te Modalitätsmenge des i–ten Konzeptknotens, so wird ein potentieller Folgezustand durch

$$\begin{aligned} &(r_1, \ldots, r_i^{(j_c)}, \ldots, r_l, h_1, \ldots, h_k) \rightarrow (r_1, \ldots, r_i^{(j_n)}, \ldots, r_l, h_1, \ldots, h_k) \\ &\text{oder} \\ &(r_1, \ldots, r_l, h_1, \ldots, h_i^{(j_c)}, \ldots, h_k) \rightarrow (r_1, \ldots, r_l, h_1, \ldots, h_i^{(j_n)}, \ldots, h_k) \end{aligned} \tag{4.17}$$

generiert. Da aufgrund von Gleichung (4.15) $j_c \neq j_n$ gilt, wird durch (4.17) in jedem Iterationsschritt entweder genau eine initiale Zuordnung oder eine Modalitätsmenge ersetzt. Ist ρ_i die Anzahl der konkurrierenden Hypothesen zum i–ten initialen Attributknoten, und entsprechend θ_j die Anzahl der — ebenfalls konkurrierenden — Modalitätsmengen zum j–ten Konzeptknoten, so hat die Zustandsmenge die Größe

$$|Z| \;=\; \prod_{i=1}^{l} \rho_i \cdot \prod_{j=1}^{k} \theta_j \tag{4.18}$$

(l und k sind wie in Gleichung 4.5 auf Seite 100 definiert) und die Größe der Nachbarschaft ist

$$|\mathcal{V}_i| \;=\; \sum_{i=1}^{l}(\rho_i - 1) + \sum_{j=1}^{k}(\theta_j - 1). \tag{4.19}$$

Wie bereits aus der physikalischen Motivation des Verfahrens ersichtlich, spielt der Entwurf der Kontrollsequenz, die aufgrund der Analogie auch als *Abkühlplan* (*cooling schedule*) bezeichnet wird, eine bedeutende Rolle für die Konvergenz des Algorithmus in einem globalen Minimum der Kostenfunktion. Kennzeichnend für die Kontrollsequenz sind der Startwert T_0, die Strategie zu dessen schrittweiser Verringerung, und ein Abbruchkriterium oder Endwert, bei dessen Erreichen der Algorithmus anhält. Eine ausführliche Diskussion der Parameterwahl und der zahlreichen vorhandenen Ansätze muß der Litertur vorbehalten bleiben, wozu wir insbesondere auf [Laa87, Kap. 5] verweisen; die folgende Darstellung beschränkt sich auf die Motivation und Erläuterung der in den experimentellen Untersuchungen in Kapitel 5 verwendeten einfachen Strategien.

Bereits in Algorithmus 4.12 wurde durch

$$T_k > 0 \wedge \forall k \geq 0 : T_k \geq T_{k+1} \wedge \lim_{k \to \infty} T_k = 0 \tag{4.20}$$

gefordert, daß die Sequenz (T_k) monoton fallend ist und gegen 0 konvergiert. Diese Basiseigenschaft aller Kontrollsequenzen bewirkt, daß mit zunehmender Anzahl an Iterationen die Wahrscheinlichkeit (4.11) für die Annahme eines Zustandes mit größeren Kosten (höherer Energie) sinkt. So würde in Bild 4.13 ausgehend vom Zustand z_n ein Übergang nach z_m mit großer Wahrscheinlichkeit akzeptiert, z_{min} jedoch aufgrund kleiner Werte von T_k (und

$P(T_k)$) nicht mehr verlassen werden. Zur Veranschaulichung verweisen wir auch auf Bild 4.15 auf Seite 112, welches den Nutzen der Bedingung (4.20) anhand des unten diskutierten *Schwellwertakzeptanz*-Verfahrens erläutert.

In [Aar85, Laa87] sind Verfahren zur Bestimmung des Startwertes T_0 angegeben. Dieser ist so zu wählen, daß anfangs nahezu alle Zustandsübergänge akzeptiert werden, wozu $\exp(-(\phi_i - \phi_j)/T_0) \approx 1$ für alle Zustände z_i, z_j gelten muß. Zu einer empirischen Bestimmung von T_0 wird zunächst die *Annahmerate*

$$\gamma \quad = \quad \frac{\text{Anzahl akzeptierter Übergänge}}{\text{Anzahl aller Übergänge}} \tag{4.21}$$

definiert. In [Kir83] werden eine anfängliche Annahmerate $\gamma_0 = 0.8$ und ein hoher Startwert T_0 vorgegeben und eine Anzahl von Zustandsübergängen ausgeführt. Ist die Annahmerate kleiner als γ_0, so wird T_0 verdoppelt und der Versuch erneut durchgeführt.

Der von uns verwendete Algorithmus greift das in [Joh86] beschriebene Verfahren auf, das wir nach [Laa87, S. 59–60] zitieren. Hierbei werden m zufällige Zustandsübergänge beobachtet, und aus den dabei auftretenden m_0 Verschlechterungen wird der mittlere Kostenzuwachs

$$\begin{aligned} \overline{\Delta\phi}^{(+)} &= 1/m_0 \sum_{i=1}^{m} \Delta_i^{(+)} \\ \Delta_i^{(+)} &= \begin{cases} \phi_n - \phi_c & \text{falls} \quad \phi_n - \phi_c > 0 \\ 0 & \text{sonst} \end{cases} \end{aligned} \tag{4.22}$$

berechnet. Zur Bestimmung von T_0 wird wiederum eine anfängliche Annahmerate $\gamma_0 \approx 1$ vorgegeben und durch eine einfache Umformung der Gleichung

$$\gamma_0 \quad = \quad \exp(-\overline{\Delta\phi}^{(+)}/T_0) \tag{4.23}$$

erhalten wir

$$T_0 \quad = \quad \frac{\overline{\Delta\phi}^{(+)}}{\ln(\gamma_0^{-1})} \tag{4.24}$$

als Startwert der Kontrollsequenz (T_k).

Neben einem großen Startwert T_0 ist ein langsames Absenken des Kontrollparameters essentiell für das Erreichen eines Zustandes, der ein globales Minimum der Kostenfunktion darstellt. In [Gem84] wird mit

$$\begin{aligned} \exists k_0 \geq 2\ \forall k \geq k_0 \quad &: \quad T_k \geq \frac{|Z| \cdot (\Delta\phi)_{max}}{\log(k)} \\ (\Delta\phi)_{max} \quad &= \quad \max_{i \in Z}\{\phi_i\} - \min_{i \in Z}\{\phi_i\} \end{aligned} \tag{4.25}$$

erstmals eine Bedingung für die Werte der Kontrollsequenz angegeben, welche die Konvergenz des Metropolis–Algorithmus garantiert. Kontrollsequenzen mit der hier geforderten *logarithmischen* Konvergenzgeschwindigkeit besitzen in der Praxis jedoch nur geringe Bedeutung, da die erforderlichen hohen Rechenzeiten nur selten zur Verfügung stehen; der in Gleichung (4.25) ausgedrückte Sachverhalt stellt jedoch die wesentliche Motivation für Untersuchungen zur Beschleunigung des Algorithmus durch problemspezifische Kontrollsequenzen und zur Parallelisierung des Verfahrens dar.

Exponentiell absinkende Abkühlpläne können zwar die Konvergenz nicht garantieren, werden jedoch in vielen Anwendungen mit begrenzter Rechenzeit eingesetzt. Eine Kontrollsequenz der Form

$$T_k \quad = \quad T_0 \cdot \alpha^k,\ \alpha \approx 1,\ k = 1, 2, \ldots \tag{4.26}$$

findet mit $\alpha = 0.95$ bereits in [Kir83] Verwendung. Ebenfalls diskutiert werden adaptive exponentielle Strategien, welche die Kontrollsequenz an die Anzahl N der bereits durchgeführten Iterationen anpassen. In [Aze92b, Cat92] wird dazu

$$\begin{aligned} T_k \quad &= \quad T_0 \cdot \alpha_N^k \\ \alpha_N \quad &= \quad (c \cdot \log(N))^{1/N},\ c = \text{const.} \end{aligned} \tag{4.27}$$

vorgeschlagen, und eine gewisse Robustheit derartiger Strategien bezüglich der Wahl der Konstanten T_0 und c festgestellt. Als Folgerung aus der Untersuchung verschiedener Methoden wird schließlich eine Anpassung der Strategie an die zur Verfügung stehende Rechenzeit und die aus der Anwendung resultierende Energiefunktion gefordert:

> "In conclusion [...] this theoretical study of annealing should encourage us to tune cooling schedules to each given application and each given amount of resources, and to free ourselves from any dogmatic preference for some 'closed formula'." [Cat92]

Diesen Vorschlag aufgreifend, verwenden wir die durch Gleichung (4.26) definierte Kontrollsequenz, und setzen für α den experimentell ermittelten Wert 0.95 ein.

Schließlich ist noch ein Abbruchkriterium zu bestimmen, bei dessen Erfüllung der Algorithmus beendet und der aktuelle Zustand als Lösung ausgegeben wird; im einfachsten Fall bestimmt dabei die zur Verfügung stehende Rechenzeit die Anzahl der auszuführenden Iterationen. Strebt (T_k) gegen 0, so nimmt die Anzahl der akzeptierten Zustandsänderun-

gen mit positiver Kostendifferenz $\Delta\phi$ ab. Ein Abbruchkriterium wird daher häufig mit Hilfe der Anzahl der nicht akzeptierten Übergänge ermittelt. In [Joh86] wird in Analogie zu (4.21–4.23) eine Schwelle γ_f definiert, und der Algorithmus bricht ab, wenn die erzielte Annahmerate kleiner als γ_f ist. Ein weiteres Kriterium ergibt sich aus einer Beobachtung der erzielten Kosten: ist der Wert der Kostenfunktion über eine bestimmte Anzahl $?_f$,

$$?_f \quad = \quad c \cdot |\mathcal{V}|, \quad c \in N, \tag{4.28}$$

von Zustandsübergängen konstant, so bricht der Algorithmus ab; $|\mathcal{V}|$ ist hierbei die für alle Zustände gleiche Größe der Nachbarschaft aus Gleichung (4.19). In unserer Implementierung des Algorithmus wird das Kriterium (4.28) als Voreinstellung bereitgestellt.

Muß aufgrund der großen Anzahl auszuführender Iterationen ohnehin auf die Garantie einer optimalen Lösung verzichtet werden, so bieten sich neben der Ersetzung der dazu erforderlichen logarithmischen Kontrollsequenzen durch suboptimale, exponentielle Abkühlpläne auch verschiedene, heuristische Algorithmen an. Ziel ist es hier, für die große Mehrzahl der Anwendungen bessere approximative Lösungen in kürzerer Zeit zu finden. Eine Verkürzung der Dauer einer Iteration ist auch zu erreichen, indem einfachere Entscheidungsregeln verwendet werden, welche rechenintensive Anweisungen wie die Generierung von Zufallszahlen und die Exponentiation zur Berechnung des Annahmekriteriums aus Gleichung (4.12) vermeiden. Diese Überlegungen motivieren die Untersuchung von heuristischen Verfahren, welche durch *nichtrandomisierte* Entscheidungsregeln zur Zustandsannahme gekennzeichnet sind; die in Gleichung (4.20) zusammengefaßten Anforderungen an die Kontrollsequenz und die Funktionsweise des Algorithmus — vermehrte Annahme von Verschlechterungen bei großen T_k — bleiben davon jedoch unberührt.

Stochastische Relaxation, Schwellwertakzeptanz und Sintflutalgorithmus

Als *stochastische Relaxation* wollen wir ein Verfahren bezeichnen, das keinen Gebrauch von der Kontrollsequenz macht. Aufgrund des Annahmekriteriums

$$accept \quad = \quad \begin{cases} \texttt{TRUE} & \text{falls} \quad \Delta\phi \leq 0 \\ \texttt{FALSE} & \text{sonst} \end{cases} \tag{4.29}$$

werden Kostenzuwächse immer abgelehnt, wodurch die Gefahr der Konvergenz in einem lokalen Minimum wächst. Die Eignung des Verfahrens hängt daher wesentlich von den Eigenschaften der Kostenfunktion ab, was in Bild 4.14 anhand zweier Beispiele mit einer unterschiedlichen Anzahl von Nebenminima verdeutlicht ist. Sind die Startzustände gleichverteilt auf der Zustandsmenge Z, so ist ein stochastisches Relaxationsverfahren zur Minimierung der Kostenfunktion ϕ_A ausreichend, da das globale Minimum mit großer Wahrscheinlichkeit $P_A(z_f \in Z_{min})$ erreicht wird; für die Funktion ϕ_B ist die Wahrscheinlichkeit P_B dagegen geringer, und Annahmekriterien, bei denen Kostenerhöhungen akzep-

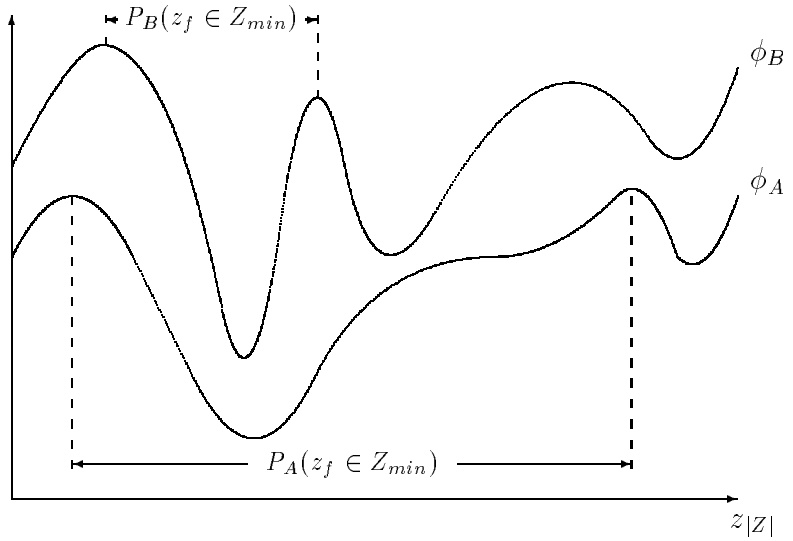


Bild 4.14: Kostenfunktionen zur Erläuterung der stochastischen Relaxation.

tiert werden, sind geeigneter. Trotz des prinzipiellen Nachteils schließen wir Kriterium (4.29) zur Minimierung der Kostenfunktion der optimalen Instantiierung (4.10) nicht aus, da die durch den Attributflußgraphen und die Analyseprozeduren realisierten Bewertungen $G(I(C_g))$ keine problemunabhängigen quantitativen Aussagen über $P(z_f \in Z_{min})$ zulassen.

In [Due90, Due93a, Due93b] werden nichtrandomisierte Annahmekriterien vorgestellt und anhand zweier Optimierungsaufgaben — eines Traveling-Salesman–Problems mit 442 Städten sowie der Erzeugung fehlerkorrigierender Codes — mit den Ergebnissen des Metropolis–Algorithmus verglichen.

Die Grundidee der *Schwellwertakzeptanz* (*threshold acceptance*), deren Annahmekriterium durch

$$accept \quad = \quad \begin{cases} \texttt{TRUE} & \text{falls} \quad \Delta\phi \leq T_k \\ \texttt{FALSE} & \text{sonst} \end{cases}$$

definiert ist, besteht darin, *alle* Zustände zu akzeptieren, die *nicht wesentlich höhere* Kosten als der aktuelle Zustand verursachen. Die zulässige Kostendifferenz wird dabei durch die Kontrollsequenz gesteuert, welche wie im Falle des Simulated Annealing einen hohen Anfangswert besitzen und monoton fallend (Gleichung (4.20)) sein muß. Stellvertretend für alle hier diskutierten Verfahren (mit Ausnahme der stochastischen Relaxation) verdeutlicht Bild 4.15 den Einfluß einer gemäß Gleichung (4.20) gewählten Kontrollsequenz, wobei insbesondere die Notwendigkeit eines großen Startwertes T_0 und der langsamen Verringerung der Schwelle sichtbar wird. Bei einem zu niedrigen Startwert T_0' oder einer zu schnellen

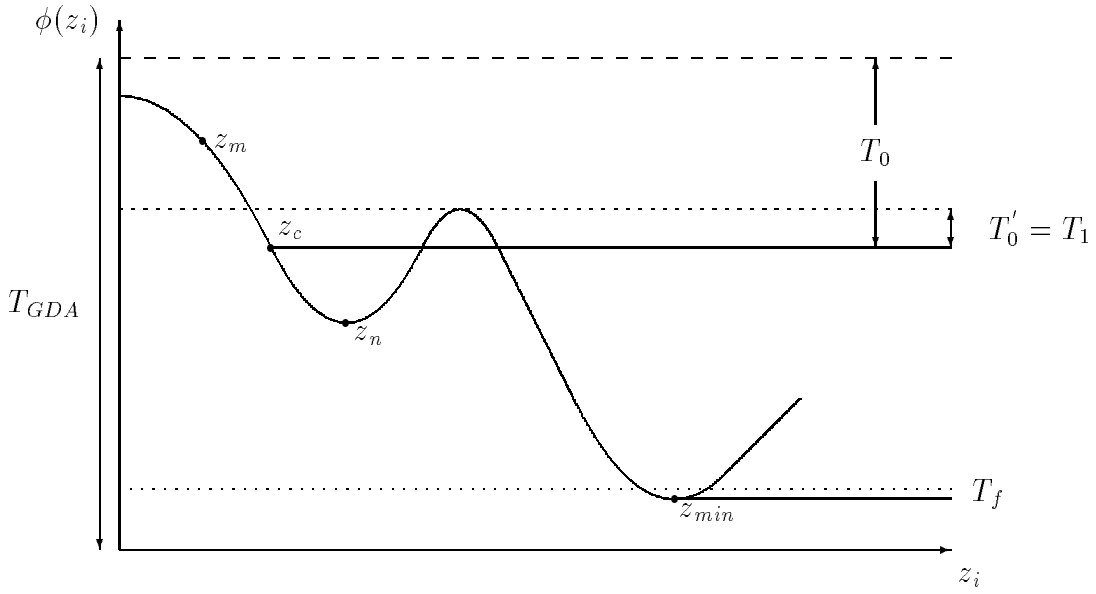


Bild 4.15: Bedeutung der Kontrollsequenz für die kombinatorische Optimierung am Beispiel der Schwellwertakzeptanz. Erläuterungen finden sich im Text.

Absenkung von T_0 auf T_1 wird zwar einerseits verhindert, daß die Suche durch Annahme von z_m in Richtung des absoluten Maximums verläuft, andererseits kann die "Kostenbarriere" des lokalen Maximums ausgehend von z_c nicht überwunden werden. Wird dagegen der genügend große Startwert T_0 langsam abgesenkt, so kann das absolute Minimum z_{min} erreicht werden, welches aufgrund der sehr kleinen Schwelle T_f am Ende der Suche nicht mehr verlassen wird.

Im Gegensatz zu Abkühlstrategie und Abbruchkriterium, die wir aus den Gleichungen (4.26) und (4.28) unverändert übernehmen können, ist die Ermittlung von T_0 gegenüber Gleichung (4.23) leicht zu modifizieren. Um eine anfängliche Annahmerate γ_0 zu erreichen, beobachten wir die Kostenzuwächse $(\Delta\phi)_i^{(+)}$, $1 \leq i \leq m_0$, und bestimmen T_0 so aus der kumulativen, empirischen Verteilungsfunktion $F_{(\Delta\phi)^{(+)}}$, daß Übergänge mit positiver Kostendifferenz mit einer Wahrscheinlichkeit von $\gamma_0 (= 0.95)$ akzeptiert werden (vgl. Bild 4.16).

Während der Kontrollparameter des Schwellwertverfahrens die zulässige Kosten*differenz* angibt, und durch die Kosten des aktuellen Zustandes die Qualität der momentanen Lösung berücksichtigt, werden beim *Sintflutalgorithmus* (*great deluge algorithm*) [1] *alle* Zustände akzeptiert, sofern ihre Kosten unterhalb eines Schwellwertes liegen. Dieser wird wiederum

[1] Der Name "Sintflutalgorithmus" hat seine Ursache in der ursprünglichen Formulierung des Algorithmus für ein *Maximierungs*problem. Der Schwellwert entspricht dabei dem beständig *ansteigenden* "Wasserstand", der beim Umherwandern auf dem "Kostengebirge" nicht unterschritten werden darf.

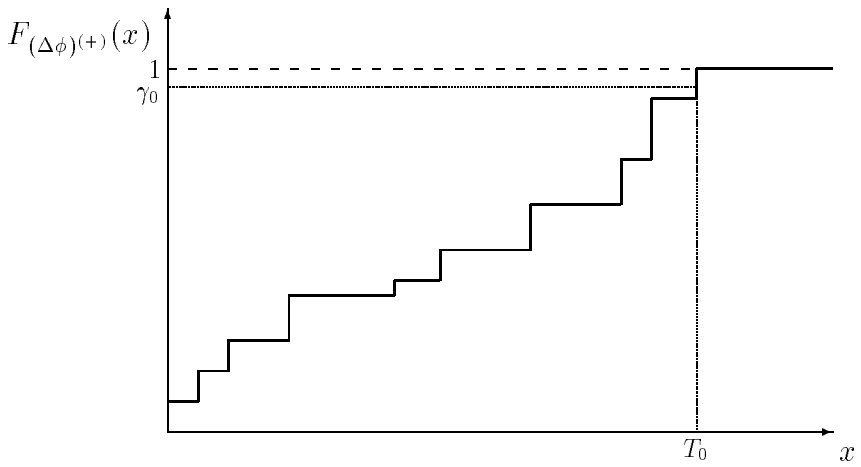


Bild 4.16: Bestimmung des Startwertes der Kontrollsequenz aus der Verteilungsfunktion der Kostendifferenzen.

durch den Kontrollparameter bestimmt, und wir erhalten

$$accept \quad = \quad \begin{cases} \texttt{TRUE} & \text{falls} \quad \phi_n \leq T_k \\ \texttt{FALSE} & \text{sonst} \end{cases} \tag{4.30}$$

als Annahmekriterium. In Bild 4.15 ist der Kontrollparameter des Sintflutalgorithmus mit T_{GDA} bezeichnet und am linken Rand eingezeichnet. Analog zum obigen Verfahren wird T_0 hier entweder aus der Verteilungsfunktion der Kosten bestimmt, oder durch den maximalen Wert der Kostenfunktion vorgegeben, $T_0 = \phi_{max}$. Als Abkühlstrategie wird in [Due93a] eine Verringerung der Differenz zwischen Kontrollparameter und aktuellen Kosten gemäß

$$T_{k+1} \quad = \quad T_k - \alpha \cdot (T_k - \phi_c), \alpha \approx 0 \tag{4.31}$$

vorgeschlagen; wir wählen für α den Wert $\alpha = 0.05$.

Eine als *Record-to-Record-Travel* bezeichnete Variante des Sintflutalgorithmus [Due93a] ergibt sich schließlich durch die Verwendung des Annahmekriteriums

$$\begin{aligned} accept \quad &= \quad \begin{cases} \texttt{TRUE} & \text{falls} \quad \phi_n \leq \phi_{record} + T_k \\ \texttt{FALSE} & \text{sonst} \end{cases} \\ \phi_{record} \quad &= \quad \min_{z_i \in Z_v} \{\phi(z_i)\} \end{aligned} \tag{4.32}$$

das alle Zustände akzeptiert, die nicht wesentlich größere Kosten als die momentan beste Lösung ϕ_{record} verursachen. Diese ergibt sich als Kostenminimum aller bereits evaluierten Zustände, die in der Menge Z_v zusammengefaßt werden.

Mit jedem der drei beschriebenen Verfahren (Schwellwert– und Sintflutalgorithmus,

```
/* Genetischer Algorithmus zur Kostenminimierung */
initialisiere eine Lösungsmenge Z_c := {z_{c,1}, ..., z_{c,p}} ⊆ Z
∀ z_{c,i} ∈ Z_c
    berechne φ_{c,i} = φ(z_{c,i})
WHILE abbruch() ≠ TRUE
    initialisiere Z_n := ∅
    bestimme eine temporäre Lösungsmenge Z_t := {z_{t,1}, ..., z_{t,p}}
     durch Anwendung der Operationen zur Selektion, Rekombination
     und Mutation auf die Zustände z_{c,i} ∈ Z_c
    ∀ z_{t,i} ∈ Z_t
        berechne φ_{t,i} := φ(z_{t,i})
    bringe die p besten Zustände aus Z_c ∪ Z_t nach Z_n
    Z_c := Z_n
gebe z_{c,best} := argmin_{z_{c,i} ∈ Z_c}{φ(z_{c,i})} als Lösung aus
```

Bild 4.17: Ein genetischer Algorithmus zur Kostenminimierung.

Record-to-Record-Travel) werden in den oben erwähnten Anwendungen bedeutend bessere Ergebnisse als mit dem Metropolis–Algorithmus erzielt [Due90, Due93a]. Neben der bereits angesprochenen kürzeren Iterationsdauer und einer rascheren Konvergenz wird insbesondere auch eine größere Unempfindlichkeit gegenüber kleinen Veränderungen der Kontrollparameter betont. Bevor wir eine weitere Beschleunigung der Algorithmen durch die Nutzung von Parallelisierungsmöglichkeiten diskutieren, wollen wir im nächsten Abschnitt den Einsatz von genetischen Algorithmen zur Kostenoptimierung erörtern.

Genetische Algorithmen

Genetische Algorithmen [Hol75] adaptieren die aus Genetik und Evolution bekannten Prinzipien der *Selektion*, der *Rekombination* und der *Mutation* zur Lösung kombinatorischer Optimierungsprobleme. Unter den zahlreichen Anwendungen — Übersichten geben beispielsweise [Gre87] und [Boo89] — finden sich sowohl Arbeiten aus dem Bereich der ikonischen Bildverarbeitung [Pal94, Abb94] als auch Ansätze zur Objekterkennung [Ank90] und zur Wissensverarbeitung in semantischen Netzen [For91]. Im folgenden werden zunächst die erwähnten Basismechanismen erläutert und Unterschiede zu den oben beschriebenen Optimierungsverfahren diskutiert. Dazu stützen wir uns weitestgehend auf die einführenden Darstellungen in [Dav87, Gol89, Dav91, Sch94]. Anschließend erörtern wir den Einsatz genetischer Algorithmen bei der iterativen Instantiierung.

Bild 4.17, das einen genetischen Algorithmus zur Kostenminimierung zeigt, läßt bereits einen wesentlichen Unterschied zu den bisher behandelten Verfahren erkennen: Während

Zustand z_i:	1	2	3	4	5	6	7	8	
Kosten ϕ_i:	0.3	0.4	0.1	0.5	0.9	0.6	0.1	0.3	
Bewertung ϕ_i^q:	0.7	0.6	0.9	0.5	0.1	0.4	0.9	0.7	$\sum = 4.8$
generiere eine Zufallszahl q aus $\mathcal{U}(0.0, 4.8)$: sei $q = 1.8$									
akkumulierte Bewertung:	0.7	1.3	2.2	2.7	2.8	3.2	4.1	4.8	
suche:			$\uparrow$						

Bild 4.18: Beispiel zur Zustandsauswahl mittels Rouletterad–Verfahren.

letztere in jedem Iterationsschritt genau einen neuen Zustand erzeugen, arbeiten genetische Algorithmen auf einer Menge von Zuständen. Sie nehmen somit eine parallele Erkundung des Lösungsraumes Z vor und besitzen dadurch eine gewisse Robustheit gegenüber lokalen Minima der Kostenfunktion.

Neue Zustände werden durch die Anwendung von Selektions–, Rekombinations– und Mutationsoperatoren erzeugt, die anhand der Bilder 4.18 und 4.19 erläutert werden. Der Grundgedanke der Selektion besteht darin, Zustände mit geringen Kosten bevorzugt zur Generierung von Folgezuständen zu verwenden, da diese mit größerer Sicherheit bereits korrekte Teillösungen enthalten; die Selektionswahrscheinlichkeit $p_s(z_{c,i})$ für den Zustand $z_{c,i}$ wird daher umgekehrt proportional zu seinen Kosten gewählt, was dem aus der Evolution bekannten Prinzip des Überleben des Stärkeren (*survival of the fittest*) entspricht. Eine verbreitete Methode zur Realisierung dieser Idee ist das Rouletterad–Verfahren, das in Bild 4.18 dargestellt ist. Zur Lösung eines Minimierungsproblems werden zunächst die Kosten ϕ_i der Zustände aus Z_c in Bewertungen ϕ_i^q konvertiert und zur Gesamtbewertung

$$\begin{aligned} \phi_\Sigma^q &= \sum_{z_i \in Z_c} \phi_i^q \\ \phi_i^q &= \phi_{max} - \phi_i \end{aligned} \tag{4.33}$$

der Zustandsmenge addiert. Auf den Index der Zustandsmenge wird hier vereinfachend verzichtet, und da wir die Kostenfunktion aus Gleichung (4.10) verwenden, dürfen wir $\phi_{max} = 1.0$ annehmen. Anschließend werden die akkumulierten Bewertungen

$$\phi_{\sigma,i}^q = \sum_{j=1}^{i} \phi_j^q \tag{4.34}$$

berechnet und mit einer Zufallszahl aus einer im Intervall $[0, \phi_\Sigma^q]$ uniform verteilten Grund-

gesamtheit verglichen. Übersteigt die akkumulierte Bewertung $\phi^q_{\sigma,i}$ den Wert der Zufallszahl, so wird der Zustand z_i als Ergebnis des Selektionsprozesses ausgegeben. Die Wahrscheinlichkeit für das Eintreten dieses Ereignisses ist

$$p_s(z_i) \;=\; \phi^q_i / \phi^q_\Sigma, \tag{4.35}$$

wodurch Zustände mit großer Bewertung (niedrigen Kosten) häufiger zur Erzeugung von Nachfolgezuständen herangezogen werden.

Genetische Algorithmen arbeiten im allgemeinen nicht mit den Parametern des zu lösenden Optimierungsproblems selbst, sondern mit einer Codierung der Zustände durch einen endlichen Zeichenvorrat. Eindeutigkeit und Vollständigkeit der Abbildung — jeder Zustand sollte durch genau ein Codewort repräsentiert werden — sind dabei für das Auffinden einer optimalen Lösung wichtig; ist die Codierung darüber hinaus eineindeutig, so erleichtert dies in zahlreichen Anwendungen den Entwurf der genetischen Operatoren [Dav85, Hou94].

In dem zur iterativ–optimierenden Instantiierung eingesetzten genetischen Algorithmus greifen wir diesen Aspekt auf und codieren einen Analysezustand $\boldsymbol{r}$ aus Gleichung (4.5) durch die Vorschrift

$$(r_1^{(i_1)}, \ldots, r_l^{(i_l)}, h_1^{(j_1)}, \ldots, h_k^{(j_k)}) \rightarrow ((i_1), \ldots, (i_l), (j_1), \ldots, (j_k)), \tag{4.36}$$

welche im Gegensatz zu den meisten in der Literatur anzutreffenden Ansätzen kein binäres Alphabet — wie etwa die Menge $\{0, 1\}$ — verwendet, sondern eine eineindeutige Abbildung zwischen den Analysezuständen und der Menge $\mathbb{N}^{l+k}$ etabliert. Zur Ermittlung und Codierung des Analysezustands werden die initialen Flußgraphknoten und die Knoten zur Bewertung von Konzepten in einer einmal festgelegten Reihenfolge durchlaufen und die Indizes der zur Instantiierung verwendeten Hypothesen bzw. Modalitäten notiert. Betrachten wir beispielsweise die in Bild 4.11 auf Seite 98 dargestellte Situation, so erhalten wir bei Verwendung der durch die schwarzen Kreise markierten Zuordnung den Zustandsvektor $\boldsymbol{r} = (1, 1, 2, 1, 1, 2, 1, 2, c)$, $c = const.$, wenn die acht Knoten der Ebene 0 von links nach rechts betrachtet werden. Die konstante Komponente c ergibt sich hierbei aufgrund der Tatsache, daß das Konzept "Quadrat" keine Modalitätsmengen besitzt, und der Konzeptknoten ("quadratBEW") somit stets unter Verwendung einer Default–Modalität — alle Bestandteile werden als obligatorisch betrachtet — zu berechnen ist.

Bild 4.19 benutzt die beschriebene Codierung zur Illustration der Generierung von Folgezuständen, die durch die Anwendung der Operatoren zur Rekombination und Mutation auf die gemäß Gleichung (4.35) selektierten Zustände erfolgt. Die Darstellung beschränkt sich auf einen Mechanismus zur Zustandsrekombination, der aufgrund der vorhandenen Analogie zur Kreuzung von Organismen in der Biologie als *Kreuzungsoperator* (engl.: *Crossover*) bezeichnet wird, sowie auf einen einfachen Mutationsmechanismus. Zur Rekombination durch den Crossoveroperator wird für jeden Zustand zufällig ein Partner

Bild 4.19: Illustration der Standardoperatoren zur Rekombination und Mutation in genetischen Algorithmen.

ausgewählt und eine "Bruchstelle" bestimmt, an der die beiden Partner "aufgetrennt" und rekombiniert werden. Ist $n = l + k$ die Länge der Codewörter und m die Bruchstelle, $1 \leq m \leq n-1$, so werden zwei Partner $z_{c,1} = (c_{1,1}, \ldots, c_{1,m}, c_{1,m+1}, \ldots, c_{1,n})$ und $z_{c,2} = (c_{2,1}, \ldots, c_{2,m}, c_{2,m+1}, \ldots, c_{2,n})$ aus der aktuellen Zustandsmenge Z_c durch die Vorschrift

$$\begin{aligned} z_{t,1} &= (c_{1,1}, \ldots, c_{1,m}, c_{2,m+1}, \ldots, c_{2,n}) \\ z_{t,2} &= (c_{2,1}, \ldots, c_{2,m}, c_{1,m+1}, \ldots, c_{1,n}) \end{aligned} \tag{4.37}$$

zu zwei neuen Zuständen der temporären Zustandsmenge Z_t rekombiniert; Bild 4.19 zeigt dies für die Paare (A, C) und (B, D) und die Bruchstellen $m = 2$ und $m = 1$.

Der Mutationsoperator sorgt für eine geringfügige stochastische Verzerrung der Zustände, und ermöglicht dadurch eine bessere Überwindung lokaler Minima der Kostenfunktion. In Bild 4.19 wird ein Zeichen des Codewortes mit einer Mutationswahrscheinlichkeit von $p_m = 0.05$ vertauscht, was durch den Vergleich dieser Schwelle mit einer Zufallszahl aus einer geeigneten Grundgesamtheit realisiert wird. Nach der Anwendung des Mutationsoperators bilden die neu entstandenen Zustände (A'' – D'') und ihre Vorgänger die Ausgangsmenge, aus denen die Zustände für die nächste Generation ausgewählt werden. Denkbare Annahmekriterien sind hier die Übernahme aller neuen Zustände, der p besten Zustände, oder aller Zustände, deren Kosten geringer als das Kostenmittel sind; im ersten Fall wäre $Z_n = \{A'', B'', C'', D''\}$ und für die beiden letztgenannten Strategien würde $Z_n = \{A'', A, B, D''\}$ gelten.

Nach dieser kurzen Einführung in die grundlegende Vorgehensweise wollen wir nun den Entwurf der genetischen Operatoren für die optimale Instantiierung des Attributfluß-graphen erläutern. Zunächst erinnern wir daran, daß die Berechnung der Kostenfunktion unabhängig von deren Wahl — tatsächlich werden wir auch für die genetische Optimierung

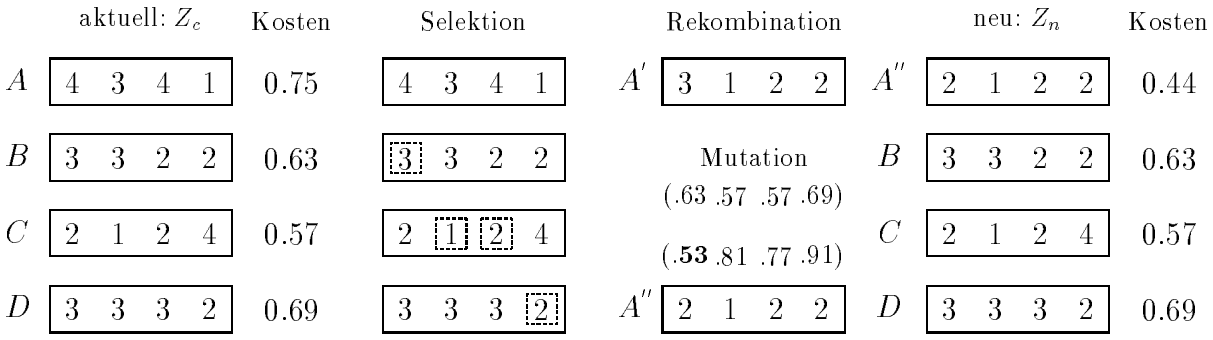


Bild 4.20: Selektions–, Rekombinations– und Mutationsoperatoren des genetischen Algorithmus zur optimalen Instantiierung.

die Kostenfunktion aus Gleichung (4.10) verwenden — die datengetriebene Instantiierung des Attributflußgraphen erfordert. Die obige Diskussion der Basismechanismen verdeutlicht, daß dieser Prozeß für jedes Element der Zustandsmenge unter Verwendung der in der jeweiligen Codierung angegebenen Hypothesen und Modalitäten zu wiederholen ist. Die Iterationsdauer wächst somit mit der Größe der Zustandsmenge $|Z_c|$ und Lösungen werden langsamer generiert als vom Metropolis–Algorithmus. Neben der Parallelisierung der Zustandsgenerierung und –evaluierung, die als eine Möglichkeit zur Verkürzung der Iterationsdauer in Abschnitt 4.4.2 behandelt wird, bietet es sich an, die genetischen Operatoren so zu entwerfen, daß in jedem Iterationsschritt lediglich ein neuer Zustand generiert wird. Durch dieses Vorgehen ist nicht zuletzt auch eine unmittelbare Vergleichbarkeit mit den Ergebnissen des Simulated Annealing und seiner Derivate gewährleistet.

Bild 4.20 veranschaulicht die zur iterativ–optimierenden Instantiierung entworfenen Operatoren zur Selektion, Rekombination und Mutation. Um alle Elemente der aktuellen Zustandsmenge zu nutzen, wird im Gegensatz zum obigen *1-Punkt*–Crossoveroperator (engl.: *Single-Point–Crossover*) ein *Mehrpunkt*–Operator (engl.: *Multi-Point–Crossover*) zur Rekombination verwendet, der im linken Teil der Abbildung dargestellt ist. Ein neuer Zustand $z_{t,1}$ — dieser ist einziges Element der temporären Zustandsmenge Z_t — wird erzeugt, indem für jede Komponente des Vektors mit Hilfe des Rouletterad–Verfahrens ein Element der aktuellen Zustandsmenge Z_c ausgewählt, und die entsprechende Komponente übernommen wird. In Bild 4.20 wurden die erste Komponente des Zustands B, die zweite und dritte Komponente von C und die vierte Komponente von D selektiert, und zum neuen Zustand $A^{'} = (3\ 1\ 2\ 2)$ rekombiniert. Um den Einfluß "schlechter" Vorgänger weiter zu unterdrücken, wählen wir einen Mutationsoperator, der eine zu den Kosten des selektierten Zustandes proportionale Mutationswahrscheinlichkeit verwendet. In unserem Beispiel wird die erste Komponente mutiert, und der daraus resultierende neue Zustand $A^{''} = (2\ 1\ 2\ 2)$

ersetzt das Element der Ausgangsmenge, welches die größten Kosten verursacht (hier: den Zustand A).

Der durch die beschriebenen Operatoren definierte Algorithmus wird in Kapitel 5 sowohl zur Lösung des Optimierungsproblems 4.4 als auch zur Ermittlung einer optimalen Zuordnung zwischen den Prozessoren eines parallelen Rechensystems und den Knoten des Attributflußgraphen eingesetzt, und mit den Standardoperatoren aus Bild 4.19 verglichen. Zuvor befaßt sich der folgende Abschnitt mit Möglichkeiten zur Parallelisierung der bislang eingeführten kombinatorischen Optimierungsverfahren.

4.4 Parallele iterative Optimierung

Nach der Formulierung der wissensbasierten Analyse als Optimierungsproblem und der Vorstellung mehrerer Lösungsverfahren wollen wir nun untersuchen, wie die parallele Instantiierung des Attributflußgraphen durch die Parallelisierung der iterativen Optimierung ergänzt werden kann, um die durch den Sensordatenstrom vorgegebene Verarbeitungsgeschwindigkeit zu erreichen oder die Qualität der in einem bestimmten Zeitraum erzielten Lösung zu verbessern. Entsprechend der Reihenfolge der Vorstellung im vorigen Abschnitt diskutieren wir zunächst Parallelisierungsmöglichkeiten für den Metropolis–Algorithmus und seine Varianten, und untersuchen anschließend die Parallelisierung genetischer Algorithmen.

4.4.1 Parallelisierung kombinatorischer Optimierungsverfahren

Neben der Motivation durch die restriktiven Zeitanforderungen der wissensbasierten Musteranalyse ergibt sich allein aus den theoretischen Eigenschaften der benötigten Kontrollsequenz ein zusätzliches Interesse an der Parallelisierung des Metropolis–Algorithmus und der verschiedenen Schwellwertverfahren: die langsame Verringerung des Kontrollparameters macht viele Iterationen erforderlich, und resultiert in einer entsprechend langen Rechenzeit, die durch den Einsatz mehrerer Prozessoren verkürzt werden soll.

Während für den Metropolis–Algorithmus bereits eine Vielzahl von Parallelisierungsansätzen existieren [Aar86a, Aar86b, Rou90, Wit90, Lee92a, Lee92b, Boi93] — Übersichten und weitere Literaturhinweise finden sich auch in [Laa87, Aar89, Aze92a] —, ist die Parallelisierung von Schwellwertakzeptanz und Sintflutalgorithmus bislang nicht untersucht. Die Formulierung der Verfahren als Spezialisierung des allgemeinen Algorithmus in Bild 4.12 erlaubt es jedoch, die entwickelten Parallelisierungsstrategien unmittelbar auf die letztgenannten Optimierungsverfahren zu übertragen und hier zusammenfassend darzustellen.

Im Sinne der in [Laa87, Kap. 8] vorgenommenen Unterscheidung wollen wir uns im folgenden auf "universelle" Parallelisierungsstrategien konzentrieren, die bestenfalls Kenntnis über die Kostenfunktion, nicht aber über den Problemkreis voraussetzen; "maßgeschnei-

derte" Verfahren, die problemabhängige Möglichkeiten zur Parallelisierung nutzen — beispielsweise durch eine geschickte Partitionierung der Zustandsmenge und der parallelen Lösung von Teilproblemen (vgl. [Fel85] für eine Anwendung auf das Traveling-Salesman-Problem) — werden nicht betrachtet. Fassen wir Algorithmus 4.12 als einen Zyklus aus Zustandgenerierung, Evaluierung der Kostenfunktion und der Berechnung des Annahmekriteriums auf, so stellt die zur Kostenberechnung notwendige datengetriebene Instantiierung des Attributflußgraphen (vgl. Abschnitt 4.3.1) einen Beitrag zur Parallelisierung der Optimierungsverfahren dar, der ebenfalls problemabhängigen Charakter besitzt. Ein derartiges Vorgehen soll daher im Rahmen dieses Abschnitts ebensowenig wie eine mögliche Verteilung disjunkter Ausschnitte des Attributflußgraphen (vgl. Bild 4.2) oder die Parallelisierung der Prozeduren zur Zustandsgenerierung und –akzeptierung weiterverfolgt werden. Die Grundidee der hier untersuchten Ansätze besteht vielmehr darin, den gesamten Algorithmus auf mehreren Prozessoren ablaufen zu lassen, und dabei Strategien zum *Zustandsaustausch* zu untersuchen, um eine möglichst effiziente Suche und eine problemunabhängige Parallelisierung zu gewährleisten. Entsprechend der verwendeten Austauschverfahren betrachten wir vier Vorgehensweisen [Aze92a]: die *unabhängige* oder *multidirektionale* Suche, die *periodisch interagierende* und die *dynamisch interagierende* Suche, sowie die *multiple* oder *unidirektionale* Suche.

Die multidirektionale Suche, deren Prinzip in Bild 4.21 dargestellt ist, verwendet p *unabhängige* Prozessoren mit p verschiedenen Startzuständen $z_{0,p}$, $1 \leq i \leq p$, und p (nicht notwendigerweise) identischen Kontrollsequenzen zur Optimierung. Können in der zur Verfügung stehenden Rechenzeit N Iterationen ausgeführt werden, so gehen die Startzustände $z_{0,i}$ in die prozessorlokalen Endzustände $z_{N,i}$ über, und als Lösungszustand der parallelen Optimierung wird

$$z_N \quad = \quad z_{N,1} \oplus z_{N,2} \oplus \ldots \oplus z_{N,p} \tag{4.38}$$

als gemeinsamer Endzustand akzeptiert. Der Operator $\oplus$ ist durch die Gleichungen

$$\begin{aligned} z_i \oplus z_j &= \begin{cases} z_i & \text{falls} \quad \phi(z_i) \leq \phi(z_j) \\ z_j & \text{sonst} \end{cases} \\ z_1 \oplus z_2 \oplus \ldots \oplus z_p &= (z_1 \oplus z_2 \oplus \ldots \oplus z_{p-1}) \oplus z_p \end{aligned} \tag{4.39}$$

definiert, wodurch wir als Lösung z_N einen der prozessorlokalen Endzustände mit geringsten Kosten erhalten.

Da jeder Prozessor von einem eigenen Startzustand ausgeht und unabhängig von allen anderen arbeitet, werden p *verschiedene* Zustandsfolgen generiert. Die simultane Suche verläuft somit in p verschiedene Richtungen, wodurch sich die Gefahr verringert, daß der globale Optimierungsprozeß in einem Nebenminimum der Kostenfunktion konvergiert.

In Bild 4.21, welches das Vorgehen schematisch darstellt, sind die unterschiedlichen Zustandsfolgen durch verschiedene Symbole (Quadrat, Kreis und Dreieck) angedeutet, deren Schwärzungsgrad proportional zum prozessorlokalen Wert der Kostenfunktion ist.

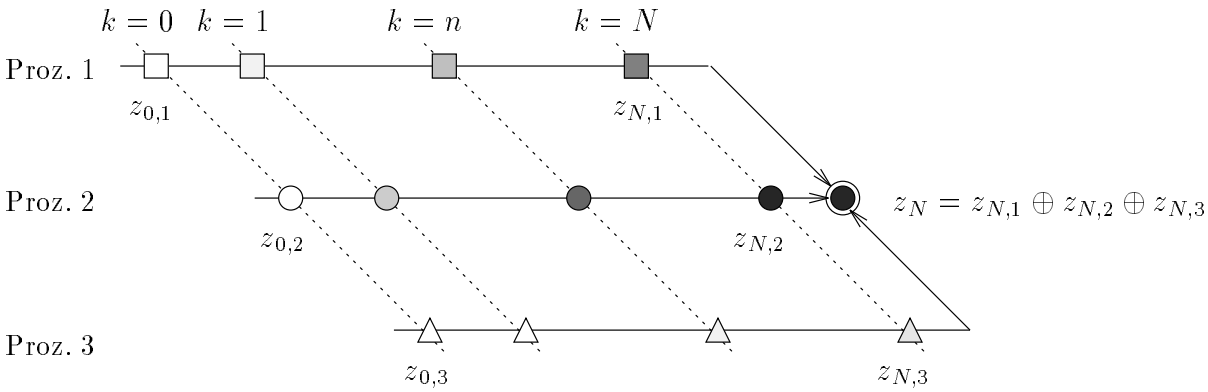


Bild 4.21: Parallele kombinatorische Optimierung durch multidirektionale Suche.

Als wesentlicher Vorteil der multidirektionalen Suche ist der sehr geringe Kommunkationsaufwand zu nennen. Ein Zustandsaustausch ist nur am Ende der Suche notwendig und beinhaltet — dies gilt gleichermaßen für die weiter unten beschriebenen Parallelisierungsstrategien — im Falle der iterativ-optimierenden Kontrolle lediglich den Austausch der Zustandsvektoren $\boldsymbol{r}$ aus Gleichung (4.5). Da $\boldsymbol{r}$ ein Vektor fester Länge ist, ist das Volumen der auszutauschenden Daten im Unterschied zur Parallelisierung der Zustandsraumsuche (vgl. Abschnitt 3.3) während der Analyse kostant. Zusammen mit der gleichmäßigen Auslastung der Rechnerknoten — jeder Prozessor instantiiert den gleichen Attributflußgraphen — sind somit die Voraussetzungen für das Erreichen eines großen Speedups und einer guten Effizienz geschaffen.

Den Vorteilen der multidirektionalen Suche steht als Nachteil eine geringe Wirksamkeit der parallelen Suche gegenüber, die insbesondere auftreten kann, wenn nur wenige Iterationen ausgeführt werden können, oder wenn ein Prozessor bereits eine gute lokale Lösung erreicht hat (vgl. Prozessor 2 im Iterationsschritt $k = n$ in Bild 4.21). In diesem Fall verrichten die übrigen Prozessoren keine nützliche Arbeit, da sie Zustandsfolgen mit größeren Kosten weiterverfolgen. Sinnvoller wäre es hier, die Suche nach einem globalen Minimum auf die Umgebung des momentan besten Zustands zu konzentrieren, wozu sich das in Bild 4.22 dargestellte Verfahren der parallelen, unidirektionalen Suche anbietet.

Bei der *unidirektionalen Suche* werden alle Prozessoren zur gemeinsamen Fortsetzung *einer* Zustandsfolge genutzt. Dazu wird ein gemeinsamer Zustand z_k, $0 \leq k \leq N-1$, an die p zur Verfügung stehenden Prozessoren verteilt. Zunächst erzeugt jeder Prozessor einen *vorläufigen* Folgezustand $y_{k+1,i}$, $1 \leq i \leq p$, indem er wie in Algorithmus 4.12 einen Nachfolger generiert, dessen Kosten berechnet, und über die Annahme entscheidet. Als

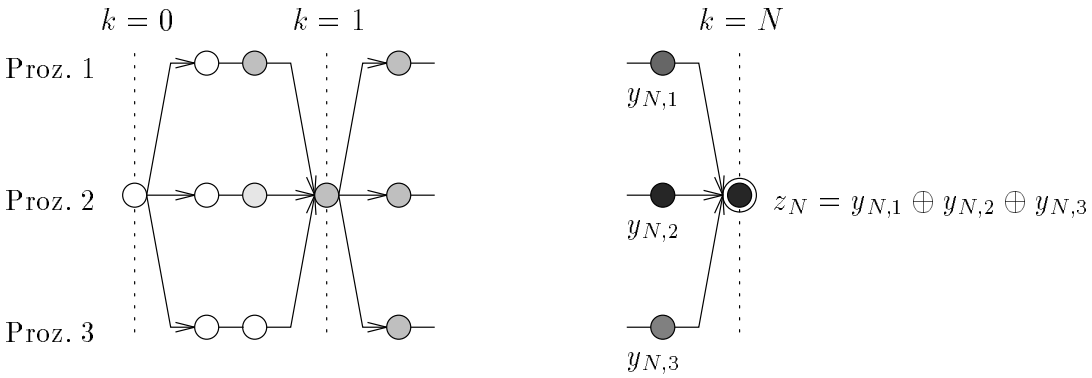


Bild 4.22: Parallele kombinatorische Optimierung durch unidirektionale Suche.

gemeinsamer Zustand z_{k+1} wird

$$z_{k+1} \quad = \quad y_{k+1,1} \oplus y_{k+1,2} \oplus \ldots \oplus y_{k+1,p} \tag{4.40}$$

gewählt und an alle Prozessoren als Ausgangszustand für die nächste Iteration verteilt. Existieren mehrere Zustände $y_{k+1,i}$ mit gleichen Kosten, so wird eine zufällige Wahl getroffen, und falls kein vorläufiger Folgezustand akzeptiert wurde ($y_{k+1,i} = z_k$ für alle Prozessoren i) wird der gemeinsame Zustand z_k beibehalten ($z_{k+1} = z_k$). Im Vergleich zur multidirektionalen Suche erfordert der Zustandsaustausch nach jedem Iterationsschritt einen wesentlich größeren Kommunikationsaufwand, und da die Konsensbildung gemäß Gleichung (4.40) streng sequentiell erfolgen muß, entstehen insbesondere bei einer hohen Annahmerate (großer Wert von T_k) zusätzliche Synchronisationsverluste. Für kleine Werte des Kontrollparameters werden dagegen nur wenige Folgezustände $y_{k+1,i}$ akzeptiert, und für eine geringe Prozessoranzahl können nahezu lineare Speedups erzielt werden [Rou92].

In zahlreichen Arbeiten wird vorgeschlagen, multi– und unidirektionale Suche zu kombinieren, um einerseits den Kommunikationsaufwand zu verringern, und andererseits die Wirksamkeit der Suche durch einen vermehrten Informationsaustausch zwischen den Prozessoren zu verbessern [Aar86a, Rou90, Lee92b]. Das am häufigsten anzutreffende Vorgehen besteht darin, die Anzahl der vorgeschlagenen und der akzeptierten Zustandsübergänge zu beobachten, und beim Absinken der Annahmerate unter einen Schwellwert von der multidirektionalen auf die unidirektionale Suche umzuschalten. Zwei weitere Verfahren, die *periodische* und die *dynamisch* interagierende Suche werden im folgenden vorgestellt.

Im Falle der periodisch interagierenden Suche kommunizieren die beteiligten Prozessoren zu festgelegten Zeitpunkten $s, 2s, 3s, \ldots$. Die natürliche Zahl $s \geq 2$ wird als Periodenlänge bezeichnet, und gibt an, wieviele Iterationen die Prozessoren zwischen den Kommunikationsschritten unabhängig voneinander ausführen. Beim Übergang vom Schritt $(js-1)$ nach js, $j = 1, 2, \ldots$, generiert jeder Prozessor einen vorläufigen Zustand $y_{js,i}$, $1 \leq i \leq p$, und wählt seinen endgültigen Zustand $z_{js,i}$ gemäß der Austauschregel

$$\begin{aligned} z_{js,1} &= y_{js,1} \\ z_{js,2} &= y_{js,1} \oplus y_{js,2} \\ &\vdots \\ z_{js,p} &= y_{js,1} \oplus y_{js,2} \oplus \ldots \oplus y_{js,p} \end{aligned} \tag{4.41}$$

Die Wirkung der Regel (4.41) und der Nutzen für die Suche sind in Bild 4.23 verdeutlicht. Zum Zeitpunkt $k = s$ lehnt Prozessor 2 die Annahme von $z_{s,1}$ (Quadrat) ab, gibt jedoch seinerseits seinen Zustand $z_{s,2}$ (Kreis) an Prozessor 3 weiter, und anschließend setzen beide die Suche unabhängig voneinander mit diesem Zustand fort. Zu Beginn erzeugt der Algorithmus also viele verschiedene Zustandsfolgen, wodurch eine anfängliche Konvergenz in einem lokalen Minimum vermieden wird; mit zunehmender Qualität der Lösung konzentriert sich die Suche jedoch auf die Zustände, welche einen schnellen Erfolg versprechen. Als Ergebnis der parallelen Suche wird der Zustand des p–ten Prozessors ausgegeben, welcher aufgrund des Zustandsaustauschs (4.41) die geringsten Kosten aufweist.

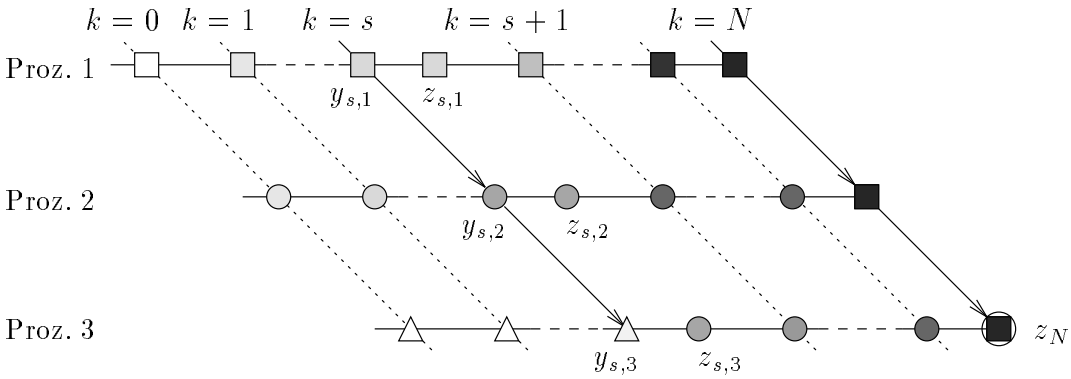


Bild 4.23: Parallele kombinatorische Optimierung durch periodisch interagierende Suche.

Die *dynamisch* interagierende Suche greift die Austauschregel (4.41) auf. Im Unterschied zur periodisch austauschenden Suche werden jedoch die Kommunikationszeitpunkte nicht apriori fixiert, sondern in Abhängigkeit von den erzielten Zwischenergebnissen bestimmt. In [Lee92b] wird vorgeschlagen, einen Zustandsaustausch durchzuführen, wenn die lokale Annahmerate für eine der p unabhängigen Zustandsfolgen unter einen Schwellwert sinkt, da dies eine signifikante Verringerung der Kosten voraussetzt. In der vorliegenden Arbeit wird anstelle der lokalen Annahmerate das Erreichen eines prozessorlokalen Kostenminimums als Kriterium für einen dynamischen Zustandsaustausch verwendet: entdeckt der i-te Prozessor ein neues Minimum ϕ_{record_i} so wird durch die Vorschrift (4.41) versucht, den zugehörigen Zustand an alle Prozessoren j, $i < j \leq p$, zu verteilen.

Neben der bereits erwähnten Möglichkeit zur problemabhängigen Parallelisierung exi-

stieren zusätzlich zu den hier beschriebenen Verfahren weitere Strategien, die im wesentlichen auf einer parallelen Untersuchung der Nachbarschaft eines Zustands beruhen [Gem84, Aze92a, Boi93]. Aufgrund einiger wenig erfolgversprechend verlaufender Voruntersuchungen und der speziellen Eignung der Verfahren für eine massiv–parallele SIMD–Architektur wollen wir diese jedoch nicht näher erläutern, sondern uns vor der experimentellen Evaluierung der ausführlicher diskutierten Ansätze mit der Parallelisierung genetischer Algorithmen befassen.

4.4.2 Parallelisierung genetischer Algorithmen

Bereits bei der Beschreibung des genetischen Basisalgorithmus (vgl. Bild 4.17, Seite 114) wurde hervorgehoben, daß durch die Betrachtung einer Zustandsmenge eine parallele Erkundung des Lösungsraums des zugrundeliegenden kombinatorischen Optimierungsproblems vorgenommen wird. Die dadurch gewonnene Robustheit der Suche wird jedoch mit einer zur Größe der Zustandsmenge proportionalen Iterationsdauer bezahlt, deren Reduzierung wiederum die wesentliche Motivation für die Untersuchung von Parallelisierungsmöglichkeiten bildet. Ebenso wie im letzten Abschnitt wollen wir problemabhängige Parallelisierungsmethoden auch hier von unseren Betrachtungen ausschließen, verweisen jedoch erneut auf die parallele Instantiierung des Attributflußgraphen als einen Beitrag zur Beschleunigung der Kostenberechnung.

Die Verwendung einer Zustandsmenge offeriert zwei Parallelisierungsmöglichkeiten unterschiedlicher Granularität, die in [Ste93] zur Ausführung auf SIMD– und MIMD–Architekturen vorgeschlagen werden:

- Bei einer feingranularen Parallelisierung [Suh87, Gor89, Man89, Spi91, Bal93] werden die genetischen Operatoren (Selektion,Rekombination,Mutation) simultan auf alle Elemente der Zustandsmenge angewendet, und auch die Kostenfunktion wird simultan für alle neu erzeugten Zustände evaluiert. Da hierzu dieselben Operationen auf einer Vielzahl von Daten auszuführen sind, wird diese Möglichkeit insbesondere für SIMD–Rechner erwogen.

- Die Verteilung einer (großen) Zustandsmenge auf mehrere Prozessoren wird als geeignete Strategie für MIMD–Architekturen beschrieben, und stellt den in der Literatur häufiger anzutreffenden Ansatz dar [Coh87, Pet87, Tan87, Müh89, Müh91, Coh91, Bia93, Sho93]. Hierbei werden sämtliche Operationen auf den Elementen der Teilmengen von einem Prozessor ausgeführt, und es findet ein gelegentlicher Zustandsaustausch zwischen den Prozessoren statt. Im allgemeinen arbeitet bei diesem Vorgehen jeder Rechnerknoten mit dem gleichen genetischen Algorithmus; es ist jedoch auch möglich, auf den verschiedenen Prozessoren unterschiedliche Selektions–, Rekombinations– und Mutationsstrategien zu realisieren.

Unter Berücksichtigung der zur Verfügung stehenden Zielhardware — einem homogenen Netz lose gekoppelter Workstations — wird in der vorliegenden Arbeit ausschließlich der letztgenannte Ansatz verfolgt. Zum Zustandsaustausch bieten sich die gleichen Strategien an, die bei der Parallelisierung des Metropolis–Algorithmus ausführlich diskutiert wurden; aus der Verwendung einer Zustandsmenge resultiert jedoch ein zusätzlicher Entwurfsparameter, da ein Austausch der n besten Elemente ($n \geq 1$) der lokalen Zustandsmengen zu erwägen ist. Für ein gegebenes Optimierungsproblem ist dabei stets zwischen dem erhöhten Kommunikationsaufwand (*längere Iterationsdauer*) und einer möglichen Steigerung der Wirksamkeit der parallelen Suche (*geringere Iterationsanzahl*) abzuwägen. Bevor wir diesen Aspekt der vorgestellten Optimierungsverfahren und ihrer Parallelisierungsmöglichkeiten im nächsten Kapitel einer experimentellen Evaluierung unterziehen, faßt der folgende Abschnitt die wesentlichen Gesichtspunkte beim Entwurf des iterativen Kontrollalgorithmus nochmals zusammen.

4.5 Zusammenfassung

Nach der Diskussion von Möglichkeiten und Problemen der Parallelisierung des ERNEST–Kontrollalgorithmus in Kapitel 3 wurde im vorliegenden Kapitel ein für die Parallelverarbeitung geeigneter, iterativ–optimierender Kontrollalgorithmus konzipiert. Als wichtige Anforderungen wurden zunächst die Unterstützung einer ergonomischen Wissensrepräsentation, die Parallelisierung von Netzwerkinferenzen und übergeordneter Kontrollstrategie sowie deren Problemunabhängigkeit betont. Während Kommunikations– und Speichereffizienz auch unter dem Gesichtspunkt der angestrebten Parallelisierung als weitere Entwurfsziele genannt wurden, berücksichtigt die abschließend geforderte Any–Time–Fähigkeit der Kontrolle Anforderungen zukünftiger informationsverarbeitender Systeme (vgl. auch Abschnitt 1.5) und legt ein iteratives Verfahren zur Generierung einer optimalen Interpretation nahe.

Die Forderung nach einem universell einsetzbaren Kontrollalgorithmus erlaubt es, die Suche nach geeigneten Elementaroperationen für die Instantiierung auf Elemente der Netzwerksprache zu beschränken. Nach einer kurzen Erörterung der Abbildung von Konzepten der Wissensbasis auf die Prozessoren eines parallelen Rechensystems wurde die Berechnung und Bewertung einzelner Attribute, Relationen, Kanten und Konzepte als Elementaroperationen definiert. Die automatische Konstruktion des als *Attributfluß* bezeichneten Datenflußgraphen aus der begriffszentrierten Repräsentation der Wissensbasis ist Aufgabe der Analysevorbereitung, welche somit nicht nur die Verteilung der Aufgaben auf die Prozessoren unterstützt, sondern insbesondere auch die ergonomische Adäquatheit sichert.

Die parallele, datengetriebene Instantiierung des Attributflußgraphen beginnt bei den Attributen, die direkt auf die Ergebnisse der initialen Segmentierung zugreifen, und en-

det mit der Bewertung der Analyseziele auf der obersten Ebene des Graphen. Um einen geringen Speicher– und Kommunikationsbedarf zu erreichen, wurde die in einer Voruntersuchung vorgeschlagene simultane Behandlung konkurrierender Zwischenergebnisse durch eine iterative Verarbeitung ersetzt.

Als Ziel der wissensbasierten Analyse wurde die Generierung einer symbolischen Beschreibung genannt, die optimal zu den Segmentierungsergebnissen paßt, und maximal kompatibel mit den in der Wissensbasis abgelegten Vorwerwartungen ist. Unter der Annahme, daß sich die Verträglichkeit der Interpretation mit Segmentierungsergebnissen und Wissensbasis in einer Instanzbewertung manifestiert, konnte die Analyse als ein kombinatorisches Optimierungsproblem formuliert werden, für das anschließend Lösungsmöglichkeiten diskutiert wurden. Anhand des bekannten Metropolis–Algorithmus wurden zunächst die Basismechanismen allgemeiner kombinatorischer Optimierungsverfahren und deren Übertragung auf die iterative Instantiierung erläutert. Darauf aufbauend wurden stochastische Relaxation, Schwellwertakzeptanz und Sintflutalgorithmus als Varianten des Verfahrens, die ein deterministisches Annahmekriterium verwenden, eingeführt.

Genetische Algorithmen wurden als eine zweite Methode zur Lösung kombinatorischer Optimierungsprobleme vorgestellt. Nach einer allgemeinen Einführung in die Prinzipien von Selektion, Rekombination und Mutation wurden spezielle genetische Operatoren für die Anwendung in der iterativen Kontrolle vorgeschlagen. Diese haben zum Ziel, einerseits die bei genetischen Algorithmen vorhandene Möglichkeit zur inhärent parallelen Erkundung der Lösungsmenge zu nutzen, andererseits jedoch eine kurze Iterationsdauer zu gewähren.

Sowohl aus den Zeitanforderungen der Anwendungen als auch aus den Eigenschaften der vorgestellten Optimierungstechniken selbst ergibt sich ein starkes Interesse an der Untersuchung von Parallelisierungsansätzen, die im letzten Abschnitt vorgestellt wurden. Problemabhängige Strategien, wie beispielsweise die Verteilung geeigneter Ausschnitte des Attributflußgraphen wurden dabei nicht weiter untersucht; durch die vorher beschriebene parallele, datengetriebene Instantiierung, die zur Berechnung der Kostenfunktion notwendig ist, wird dieser Aspekt jedoch ebenfalls berücksichtigt. Anhand des Metropolis–Algorithmus wurden die multi– und unidirektionale sowie die periodisch oder dynamisch interagierende Suche als problemunabhängige Parallelisierungsstrategien mit unterschiedlich großem Kommunikationsaufwand vorgestellt. Zur grobgranularen Parallelisierung genetischer Algorithmen wird die Lösungsmenge auf mehrere Prozessoren verteilt. Zum Zustandsaustausch können dabei die gleichen Verfahren wie beim Simulated Annealing eingesetzt werden, es bietet sich jedoch zusätzlich an, den Austausch auf die n besten Elemente der lokalen Zustandsmenge zu erweitern.

Die im folgenden Kapitel vorgenommene experimentelle Überprüfung der datengetriebenen Instantiierung des Attributflußgraphen, der verschiedenen Optimierungsverfahren und der beschriebenen Parallelisierungsstrategien dient der Validierung des konzipierten Kontrollalgorithmus.

Kapitel 5

Ein Anwendungsbeispiel

Die experimentelle Evaluierung des im vorigen Kapitel konzipierten Kontrollalgorithmus und der aufgezeigten Parallelisierungsmöglichkeiten für die datengetriebene Instantiierung sowie die verschiedenen kombinatorischen Optimierungsverfahren ist Gegenstand dieses Kapitels. Als Pilotanwendung wurde eine Analyseaufgabe aus dem Bereich der Interpretation von Bildfolgen natürlicher Straßenverkehrsszenen gewählt, welche einerseits die Überprüfung der Verfahren in einem potentiell bedeutenden Szenario realistischer Größenordnung gestattet, sich andererseits jedoch durch eine Überschaubarkeit auszeichnet, die im Rahmen der vorliegenden Arbeit eine rasche Adaption des ursprünglich für den ERNEST-Kontrollalgorithmus entwickelten prozeduralen Wissens ermöglichte.

Der Aufbau der Wissensbasis, in der das für die Bestimmung der Fahrbahn und ihrer Markierungen benötigte Wissen abgelegt ist, wird in Abschnitt 5.1 erläutert, der auch eine kurze Beschreibung der initialen Segmentierung gibt. Im Anschluß daran wird in Abschnitt 5.2 die Parallelisierung der datengetriebenen Instantiierung des Attributflußgraphen evaluiert. Unter Verwendung eines Simulationspaketes zur Modellierung und Leistungsbewertung paralleler Programme wird dabei insbesondere die in Abschnitt 4.2 vorgeschlagene feingranulare Auflösung der Wissensbasis validiert. Abschließend untersucht Abschnitt 5.3 die Eignung der beschriebenen Optimierungsverfahren und gibt Ergebnisse für deren Parallelisierung in einem homogenen Netz lose gekoppelter Workstations.

5.1 Wissensbasis und initiale Segmentierung

Das in diesem Abschnitt vorgestellte, unter Verwendung der ERNEST–Netzwerkschale realisierte Modell zur Identifikation der Fahrbahn in natürlichen Straßenverkehrsszenen geht auf die in [Ste91] entwickelte Wissensbasis zurück. Das semantische Netzwerk besteht aus 13 Konzepten, die in zwei Spezialisierungs– und zwei Konkretisierungsebenen angeordnet sind. Bild 5.1 gibt einen Überblick über die elf Konzepte und die Kanten der ersten Spe-

zialisierungsebene; zwei nicht abgebildete Konzepte mit Spezialisierungsgrad 0 besitzen für die Analyse keine Bedeutung, sondern wurden nur eingeführt, um durch die Ausnutzung der Vererbung einen kompakteren Aufbau der Wissensbasis zu ermöglichen. [1]

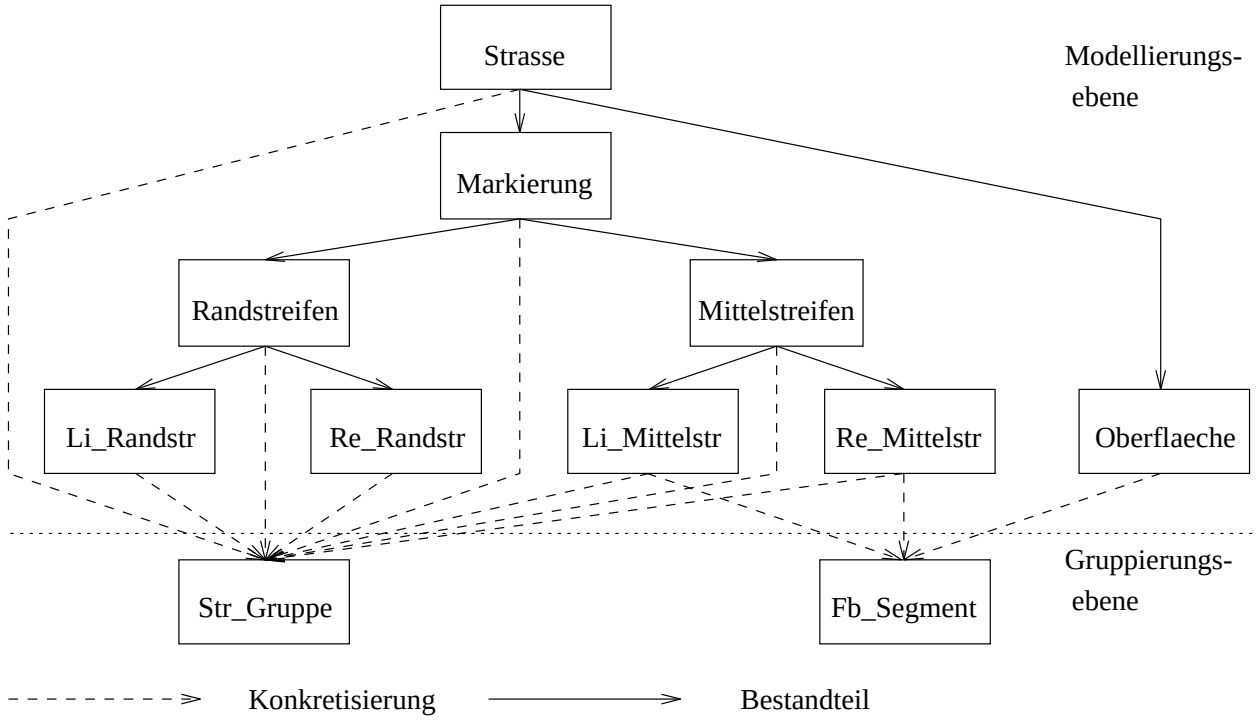


Bild 5.1: Überblick über die Wissensbasis des Anwendungsszenarios.

Einziges Zielkonzept ist das Konzept "Strasse", zu dessen Instantiierung die Fahrbahnoberfläche und die Markierung ermittelt werden müssen. Letztere setzt sich aus den Markierungen der Fahrbahnränder und –mitte zusammen, welche durch die Konzepte "Randstreifen" und "Mittelstreifen" modelliert werden. Beide Konzepte besitzen je zwei Bestandteile, welche zum einen den linken und rechten Fahrbahnrand, und zum anderen einen oder — im Falle einer dreispurigen Straße — zwei Trennstreifen zwischen den Fahrbahnen beschreiben.

Zu beachten ist, daß alle Bestandteilskanten als *obligatorisch* gekennzeichnet sind, obwohl — beispielsweise aufgrund von Verdeckungen durch ein Fahrzeug, man betrachte die Szenen in Bild 5.4 — nicht alle Teile der Fahrbahnmarkierung in einem Bild vorhanden sein müssen. Das Fehlen einzelner Teile kann zwar durch die Angabe mehrerer Modalitätsmengen und die Markierung der entsprechenden Kanten als *optional* leicht modelliert werden, würde jedoch bei der ursprünglich vorgesehenen Steuerung der Analyse durch den ERNEST–Kontrollalgorithmus zu einer zusätzlichen modellbedingten Aufspaltung des

[1] Für die durch die Überlassung ihrer Arbeit gewährte Unterstützung dankt der Autor Frau S. Richter, Forschungsgruppe "Kognitive Systeme" des Bayerischen Forschungszentrums für wissensbasierte Systeme (FORWISS, München), sehr herzlich.

Konzept	Attribute	Bedeutung
Strasse	–	–
Markierung	fluchtpunkt	Neuschätzung des Fluchtpunkts
	grp_konsistenz	Überprüfung auf mehrfache Zuordnung
Mittelstreifen	schnittwinkel	Schnittwinkel des li. und re. Mittelstreifens
	schnittpunkt	Schnittpunkt des li. und re. Mittelstreifens
Li_Mittelstr	flaeche	Fläche der Segmentgruppe
	seganz	Anzahl der Segmente der Segmentgruppe
	neigung	Steigung der Hauptachse der Segmentgruppe
	gleichung	Geradengleichung der Hauptachse
	position	Schwerpunkt der Segmentgruppe
Re_Mittelstr	analog zum Konzept Li_Mittelstr	
Randstreifen	schnittwinkel	Schnittwinkel des li. und re. Randstreifens
	schnittpunkt	Schnittpunkt des li. und re. Randstreifens
Li_Randstr	flaeche	Fläche der Segmentgruppe
	neigung	Steigung der Hauptachse der Segmentgruppe
	gleichung	Geradengleichung der Hauptachse
	position	Schwerpunkt der Segmentgruppe
Re_Randstr	analog zum Konzept Li_Randstr	
Oberflaeche	schwerpunkt	Schwerpunkt des Oberflächensegments
	einschluss	Anzahl der Einschlußsegmente
Fb_Segment	seg_anforderung	Einlesen der Regionensegmentierung
Str_Gruppe	grp_anforderung	Einlesen der Gruppierungsergebnisse

Bild 5.2: Überblick über die Attribute der Wissensbasis zur Fahrbahnidentifikation.

Suchbaums führen (vgl. Abschnitt 3.2.2). Um dies zu vermeiden, werden in [Ste91] alle Teile der Fahrbahnmarkierung als obligatorisch gekennzeichnet und gegebenenfalls durch eine "leere" Segmentgruppe instantiiert; insbesondere bei den hier betrachteten Szenen, in denen ausschließlich zweispurige Fahrbahnen vorhanden sind, ist dies für einen der beiden Mittelstreifen stets der Fall. Da in der Analysevorbereitung der iterativ–optimierenden Kontrolle obligatorische und optionale Kanten identisch behandelt werden (vgl. Abschnitt 4.2.2), wurde im Rahmen dieser Arbeit auf eine Änderung der Modellierung zugunsten einer größeren Übersichtlichkeit verzichtet; dies erspart uns nicht zuletzt eine zur Bewertung

unterschiedlicher Modalitäten notwendige Erweiterung des prozeduralen Wissens. Die beiden minimalen Konzepte "Fb_Segment" und "Str_Gruppe" stellen als Konkretisierungen der zur Fahrbahnmodellierung benötigten Konzepte eine Verbindung zur initialen symbolischen Beschreibung her. Ihre Aufgabe besteht darin, die Ergebnisse einer anfänglichen Regionensegmentierung und der Gruppierung der Regionen einzulesen. Dazu werden die zwei Attribute "seg_anforderung" und "grp_anforderung" verwendet, welche die einzigen Elemente der Wissensbasis sind, die direkt auf die initiale symbolische Beschreibung zugreifen.

Bild 5.2 gibt einen Überblick über die Attribute der Wissensbasis, die in erster Linie Informationen über die Richtung, Position und Fläche der Segmente und Segmentgruppen verwenden, um festzustellen, ob eine Gruppe oder ein Segment zur Instantiierung des jeweiligen Konzepts benutzt werden kann. Attribute und Instanzen werden mit Hilfe der Theorie der vagen Mengen (*Fuzzy–Sets*) [Zad65, Zad75] beurteilt, die im Bereich der Musteranalyse einen weitverbreiteten Ansatz zur Bewertung unsicherer Aussagen darstellen (vgl. auch [Nie90a, Sag90a]). Für eine ausführliche Erläuterung der zur Attributberechnung benutzten Methoden und der Bewertungskriterien verweisen wir auf [Ste91]; eine Weiterentwicklung des Modells wird in [Ric94] vorgestellt, und verwandte Ansätze werden unter anderem in [Lio86, Oza86] beschrieben.

Waren in der ursprünglichen Implementierung der Wissensbasis alle zur Beschreibung eines Konzeptes verwendeten Attribute (beispielsweise die fünf Attribute des Konzepts "Li_Mittelstr" aus Bild 5.2) in *einer* Attributbeschreibung mit dem Werttyp `RECORD` (vgl. Abschnitt 3.1) zusammengefaßt, so wurden diese zur Überprüfung des neu entwickelten Kontrollalgorithmus in ihre Komponenten zerlegt. Dies ermöglicht es zum einen, den Algorithmus an Flußgraphen mit größerer Knotenanzahl und Tiefe zu evaluieren, zum anderen vergrößert sich auch die der Anwendung inhärente Parallelität, was wir jedoch lediglich als Nebenaspekt der Neumodellierung bewerten wollen. Auf diese Weise wurde die Anzahl der Attributbeschreibungen in der Wissensbasis von 16 auf 50 erhöht, und der resultierende Attributflußgraph von 46 auf 119 Knoten vergrößert, die in elf (ursprünglich: sieben) Ebenen angeordnet sind. Bild 5.3 gibt einen Überblick über den Graphen, dessen Größe und geringe Strukturiertheit inbesondere die Notwendigkeit der automatischen Generierung aus der weitaus übersichtlicheren, begriffzentrierten Repräsentation in Bild 5.1 verdeutlicht[2]. Zwar bleibt die durch die Konkretisierungskanten gegebene Aufteilung der Wissensbasis in Gruppierungs– und Modellierungsebene weitgehend erhalten — die zu den Konzepten der Gruppierungsebene gehörenden Knoten finden sich ausschließlich auf den unteren drei Ebenen des Attributflußgraphen —, die Gliederung der Modellierungsebene durch die

[2] Der Graph wurde mit Hilfe der Erklärungskomponente der ERNEST-Systemschale [Pre92] automatisch generiert. Aufgrund des verwendeten Layoutalgorithmus, der ursprünglich nicht für die Darstellung des Attributflußgraphen entwickelt wurde, besitzt der Graph eine andere Orientierung als die Graphen in den Bildern 4.6 und 4.11. Knoten aufsteigenden Grades sind hier von oben nach unten angeordnet.

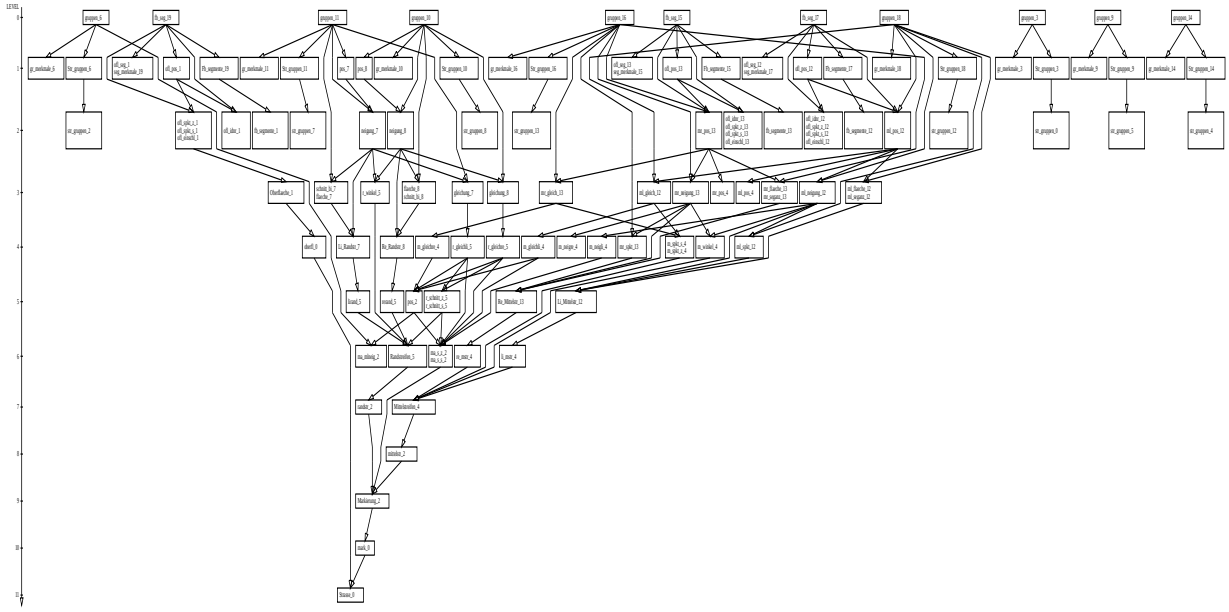


Bild 5.3: Der vollständige Attributflußgraph für die Wissensbasis zur Fahrbahnidentifikation aus Bild 5.1.

Bestandteilskanten wird jedoch aufgelöst. So werden beispielsweise die Kanten des Zielkonzepts ("Strasse") nicht gleichzeitig bewertet, sondern einerseits bereits auf der vierten Ebene des Graphen ("oberflaeche"), andererseits erst auf der zehnten Ebene ("markierung").

Als Untersuchungsmaterial standen drei Bildsequenzen mit je 29 Grauwertbildern der Größe 512×512 zur Verfügung, die im Abstand von 40 Millisekunden von einer in einem Fahrzeug fixierten, monokularen Kamera aufgenommen wurden. Das erste und letzte Bild jeder Sequenz sind in Bild 5.4 auf Seite 132 abgebildet; zwei aufeinanderfolgende Bilder der ersten Folge wurden bereits in Bild 1.2 auf Seite 3 gezeigt.

Die initiale symbolische Beschreibung der Szenen umfaßt eine Regionensegmentierung und eine Zusammenfassung von Segmenten zu Segmentgruppen. Während uns zur Segmentierung selbst keine weiteren Unterlagen zur Verfügung stehen, ist die richtungsorientierte Gruppierung der Segmente in [Ste90] beschrieben. In einem mehrstufigen Prozeß, der aus einer Segmentvorauswahl, der eigentlichen Gruppenbildung und einer anschließenden Gruppenfilterung und –vereinigung besteht, werden die Segmente aufgrund bestimmter Größen–, Form– und Grauwertkriterien ausgewählt und anhand einiger Orientierungsmerkmale (Richtung der Hauptträgheitsachsen, Lage des Schwerpunktes) zu Segmentgruppen zusammengefaßt, welche Kandidaten für die verschiedenen Markierungen darstellen. Eine ausführlichere Diskussion der verwendeten Verfahren und der Frage, inwieweit dabei bereits problem– oder szenenabhängiges Wissen eingesetzt wird, soll hier nicht geführt werden, da dies für eine Evaluierung des entwickelten Kontrollalgorithmus ohne Bedeutung ist. Um

Bild 5.4: Erstes und letztes Bild der drei verwendeten Bildfolgen (Abstand 1.2 sec).

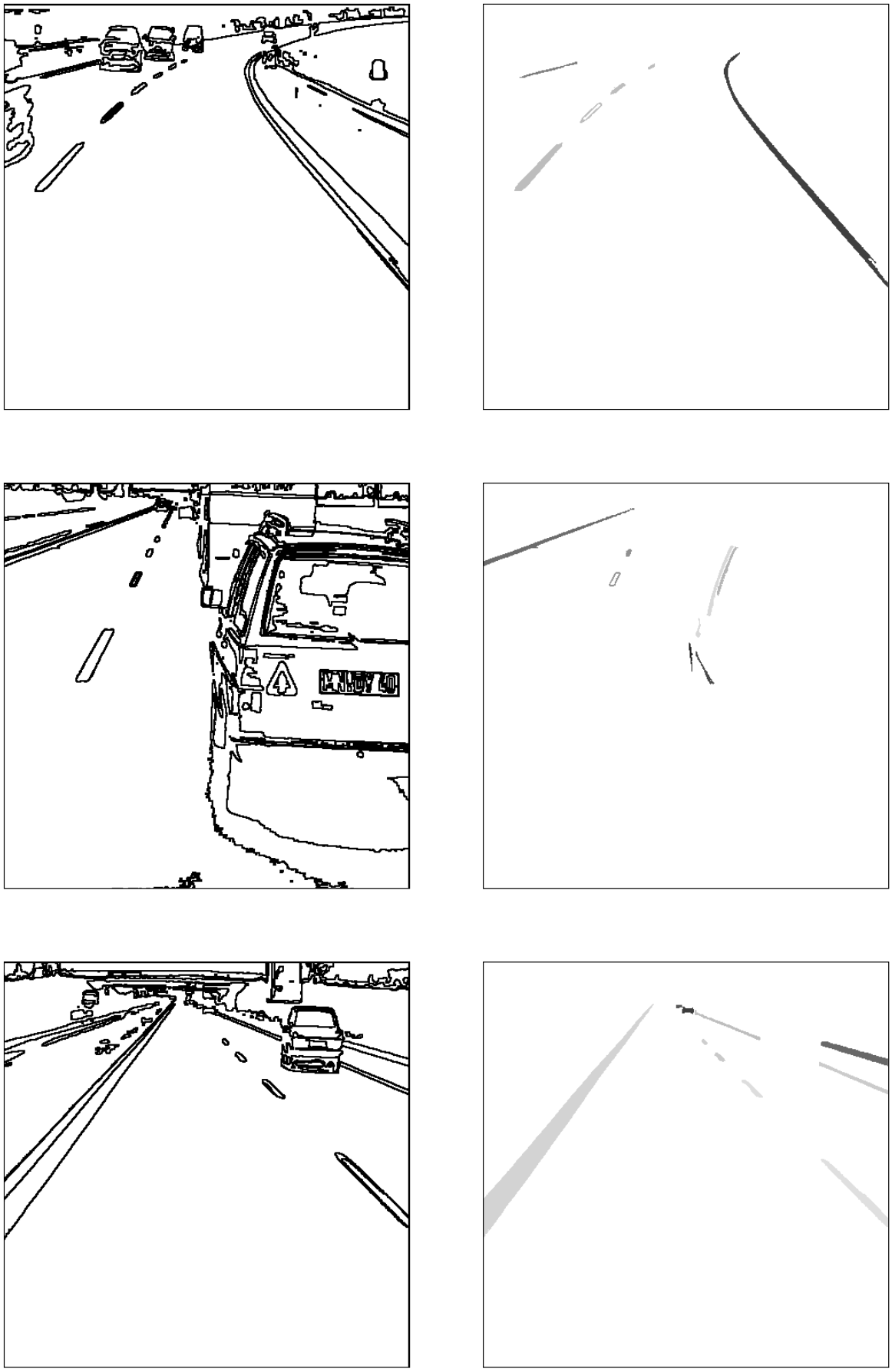


Bild 5.5: Ergebnisse von Segmentierung (links) und richtungsorientierter Vorauswahl und Gruppierung der Segmente (rechts).

einen visuellen Eindruck zu geben, zeigt Bild 5.5 auf Seite 133 für jede der drei Bildsequenzen ein Regionenbild und die Ergebnisse der Gruppierung. Zu Darstellungszwecken sind im Regionenbild (links) lediglich die Regionengrenzen eingetragen, und zur Veranschaulichung der Gruppierungsergebnisse (rechts) wurden die zu einer Gruppe gehörigen Segmente mit einem einheitlichen Grauwert versehen. Während für das gezeigte Bild der ersten Folge (oben) drei Segmentgruppen gefunden werden, die der tatsächlichen Fahrbahnmarkierung entsprechen, liefert die Vorauswahl bei den Bildern der zweiten und dritten Sequenz jeweils fünf Gruppen, von denen einige entweder von Fahrzeugsegmenten oder durch den Übergang zwischen Stand– und Grünstreifen verursacht werden.

Anhand der hier skizzierten Analyseaufgabe — der Identifikation von Fahrbahn und Markierung zur Erstellung einer Beschreibung des Straßenverlaufs — wird zunächst die parallele datengetriebene Instantiierung des Attributflußgraphen evaluiert, und anschließend die Parallelisierung der übergeordneten Optimierungsverfahren untersucht. Zur Auswertung werden die bereits in Kapitel 1 auf Seite 11 eingeführten Performanzmaße *Speedup* und *Effizienz* sowie das daraus abgeleitete Kriterium für die *Wirtschaftlichkeit* (Gleichungen (1.1–1.3)) verwendet. Alle Experimente wurden — sofern nicht anders angegeben — auf einer bzw. mehreren identischen Workstations mit ca. 124 MIPS Rechenleistung und 64 MB Hauptspeicher durchgeführt. Die Hauptspeichergröße erweist sich dabei lediglich bei den Untersuchungen zur *konkurrierenden* Instantiierung als relevant, da die *iterative* Kontrolle durch eine konstante (und in der vorliegenden Anwendung mit ca. einem Megabyte sehr geringe) Speicheranforderung gekennzeichnet ist.

5.2 Parallele datengetriebene Instantiierung

Ziel der in diesem Abschnitt beschriebenen Experimente ist die Überprüfung der in Kapitel 4 getroffenen Entwurfsentscheidungen für eine feingranulare Auflösung der Wissensbasis und die iterative Verarbeitung konkurrierender Hypothesen. Für die einleitend bereits erwähnte Durchführung einer Parallelrechner*simulation* spricht dabei vor allem die Möglichkeit, Anforderungen und Erfolgschancen der Verfahren für eine beliebige Prozessoranzahl abzuschätzen, ohne aufwendige Implementierungsarbeiten vornehmen zu müssen.

Das Programmpaket PEPP (*Perfomance Evaluation of Parallel Programs*) [3] [Dau92, Har93a, Har93b] ist ein Werkzeug zur Erstellung und Auswertung stochastischer Graphmodelle paralleler Programme, welches verschiedene Methoden zur Vorhersage der Laufzeit zur Verfügung stellt. Der folgende Exkurs gibt eine kurze Einführung in die ebenfalls in

[3] PEPP entstand am Lehrstuhl für Rechnerarchitektur und Verkehrstheorie (Informatik 7) der Friedrich-Alexander-Universität Erlangen-Nürnberg im Rahmen des Teilprojekts C1 "Messung, Modellierung und Bewertung von Multiprozessoren und Rechnernetzen" des Sonderforschungsbereichs 182. Für eine Einführung in die Leistungsbewertung mit PEPP danke ich meinem Freund und Kollegen Dr. F. Hartleb.

PEPP integrierte *Zustandsraumanalyse* [Tho86, Sah87, Söt90], welche in Abschnitt 4.3.1 bereits exemplarisch angewendet wurde und die Grundlage der anschließend beschriebenen Evaluierung bildet.

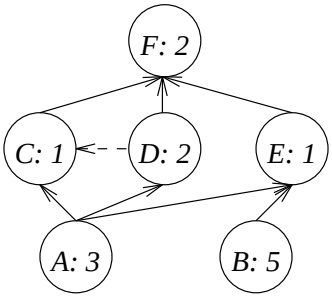
Zustandsraumanalyse

Ausgangspunkt der Zustandsraumanalyse ist ein gerichteter, azyklischer Graph $G = (V, E)$, der *Taskgraph*, dessen Knoten die auszuführenden (Teil-)Aufgaben eines Programms repräsentieren und dessen Kanten eine partielle Ordnung auf der Knotenmenge definieren.

Eine Kante e_{ij} zwischen den Knoten v_i und v_j besagt, daß die Bearbeitung der Aufgabe im Knoten v_i beendet sein muß, bevor die Berechnung im Knoten v_j starten kann. Die Knoten sind mit den Parametern einer Dichtefunktion beschriftet, welche die Knotenlaufzeit beschreibt; häufig wird hierzu die Exponentialverteilung benutzt, oder es wird — wie bei den weiter unten beschriebenen Experimenten — eine deterministische Laufzeit angenommen. In diesem Fall geben die Knotenbeschriftungen die mittleren Laufzeiten der jeweiligen Teilaufgaben an. Per constructionem ist auch der Attributflußgraph ein Taskgraph, sofern

/* *Algorithmus zur Zustandsraumanalyse* */
gegeben: ein Taskgraph $G = (V, E)$
definiere die Mengen $\Xi, V_{pred}(v)$ und $V_{succ}(v)$ wie in Algorithmus 4.8
bezeichne μ_v (ϱ_v) die mittlere (restliche) Bearbeitungszeit von Knoten v
$\forall \;\; v \in V$
initialisiere $\xi_v := -\lvert V_{pred}(v)\rvert$, $\varrho_v := \mu_v$
initalisiere zwei Mengen LAUFEND $:= \emptyset$ und REST $:= V$, $t_p := 0$
$\forall \;\; v \in$ REST
IF $\xi_v = 0$ $(\star)$
THEN REST := REST $\setminus \{v\}$; LAUFEND := LAUFEND $\cup \{v\}$ $(\star\star)$
bestimme $\varrho_{min} := \min_{v \in \text{LAUFEND}}\{\varrho_v\}$, setze $t_p := t_p + \varrho_{min}$
$\forall \;\; v \in$ LAUFEND
$\varrho_v := \varrho_v - \varrho_{min}$
IF $\varrho_v = 0$
THEN LAUFEND := LAUFEND $\setminus \{v\}$
$\forall \;\; w \in V_{succ}(v)$
$\xi_w = \xi_w + 1$
UNTIL (REST $= \emptyset \wedge$ LAUFEND $= \emptyset$)
Ergebnis: Dauer t_p der parallelen Ausführung von G

Bild 5.6: Zustandsraumanalyse zur Bestimmung der Dauer der parallelen datengetriebenen Instantiierung.



Zeit t_p	Zustand
0	$A \parallel B$
3	$D \parallel C \parallel B$
4	$D \parallel B$
5	E
6	F
8	$\emptyset$

$p = 3,\ t_1 = 14,\ t_p = 8,$
$s_p = 1.75,\ e_p = \mathbf{0.583}$

Zeit t_p	Zustand
0	$A \parallel B$
3	$D \parallel B$
5	$C \parallel E$
6	F
8	$\emptyset$

$p = 2,\ t_1 = 14,\ t_p = 8,$
$s_p = 1.75,\ e_p = \mathbf{0.875}$

Bild 5.7: Beispiel zur Zustandsraumanalyse nach Algorithmus 5.6. Dargestellt sind ein Taskgraph (links), die Zustände und das Ergebnis der Zustandsraumanalyse (mitte), und die Auswirkung der Einführung einer zusätzlichen, gestrichelt eingezeichneten Abhängigkeit (rechts).

wir die Attributflußknoten zusätzlich mit den Ergebnissen einer Laufzeitmessung versehen.

Bild 5.6 zeigt den in PEPP integrierten Algorithmus zur Zustandsraumanalyse, dessen Ziel die Bestimmung der parallelen Laufzeit eines durch einen Taskgraphen modellierten Programms ist. Dazu wird die Menge LAUFEND mit den Knoten initialisiert, die keine Vorgänger besitzen. Wird die Bearbeitung eines oder mehrerer Knoten mit minimaler Restlaufzeit ϱ beendet, so werden diese aus der Menge entfernt und — sofern vorhanden — durch ausführbare Nachfolger ersetzt. Die Analyse des Graphen endet, wenn alle Knoten bearbeitet sind (REST $= \emptyset$), und die in t_p akkumulierten minimalen Restlaufzeiten geben die parallele Laufzeit des Programms an. Die Anzahl p der eingesetzten Prozessoren entspricht der maximalen Knotenanzahl in der Menge LAUFEND, und den erzielbaren Speedup s_p erhalten wir aus der Division der Summe aller Knotenlaufzeiten durch die parallele Laufzeit.

Bild 5.7 gibt ein Beispiel und verdeutlicht, wie das Ergebnis der Analyse zu interpretieren ist. Da davon ausgegangen wird, daß stets genügend freie Prozessoren zur Verfügung stehen, wird jeder Knoten ausgeführt, sobald sämtliche Vorgänger ihre Aufgabe beendet haben. Unter dieser Annahme entstehen keine Wartezeiten, und der erreichte Speedup ist maximal. Dies gilt jedoch nicht für die Effizienz, da der gleiche Speedup durch die Einführung weiterer Abhängigkeiten (Kanten) mit weniger Prozessoren erzielt werden kann.

Da die (manuelle) Angabe günstiger Abhängigkeiten nur für Graphen mit geringer Knotenanzahl einfach möglich ist, erscheint es sinnvoller, anstelle einer unbegrenzten Prozessoranzahl eine Zahl p vorzugeben und Speedup und Effizienz bei der Verwendung von p Prozessoren zu berechnen. Dazu ist eine Zuordnung der Aufgaben zu den Prozessoren erforderlich, die in Bild 5.8 für zwei Prozessoren und die Knotenmenge $V = \{A, B, \ldots, F\}$ des Graphen aus Bild 5.7 dargestellt ist. Die aus den unterschiedlichen Zuordnungen re-

Knoten:	A	B	C	D	E	F
Proz.:	1	1	0	0	0	1

$t_1 = 14, t_2 = 11$

$s_2 = 1.27, e_2 = 0.64$

A	B	C	D	E	F
1	0	0	1	0	1

$t_1 = 14, t_2 = 9$

$s_2 = 1.56, e_2 = 0.78$

A	B	C	D	E	F
0	1	0	0	1	1

$t_1 = 14, t_2 = 8$

$s_2 = 1.75, e_2 = 0.875$

Bild 5.8: Alternative Abbildung der Knoten des Taskgraphen (A – F) aus Bild 5.7 auf zwei Prozessoren (0,1) und die daraus resultierenden Ergebnisse der Zustandsraumanalyse.

sultierenden Leistungsgrößen verdeutlichen, daß hier wiederum ein *Optimierungsproblem* vorliegt, zu dessen Lösung sich die in Abschnitt 4.3.2 diskutierten Verfahren anbieten. Die Minimierung der Kostenfunktion entspricht dabei der Optimierung der Leistungsgrößen, und zur Kostenberechnung ist eine Zustandsraumanalyse durchzuführen, welche die nunmehr vorgegebene Zuordnung zwischen Aufgaben und Prozessoren berücksichtigt. Dazu sind in Algorithmus 5.6 die mit ($\star$) und ($\star\star$) markierten Anweisungen zur Auswahl lauffähiger Knoten zu modifizieren. Falls sämtliche Vorgänger des Knotens v beendet sind ($\xi_v = 0$), ist zusätzlich zu überprüfen, ob der zugeordnete Prozessor zur Verfügung steht. Trifft dies für mehrere Knoten zu, so ist eine *Zuteilungsstrategie* erforderlich, da ein Prozessor zu jedem Zeitpunkt nur von einem Knoten belegt werden kann.

Die im nächsten Abschnitt beschriebenen Experimente verwenden den genetischen Algorithmus aus Bild 4.20 zur Minimierung der parallelen Laufzeit und die Zuteilungsstrategie *Shortest Job First* [Hof84], welche den Prozessor stets an den noch zu bearbeitenden Knoten mit der kleinsten mittleren Laufzeit vergibt. Zur Ermittlung der parallelen Laufzeit beim Einsatz von p Prozessoren, $p \geq= 2$, wurden je 100 Iterationen über einer Zustandsmenge Z_c der Größe $|Z_c| = 16$ ausgeführt. Für den durch Algorithmus 5.6 exakt lösbaren Spezialfall der unbegrenzten Prozessoranzahl ergab ein Vergleich mit der so ermittelten approximativen Lösung eine mittlere Abweichung der parallelen Laufzeiten von 3.5 Prozent.

Ergebnisse der konkurrierenden Instantiierung

Bereits bei der Beschreibung der Algorithmen zur datengetriebenen Instantiierung in Abschnitt 4.3.1 wurde von Voruntersuchungen berichtet, welche sich mit der simultanen Weiterverarbeitung der initialen Hypothesen befaßten. Das skizzierte Hauptproblem der *konkurrierenden* Instantiierung — die kombinatorische Explosion von Zwischenergebnissen auf den oberen Ebenen des Attributflußgraphen — soll zunächst anhand einiger experimenteller Ergebnisse belegt werden, bevor wir uns anschließend der *iterativen* Instantiierung zuwenden, die als Teil der Kostenberechnung in die Optimierungsverfahren aus Abschnitt

4.3.2 eingebettet ist.

	1-best			2-best		
Konzept	$t_1[s]$	$t_p[s]$	p	$t_1[s]$	$t_p[s]$	p
Strasse	0.339	0.139	29	42.577	21.075	29
Markierung	0.293	0.127	23	37.039	18.338	23
Mittelstreifen	0.038	0.019	15	0.050	0.029	15
Randstreifen	0.022	0.016	8	0.305	0.295	8
Li_Mittelstr	0.014	0.013	7	0.020	0.014	7
Re_Mittelstr	0.010	0.010	7	0.010	0.010	7
Li_Randstr	0.010	0.010	3	0.014	0.014	3
Re_Randstr	0.010	0.010	3	0.015	0.015	3

Bild 5.9: Parallelisierung der datengetriebenen Instantiierung des Attributflußgraphen: CPU–Zeiten der konkurrierenden Instantiierung.

Bild 5.9 faßt die Ergebnisse der konkurrierenden Instantiierung für die Konzepte der Modellierungsebene aus Bild 5.1 zusammen, wobei die Konzepte nach absteigender Größe und Tiefe des Attributflußgraphen angeordnet sind. Für die Konzepte der Gruppierungsebene und das Konzept "Oberflaeche" konnten mit der verwendeten Software-Uhr keine sinnvollen Meßwerte gewonnen werden. Die Spalten der Tabelle enthalten die sequentielle Instantiierungsdauer t_1, die sich aus der Summe der gemessenen Knotenlaufzeiten ergibt, und die parallele Instantiierungsdauer t_p, welche durch den Einsatz von p Prozessoren erzielt wird. Die Prozessoranzahl p gibt hierbei die (über alle Bilder gemittelte) maximale Größe der Menge LAUFEND aus Algorithmus 5.6 an; beispielsweise gilt in Bild 5.7 $p = 3$ für die in der Mitte abgebildete Zustandraumanalyse und $p = 2$ bei der Einführung der zusätzlichen Abhängigkeit zwischen den Knoten D und C. Die mit *1-best* und *2-best* markierten Spalten geben die Ergebnisse für eine Pruningstrategie wider, bei der nur die Hypothesen mit den n (hier: $n = 1, 2$) besten Bewertungen weiterverfolgt werden.

Bereits für die Konzepte "Randstreifen" und "Mittelstreifen", in größerem Maße jedoch für "Markierung" und "Strasse", ist eine starke Zunahme der Analysedauer zu erkennen, die nur durch die sehr rigide Beschneidung der Hypothesenmenge für $n = 1$ verhindert werden kann. Allerdings wird die optimale Interpretation in diesem Fall lediglich bei 60 der 90 Bilder erreicht, und nur bei einer Erhöhung der Hypothesenanzahl ($n = 2$) kann für alle Bilder die beste symbolische Beschreibung ermittelt werden. Die daraus resultierenden hohen Rechenzeiten können — gemessen an der Zahl p der verwendeten Prozessoren — selbst bei einem massiv-parallelen Vorgehen nur unwesentlich reduziert werden, da einzelne Attribute den Flaschenhals der Instantiierung bilden: Die sequentielle Laufzeit t_1 wird zu circa 95 Prozent von zwei Attributberechnungen im Konzept "Markierung" verursacht,

welche die Hypothesen aus den Teilnetzen für die Rand– und Mittelstreifen kombinieren. Da diese Ausschnitte des Attributflußgraphen unabhängig voneinander instantiiert werden, sind hier eine Vielzahl von Kombinationen zu überprüfen, und solche Hypothesen zu verwerfen, bei denen eine mehrfache Interpretation einzelner Segmentgruppen — beispielsweise kann eine Gruppe zugleich als linker Randstreifen und als Mittelstreifen interpretiert werden — erfolgt.

Nachdem sich die in Abschnitt 4.3.1 skizzierte Gefahr der kombinatorischen Explosion somit bereits für eine relativ kleine Wissensbasis experimentell bestätigt, besitzen die in Bild 5.10 gezeigten Leistungsgrößen für die Parallelisierung lediglich ergänzenden Charakter. Diese wurden für $n = 1$ (links) und $n = 2$ (rechts) durch die Abbildung der Flußgraphknoten auf p Prozessoren, $2 \leq p \leq 30$, ermittelt, da $p = 30$ die über alle Bilder zu beobachtende größte Anzahl gleichzeitig "aktiver" Knoten war, und somit der maximal benötigten Prozessoranzahl entspricht. Die erzielten Ergebnisse sind als optimistische Abschätzungen des Erfolgs einer Parallelrechnerimplementierung zu verstehen, da der hohe Kommunikationsaufwand beim Austausch der Hypothesenmengen in der Simulation unberücksichtigt bleibt. Der bei der Instantiierung der Konzepte "Strasse" und "Markierung" auftretende Engpaß durch die Konsistenzüberprüfung und Fluchtpunktberechnung in den beiden Attributberechnungen des Konzepts "Markierung" bestätigt sich hier durch das Erreichen des optimalen Arbeitspunktes bei der Verwendung von zwei Prozessoren.

Ergebnisse der iterativen Instantiierung

Anhand der in die iterative Optimierung integrierten Instantiierung des Attributflußgraphen, bei der anstelle einer Hypothesenmenge lediglich ein Zwischenergebnis pro Knoten weiterverarbeitet wird, wollen wir insbesondere die Entscheidung für eine feingranulare Parallelisierung validieren. Dazu stellt Bild 5.11 die Instantiierungsdauern (mittlere CPU-Zeiten in Sekunden) einer *feinen* und einer *groben* Parallelisierung gegenüber. Während erstere durch die Überführung der Wissensbasis in den Attributflußgraphen entsteht, gewinnen wir die gröbere Parallelisierung durch die Zusammenfassung der Laufzeiten der Attributflußknoten in den Knoten der von Algorithmus 4.1 erzeugten Zwischenrepräsentation, welche die benötigten Instanzen repräsentieren; der Taskgraph besitzt in diesem Fall also die Struktur des nach der ersten Phase der Analysevorbereitung vorliegenden expandierten Netzwerks. Für beide Parallelisierungsansätze sind Knotenanzahl und Tiefe des jeweiligen Taskgraphen — sie entsprechen der Mächtigkeit der Mengen $V_{\mathcal{E}}$ und $V_{\mathcal{A}}$ aus Abschnitt 4.2.2 bzw. der Tiefe δ_{teil} und d_A aus den Gleichungen (3.15) und (4.3) — zusätzlich angegeben.

Als wichtiges Ergebnis ist hervorzuheben, daß der in Bild 5.9 zu beobachtende starke Anstieg der Dauer im Falle der iterativen Instantiierung sowohl für die grobe als auch für die feine Parallelisierung vermieden wird. Aufgrund des gewählten Versuchsaufbaus ist es

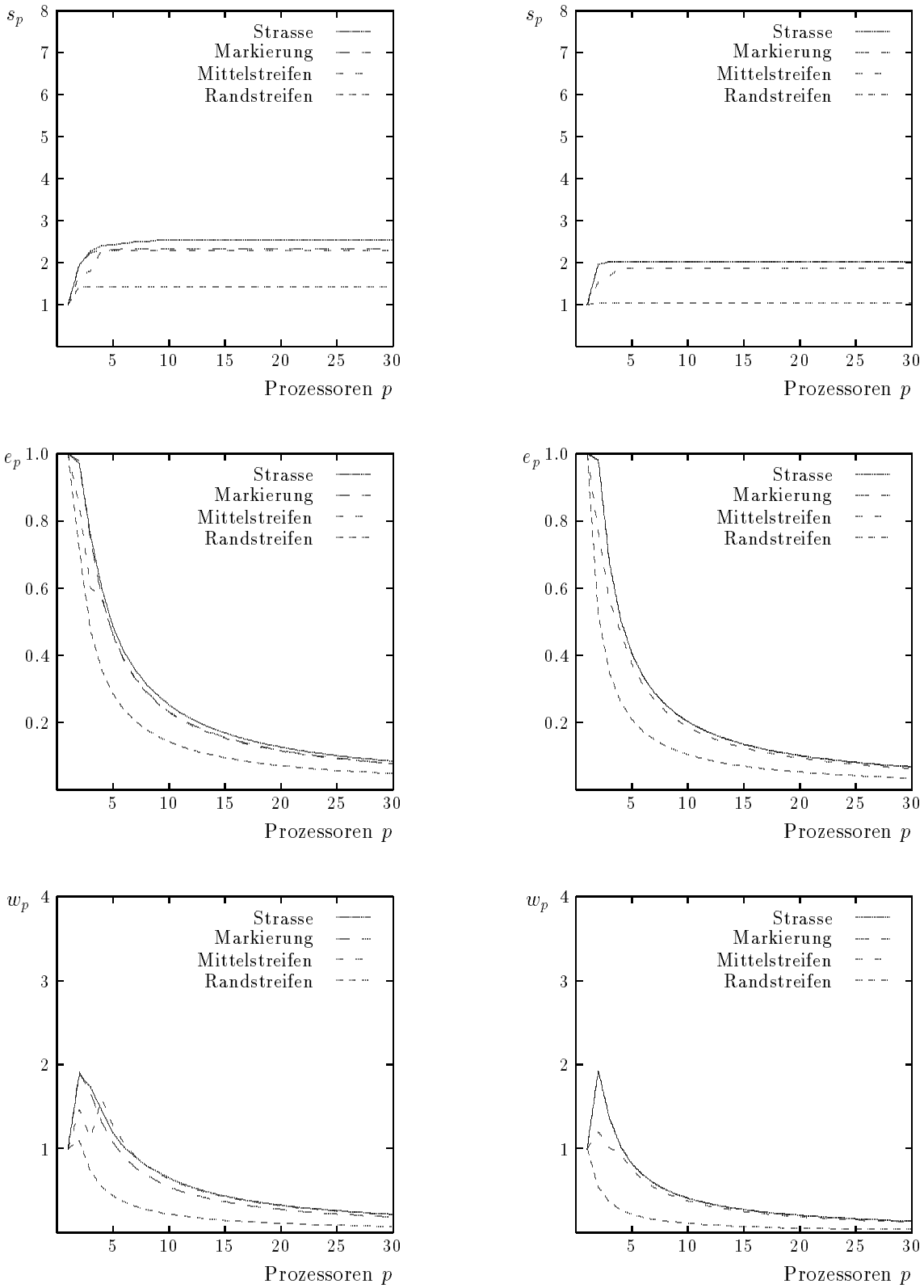


Bild 5.10: Parallelisierung der datengetriebenen Instantiierung des Attributflußgraphen: Speedup (oben), Effizienz (mitte) und Arbeitspunkt (unten) der konkurrierenden Instantiierung. Dargestellt sind die Ergebnisse bei einer Beschränkung auf die Hypothesen mit der besten (n=1, links) bzw. den zwei besten (n=2, rechts) Bewertungen.

	grob					fein				
Konzept	$t_1[s]$	$t_p[s]$	$\|V_\mathcal{E}\|$	δ_{teil}	p	$t_1[s]$	$t_p[s]$	$\|V_\mathcal{A}\|$	d_A	p
Strasse	0.042	0.018	20	4	5	0.042	0.006	119	11	25
Markierung	0.036	0.016	15	3	5	0.036	0.005	101	9	20
Mittelstreifen	0.017	0.010	8	2	5	0.017	0.003	58	7	14
Randstreifen	0.009	0.006	2	6	3	0.009	0.003	32	6	7
Li_Mittelstr	0.005	0.005	3	1	2	0.005	0.002	21	5	7
Re_Mittelstr	0.007	0.006	3	1	2	0.007	0.005	21	5	7
Li_Randstr	0.003	0.003	2	1	1	0.003	0.003	10	4	3
Re_Randstr	0.003	0.003	2	1	1	0.003	0.002	10	4	3

Bild 5.11: Parallelisierung der datengetriebenen Instantiierung des Attributflußgraphen: CPU–Zeiten bei iterativer Instantiierung des expandierten Netzwerkes (grob) und des Attributflußgraphen (fein). Zusätzlich sind die Kenngrößen der jeweiligen Taskgraphen angegeben, auf die im Text eingegangen wird.

zunächst klar, daß die parallele Laufzeit für die Instantiierung des Attributflußgraphen geringer sein muß als für den groben Parallelisierungsansatz. Aus Bild 5.11 wird jedoch auch deutlich, daß der feingranulare Ansatz die drei– bis fünffache Anzahl an Prozessoren benötigt, wenn bei der Abarbeitung der Knoten keine Wartezeiten entstehen sollen. Es ist daher apriori nicht ersichtlich, welcher der beiden Ansätze bei einer begrenzten Prozessoranzahl und unter den Gesichtspunkten von Effizienz und Wirtschaftlichkeit zu bevorzugen ist.

Um diese Frage zu klären, betrachten wir die Leistungsgrößen in Bild 5.12, die ebenfalls durch die Bestimmung einer optimalen Zuordnung von Flußgraphknoten und Prozessoren ermittelt wurden. Aus Darstellungsgründen beschränken wir uns hier auf die Wiedergabe der Kurven von Speedup, Effizienz und Arbeitspunkt (von oben nach unten) für die komplexeren Konzepte der Wissensbasis. Bei der groben Parallelisierung (links) ist eine Diskussion der Ergebnisse nur bis zur Knotenanzahl $|V_\mathcal{E}|$ des expandierten Netzwerkes sinnvoll, die dem linken Teil der Tabelle in Bild 5.11 zu entnehmen ist. Bei der Verwendung von sechs bzw. vier Prozessoren (zur Instantiierung der Konzepte "Strasse" und "Markierung" bzw. "Mittelstreifen" und "Randstreifen") wird der größte Speedup erreicht; ein optimaler Arbeitspunkt stellt sich jedoch bereits bei einer Parallelisierung mit zwei Prozessoren ein. Die dabei erzielte Effizienz liegt zwischen 0.90 ("Strasse") und 0.79 bei der Instantiierung des Konzepts "Randstreifen".

Bei der feinen Parallelisierung wird die maximale Effizienz zwar ebenfalls bereits bei $p = 2$ erreicht, ist jedoch mit Werten zwischen 0.98 und 0.96 um circa 17 Prozent größer. Die Vergrößerung des Speedups bei der Hinzunahme weiterer Prozessoren und die gleichmäßigere Auslastung aufgrund der feineren Elementareinheiten bewirken zudem, daß sich der

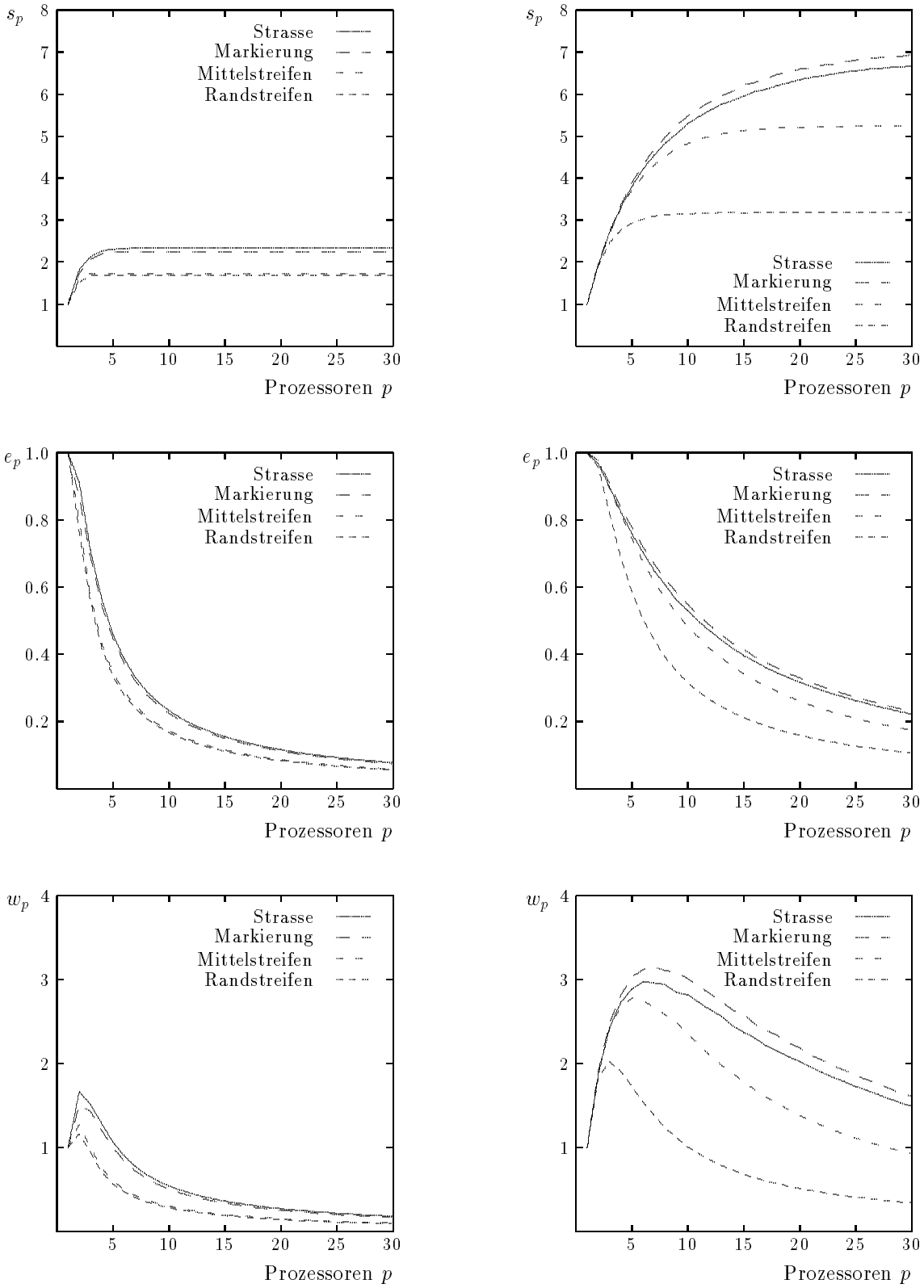


Bild 5.12: Parallelisierung der datengetriebenen Instantiierung des Attributflußgraphen: Speedup (oben), Effizienz (mitte) und Arbeitspunkt (unten) der iterativen Instantiierung. Im linken Teil sind die Ergebnisse für die Instantiierung kompletter Konzepte abgebildet (grobe Granularität); der rechte Teil gibt die Ergebnisse für den Attributflußgraphen (feine Granularität) wieder.

optimale Arbeitspunkt später — für das Zielkonzept "Strasse" bei $p = 7$ — einstellt und durchweg bessere Werte für alle drei Leistungsgrößen erzielt werden.

Abschließend können wir festhalten, daß die Wahl der Elementareinheiten in Abschnitt 4.2.1 durch den Vergleich mit der grobgranularen Parallelisierungsstrategie experimentell uneingeschränkt bestätigt wird. Zusätzlich belegt der Verlauf der Kurven für den Speedup und die Wirtschaftlichkeit, daß die für verschiedene parallele Netzwerkformalismen — man vergleiche die Literaturhinweise in Kapitel 1 — geforderte Realisierung jeder Elementaroperation auf einem eigenen Prozessor offensichtlich nicht erforderlich ist. Vielmehr erweist es sich als sinnvoll, eine der Aufgabenstellung oder der Größe der Wissensbasis angemessene Prozessoranzahl zu bestimmen und diese optimal zu nutzen; das in diesem Abschnitt beschriebene Vorgehen verdeutlicht dabei den Nutzen geeigneter Simulationswerkzeuge und allgemein einsetzbarer Optimierungstechniken. Nachdem damit die wichtigsten Ergebnisse zur parallelen datengetriebenen Instantiierung nochmals zusammengefaßt sind, wenden wir uns nun den Experimenten zur Parallelisierung der übergeordneten kombinatorischen Optimierung zu.

5.3 Parallelisierung der kombinatorischen Optimierungsverfahren

Nachdem die Notwendigkeit eines iterativen Kontrollalgorithmus durch die im letzten Abschnitt beschriebenen Untersuchungen auch experimentell belegt wurde, werden nun die in Abschnitt 4.3.2 diskutierten Optimierungsverfahren und ihre Parallelisierung evaluiert. Dabei stehen die folgenden Fragestellungen im Vordergrund:

- Welcher der vorgestellten Algorithmen besitzt die besten Konvergenzeigenschaften, findet also eine optimale Lösung mit einer möglichst geringen Anzahl an Iterationen?
- Welchen Erfolg zeigt eine Parallelisierung der Optimierung unter Verwendung der in Abschnitt 4.4 beschriebenen Verfahren? Hierbei ist insbesondere von Interesse, ob die Effizienz der Suche von einem vermehrten Zustandsaustausch profitiert.
- Schließlich ist zu untersuchen, ob durch eine größere Effizienz der Suche die zu erwartenden Verluste aufgrund des höheren Kommunkationsaufwands kompensiert werden können.

Eine Klärung der ersten Frage ist insbesondere dann von Interesse, wenn die Optimierung nicht parallelisiert werden kann, entweder weil keine geeignete Hardware zur Verfügung steht oder weil sich die Kommunikation zwischen den Prozessoren als zu aufwendig erweist. Zur Untersuchung der letztgenannten Punkte wurden die verschiedenen Suchverfahren aus Abschnitt 4.3.2 auf der Basis eines Client/Server–Modells in einem Netzwerk identischer

Workstations implementiert. Zur Interprozeßkommunikation zwischen dem Client und den bis zu fünf Servern wurde die*parallele, virtuelle Maschine* PVM [Sun90a, Gei91, Gei92] — eine Unterprogrammbibliothek, die sich als De-facto–Standard zur Programmierung verteilter Anwendungen in lokalen Netzen etabliert hat — verwendet. Eine Beschränkung auf fünf Server mußte erfolgen, da nicht mehr *identische* Workstations zur Verfügung standen. Die Hinzunahme zusätzlicher Rechner führte zwar im allgemeinen zu einer weiteren Beschleunigung der Analyse; die dabei entstehende heterogene Konfiguration mit Prozessoren unterschiedlicher Geschwindigkeit erschwert jedoch eine Beantwortung der oben aufgeworfenen Fragen erheblich.

Bevor wir mit einem Vergleich der Optimierungsverfahren beginnen, muß noch kurz auf die Methode zur Auswertung der Experimente eingegangen werden. Wie bei der Erläuterung der Verfahren deutlich wurde, ist sowohl die Anzahl der durchzuführenden Iterationen als auch die Qualität der erreichten Lösung von der zufälligen Wahl eines Startzustandes oder — im Falle des genetischen Algorithmus — einer initialen Zustandsmenge abhängig; die Generierung von Nachfolgern und das Annahmekriterium des Metropolis–Algorithmus erfordern weitere zufällige Entscheidungen. Die mehrfache Ausführung des jeweiligen Algorithmus und die Bestimmung der als *Konvergenzgeschwindigkeit* bezeichneten Größe

$$\nu(n) \quad = \quad P(\frac{|\phi_n - \phi_{min}|}{|\phi_{max} - \phi_{min}|} \leq \varepsilon) \tag{5.1}$$

aus der Stichprobe der durchgeführten Experimente sorgt dafür, daß der Einfluß derartiger zufälliger Entscheidungen auf die Beurteilung der Verfahren reduziert wird. Die Konvergenzgeschwindigkeit gibt dabei an, mit welcher Wahrscheinlichkeit der Algorithmus nach n Iterationen eine Lösung findet, die nur unwesentlich größere Kosten verursacht als eine optimale Lösung aus der Menge Z_{min}. Im Falle der hier verwendeten Kostenfunktion (4.10) gilt die Ungleichungskette $0.0 = \phi_{min} \leq \phi_n \leq \phi_{max} = 1.0$, wodurch sich Gleichung (5.1) zu

$$\nu(n) \quad = \quad P(\phi_n - \phi_{min} \leq \varepsilon) \tag{5.2}$$

vereinfacht. Als zulässige Abweichung von den minimalen Kosten wurde in den im folgenden beschriebenen Experimenten ein Schwellwert von $\varepsilon = 0.001$ gewählt, und jedes der 87 Bilder wurde mit zwei verschiedenen Initialisierungen des Zufallszahlengenerators analysiert. Aus der somit für jedes Verfahren vorliegenden Stichprobe von 174 Versuchen wurden alle Werte (Anzahl der Iterationen bzw. Verweilzeit) entfernt, die mit Hilfe der 4σ–Regel [4] [Sac92, S. 219] als Ausreißer identifiziert werden konnten; dies betraf im Mittel vier Werte.

[4] Die 4σ–Regel besagt, daß beim Vorliegen von mindestens zehn Meßwerten ein Wert dann als Ausreißer verworfen werden darf, wenn er außerhalb des Bereichs $[\bar{x} - 4s, \bar{x} + 4s]$ liegt, wobei Mittelwert $\bar{x}$ und Standardabweichung s ohne den fraglichen Einzelwert zu berechnen sind.

Die Beschreibung der Rahmenbedingungen wird durch eine Abschätzung der Größe der Zustandsmenge abgeschlossen: Im Attributflußgraphen des Konzepts "Strasse", das bei den im folgenden beschriebenen Experimenten stets das Analyseziel darstellt, existieren elf initiale Knoten, welche die Gruppierungs– und Segmentierungsergebnisse einlesen und mittels der Restriktionen aus den Attributbeschreibungen eine Vorauswahl vornehmen. Dabei werden für die acht Knoten zum Attribut "grp_anforderung" im Mittel sechs konkurrierende Hypothesen erzeugt. Die restlichen drei Knoten repräsentieren das Attribut "seg_anforderung" aus dem Konzept "Fb_Segment" — hiervon werden drei Instanzen benötigt, vgl. Bild 5.1 —, zu denen lediglich eine initiale Hypothese erzeugt wird, da nur das größte Segment als mögliches Fahrbahnsegment betrachtet wird. Für die bei der kombinatorischen Optimierung zu untersuchende Zustandsmenge erhalten wir somit durch Gleichung (4.18) eine Mächtigkeit von $|Z| = 6^8 \cdot 1^3 \cdot 1^{20} \approx 1.7 \cdot 10^6$, wobei sich der letzte Faktor daraus ergibt, daß jeder der 20 Konzeptknoten im Attributflußgraph genau eine Modalitätsmenge besitzt.

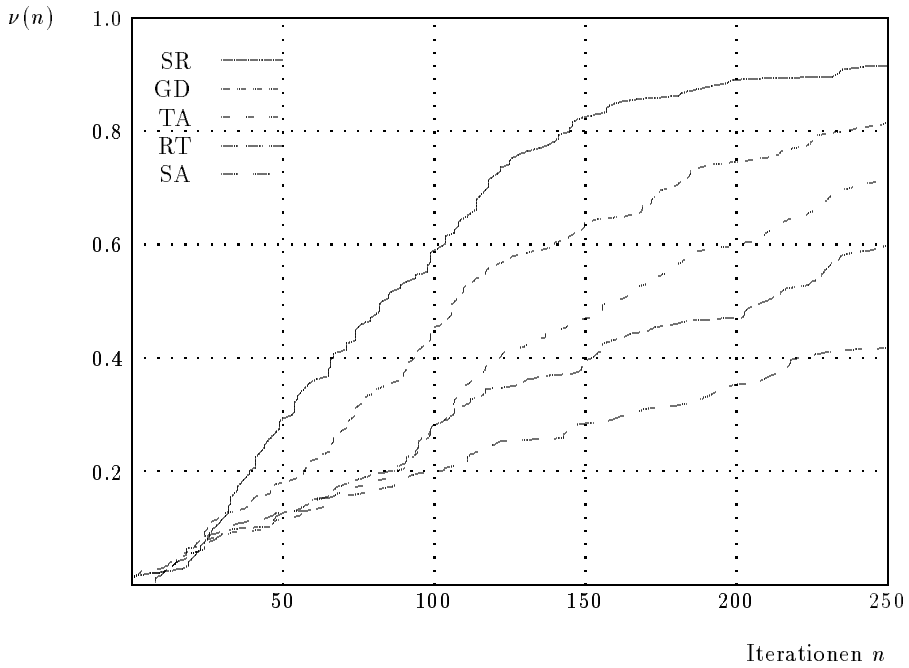


Bild 5.13: Ergebnisse für den Vergleich der kombinatorischen Optimierungsverfahren: Konvergenzgeschwindigkeit bei unterschiedlichen Annahmekriterien. Die Abkürzungen folgen der Bezeichnung der Verfahren im Text (SR: stochastische Relaxation, GD: Sintflutalgorithmus, TA: Schwellwertakzeptanz, RT: Record-to-Record-Travel, SA: Simulated Annealing).

Die Konvergenzgeschwindigkeiten der verschiedenen Optimierungsverfahren, die durch die Variation des Annahmekriteriums aus dem allgemeinen Algorithmus 4.12 entstehen, sind in Bild 5.13 gegenübergestellt. Vergleichen wir zunächst den Metropolis–Algorithmus (SA) mit den verschiedenen deterministischen Annahmekriterien, so bestätigt sich hier im wesentlichen die auch in [Due90, Due93a] anhand verschieden großer Traveling-Salesman-Probleme festgestellte Überlegenheit von Sintflutalgorithmus (GD), Schwellwertakzeptanz (TA) und Record-to-Record-Travel (RT), die deutlich weniger Iterationen benötigen, um eine optimale Lösung zu finden. So wird — dies als Beispiel zur Lesart des Diagramms — ein Zustand mit minimalen Kosten vom Metropolis-Algorithmus nach 200 Iterationsschritten mit einer Wahrscheinlichkeit von $\nu \approx 0.35$ eingenommen, während diese beim Sintflutalgorithmus mit $\nu \approx 0.75$ mehr als doppelt so groß ist. Die ebenfalls in [Due90, Due93a] konstatierte Verkürzung der mittleren Iterations*dauer* bei der Verwendung eines deterministischen Annahmekriteriums war in den durchgeführten Experimenten hingegen *nicht* zu beobachten.

Die in der vorliegenden Anwendung festzustellende sehr gute Konvergenzgeschwindigkeit des stochastischen Relaxationsverfahrens (vgl. die Kurve SR in Bild 5.13), bei dem Zustände mit höheren Kosten gemäß Gleichung (4.29) stets abgelehnt werden, kann als ein Indiz dafür gewertet werden, daß die Kostenfunktion nur wenige lokale Minima besitzt. Nach der anhand von Bild 4.14 geführten Diskussion des Annahmekriteriums ist jedoch klar, daß eine generelle Überlegenheit des Verfahrens sicherlich nicht angenommen werden sollte.

Die Bilder 5.14 und 5.15 dienen einerseits zum Vergleich der stochastischen Relaxation mit den genetischen Algorithmen aus Abschnitt 4.3.2, und untersuchen andererseits den Einfluß der Größe der Zustandsmenge auf die Konvergenzgeschwindigkeit der genetischen Optimierung. In Bild 5.14 sind zunächst das stochastische Relaxationsverfahren (SR) und zwei genetische Algorithmen gegenübergestellt, die beide mit einer Zustandsmenge der Größe $|Z_c| = 16$ arbeiten. Die fein gestrichelte Kurve (SP-GA) gibt die Konvergenzgeschwindigkeit für den 1-Punkt–Algorithmus aus Bild 4.19 an, bei dem ein herkömmlicher 1-Punkt–Crossoveroperator und eine feste Mutationswahrscheinlichkeit $p_m = 0.05$ eingesetzt wurden. Die durchgezogene Kurve (MP-GA) zeigt die Konvergenzgeschwindigkeit für den im Rahmen der optimalen Instantiierung entwickelten genetischen Algorithmus, der sich durch eine variable Mutationswahrscheinlichkeit sowie die Verwendung eines Mehrpunkt–Crossoveroperators auszeichnet (vgl. Bild 4.20 auf Seite 118) und daher im folgenden als *Mehrpunkt–Algorithmus* bezeichnet wird.

Bei einer Bewertung der Algorithmen ist zunächst festzuhalten, daß gegenüber der stochastischen Relaxation eine nochmalige Steigerung der Konvergenzgeschwindigkeit erzielt wird, der Standardalgorithmus (SP-GA) bei der hier gewählten Größe der Zustandsmenge jedoch etwas häufiger in einem Nebenminimum der Kostenfunktion steckenbleibt. Der Mehrpunkt–Algorithmus (MP-GA) dagegen liefert eine optimale Lösung mit einer Wahr-

scheinlichkeit von über 95 Prozent und kann in der hier untersuchten Anwendung als der Algorithmus mit den besten Konvergenzeigenschaften bezeichnet werden. Die bis $n \approx 35$ zu beobachtende größere Konvergenzgeschwindigkeit des Standardalgorithmus kann diese Wertung nicht in Frage stellen, wenn wir zusätzlich die Iterationsdauer berücksichtigen. In Abschnitt 4.4 wurde dargelegt, daß in einer Iteration des Standardalgorithmus die Kosten sämtlicher (hier: 16) Zustände der Menge Z_c zu berechnen sind — dazu ist die datengetriebene Instantiierung des Attributflußgraphen notwendig —, während dies beim Mehrpunkt–Algorithmus nur für genau einen neuen Zustand erfolgen muß. Ein Iterationsschritt des letztgenannten Algorithmus ist somit nicht aufwendiger als im Falle der stochastischen Relaxation, kostet jedoch nur einen Bruchteil ($1/|Z_c|$, hier: 1/16) der Zeit, die der Standardalgorithmus benötigt. Legen wir die Rechenzeit als zusätzliches Kriterium zugrunde, so schneidet der Mehrpunkt–Algorithmus also bereits in diesem Bereich besser ab.

Da die Iterationsdauer beim Mehrpunkt–Algorithmus nahezu unabhängig von der Größe der Zustandsmenge ist — ein mit der Größe wachsender, geringer Mehraufwand entsteht bei der Bestimmung der besten Zustände aus den Mengen Z_c und Z_t —, darf letztere viele

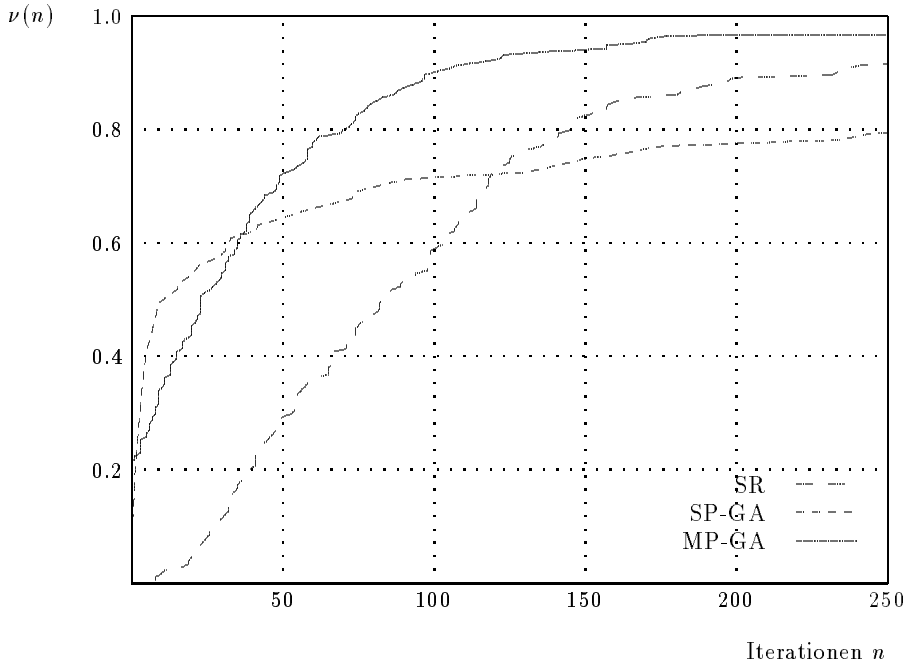


Bild 5.14: Vergleich der Konvergenzgeschwindigkeit von stochastischer Relaxation und genetischer Optimierung. Die Abkürzungen folgen der Bezeichnung der Verfahren im Text (SR: stochastische Relaxation, SP-GA: 1-Punkt–Algorithmus, MP-GA: Mehrpunkt–Algorithmus).

Elemente enthalten, um die parallele Erkundung des Lösungsraumes rascher voranzutreiben. In Bild 5.15 wird die Konvergenzgeschwindigkeit für Zustandsmengen verschiedener Größe verglichen. Die kaum vorhandenen Unterschiede zwischen den Kurven für $|Z_c| = 16$

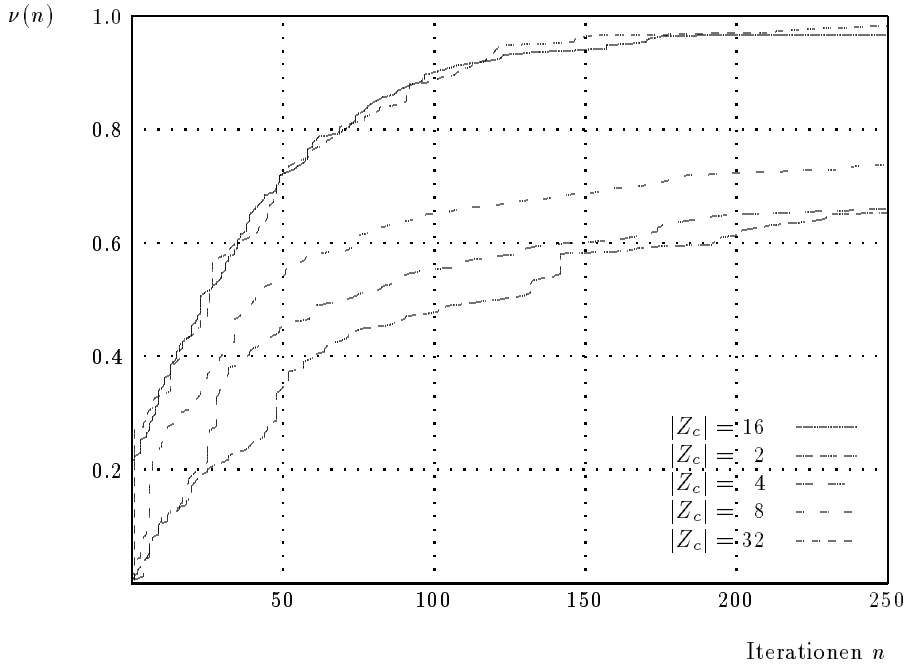


Bild 5.15: Konvergenzgeschwindigkeit des genetischen Algorithmus (MP-GA) bei unterschiedlicher Größe der Zustandsmenge.

und $|Z_c| = 32$ zeigen dabei, daß die oben bereits erwähnte Verwendung einer Zustandsmenge mit 16 Elementen eine in dieser Anwendung sinnvolle Wahl darstellt, da auch eine weitere Vergrößerung keine Verbesserung mehr ergab.

Vor einer Erörterung der Ergebnisse der Parallelisierung soll im folgenden anhand der Bilder 5.16 und 5.17 die Eignung des — in Kapitel 4 ohne Berücksichtigung der Erfordernisse eines speziellen Einsatzgebietes entwickelten — Kontrollalgorithmus in dem hier vorliegenden Szenario demonstriert werden. Beide Abbildungen zeigen neben den Grauwertbildern (links) die zugehörigen Gruppierungsergebnisse (mitte) und die Ergebnisse der Fahrbahnidentifikation (rechts). Die in Bild 5.16 dargestellten Ergebnisse dürfen dabei als korrekte Interpretationen betrachtet werden, da hier stets die richtige Zuordnung zwischen den Segmentgruppen und den Konzepten für die Fahrbahnmarkierungen vorgenommen wurde. Insbesondere wurde in den oberen beiden Beispielen keine der an der Karosserie eines Fahrzeugs detektierten Segementgruppe(n) zur Instantiierung des fehlenden linken bzw. rechten Randstreifens verwendet.

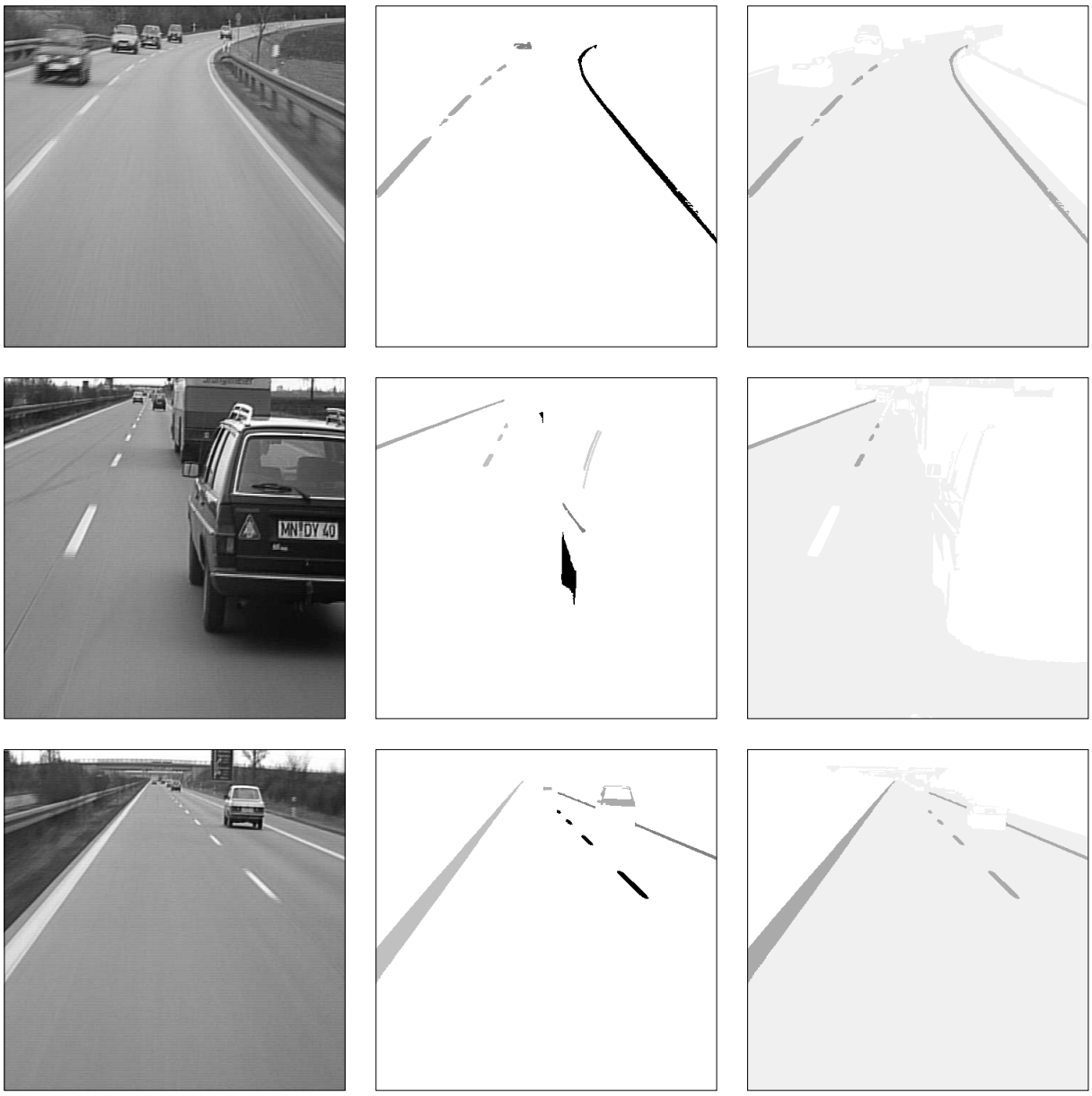


Bild 5.16: Grauwertbilder (links), Segmentgruppen (mitte) und Ergebnisse der Fahrbahnidentifikation (rechts).

In Bild 5.17 wurden dagegen in den oberen beiden Beispielen zwei korrekte Zuordnungen verfehlt: im ersten Fall konnte das — zu kurze — Einzelsegment nicht als linker Randstreifen interpretiert werden, und in dem Beispiel aus der zweiten Sequenz wurde der Mittelstreifen nicht instantiiert. Im unteren Beispiel aus der dritten Sequenz wurden zwar die am Fahrzeugdach oder dem Fahrbahnbankette detektierten Segemente bzw. Segmentgruppen verworfen; eine geringe Ungenauigkeit ergab sich hier jedoch bei der Interpretation

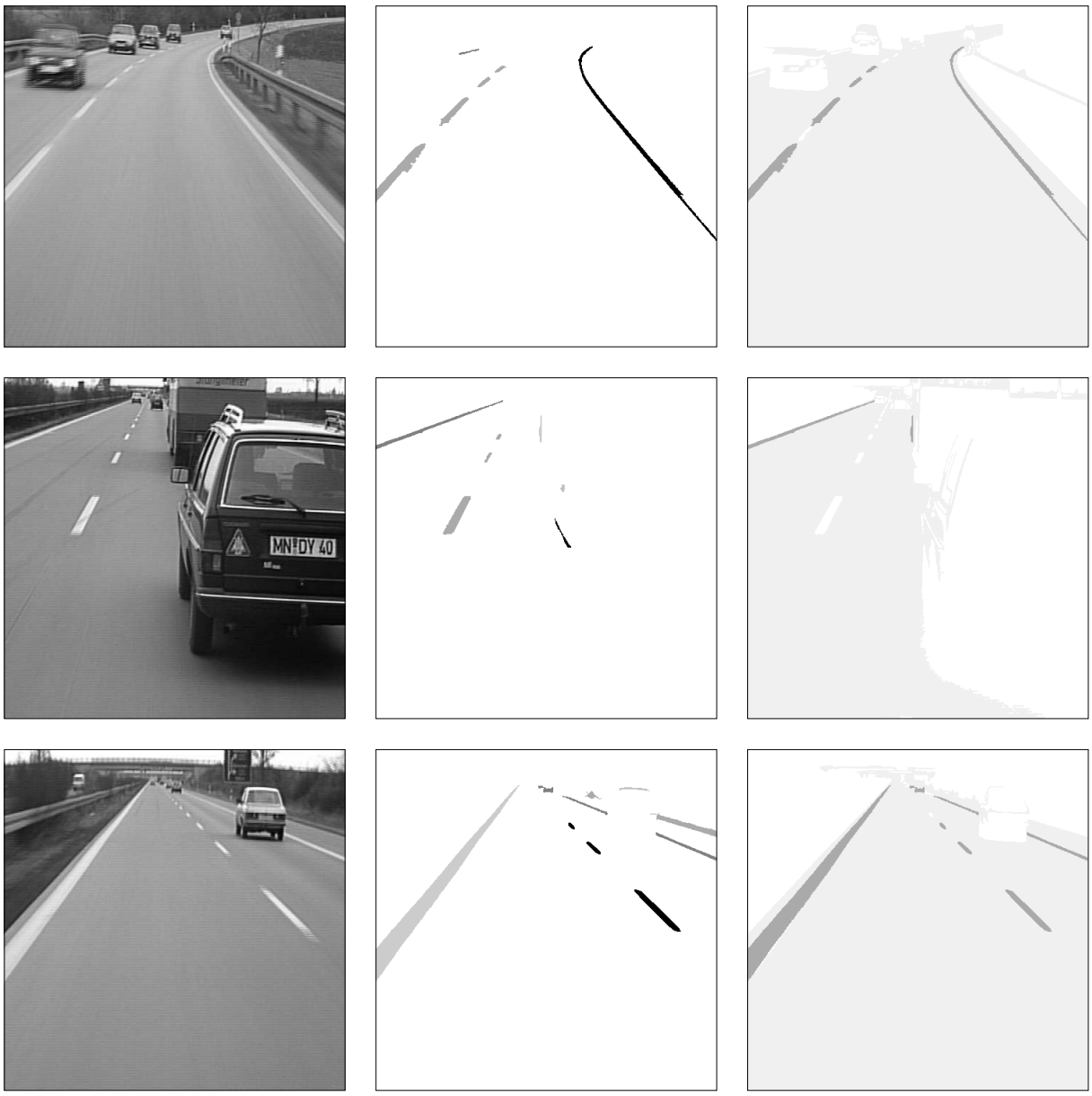


Bild 5.17: Ergebnisse der Fahrbahnidentifikation (Fortsetzung).

des rechten Randstreifens, da die hierzu verwendete Segementgruppe auch Teile der weiter entfernten Fahrzeuge beinhaltet. Zur Verbesserung der Fahrbahnidentifikation erscheint somit vor allem eine Beseitigung von Segmentierungs– und Gruppierungsfehlern als sinnvoll, die auch zu einer besseren Detektion der Ränder des Oberflächensegments (in allen Bildern hellgrau eingezeichnet) führen sollte.

Bild 5.18 illustriert die in Abschnitt 4.1 geforderte Any–Time–Fähigkeit des Kontrollalgorithmus, die durch die Konvergenzgeschwindigkeit ν in den Abbildungen 5.13 – 5.15

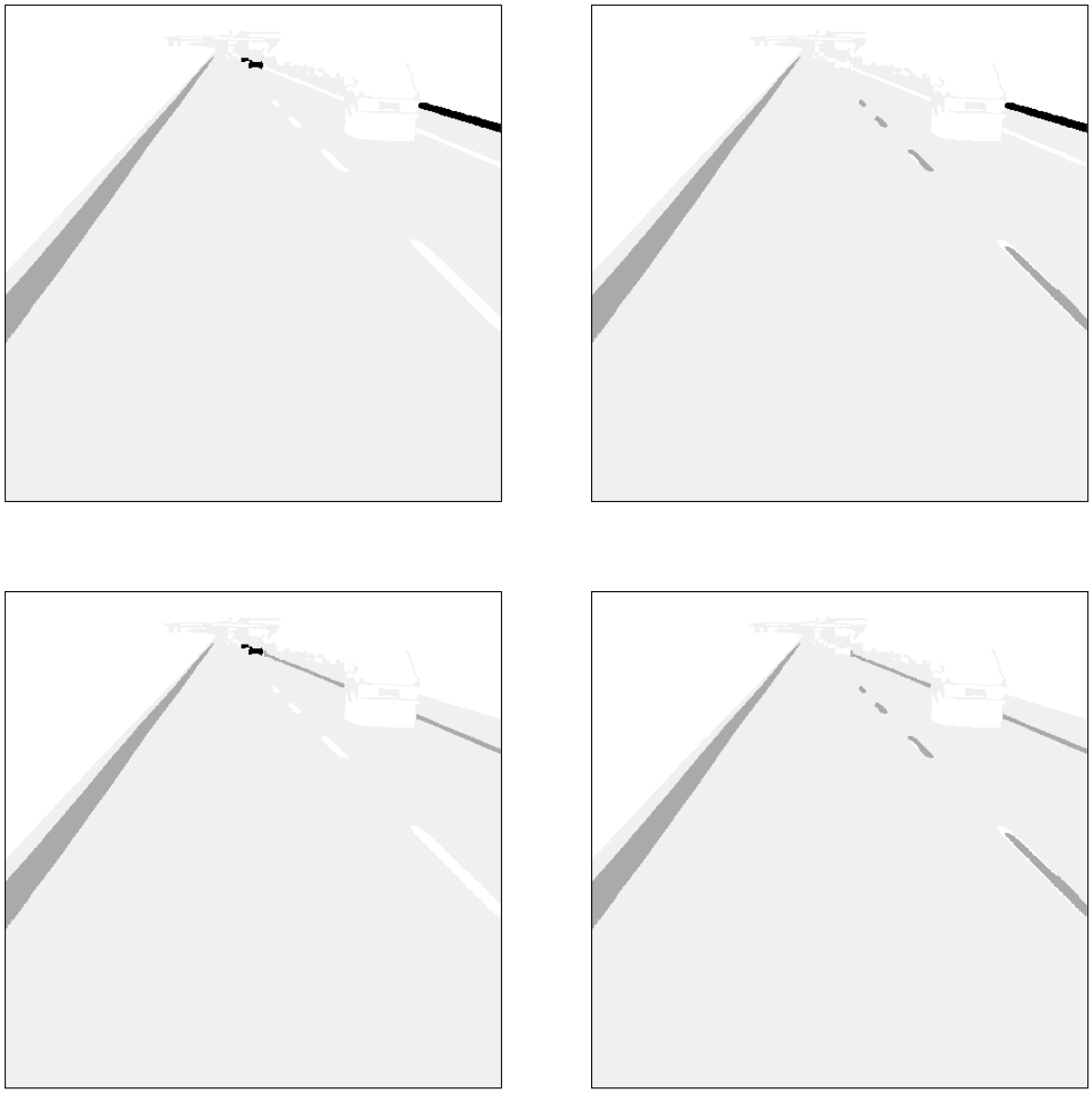


Bild 5.18: Veranschaulichung der iterativen Verbesserung der erzielten Interpretation: Initiale Zuordnung (oben links), Zwischenergebnisse nach 30 (oben rechts) und 85 Iterationen (unten links), und die nach 139 Iterationen erzielte optimale Zuordnung (unten rechts). Weitere Erläuterungen finden sich im Text.

bereits quantitativ beschrieben wurde. Für ein Folgebild der dritten Sequenz sind vier ausgewählte Analysezustande dargestellt, nämlich (von links oben nach rechts unten) die zufällig gewählte initiale Zuordnung, die Zwischenergebnisse nach 30 und 85 Schritten des stochastischen Relaxationsverfahrens sowie das Endergebnis, welches hier nach 139 Iterationen erreicht wurde. Wie auch in den obigen Abbildungen ist das hellgrau eingezeichnete Segment das Segment mit der größten Fläche, welches aufgrund einer entsprechenden

Restriktion in der Attributberechnung als einziger Segmentkandidat für die Fahrbahnoberfläche betrachtet wird. Segmentgruppen, die korrekt interpretiert wurden, sind dunkelgrau dargestellt; falsch zugeordnete Gruppen sind dagegen schwarz markiert. Bei der initialen Zuordnung (oben links) wurde der linke Randstreifen korrekt interpretiert; zur Instantiierung des rechten Randstreifens (Konzept "Re_Randstr") wurde jedoch eine Gruppe verwendet, die aus einem Segment besteht, das am Übergang vom Standstreifen zum Fahrbahnbankett detektiert wurde. Dem Mittelstreifen wurde schließlich eine Segmentgruppe zugeordnet, die zwar vom Fahrbahnsegment eingeschlossen wird — dies ist eine Grundvorraussetzung für potentielle Mittelstreifensegmente —, die aber tatsächlich zwei in der Ferne vorhandenen Fahrzeugen entspricht. Nach 30 Iterationen (oben rechts) wurde diese Fehlinterpretation beseitigt, und die richtige Zuordnung für den Mittelstreifen vorgenommen. Können aufgrund bestehender Zeitschranken für die Verarbeitungsdauer nicht mehr Iterationen ausgeführt werden, so muß dieses Analyseergebnis, bei dem der rechte Randstreifen noch nicht korrekt instantiiert wurde, als suboptimale Interpretation akzeptiert werden. Die nach 85 Iterationen verfügbare Interpretation ist im linken unteren Teil von Bild 5.18 dargestellt. Hier wurde das Konzept "Re_Randstr" durch eine Segmentgruppe mit zwei Segmenten instantiiert, welcher tatsächlich der — durch das Fahrzeug teilweise verdeckten — Markierung des rechten Fahrbahnrandes entsprechen. Obwohl die korrekte Interpretation des Mittelstreifens hier wieder verloren wurde, besitzt die gefundene Lösung geringere Kosten als das nach 30 Iterationen erzielte Ergebnis — möglicherweise, weil die Detektion der Randstreifen bei der Modellierung als besonders wichtig für eine sichere Fahrzeugführung erachtet wurde. Die vollständige und korrekte Interpretation (unten rechts) führt schließlich zum Unterschreiten der Fehlerschranke ε und zur Beendigung der Analyse.

Mit den fünf verschiedenen Annahmekriterien aus Bild 5.13, den zwei genetischen Algorithmen aus Bild 5.14 und den in Abschnitt 4.4 vorgestellten vier Strategien zum Zustandsaustausch lassen sich eine Vielzahl paralleler Kontrollalgorithmen (baukastenartig) realisieren, die zusätzlich durch die Wahl einiger Parameter — beispielsweise der Größe des Austauschintervalls bei der periodisch interagierenden Suche oder der Zustandsmenge bei der genetischen Optimierung — variiert werden können. Anstelle einer vollständigen Darbietung sämtlicher Ergebnisse, die durch die Fülle des Zahlenmaterials vermutlich eher verwirren würde, als zur Klärung der eingangs aufgeworfenen Fragestellungen beizutragen, wollen wir uns hier auf eine exemplarische Darstellung einiger wichtiger Resultate beschränken. Dazu konzentrieren wir uns bei der Untersuchung der Parallelisierungsstrategien zunächst auf die stochastische Relaxation als das in der hier vorliegenden Anwendung günstigste der aus Algorithmus 4.12 abgeleiteten Verfahren. Die dabei gewonnenen Einsichten verwenden wir anschließend zur Parallelisierung des für die optimale Instantiierung entworfenen genetischen Algorithmus (MP-GA), der — vgl. Bild 5.14 — das bessere der beiden untersuchten genetischen Optimierungsverfahren darstellt.

Die in Bild 5.19 zusammengefaßten Ergebnisse für die Parallelisierung von Simulated Annealing und stochastischer Relaxation durch eine multidirektionale Suche begründen diese Vorgehensweise. Hier ist die mittlere Anzahl der zur Bestimmung einer optimalen Lösung

multidirektional	Iterationen					Verweilzeit $[s]$				
	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
Sim. Annealing	(105)	(146)	(157)	(165)	(169)	(105)	(146)	(157)	(165)	(169)
	191	155	120	108	94	17.67	14.40	11.09	10.08	8.71
Stoch. Relaxation	(168)	(173)	(174)	(174)	(174)	(168)	(173)	(174)	(174)	(174)
	91	62	50	40	30	8.12	5.62	4.53	3.61	2.73

Bild 5.19: Anzahl der benötigten Iterationen und Verweilzeit der multidirektionalen Suche bei verschiedenen Annahmekriterien und unterschiedlicher Anzahl an Prozessoren. In Klammern: Anzahl der Bilder, für die bei einer oberen Schranke von 500 Iterationen eine optimale Interpretation erzielt wurde.

benötigten Iterationen und die mittlere *Verweilzeit* bei der Verwendung von $p = 1, \ldots, 5$ Servern dargestellt. Ein Vergleich der Ergebnisse zeigt einerseits keinen unterschiedlichen Trend bei der näheren Betrachtung der Leistungsgrößen — diese werden wir weiter unten diskutieren —, andererseits wird jedoch deutlich, daß eine kurze Analysedauer vorrangig durch ein für die jeweilige Anwendung geeignetes Optimierungsverfahren erzielt wird. So ist selbst bei einer Verwendung von $p = 5$ Servern sowohl die Anzahl der benötigten Iterationen als auch die Verweilzeit des Simulated Annealing noch größer als für die sequentielle stochastische Relaxation. Neben der Geschwindigkeitssteigerung zeigt sich jedoch bereits hier die schon angesprochene geringere Empfindlichkeit gegenüber lokalen Minima der Kostenfunktion als ein weiterer Vorteil der Parallelisierung. Dies wird — vergleiche auch Bild 5.20 — insbesondere bei der multidirektionalen Suche deutlich; hier werden bei einer oberen Schranke von 500 Iterationsschritten vom sequentiellen Algorithmus 168 (105) Bilder optimal interpretiert, während bei $p = 5$ Servern alle (bzw. 169) der durchgeführten 174 Experimente mit einer kostenminimalen Interpretation beendet werden.

Bild 5.20 faßt die Untersuchung der unterschiedlichen Parallelisierungsstrategien anhand des stochastischen Relaxationsverfahrens zusammen. Im linken Teil der Tabelle ist die mittlere Anzahl der benötigten Iterationen bei der Verwendung von $p = 1, \ldots, 5$ Servern angegeben, und der rechte Teil beinhaltet die mittlere Iterationsdauer. Letztere umfaßt die Dauer zwischen dem Verbindungsaufbau zwischen Client und Server(n) und dem Vorliegen des Endzustandes am Client, und setzt sich aus der — je nach Verfahren unterschiedlichen — Zeit für die Kommunikation und die wiederholte Instantiierung des Attributflußgraphen zusammen. Die tabellarische Anordnung der Strategien (multidirektional, dynamisch, k–

periodisch und unidirektional) gibt den wachsenden Kommunikationsaufwand wieder. Dieser ist im Falle der dynamisch interagierenden Suche aufgrund der Abhängigkeit von der verwendeten Kostenfunktion variabel, und die hier vorgenommene Einordnung des Verfahrens erfolgt gemäß der mittleren Anzahl der beobachteten Austauschschritte.

Betrachten wir zunächst die (in Klammern angegebene) Häufigkeit, mit der eine optimale Lösung ermittelt wird, und die Anzahl der dazu benötigten Iterationen, so zeigt sich einerseits eine größere Robustheit der multidirektionalen Suche — eine kostenminimale Lösung wird hier bereits bei Verwendung von $p = 3$ Servern immer gefunden —; andererseits ist festzuhalten, daß die Wirksamkeit der parallelen Optimierung von einem vermehrten Zustandsaustausch deutlich profitiert.

Die im linken Teil von Bild 5.21 zusammengestellten Leistungsgrößen (Speedup s_p, Effizienz e_p, Wirtschaftlichkeit w_p), die zu einer rascheren Orientierung in Bild 5.22 nochmals graphisch dargestellt sind, quantifizieren diesen Zusammenhang. Hier ist ein mit zunehmender Anzahl an Austauschschritten wachsender Speedup bezüglich der benötigten Iterationen festzustellen, und insbesondere bei $p = 2$ oder $p = 3$ Servern zeigt sich eine deutliche Überlegenheit der unidirektionalen Suche, welche für $p = 2$ die größte Effizienz ($e_p \approx 0.90$) besitzt. Die Hinzunahme weiterer Server ($p = 4, 5$) führt zu einer Erhöhung des Speedups bei allen Verfahren und zu einer Verringerung der Unterschiede; eine beginnende Sättigung, die sich in einer sinkenden Wirtschaftlichkeit äußert, zeichnet sich jedoch allenfalls für die unidirektionale Suche ab.

	Iterationen					Verweilzeit $[s]$				
Stoch. Relaxation	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
multidirektional	(168)	(173)	(174)	(174)	(174)	(168)	(173)	(174)	(174)	(174)
	91	62	50	40	30	8.12	5.62	4.53	3.62	2.73
dynamisch	(168)	(170)	(171)	(171)	(172)	(168)	(170)	(171)	(171)	(172)
	91	56	43	34	28	8.83	5.51	4.34	3.64	3.12
4-periodisch	(168)	(171)	(172)	(172)	(173)	(168)	(171)	(172)	(172)	(173)
	91	55	40	32	26	8.87	5.47	4.06	3.31	2.77
2-periodisch	(168)	(170)	(171)	(171)	(172)	(168)	(170)	(171)	(171)	(172)
	91	54	41	31	25	8.56	5.29	4.16	3.41	2.83
unidirektional	(168)	(168)	(168)	(170)	(171)	(168)	(168)	(168)	(170)	(171)
	91	51	35	30	26	8.83	4.96	3.46	3.03	2.71

Bild 5.20: Vergleich der Strategien zum Zustandsaustausch anhand der parallelen stochastischen Relaxation. In Klammern: Anzahl der Bilder, für die bei einer oberen Schranke von 500 Iterationen eine optimale Interpretation erzielt wurde.

Dies wird in stärkerem Maße bei der Betrachtung der Verweilzeiten (vgl. Bild 5.20) und der daraus ermittelten Leistungsgrößen im rechten Teil von Bild 5.21 bzw. Bild 5.22 deutlich: Für die "austauscharmen" Verfahren ist bei der Verwendung von bis zu $p = 5$ Servern noch keine obere Grenze für die Wirtschaftlichkeit abzusehen, und insbesondere im Falle der multidirektionalen Suche sind noch deutliche Anstiege zu verzeichnen. Dagegen nimmt die Wirtschaftlichkeit w_p für die unidirektionale Suche bereits bei $p = 3$ ein Maximum an, obwohl der Speedup auch für $p = 4$ und $p = 5$ noch ansteigt, und bei der Verwendung von fünf Servern mit $s_5 = 3.26$ der größte Speedup aller Versuchsreihen erzielt wird. — im Vergleich zu den aus der Iterationsanzahl bestimmten Leistungsgrößen — geringeren Unterschiede zwischen den Verfahren. Dies unterstreicht den wachsenden Einfluß der Kommunikationsdauer, und die nur noch geringen Differenzen zwischen den Verweilzeiten von multidirektionaler und unidirektionaler Suche bei der Verwendung von $p = 5$ Servern lassen schließlich erwarten, daß bei der Hinzunahme weiterer Rechner auf einen Zustandsaustausch verzichtet werden kann. Zur weiteren Absicherung des hier angedeuteten Sachverhaltes sind aber — wie auch zur genaueren Analyse der dynamisch und periodisch interagierenden Suche — weitere Untersuchungen mit mehr identischen Workstations erforderlich, die auch anhand anderer Anwendungsszenarien geführt werden sollten.

Stoch. Relaxation		bzgl. der Iterationsanzahl					bzgl. der Verweilzeit				
		$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
multidirektional	s_p	1.00	1.48	1.82	2.28	3.03	1.00	1.44	1.79	2.24	2.97
	e_p	1.00	0.74	0.61	0.57	0.61	1.00	0.72	0.60	0.56	0.59
	w_p	1.00	1.08	1.10	1.29	1.84	1.00	1.04	1.07	1.26	1.77
dynamisch	s_p	1.00	1.63	2.12	2.68	3.25	1.00	1.60	2.03	2.43	2.83
	e_p	1.00	0.81	0.71	0.67	0.65	1.00	0.80	0.68	0.61	0.57
	w_p	1.00	1.32	1.49	1.79	2.11	1.00	1.28	1.38	1.47	1.60
4-periodisch	s_p	1.00	1.65	2.28	2.84	3.50	1.00	1.63	2.18	2.68	3.20
	e_p	1.00	0.83	0.76	0.71	0.70	1.00	0.81	0.73	0.67	0.64
	w_p	1.00	1.37	1.73	2.02	2.45	1.00	1.31	1.59	1.80	2.05
2-periodisch	s_p	1.00	1.69	2.22	2.94	3.64	1.00	1.62	2.06	2.51	3.02
	e_p	1.00	0.84	0.74	0.73	0.73	1.00	0.81	0.69	0.63	0.60
	w_p	1.00	1.42	1.64	2.15	2.65	1.00	1.31	1.41	1.58	1.83
unidirektional	s_p	1.00	1.78	2.60	3.03	3.50	1.00	1.78	2.55	2.91	3.26
	e_p	1.00	0.89	0.87	0.76	0.70	1.00	0.89	0.85	0.73	0.65
	w_p	1.00	1.59	2.25	2.30	2.45	1.00	1.58	2.17	2.12	2.12

Bild 5.21: Speedup, Effizienz und Arbeitspunkt der parallelen stochastischen Relaxation bei unterschiedlichen Strategien zum Zustandsaustausch.

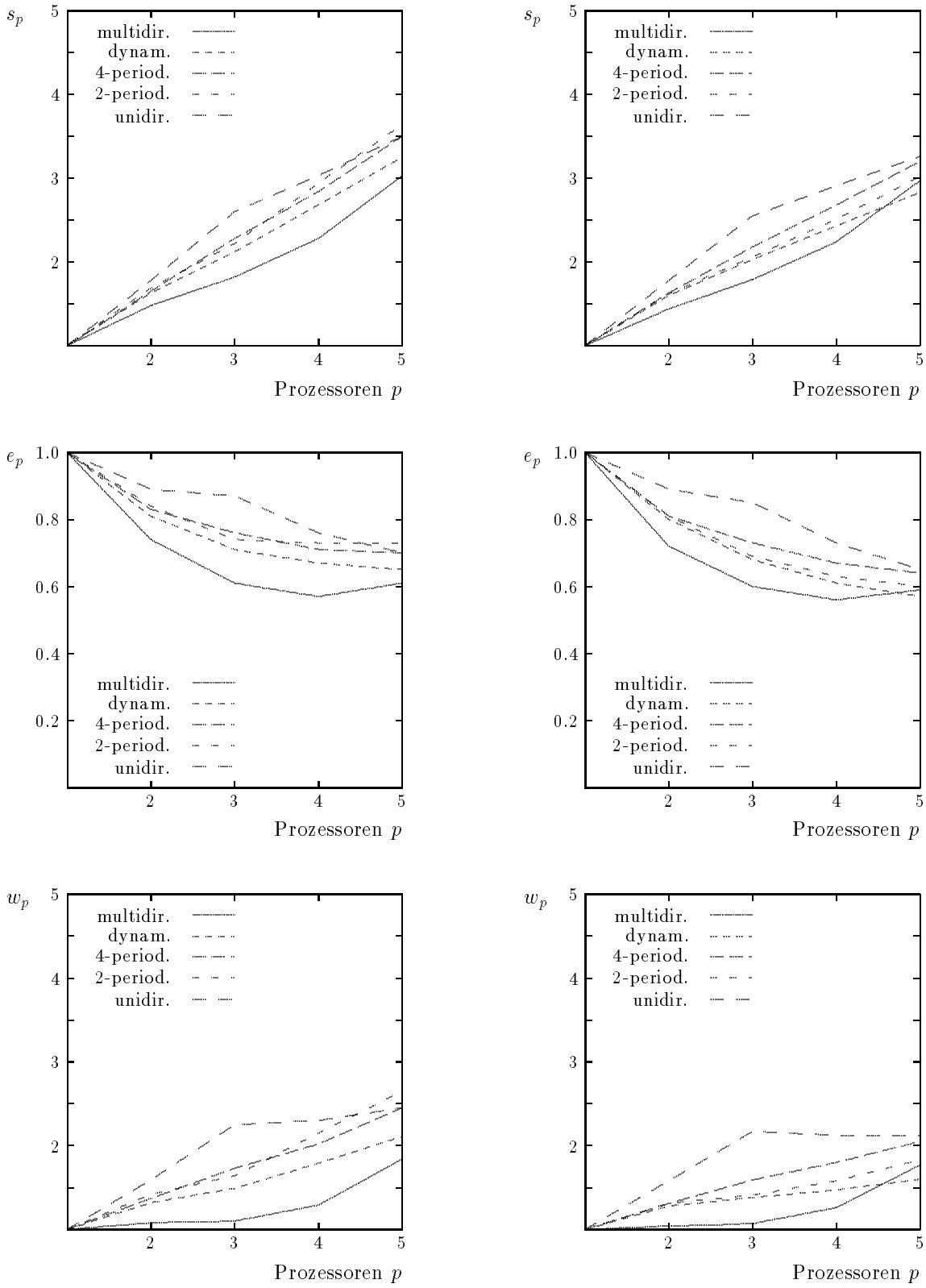


Bild 5.22: Übersicht über Speedup (oben), Effizienz (mitte) und Arbeitspunkt (unten) der parallelen stochastischen Relaxation bei unterschiedlichen Strategien zum Zustandsaustausch. Dargestellt sind die Ergebnisse für die Anzahl ausgeführter Iterationen (links) und die erzielte Verweilzeit (rechts).

Gen. Algorithmus	Iterationen					Verweilzeit $[s]$				
	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
multidirektional	(170)	(174)	(174)	(174)	(174)	(170)	(174)	(174)	(174)	(174)
	48	29	21	18	16	5.04	3.27	2.48	2.26	2.08

Bild 5.23: Anzahl der benötigten Iterationen und Verweilzeit der parallelen genetischen Optimierung.

In Abschnitt 4.4.2 wurde erwähnt, daß alle der hier diskutierten Strategien zum Zustandsaustausch auch bei der Parallelisierung der genetischen Algorithmen eingesetzt werden können, wobei zusätzlich die Frage nach für den Austausch geeigneten Teilmengen der prozessorlokalen Lösungsmengen zu untersuchen ist.

Nachdem die Ergebnisse für die parallele stochastische Relaxation für eine größere Prozessoranzahl den Verzicht auf einen Zustandsaustausch nahelegten, konzentrierten sich die Untersuchungen zur Parallelisierung des genetischen Algorithmus bislang vorrangig auf die Implementierung einer Strategie, die hier ebenfalls als "multidirektional" bezeichnet werden soll, da — genau wie im Falle der unabhängigen stochastischen Relaxation — kein Zustandsaustausch zwischen den Servern stattfindet. Analog zur tabellarischen Darstellung in Bild 5.20 gibt Bild 5.23 die nach Iterationsanzahl und –dauer aufgeschlüsselten Ergebnisse der Parallelisierung wieder, und die zugehörigen Leistungsgrößen sind in Bild 5.24 zusammengestellt. Ein Vergleich mit der multidirektionalen stochastischen Relaxation in Bild 5.20 zeigt zunächst, daß der genetische Mehrpunkt–Algorithmus ungefähr die halbe Anzahl an Iterationen benötigt, um eine optimale Lösung zu ermitteln. Der etwas geringere Geschwindigkeitsvorteil des Verfahrens — er beträgt im Mittel circa 40 Prozent — ist mit der aufwendigeren Initialisierungsphase zu erklären, in der die Zustandsmenge erzeugt wird, wozu die wiederholte Instantiierung des Attributflußgraphen erforderlich ist.

Gen. Algorithmus		bzgl. der Iterationsanzahl					bzgl. der Verweilzeit $[s]$				
		$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
multidirektional	s_p	1.00	1.66	2.29	2.67	3.00	1.00	1.54	2.03	2.23	2.42
	e_p	1.00	0.83	0.76	0.67	0.60	1.00	0.77	0.68	0.56	0.48
	w_p	1.00	1.37	1.74	1.78	1.80	1.00	1.19	1.38	1.24	1.17

Bild 5.24: Speedup, Effizienz und Arbeitspunkt der parallelen genetischen Optimierung (MPGA). In Klammern: Anzahl der Bilder, für die eine optimale Interpretation ermittelt wurde.

Aufgrund der geringen Iterationsanzahl werden jedoch durchweg kürzere Verweilzeiten erzielt, die letztendlich den Einsatz der parallelen genetischen Optimierung bei einer geringen Serveranzahl nahelegen. Der durch die längere Initialisierung gegebene größere sequentielle Programmanteil führt schließlich dazu, daß größere Speedupverluste entstehen als im Fall der multidirektionalen stochastischen Relaxation und eine maximale Wirtschaftlichkeit bereits bei $p = 3$ Servern erreicht wird. Betrachten wir den aus der Anzahl durchgeführter Iterationen berechneten Speedup $s_{p,iter}$ im linken Teil der Bilder 5.20 und 5.24 als den — über sämtliche Experimente gemittelten — idealen Geschwindigkeitsgewinn, und den aus der Verweilzeit bestimmten tatsächlichen Speedup $s_{p,time}$ (rechter Teil der Bilder 5.20 und 5.24), so stellt der Quotient

$$l_p = \frac{s_{p,time}}{s_{p,iter}} \tag{5.3}$$

ein Maß für den Speedupverlust bei der Verwendung von p Servern dar. Die für alle hier diskutierten Verfahren in Bild 5.25 zusammengestellten Verluste sind insbesondere für die multidirektionale und die unidirektionale Suche als gering einzustufen, und dürfen unter Berücksichtigung der verwendeten Ressourcen — es wird lediglich eine Public–Domain–Softwarebibliothek zur Interprozeßkommunikation, jedoch keine spezielle Hardware eingesetzt — auch für die anderen Verfahren als zufriedenstellend bezeichnet werden. Die Ergebnisse sprechen hier einerseits sicherlich für die prinzipielle Eignung der in Abschnitt 4.3.2 vorgenommenen Definition des Analysezustandes (Gl. (4.5)); die relativ großen Verluste bei dynamisch und periodisch interagierender Suche legen andererseits jedoch auch die Untersuchung weiterer Verbindungsstrukturen nahe, die den für diese Verfahren unnötigen Engpaß eines zentralen Client vermeiden. Bevor wir im anschließenden Kapitel auf zusätzlich denkbare Erweiterungen des iterativ–optimierenden Kontrollalgorithmus einge-

	Speedupverluste (Gl. (5.3))			
Stoch. Relaxation	$p = 2$	$p = 3$	$p = 4$	$p = 5$
multidirektional	0.97	0.98	0.98	0.98
dynamisch	0.98	0.96	0.91	0.87
4-periodsch	0.99	0.96	0.94	0.91
2-periodisch	0.96	0.93	0.85	0.83
unidirektional	1.0	0.98	0.96	0.93
Gen. Algorithmus	$p = 2$	$p = 3$	$p = 4$	$p = 5$
multidirektional	0.93	0.89	0.84	0.81

Bild 5.25: Speedupverluste der untersuchten parallelen Optimierungsverfahren.

hen, faßt der folgende Abschnitt die wichtigsten Ergebnisse der hier vorgestellten Pilotanwendung nochmals zusammen.

5.4 Zusammenfassung

Im vorliegenden Kapitel wurde die experimentelle Überprüfung des in Kapitel 4 vorgestellten iterativ–optimierenden Kontrollalgorithmus und der dort entwickelten Konzepte zur Parallelverarbeitung beschrieben. Die Untersuchungen wurden anhand einer Analyseaufgabe aus dem Bereich der Interpretation von Bildfolgen natürlicher Straßenverkehrsszenen geführt; dazu standen drei segmentierte Sequenzen mit je 29 Grauwertbildern der Größe 512×512 zur Verfügung, die im Abstand von 40 Millisekunden mit einer monokularen Kamera aus einem fahrenden Fahrzeug aufgenommen wurden.

Die Wissensbasis besitzt 13 Konzepte, welche das zur Identifikation der Fahrbahn und deren Markierung benötigte problemabhängige Wissen aufnehmen. Nach der Aufspaltung einiger komplexer Attribute umfaßt der aus der Analysevorbereitung resultierende Attributflußgraph 119 Knoten, von denen elf Knoten die Verbindung zur initialen symbolischen Beschreibung herstellen. Letztere besteht aus den Ergebnissen einer anfänglichen Regionensegmentierung und einer richtungsorientierten Vorauswahl und Gruppierung der Segmente in Segmentgruppen, welche die Kandidaten für die unterschiedlichen Fahrbahnmarkierungen (Randstreifen, Mittelstreifen) darstellen. Aus der Größe des Attributflußgraphen und der mittleren Anzahl erzeugter Segmentgruppen resultiert ein kombinatorisches Optimierungsproblem mit circa $1.7 \cdot 10^6$ Zuständen.

Die Untersuchungen zur Parallelisierung der datengetriebenen Instantiierung des Attributflußgraphen wurden unter Verwendung eines Programmpaketes zur Leistungsbewertung paralleler Programme mit Hilfe stochastischer Graphmodelle durchgeführt. Um die in Abschnitt 1.3 eingeführten Leistungsgrößen für eine beliebige Prozessoranzahl zu berechnen, wurde die eingesetzte Zustandsraumanalyse zunächst um einen genetischen Algorithmus zur Bestimmung einer optimalen Zuordnung zwischen den zur Verfügung stehenden Prozessorknoten und den Knoten eines Aufgabengraphen erweitert. Die anschließend durchgeführten Experimente bestätigten sowohl die Notwendigkeit des in Abschnitt 4.3 vorgeschlagenen iterativen Vorgehens als auch die feingranulare Auflösung der Wissensbasis durch den Attributflußgraphen. Die im Rahmen der vorliegenden Arbeit eingesetzten und weiterentwickelten Simulationswerkzeuge erwiesen sich dabei als wichtige Hilfsmittel zur Bestimmung einer der Problemstruktur angemessenen Prozessorzahl.

Die Evaluierung der kombinatorischen Optimierungsverfahren und ihrer Parallelisierung bildete den zweiten Schwerpunkt der durchgeführten Experimente. Bei der Untersuchung der in Abschnitt 4.3.2 vorgestellten Annahmekriterien zeigten sich die auch für andere Optimierungsaufgaben berichteten Konvergenzeigenschaften von Simulated Anne-

aling, Schwellwertakzeptanz und Sintflutalgorithmus; in der hier betrachteten Anwendung empfahl sich jedoch (überraschenderweise) die stochastische Relaxation als Verfahren mit einer noch größeren Konvergenzgeschwindigkeit. Eine weitere Verringerung der Iterationsanzahl gelang mit Hilfe der genetischen Optimierungsverfahren, bei denen sich die Wahrscheinlichkeit für das Erreichen einer optimalen Lösung im Vergleich zur stochastischen Relaxation mehr als verdoppelt, wenn nur wenige ($n \leq 50$) Iterationen ausgeführt werden können.

Die Parallelisierung der Verfahren erfolgte in einem lokalen Netzwerk aus bis zu sechs identischen Workstations. Unter Verwendung der PVM–Unterprogrammbibliothek wurde ein Client/Server–Modell implementiert und die verschiedenen Strategien zum Zustandsaustausch untersucht. Anhand der stochastischen Relaxation wurde gezeigt, daß die Wirksamkeit der Suche von einem vermehrten Zustandsaustausch profitiert, und eine effiziente Parallelisierung des entwickelten Kontrollalgorithmus möglich ist. Mit der unidirektionalen Suche wurde bei der Verwendung von zwei Servern die größte Effizienz erzielt ($e_p = 0.89$) und für fünf Server lieferte das Verfahren mit $s_p = 3.26$ den größten Speedup aller Versuchsreihen. Im Gegensatz zu den anderen betrachteten Verfahren wurde hierbei jedoch bereits für drei Server eine maximale Wirtschaftlichkeit erreicht.

Bei der Parallelisierung des genetischen Algorithmus wurde bislang lediglich ein Zustandsaustausch nach dem Prinzip der multidirektionalen Suche untersucht. Aufgrund der geringen Iterationsanzahl lieferte dieses Verfahren in allen Experimenten die kürzesten Verweilzeiten; der durch die Initialisierung der Zustandsmenge bedingte größere sequentielle Programmanteil führte jedoch zu geringeren Speedups ($s_p = 2.42$ bei $p = 5$ Servern).

Kapitel 6

Schlußbemerkungen

Ausgehend von der Beobachtung, daß bislang zwar zahlreiche parallele Algorithmen für die ikonische Bildverarbeitung existieren, jedoch nur wenige Ansätze zur parallelen symbolischen Bildverarbeitung zu finden sind, wurde in den zurückliegenden Kapiteln die Entwicklung eines parallelen Kontrollalgorithmus für ein semantisches Netzwerksystem zur wissensbasierten Analyse komplexer Muster beschrieben. Zum Abschluß sollen nun einige ergänzende und weiterführende Untersuchungen diskutiert werden, und anschließend die wichtigsten Aspekte der Arbeit nochmals zusammengefaßt werden.

6.1 Ausblick

Die experimentelle Auswertung des iterativ–optimierenden Kontrollalgorithmus anhand des in Abschnitt 5.1 beschriebenen Anwendungsszenarios liefert einerseits wichtige Ergebnisse für die Parallelisierung der datengetriebenen Instantiierung und zeigt andererseits die gute Parallelisierbarkeit der übergeordneten kombinatorischen Optimierungsverfahren. Die parallele Implementierung der Instantiierungsphase und eine anschließende Evaluierung des dann durchgehend parallelen Algorithmus stellen sicherlich ein vorrangiges Ziel weiterer Arbeiten dar. Entsprechend der hierarchischen Struktur des Kontrollalgorithmus bietet sich hierzu der Einsatz eines ebenso organisierten Multiprozessorsystems an. So könnten in einem pyramidenförmig aufgebauten System (vgl. Bild 1.7, S. 13) die Rechnerknoten der untereren Ebene zur parallelen Instantiierung des Attributflußgraphen verwendet werden, während die übergeordneten Prozessoren zur Durchführung der Optimierung dienen, also beispielsweise die Berechnung von Kostenfunktion und Annahmekriterium sowie die Verwaltung und Veränderung von Analysezuständen (oder Zustandsmengen) vornehmen.

Eine Verkürzung der Instantiierungsdauer und die Verringerung der Anzahl benötigter Iterationen können neben der Parallelisierung zu einer Verbesserung der Echtzeitfähigkeit des Kontrollalgorithmus beitragen. Zur Senkung der "Taktfrequenz" der Optimierung

bietet es sich an, in einem Iterationsschritt nicht den ganzen Attributflußgraphen neu zu instantiieren, sondern lediglich die Teile des Graphen, die von dem bei der Zustandsänderung vorgeschlagenen Hypothesenaustausch betroffen sind. Im Falle des Metropolis–Algorithmus wird genau eine Zuordnung zwischen einem initialen Attributflußknoten und einem Segment geändert, oder eine neue Modalitätsmenge gewählt, bezüglich derer ein Konzeptknoten zu bewerten ist. Eine Neuinstantiierung ist hier also nicht für den kompletten Attributflußgraphen notwendig, sondern lediglich für sämtliche (nicht nur die direkten) Nachfolger des ausgewählten Knotens.

Die Verwendung problemspezifischer Kostenfunktionen und Zustandsübergänge sind vielversprechende Möglichkeiten zur Reduzierung der Anzahl auszuführender Iterationen. In der vorgestellten Analyseaufgabe aus dem Bereich der Bildfolgeninterpretation bietet es sich an, die Kostenfunktion so zu wählen, daß die Kontinuität der betrachteten Szene ausgenutzt wird. Nach der Interpretation eines Folgebildes könnten dazu bei der Analyse des nächsten Bildes beispielsweise nur noch die Abweichungen der Attributwerte herangezogen werden. Die systematische Variation des Zustandsvektors stellt neben einer problemabhängige Einschränkung der Nachbarschaft eines Analysezustands eine weitere Möglichkeit zur Verringerung der Iterationsanzahl dar, die insbesondere aufgrund der Diskretheit des vorliegenden Optimierungsproblems erfolgversprechend sein dürfte.

Die zuletzt diskutierten Ergänzungen erfordern im wesentlichen den Austausch der vom Benutzer zu definierenden Methoden zur Kostenberechnung und zur Zustandsgenerierung und sind daher mit geringem Aufwand zu realisieren. Zusätzliche Eingriffe in die Analysevorbereitung sind erforderlich, um eine Behandlung der bislang unberücksichtigten Elemente der ERNEST–Netzwerksprache wie zum Beispiel die Einträge zur lokalen Steuerung der Analyse (Identifikationen, vgl. S. 35) oder zur Darstellung referentieller Bezüge (Referenzkanten, vgl. S. 28) zu ermöglichen. Eine Integration dieser Netzwerkelemente wird gegenwärtig angestrebt, um den entwickelten Kontrollalgorithmus zur Steuerung der linguistischen Analyse im Spracherkennungs– und Dialogsystem EVAR einzusetzen. Eine besondere Herausforderung ergibt sich hierbei aus der Größe der Wissensbasis, die in einen Attributflußgraphen mit ca. 30.000 Knoten transformiert wird, und somit ebenfalls eine starke Motivation für die oben skizzierten Arbeitspunkte bildet.

Weiterführende Untersuchungen zur Parallelisierung sollten sich schließlich mit der Behandlung der problemabhängigen Analyseprozeduren befassen. Ansätze zu einer Integration von neuronalen Netzen zur Objekterkennung in den graphbasierten ERNEST–Kontrollalgorithmus finden sich beispielweise in [Kum93a, Mor94]; als aussichtsreiche Alternative zur Instantiierung kompletter Objekte ist jedoch auch der Einsatz statistischer Objektmodelle zu erwägen [Hor94], deren Verwendung ebenfalls eine gute Parallelisierbarkeit erwarten läßt. Der Einsatz dieser Techniken ist jedoch nicht nur ein Schritt zur weiteren Ausnutzung von Parallelisierungsmöglichkeiten, sondern stellt eine natürliche Erweiterung des in der Kontrolle erprobten Prinzips der iterativen Optimierung dar. Hierdurch eröffnet

sich nicht nur die Möglichkeit zu einer schnelleren und robusteren Analyse, sondern auch die Chance zur Vereinfachung und automatischen Akquisition des bislang überwiegend manuell erworbenen prozeduralen Wissens.

6.2 Zusammenfassung

Obwohl bereits handelsübliche Arbeitsplatzrechner dem Menschen bei einer Vielzahl von Aufgaben — wie etwa dem Sortieren großer Datenbestände — an Geschwindigkeit und Zuverlässigkeit weit überlegen sind, können selbst Höchstleistungsrechner die perzeptiven Leistungen des Menschen bislang nur unvollkommen nachbilden. Die massiv–parallele Verarbeitung von Sinneseindrücken im menschlichen Gehirn wird als eine Ursache dafür angesehen, daß der Mensch Perzeptionsaufgaben selbst unter schwierigen Bedingungen nahezu mühelos erledigt, und dient daher insbesondere auch zur Motivation des Einsatzes von Parallelrechnern in der Mustererkennung, welche sich mit der Auswertung und Interpretation von Sinneseindrücken durch digitale Rechenanlagen befaßt. Versprechen Parallelrechner einerseits die Bereitstellung von hohen Rechenleistungen, die auch zur Lösung einer Vielzahl anderer natur– und ingenieurwissenschaftlicher Aufgabenstellungen benötigt werden, so setzt deren effiziente Nutzung gleichzeitig die Entwicklung geeigneter paralleler Algorithmen voraus.

In der vorliegenden Arbeit werden Möglichkeiten zur Parallelverarbeitung in einem semantischen Netzwerksystem für die wissensbasierte Musteranalyse untersucht. Ziel der Analyse ist die automatische Generierung einer symbolischen Beschreibung. Während sich deren möglicher Inhalt aus den individuellen Erfordernissen der jeweiligen Anwendung ergibt, ist die Forderung nach einer bestmöglichen Beschreibung anwendungsunabhängig und verdeutlicht, daß die Analyse komplexer Muster als Optimierungsproblem aufgefaßt werden kann. Zur Lösung dieses Problems wird sowohl beim Verstehen fließend gesprochener Sprache als auch bei der Interpretation von Bildern oder Bildfolgen natürlicher Szenen — dies sind die wohl wichtigsten Beispiele komplexer Muster — von einem geschichteten Verarbeitungsmodell ausgegangen, das eine Transformation der Sensordaten über zunehmend abstraktere Zwischenrepräsentationen bis hin zur gewünschten symbolischen Beschreibung vorsieht.

Neben der Existenz einer sehr großen, im allgemeinen jedoch konstanten (inhaltsunabhängigen) Anzahl kleiner Datenelemente sind die Lokalität und Ortsinvarianz charakteristisch für die signalnahen Verarbeitungsebenen. Als Konsequenz daraus ergeben sich konzeptuell einfache Parallelisierungsmöglichkeiten nach dem Prinzip der Datenverteilung, die eine gleichmäßige Auslastung der Prozessoren und eine effiziente Parallelisierung gestatten. Kennzeichnend für die wissensbasierte Analyse sind dagegen das Vorhandensein einer variablen, vom Signalinhalt und den Ergebnissen der initialen Segmentierung abhängigen

Anzahl zu verarbeitender Objekte und die explizite Repräsentation und Nutzung problemabhängigen Wissens. Eine Parallelisierung muß daher im allgemeinen nach dem Prinzip der Aufgabenverteilung erfolgen und macht es erforderlich, die eingesetzten Inferenzmechanismen auf gleichzeitig ausführbare Teilaufgaben zu untersuchen.

Das Erlanger semantische Netzwerksystem ERNEST bildet den weiteren Rahmen für die durchgeführten Untersuchungen. Das System besitzt einen modularen Aufbau, der allgemeine Überlegungen zur Architektur wissensbasierter Musteranalysesysteme berücksichtigt; die Module zur Wissensrepräsentation und zur Kontrolle des Analyseprozesses werden ausführlich vorgestellt. Konzepte repräsentieren beliebige Objekte oder Begriffe und können eine beliebige Anzahl von Attributen und Relationen besitzen, welche ihre (physikalischen) Eigenschaften und deren Beziehungen näher beschreiben. Beziehungen zwischen Konzepten werden durch Kanten ausgedrückt, die in Modalitätsmengen gruppiert werden können, um unterschiedliche Begriffsvarianten in einem Netzwerkknoten zusammenzufassen. Die wichtigsten Kantentypen sind die Spezialisierungs–, die Bestandteils– und die Konkretisierungskante, welche zum Aufbau einer Vererbungshierarchie, zur Dekomposition komplexer Konzepte und zur Verbindung unterschiedlicher Abstraktionsebenen dienen. Ebenfalls in der Wissensbasis verankert sind Einträge zur lokalen Steuerung der Analyse, die eine effiziente Nutzung des deklarativen Wissens ermöglichen.

Ausprägungen von Konzepten in den Sensordaten und Zwischenergebnisse der Analyse werden durch Instanzen und modifizierte Konzepte repräsentiert. Ziel des Analyseprozesses ist die Erzeugung einer möglichst gut bewerteten Instanz zu einem Zielkonzept. Die Analyse beruht auf sechs Inferenzregeln zur Generierung von Instanzen und modifizierten Konzepten, welche nur von der Netzwerksyntax, nicht aber von den dargestellten Begriffen abhängig sind. Instanzen und modifizierte Konzepte werden zu Analysezuständen zusammengefaßt, die mit Hilfe eines übergeordneten Graphsuchverfahren auf der Basis des A^*–Algorithmus verwaltet werden.

Überlegungen zur Parallelisierung des graphbasierten Kontrollalgorithmus der Systemschale gehen von einer Teilmenge der Netzwerksprache aus, die neben den epistemologisch primitiven Elementen (Konzepte, Kanten, Attribute und Relationen) lediglich die Verwendung von Modalitätsmengen gestattet, und werden auf der Konzept–, der Netzwerk– und der Suchraumebene geführt. Zur parallelen Instantiierung einzelner Konzepte wird der erweiterte Konzeptflußgraph eingeführt, der die Abhängigkeiten sämtlicher Attributberechnungen und Bewertungen von Relationen, Kanten und Instanzen festhält. Bei der Betrachtung der Netzwerkebene muß zwischen der gleichzeitigen Behandlung unterschiedlicher Begriffsvarianten und der parallelen Instantiierung der Konzepte einer Regelprämisse unterschieden werden. Kann erstere auf die simultane Weiterverfolgung alternativer Suchbaumpfade zurückgeführt werden, so führt die parallele Instantiierung von Bestandteilen und Konkretisierungen zu einer starken Vermehrung der zu berücksichtigenden Analysezustände, da die Möglichkeit zur schrittweisen Einschränkung der Wissensbasis aufgrund

von bereits berechneten Zwischenergebnissen entfällt.

Zur Beschleunigung der Graphsuche werden Parallelisierungsansätze nach dem Prinzip der Datenverteilung und der Aufgabenverteilung diskutiert, und letztere wird für den ERNEST–Kontrollalgorithmus unter Verwendung eines Client/Server–Modells zur Interprozeßkommunikation in einem Workstationnetz experimentell untersucht. Der komplexe Aufbau der Suchbaumknoten, der die Übertragung sämtlicher Instanzen und modifizierter Konzepte sowie zusätzlicher Kontrollinformation erfordert, erweist sich als wesentliche Ursache dafür, daß akzeptable Geschwindigkeitsgewinne nur bei wenig informierten Restkostenschätzungen und einer geringen Prozessoranzahl erzielt werden.

Zur Überwindung dieser Probleme wird ein neuer Kontrollalgorithmus vorgeschlagen und realisiert, der durch die iterative Optimierung einer heuristischen Gütefunktion charakterisiert ist. Wichtigste Entwurfsziele sind dabei neben der Eignung für die Parallelverarbeitung die Any–Time–Fähigkeit des Verfahrens und die Unterstützung einer ergonomischen Wissensrepräsentation. Um einerseits die Vorteile der objektzentrierten Darstellung des Wissens durch den verwendeten Netzwerkformalismus zu gewährleisten, andererseits jedoch eine größtmögliche Parallelität bei der Abarbeitung der zur Instantiierung notwendigen Inferenzen zu erreichen, wird zunächst eine automatische Transformation der Wissensbasis in einen feingranularen Datenflußgraphen, den Attributflußgraphen, vorgenommen. Die Knoten dieses Graphen repräsentieren die Attribute, Relationen, Kanten oder Konzepte des Netzwerkes und sind mit den entsprechenden Analyseprozeduren zur Werteberechnung und Bewertung versehen.

Die parallele Instantiierung des Attributflußgraphen erfolgt durch die Aktivierung dieser Prozeduren in der durch die Kanten des Graphen vorgegebenen Reihenfolge; sie beginnt mit den Attributen, die direkt auf die initiale symbolische Beschreibung zugreifen, und endet mit der Bewertung der Knoten, welche die Analyseziele repräsentieren. Die so berechneten Bewertungen werden durch eine heuristische Fehlerfunktion verknüpft und sind allein von der gewählten Zuordnung zwischen Segmentierungsergebnissen und initialen Attributflußknoten sowie den verwendeten Modalitätsmengen abhängig. Ein Analysezustand kann daher als ein Vektor aufgefaßt werden, der die zur Instantiierung verwendeten Zuordnungen und Modalitätsmengen repräsentiert; in Verbindung mit dem iterativen Vorgehen ermöglicht die Definition eine effiziente Parallelisierung, da sich der Informationsaustausch zwischen den Prozessoren auf diesen Vektor beschränken kann.

Zur Minimierung der Fehlerfunktion werden allgemeine kombinatorische Optimierungsverfahren wie beispielsweise verschiedene Varianten des bekannten Simulated Annealing oder genetische Algorithmen eingesetzt; das Ergebnis der Minimierung ist eine Interpretation, welche optimal zu den Segmentierungsergebnissen paßt, und maximal kompatibel mit den in der Wissensbasis abgelegten Vorerwartungen ist.

Neben den restriktiven Zeitanforderungen zahlreicher Anwendungen bildet die — aufgrund der Konvergenzeigenschaften der Verfahren notwendige — große Anzahl auszuführen-

der Iterationen eine wesentliche Motivation für die Parallelisierung der Optimierung. Die hierzu vorgestellten Ansätze sehen die Ausführung des gleichen Algorithmus auf jedem der zur Verfügung stehenden Prozessoren (Rechnerknoten) vor, verwenden jedoch unterschiedliche Strategien zum Austausch von Analysezuständen.

Zum Ende der Arbeit werden der neu entwickelte Kontrollalgorithmus und die vorgeschlagenen Parallelisierungsstrategien in einer Pilotanwendung aus dem Bereich der Interpretation von Bildfolgen natürlicher Straßenverkehrsszenen experimentell ausgewertet. Die zur Parallelisierung der Netzwerkinferenzen vorgenommene feingranulare Auflösung der Wissensbasis durch den Attributflußgraphen und die zur Vermeidung der kombinatorischen Explosion von Zwischenergebnissen vorgeschlagene iterative Instantiierung werden anhand einer Parallelrechnersimulation bestätigt, welche das aus der Leistungsbewertung bekannte Verfahren der Zustandsraumanalyse verwendet. Die im Rahmen der durchgeführten Untersuchungen vorgenommene Kombination der Zustandsraumanalyse mit einem genetischen Algorithmus erlaubt hierbei sowohl die optimale Abbildung der Knoten des Attributflußgraphen auf die zur Verfügung stehenden Prozessoren als auch die Bestimmung einer bezüglich der Wirtschaftlichkeit optimalen Prozessoranzahl. Zur Parallelisierung der übergeordneten kombinatorischen Suchverfahren wird ein aus bis zu sechs Workstations bestehendes lokales Netzwerk und eine Public–Domain–Unterprogammbibliothek zur Interprozeßkommunikation eingesetzt. Für alle der betrachteten Verfahren werden gute Geschwindigkeitssteigerungen erzielt. Während die — auf die Anzahl der ausgeführten Iterationen bezogene — Wirksamkeit der Suche durch die Hinzunahme weiterer Rechner noch zu steigern sein dürfte, zeichnet sich bei den Parallelisierungsansätzen, die einen häufigen Zustandsaustausch oder eine große Initialisierungsdauer erfordern, eine beginnende Sättigung der Wirtschaftlichkeit ab.

Literaturverzeichnis

[Aar85] E. Aarts, P. van Laarhoven: *Statistical Cooling: A general Approach to Combinatorial Optimization Problems*, *Philips Journal of Research and Development*, Bd. 40, Nr. 4, 1985, S. 193–226.

[Aar86a] E. Aarts, F. de Bont, E. Habers, P. van Laarhoven: *Parallel Implementations of the Statistical Cooling Algorithm*, *Integration, the VLSI journal*, Bd. 4, 1986, S. 209–238.

[Aar86b] E. Aarts, F. de Bont, E. Habers, P. van Laarhoven: *A Parallel Statistical Cooling Algorithm*, in B. Monien, G. Vidal-Naquet (Hrsg.): *STACS 86. Proc. of the 3rd Annual Symposium on Theoretical Aspects of Computer Science,* Orsay, France, Bd. 210 von *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1986, S. 87–98.

[Aar87] E. Aarts, P. van Laarhoven: *Simulated Annealing: A pedestrian review of the theory and some applications*, in [Dev87], 1987, S. 179–192.

[Aar89] E. Aarts, J. Korst: *Simulated Annealing and Boltzmann Machines. A Stochastic Approach to Combinatorial Optimization and Neural Computing*, Wiley–Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, 1989.

[Abb94] F. Abbattista, A. Fanelli, M. Pelillo: *An Evolutionary Approach to Vector Quantizer Design*, in S. Impedovo (Hrsg.): *Progress in Image Analysis and Processing III. Proc. of the 7th Int. Conf on Image Analysis and Processing.* Bari, Italy, World Scientific, Singapore, 1994, S. 254–257.

[Ain88] W. A. Ainsworth: *Speech Recognition by Machine*, Peter Peregrinus, Ltd., London, 1988.

[Ank90] C. Ankenbrandt, B. Buckles, F. Petry: *Scene recognition using genetic algorithms with semantic nets*, *Pattern Recognition Letters*, Bd. 11, Nr. 4, 1990, S. 285–293.

[Aze92a] R. Azencott: *Parallel Simulated Annealing: An Overview of Basic Techniques*, in [Aze92c], Kap. 4, 1992, S. 37–46.

[Aze92b] R. Azencott: *Sequential Simulated Annealing: Speed of Convergence and Acceleration Techniques*, in [Aze92c], Kap. 1, 1992, S. 1–10.

[Aze92c] R. Azencott (Hrsg.): *Simulated Annealing. Parallelization Techniques.*, Wiley–Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, 1992.

[Bak93] J. Baker: *Dictation, Directories, and Data Bases: Emerging PC Applications for Large Vocabulary Speech Recognition*, in *EUROSPEECH '93. Proc. of the 3rd European Conf. on Speech Communication and Technology*, Bd. 1, Berlin, 1993, S. 3–10.

[Bal82] D. Ballard, C. Brown: *Computer Vision*, Prentice Hall, Inc., Englewood Cliffs, N.J., 1982.

[Bal93] S. Baluja: *Structure and Performance of Fine-Grained Parallelism in Genetic Search*, in [For93], 1993, S. 155–162.

[Bel91] R. Belew (Hrsg.): *Proc. of the 4th Int. Conf. on Genetic Algorithms.* University of California, San Diego, Calif., Morgan Kaufmann, San Mateo, Calif., 1991.

[Bia93] R. Bianchini, C. Brown: *Parallel Genetic Algorithms on Distributed-Memory Architectures*, Technical Report 436 (Revised Version), Computer Science Department, University of Rochester, May 1993.

[Bic85] L. Bic: *Processing of Semantic Nets on Dataflow Architectures*, *Artificial Intelligence*, Bd. 27, 1985, S. 219–227.

[Bod93] A. Bode, M. Dal Cin (Hrsg.): *Parallel Computer Architectures. Theory, Hardware, Software, Applications*, Bd. 732 von *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1993.

[Boi93] N. Boissin, J. Lutton: *A parallel simulated annealing algorithm*, *Parallel Computing*, Bd. 19, Nr. 8, 1993, S. 859–872.

[Boo89] L. Booker, D. Goldberg, J. Holland: *Classifier Systems and Genetic Algorithms*, *Artificial Intelligence*, Bd. 40, Nr. 1–3, 1989, S. 235–282.

[Bra77] R. Brachman: *What's in a Concept: Structural Foundations for Semantic Networks*, Bolt, Beranek and Newman, Cambridge, Mass., 1977.

[Bra79] R. Brachman: *On the epistemological status of semantic networks*, in N. Findler (Hrsg.): *Associative Networks*, Academic Press, Inc., New York, 1979, S. 3–50.

[Bra84] R. Brachman, H. Levesque: *The tractability of subsumption in frame-based description languages*, in *Proc. of the 4th National Conf. on Artificial Intelligence*, Austin, Tex., 1984, S. 109–113.

[Bra85a] R. Brachman, J. Schmolze: *An Overview of the KL-ONE Knowledge Representation Language*, *Cognitive Science*, Bd. 9, 1985, S. 171–216.

[Bra85b] R. Brachman, V. Gilbert, H. Levesque: *An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of Krypton*, in *Proc. of the 9th Int. Joint Conf. on Artificial Intelligence*, Los Angeles, Calif., 1985, S. 532–539.

[Bro81] R. Brooks: *Symbolic Reasoning about 3-D Models and 2-D Images*, *Artificial Intelligence*, Bd. 17, 1981, S. 285–348.

[Bro89] *Brockhaus Enzyklopädie. Stichwort "Intelligenz"*, 19. Auflage, Bd. 10, F.A. Brockhaus Verlag, Mannheim, 1989.

[Brü90] H. Brünig: *Konzeption und Realisierung einer flexiblen Bildsegmentierung für ein wissensbasiertes Bildanalysesystem*, Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, Erlangen, 1990.

[Buc85] B. Buchanan, E. Shortliffe: *Rule–Based Expert Systems*, Addison–Wesley Publ. Co., Reading, Mass., 1985.

[Bur89] H. Burkhart, R. Millen: *Performance Measurement Tools in a Multiprocessor Environment*, *IEEE Trans. on Computers*, Bd. 38, Nr. 5, 1989, S. 725–737.

[Bur91] H. Burkhardt, Y. Neuvo, J. Simon (Hrsg.): *From Pixels to Features II. Parallelism in Image Processing*, North–Holland, Amsterdam, 1991.

[Car93] R. Cardin, Y. Normandin, E. Millien: *Inter-Word Coarticulation Modeling and MMIE Training for Improved Connected Digit Recognition*, in *Proc. of the 1993 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Bd. II, Minneapolis, Minn., 1993, S. 243–246.

[Cat92] O. Catoni: *Rates of Convergence for Sequential Annealing: A Large Deviation Approach*, in [Aze92c], Kap. 3, 1992, S. 25–35.

[Cha90] V. Chaudhary, J. Aggarwal: *Parallelism in Computer Vision: A Review*, in [Kum90], 1990, S. 271–309.

[Cho91] S. Chou, W. Tsai: *Recognizing Handwritten Chinese Characters by Stroke-Segment Matching Using an Iteration Scheme*, in *Character & Handwriting Recognition. Expanding Frontiers*, Bd. 30 von *World Scientific Series in Computer Science*, World Scientific, Singapore, 1991, S. 175–198.

[Coh87] J. Cohoon, S. Hedge, W. Martin, D. Richards: *Punctuated Equilibria: A Parallel Genetic Algorithm*, in [Gre87], 1987, S. 148–154.

[Coh91] J. Cohoon, W. Martin, D. Richards: *A multi-population genetic algorithm for solving the k-partition problem on hyper-cubes*, in [Bel91], 1991, S. 244–248.

[Cor91] J. Corbin: *The Art of Distributed Applications*, SUN Technical Reference Library, Springer-Verlag, New York, 1991.

[Cyp89] R. Cypher, J. Sanz: *SIMD architectures and algorithms for image processing and computer vision*, *IEEE Trans. on Acoustics, Speech and Signal Processing*, Bd. ASSP-37, Nr. 12, 1989, S. 2158–2174.

[Dau92] P. Dauphin, F. Hartleb, M. Kienow, V. Mertsiotakis, A. Quick: *PEPP: Performance Evaluation of Parallel Programs. Users's Guide – Version 3.1*, 5/92, Lehrstuhl für Informatik 7, Friedrich-Alexander-Univerität Erlangen-Nürnberg, September 1992.

[Dav85] L. Davis: *Job shop scheduling with genetic algorithms*, in *Proc. of the 1st Int. Conf. on Genetic Algorithms and their Applications*, Carnegie-Mellon University, Pittsburgh, Pa., 1985, S. 136–140.

[Dav87] L. Davis (Hrsg.): *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, Pitman and Morgan Kaufmann Publ. Inc., London and San Mateo, Calif., 1987.

[Dav91] L. Davis (Hrsg.): *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[Der87] M. Derthick: *A Connectionist Architecture for Representing and Reasoning about Structured Knowledge*, in *9th Annual Conference of the Cognitive Science Society*, Seattle, Wash., Lawrence Erlbaum Associates, Hillsdale, N.J., 1987, S. 131–142.

[Der90] M. Derthick: *Mundane Reasoning by Parallel Constraint Satisfaction*, Research Notes in Artificial Intelligence, Pitman and Morgan Kaufmann Publ. Inc., London and San Mateo, Calif., 1990.

[Der93] A. Derouault, E. Keppel, S. Fusi, J. Marcadet, E. Janke: *The IBM Speech Server Series and its Applications in Europe*, *Applications of Speech Technology*, 1993.

[Dev82] P. Devijver, J. Kittler: *Statistical Pattern Recognition*, Prentice Hall, Inc., Englewood Cliffs, N.J., 1982.

[Dev87] P. Devijver, J. Kittler (Hrsg.): *Pattern Recognition Theory and Application*, Bd. F30 von *NATO ASI Series*, Springer-Verlag, Berlin, 1987.

[Dix87] V. Dixit, D. Moldovan: *Semantic Network Array Processor and Its Application to Image Understanding*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 9, Nr. 1, 1987, S. 153–160.

[Dud73] R. Duda, P. Hart: *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

[Due90] G. Dueck, T. Scheuer: *Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing*, *Journal of Computational Physics*, Bd. 90, Nr. 1, 1990, S. 161–175.

[Due93a] G. Dueck: *New optimization heuristics: The great deluge algorithm and the record-to-record-travel*, *Journal of Computational Physics*, Bd. 104, Nr. 1, 1993, S. 86–92.

[Due93b] G. Dueck, T. Scheuer, H.-M. Wallmeier: *Toleranzschwelle und Sintflut: neue Ideen zur Optimierung*, *Spektrum der Wissenschaft*, Nr. 3,1993, S. 42–51.

[Eck92] W. Eckert, G. Fink, A. Kießling, R. Kompe, T. Kuhn, F. Kummert, M. Mast, H. Niemann, E. Nöth, R. Prechtel, S. Rieck, G. Sagerer, A. Scheuer, E. Schukat-Talamazzini, B. Seestaedt: *EVAR: Ein sprachverstehendes Dialogsystem*, in G. Görz (Hrsg.): *KONVENS 92*, Informatik aktuell, Springer-Verlag, Berlin, 1992, S. 49–58.

[Eve94] M. Evett, J. Hendler, L. Spector: *Parallel Knowledge Representation on the Connection Machine*, *Journal of Parallel and Distributed Computing*, Bd. 22, Nr. 2, 1994, S. 168–184.

[Fah79] S. Fahlman: *NETL: A system for Representing and Using Real-World Knowledge*, The MIT Press, Cambridge, Mass., 1979.

[Fah83] S. Fahlman, G. Hinton, T. Sejnowski: *Massively Parallel Architectures for AI: NETL, THISTLE, and BOLTZMANN Machines*, in *Proc. of the 3rd National Conf. on Artificial Intelligence*, Washington, D.C., 1983, S. 109–113.

[Fel50] W. Feller: *An Introduction to Probability Theory and Applications, Volume I*, Wiley, New York, 1950.

[Fel85] E. Felten, S. Karlin, S. Otto: *The Traveling Salesman Problem on a Hybercubic MIMD Computer*, in *Proc. of the 14th Int. Conf. on Parallel Processing*, St. Charles, 1985, S. 6–10.

[Fid86] M. Fidelak: *Petri Nets, a Formal Language for Knowledge Representation*, in *Proc. of the 7th Europ. Conf. on Artificial Intelligence*, Brighton, 1986, S. 164–168.

[Fil68] C. Fillmore: *A Case for Case*, in E. Bach, R. Harms (Hrsg.): *Universals in Linguistic Theory*, Holt, Rinehart and Winston, New York, 1968.

[Fin79] N. Findler (Hrsg.): *Associative Networks*, Academic Press, Inc., New York, 1979.

[Fis93a] V. Fischer, H. Niemann: *Parallelism in a Semantic Network for Image Understanding*, in [Bod93], 1993, S. 203–218.

[Fis93b] V. Fischer, H. Niemann, D. Paulus, A. Winzen: *Ein paralleler Kontrollalgorithmus für die wissensbasierte Bildanalyse*, in H. Reichel (Hrsg.): *Informatik – Wirtschaft – Gesellschaft, 23. GI-Jahrestagung, Dresden*, Informatik aktuell, Springer-Verlag, Berlin, 1993, S. 515–519.

[Fla89] H. Flatt, K. Kennedy: *Performance of Parallel Processors, Parallel Computing*, Bd. 12, Nr. 1, 1989, S. 1–20.

[Fly72] M. Flynn: *Some Computer Organizations and their Effectiveness, IEEE Trans. on Computers*, Bd. 21, Nr. 9, 1972, S. 948–960.

[For91] S. Forrest: *Parallelism and Programming in Classifier Systems*, Research Notes in Artificial Intelligence, Pitman and Morgan Kaufmann Publishers, Inc., London and San Mateo, Calif., 1991.

[For93] S. Forrest (Hrsg.): *Proc. of the 5th Int. Conf. on Genetic Algorithms.* University of Illinois, Urbana-Champaign, Morgan Kaufmann, San Mateo, Calif., 1993.

[Fri89] G. Fritsch, W. Henning, H. Hessenauer, R. Klar, C.-U. Linster, C. Oehlrich, P. Schlenk, J. Volkert: *Distributed Shared Memory Multiprocessor Architecture MEMSY for High Performance Parallel Computations, Computer Architecture News*, Bd. 17, Nr. 6, 1989, S. 22–35.

[Fri94] G. Fritsch, W. Henning, M. Peric, M. Schäfer, E. Schreck, H. Früchtl, P. Otto: *Numerische Anwendungen auf Verteilten Rechensystemen*, in [Wed94], 1994, S. 244–288.

[Gar79] M. Garey, D. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completness*, Freeman, San Francisco, Calif., 1979.

[Gas79] J. Gaschnig: *Performance measurement and analysis of certain search algorithms*, CMU-CS-79-124, Computer Science Dept., Carnegie-Mellon-University, Pittsburgh, Pa., 1979.

[Gei91] A. Geist, V. Sunderam: *The PVM System: Supercomputing Level Concurrent Computations on a Heterogeneous Network of Workstations*, in *Proc. of the 6th IEEE Conference on Distributed Memory Computing*, Portland, Oreg., 1991, S. 258–261.

[Gei92] A. Geist, V. Sunderam: *Network-based concurrent computing on the PVM system, Concurrency: Practice & Experience*, Bd. 4, Nr. 4, 1992, S. 293–311.

[Gem84] S. Geman, D. Geman: *Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 6, Nr. 6, November 1984, S. 721–741.

[Gol89] D. Goldberg: *Genetic Algorithms: Search, Optimization and Machine Learning*, Addison–Wesley Publ. Co., Reading, Mass., 1989.

[Gor89] M. Gorges-Schleuter: *ASPARGOS: A asynchronous parallel genetic optimization strategy*, in [Sch89], 1989, S. 422–427.

[Gre87] J. Grefenstette (Hrsg.): *Genetic Algorithms and Their Applications. Proc. of the 2nd Int. Conf. on Genetic Algorithms.* Massachusetts Institute of Technology, Cambridge, Ma., Lawrence Erlbaum Associates, Hillsdale, N.J., 1987.

[Gup87] A. Gupta: *Parallelism in production systems*, Research Notes in Artificial Intelligence, Pitman and Morgan Kaufmann Publishers, Inc., London and San Mateo, Calif., 1987.

[Han78] A. Hanson, E. Riseman: *VISIONS: A Computer System for Interpreting Scenes*, in A. Hanson, E. Riseman (Hrsg.): *Computer Vision Systems*, Academic Press, Inc., New York, 1978, S. 303–333.

[Han80] A. Hanson, E. Riseman: *Processing Cones: A Computational Structure for Scene Analysis*, in S. Tanimoto, A. Klinger (Hrsg.): *Structured Computer Vision*, Academic Press, Inc., New York, 1980.

[Har93a] F. Hartleb: *Graph Models for Performance Evaluation of Parallel Programs*, in [Bod93], 1993, S. 80–87.

[Har93b] F. Hartleb, A. Quick: *Performance Evaluation of Parallel Programs — Modeling and Monitoring with the Tool PEPP*, in B. Walke, O. Spaniol (Hrsg.): *Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen. 7. ITG/GI-Fachtagung, Aachen*, Informatik aktuell, Springer-Verlag, Berlin, 1993, S. 51–63.

[Hen88] J. Hendler: *Integrating Marker-Passing and Problem-Solving: A Spreading Activation Approach to Improved Choice in Planning*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1988.

[Hin86] G. Hinton, T. Sejnowski: *Learning and Relearning in Boltzmann Machines*, in D. Rumelhart, J. McClelland (Hrsg.): *Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, Cambridge, Mass., 1986, S. 282–317.

[Hof84] F. Hofmann: *Betriebssysteme: Grundkonzepte und Modellvorstellungen*, B.G. Teubner Verlag, Stuttgart, 1984.

[Hof93] F. Hofmann, M. Dal Cin, A. Grygier, H. Hessenauer, U. Hildebrand, C.-U. Linster, T. Thiel, S. Turowski: *MEMSY – A Modular Expandable Multiprocessor System*, in [Bod93], 1993, S. 15–30.

[Hol75] J. Holland: *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich., 1975.

[Höp90] F. Höpfl: *Kombinatorische Suchverfahren und ihre Parallelisierung auf Multiprozessorsystemen*, Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, 1990.

[Hor94] J. Hornegger, H. Niemann: *A Bayesian Approach to Learn and Classify 3D Objects from Intensity Images*, in *Proc. of the 12th Int. Conf on Pattern Recognition. Jerusalem, Oktober 1994*, IEEE Computer Society Press, 1994, S. 557–559.

[Hou94] E. Hou, N. Asari, H. Ren: *A Genetic Algorithm for Multiprocessor Scheduling, IEEE Trans. on Parallel and Distributed Systems*, Bd. 5, Nr. 2, 1994, S. 113–120.

[Hua90] X. Huang, Y. Ariki, M. Jack: *Hidden Markov Models for Speech Recognition*, Information Technology Series, Edinburgh University Press, Edinburgh, 1990.

[Hua93] P. Hua: *Parallelisierung des Kontrollalgorithmus in* ERNEST, Studienarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, Lehrstuhl für Mustererkennung (Informatik 5), Erlangen, 1993.

[Huy80] N. Huyn, R. Dechter, J. Pearl: *Probabilistic Analysis of the Complexity of A**, *Artificial Intelligence*, Bd. 15, 1980, S. 241–254.

[Jäh93] B. Jähne: *Digital Image Processing.* 2nd Edition, Springer-Verlag, Berlin, 1993.

[Joh86] D. Johnson, C. Aragon, L. McGeoch, C. Schevon: *Optimization by Simulated Annealing: an Experimental Evaluation*, in *List of Abstracts, Workshop on Statistical Physics in Engineering and Biology. Revised Version.*, Yorktown Heights, N.J., 1986.

[Kac86] T. Kaczmarek, R. Bates, G. Robins: *Recent Developments in NIKL*, in *Proc. of the Nat. Conf. on Artificial Intelligence*, Philadelphia, Pa., 1986, S. 978–985.

[Kim90] J. Kim, D. Moldovan: *Parallel Knowledge Classification on SNAP*, in *Proc. of the 19th Int. Conf. on Parallel Processing*, Bd. 1, Urbana-Champaign, Ill., 1990, S. 482–488.

[Kir82] S. Kirkpatrick, C. Gelatt Jr., M. Vecchi: *Optimization by Simulated Annealing*, IBM Research Report RC 9355, 1982.

[Kir83] S. Kirkpatrick, C. Gelatt Jr., M. Vecchi: *Optimization by Simulated Annealing, Science*, Bd. 220, 1983, S. 671–680.

[Kol90] P. Koller: *Gewinnung von dichten Tiefenbildern aus Farbstereoaufnahmen mit Simulated Annealing*, in T. Härder, H. Wedekind, G. Zimmermann (Hrsg.): *Entwurf und Betrieb verteilter Systeme. Fachtagung der Sonderforschungsbereiche 124 und 182*, Bd. 264 von *Informatik-Fachberichte*, Springer-Verlag, Berlin, 1990, S. 248–259.

[Kra87] S. Kravitz, R. Rutenbar: *Placement by Simulated Annealing on a Multiprocessor, IEEE Trans. on Computer-Aided Design*, Bd. 6, Nr. 4, 1987, S. 534–549.

[Kuh94] T. Kuhn: *Die Erkennungsphase in einem Dialogsystem*, Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, Erlangen, 1994.

[Kum90] V. Kumar, P. Gopalakrishnan, L. Kumar (Hrsg.): *Parallel Algorithms for Machine Intelligence and Vision*, Springer-Verlag, New York, 1990.

[Kum92a] F. Kummert: *Flexible Steuerung eines sprachverstehenden Systems mit homogener Wissensbasis*, Bd. 12 von *Dissertationen zur Künstlichen Intelligenz*, Infix, Sankt Augustin, 1992.

[Kum92b] F. Kummert, G. Sagerer, H. Niemann: *A Problem-Independent Control Algorithm for Image Understanding*, in *Proc. of the 11th Int. Conf. on Pattern Recognition*, Den Haag, 1992, S. 297–301.

[Kum93a] F. Kummert, E. Littmann, A. Meyering, S. Posch, H. Ritter, G. Sagerer: *A Hybrid Approach to Signal Interpretation Using Neural and Semantic Networks*, in S. Pöppl, H. Handels (Hrsg.): *Mustererkennung 1993. Mustererkennung im Dienste der Gesundheit. 15. DAGM-Symposium, Lübeck*, Informatik aktuell, Springer-Verlag, Berlin, 1993, S. 245–252.

[Kum93b] F. Kummert, H. Niemann, R. Prechtel, G. Sagerer: *Control and explanation in a signal understanding environment, Signal Processing*, Bd. 32, Nr. 1-2, Special Issue on Intelligent Systems for Signal and Image Understanding, 1993, S. 111–145.

[Kum94] V. Kumar, A. Grama, A. Gupta, G. Karypis: *Introduction to Parallel Computing*, The Benjamin/Cummings Publ. Co., Inc., Redwoos City, Calif., 1994.

[Laa87] P. v. Laarhoven, E. Aarts: *Simulated Annealing: Theory and Applications*, D. Reidel Publ. Comp., Dordrecht, 1987.

[Lee90] D. Lee, W. Davis: *On linear speedup of a class of neighborhood functions in an array processor*, in *Proc. of the IEEE 2nd Symp. on Parallel and Distributed Processing*, Dallas, Tex., 1990, S. 441–446.

[Lee92a] K. Lee, S. Lee: *Asynchronous Communication of Multiple Markov Chains in Parallel Simualted Annealing*, in *Proc. of the 21st Int. Conf. on Parallel Processing*, Bd. 3, Ann Arbor, Mich., 1992, S. 169–176.

[Lee92b] K. Lee, S. Lee: *Efficient Parallelization of Simulated Annealing using Multiple Markov Chains: An Application to Graph Partitioning*, in *Proc. of the 21st Int. Conf. on Parallel Processing*, Bd. 3, Ann Arbor, Mich., 1992, S. 177–180.

[Lev79] H. Levesque, J. Mylopoulos: *A Procedural Semantics for Semantic Networks*, in [Fin79], 1979, S. 93–121.

[Lev87] S. Levitan, C. Weems, A. Hanson, E. Riseman: *The UMass Image Understanding Architecture*, in [Uhr87], 1987, S. 215–248.

[Lie89] C. Liedtke, M. Ender: *Wissensbasierte Bildverarbeitung*, Springer-Verlag, Berlin, 1989.

[Lin94] C. Linster, W. Stukenbrock, T. Thiel, S. Turowski: *Das Multiprozessorsystem MEMSY. Programmiermodell, Betriebssystemarchitektur, Kommunikation und dynamische Rekonfiguration*, in [Wed94], 1994, S. 206–228.

[Lio86] S. Liou, R. Jain: *Road Following using Vanishing Points*, in *Proc. of the 8th Int. Conf. on Computer Vision and Pattern Recognition*, Miami Beach, Fla., 1986, S. 41–46.

[Man89] B. Manderick, P. Spiessens: *Fine-Grained Parallel Genetic Algorithms*, in [Sch89], 1989, S. 428–433.

[Mar77] A. Martelli: *On the Complexity of Admissible Search Algorithms*, *Artificial Intelligence*, Bd. 8, 1977, S. 1–13.

[Mas93] M. Mast: *Ein Dialogmodul für ein Spracherkennungs– und Dialogsystem*, Bd. 50 von *Dissertationen zur Künstlichen Intelligenz*, Infix, Sankt Augustin, 1993.

[Mas94] M. Mast, F. Kummert, U. Ehrlich, G. Fink, T. Kuhn, H. Niemann, G. Sagerer: *A Speech Understanding and Dialog System with a Homogeneous Linguistic Knowledge Base*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 16, Nr. 2, 1994, S. 179–194.

[Mat90a] R. Mathar, D. Pfeifer: *Stochastik für Informatiker*, B.G. Teubner Verlag, Stuttgart, 1990.

[Mat90b] T. Matsuyama, V. Hwang: *SIGMA. A Knowledge-Based Aerial Image Understanding System*, Bd. 12 von *Advances in Computer Vision and Machine Intelligence*, Plenum Press, New York and London, 1990.

[McK85] D. McKeown, W. Harvey, J. McDermott: *Rule-Based Interpretation of Aerial Imagery*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 7, Nr. 5, 1985, S. 570–585.

[Met53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller: *Equation of state calculations for fast computing machines*, *Journal of Chemical Physics*, Bd. 21, Nr. 6, 1953, S. 1087–1092.

[Min75] M. Minsky: *A framework for Representing Knowledge*, in P. Winston (Hrsg.): *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975, S. 211–277.

[Mol89] D. Moldovan, W. Lee, C. Lin: *SNAP: A Marker Propagation Architecture for Knowledge Processing*, Technical Report CENG 89-10, University of Southern California, 1989.

[Mol90] D. Moldovan, W. Lee, C. Lin, S. Chung: *Parallel Knowledge Processing on SNAP*, in *Proc. of the 19th Int. Conf. on Parallel Processing*, Bd. 1, Urbana-Champaign, Ill., 1990, S. 474–481.

[Mon87] B. Monion, O. Vornberger: *Parallel Processing of Combinatorial Search Trees*, in A. Albrecht, H. Jung, K. Mehlhorn (Hrsg.): *Parallel Algorithms and Architectures*, Bd. 269 von *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1987, S. 60–69.

[Mor94] R. Moratz, S. Posch, G. Sagerer: *Controlling Multiple Neural Nets with Semantic Networks*, in W. Kropatsch, H. Bischof (Hrsg.): *Mustererkennung 1994. Erkennen und Lernen. 16. DAGM-Symposium, Wien*, Informatik Xpress 5, Springer-Verlag, Berlin, 1994, S. 288–295.

[Müh89] H. Mühlenbein: *Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization*, in [Sch89], 1989, S. 416–421.

[Müh91] H. Mühlenbein, M. Schomisch, J. Born: *The parallel genetic algorithm as function optimizer*, *Parallel Computing*, Bd. 17, Nr. 6 & 7, 1991, S. 619–632.

[Nag80] M. Nagao, T. Matsuyama: *A Structural Analysis of Complex Aerial Photographs*, Plenum Press, New York, 1980.

[Naz84] A. Nazif, M. Levine: *Low Level Image Segmentation: An Expert System*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 6, Nr. 5, 1984, S. 555–577.

[Neb88] B. Nebel: *Computational Complexity of Terminological Reasoning in BACK*, *Artificial Intelligence*, Bd. 34, Nr. 3, 1988, S. 371–383.

[Neb90] B. Nebel: *Reasoning and Revision in Hybrid Representation Systems*, Bd. 422 von *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 1990.

[Nev80] R. Nevatia, R. Babu: *Line Feature Extraction and Description*, *Computer Vision, Graphics and Image Processing*, Bd. 13, 1980, S. 257–269.

[Nie81] H. Niemann: *Pattern Analysis*, Nr. 4 in Springer Series in Information Sciences, Springer-Verlag, Berlin, Heidelberg, New York, 1981.

[Nie83] H. Niemann: *Klassifikation von Mustern*, Springer-Verlag, Berlin, 1983.

[Nie85a] H. Niemann: *Wissensbasierte Bildanalyse*, *Informatik-Spektrum*, Bd. 8, Nr. 4, 1985, S. 201 – 214.

[Nie85b] H. Niemann, H. Bunke, I. Hofmann, G. Sagerer, F. Wolf, H. Feistel: *A Knowledge Based System for Analysis of Gated Blood Pool Studies*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 7, Nr. 3, 1985, S. 246 – 259.

[Nie87] H. Niemann, H. Bunke: *Künstliche Intelligenz in Bild- und Sprachanalyse*, B.G. Teubner Verlag, Stuttgart, 1987.

[Nie90a] H. Niemann: *Pattern Analysis and Understanding* Second Edition, Nr. 4 in Springer Series in Information Sciences, Springer-Verlag, Berlin, 1990.

[Nie90b] H. Niemann, H. Brünig, R. Salzbrunn, S. Schröder: *Interpretation of Industrial Scenes by Semantic Networks*, in *Proc. of the IAPR Workshop on Machine Vision Applications*, Tokyo, 1990, S. 39–42.

[Nie90c] H. Niemann, H. Brünig, R. Salzbrunn, S. Schröder: *A Knowledge-Based Vision System for Industrial Applications*, in *Machine Vision and Applications*, Bd. 3, Springer-Verlag, New York, 1990, S. 201–229.

[Nie90d] H. Niemann, G. Sagerer, S. Schröder, F. Kummert: ERNEST*: A Semantic Network System for Pattern Understanding*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 12, Nr. 9, 1990, S. 883–905.

[Nie92] H. Niemann, G. Sagerer, U. Ehrlich, E. Schukat-Talamazzini, F. Kummert: *The Interaction of Word Recognition and Linguistic Processing in Speech Understanding*, in R. De Mori, P. Laface (Hrsg.): *Recent Advances in Speech and Language Modelling*, NATO ASI Series F, Springer-Verlag, Berlin, 1992.

[Nil82] N. Nilsson: *Principles of Artificial Intelligence*, Springer-Verlag, Berlin, 1982.

[Oza86] S. Ozawa, A. Rosenfeld: *Synthesis of a Road Image as seen from a Vehicle*, *Pattern Recognition*, Bd. 19, Nr. 2, 1986, S. 123–145.

[Pal94] S. Pal, D. Bhandari, M. Kundu: *Genetic algorithms for optimal image enhancement, Pattern Recognition Letters*, Bd. 15, Nr. 3, 1994, S. 261–271.

[Pan90] Y. Pan, H. Chuang: *Parallel Hough Transform Algorithm on SIMD Hypercube Arrays*, in P.Yew (Hrsg.): *Proc. of the 19th Int. Conf. on Parallel Processing*, Bd. 3, 1990, S. 83–86.

[Pap82] C. Papadimitriou, K. Steiglitz: *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Inc., Englewood Cliffs, N.J., 1982.

[Pap94] C. Papadimitriou: *Computational Complexity*, Addison–Wesley Publ. Co., Reading, Mass., 1994.

[Pat90] P. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. MacGregor, W. Mark, D. McGuinness, B. Nebel, A. Schmiedel, J. Yen: *Term Subsumption Languages in Knowledge Representation, AI Magazine*, Bd. 11, Nr. 2, 1990, S. 16–23.

[Pau92] D. Paulus: *Objektorientierte und wissensbasierte Bildverarbeitung*, Vieweg, Braunschweig, 1992.

[Pea83] J. Pearl: *Knowledge versus Search: A Quantitative Analysis Using A**, *Artificial Intelligence*, Bd. 20, 1983, S. 1–13.

[Pea84] J. Pearl: *Heuristics. Intelligent Search Strategies for Computer Problem Solving*, Addison–Wesley Publ. Co., Reading, Mass., 1984.

[Per93] M. Perić, M. Schäfer, E. Schreck: *Numerical Simulation of Complex Fluid Flows on MIMD Computers*, in [Bod93], 1993, S. 292–306.

[Pet87] C. Pettey, M. Lenze, J. Grefenstette: *A Parallel Genetic Algorithm*, in [Gre87], 1987, S. 155–161.

[Pit93] I. Pitas (Hrsg.): *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks*, Wiley Series in Parallel Computing, John Wiley & Sons, Chichester, 1993.

[Pos90] S. Posch: *Automatische Bestimmung von Tiefeninformation aus Grauwert-Stereobildern*, Deutscher Universitätsverlag, Wiesbaden, 1990.

[Pra91] V. Prasanna Kumar: *Parallel Architectures and Algorithms for Image Understanding*, Academic Press, Inc., Boston, Mass., 1991.

[Pre91] R. Prechtel, H. Niemann, E. Nöth, G. Sagerer: *Graphical explanations for large semantic networks*, in H. Santo (Hrsg.): *Compugraphics 91. Proc. of the 1st International Conference on Computational Graphics and Visualization Techniques*, Bd. 2, Sesimbra, Portugal, 1991, S. 386–395.

[Pre92] R. Prechtel: *Erklärungen für komplexe Wissensbasen*, Bd. 28 von *Dissertationen zur Künstlichen Intelligenz*, Infix, Sankt Augustin, 1992.

[Pro90] H. Profitlich: *SB-ONE: Ein Wissensrepräsentationssystem basierend auf KL-ONE*, Memo 43, Sonderforschungsbereich 314: Künstliche Intelligenz – Wissensbasierte Systeme, Universität des Saarlandes, Saarbrücken, 1990.

[Qui68] M. Quillian: *Semantic Memory*, in M. Minsky (Hrsg.): *Semantic Information Processing*, Kap. 4, The MIT Press, Cambridge, Mass., 1968, S. 216–270.

[Rab85] L. Rabiner, B. Juang, S. Levinson, M. Sondhi: *Recognition of Isolated Digits Using Hidden Markov Models with Continuous Mixture Densities, Bell Systems Technical Journal*, Bd. 64, 1985, S. 1211–1234.

[Rab89] L. Rabiner: *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, *Proceedings of the IEEE*, Bd. 77, Nr. 2, 1989, S. 257–285.

[Rad93] B. Radig (Hrsg.): *Verarbeiten und Verstehen von Bildern*, Bd. 6.4 von *Handbuch der Informatik*, R. Oldenbourg Verlag, München, Wien, 1993.

[Ran89] S. Ranka, S. Sahni: *Hypercube algorithms for Image Transformations*, in F. Ris, P. Kogge (Hrsg.): *Proc. of the Int. Conf. on Parallel Processing*, Bd. 3, 1989, S. 24–31.

[Ree91] A. Reeves: *Software Computer Vision Environments for Parallel Computers*, in [Pra91], 1991, S. 453–472.

[Rei86] W. Reisig: *Petrinetze.* 2. Auflage, Springer-Verlag, Berlin, 1986.

[Rei92] U. Reimer: *Einführung in die Wissensrepräsentation*, B.G. Teubner Verlag, Stuttgart, 1992.

[Rib88] S. Ribarić: *Knowledge Representation Scheme based on Petri Net Theory*, *Int. Journal of Pattern Recognition and Artificial Intelligence*, Bd. 2, Nr. 4, 1988, S. 691–700.

[Rib93] S. Ribarić: *Inheritance in Knowledge Representation Scheme based on Petri's Net Theory*, *Pattern Recognition and Image Analysis*, Bd. 3, Nr. 3 (Special Issue on the Proceedings of the 3rd Open German-Russian Workshop on Image Interpretation and Pattern Recognition), 1993, S. 300–304.

[Ric83] E. Rich: *Artificial Intelligence*, McGraw-Hill, New York, 1983.

[Ric94] S. Richter, D. Wetzel: *A Robust and Stable Road Model*, in *Proc. of the 12th Int. Conf on Pattern Recognition. Jerusalem, Oktober 1994*, Bd. I, IEEE Computer Society Press, 1994, S. 777–780.

[Ros88] J. Rost, E. Mähle: *Implementation of a Parallel Branch-and-Bound Algorithm for the Travelling Salesman Problem*, in C. Jesshope, K. Reinartz (Hrsg.): *CONPAR 88*, Cambridge University Press, Aachen, 1988, S. 152–159.

[Rou89] C. Roucairol: *Parallel Branch and Bound Algorithms — an overview*, in M. Cosnard, Y. Robert, P. Quinton, M. Raynal (Hrsg.): *Parallel & Distributed Algorithms*, North–Holland, Amsterdam, 1989, S. 153–163.

[Rou90] P. Roussel-Ragot, G. Dreyfus: *A Problem Independent Parallel Implementation of Simulated Annealing: Models and Experiments*, *IEEE Trans. on Computer-Aided Design*, Bd. 9, Nr. 8, 1990, S. 827–835.

[Rou92] P. Roussel-Ragot, G. Dreyfus: *Parallel Annealing by Multiple Trials: An Experimental Study on a Transputer Network*, in [Aze92c], Kap. 7, 1992, S. 91–108.

[Rut86] R. Rutenbar, S. Kravitz: *Layout by Annealing in a Parallel Environment*, in *Proc. of the IEEE Int. Conf. on Computer Design*, Port Chester, N.Y., October 1986, S. 434–437.

[Sac92] L. Sachs: *Angewandte Statistik. Statistische Methoden und ihre Anwendungen.* Siebente Auflage, Springer-Verlag, Berlin, 1992.

[Sag85] G. Sagerer: *Darstellung und Nutzung von Expertenwissen für ein Bildanalysesystem*, Bd. 104 von *Informatik-Fachberichte*, Springer-Verlag, Berlin, 1985.

[Sag88] G. Sagerer: *Automatic interpretation of medical image sequences*, *Pattern Recognition Letters*, Bd. 8, Nr. 2, 1988, S. 87–102.

[Sag90a] G. Sagerer: *Automatisches Verstehen gesprochener Sprache*, Bd. 74 von *Reihe Informatik*, B.I. Wissenschaftsverlag, Mannheim, Wien, Zürich, 1990.

[Sag90b] G. Sagerer, R. Prechtel, H.-J. Blickle: *Ein System zur automatischen Analyse von Sequenzszintigrammen des Herzens*, *Der Nuklearmediziner*, Bd. 3, 1990, S. 137–154.

[Sag93] G. Sagerer: *Neuronal, Statistisch, Wissensbasiert: Ein Beitrag zur Paradigmendiskussion für die Mustererkennung*, in S. Pöppl, H. Handels (Hrsg.): *Mustererkennung 1993. Mustererkennung im Dienste der Gesundheit. 15. DAGM-Symposium, Lübeck*, Informatik aktuell, Springer-Verlag, Berlin, 1993, S. 158–177.

[Sah87] R. Sahner, K. Trivedi: *Performance Analysis and Reliability Analysis using Directed Acyclic Graphs*, *IEEE Trans. on Software Engineering*, Bd. SE-13, Nr. 10, 1987, S. 1105–1114.

[Sak85] K. Sakaue, H. Tamura: *Automatic Generation of Image Processing Programs*, in *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*, San Francisco, Calif., 1985, S. 189–192.

[Sal92] R. Salzbrunn, H. Niemann, M. Harbeck, A. Winzen: *Object Recognition by a robust Matching Technique*, in *Advances in Structural and Syntactic Pattern Recognition. Proc. of the International Workshop on Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, S. 481–495.

[Sch78] J. Schürmann: *A Multifont Word Recognition System for Postal Address Reading*, *IEEE Trans. on Computers*, Bd. 27, 1978, S. 721–732.

[Sch83] J. Schmolze, T. Lipkis: *Classification in the KL-ONE Knowledge Representation System*, in *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence*, Karlsruhe, 1983, S. 330–332.

[Sch86] P. Schefe: *Künstliche Intelligenz — Überblick und Grundlagen*, B.I. Wissenschaftsverlag, Mannheim, Wien, Zürich, 1986.

[Sch89] J. Schaffer (Hrsg.): *Proc. of the 3rd Int. Conf. on Genetic Algorithms.* George Mason University, Fairfax, Va., Morgan Kaufmann, San Mateo, Calif., 1989.

[Sch90a] H. Schomberg: *A transputer-based shuffle-shift machine for image processing and reconstruction*, in *Proc. of the 10th Int. Conf. on Pattern Recognition*, Atlantic City, N.J., 1990, S. 445–450.

[Sch90b] S. Schröder: *Integration einer Wissenserwerbskomponente in eine Systemumgebung für die Musteranalyse*, Bd. 136 von *Reihe 10: Informatik/Kommunikationstechnik*, VDI Verlag, Düsseldorf, 1990.

[Sch92a] A. Schill: *Remote Procedure Call: Fortgeschrittene Konzepte und Systeme — ein Überblick. Teil 1: Grundlagen*, *Informatik-Spektrum*, Bd. 15, 1992, S. 79–87.

[Sch92b] A. Schill: *Remote Procedure Call: Fortgeschrittene Konzepte und Systeme — ein Überblick. Teil 2: Erweiterte RPC-Ansätze*, *Informatik-Spektrum*, Bd. 15, 1992, S. 145–155.

[Sch94] H. Schwefel, U. Hammel, T. Bäck: *Evolutionäre Algorithmen: Optimieren nach dem Vorbild der biologischen Evolution*, *Der GMD-Spiegel. Systemanalyse und parallele Simulation*, Bd. 24, Nr. 1/2, 1994, S. 49–58.

[Sen81] E. Seneta: *Non-negative Matrices and Markov Chains*, Springer-Verlag, New York, 1981.

[Sha87a] M. Sharma, N. Ahuja, J. Patel: *NETRA: An Architecture for a Large Scale Multiprocessor Vision System*, in [Uhr87], 1987, S. 87–105.

[Sha87b] L. Shastri: *A Connectionist Encoding of Semantic Networks*, in *Proc. of the 9th Annual Conference of the Cognitive Science Society. Seattle, Wash.*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1987, S. 143–154.

[Sha88] L. Shastri: *Semantic Networks: An Evidential Formalization and its Connectionist Realization*, Research Notes in Artificial Intelligence, Pitman and Morgan Kaufmann Publishers, Inc., London and San Mateo, Calif., 1988.

[Sha89] L. Shastri: *Connectionism, Knowledge Representation, and Effective Reasoning*, in W. Brauer, C. Freksa (Hrsg.): *Wissensbasierte Systeme. 3. Internationaler GI-Kongreß,* München, Bd. 227 von *Informatik Fachberichte*, Springer-Verlag, Berlin, 1989, S. 186–195.

[Sha91] L. Shastri: *Why Semantic Networks ?*, in [Sow91], Kap. 3, 1991, S. 109–136.

[Sho93] R. Shonkwiler: *Parallel Genetic Algorithms*, in [For93], 1993, S. 199–205.

[Shu90] D. Shu, J. Nash, C. Weems: *A Multiple-Level Heterogenous Architecture for Image Understanding*, in *Proc. of the 10th Int. Conf. on Pattern Recognition*, Atlantic City, N.J., 1990, S. 629–634.

[Son85] E. Sontag, H. Sussmann: *Image Restoration and Segmentation using the Annealing Algorithm*, in *Proc. of the 24th Conf. on Decision and Control*, Ft. Lauderdale, Tex., December 1985, S. 768–773.

[Söt90] F. Sötz: *A Method for Performance Prediction of Parallel Programs*, in H. Burkhart (Hrsg.): *CONPAR 90 – VAPP IV. Proc. of the Joint Int. Conf. on Vector and Parallel Processing,* Zürich, Switzerland, Bd. 457 von *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1990, S. 98–107.

[Sou92] L. Sousa, J. Bárrios, A. Costa, M. Piedade: *Parallel Image Processing for Transputer Based Systems*, in *Int. Conf. on Image Processing and its Applications*, Maastricht, 1992, S. 33–36.

[Sow91] J. Sowa (Hrsg.): *Principles of Semantic Networks. Explorations in the Representation of Knowledge*, Morgan Kaufmann Publishers, Inc., San Mateo, Calif., 1991.

[Spi91] P. Spiessens, B. Manderick: *A Massively Parallel Genetic Algorithm. Implementation and First Analysis*, in [Bel91], 1991, S. 279–286.

[Ste90] S. Steuer: *Gruppierung flächenhafter Segmente in Bildfolgen*, Bayerisches Forschungszentrum für Wissensbasierte Systeme (FORWISS), Forschungsgruppe Kognitive Systeme, München, 1990.

[Ste91] S. Steuer: *Erstellung eines ersten Modells in* ERNEST *zur Identifikation der Strasse und der Position des Kamerafahrzeugs im statischen Bild*, Projekt MOVIE, Bericht 3.2.B1, Bayerisches Forschungszentrum für Wissensbasierte Systeme (FORWISS) und Bayerische Motorenwerke AG (BMW AG), München, 1991.

[Ste93] J. Stender (Hrsg.): *Parallel Genetic Algorithms: Theory and Applications*, IOS Press, Amsterdam, 1993.

[Suh87] J. Suh, D. Gucht: *Distributed genetic algorithms*, Technical Report 225, Computer Science Department, Indiana University, 1987.

[Sun90a] V. Sunderam: *PVM: A Framework for Parallel Distributed Computing, Concurrency: Practice & Experience*, Bd. 2, Nr. 4, 1990, S. 315–339.

[Sun90b] M. Sunwoo, J. Aggarwal: *VisTA for a General Purpose Computer Vision System*, in *Proc. of the 10th Int. Conf. on Pattern Recognition*, Atlantic City, N.J., 1990, S. 635–641.

[Tan83] S. Tanimoto: *A pyramidal approach to parallel processing*, in *Proc. of the 10th ACM Int. Symp. on Computer Architecture*, Stockholm, Sweden, 1983, S. 372–378.

[Tan87] R. Tanese: *Parallel Genetic Algorithm for a Hypercube*, in [Gre87], 1987, S. 177–183.

[Tav95] D. Tavangarian, M. Koch, M. Klein, G. Hipper: *Hochleistungs–Datenverarbeitung in Workstation–Clustern*, **it + ti** – *Informationstechnik und Technische Informatik*, Bd. 37, Nr. 1, 1995, S. 29–37.

[Tes66] L. Tesniere: *Elements de syntaxe structurale* Second Edition, Klincksieck, Paris, 1966.

[Tho86] A. Thomasian, P. Bay: *Analytic Queueing Network Models for Parallel Processing of Task Systems*, *IEEE Transactions on Computers*, Bd. C-35, Nr. 12, 1986, S. 1045–1054.

[Tou87] D. Touretzky, S. Geva: *A distributed connectionist representation of concept structures*, in *Proc. of the 9th Annual Conference on the Cognitive Science Society*, Seattle, Wa., 1987, S. 155–164.

[Tso80] J. Tsotsos, J. Mylopoulos, J. Covvey, S. Zucker: *A Framework for Visual Motion Understanding*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 2, 1980, S. 563–573.

[Tuc91] L. Tucker: *Data Parallelism: Image Understanding and the Connection Machine System*, in [Pra91], 1991, S. 473–497.

[Uhr87] L. Uhr: *Parallel Computer Vision*, Academic Press, Inc., Boston, Mass., 1987.

[Wah84] B. Wah, Y. Ma: *MANIP – A Multicomputer Architecture for Solving Combinatorial Extremum Problems*, *IEEE Trans. on Computers*, Bd. C-33, Nr. 5, 1984, S. 377–390.

[Wah85] B. Wah, G. Li, C. Yu: *Multiprocessing of Combinatorial Search Problems*, *IEEE Computer*, Bd. 18, Nr. 6, 1985, S. 93–108.

[Wah90] B. Wah, G. Li, C. Yu: *Multiprocessing of Combinatorial Search Problems*, in [Kum90], 1990, S. 103–145.

[Wed94] H. Wedekind (Hrsg.): *Verteilte Systeme. Grundlagen und zukünftige Entwicklungen aus der Sicht des Sonderforschungsbereichs 182 "Multiprozessor– und Netzwerkkonfigurationen*, B.I. Wissenschaftsverlag, Mannheim, 1994.

[Wee91] C. Weems, E. Riseman, A. Hanson, A. Rosenfeld: *Preliminary Results from the DARPA Integrated Image-Understanding Benchmark*, in [Pra91], 1991, S. 399–449.

[Wei93] P. Weierich, D. Wetzel, B. Bühnemann, H. Niemann, K. Glückert: *AUDIGON – Ein medizinisches Expertensystem zur Diagnose der Kniegelenksarthrose aus kernspintomographischen Bildern*, in S. Pöppl, H. Handels (Hrsg.): *Mustererkennung 1993. Mustererkennung im Dienste der Gesundheit. 15. DAGM-Symposium, Lübeck*, Informatik aktuell, Springer-Verlag, Berlin, 1993, S. 110–117.

[Wet93] D. Wetzel, A. Zins, H. Niemann: *Edge- and Motion-Based Segmentation for Traffic Scene Analysis*, *Pattern Recognition and Image Analysis*, Bd. 3, Nr. 3 (Special Issue on the Proceedings of the 3rd Open German-Russian Workshop on Image Interpretation and Pattern Recognition), 1993, S. 385–387.

[Win84] P. Winston: *Artificial Intelligence*, Addison–Wesley Publ. Co., Reading, Mass., 1984.

[Wit90] E. Witte, R. Chamberlain, M. Franklin: *Parallel Simulated Annealing using Speculative Computation*, in *Proc. of the 19th Int. Conf. on Parallel Processing*, Bd. 3, Urbana-Champaign, Ill., 1990, S. 286–290.

[Woo75] W. Woods: *What's in a Link? Foundations for Semantic Networks*, in D. Bobrow, A. Collins (Hrsg.): *Representation and Understanding*, Academic Press, New York, 1975, S. 35–82.

[Zad65] L. Zadeh: *Fuzzy Sets*, *Information and Control*, Bd. 8, Nr. 3, 1965, S. 338–353.

[Zad75] L. Zadeh: *Fuzzy Sets and Their Applications*, in L. Zadeh, K. Fu, K. Tanaka, M. Shimura (Hrsg.): *Fuzzy Sets and Their Application to Cognitive and Decision Processes*, Academic Press, New York, 1975.

Sachregister