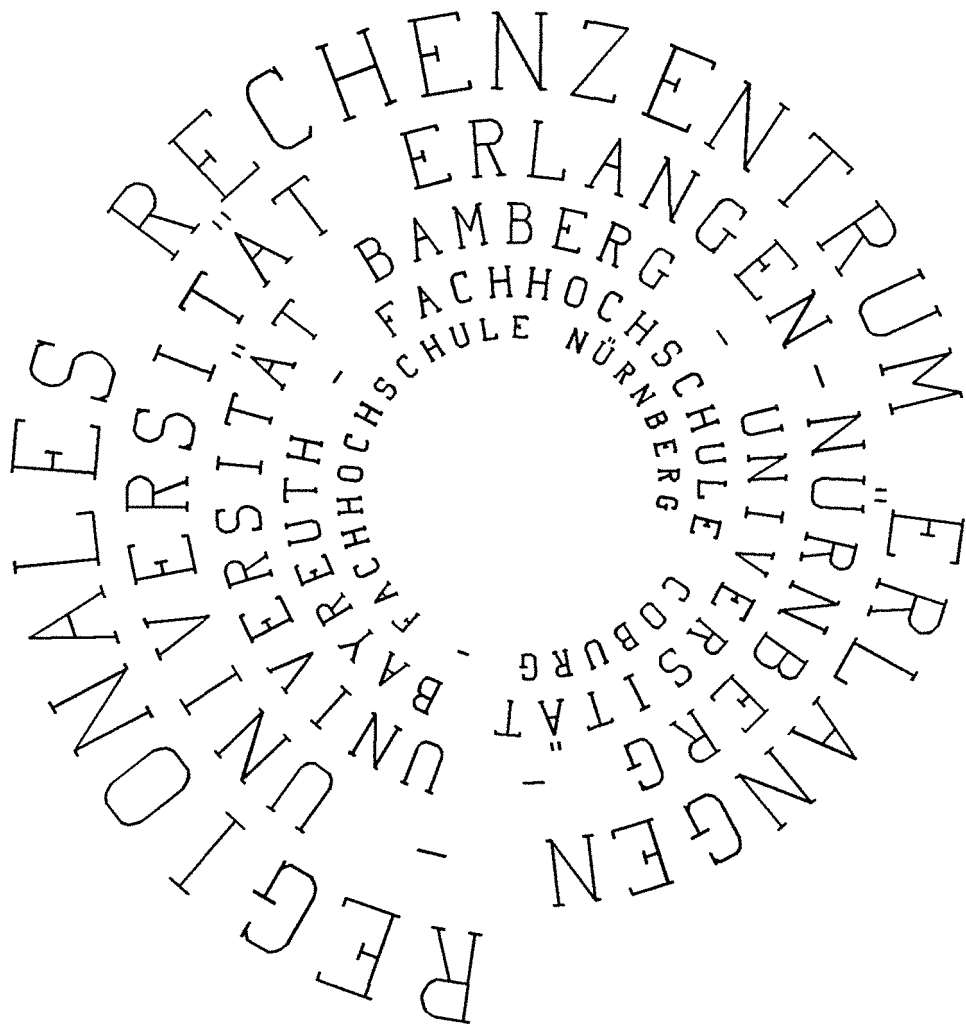


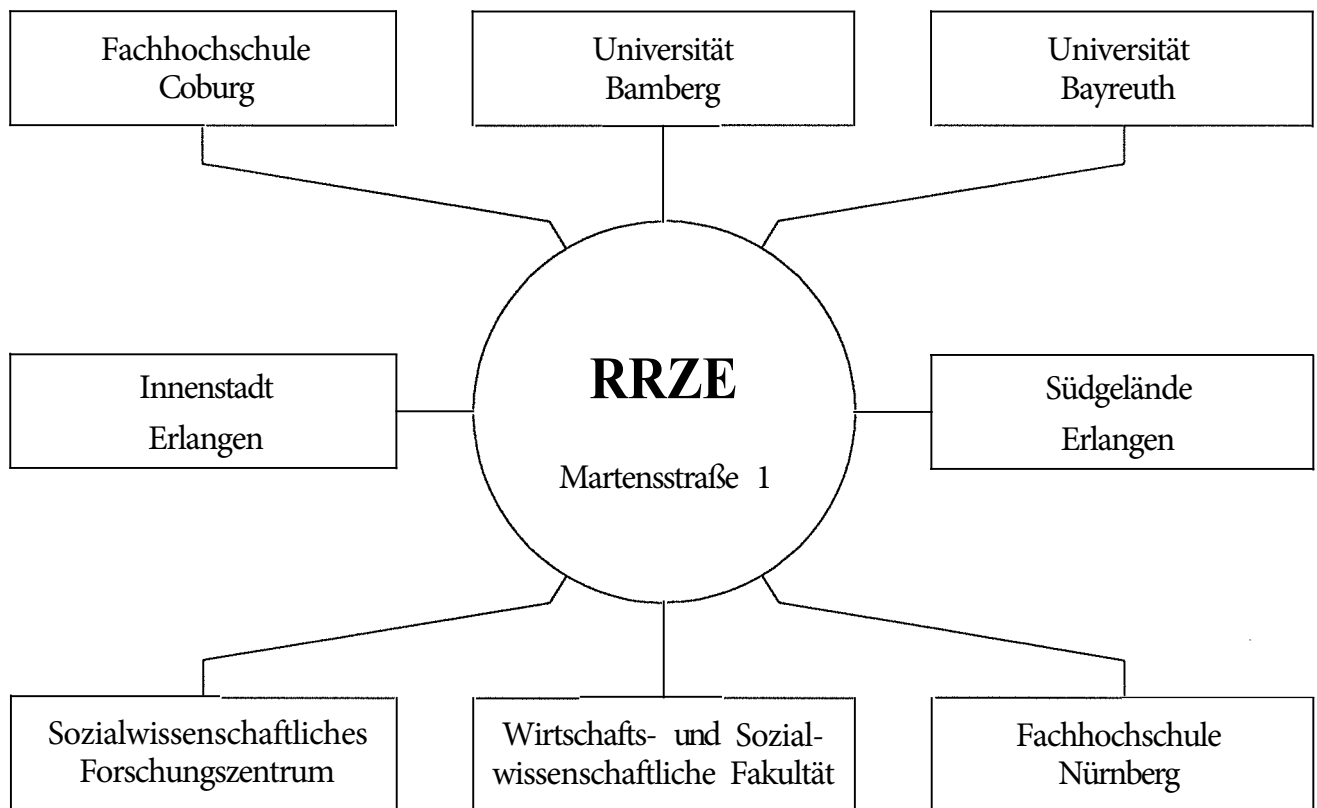
# MITTEILUNGSBLATT

DES REGIONALEN  
RECHENZENTRUMS ERLANGEN

HERAUSGEBER F. WOLF



Nr. 54 - ERLANGEN - DEZEMBER 1989



## Regionales Rechenzentrum Erlangen

Martensstraße 1  
D-8520 Erlangen  
Tel. 09131/85-7031

Beteiligte Institutionen:  
Universität Erlangen-Nürnberg  
Universität Bamberg  
Universität Bayreuth  
Fachhochschule Coburg  
Fachhochschule Nürnberg

ISSN 0172-2905 (Mitteilungsblatt des Regionalen Rechenzentrums)  
Redaktion dieser Ausgabe: H. Henke

**ALLOC - ein wissensbasierter Ansatz**

**zur Lösung des Allokationsproblems von Tasks**

**in verteilten Realzeitsystemen**

von

Gerhard Hergenröder

Erlangen 1989

# MITTEILUNGSBLATT

DES REGIONALEN  
RECHENZENTRUMS ERLANGEN

HERAUSGEBER F. WOLF

**ALLOC - ein wissensbasierter Ansatz  
zur Lösung des Allokationsproblems von Tasks  
in verteilten Realzeitsystemen**

Gerhard Hergenröder

Nr. 54 - ERLANGEN - DEZEMBER 1989



**ALLOC - ein wissensbasierter Ansatz**  
**zur Lösung des Allokationsproblems von Tasks**  
**in verteilten Realzeitsystemen**

Der Technischen Fakultät der  
Universität Erlangen-Nürnberg

zur Erlangung des Grades

**DOKTOR-INGENIEUR**

vorgelegt von

Gerhard Hergenröder

Erlangen 1989

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

Tag der Einreichung:	06.07.1989
Tag der Promotion:	27.09.1989
Dekan:	Prof. Dr. F. Hofmann
Berichterstatte:	Prof. Dr. F. Hofmann Prof. Dr. H. Wedekind

## Vorwort

Die vorliegende Arbeit beschäftigt sich mit dem Problem der Allokation (Zuordnung) von Tasks (Prozessen) zu Prozessoren in verteilten Realzeitsystemen. Für Verteilungsprobleme, an denen mehr als zwei Prozessoren beteiligt sind, gibt es keine optimalen Methoden, die in akzeptabler Zeit ein Ergebnis liefern können. Deshalb muß ein heuristischer Ansatz zu einer Lösung führen.

Das Allokationsproblem wird in dieser Arbeit dadurch gelöst, daß die Strategien von Spezialisten im Rechner modelliert werden. Dazu wird zunächst das Expertenwissen zur Lösung des Allokationsproblems analysiert und bewertet. Anschließend wird es als Wissensbasis des Expertensystems ALLOC modelliert. Für die Modellbildung werden Phasen, Regelmengen und Regeln als geeignete Strukturierungsmittel verwendet.

Ziel ist es zu zeigen, daß mit Hilfe wissensbasierter Methoden das Problem der Allokation in annehmbarer Zeit lösbar ist, und die erhaltene Lösung den Anforderungen aus der Praxis der Realzeitprogrammierung gerecht wird.

Um ein Realzeitsystem auf abstraktem Niveau handhaben zu können, benötigt man eine Spezifikation des Systems. Aus dieser Spezifikation wird die gesamte Information für das Allokationssystem gewonnen. Ebenso werden die Ergebnisse der Allokation auf hohem Abstraktioniveau dargestellt. Es existieren dazu eine graphische Ausgabeschnittstelle und eine Erklärungskomponente. Mit der graphischen Ausgabe wird das Ergebnis dem Benutzer in übersichtlicher Weise dargeboten. In der Erklärungskomponente werden die einzelnen Verteilungsüberlegungen des Allokationssystems dem Benutzer zugänglich gemacht.

**Für die Überlassung und Betreuung der Arbeit möchte ich mich bei Herrn Dr. P. Holleczeck und Herrn Prof. Dr. F. Hofmann sehr herzlich bedanken.**

**Herrn Prof. Dr. H. Wedekind danke ich für die Übernahme des Zweitgutachtens.**

**Den (Allokations-) Experten danke ich, daß sie mir ihr Wissen in zahlreichen langen Gesprächen zur Verfügung gestellt haben.**

**Meine Kollegen M. Tieleman und C. Beckstein trugen mit vielen Hinweisen zur sachlich richtigen Darstellungen auf dem Gebiet der künstlichen Intelligenz bei. Dafür danke ich ihnen.**

**Herrn Dr. P. Holleczeck, meinen Kollegen R. Müller und M. Trautner und meiner Ehefrau Christiane Barton-Hergenröder danke ich für alle Anregungen und die sorgfältige Durchsicht der Arbeit.**

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
1.1. Motivation	2
1.2. Aufbau der Arbeit	3
<b>2. Realzeitaufgaben und Allokation</b>	<b>5</b>
2.1. Realzeitsysteme	5
2.2. Grundlegende Definitionen	6
2.3. Formulierung des Allokationsproblems	11
2.3.1. Definition des Allokationsproblems	11
2.3.1.1. Zielfunktion der Allokation	13
2.3.1.2. Randbedingungen	13
2.3.1.2.1. Betriebsmittel	14
2.3.1.2.2. Gleichmäßige Auslastung	15
2.3.1.2.3. Benutzerdefinierte Einschränkungen für Tasks	15
2.3.2. Definition des Allokationsproblems in der vorliegende Arbeit	17
2.4. Dynamische/statische Allokation von Tasks	18
2.5. Abgrenzung zu technisch/wissenschaftlichen Berechnungen	18
2.6. Abgrenzung gegenüber Scheduling	19
<b>3. Einbindung in das bestehende Umfeld.</b>	<b>21</b>
3.1. Die Spezifikationsmethode PASS	21
3.2. Umsetzung der graphischen Eingabe	23
3.3. Bestimmen der Kommunikationskosten	24
3.4. Zusammenfassung	24
<b>4. Dem Allokationsproblem verwandte Probleme</b>	<b>26</b>
4.1. Das quadratische Zuordnungsproblem	26
4.2. Das Problem des Handlungsreisenden	28
<b>5. Lösungsverfahren</b>	<b>30</b>
5.1. NP-vollständige Probleme	30
5.2. Optimale Verfahren	33
5.2.1. Mathematische Verfahren	33
5.2.2. Enumeration	34

9.3.2.4. Abbildung auf die vorhandenen Prozessoren. . . . .	137
9.3.2.5. Verschieben einzelner Tasks . . . . .	139
9.3.2.6. Verschieben einzelner Cluster . . . . .	140
9.4. Graphische Ausgabe . . . . .	141
9.5. Erklärungskomponente . . . . .	142
9.6. Ablauf des Allokationssystems . . . . .	144
9.7. Zusammenfassung . . . . .	147
<b>10. Bewertung der Ergebnisse . . . . .</b>	<b>149</b>
10.1. Optimale Methoden . . . . .	151
10.2. Heuristische Methoden. . . . .	153
10.3. Vergleich mit einem heuristischen Ansatz . . . . .	156
10.4. Beispiel aus der Realzeitpraxis . . . . .	160
10.5. Zusammenfassung . . . . .	162
<b>11. Zusammenfassung . . . . .</b>	<b>164</b>

## Anhang

<b>I. Beispiel für die Anwendung . . . . .</b>	<b>1</b>
A. Beschreibung der Eingabe . . . . .	1
B. Bewertung der Ergebnisse . . . . .	8
<b>II. Syntaxdefinition . . . . .</b>	<b>16</b>
A. Zulässige Einheiten . . . . .	16
B. Beschreibung der Syntax des Kommunikationsdiagramms . . . . .	16
C. Beschreibung der Syntax des Hardwarediagramms . . . . .	17
<b>III. Definition der Symbole für die Eingabe . . . . .</b>	<b>19</b>

---

5.2.2.1. Branch and Exclude . . . . .	35
5.2.2.2. Branch and Bound . . . . .	36
5.2.2.3. Dynamische Programmierung . . . . .	37
5.3. Suboptimale (heuristische) Verfahren . . . . .	37
5.3.1. Eröffnungsverfahren. . . . .	38
5.3.2. Iterationsverfahren . . . . .	39
5.3.3. Beispiel für ein heuristisches Verfahren . . . . .	39
5.4. Zusammenfassung . . . . .	40
<b>6. Allokationssysteme . . . . .</b>	<b>42</b>
6.1. Lösung durch optimale Verfahren . . . . .	43
6.2. Lösung durch suboptimale (heuristische) Verfahren. . . . .	50
6.2.1. Graphentheoretische Verfahren . . . . .	50
6.2.2. Verfahren mit heuristischen Abbruchkriterien . . . . .	53
6.2.2.1. Das Verfahren nach Ma. . . . .	53
6.2.2.2. Das Verfahren nach Bormann . . . . .	55
6.3. Zusammenfassung . . . . .	57
<b>7. Wissensbasierte Systeme . . . . .</b>	<b>60</b>
7.1. Abriß zur KI . . . . .	60
7.2. Grundlagen wissensbasierter Systeme . . . . .	63
7.3. Techniken der Wissensrepräsentation . . . . .	66
7.4. Produktions- und Regelsysteme . . . . .	73
7.4.1. Grundlagen der Regelsysteme . . . . .	73
7.4.2. Systemkomponenten eines regelbasierten Systems . . . . .	74
7.4.2.1. Recognize-Act-Cycle. . . . .	74
7.4.2.2. Aufbau der Regeln. . . . .	77
7.4.2.3. Strategien . . . . .	77
7.4.3. Regeln beim Allokationssystem . . . . .	80
7.5. Expertensysteme . . . . .	81
7.6. Expertensystem - ein Beispiel . . . . .	83
7.7. Zusammenfassung . . . . .	86

---

<b>8. Wissensbasierte Lösung des Allokationsproblems</b>	88
8.1. Das Expertenwissen bei der Allokation	88
8.2. Aufbereitung des Wissens	92
8.2.1. Zuordnungseinschränkungen	92
8.2.2. Verteilung der Tasks auf die Prozessoren	98
8.2.3. Verbessern der Lösung	99
8.2.4. Optimieren nach verschiedenen Kriterien	100
8.3. Repräsentation des Wissens	101
8.3.1. Verfeinerung der Arbeitsschritte	101
8.3.2. Forderungen an die Repräsentation	102
8.3.3. Repräsentation des Wissens bei der Allokation	103
8.3.4. Wissensrepräsentation am Beispiel der Clusteranalyse	106
8.3.4.1. Kriterien	106
8.3.4.2. Vorgehensweise	107
8.3.4.3. Umsetzung	108
8.3.4.4. Erläuterung der Regelmengen und der Regeln	109
8.3.4.4.1. Regelmenge I	109
8.3.4.4.2. Regelmenge II	110
8.3.4.4.3. Regelmenge III	112
8.3.4.4.4. Vergleich mit anderen Methoden	115
8.4. Übersicht über die Regeln	117
8.5. Zusammenfassung	120
<b>9. Das Allokationssystem</b>	121
9.1. Übersicht	121
9.2. Eingabedaten für das Allokationssystem	126
9.2.1. Umsetzen der graphischen Eingabe	126
9.2.2. Erzeugen der Working Memory Elemente	127
9.3. Das wissensbasierte System ALLOC	129
9.3.1. Beschreibung der Sprache OPS	129
9.3.2. Implementierung des Allokationssystems	135
9.3.2.1. Auswerten der festen Randbedingungen	135
9.3.2.2. Auswerten der Zuordnungseinschränkung	136
9.3.2.3. Clusterbildung	137



<b>IV. Regeln des Systems ALLOC</b>	<b>20</b>
A. Auswerten der festen Randbedingungen	21
B. Auswerten der Zuordnungseinschränkungen.	29
C. Einteilen der Tasks in Cluster	<b>36</b>
D. Verteilen der Cluster auf die Prozessoren.	46
E. Verschieben einzelner Tasks	<b>66</b>
F. Verschieben ganzer Cluster	71

## Literaturverzeichnis

## Stichwortverzeichnis

---

### Verzeichnis der Abbildungen

2.1. Einbettung von Realzeitsystemen . . . . .	7
2.2. Hardware eines verteilten Realzeitsystems /HEIL85/ . . . . .	10
3.1. Einbindung in das bestehende Umfeld . . . . .	25
6.1. Kommunikationsdiagramm . . . . .	44
6.2. Kommunikationsdiagramm mit Rechenkosten . . . . .	45
6.3. Minimaler Schnitt durch das Netzwerk. . . . .	47
6.4. Programmgraph mit der Ausgangslösung. . . . .	51
6.5. Programmgraph mit optimaler Lösung. . . . .	52
7.1. Klassifizierung der KI - Systeme . . . . .	64
7.2. Semantisches Netz . . . . .	69
7.3. Framerepräsentation am Beispiel eines Berichtes . . . . .	71
7.4. Der Recognize-Act-Cycle in einem regelbasierten System . . . . .	76
7.5. Ausführen einer Regel . . . . .	80
7.6. Systemkern eines Expertensystems . . . . .	81
7.7. Schematischer Aufbau eines Expertensystems . . . . .	82
7.8. Eintrag in der Wissensbasis (im System R1) . . . . .	84
7.9. Regel im System R1 . . . . .	84
7.10. Aufteilung der Regeln im System R1 . . . . .	85
8.1. Repräsentationsschema . . . . .	105
8.2. Phasen, Regelmengen und Regeln . . . . .	119
9.1. Ablauf des Allokationssystems . . . . .	125
9.2. Umsetzung der Graphik in Working-Memory-Elemente . . . . .	126
9.3. Graphische Darstellung der Ergebnisse . . . . .	142
9.4. Übersicht über das gesamte Allokationssystem . . . . .	146
10.1. Kommunikationsstruktur . . . . .	151
10.2. Kommunikationsstruktur nach /BORR86/ . . . . .	157

# **Kapitel 1**

## **Einleitung**

### **1. Einleitung**

Seit einigen Jahren wird zunehmend der Versuch unternommen, komplexe Aufgaben der Prozeßautomatisierung durch verteilte Systeme zu lösen. Dabei zieht die Dezentralisierung der Hardwarekomponenten zwangsläufig eine Zerlegung der Aufgabenstellung in kleinere Subsysteme nach sich. Im Rahmen der Realzeitprogrammierung wird dazu das Konzept der Task (Rechen-Prozeß) verwendet.

Die Tasks müssen so auf die Prozessoren abgebildet werden, daß das gesamte Realzeitsystem das vom Entwerfer gewünschte Verhalten zeigt. Da die Anforderungen an die Realzeitprogrammierung in der Praxis /NIEM88/ meist nur durch sehr umfangreiche Systeme gelöst werden können, soll die Verteilung der Tasks auf die Hardwarekomponenten durch rechnergestützte Hilfsmittel erleichtert werden. Die vorliegende Arbeit beschäftigt sich mit bestehenden Methoden, die für die Verteilung von Tasks (Allokation) bei Realzeitsystemen anwendbar sind und entwickelt ein eigenes wissensbasiertes System (ALLOC) zur Lösung des Allokationsproblems.

### 1.1. Motivation

Die Verteilung von Tasks auf Prozessoren ist von hoher Komplexität, so daß Rechnerunterstützung bei der Allokation notwendig ist. Trotz der Unterstützung durch Rechner erweist sich das Allokationsproblem noch als so umfangreich, daß optimale Methoden nur für kleine Systeme {dies sind Systeme mit maximal 3 Prozessoren} anwendbar sind. Deshalb können Verteilungsprobleme im Rahmen der Realzeitprogrammierung nur mit heuristischen Methoden gelöst werden.

Die Verteilung hängt von einer Reihe von Kriterien ab, die die Zuteilung einer Task zu einem Prozessor beeinflussen. Ziel eines Allokationssystems ist es, eine möglichst gute Verteilung der Tasks zu erreichen; dies bedeutet, daß die Verteilung den Anforderungen des Entwicklers von Realzeitsystemen gerecht werden muß. Damit diese Forderung erfüllt werden kann, muß ein Allokationssystem Entscheidungen treffen, die zu einer korrekten Verteilung führen. Das bedeutet, daß das System in jeder Situation in der Lage sein muß, aufgrund der aktuellen Situation und detaillierter Kenntnisse über die Verteilung die notwendigen Schritte auszuführen.

Ein Beispiel für solche Kenntnisse sind die Randbedingungen (Speicherplatz der Prozessoren, Prozeßperipherie, ...), die die Verteilung in entscheidender Weise beeinflussen. Diese Informationen schränken das Allokationsproblem ein.

Um das vorhandene Wissen für ein Allokationssystem verfügbar zu machen, werden Experten befragt und die erhaltene Information geordnet. Dieses problemspezifische Wissen (z.B. über Randbedingungen) wird in geeignete Strukturen umgesetzt. Zusammen mit dem Wissen über die Anwendung des problemspezifischen Wissens wird so die Vorgehensweise eines {Allokations-) Experten nachvollzogen. Diese Art der Wissensrepräsentation wird im Bereich der KI<sup>1</sup> insbesondere bei wissensbasierten System angewandt.

Aufgrund des vorhandenen Expertenwissens wird ein System {ALLOC) entwickelt, das in der Lage ist, das Allokationsproblem bei der Realzeitprogrammierung zu lösen.

---

<sup>1</sup> KI = Künstliche Intelligenz

Das Allokationssystem ist in eine Programmierungsumgebung für verteilte Realzeitanwendungen integriert /HOLL89/. In dieser Umgebung ist es möglich, eine Realzeitaufgabe (in PASS<sup>2</sup> /FLEI84/) zu spezifizieren, die Spezifikation zu überprüfen /ANDR88/ und aus ihr (automatisch) Code für eine Programmiersprache (PEARL) zu generieren /KRA86/. Es empfiehlt sich nun, die Spezifikation für das Allokationssystem mitzubedenken, um eine einheitliche Darstellung bei der Softwareentwicklung zu garantieren. Die Spezifikation für eine Realzeitaufgabe dient deshalb zusätzlich als Eingabe für das Allokationssystem, wobei die gleiche Beschreibung sowohl für technische Prozesse als auch für Tasks verwendet wird.

Die Zielsetzung der vorliegenden Arbeit ist es, ein Allokationssystem zu schaffen, das den Anforderungen aus der Praxis der Realzeitprogrammierung gerecht wird. Um diesen Anforderungen zu genügen, müssen vor allem die Einhaltung der Randbedingungen (z.B. die gleichmäßige Prozessorauslastung, Zuordnungsgebote, Einbeziehung der Prozeßperipherie, ...), ausreichende Möglichkeiten der Optimierung und nicht zuletzt eine leicht verständliche Benutzeroberfläche berücksichtigt werden.

## 1.2. Aufbau der Arbeit

Kapitel 2 beschreibt das Allokationsproblem sowohl informell als auch in einer ausführlichen, formalen Darstellung. Anschließend wird das Problem der Allokation von Forschungsgebieten abgegrenzt, die sich ebenfalls mit der Verarbeitung und Verteilung paralleler Programme beschäftigen.

Kapitel 3 beschreibt die Einbindung dieser Arbeit in die bestehende Programmierungsumgebung für verteilte Realzeitanwendungen. Mit der Allokation wird diese Programmierungsumgebung um einen wichtigen Baustein erweitert.

Kapitel 4 stellt Probleme vor, die dem Allokationsproblem verwandt sind. Anhand dieser Probleme kann gezeigt werden, welche Komplexität die Aufgabenklasse hat, zu der das Allokationsproblem zählt.

---

<sup>2</sup> PASS = farallel Activities Specification Scheme

In Kapitel 5 werden Lösungsverfahren für die Klasse der (NP-vollständigen) Probleme aufgezeigt, zu denen auch das Allokationsproblem gehört. Optimale und suboptimale (heuristische) Verfahren werden dargestellt und die Vor- und Nachteile beider Verfahren aufgezeigt.

Den allgemeinen Betrachtungen zur Lösung von NP-vollständigen Problemen schließt sich eine Übersicht (Kapitel 6) über bestehende Allokationssysteme an. Auch hier werden optimale und suboptimale Lösungsansätze dargestellt.

Da in dieser Arbeit ein wissensbasierter Ansatz zur Lösung des Allokationsproblems führt, werden in Kapitel 7 allgemeine Bemerkungen zur KI gemacht und insbesondere die wissensbasierten Systeme näher betrachtet.

Kapitel 8 beschäftigt sich mit dem Wissen eines Experten, der die Verteilung von Tasks auf Prozessoren vornimmt. Hier werden die einzelnen Randbedingungen und Strategien der Verteilung untersucht, um daraus das Wissen zu erhalten, das durch das wissensbasierte System repräsentiert werden soll.

Anschließend wird gezeigt, wie das Wissen (Fakten-, Strategiewissen) über die Allokation in Strukturen eines wissensbasierten Systems umgesetzt werden konnte.

Kapitel 9 beschreibt das im Rahmen dieser Arbeit entwickelte System (ALLOC) zur Lösung des Allokationsproblems. In diesem wissensbasierten System wird das Expertenwissen durch Regeln repräsentiert. Darüberhinaus wird das System in Phasen und Regelmengen unterteilt, um die Vorgehensweise eines Experten besser zu repräsentieren. In diesem Kapitel werden ferner die Teile des Systems erläutert, die zur graphischen Ein- und Ausgabe (/SCHISS/) notwendig sind. Außerdem wird die Erklärungskomponente (/WISSS8/) vorgestellt, mit deren Hilfe dem Benutzer die Ergebnisse verständlich gemacht werden.

Da es für heuristische Verfahren kaum Möglichkeiten gibt, die Qualität einer Lösung zu beweisen, muß mittels geeigneter Testverfahren die Richtigkeit der Lösungen nachgeprüft werden. Einige Ergebnisse der Tests (/BAUE89/) sind in Kapitel 10 dargelegt.

In Kapitel 11 werden die Ergebnisse dieser Arbeit zusammengefaßt und ein Ausblick gegeben.

Im Anhang werden an einem Beispiel alle Möglichkeiten aufgezeigt, die das Allokationssystem zur Verfügung stellt. Dort finden sich auch alle Regeln in umgangssprachlicher Darstellung.

## Kapitel 2

### Realzeitaufgaben und Allokation

#### 2. Realzeitaufgaben und Allokation

##### 2.1. Realzeitsysteme

Ein Realzeitsystem besteht aus einer Menge zyklischer Tasks (Rechen-Prozesse), von denen jede eine bestimmte Teilaufgabe eigenständig erfüllt. Die Tasks können miteinander kommunizieren. Dabei werden entweder Botschaften über gemeinsame Daten ausgetauscht, oder es wird mit Hilfe von Tasking-Operationen in den Ablauf anderer Tasks eingegriffen. Darüber hinaus können die Tasks mit dem Benutzer des Realzeitsystems kommunizieren und ein technisches System steuern. Bei der Realzeitverarbeitung hängt die Ausführung einer Operation vom Zustand des Gesamtsystems, vom Benutzer und vom technischen Prozeß ab.

Von einem verteilten Realzeitsystem wird gesprochen, wenn die Tasks auf verschiedene, durch Netze gekoppelte Rechnersysteme verteilt sind. Die Tasks können dann untereinander nur durch Botschaften kommunizieren. Evident ist, daß das Problem der Allokation natürlich nur bei verteilten Realzeitsystemen auftritt, da nur hier die Tasks verschiedenen Rechnern zugeteilt werden.

## 2.2. Grundlegende Definitionen

### Prozeßautomatisierung

Prozeßautomatisierung beschäftigt sich mit der Regelung und Steuerung von technischen Prozessen mit Hilfe eines Realzeitsystems. Ein technischer Prozeß ist nach DIN 66201

ein Prozeß, dessen Zustandsgrößen mit technischen Mitteln gemessen, gesteuert und/oder geregelt werden können. Technische Prozesse dienen der Umformung und/oder dem Transport von Materie, Energie und/oder Information.

Die Prozeßautomatisierung wird folgendermaßen definiert:

"Unter Prozeßautomatisierung versteht man die Automatisierung technischer Prozesse. Dabei schränkt der Begriff technischer Prozeß den Terminus Prozeß auf solche Vorgänge ein, bei denen Zustandsgrößen mit technischen Mitteln gemessen, gesteuert und/oder geregelt werden" (/DIN 66201/).

Die Prozeßautomatisierung beschäftigt sich also mit der Regelung und Steuerung von technischen Prozessen. Diese sind dadurch gekennzeichnet, daß gleichzeitig und nichtdeterministisch Ereignisse auftreten können, auf die das System mit Hilfe eines dafür geeigneten Realzeitbetriebssystems reagieren können muß.

### Realzeitsysteme

Mit Realzeitsystemen werden die Anforderungen der Prozeßautomatisierung realisiert. Aufgabe des Realzeitsystems ist es, mit fest vorgegebenen Algorithmen innerhalb bestimmter zu garantierender Zeiten auf äußere Signale zu reagieren. Das technische System wird durch die Tasks im Realzeitsystem gemessen, gesteuert und/oder geregelt.

Abbildung 2.2. zeigt die Einbindung von Realzeitsystemen in die Umgebung der Prozeßautomatisierung:



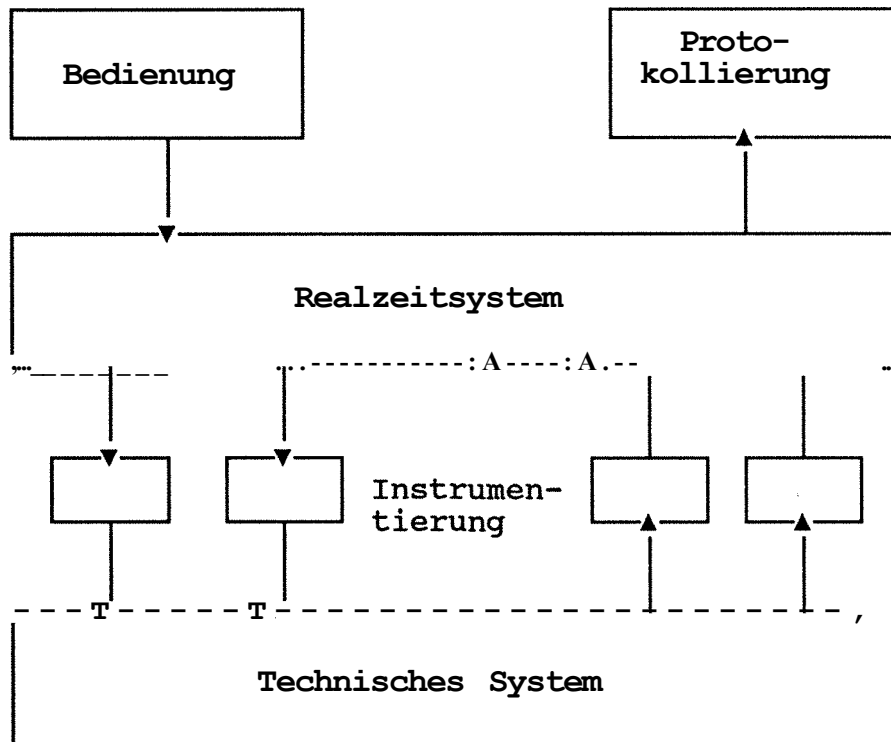


Abb.: 2.1. Einbettung von Realzeitsystemen

### Task (Prozeß)

Die verschiedenen Aufgaben in einem Realzeitsystem werden durch Tasks ausgeführt. Eine Task wird durch ein Quadrupel  $P = (A, D, \text{adr}, w)$  beschrieben (nach /HOFM84/). Dabei gelten folgende Vereinbarungen:

1.  $A$  ist eine endliche Menge von Aktionen
2.  $D = D_1 \times D_2$  ist die Zustandsmenge, wobei  $D_1$  die Menge der Aktionsidentifikationen bezeichnet und  $D_2$  das kartesische Produkt aller Objekte ist, auf die Aktionen Bezug nehmen.
3.  $\text{adr.: } A \rightarrow D_1$  ist eine injektive Abbildung, die jeder Aktion eine Identifikation zuordnet.
4.  $w \in D$  ist der Anfangszustand der Task.

Tasks müssen miteinander kommunizieren und sich synchronisieren können. Die Kommunikation in Realzeitsystemen wird durch gemeinsame Daten realisiert. Die Synchronisation wird durch geeignete Konzepte (z.B. Teste und Setze, Semaphore, kritische Bereiche, Monitore..) erreicht, wobei die Überwachung der Koordinierung durch das Betriebssystem gesteuert wird.

### Verteilte Realzeitsysteme

Ein Realzeitsystem, bei dem die Aufgabe im Verbund von mehreren Rechnern ausgeführt wird, bezeichnet man als verteiltes Realzeitsystem (Bild 2.1. zeigt die Hardware eines verteilten Realzeitsystems). Da die betrachteten verteilten Systeme nicht über einen gemeinsamen Speicher verfügen, sondern der Datenaustausch über Netze stattfindet, muß durch geeignete Mechanismen sichergestellt sein, daß die Tasks dennoch miteinander kommunizieren und sich synchronisieren können. In einem verteilten System ist aber Prozeßsynchronisation ohne Prozeßkommunikation nicht möglich. Für diese Aufgabe wurde das Konzept der Botschaft eingeführt. Die Tasks kommunizieren über Botschaften miteinander, die von dem auf jedem Rechner vorhandenen Betriebssystem verwaltet werden. Durch geeignete Protokolle ist sichergestellt, daß die Botschaften an die richtige Adresse (Task) versandt werden.

Nach /LAUE79/ sind für Systeme, bei denen die Tasks mittels Botschaften synchronisiert werden (oder bei denen die Botschaften zur Kommunikation zwischen Tasks verwendet werden), folgende Punkte zu beachten:

- Die Synchronisation des Zugriffs auf gemeinsame Ressourcen wird mit Warteschlangen implementiert, welche den einzelnen Tasks die die Ressourcen verwalten, fest zugeordnet sind.
- Daten, die von mehreren Tasks bearbeitet werden, werden ebenfalls durch Tasks verwaltet. Diese erhalten die Zugriffsaufträge durch Botschaften.
- Periphere Geräte werden als Tasks betrachtet.
- Den Tasks sind Prioritäten fest zugeordnet.
- Tasks arbeiten mit einer geringen Anzahl von Nachrichten und beenden zuerst die Bearbeitung einer Nachricht, bevor sie die nächste aus der Warteschlange nehmen.
- Die Anzahl der Tasks ist statisch.

Von diesen Anforderungen an ein Realzeitsystem sind für die Allokation folgende besonders zu berücksichtigen:

- Für die Allokation ist es nicht von Bedeutung, welche Art von Botschaftsmechanismus Verwendung findet. Entscheidend für die Verteilung sind vielmehr:
  - die Topologie des Kommunikationsnetzes des verteilten Realzeitsystems und
  - die Kommunikationskosten, die beim Austausch von Botschaften zwischen Tasks entstehen.
- Die peripheren Geräte werden von der Spezifikationsmethode, die in der vorliegenden Arbeit Anwendung findet, als Tasks aufgefaßt. Für das Allokationssystem wird daraus Information zur Einschränkung der Verteilung gewonnen.
- Das entwickelte Allokationssystem geht bei der Verteilung von einer (statisch) festgelegten Anzahl von Tasks aus.

In Bild 2.2. ist die Hardware eines verteilten Realzeitsystems dargestellt.

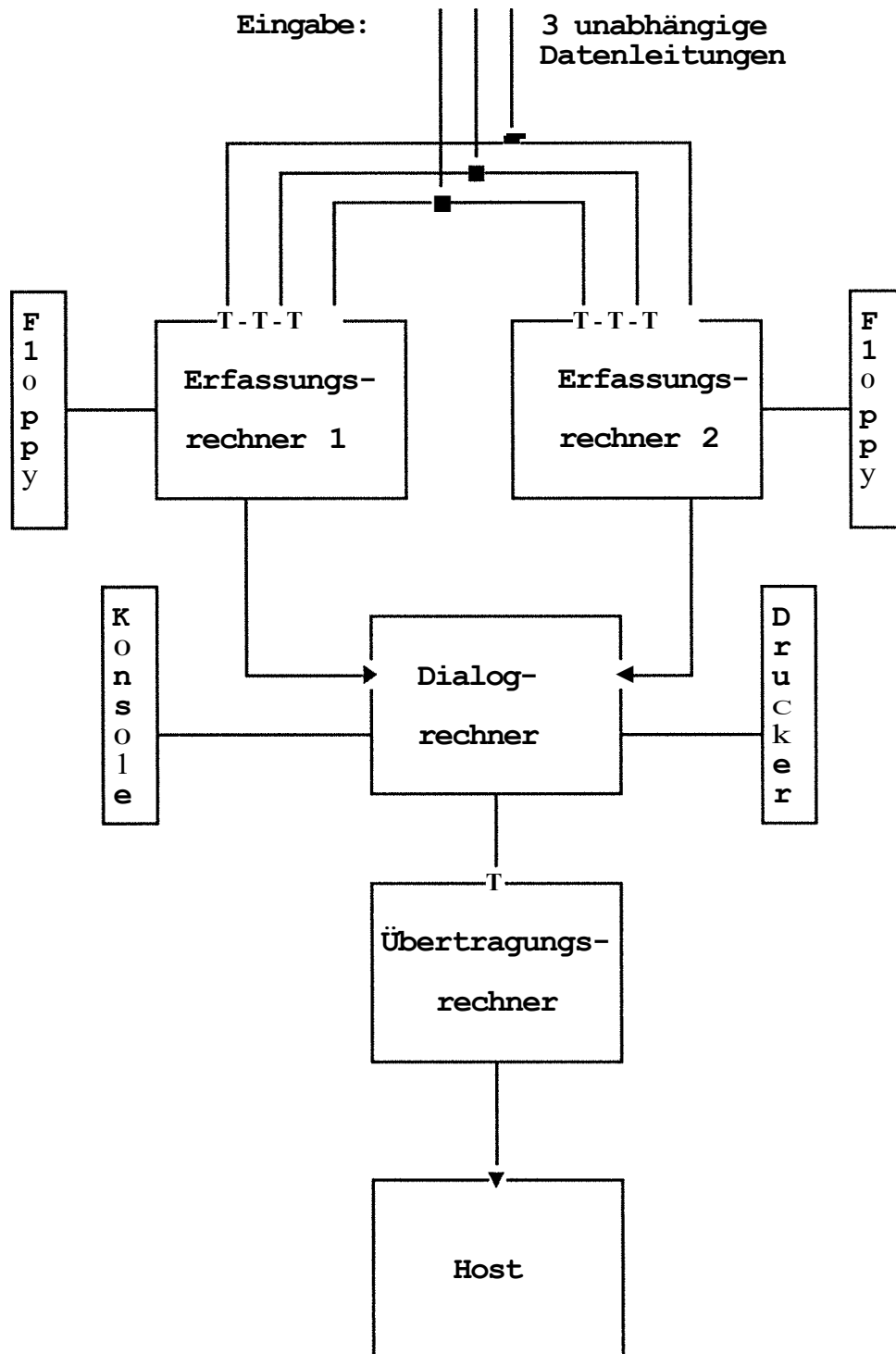


Abb.: 2.2. Hardware eines verteilten Realzeitsystems /HEIL85/

### 2.3. Formulierung des Allokationsproblems

Ziel der Allokation ist es, die Tasks eines gegebenen (verteilten) Realzeitsystems so auf ein vorgegebenes System von Prozessoren (mit Verbindungen zum technischen System) zu verteilen, daß das Gesamtverhalten des Systems optimiert wird. Dabei steht die Optimierung der Inter-Prozessor-Kommunikation im Vordergrund des Interesses; die Menge der Daten, die zwischen Tasks auf getrennt liegenden Prozessoren ausgetauscht werden, soll minimal sein. Tasks, die große Datenmengen miteinander auszutauschen haben, sollen deshalb nach Möglichkeit auf den selben Prozessor platziert werden. Da die Forderung nach Optimierung der Kommunikationskosten nicht ausreichend erschien, werden in der vorliegenden Arbeit weitere Optimierungskriterien zugelassen.

#### 2.3.1. Definition des Allokationsproblems

Einige Definitionen sollen das Allokationsproblem nun näher erläutern:

##### **Informelle Beschreibung des Allokationsproblems (nach /MA82/, /LEE82/):**

Unter Allokation versteht man die Verteilung von Tasks auf Prozessoren (unter Berücksichtigung der Ressourcen) mit dem Ziel, die Inter-Prozessor-Kommunikation zu minimieren.

##### **Formale Beschreibung des Allokationsproblems (nach /BORR86/ u.a.):**

Ziel der Allokation ist es, die Inter-Prozessor-Kommunikation zu minimieren. Als Zielfunktion wird dazu die IPC<sup>3</sup>-Kostenfunktion (DEF.: 2.5. siehe unten) herangezogen.

---

<sup>3</sup> Inter processor Communication = IPC

Zur Beschreibung der IPC-Kostenfunktion sind zunächst einige Definitionen notwendig:

Es existiert eine Menge von Tasks, die auf die zur Verfügung stehenden Prozessoren optimal verteilt werden soll:

$M = \{ 1..m \}$  Menge der Tasks

$N = \{ 1..n \}$  Menge der Prozessoren

Zu jeder Verbindung zwischen den Tasks wird eine Größe angegeben, mit deren Hilfe die Kosten für die Kommunikation dargestellt werden. Für jedes Paar  $i, j$  von Prozessen wird die Größe der Verbindungskosten folgendermaßen definiert

$v_{ij} = 1 \iff$  die Prozesse kommunizieren miteinander

$v_{ij} = 0 \iff$  die Prozesse haben keine Verbindung

Manche der in der Literatur vorgestellten Ansätze arbeiten ausschließlich mit der Einteilung, nach der Prozesse entweder miteinander kommunizieren oder keine Verbindung haben. Für eine genauere Modellierung werden für die  $v_{ij}$  die (Kommunikations-) Kosten der jeweiligen Verbindung eingesetzt. Die Kosten für eine Task-Task-Verbindung werden in der Volumenmatrix abgelegt (DEF. : 2.1.) :

**Volumenmatrix:**  $V = [ v_{ij} ], \quad i, j \in M$

(Die Matrizen sind symmetrisch aufgebaut. Im wissensbasierten System ALLOC werden die Kommunikationsstrukturen unsymmetrisch dargestellt, um eine "doppelte" Verarbeitung zu verhindern).

Die Zuordnung der Tasks zu den Prozessoren wird durch die Zuordnungsmatrix formuliert (DEF. : 2.2.) :

**Zuordnungsmatrix:**  $A = [ a_{ij} ], \quad i \in M, j \in N$

Dabei bedeutet

$a_{ij} = 1 \iff$  Task  $i$  wird von Prozessor  $j$  bearbeitet

$a_{ij} = 0 \iff$  Task  $i$  wird nicht von Prozessor  $j$  bearbeitet

Für die Kosten, die durch die Übertragung der Daten auf einem Kommunikationsmedium verursacht werden, wird die Entfernungsmatrix angegeben (DEF. : 2.3.) :

Entfernungsmatrix:  $E = [e_{ij}]$ ,  $i, j \in N$

#### 2.3.1.1. Zielfunktion der Allokation

Die Kosten für eine Verteilung sind dann minimal, wenn die IPC-Kosten minimal sind. Für ein homogenes System, bei dem alle Prozessoren miteinander verbunden sind, müssen die Tasks und die Kommunikationskosten in der Berechnung berücksichtigt werden. Bei der allgemeineren Form eines heterogenen Netzwerkes müssen zusätzlich die Leitungsparameter (aus der Entfernungsmatrix) berücksichtigt werden. Um in der Summenbildung symmetrische Verteilungen (z.B.  $a_{12} * a_{23}$  und  $a_{23} * a_{12}$ ) auszuschließen, wird in der folgenden Funktion nur eine Hälfte der möglichen Kombinationen betrachtet<sup>4</sup>. Der gesamte IPC-Aufwand ergibt sich somit zu (DEF. : 2.5. IPC-Kostenfunktion) :

$$Z = \sum_{j=1}^m \sum_{i=1}^j \sum_{k=1}^n \sum_{l=1}^n v_{ij} e_{kl} a_{ik} a_{jl}$$

Das Minimum dieser Funktion (bekannt als Zielfunktion des klassischen Zuordnungsproblems) ist die optimale Lösung des Allokationsproblems.

Die IPC-Kostenfunktion berücksichtigt die Kommunikationskosten (DEF.: 2.1.) und die Entfernungskosten (DEF. : 2.3.). Für die Verteilung bei Realzeitsystemen sind aber weitere Bedingungen notwendig, die hier formalisiert werden:

#### 2.3.1.2. Randbedingungen

Als Randbedingung der Allokation werden solche Größen angesehen, die die

---

<sup>4</sup> Betrachtet wird nur eine Dreiecksmatrix einschließlich der Diagonalen. Erreicht wird dies dadurch, daß der Index i nur bis zur jeweiligen Position des Vorgängerindex j läuft.

Verteilung beeinflussen können. Dabei kann es sich um hardwarespezifische Randbedingungen (Betriebsmittel: Rechnerleistung, Speicherplatz, Peripherie..), aber auch benutzerspezifische Randbedingungen (Verbot einer Zuweisung durch den Benutzer, feste Allokation zweier Tasks..) handeln.

#### 2.3.1.2.1. Betriebsmittel

Wie viele Elemente von einem bestimmten Typ an Betriebsmitteln (z.B. Periphere Geräte, Hauptspeicher...) vorhanden sind, wird in der Menge der Ressourcen (Betriebsmitteltypen) angegeben.

$P = \{ 1..p \}$  Menge der Ressourcen (Betriebsmitteltypen)

Die Bedarfsmatrix gibt an, welche Anforderungen die Tasks an das Hardware-system stellen. Hier wird der Bedarf der Tasks an den Ressourcen beschrieben (DEF. : 2.6.) :

Bedarfsmatrix:  $B = [ b_{i,r} ], \quad i \in M, r \in P$

In der Kapazitätsmatrix wird festgelegt, wieviele Ressourcen eines bestimmten Typs ein Prozessor zur Verfügung stellt (DEF. : 2.7.) :

Kapazitätsmatrix:  $K = [ k_{j,r} ], \quad j \in M, r \in P$

Die Randbedingungen werden in Ungleichungen formuliert, die bei der Lösung der IPC-Kostenfunktion zu berücksichtigen sind.

Das folgende Beispiel zeigt eine solche Ungleichung, wobei als Randbedingungen die Betriebsmittelkapazitäten dienen.

Sei:

$P = \{ 1..p \}$  die Menge aller vorhandener Ressourcen

$b_{i,r}, \quad i \in M, r \in P$  der Bedarf von Prozeß  $i$  an Betriebsmittel  $r$

$k_{j,r} \quad j \in N, r \in P$  die Kapazität von Rechner  $j$  an Betriebsmittel  $r$

$a_{ij}$  siehe DEF. 2.2.

$m$

$1: b_{i,r} + a_{ij} k_{j,r} \quad j \in N, r \in P$

$i=1$



Diese  $(n \cdot p)$  Ungleichungen sorgen dafür, daß der Bedarf der Tasks an Betriebsmitteln (wie z.B. Prozessorleistung und Speicher) kleiner ist als die jeweilige Kapazität des Rechners  $j$  an Betriebsmittel  $r$ .

#### 2.3.1.2.2. Gleichmäßige Auslastung

Ein wichtiges Kriterium für die Verteilung (neben der Optimierung der IPC-Kosten) ist die relative Gleichauslastung der beteiligten Prozessoren (vergleiche Kapitel 8). Die Lastverteilung wird als Randbedingung in die Allokation einbezogen. Gefordert wird, daß alle Prozessoren in ihrer Auslastung durch die zugeteilten Tasks jeweils maximal um einen vorgegebenen Betrag abweichen. Der Betrag der Abweichung  $\delta u$  kann vom Benutzer vorgegeben werden oder fest im Allokationssystem vereinbart sein.

Sei:

$u_j$  die Auslastung von Prozessor  $j$ ,  
 $\bar{u}$  die mittlere Auslastung eines Prozessors

dann gilt (DEF. : 2.8.) :

$$|u_j - \bar{u}| \leq \delta u \quad j \in N \quad \text{mit}$$

$$\bar{u} = \frac{1}{n} \sum_{k=1}^n u_k$$

#### 2.3.1.2.3. Benutzerdefinierte Einschränkungen für Tasks

In der vorliegenden Arbeit sollen neben den erwähnten Randbedingungen Eingriffe in die Task-Zuordnungen erlaubt werden. Zur Erklärung dieser Einschränkungen sei auf das Kapitel 8 verwiesen.

#### Task-Task-Zuordnung

##### - Verbot

Für bestimmte Tasks soll eine gemeinsame Zuordnung auf den selben Prozessor verboten werden.

Das Verbot der Zuordnung für zwei Tasks  $i, j \in M$  zu einem gemeinsamen Prozessor wird folgendermaßen dargestellt (DEF. : 2.9.) :

$$a_{il} * a_{jl} = 0 \quad \text{für alle } l \in N$$

Damit ist sichergestellt, daß die beiden Prozesse  $i, j$  nicht dem gleichen Prozessor zugeteilt werden.

#### - Gebot

Falls zwei Tasks  $i, j \in M$  gemeinsam dem selben Prozessor zugewiesen werden sollen, wird folgende Definition verwendet (DEF. : 2.10.) :

$$\begin{aligned} a_{iu} * a_{jk} &= 0 && \text{für alle } l, k \in N \text{ mit } l \neq k \\ a_{ik} * a_{jk} &= 1 && \text{für wenigstens ein } k \end{aligned}$$

Mit dieser Definition können die Prozesse  $i$  und  $j$  nur gemeinsam einem Prozessor zugewiesen werden.

### Task-Prozessor-Zuordnung

#### - Verbot

Falls eine Task einem bestimmten Prozessor nicht zugeordnet werden darf, kann dies folgendermaßen definiert werden (DEF.: 2.11.):

$$a_{ul} = 0 \quad \text{falls } l \text{ der verbotene Prozessor ist}$$

-

Falls eine Task einem bestimmten Prozessor fest zugeordnet werden soll, wird folgende Definition angewendet (DEF. : 2.12.) :

$$\begin{aligned} a_{il} &= 1 && \text{falls } l \text{ der ausgesuchte Prozessor ist} \\ a_{ik} &= 0 && \text{falls } k \text{ nicht der gesuchte Prozessor ist} \end{aligned}$$

### 2.3.2. Definition des Allokationsproblems in der vorliegende Arbeit:

Unter Allokation soll die Verteilung von Tasks auf Prozessoren bei Realzeitsystemen (unter Berücksichtigung der Ressourcen) verstanden werden. Die Verteilung soll so erfolgen, daß sie den Vorstellungen des Realzeitsystem-Entwicklers entspricht.

Um dieser Definition gerecht werden zu können, sollen folgende Forderungen erfüllt werden:

- Die Tasks werden auf Prozessoren verteilt, an die periphere Geräte angeschlossen sein können. Es kann sich dabei um ein homogenes oder ein heterogenes System von Prozessoren handeln.
- Die Kommunikationskosten zwischen den Prozessoren sollen möglichst gering sein.
- Die Auslastung der beteiligten Prozessoren soll möglichst gleichmäßig sein.
- Die Laufzeiten der Tasks auf den Prozessoren sollen möglichst gleichmäßig sein.
- Die Netzwerktopologie soll berücksichtigt werden.
- Die Eingabe an das Allokationssystem soll einfach zu handhaben und übersichtlich sein (Eingabe auf Spezifikationsniveau).

## 2.4. Dynamische/statische Allokation von Tasks

Nach der Gültigkeitsdauer der festgelegten Allokationsvorschrift wird zwischen dynamischer und statischer Allokation unterschieden.

### - Dynamische Allokation

Die dynamische Allokation erlaubt es, die Zuordnung während des Betriebes einer Anlage zu verändern. Anhand aktueller Belastungsdaten kann das Realzeitsystem der jeweiligen Lastsituation angepaßt werden. Dies wird dadurch erreicht, daß Tasks während der Laufzeit auf andere Rechner verlegt werden, um so das Ziel der dynamischen Allokation, ein Lastgleichgewicht abhängig vom jeweiligen Zustand des Gesamtsystems, zu erreichen. Dabei spielt der Kommunikationsaufwand zwischen den Prozessoren nur eine untergeordnete Rolle. Während der Laufzeit müssen bei der dynamischen Allokation Verwaltungsarbeiten (Messen der aktuellen Last, Verschieben ausgewählter Tasks, Führen von Listen mit Information über den Systemzustand) vorgenommen werden, die den Durchsatz des Gesamtsystems belasten.

### - Statische Allokation

Bei der statischen Allokation ist die einmal getroffene Entscheidung endgültig, d. h., die Tasks werden auf die Prozessoren verteilt, wo sie während der gesamten Laufzeit bleiben. Diese Verfahren werden deshalb auch als deterministisch bezeichnet.

Da in dieser Arbeit nicht die Betriebsphase eines Realzeitsystems im Vordergrund stehen soll, sondern die Erstellung einer "optimalen" Allokation für das vorgelegte Prozeßsystem, wird das statische Verfahren betrachtet.

## 2.5. Abgrenzung zu technisch/wissenschaftlichen Berechnungen

Parallele Programme werden im technisch/wissenschaftlichen Bereich eingesetzt, um Programmstrukturen (z.B. Vektoroperationen) simultan zu

verarbeiten. Dazu stehen Multiprozessorsysteme (z.B. /HÄND84/) mit Programmiersprachen zur Verfügung, die eine vom Benutzer vorgegebene (parallele) Struktur bearbeiten (MIMP = Multiple Instructions on Multiple frocessors; SIMP = Simple Instructions on Multiple frocessors). Die Vektoraddition

$$a + b$$

( wobei gilt : (  $A = a^1, a^2, \dots, a^n$ ;  $B = b^1, b^2, \dots, b^n$  )), wird folgendermaßen bearbeitet:

Prozessor 1	Prozessor 2	....	Prozessor n
$a^1 + b^1$	$a^2 + b^2$	....	$a^n + b^n$

Die Addition des Vektors A mit dem Vektor B geschieht hier elementweise und gleichzeitig auf allen Prozessoren. Dieser Bereich verteilter Anwendungen ist nicht Bestandteil der vorliegenden Arbeit, die sich mit Realzeitaufgaben unter Verwendung des Konzepts der "Task" als Strukturierungseinheit befaßt. Im Gegensatz zu technisch/wissenschaftlichen Berechnungen, bei denen genau bekannt ist, wann eine bestimmte Operation auf mehrere Prozessoren gleichzeitig verteilt wird, hängt bei der Realzeitverarbeitung die Ausführung einer Operation vom Zustand des Gesamtsystems, vom Benutzer und dem technischen Prozeß ab.

## 2.6. Abgrenzung gegenüber Scheduling

Ein Schedule ist nach /ECKE77/ "im wesentlichen eine Reihung der Tasks für deren Bearbeitung auf den Prozessoren". Dabei gilt es, die zeitliche Reihenfolge der Tasks<sup>5</sup> so festzulegen, daß die gesamte Bearbeitungszeit des Systems minimiert wird. Eine Task im Sinne des Scheduling ist eine zusammenhängende Aufgabe, der Betriebsmittel zugeteilt werden müssen, die sie

---

<sup>5</sup> Eine Task ist beim Scheduling ein abgeschlossener Rechenauftrag.

mit anderen Tasks teilt (z.B Prozessoren, Speicher...). Falls mehrere Prozesse gleichzeitig Zugriff auf ein bestimmtes Betriebsmittel benötigen, muß eine Entscheidung darüber getroffen werden, in welcher Reihenfolge das Betriebsmittel den Prozessen zugeteilt wird. Bei Mehrrechnersystemen muß zusätzlich entschieden werden, wie die Schedules auf die Prozessoren verteilt werden sollen. Je nach Typ des Betriebssystems werden für das Scheduling verschiedene Zielsetzungen betrachtet:

- **BATCH:**                      hoher Durchsatz
- **TIMESHARING:**              schnelle Reaktion auf Eingaben
- **REALZEIT:**                  garantierte Abarbeitungszeiten für bestimmte Algorithmen

Um einen möglichst guten Schedule zu erreichen, können je nach Anforderung verschiedene Bewertungsfunktionen benutzt werden:

- Der Durchsatz des Systems soll maximiert werden.
- Die mittlere Wartezeit für einen Rechenauftrag soll minimal sein.
- Für die Aufträge sind Fertigstellungstermine einzuhalten.
- Falls die Rechenaufträge Termine überschreiten, soll die Zahl der terminüberschreitenden Aufträge minimal sein.

Im Gegensatz zum Scheduling, das zeitabhängige Belegungspläne erzeugen soll, muß bei der Allokation ein "räumlicher" Plan erstellt werden, wobei jeder Task für die gesamte Laufzeit genau ein Prozessor zugeordnet wird. Ziel einer "guten" Allokation ist die Optimierung der Systemleistung. Da bei Realzeitsystemen alle Tasks gemeinsam eine Aufgabe bearbeiten, müssen die Tasks so verteilt werden, daß der Kommunikationsaufwand zwischen den beteiligten Prozessoren reduziert wird und die Rechnerknoten gleichmäßig ausgelastet sind.

## **Kapitel 3**

### **Einbindung in das bestehende Umfeld**

#### **3. Einbindung in das bestehende Umfeld**

Im Rahmen eines von der DFG (Deutsche Forschungsgemeinschaft) geförderten Projektes wurde an der Universität Erlangen-Nürnberg eine Programmierungsumgebung für verteilte Realzeitsysteme entwickelt /HOLL89/. Diese Umgebung basiert auf der Spezifikationsmethode PASS (parallel Activities Specification Scheme), mit der die Kommunikation und Synchronisation zwischen verteilten Tasks dargestellt werden können.

##### **3.1. Die Spezifikationsmethode PASS**

Im Rahmen einer Dissertation /FLEI84/ wurde am RRZE (Regionales Rechen-Zentrum Erlangen) die Spezifikationsmethode PASS konzipiert, die den Entwerfer eines verteilten Prozeßautomatisierungssystems unterstützt bei

- der Zerlegung des Automatisierungsproblems in Komponenten,

- der Bestimmung der Nachrichten, die zwischen den einzelnen Prozessen<sup>6</sup> ausgetauscht werden und
- der Festlegung der Eigenschaften der Strukturierungseinheiten.

In PASS werden als Strukturierungseinheiten Prozesse, Prozeßgruppen und Prozeßbündel verwendet, wobei der Unterschied zwischen den Strukturierungseinheiten im Kommunikationsverhalten besteht. Die einzelnen Strukturierungseinheiten können untereinander nur durch Botschaften kommunizieren bzw. sich synchronisieren. Innerhalb von Prozeßgruppen und -bündeln kann zusätzlich über gemeinsame Daten kommuniziert werden. Diese Einheiten müssen somit über einen gemeinsamen Speicher verfügen. Der Unterschied zwischen Prozeßgruppen und Prozeßbündeln besteht darin, daß die einzelnen Prozesse einer Prozeßgruppe von "außen" nicht sichtbar sind, während die Prozesse in Prozeßbündel von anderen Strukturierungseinheiten direkt ansprechbar sind.

Eine PASS-Spezifikation besteht aus vier Teilen:

- In der Kommunikationsstruktur wird die Netzwerktopologie beschrieben, gleichzeitig werden alle Prozesse mit ihren Kommunikationsbeziehungen dargestellt.
- Die Ablaufsteuerung legt die möglichen Aktionsfolgen fest, wobei zwischen internen Aktionen und Kommunikationsaktionen unterschieden wird.
- In der Kommunikationsmaschine wird die Art des Botschaftsaustausches festgelegt.
- Die Benutzermaschine beschreibt die Daten und die Wirkung, die interne Berechnungen auslösen.

Die Spezifikationsmethode PASS ist Grundlage für weitere Arbeiten (vgl. /ANDR88/, /Krag85/). Da die vorliegende Arbeit dieses System erweitern soll, liegt es nahe, die Strukturierungseinheiten von PASS für die Darstellung eines Verteilungsproblems zu verwenden.

Für das Allokationssystem wird die bestehende Kommunikationsstruktur verwendet, da das Kommunikationsverhalten der Tasks aus dieser Struktur

---

<sup>6</sup> Abweichend von der Bezeichnung "Task" wird bei der Beschreibung von PASS der Begriff Prozeß verwendet.



eindeutig hervorgeht. Um den Anforderungen der Allokation zu genügen, mußte die Beschriftung der Kommunikationsstruktur erweitert werden. In den Knoten der Kommunikationsstruktur werden (für die Allokation) der Speicherbedarf und der Laufzeitbedarf der Tasks und ggf. Zuordnungseinschränkungen angegeben. Die Kantenbeschriftung des Diagramms wird um die Kommunikationskosten erweitert<sup>7</sup>. Zusätzlich zu den vorhandenen Strukturen wird für das Allokationssystem eine Hardwarebeschreibung eingeführt.

### 3.2. Umsetzung der graphischen Eingabe

Um die graphischen Symbole auswerten und als Eingabe für das Allokationssystem verwenden zu können, wird ein Teil eines Programmsystems verwendet, das in /Krag86/ entwickelt wurde. Bei diesem System werden die Kommunikationszustände der Ablaufsteuerung und die Kommunikationsstruktur analysiert, und anhand der Ergebnisse wird Code für eine Programmiersprache (PEARL) erzeugt.

Bei der Umsetzung der Kommunikationsstruktur in PEARL-Code wird die graphische Information der Struktur in einen Zwischencode übersetzt. Aus diesem Code werden (nach semantischer Analyse) Eingabeelemente für das Allokationssystem erstellt (siehe Kapitel 9).

Zusätzlich zur Kommunikationsstruktur muß der Benutzer des Allokationssystems die Hardwarestruktur des Realzeitsystems in Form einer Graphik erstellen, die ebenfalls automatisch ausgewertet wird. Für diese Struktur wurde das Programm zur Zwischencodeerzeugung angepaßt, so daß nun eine einheitliche Schnittstelle für alle Strukturen existiert.

---

<sup>7</sup> Zur weiteren Erläuterung der Diagramme sei auf den Anhang I verwiesen.

### 3.3. Bestimmen der Kommunikationskosten

Ein Problem bei der Allokation ist die Berechnung der Kommunikationskosten. Solche Kosten können nur im Betrieb eines Realzeitsystems (abhängig vom Zeitverhalten des zu steuernden Prozesses) exakt "gemessen" werden.

In /ANDR88/ wurde ein System zur Validierung der Ablaufsteuerung der Spezifikationsmethode PASS erstellt. Mit diesem System ist es möglich, das Laufzeitverhalten des Realzeitsystems zu simulieren, und daraus die Häufigkeit der zwischen den Tasks ausgetauschten Botschaften näherungsweise zu bestimmen. Aus Anzahl und Größe der betrachteten Botschaften können dann die Kommunikationskosten abgeleitet werden.

### 3.4. Zusammenfassung

Die Programmierungsumgebung für verteilte Realzeitanwendungen wird mit der Allokation um ein wichtiges Element erweitert. Eine (Realzeit-) Spezifikation kann nun folgenden Prozeduren unterzogen werden:

- Validierung der Spezifikation.
- Automatische Codeerzeugung.
- Automatische Verteilung auf die Hardware.

In Abbildung 3.1. sind die einzelnen Komponenten und die Übergänge dargestellt.

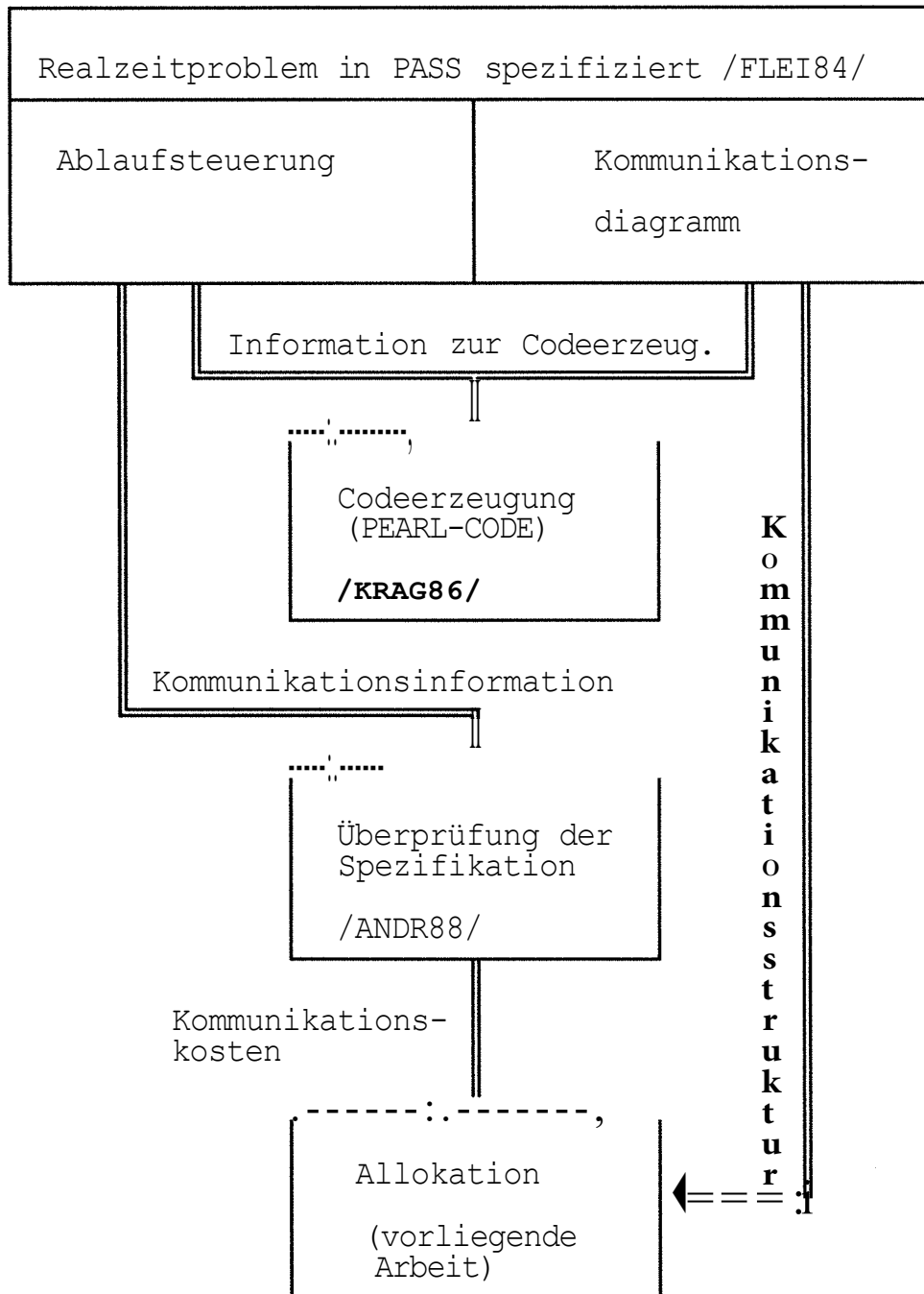


Abb. : 3.1. Einbindung in das bestehende Umfeld

## **Kapitel 4**

### **Dem Allokationsproblem verwandte Probleme**

#### **4. Dem Allokationsproblem verwandte Probleme**

Im Bereich des Operations Research (vgl. /MEYE79/) existieren Probleme (komplexe Planungsprobleme), die dem der Allokation ähnlich sind. Diese Probleme werden meist mit Methoden der linearen Programmierung oder der Netzplantechnik gelöst. Im folgenden werden zwei derartige Probleme betrachtet, die Parallelen mit dem Allokationsproblem aufweisen. Für diese Probleme existieren eine Reihe von Lösungsansätze, die in Kapitel 5 erläutert werden.

##### **4.1. Das quadratische Zuordnungsproblem**

Das Quadratische Zuordnungsproblem /MUEL70/ ist mit dem Allokationsproblem (mit bestimmten Einschränkungen) identisch :

Gegeben sind  $n$  Orte (Fabriken), die bestimmte Entfernungen voneinander haben. Diesen Orten sollen  $n$  Elemente (z.B. Maschinen) zugeordnet werden. Gesucht wird eine Zuordnung, bei der die Summe der Bearbeitungswege eines Werkstückes zwischen den Maschinen minimal ist.

$$z = \sum_{j=1}^n \sum_{i=1}^n \sum_{k=1}^n \sum_{l=1}^n d_{ij} f_{kl} x_{ik} x_{jl}$$

Matrix:  $X$  gibt an, welches Element welchem Ort zugeordnet wird

$D$  bezeichnet die Entfernungen zwischen den Orten

$F$  stellt die Transportbeziehungen zwischen zwei Orten dar

Ferner gelten die Restriktionen, durch die sichergestellt wird, daß jedes Element nur einem Ort und jeder Ort nur einem Element zugeordnet wird:

$$\text{a) } \sum_{i=1}^n x_{ik} = 1 \quad \text{für } k = 1, 2, \dots, n$$

$$\text{b) } \sum_{k=1}^n x_{ik} = 1 \quad \text{für } i = 1, 2, \dots, n$$

$$\text{c) } x_{ik} = 0/1 \quad \text{für } i = 1, 2, \dots, n \text{ und } k = 1, 2, \dots, n$$

Beim quadratischen Zuordnungsproblem wird (wie beim Allokationssystem) eine Zuordnung eines Elementes  $i$  (Task) zu einem Ort  $k$  (Prozessor) dadurch angezeigt, daß (Bedingung c)  $x_{ik} = 1$  ist (ansonsten ist  $x_{ij} = 0$ ). Die Zielfunktion besagt in diesem Fall, daß die Kosten  $d_{ij} f_{kl}$  entstehen, wenn das  $k$ -te Element dem Ort  $i$  und das  $l$ -te Element dem Ort  $j$  zugeordnet werden.

Das quadratische Zuordnungsproblem kann als ein Spezialfall des Allokationsproblems betrachtet werden. Die Zielfunktion für das quadratische Zuordnungsproblem ist mit der IPC-Kostenfunktion identisch, wenn dort  $m = n$  gesetzt wird. Das bedeutet, daß beim Zuordnungsproblem  $n$  Tasks auf  $n$  Prozessoren verteilt werden sollen, unter dem Gesichtspunkt, daß die Verbindungskosten minimal werden. Die Komplexität des quadratischen Zuordnungsproblems ist

geringer als die des Allokationsproblems, da folgende Einschränkungen gemacht werden:

- Die Anzahl der Orte (Tasks) ist gleich der Anzahl der Elemente (Prozessoren).
- Jedes Element (Task) wird genau einem Ort (Prozessor) zugeordnet.
- Beim quadratischen Zuordnungsproblem werden keine Randbedingungen berücksichtigt (Peripherie, Speicherbedarf beim Allokationsproblem).

Die Komplexität des quadratischen Zuordnungsproblems wird in /MUEL70/ mit  $O(n!)$  angegeben. Als Lösungsverfahren für das quadratische Zuordnungsproblem werden heuristische Verfahren diskutiert, da optimale Verfahren an der Komplexität des Problems scheitern.

Die Komplexität des Allokationsproblems ist mit  $O(n^m)$  (wobei  $n$  die Anzahl der Tasks und  $m$  die Anzahl der Prozessoren ist) größer als die des quadratischen Zuordnungsproblems, da  $n^m > n!$  ist (weil  $m \gg n$  gefordert ist).

#### 4.2. Das Problem des Handlungsreisenden

Beim Problem des Handlungsreisenden geht es darum, die optimale Reiseroute eines Handlungsreisenden zu finden, was in diesem Fall bedeutet, daß der Handlungsreisende die Orte in der Reihenfolge besucht, die seinen Reiseweg minimieren. Damit ist diese Aufgabe ein typischer Vertreter der Reihenfolgeprobleme. Der minimale Weg berechnet sich nach folgender Formel:

$$z = \sum_{i=1}^m \sum_{j=1}^m d_{ij} x_{ij}$$

Matrix:  $X$  gibt an, ob ein Teilweg  $i$ - $j$  in der gesamten Reiseroute vorhanden ist.

$D$  bezeichnet die Kosten, die für einen Weg von  $i$  nach  $j$  entstehen.

Jeder Ort muß vom Handlungsreisenden genau einmal besucht werden, somit ergibt sich für jeden besuchten Ort  $j$  also folgende Einschränkung:

$$\sum_i x_{ij} = 1 \quad \text{für } j = 1, 2, \dots, n$$

Außerdem muß von jedem Ort einmal zu einem anderen gefahren werden, somit gilt also für jeden Ort  $i$ :

$$\sum_j x_{ij} = 1 \quad \text{für } i = 1, 2, \dots, n$$

Beim Problem des Handlungsreisenden sind Start- und der Zielort vorgegeben, deshalb gibt es für  $n$  Orte  $(n-1)!$  mögliche Rundreisen.

Am Problem des Handlungsreisenden können die Prinzipien aufgezeigt werden, die bei Problemen dieser Komplexität entstehen. Anhand des "Handlungsreisenden" wurden in zahlreichen Arbeiten (in /MUEL70/ sind diese Verfahren dargestellt), verschiedene optimale und heuristische Verfahren entwickelt, die auch für das Allokationsproblem betrachtet wurden. Im nächsten Kapitel werden einige dieser Lösungsverfahren dargestellt.

## Kapitel 5

### Lösungsverfahren

#### S. Lösungsverfahren

##### 5.1. NP-vollständige Probleme

Grundsätzlich gibt es eine sehr einfache Möglichkeit, das Allokationsproblem zu lösen: Man zähle alle möglichen Kombinationen auf und wähle daraus dann diejenige aus, die die geringsten Kommunikationskosten verursacht. Eine solche Liste enthält dann  $n^m$  verschiedene Kombinationen ( $n$  ist die Anzahl der Tasks und  $m$  die Anzahl der Prozessoren), der Berechnungsaufwand steigt also exponentiell. Das Allokationsproblem gehört in den Bereich solcher "NP-vollständiger" Probleme. Über die Komplexität solcher Probleme wird in /AH074/ folgendes festgestellt:

"There is general agreement that if a problem cannot be solved in less than exponential time, then the problem be considered completely intractable. The implication of this "rating scheme" is that problems having polynomial-time-bounded algorithms are tractable".

Zum besseren Verständnis des Allokationsproblems vor dem Hintergrund der NP-vollständigen Probleme mögen folgende Definitionen dienen:

Beim "Messen" der Komplexität eines Algorithmus verwendet /HOR081/ die



Länge der Eingabe als Parameter:

DEF. 5.1. : Ein Algorithmus  $A$  hat eine polynomiale Komplexität, wenn es ein Polynom  $p(n)$  gibt, so daß die Rechenzeit von  $A$  gleich  $O(p(n))$  für jede Eingabe der Länge  $n$  ist.

DEF. 5.2. :  $P$  ist die Menge aller Entscheidungsprobleme, die von einem deterministischen Algorithmus in polynomialer Zeit gelöst werden können.

$NP$  ist die Menge aller Entscheidungsprobleme, die von einem nichtdeterministischen Algorithmus in polynomialer Zeit gelöst werden können.

Als Folgerung aus dieser Definition ergibt sich, daß  $P$  eine Teilmenge von  $NP$  ist, da deterministische Algorithmen ein Spezialfall der nichtdeterministischen sind. Ein bislang ungelöstes Problem ist die Frage, ob  $P$  gleich  $NP$  ist.

Um die Problemklasse NP-vollständig zu definieren, benötigt man den Begriff der Reduzierbarkeit

DEF. 5.3.: Gegeben seien zwei NP-vollständige Probleme  $L_1$  und  $L_2$ .

$L_1$  ist genau dann auf  $L_2$  reduzierbar, wenn es einen polynomial begrenzten Algorithmus gibt, der die Lösungen von  $L_1$  auf  $L_2$  transformiert.

DEF. 5.4. : Ein Problem  $L$  ist NP-vollständig, wenn  $L$  in  $NP$  liegt und jedes andere Problem in  $NP$  darauf polynomial reduzierbar ist.

In /HOR081/ wird gezeigt, daß das quadratische Zuordnungsproblem NP-vollständig ist. Dieses Problem ist (wie in Kapitel 4 ausgeführt) mit dem Allokationsproblem identisch, wenn die Anzahl der Prozessoren mit der Anzahl der Tasks identisch ist und keine Randbedingungen berücksichtigt werden. Das quadratische Zuordnungsproblem kann also als Spezialfall des Allokationsproblems angesehen werden. Somit ist auch das Allokationsproblem NP-vollständig.

Vom quadratischen Zuordnungsproblem ist bekannt, daß es darüberhinaus ein sogenanntes  $\epsilon$  - Approximationsproblem ist, das für alle  $\epsilon > 0$  NP-vollständig ist. Das heißt, daß es für dieses Problem nicht einmal möglich ist, eine Näherungslösung zu finden, die in keinem Fall um mehr als einen gegebenen Prozentsatz ( $\epsilon$ ) von der optimalen Zuordnung abweicht. Um diese Einschränkung fassen zu können, wird der Begriff des  $\epsilon$  - Approximationsproblem definiert:

Es sei  $A$  ein Algorithmus, der zu jeder Problemstellung  $I$  eines Problems  $P$  eine möglichst gute Lösung erzeugt.  $F^*(I)$  sei der Wert einer optimalen Lösung für  $I$ .  $F'(I)$  sei der Wert der von  $A$  erzeugten möglichen Lösung.

DEF. 5.5. : Ein Algorithmus  $A$  ist genau dann ein absoluter Algorithmus für das Problem  $P$ , wenn für jede Problemstellung  $I$  zu  $P$  gilt:

$$| F^*(I) - F'(I) | \leq k, \text{ wobei } k \text{ eine Konstante ist.}$$

DEF. 5.6. : Ein Algorithmus  $A$  ist genau dann ein  $f(n)$  approximativer Algorithmus, wenn für jede Problemstellung  $I$  der Größe  $n$  gilt:

$$| F^*(I) - F'(I) | / F^*(I) \leq f(n). \text{ Wobei angenommen wird, daß } F^*(I) > 0.$$

DEF. 5.7.: Ein  $\epsilon$  - approximativer Algorithmus ist ein  $f(n)$  - approximativer Algorithmus, für den  $f(n) \leq \epsilon$  ist, wobei  $\epsilon$  eine Konstante ist.

Das Allokationsproblem ist -wie das quadratische Zuordnungsproblem- NP-vollständig und es besteht keine Aussicht, eine Näherungslösung zu finden, die von der optimalen Lösung um weniger als einen vorgegebenen Wert abweicht. Da also NP-vollständige Probleme nicht in vernünftiger (polynomialer) Zeit mit optimalen Algorithmen gelöst werden können, werden auf sie meist suboptimale (heuristische) Strategien angewendet. Für kleinere Probleme (beim Allokationsproblem beispielsweise für Systeme, die nur zwei Prozessoren besitzen) können optimale Verfahren aber in vernünftiger Zeit zu Lösungen führen.

## 5.2. Optimale Verfahren

Bei den optimalen Verfahren wird die Lösung eines Problems dadurch erreicht, daß alle Lösungsmöglichkeiten generiert werden, um die optimale Lösung zu finden. Optimale Verfahren eignen sich deshalb nur für "kleine" Probleme.

### S.2.1. Mathematische Verfahren

Bei den mathematischen Verfahren wird die Lösung eines Problems durch exakte Funktionen erreicht. Man versucht, das gegebene Problem mit Hilfe einer Funktion zu beschreiben. Die Lösung dieser Funktion ist zugleich die Lösung des Problems.

#### Lineare Programmierung

Bei der linearen Programmierung wird eine Funktion der Form:

$$Z(x_1, x_2, \dots, x_n) = f_1(x_1) + f_2(x_2) + \dots + f_n(x_n) \rightarrow \max !$$

verwendet. Das Maximum (bzw. Minimum) dieser Funktion ist die Lösung des Allokationsproblems, wenn für die einzelnen Variablen Funktionen  $f_i$  eingesetzt werden, die das Problem repräsentieren. Die Randbedingungen (im Falle der Allokation beispielsweise die Betriebsmittellanzahl) werden durch Ungleichungen in den Berechnungsvorgang mit einbezogen:

$$g_i(x_1, x_2, \dots, x_n) \leq b_i$$

Als Beispiel zur Linearen Programmierung soll das 0/1 - Verfahren dargestellt werden, für welches eine große Zahl von Lösungsmethoden entwickelt wurde (/WUND79/, /PETR76/). Beim 0/1-Verfahren wird eine ähnliche Funktion wie die oben aufgeführte benutzt, wobei alle Variablen  $x_i$  nur die Werte 0 oder 1 annehmen dürfen.

Das System von Funktionen sieht folgendermaßen aus:

$$\max_{\substack{j \\ i=1}}^n (1: r_i \cdot x_{ij}) \rightarrow \max!$$

$$x \in \{0,1\}; \quad r_i \geq 0; \quad 1 \leq i \leq n; \quad 1 \leq j \leq m;$$

Wendet man dieses System nun auf das Allokationsproblem an, so gelten folgende Bedingungen:

Die Variable  $x_{ij}$  nimmt den Wert 1 an, wenn die Task  $i$  dem Prozessor  $j$  zugeteilt wird. Die Rechenzeit (mit  $r_i$  bezeichnet) sei die Zeit, in der ein Prozessor mit der Task  $i$  beschäftigt ist.

Die Lösung des Systems (das Maximum) ist die Lösung des repräsentierten Allokationsproblems. Diese Lösung des Allokationsproblems ist allerdings für die Praxis ungeeignet, da folgende Einschränkungen zugrunde liegen:

- es wird ein homogenes Rechnersystem benutzt
- es werden keine Ressourcen berücksichtigt
- es werden keine Kommunikationskosten betrachtet.

Mathematische Verfahren sind zur Lösung von NP-vollständigen Problemen nur in speziellen Fällen geeignet. Für das Allokationsproblem sind solche Verfahren für Systeme mit maximal 3 Prozessoren unter gleichzeitiger Vernachlässigung der Randbedingungen geeignet.

### 5.2.2. Enumeration

Bei der Lösung des Allokationsproblems mit Methoden der Enumeration gilt (in Bezug auf die Größe des Allokationsproblems) das gleiche wie bei den mathematischen Verfahren. Enumerationsverfahren sind jedoch relativ einfach zu implementieren.

Bei der Vollenumeration werden alle möglichen Kombinationen der Reihe nach aufgezählt. Für jedes dieser Listenelemente wird geprüft, ob der

errechnete Wert besser als der aktuell gültige Optimalwert ist. Im positiven Fall wird der neue Wert als Optimalwert übernommen, andernfalls wird er verworfen. Die Aufzählung der möglichen Werte erfolgt in Form eines Baumes (Entscheidungsbaum). Bei jedem Schritt von der Wurzel zu den Ästen dieses Baumes wird jedem Prozessor eine weitere Task zugeordnet und diese Schicht bewertet.

Der Entscheidungsbaum besteht aus

$$\begin{matrix} n \\ L \quad m^i \\ i=0 \end{matrix}$$

Knoten (n ist die Anzahl der Tasks; m die Anzahl der Prozessoren).

Um das Verfahren der Vollenumeration, bei dem alle Kombinationen untersucht werden, etwas abzukürzen, wurden Methoden entwickelt, mittels derer man Teile des Entscheidungsbaumes "abschneiden" kann. In jedem Knoten des Entscheidungsbaumes wird dabei geprüft, ob die Suche an diesem Knoten enden soll. Falls man sich für den Abbruch entscheidet, wird der Ast ab diesem Knoten nicht mehr weiter durchsucht. Somit wird - abhängig vom Abbruchkriterium - ab einer bestimmten Tiefe des Baumes das Enumerationsverfahren abgebrochen. Das Abbruchkriterium muß in geeigneter Weise gewählt werden, denn ein Abbruch der Suche an einer falschen Stelle verhindert das Finden des Optimums. Die Entscheidungsbaumverfahren sind also verkürzte Enumerationsverfahren.

Aus der Literatur (z.B. /MUEL69/, /GARF72/, u.a.) sind verschiedene Verfahren bekannt, die durch geeignete Methoden den Aufbau des Entscheidungsbaumes verkürzen.

#### 5.2.2.1. Branch and Exclude

Die Branch and Exclude-Methoden gehen prinzipiell von der Generierung aller Lösungen aus, brechen aber nach vorgegebenen Kriterien die Bearbeitung eines Astes eventuell vorzeitig ab und verfolgen den nächsten (in der Hierarchie des Suchbaumes) zu bearbeitenden Ast weiter. Durch die feste Bestimmung des nächsten Astes sind diese Methoden nur geringfügig besser als die Vollenumeration: Auch diese Methoden sind NP-vollständig.

#### 5.2.2.2. Branch and Bound

Im Gegensatz zu Branch and Exclude wird bei Branch-and-Bound-Algorithmen nach Abbruch eines Astes nicht der nächstfolgende, sondern der geeignetste Ast bestimmt. Um gute Abbruchkriterien zu finden, bedienen sich diese Algorithmen folgender Schritte:

- Suchstrategie

Hier wird eine Untermenge gesucht, die als nächste betrachtet werden soll. Zum Auffinden des besten Nachfolgers im Suchbaum kann zwischen zwei bekannten Verfahren gewählt werden:

- **Breitensuche:** Hier wird der Knoten als Nachfolger ausgewählt, der in einer Ebene (des Suchbaumes) den besten unteren Grenzwert besitzt. (Die Grenzwerte werden durch eine Kostenfunktion berechnet).
- **Tiefensuche:** Derjenige Knoten wird ausgesucht, der in Bezug auf die Kostenfunktion der beste Nachfolgeknoten des jeweils betrachteten ist.

- Verzweigung

Die Verzweigung wird benutzt, um den Enumerationsbaum aufzubauen und die Nachfolger und Vorgänger der einzelnen Knoten festzulegen.

- Berechnung der Schranken

Die Schranken geben dem Algorithmus an, an welcher Stelle die Äste abgeschnitten werden dürfen. Durch große Schranken werden wenige Äste abgeschnitten, bei kleinen Schranken hingegen werden mehr Äste frühzeitig gekürzt. Die Berechnung der Schranken beeinflusst somit die Qualität des Algorithmus entscheidend.

Diese Algorithmen arbeiten im schlechtesten Fall genauso wie die Vollenumeration. Abhängig von der Qualität der Berechnung geeigneter Schranken, kann wenigstens ein Weg im Baum gefunden werden, der zur optimalen Lösung führt.

Im Falle der Vollenumeration ist der Algorithmus NP-vollständig in Bezug auf

die Zeitkomplexität. In den Fällen, bei denen die richtigen Wege des Suchbaumes durchschritten werden, wächst der Speicherbedarf des Algorithmus exponentiell und ist das Problem somit NP-vollständig in Bezug auf den Speicherbedarf. Die besten Bedingungen für solche Algorithmen liegen zwischen den beiden Extremwerten.

#### S.2.2.3. Dynamische Programmierung

Nach /MUEL69/ ist die dynamische Programmierung das älteste Entscheidungsbaumverfahren. Anders als bei den oben erwähnten Methoden werden bei der dynamischen Programmierung die Äste des Entscheidungsbaumes nicht abgeschnitten, sondern parallel aufgebaut. Dieser Algorithmus kommt deshalb mit einer Zeit aus, die sich aus der Anzahl paralleler Knoten berechnet. Dynamische Programmierung eignet sich vor allem für Suchbäume, die sehr breit sind; d.h. sehr viele parallele Knoten besitzen. Dies ist für das Allokationsproblem genau dann der Fall, wenn viele Tasks auf wenige Prozessoren abgebildet werden.

Wie bei allen bekannten optimalen Methoden können auch bei der dynamischen Programmierung die Randbedingungen nur unzureichend berücksichtigt werden.

### 5.3. Suboptimale (heuristische) Verfahren

In vielen Fällen (vgl. /MUELL69/) arbeiten die Entscheidungsbaumverfahren so langsam, daß der Rechenaufwand durch das Ergebnis nicht mehr gerechtfertigt ist. Bei solchen Problemen muß mit heuristischen Verfahren gearbeitet werden. Diese führen zwar nicht immer zu einem Optimum, häufig aber zu vertretbar guten Lösungen, die dem Optimum sehr nahe kommen.

Suboptimale Strategien sind nach /NEVE79/ definiert als "programs that perform tasks requiring intelligence when performed by human beings". In /STRE75/ wird der Begriff der "heuristischen Lösungsverfahren" folgendermaßen konkretisiert:

- Heuristische Verfahren dienen in erster Linie zur Reduktion des Lösungsaufwandes.
- Heuristische Verfahren sind so strukturiert, daß ein Teil der poten-

tiellen Lösungen nicht gesucht wird.

- Durch die fehlende Lösungsgarantie sind heuristische Verfahren von optimalen Verfahren abgegrenzt.
- Verfahren, bei denen willkürlich Äste des Entscheidungsbaumes abgeschnitten werden, gelten nicht als heuristische Verfahren.

Als wichtige Kriterien für Heuristiken, die bislang Eingang in die Problemlösung fanden, werden in der Literatur genannt:

- Die Gesamtaufgabe wird in mehrere Teilaufgaben zerlegt und über die Einzellösungen wird dann die Lösung angestrebt.
- Aus der Menge der bislang erreichten Lösungszustände wird derjenige Zustand ausgewählt, der der gesuchten Lösung am nächsten kommt.
- Die Suche nach Lösungen wird auf die Nachbarschaft der erreichten "guten" Lösung begrenzt.
- Ein einfaches analoges Problem wird gelöst und die benutzte Vorgehensweise als "Plan" für den Lösungsweg benutzt.
- Der Bereich potentieller Lösungen wird durch zusätzliche Beschränkungen begrenzt.

In vielen Arbeiten (/MUEL69/, HOPP84/...) hat sich die Zerlegung des Lösungsverfahrens in zwei "Teilschritte", ein Eröffnungsverfahren und ein Iterationsverfahren, als adäquates Mittel erwiesen.

### 5.3.1. Eröffnungsverfahren

Mit einem Eröffnungsverfahren werden für ein Problem erste Ausgangslösungen gesucht (vergleiche z.B. Schachprogramme). Von einer guten Ausgangslösung hängt die Qualität der nachfolgenden Phasen stark ab.

Ein häufig benutztes Eröffnungsverfahren sind die "Greedy"-Methoden (siehe /GARF78/). "Greedy"-Methoden betrachten nur einen einzigen Pfad im Entscheidungsbaum und ermitteln dabei in jedem Knoten genau einen Nachfolgerknoten. Wenn eine Entscheidung getroffen ist, wird sie nie wieder rückgängig gemacht, auch wenn sich im späteren Verlauf des Algorithmus eine



solche Entscheidung als fehlerhaft herausstellt. Es besteht also keine Möglichkeit diesen "Fehler" wieder gut zu machen. Durch die Beschränkung auf einen Pfad im Entscheidungsbaum sind die Greedy-Algorithmen allerdings sehr schnell. Zu diesen Methoden zählen die lokalen Planungsregeln aus der Betriebswirtschaftslehre (/MERT78/) und die Prioritätsregeln, nach denen Scheduling von Programmen gelöst wird.

### 5.3.2. Iterationsverfahren

Nachdem einmal eine (gute) Ausgangslösung gefunden ist, wird diese mittels geeigneter Iteration schrittweise verbessert. Ein Beispiel für solche Iterationsverfahren sind die in /HOPP84/ angegebenen Tausch- und Suchverfahren. Bei der Iteration wird stets versucht, die Zielfunktion zu verbessern. Nur solche Schritte werden weiter verfolgt, die eine Verbesserung gegenüber der gegenwärtigen Situation darstellen.

Ein Vertreter der Iterationsverfahren ist das Tauschverfahren von /BOKH81/, bei dem ausgehend von einer bereits existierenden Anfangslösung schrittweise die Lösung ermittelt wird. Es wird eine Nachbarschaftsstruktur definiert: Darunter versteht man eine allgemeine strukturelle Beschreibung aller relevanten Nachbarschaftsbeziehungen zwischen den einzelnen Lösungen des Problems. Aus diesen Nachbarschaftslösungen wird nun diejenige ausgewählt, die den meisten Gewinn bezüglich der Zielfunktion bringt. Der Prozeß der Generierung von Nachbarschaften und Auswählen der besten Lösungen wird solange fortgesetzt, bis sich der Wert der Zielfunktion nicht mehr verbessern läßt.

Dieses Verfahren führt nach /HOPP84/ mit vertretbarem Rechenaufwand zu akzeptablen Lösungen.

### 5.3.3. Beispiel für ein heuristisches Verfahren

Ein heuristisches Verfahren besteht nach /MUEL69/ aus einem Eröffnungsverfahren und verschiedenen Iterationsverfahren, die in einer bestimmten Reihenfolge angewendet werden. Die Umschaltung zwischen den suboptimierenden Verfahren wird unterschiedlich organisiert. Bei einer hierarchischen Gliederung arbeitet man so lange mit dem Verfahren der höchsten Priorität, bis man zu einem Suboptimum gekommen ist. Dann wird das Verfahren der

zweithöchsten Priorität eingesetzt. Wird damit eine Verbesserung erreicht, erfolgt Wiederaufnahme des ersten Verfahrens. Im anderen Fall wird auf das nächstniedrige Verfahren umgeschaltet. Abgeschlossen ist dieser Vorgang erst dann, wenn mit dem letzten Verfahren keine Verbesserung mehr erreicht wird. Neben der hier vorgeschlagenen hierarchischen Organisationsform wird auch eine zyklische Form diskutiert. Die Verfahren hängen aber alle in entscheidendem Maß von der Qualität der eingesetzten Iterationsverfahren ab.

Die Forderungen an heuristische Verfahren aus /HOPP84/ sollen die Diskussion hier abschließen. Demnach sollen heuristische Verfahren folgende Eigenschaften erfüllen:

- Sie sollen in vernünftiger Zeit eine Lösung finden.
- Die Lösungen sollen nahe dem Optimum sein.
- Die Wahrscheinlichkeit, daß eine Lösung weit vom Optimum entfernt ist, sollte gering sein.

#### 5.4. Zusammenfassung

Das NP-vollständige Problem der Verteilung läßt sich nur in sehr speziellen Fällen durch optimale Verfahren lösen, nämlich dann, wenn Systeme mit nicht mehr als drei Prozessoren und ohne Randbedingungen betrachtet werden. Andere Lösungsverfahren sind derzeit nicht bekannt. Für diese Prozessoranzahl stehen Lösungsverfahren zur Verfügung, die in vertretbarer Zeit das Optimum einer Verteilung berechnen können. Für den praktischen Einsatz sind solche Verfahren sicherlich ungeeignet, da die Anzahl der Prozessoren im Bereich der Prozeßautomatisierung erheblich größer ist. Außerdem sind bei Realzeitaufgaben die Randbedingungen von entscheidender Bedeutung, diese können aber bei den optimalen Verfahren nicht angemessen berücksichtigt werden.

Die suboptimalen (heuristischen) Verfahren können NP-vollständige Probleme "beliebiger" Größe in vertretbarer Zeit "lösen", sie liefern allerdings (in den meisten Fällen) keine optimalen Lösungen. Für das Allokationsproblem bedeutet dies, daß zwar Verteilungen gefunden werden, aber keine Möglichkeit zur Überprüfung der Optimalität besteht.

Bei den heuristischen Verfahren wird der Aufbau eines Entscheidungsbaumes

an geeigneten Stellen abgebrochen. Die Qualität der Lösung hängt in entscheidendem Maße davon ab, wie gut die Abbruchskriterien gewählt sind.

## Kapitel 6

### Allokationssysteme

#### 6. Allokationssysteme

Das Problem der Allokation wird (z.B. /EFE82/, STON78/) meist durch Minimierung der Interprozessorkommunikation gelöst.

Das Ziel der Allokation ist es, die Tasks so zu verteilen, daß zwischen den Prozessoren möglichst wenig Kommunikation stattfindet. Zu diesem Zweck muß der gesamte IPC-Aufwand minimiert werden. Dieses Ziel, die IPC-Kosten zu minimieren, steht bei allen bislang existierenden Allokationssystemen im Vordergrund.

In Abhängigkeit von den zugrundeliegenden Hardwaresystemen wird das Allokationsproblem aber unterschiedlich gelöst:

- So führen beispielsweise Ansätze, bei denen ein homogenes Kommunikationsnetz zugrundeliegt, das Allokationsproblem auf ein Partitionierungsproblem zurück.
- Bei inhomogenen Netzen wird meist ein Enumerationsverfahren oder Entscheidungsbaumverfahren favorisiert.
- Vielfach wird auch eine Zerlegung in kleinere Subsysteme vorgeschlagen.

## 6.1. Lösung durch optimale Verfahren

### Mathematische Verfahren

In einer Reihe von Ansätzen wurden die mathematischen Verfahren und insbesondere das 0/1-Verfahren verwendet /HUAN80/, /CHU69/. Diese Methoden zählen alle Ergebnisse auf und vergleichen die aktuelle Lösung mit dem unmittelbaren Vorgänger. Stellt die neueste Lösung eine Verbesserung gegenüber der letzten dar, wird diese als lokales Optimum übernommen; andernfalls bleibt das alte Optimum bestehen.

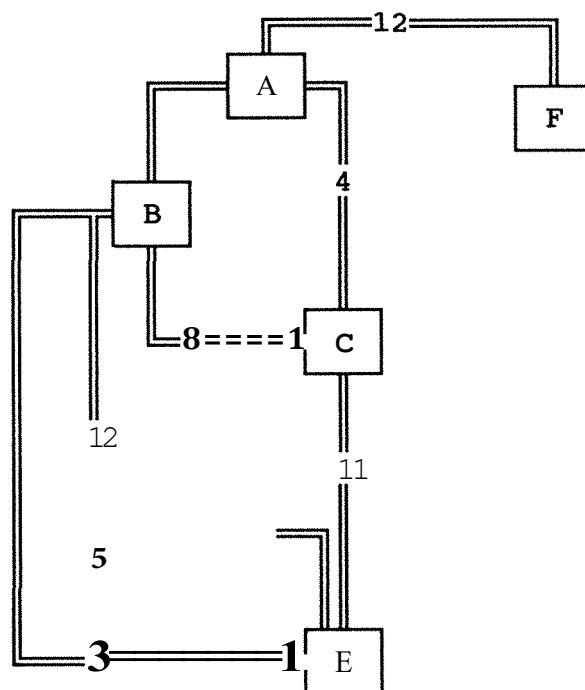
Um das (in der vorliegenden Arbeit entwickelte) Allokationssystem zu testen, wurde ein 0/1-Verfahren /BAUE89/ konstruiert, das die Kommunikationskosten berücksichtigt. Für ein homogenes Rechnersystem (mit maximal 3 Prozessoren) ohne Ressourcen liefert dieses Verfahren optimale Ergebnisse. Für ein Problem mit vier Prozessoren und 16 Tasks wäre eine Rechenzeit von mehreren Tagen notwendig. Zudem ist dieser Ansatz für praktische Fälle ungeeignet, die heterogene Rechnersysteme und beliebige Randbedingungen (Ressourcen) voraussetzen.

Die optimalen Methoden sind insofern alle gleich, als sie die Lösung durch Überprüfung aller möglichen Lösungen finden.

Die optimalen Methoden werden exemplarisch anhand eines graphischen Verfahrens erläutert. Graphische Verfahren eignen sich wegen der übersichtlichen Darstellung besonders gut, um Lösungsmethoden bei der Allokation verständlich darzulegen.

Graphentheoretische Verfahren

Bei dem graphentheoretischen Verfahren von /STON87/ werden die Tasks als Knoten und die Kommunikation zwischen den Tasks als Kanten eines Graphen aufgefaßt. Die Kanten sind mit den Kommunikationskosten beschriftet.



— x == Kommunikationskosten

Abb. : 6.1. Kommunikationsdiagramm

Zur Verteilung der Tasks auf die Prozessoren wird der Graph um die Prozessorknoten erweitert. Alle Taskknoten werden mit allen Prozessorknoten verbunden. Die Kanten, die von den Prozessorknoten zu den Taskknoten führen, werden mit den Kosten (Rechenkosten) beschriftet, die entstehen, wenn die Task vom entsprechenden Prozessor bearbeitet wird.

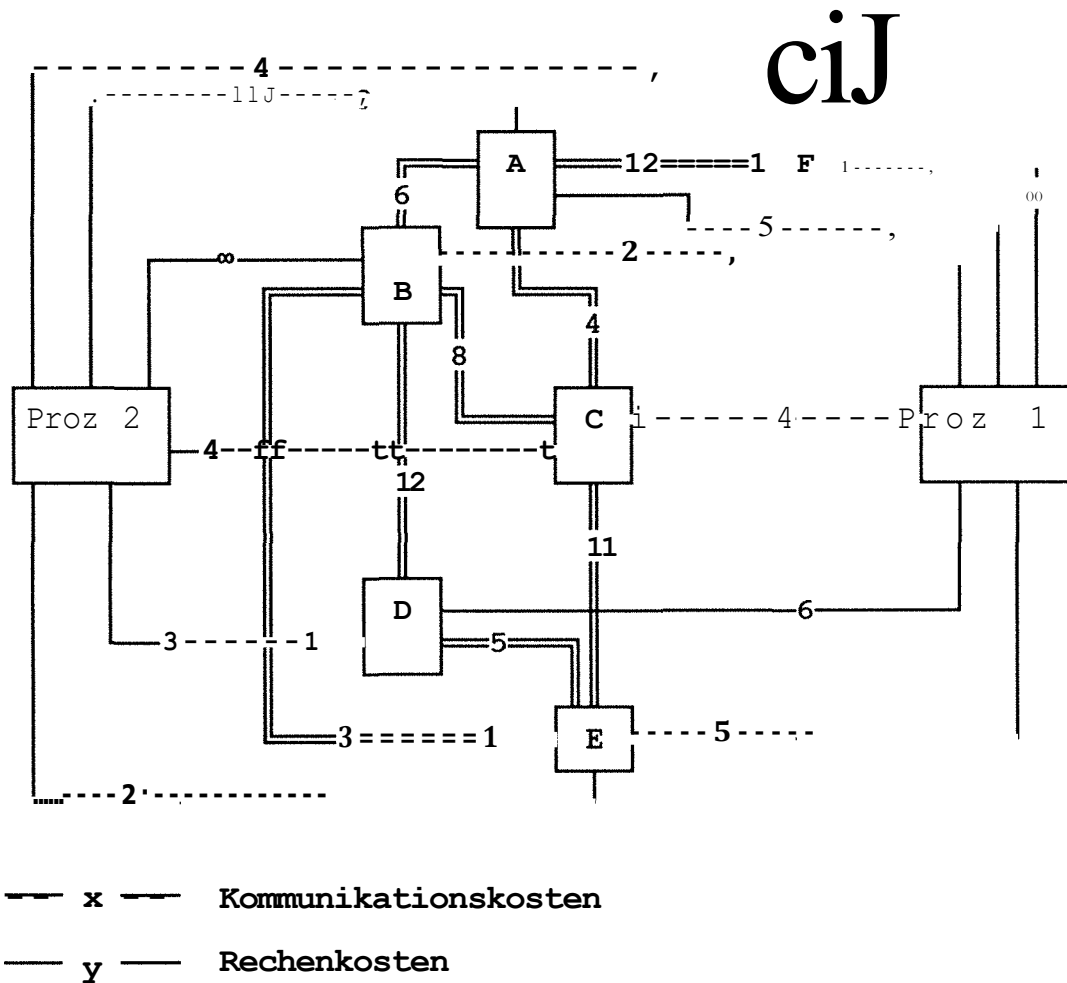


Abb .: 6.2. Kommunikationsdiagramm mit Rechenkosten

Für die Lösung des Allokationsproblems mit Hilfe graphentheoretischer Modelle werden Netzflußtechniken herangezogen.

Der Ford-Fulkerson-Algorithmus löst das Problem des maximalen Flusses zwischen zwei ausgezeichneten Punkten in einem Netzwerk. Dieser Algorithmus wird nun (in /STON87/) benutzt, um die Lösung für das Allokationsproblem in graphischer Darstellung zu erhalten.

**Ford-Fulkerson-Algorithmus:**

Gesucht wird der maximal mögliche Fluß (von einem Quellknoten zu einer Senke) in einem Netzwerk. Es muß also ein Schnitt hergestellt werden, der das Netzwerk in zwei Teile zerlegt, wobei die Summe der Kapazitätskosten entlang dieses Schnittes minimal sein soll. Außerdem müssen Quelle und Senke in unterschiedlichen Mengen liegen. Zunächst werden nun die S-T-Schnittmengen<sup>8</sup> (nach /BODE80/) definiert:

- Es gibt eine S-T Schnittmenge, deren Kapazität gleich dem maximalen Fluß zwischen S und T ist.
- Der maximale Fluß ist gleich der minimalen Kapazität der S-T Schnittmengen ("Max Flow-Min-Cut"- Theorem).
- Der Ford-Fulkerson-Algorithmus liefert eine "saturierte Schnittmenge", in der alle Kanten maximalen Fluß (bezüglich des Gesamt-Flusses, nicht jedoch bezüglich der Kapazität der einzelnen Kante) aufweisen.

**Definition (FFA):**

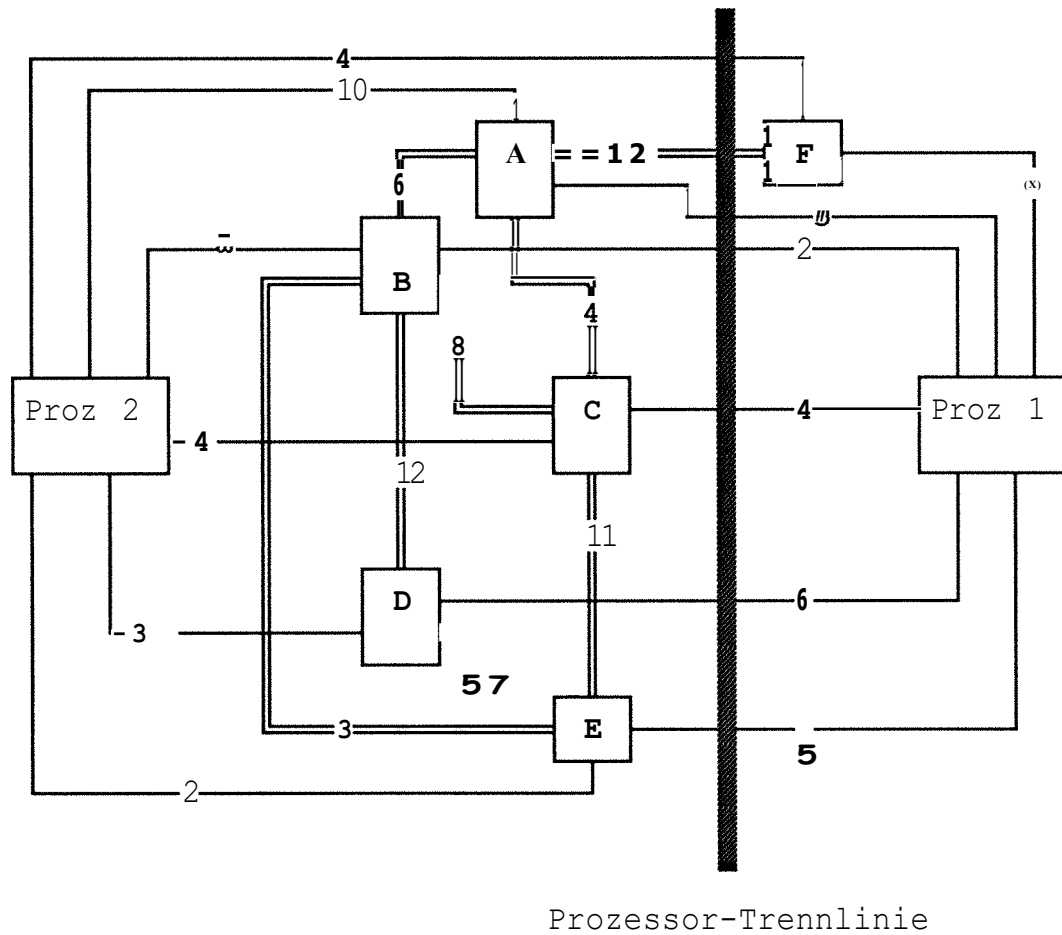
Der Ford-Fulkerson-Algorithmus findet den maximalen Fluß zwischen zwei Knoten S und T eines Netzwerkes mit Verbindungen, deren Kapazität gegeben sind.

Im Falle der Allokation entsprechen die beiden ausgezeichneten Knoten (Quelle und Senke) den beiden Prozessoren. Die Kanten sind beim Ford-Fulkerson-Algorithmus mit der Kapazität einer Verbindung beschriftet. Die Kapazität wird unabhängig von der Richtung des (Informations-) Flusses angegeben. Bei der Allokation sind die Kanten mit den Kommunikationskosten, bzw. den Rechenkosten beschriftet. Der minimale "Schnitt" in einem Netzwerk zerlegt den Netzwerkgraphen in zwei Teile. Diese Teile haben miteinander den geringsten Kommunikationsfluß. Der Schnitt durch den Graph stellt die optimale Lösung (und für das Allokationsproblem somit die geringsten Kommunikationskosten) dar.

---

<sup>8</sup> Die Quelle wird mit S, die Senke mit T bezeichnet.





— x — Kommunikationskosten  
 — y — Rechenkosten

Abb. : 6.3. Minimaler Schnitt durch das Netzwerk.

Die Prozessor-Trennlinie entspricht dem minimalen Schnitt, der vom Ford-Fulkerson-Algorithmus errechnet wird; die Verteilung der Tasks auf die Prozessoren dem minimalen Schnitt durch das Netzwerk. Die Beschriftung <sup>00</sup> bedeutet, daß die zugehörige Task dem entsprechenden Prozessor zugewiesen werden muß.

In /STON78/ wird ein Algorithmus entworfen, der einen Graphen in drei Teile zerlegt, wobei auch hier der minimale Schnitt gefunden wird. Dies bedeutet für das Allokationsproblem eine graphentheoretische Verteilung eines Softwaresystems auf drei Prozessoren. Bei der Erweiterung des Ford-Fulkerson Algorithmus auf drei Prozessoren wird analog zum zwei-Prozessor-Modell von drei Quellen (Senken) ausgegangen. Der Algorithmus verbindet alle Tasks mit den Prozessoren und berechnet den Fluß zwischen jeweils zwei Prozessoren (Quellen/Senken). Aus diesen Teilmengen wird dann die Gesamtlösung erstellt. Der Algorithmus führt nicht in allen Fällen zu einem Ergebnis. Das Optimum der Lösung kann ebenfalls nicht garantiert werden.

Für Systeme mit mehr als drei Prozessoren schlägt /STON78/ ein Verfahren vor, bei dem das  $n$ -Prozessor-Problem auf mehrere zwei-Prozessor-Probleme reduziert werden kann. Dieser Algorithmus führt ebenso wie der Algorithmus für drei Prozessoren nicht immer zu einer optimalen Lösung.

In jedem Schritt berechnet der Algorithmus den minimalen Fluß zwischen zwei Prozessoren. Daraus ergeben sich für je zwei Prozessoren entsprechende Schnittmengen. Aus diesen Mengen wird auf die optimale Zuordnung bei  $n$  Prozessoren (Teilmengen) geschlossen. Dieser Algorithmus wird mit einer Komplexität von  $O(n^2)$  angegeben.

Die graphentheoretischen Verfahren erlauben für den zwei-Prozessor-Fall ohne Beachtung der Ressourcen optimale Ergebnisse in vertretbarer Zeit. Für Fälle mit mehr als zwei Prozessoren sind graphentheoretische Methoden ungeeignet, da der Berechnungsaufwand sehr schnell wächst. Außerdem werden die Ressourcen nicht berücksichtigt. Allerdings ist es möglich, ein heterogenes Prozessorsystem durch geeignete Wahl der Beschriftung der Task-Prozessor-Knoten zu simulieren. Um (im Fall für zwei Prozessoren) zu erreichen, daß eine Task einem der Prozessoren bevorzugt zuzuteilen ist, kann die entsprechende Kante mit höheren Kosten belegt werden als die Verbindung zum anderen Prozessor. Dadurch wird vom Algorithmus die Zuteilung der Task zum ersten Prozessor bevorzugt.

Für den praktischen Einsatz eignen sich die graphentheoretischen Methoden aufgrund der zitierten Nachteile nicht.

Verfahren, die mit optimalen Methoden arbeiten, können also das Allokationsproblem für maximal drei Prozessoren in vertretbarer Zeit lösen. Für größere Systeme sind die optimalen Verfahren ungeeignet. Außerdem können mit diesen Algorithmen Randbedingungen der Allokation nicht beachtet werden. Des weiteren sind die Rechenkosten, wie sie in /STON78/ benutzt werden, kein geeignetes Mittel, um die Beziehungen zwischen Tasks und Prozessoren auszudrücken. Diese Kosten müssen im Voraus vom Benutzer dem Algorithmus mitgeteilt werden; somit wird also vom Benutzer für jede Task eine Zuordnungspriorität festgelegt. Das Allokationssystem hat in diesem Fall wenig Möglichkeiten die Tasks aufgrund "eigener Überlegungen" zuzuordnen. Diesen Nachteil bei der Verteilung handelt man sich dadurch ein, daß bei optimalen Verfahren die Randbedingungen nicht geeignet berücksichtigt werden. Deshalb wird vom Benutzer verlangt, daß er auf dem Umweg der Rechenkosten die Verteilung einschränkt.

In Realzeitsystemen müssen aber die Beziehungen zwischen Task und Peripherie, sowie zwischen Task und Prozessor in geeigneter Weise in das Allokationssystem miteinbezogen werden.

Die optimalen Verfahren eignen sich somit nur für kleine Systeme ohne Randbedingungen. Für solche Systeme existieren eine Reihe von Lösungsverfahren, die in vernünftiger Zeit das Optimum einer Verteilung berechnen können.

## 6.2. Lösung durch suboptimale (heuristische) Verfahren

### 6.2.1. Graphentheoretische Verfahren

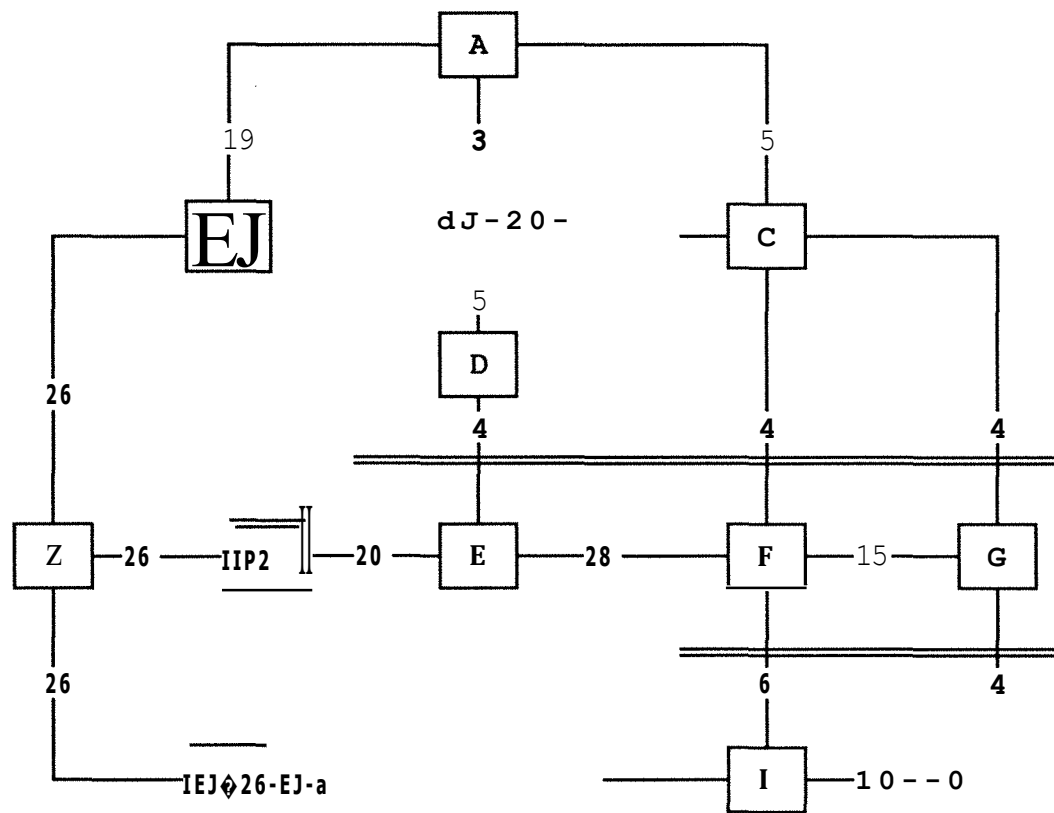
Auf einen Prozeßgraphen, wie er in Abschnitt 6.1. benutzt wurde, lassen sich auch graphentheoretische Algorithmen anwenden, die mit suboptimalen Methoden arbeiten. Ein von /AROR80/ vorgeschlagener Algorithmus nimmt sich in jedem Schritt einen Knoten des Prozeßgraphen vor, der mit dem Partnerknoten verschmolzen wird, mit dem er die größten Kommunikationskosten hat. Die Anfangsbelegung der Prozessoren (die Anfangszuordnung) ist beliebig. Da maximal  $n + m - 1$  Kanten korrigiert werden müssen, besitzt dieser Algorithmus eine Zeitkomplexität von  $O(n * m + m^2)$ . Dieser Algorithmus ist jedoch für den Einsatz in der Praxis ungeeignet, da außer den Kommunikationskosten keinerlei Randbedingungen berücksichtigt werden.

Eine Verbesserung dieses Algorithmus wird in /HAES80/ vorgenommen. In zwei Schritten wird zunächst eine Ausgangszerlegung eines Graphen bestimmt. In einem weiteren Schritt wird diese Lösung verbessert. Der Algorithmus arbeitet (wie in Kapitel 5 beschrieben) mit einem Eröffnungsverfahren und einem Iterationsverfahren. Das Eröffnungsverfahren arbeitet nach dem Algorithmus "maximum spanning tree" von Kruskal /REIN77/. Für diesen Algorithmus werden die Tasks als Knoten und die Kommunikation zwischen den Tasks als Kanten dargestellt. Die Kommunikationskosten sind als Kantenbeschriftung angeführt. Die Prozessoren werden als sogenannte "preferred nodes" zusätzlich in den Graphen eingebracht. Außerdem wird ein zusätzlicher Knoten (Z) benötigt, der mit allen Prozessorknoten verbunden wird.

Für die Anfangslösung wird kein beliebiger Knoten ausgewählt, sondern es werden zuerst die Kanten zwischen den Knoten in absteigender Reihenfolge ihrer Kosten daraufhin geprüft, ob sie eliminiert werden können. Nach Abschluß dieser Prüfung liegt ein Graph vor, bei dem jeder Teilgraph genau einen Prozessorknoten besitzt (Abbildung 6.4.). Ausgehend von dieser Lösung wird durch Verschieben der Taskknoten versucht, eine Verbesserung zu erreichen. Die Knoten, die verschoben werden sollen, werden durch den Ford-Fulkerson-Algorithmus bestimmt. Die Lösung, die damit erreicht wird, ist in Bild 6.5. dargestellt.

Die Randbedingungen werden jedoch auch in diesem Algorithmus nicht

berücksichtigt. Als einziges Verteilungskriterium dienen wieder die Kommunikationskosten.



Tasks (A -K)



Ausgezeichneter Knoten

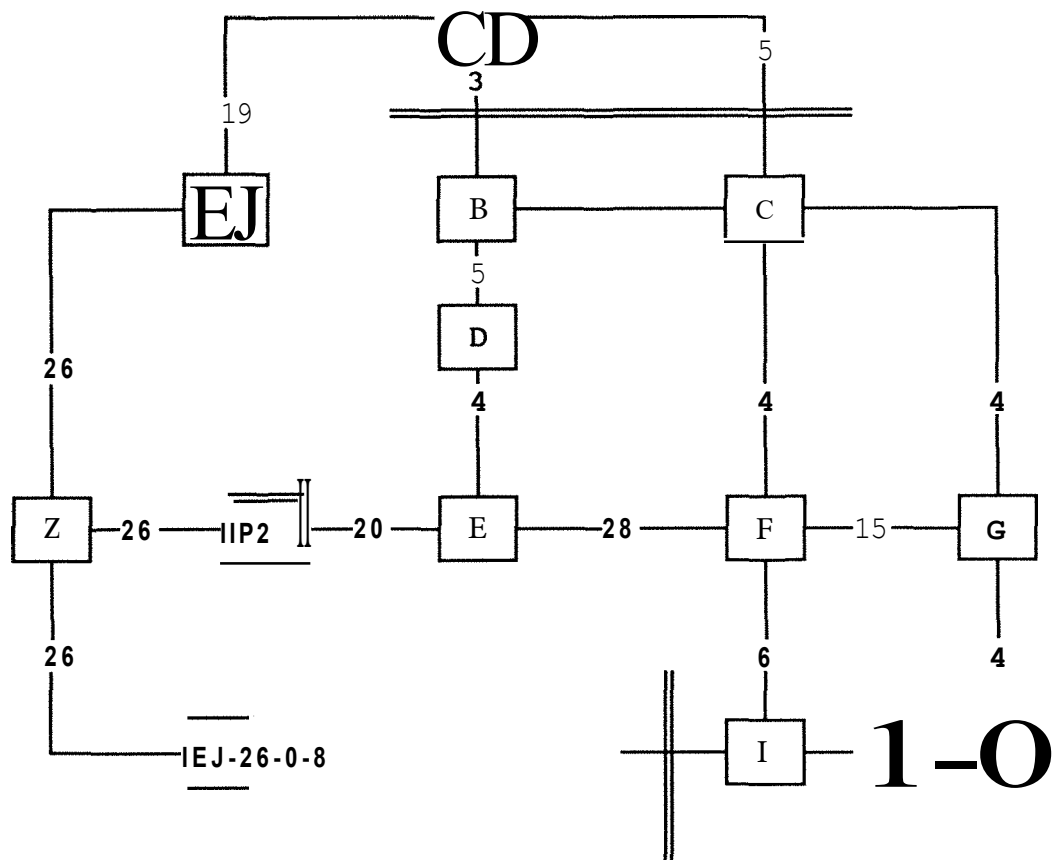


Prozessoren



Prozessortrennlinie

Abb. : 6.4. Programmgraph mit der Ausgangslösung



Tasks ( A -K)



Ausgezeichneter Knoten



Prozessoren



Prozessortrennlinie

Abb. : 6.5. Programmgraph mit optimaler Lösung

### 6.2.2. Verfahren mit heuristischen Abbruchkriterien

#### 6.2.2.1. Das Verfahren nach Ma

Wie in Kapitel 5 gezeigt, versuchen heuristische Verfahren den Aufbau des Lösungsbaumes an geeigneten Stellen abubrechen. Die verschiedenen Möglichkeiten, die zu einem geeigneten Abbruch führen, wurden dort erörtert.

Ein System, das mit Hilfe von vier Bedingungen den Aufbau des Entscheidungsbaumes abbricht, wird in /MA82/ vorgestellt. Das System arbeitet mit folgenden Einschränkungen:

- Es werden feste Randbedingungen berücksichtigt (z.B. die Größe eines Prozessors).
- Bestimmte Tasks werden bevorzugt betrachtet (eine Task kann dadurch einem bestimmten Prozessor bevorzugt zugewiesen werden).
- Tasks können von einer Zuordnung ausgeschlossen werden (bestimmte Paare von Tasks dürfen nicht dem gleichen Prozessor zugewiesen werden).
- Redundanz wird berücksichtigt (redundante Tasks müssen zwei oder mehr Prozessoren zugewiesen werden).

Diese Einschränkungen werden in der beschriebenen Arbeit durch Matrizen repräsentiert (Präferenz-Matrix, Ausschluß-Matrix, Redundanz-Matrix). Der von /MA82/ verwendete Algorithmus, der eine Erweiterung der Methode von Kohler und Steiglitz /COFF76/ darstellt, besteht aus 6 Bedingungen (F, E, RB, D, S, BR), die zum Abbruch beim Aufbau des Entscheidungsbaumes führen sollen.

Die Bedingungen entscheiden darüber, ob ein betrachteter Zweig  $i$  eines gegebenen Knotens  $k$  eliminiert wird:

#### 1. Bedingung F:

Diese Bedingung prüft die Präferenz-Matrix  $P$  für Task  $k$ . Falls  $P_{ik} = 0$  ist, wird der entsprechende Zweig im Entscheidungsbaum eliminiert.

### 2. Bedingung E:

Falls eine Task  $l$ , die nicht mit einer Task  $k$  gemeinsam einem Prozessor zugewiesen werden darf, bereits dem Prozessor  $i$  zugeordnet ist, wird der Zweig  $i$  im Knoten  $k$  eliminiert.

### 3. Bedingung RB:

Falls eine Task  $k$  auf einen Prozessor  $i$  gelegt werden soll, dessen Speicher nicht ausreicht, um die Task zusätzlich aufzunehmen, wird der Zweig  $i$  im Knoten  $k$  gelöscht.

### 4. Bedingung D:

Diese Bedingung vergleicht die partiellen Kosten  $L$  mit den Gesamtkosten  $U$ . Falls  $L$  größer als  $U$  ist, kann die Lösung nicht verbessert werden. Der Zweig  $i$  des Knotens  $k$  wird somit gelöscht.

Falls alle Bedingungen (1-4) erfüllt sind, wird die entsprechende Zuweisung vorgenommen. Ein Iterationszähler wird erhöht, um anzuzeigen, daß ein Zweig für den Knoten untersucht wurde. Mit den vier angegebenen Bedingungen wird eine Tiefensuche realisiert. Zwei weitere Bedingungen sorgen für die Knotenauswahl (Bedingung S) und für den Abbruch des Algorithmus (Bedingung BR).

Dieses Verfahren ist ein typischer Vertreter der heuristischen Allokationssysteme. Es werden Bedingungen angegeben, die beim Aufbau des Lösungsbaumes zu geeigneten Abbruchstellen führen sollen. Die Qualität der Lösung hängt in entscheidendem Maße von der Wahl der Abbruchbedingungen ab. Außerdem ist die Abarbeitungsreihenfolge wichtig. Die Bedingungen, die hier angegeben werden, sind aber nicht ausreichend, um eine Verteilung für Realzeitsysteme geeignet vorzunehmen. Außerdem ist keine Möglichkeit vorgesehen, eine falsche Entscheidung rückgängig zu machen. Falls der Lösungsbaum an falscher Stelle abgebrochen wird, wird diese Entscheidung nicht mehr verändert.

Für den praktischen Einsatz ist dieses Verfahren ungeeignet, da die Randbedingungen nicht die Anforderungen der Realzeitprogrammierung erfüllen.



#### 6.2.2.2. Das Verfahren nach Borrmann

Ein Verfahren, das neben verschiedenen Heuristiken auch eine Unterteilung des Allokationsvorganges in Teilschritte berücksichtigt, wird in /BORR86/ beschrieben. Diese einzelnen Schritte werden in einer festen Reihenfolge abgearbeitet. Ein Schwerpunkt dieser Arbeit liegt in der Zerlegung in Eröffnungsverfahren und Iterationsverfahren.

Das System benutzt folgende Schritte bei der Lösung des Allokationsproblems:

- Problemanalyse und Erzeugen eines Startsystems
- Einfügen weiterer Prozesse
- Vervollständigung und
- Iterative Optimierung

Die wichtigsten Entscheidungen werden bereits bei der Problemanalyse getroffen. Dazu wird nun weiter in verschiedene Heuristiken unterteilt:

- Einengung des Lösungsraumes  
Mit dieser Heuristik wird für jede Task untersucht, inwieweit die Allokationsfreiheit durch die Randbedingungen eingeschränkt ist. Als Ergebnis wird jeder Task eine Bewertungsziffer zugeteilt, die sie für den weiteren Verlauf entsprechend einschränkt.
- Cluster-Analyse  
Hiermit werden die Tasks aufgrund der Kommunikationskosten in Cluster eingeteilt. Dabei werden feste Zuordnungen (maximale Bewertungsziffer) aus dem vorherigen Schritt in die Entscheidung mit einbezogen.
- Betriebsmittel-Anpassung  
In der Betriebsmittel-Anpassung werden besonders große Tasks für eine bevorzugte Zuordnung selektiert. Die gleichmäßige Lastverteilung des Systems wird dadurch erreicht, daß man mit einer Heuristik versucht, jeder Task eine Bewertungsziffer entsprechend ihrer Größe zuzuordnen.

Nach Abschluß dieses Teiles des Allokationsalgorithmus liegt für jede Task eine Bewertung für die Zuteilung an die Prozessoren vor. Einige Tasks sind in einem Startsystem (dargestellt durch die maximale Bewertungsziffer) einem Prozessor bereits fest zugeordnet. In den weiteren Schritten werden nun die anderen Tasks aufgrund ihrer Bewertungsziffern diesem "Startsystem" eingegliedert.

Für die Eingliederung einer weiteren Task wird deren Anteil an den Gesamt-IPC-Kosten bestimmt. Aufgrund dieses Anteils wird dann entschieden, welchem Prozessor die Task am besten zugeordnet wird. Nach Zuteilung aller Tasks an die Prozessoren wird durch eine iterative Optimierung versucht, die Zielfunktion (siehe DEF 2.5. in Kapitel 5) durch Verschieben einzelner Tasks noch zu verbessern.

Das Allokationsverfahren von Borrmann arbeitet in allen Schritten mit der Berechnung von Funktionen zur Bestimmung von Bewertungsziffern. Aufgrund dieser Ziffern trifft der Algorithmus dann Entscheidungen, welchem Prozessor eine Task zugeordnet wird. Das Kommunikationsverhalten der Tasks, bzw. der Aufbau des Hardwaresystems, wird in einer eigenen Programmiersprache dargestellt. Ein Compiler gewinnt aus dieser Syntax die notwendigen Informationen für das Allokationssystem. Über die Darstellung der Ergebnisse ist bei Borrmann nichts ausgesagt.

Zusammenfassend lassen sich über das Allokationssystem von Borrmann folgende Aussagen machen:

- Das System erreicht eine Verteilung aufgrund der Optimierung der IPC-Kosten.
- Als Randbedingungen werden Speichergröße der Prozessoren, Task-Prozessor-Verbot und feste Hardwareanforderungen betrachtet.
- Ein Benutzer dieses Allokationssystems muß sein Problem mittels einer eigenen Programmiersprache darstellen.

Dieses Verfahren deckt weitaus mehr Anforderungen aus der Realzeitprogrammierung ab, als der von Ma vorgestellte Ansatz. Dennoch wird auch hier eine Optimierung des Allokationssystems nur nach den !PC-Kosten vorgenommen. Die Tasks werden zwar durch die betrachteten Randbedingungen in ihrer Allokationsfreiheit eingeschränkt, die Zuteilung erfolgt aber nur aufgrund eines berechneten Anteils der jeweiligen Task an den !PC-Kosten. Für die Verteilung im Rahmen der Realzeitprogrammierung sind aber weitere Kriterien zur Optimierung des Allokationssystems notwendig. Außerdem erscheint es wenig geeignet, die Zuteilung einer Task ausschließlich von einer einmal bestimmten Bewertungsziffer und deren Anteil an der Gesamtfunktion abhängig zu machen.

### 6.3. Zusammenfassung

Das Problem der Allokation wurde in der Literatur mit verschiedenen optimalen und suboptimalen Verfahren analysiert. Vor allem die suboptimalen (heuristischen) Verfahren waren bislang Gegenstand ausführlicher Untersuchungen.

- Die optimalen Verfahren eignen sich für Allokationsprobleme mit maximal drei Prozessoren. Für diese Prozessoranzahl stehen Lösungsverfahren zur Verfügung, die in vertretbarer Zeit das Optimum einer Verteilung berechnen können. Ein gravierender Nachteil ist, daß keine Randbedingungen (wie zum Beispiel die Einschränkung aufgrund der Hardwareforderungen) berücksichtigt werden können.
- Die suboptimalen (heuristischen) Verfahren können Allokationsprobleme "beliebiger" Größe in vertretbarer Zeit lösen. Diese Verfahren liefern allerdings nicht immer die optimale Lösung.
- In einigen Ansätzen wird versucht, die Ressourcen bei der Verteilung zu berücksichtigen (/KAS82/, /BANN82/).
- Die Lösung des Allokationsproblems wird durch Lösen einer Funktion (Bestimmung des Minimums) oder durch Enumeration (Aufbau des Lösungsbaumes) erreicht. Bei der Enumeration wurden verschiedene

Verfahren untersucht, mit deren Hilfe der Lösungsbaum "verkürzt" wird. Durch das "Kürzen" des Baumes kann das Verfahren der Allokation beschleunigt werden.

- Aus der Literatur sind nur wenige Arbeiten bekannt, bei denen Algorithmen zur Lösung des Allokationsproblem implementiert wurden. Die meisten Arbeiten beschränken sich auf theoretische Aussagen über Aufwand und Qualität der Verteilungen.

Es existierte bislang kein System, das sich die Information (in der Entwurfsphase) aus der Spezifikation zunutze macht, um daraus die Eingabe für ein Allokationssystem zu erstellen. Im Rahmen einer vollständigen Programmierungsumgebung kann eine Spezifikation (neben der Allokation) auch für andere Teilbereiche verwendet werden. Die Beschreibung eines Allokationsproblems mit Hilfe einer Spezifikationsmethode garantiert ein hohes Abstraktionsniveau, das für die komplexen Beziehungen zwischen den Tasks notwendig ist. Bislang wurden lediglich (bei theoretischen Arbeiten) Eingaben in Form von Variablen für Gleichungen, oder eigene Sprachen zur Darstellung des Allokationssystems (z.B. bei /BORR86/) benutzt.

Bestehende Systeme ignorieren die Tatsache, daß Allokationsprobleme bisher recht gut von menschlichen Experten gelöst wurden. Es ist also offensichtlich Wissen um Verteilungsstrategien vorhanden, das nutzbar gemacht werden sollte. Außerdem wird die Wichtigkeit der Randbedingungen nicht angemessen berücksichtigt: Optimiert werden meist nur die !PC-Kosten, in manchen Fällen wird zusätzlich noch ein Lastgleichgewicht angestrebt. Diese Kriterien reichen jedoch für den Einsatz in der Realzeitprogrammierung nicht aus.

Wünschenswert wäre es, verschiedene Randbedingungen in geeigneter Weise angeben zu können. Außerdem müssen verschiedene Kriterien zur Auswahl angeboten werden, nach denen die Verteilung optimiert werden kann. Aufgrund verschiedener Optimierungsmöglichkeiten können dann vom Benutzer Aussagen über die Qualität einer Verteilung gemacht werden.

Ein Allokationssystem, das für eine Realzeitumgebung Lösungen erarbeitet, muß folgende Bedingungen erfüllen:

- Beliebige Randbedingungen müssen in geeigneter Weise in das Allokationssystem einbezogen werden können.
- Verschiedene Optimierungskriterien sollen möglich sein.
- Das Allokationssystem soll in vertretbarer Zeit Ergebnisse liefern.
- Die Ergebnisse des Allokationssystems sollen in einer dem Benutzer verständlichen Form dargestellt werden.
- Da heuristische Methoden kein optimales Ergebnis liefern, soll das Ergebnis dem Benutzer erklärt werden.

Der Lösungsansatz im Rahmen der vorliegenden Arbeit soll so gestaltet werden, daß die folgenden Forderungen eingehalten werden können:

- Das Allokationssystem soll die Vorgehensweise eines menschlichen Experten simulieren.
- Die Spezifikation des Realzeitproblems soll überschaubar und leicht zu erstellen sein. Die Behandlung der Allokation sollte auf möglichst hohem Niveau, also bereits in der Entwurfsphase eines zu erstellenden verteilten Realzeitsystems erfolgen.

## Kapitel 7

### Wissensbasierte Systeme

#### 7. Wissensbasierte Systeme

##### 7.1. Abriß zur KI

Ausgehend von der Einsicht, daß menschliche Experten Allokationsprobleme durchaus vernünftig lösen, wird das Allokationsproblem in der vorliegenden Arbeit mit einem wissensbasierten Ansatz gelöst, der sich genau dieses Expertenwissen zunutze macht. Wissensbasierte Systeme sind als eine Anwendung der Künstlichen Intelligenz (**KI**) zu verstehen (im englischen Sprachraum: Artificial Intelligence (**AI**)). Zum besseren Verständnis der KI und deren Zielsetzungen seien einige Definitionen angegeben:

- [Artificial Intelligence is] "The capability of a machine to simulate the human thought process by performing functions that are usually associated with human intelligence, such as reasoning, learning, and self-improvement" (/IS088/).

- "Artificial Intelligence (AI) is the Part of computer science concerned with designing intelligent systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior - understanding language, learning, reasoning, solving problems, and so on. Many believe that insights into the nature of the mind can be gained by studying the operation of such programs. Since the field first evolved in the mid-1950s, AI researchers have invented dozens of programming techniques that support some sort of intelligent behavior" (/FEIG81/).
- "Die "Künstliche Intelligenz" ist eine Wissenschaft, deren Ziel die Simulation intelligenten Verhaltens auf Rechenanlagen ist, d.h. **künstlich erzeugte Intelligenz**. Sie versucht dieses Ziel durch die Entwicklung von Programmen zu erreichen, die auf Eingaben in gewisser Weise intelligent reagieren (intelligent Aufgaben lösen)" (/STOY87/).

Die KI beschäftigt sich also mit der Aufgabe, Analogien zu "Problemlösern" in der Natur, insbesondere zum menschlichen Gehirn zu finden. Auf praktischer Ebene sind vor allem Systeme (z.B. Expertensysteme) von Interesse, bei denen Teilbereiche automatisiert werden sollen.

Die ersten Forschungsarbeiten im Bereich der KI wurden von A. Newell, H. Simon, M. Minsky und J. McCarthy betrieben. Dabei standen die Tätigkeitsfelder Problemlösen, Theorembeweisen, induktives Schließen und Lernen im Vordergrund des Interesses. Wichtigstes Anliegen war damals die Nachbildung menschlicher Intelligenz auf dem Computer. Eine erste Definition der "Maschinellen Intelligenz" stammt von Alan Turing /TURI63/ und wird als Turing-Test bezeichnet. Damit kann ein Computer dann als intelligent bezeichnet werden, wenn er folgenden Test besteht

Ein Mensch stellt Fragen an einen anderen Menschen und an einen Computer. Alle Beteiligten sitzen in voneinander getrennten Räumen. Die Fragen werden nach dem Zufallsprinzip entweder vom Menschen oder vom Computer beantwortet. Nachdem eine Frage beantwortet ist, muß der Fragesteller eine Vermutung äußern, ob die Antwort vom Computer oder vom Menschen stammt. Wenn sich der Fragesteller in mehr als 50% der Fälle irrt, kann der Computer als intelligent bezeichnet werden.

Eine der ersten KI-Arbeiten zur Maschinellen Intelligenz war der General Problem Solver (GPS /NEVE63/). Mit ihm versuchte man, unabhängig von einem Wissensgebiet, Probleme (z.B. der Geometrie) zu lösen. Nachdem man aber zur Erkenntnis gelangt war, daß die dabei verwendeten allgemeinen Heuristiken nicht zur Lösung "beliebig" komplexer Probleme ausreichten, begrenzte man das Anwendungsgebiet. Systeme, die auf diesem Grundgedanken basieren, nennt man Expertensysteme. Sie beschäftigen sich hauptsächlich damit, das Spezialwissen und die Schlußfolgerungsfähigkeit qualifizierter Fachleute auf einem Rechner nachzubilden.

Die ersten wissensbasierten Systeme wurden Anfang der 60er Jahre in verschiedenen Universitäten der USA (MIT, Rutgers, Stanford) entwickelt. Die bekanntesten Systeme aus dieser Zeit sind medizinische Konsultationssysteme (z.B. MYCIN /SHOR76/) und natürlichsprachliche Beratungssysteme (z.B. SRI Consultant /HART75/).

Die KI, wie sie sich heute dem Betrachter darstellt, besteht aus mehreren "Teildisziplinen", die sich weitgehend verselbständigt haben:

- Spielprogrammierung

Mit dieser Art der Programmierung wird versucht, menschliche Problemlösestrategien anhand bestimmter Problemstellungen, wie beispielsweise das Schachspielen, auf Rechnern nachzubilden.

- Automatisches Beweisen

Als Spezialfall des Problemlösens kann das automatische Beweisen angesehen werden. Dabei geht es um Beweise von mathematischen Formeln.

- Natürlichsprachliche Systeme

Als Ziel dieser Systeme wird eine natürlichsprachliche Mensch-Maschine-Schnittstelle genannt, wobei beide "Richtungen" der Kommunikation zwischen der Maschine und dem Menschen möglich sein sollen.

- Bildverarbeitende Systeme

Die Bildverarbeitung ist ein klassisches Gebiet der Mustererkennung, die sich mit KI-Methoden den Aufwand für die Erkennung eines Bildes



erleichtern will. Dieser Zweig der KI ist in jüngster Zeit eng verknüpft mit der Robotik. Die Roboter werden mit Sensoren ausgestattet, mit denen sie in der Lage sind mit ihrer Umwelt zu kommunizieren.

- Robotik

Die Robotik ist das jüngste Anwendungsgebiet der KI, die sich vor allem mit der Planung von Aktionssequenzen und von Robotern beschäftigt.

- Wissensbasierte Systeme (Expertensysteme)

Wissensbasierte Systeme sollen die Tätigkeiten von (menschlichen) Experten unterstützen und (teilweise) mechanisieren.

## **7.2. Grundlagen wissensbasierter Systeme**

Die wissensbasierten Systeme, deren Anwendungen als Expertensysteme zusammengefasst werden, stützen sich auf Methoden der KI und lösen ein bestimmtes Problemgebiet. Die Expertensysteme werden als "special-purpose" Systeme<sup>9</sup> bezeichnet, da sie einen sehr eng umgrenzten Wissensbereich verarbeiten.

Um dem Leser die Einordnung der vorliegende Arbeit vor dem Hintergrund der KI (und insbesondere der wissensbasierten Systeme) zu erleichtern, werden im folgenden die Methoden näher erläutert, die bei wissensbasierten Systemen benutzt werden.

Eine mögliche Klassifizierung von KI-Systemen findet sich in der Darstellung 7.1.

---

<sup>9</sup> Im Gegensatz dazu werden Systeme, die universelle Probleme lösen als "general-purpose" Systeme bezeichnet.

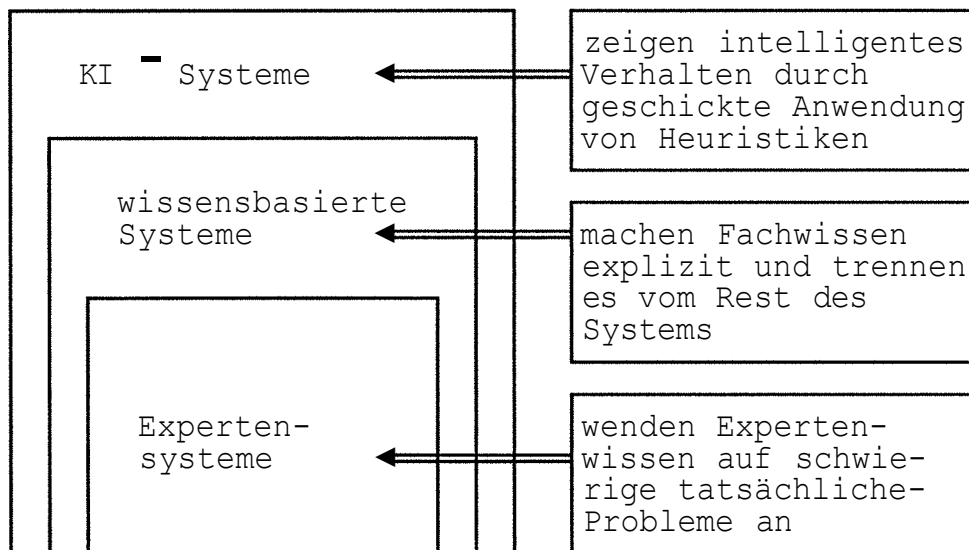


Abb.: 7.1. Klassifizierung der KI - Systeme

Wissensbasierte Systeme werden nach (/IS088/) folgendermaßen definiert:

"A heuristic system that provides for solving problems in a particular application area by drawing inference from a knowledge base acquired by human expertise".

Um das vorhandene (Experten-) Wissen in einem System besser handhaben zu können, werden nach /FEIG81/ die folgenden Komponenten benötigt:

- Acquisition (Erwerb von Wissen)

Lernen wird gewöhnlich als Akkumulation von Wissen bezeichnet, aber es ist mehr als das Aneignen neuer Fakten. Der Erwerb von Wissen wird bei wissensbasierten Systemen entweder durch geeignete Programmierung (z.B. in einer Wissensrepräsentationssprache (OPS83)), oder in einer dem menschlichen Vorgehen ähnlichen Art, also durch Lernen aus einer systematischen Sequenz von Beispielen und Gegenbeispielen (computer-aided instruction).

- Retrieval (Zugriff auf das Wissen)

Eine adäquate Organisation des "Gedächtnisses" soll die Auswahl des

Wissens ermöglichen. Sie bestimmt, welches Wissen in einer Situation relevant ist oder möglicherweise erwartet werden kann.

- Inferenz (Schließen aus vorhandenem Wissen)  
Schließen, Schlußfolgern ist ein Denkprozeß, in dem aus vorhandenem Wissen bzw. vorhandenen Annahmen oder Vermutungen neues Wissen, bzw. weitere Annahmen oder Vermutungen gewonnen werden (/SCHE87/).

Ein wissensbasiertes System besteht im allgemeinen aus der Wissensbasis und einer Schlußfolgerungskomponente. Es wird also das zu verarbeitende Wissen in problemspezifisches Wissen und Meta-Wissen aufgeteilt.

Unter problemspezifischem Wissen wird das Wissen verstanden, das den konkreten Anwendungsbereich beschreibt; es werden Sachverhalte ("Wissen") formuliert.

Das Meta-Wissen gibt an, wie das problemspezifische Wissen zu bearbeiten ist, die Wissensbasis wird mit Hilfe des Metawissens manipuliert. Dies geschieht dadurch, daß gewisse Schlüsse über den Repräsentationen der Sachverhalte (des problemspezifischen Wissens) zugelassen werden. "... Wenn immer eine gewisse Situation eintritt, ist eine bestimmte Folge von Aktionen durchzuführen, die korrespondierende Regel zu "aktivieren" ("feuern"). Diese Aufgabe wird von einem Regelinterpreter<sup>10</sup> wahrgenommen." (/BECK88/).

Der Mechanismus des logischen Schließens über gegebenen Wissensbeständen wird Inferenz genannt. Die Bearbeitungsmittel, auf die sich die wissensbasierten Systeme stützen, lassen sich wie folgt grob zusammenfassen:

- Repräsentation und Verarbeitung von Wissen
- Verwendung von Heuristiken
- Inferenz als Verarbeitungstechnik
- Inferenz mittels explizit definierter Wissensbasen

---

<sup>10</sup> Den Spezialfall eines *logische Schlüsse* vollziehenden Regelinterpreters nennen wir im folgenden auch *Inferenzmaschine*. In der Literatur wird diese Unterscheidung zwischen Inferenzmaschine und Regelinterpreter meist verwischt.

Bei den wissensbasierten Systemen wird das Wissen nicht (wie in herkömmlichen Systemen) fest in den realisierten Algorithmen codiert. Es wird vielmehr versucht, das Wissen über die problemspezifische Verarbeitung zu extrahieren und in der Wissensbasis zu isolieren.

In konventionellen Programmen werden (nach Watermann /WATE78/) folgende Vorgehensweisen berücksichtigt:

- Repräsentation und Verarbeitung von Daten
- Manipulierung der Daten durch Algorithmen
- Wiederholung als Technik der Abarbeitung
- Effektive Verarbeitung großer Datenbestände

Der Unterschied zwischen konventionellen und wissensbasierten Systemen liegt also (/STEF84/) darin, daß bei ersteren zwischen Daten und Programmen und bei den zweiten zwischen Daten, Programmen und Wissensbasis unterschieden wird. Im Unterschied zu konventionellen Systemen, wo Wissen in Algorithmen codiert ist, wird bei wissensbasierten Lösungen von Problemen das Wissen in gebietsspezifisches Wissen, allgemeines Inferenzwissen und Kontrollwissen unterteilt. Wissensbasierte Systeme lassen sich als eine Verschmelzung von Systemen zur Wissensrepräsentation und zur (algorithmischen) Problemlösung beschreiben.

### 7.3. Techniken der Wissensrepräsentation

Um (menschliches) Wissen in einem Rechner darstellen zu können, bedarf es einer Repräsentationsstruktur, in der das Wissen so erfaßt werden kann, daß das (wissensbasierte) System die Beziehungen zwischen einzelnen Objekten verstehen und diese Beziehung manipulieren kann.

Die Wissensrepräsentation wird gegenwärtig mit verschiedenen Formalismen erreicht. Die bekanntesten sind:

- Prädikatenlogische Wissensrepräsentation,
- Semantische Netze,
- Frames,
- Wissensrepräsentation mit Hilfe von Regeln.

Um eine für das Allokationsproblem geeignete Repräsentationsmethode zu finden, untersuchen wir die unterschiedlichen Methoden etwas näher:

- **Prädikatenlogische Wissensrepräsentation**

Bei der prädikatenlogischen Wissensrepräsentation werden logische Aussageformen benutzt, um Wissen als Axiome darzustellen. Der Vorteil dieser Methode besteht darin, daß die Theorie wohlbekannt und die Semantik genau definiert ist. Probleme treten bei der prädikatenlogischen Repräsentation jedoch auf bei

- der Erkennung und Behandlung von Inkonsistenzen,
- der Behandlung des Metawissens,
- der Formalisierung und Behandlung von vagem Wissen und
- bei der Zeitabhängigkeit von Bewertungen.

Ein Vertreter dieser Methode findet sich in der Programmiersprache PROLOG (n.mgramming in )Qgic, Clocksin/Mellish 1981). Prolog ist aber für das Allokationsproblem semantisch nicht akzeptabel, da Backtracking<sup>11</sup> als wichtigstes Verarbeitungsmittel angewendet wird. Treten Seiteneffekte auf, können diese nicht mehr rückgängig gemacht werden. Unter Seiteneffekten sind in diesem Zusammenhang Veränderungen an der Wissensbasis zu verstehen. Um bei einem Backtracking alle Seiteneffekte rückgängig machen zu können, müßte man sich alle ausgeführten Operationen merken und diese rückgängig machen können. Prolog unterstützt deklarative und prozedurale Wissensrepräsentation, erlaubt jedoch keine explizite Darstellung von Kontrollwissen.

Das von Prolog unterstützte Backtracking ist für die Anforderungen, wie sie bei der Allokation gestellt werden, nicht ausreichend.

Wegen der geschilderten Probleme ist die prädikatenlogische Wissensrepräsentation für das Allokationsproblem ungeeignet.

---

<sup>11</sup> Unter Backtracking wird die Zurücknahme einer Aktion verstanden, die zu keinem Ergebnis (oder einem falschen Ergebnis) führt.

### - Semantische Netze

Mit der graphischen Notation der semantischen Netze soll -neben den visuellen Vorteilen- auch die Nähe zur Implementierung im Computer und die Verwandtschaft zu assoziativen Gedächtnismodellen der Psychologie illustriert werden. In einem relationenorientierten Ansatz werden Objekte und Relationen zwischen den Objekten mit Hilfe von Knoten und Kanten in einem markierten und gerichteten Graphen dargestellt. Die Knoten repräsentieren die Objekte, während die Kanten die Beziehungen zwischen den Objekten herstellen. Man beschränkt sich vorteilhaft auf einige wenige Relationen (z.B. IS\_A, INSTANZ\_OF, MEMBER\_OF...), um die Anzahl der Kanten- und Knotentypen gering zu halten. Dadurch wird die Übersichtlichkeit des Netzes unterstützt, was den Betrachter zu einem schnelleren Verständnis der logischen Zusammenhänge verhilft. Diese Art der Darstellung bringt aber auch Nachteile mit sich. In Netzen können nur zweistellige Relationen dargestellt werden (die Kanten entsprechen den Relationen) und Quantoren sind graphisch sehr schwierig zu repräsentieren. Da nur zweistellige Relationen dargestellt werden können, müssen für verschiedene zusammenhänge der Objekte jeweils eigene Relationen eingeführt werden. Für das Allokationsproblem sind sehr viele Relationen zu erwarten, deshalb werden die semantischen Netze nicht für die Wissensrepräsentation verwendet.

Anhand eines Ausschnittes aus dem Allokationsproblem ist in der Abbildung 72 ein semantisches Netzwerk dargestellt. Die Beziehungen für die Task DIAL würden dabei beispielsweise folgendermaßen aussehen:

<u>Objekte:</u>	DIAL	
	KOPIE	
<u>Relationen:</u>	DIAL ist_eine	Task
	DIAL hat_Verbindung_mit	KOPIE
	DIAL ist- verbunden- mit	DRUCKER__1

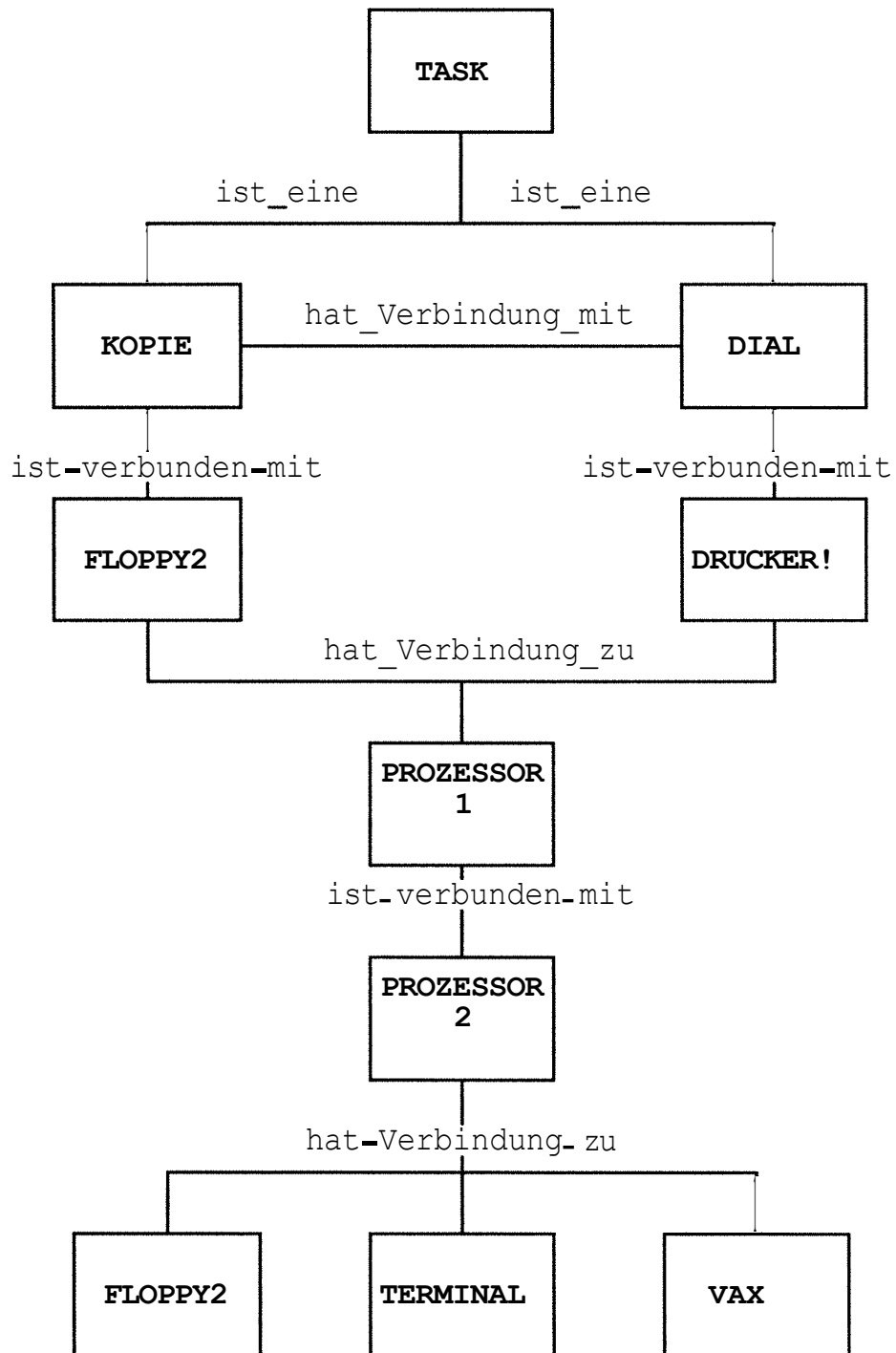


Abb.: 7.2. Semantisches Netz

- Frames

Eine Weiterentwicklung der semantischen Netze zu Frames (/MINS75/) war die Zusammenfassung von Wissen in begriffliche Einheiten. Die "Frame-Theorie" ist eine Theorie kognitiver Prozesse, die davon ausgeht, daß Erkennen, Verstehen und Inferenz durch "Gestalt"-artige Strukturen erklärbar sind. Minsky beschreibt die Arbeitsweise der Frames so:

- "A *frame* is a data structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed":

Die Frames stellen (im Gegensatz zum relationenorientierten Ansatz der semantischen Netze) einen objektorientierten Ansatz dar. Zusätzlich zu den Techniken der semantischen Netze können hier Mechanismen zur Vererbung und Instanzenspezifikation angewandt werden. Ein Frame besteht aus einer Reihe von 'Slots'. Slots enthalten Werte, Funktionen oder Zeiger auf untergeordnete Frames. Durch diese Strukturierung sind Slots hierarchisch aufgebaut. Innerhalb dieser Hierarchie können Objekte von "höheren" an "niedrigere" Instanzen vererbt werden. Die höchste Hierarchiestufe eines Frames repräsentiert das generelle Konzept, während die niedrigeren Knoten spezifischere Instanzen eines Konzeptes darstellen. Frames können als Knoten in semantischen Netzen verstanden werden, die innerhalb eines Knotens zusätzliche Information besitzen. Durch die hierarchische Strukturierung sind Frames in vielen Fällen besser zur Wissensrepräsentation geeignet als semantische Netze. Systeme, die auf der Basis der Frames arbeiten, bieten die Möglichkeit, die Eigenschaften von Objekten zu vererben.

In der Abbildung 7.3. ist eine Framerepräsentation für das Konzept eines Berichtes aufgezeigt. Oberste Hierarchiestufe ist hier der "Bericht", während die niedrigeren Knoten die Instanzen eines Berichtes darstellen. In den Slots sind beispielhaft der Autorenname, der Geschäftsbereich, das Datum und die Länge des Berichtes aufgeführt.



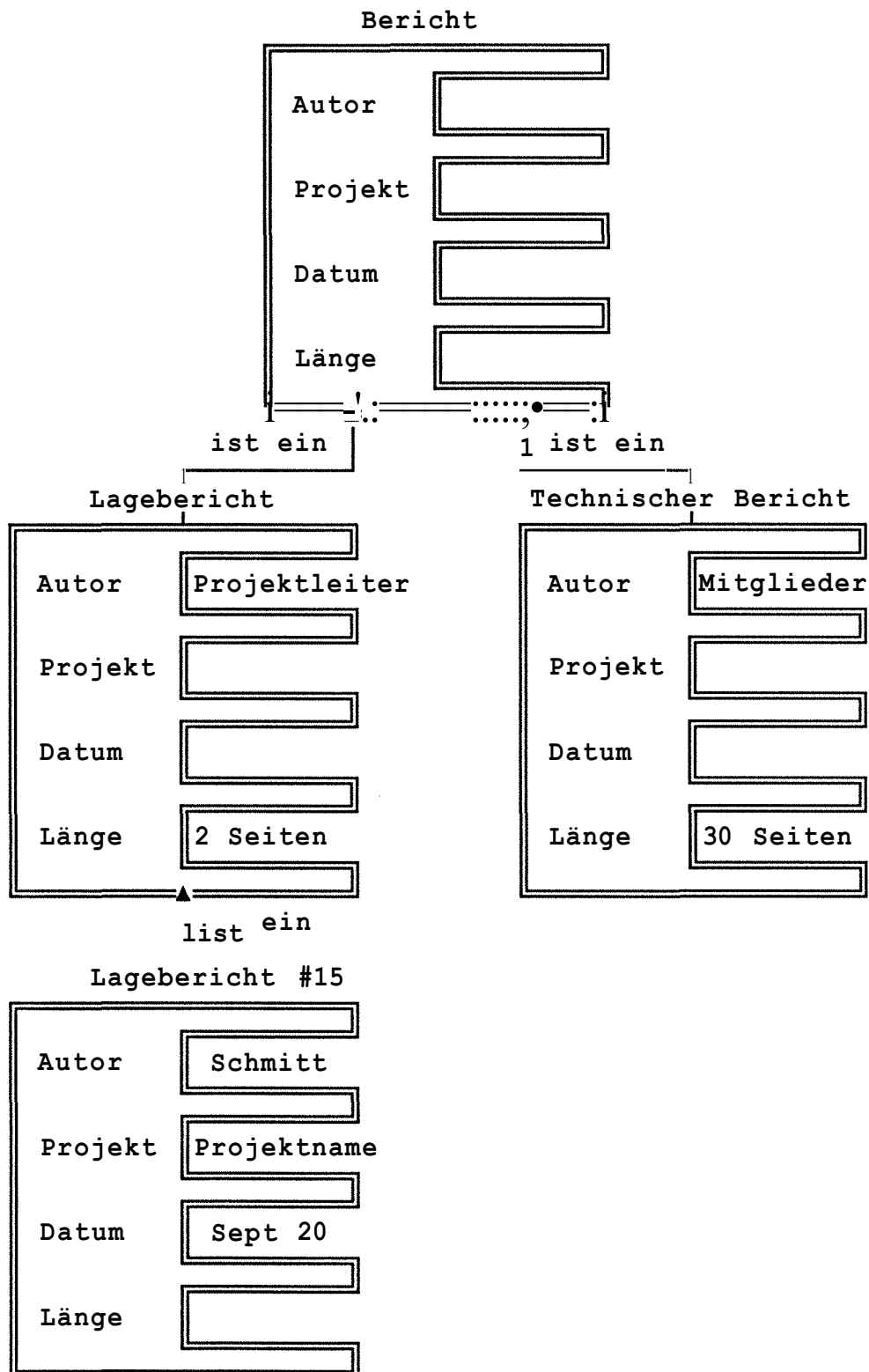


Abb.: 7.3. Framerepräsentation am Beispiel eines Berichtes

Für die Frames gelten in Bezug auf die Menge der benötigten Relationen zur Darstellung eines Problems die gleichen Aussagen wie für die semantischen Netze. Außerdem erscheint der Mechanismus der Vererbung für das Allokationsproblem nicht hilfreich.

- Wissensrepräsentation mit Hilfe von Regeln

Die Wissensrepräsentation mit Hilfe von Regeln findet immer größere Verbreitung und soll auch für die vorliegende Arbeit Anwendung finden. Der Begriff "Regel" in wissensbasierten Systemen hat eine sehr viel speziellere Bedeutung als in der Umgangssprache. Mit Hilfe von Regeln läßt sich vor allem Wissen repräsentieren, das (aus Erfahrungen) von einem Experten über einen längeren Zeitraum gewonnen wurde.

Nach /HAZE85/ teilen alle regelbasierten Systeme folgende Eigenschaften:

- Wissensrepräsentation in der Form bedingter Ausdrücke
- Effiziente Problemlösung durch geeignete Selektion von Regeln
- Datenabhängige (automatische) Selektion von Regeln
- Anwendung einer ausgewählten Regel zur Lösung des gestellten Problems
- Erklärungsmöglichkeit der ausgeführten Inferenz
- Möglichkeit der inkrementellen Erweiterung einer Wissensbasis
- Besondere Art der Kontrollstrukturen (Inferenzmechanismus)

Regeln werden in IF {premise} - THEN {action} - Anweisungen ausgedrückt, wie folgendes Beispiel aus dem System MYCIN<sup>12</sup> (/SHOR76/) verdeutlicht:

```
IF      (1)   es liegt ein zu therapierender Organismus vor
      & (2)   es können noch weitere, zu therapierende Organismen
              vorliegen
THEN    (1)   stelle Liste möglicher Therapien auf
      & (2)   bestimme Empfehlung für beste Therapie
```

---

<sup>12</sup> MYCIN ist ein System zur Beratung von Ärzten für Entscheidungen bei der Behandlung bakteriogener Infektionskrankheiten.

Die Repräsentierung mit Regeln ist dem Problem der Allokation am besten angepaßt. Wie in Kapitel 9 gezeigt wird, eignen sich Regeln vor allem für die Implementierung der Randbedingungen der Allokation besonders gut. Ein weiterer Grund für die Verwendung von Regeln war die Tatsache, daß die befragten Experten (/NIEM87/, /BEIE88/) das Problem in einer Art darstellten, die der Arbeitsweise der Regeln ähnlich ist.

Da dem hier verfolgten wissensbasierten Ansatz des Allokationssystems ein regelbasiertes System zugrundeliegt (für das Allokationssystem werden Produktionsregeln zur Beschreibung des Expertenwissens benutzt), werden die wichtigsten Prinzipien solcher Systeme im folgenden Abschnitt näher dargelegt.

## 7.4. Produktions- und Regelsysteme

### 7.4.1. Grundlagen der Regelsysteme

Vorläufer der heutigen Regelsysteme sind die **Produktionensysteme**, deren theoretische Grundlagen auf Post /POST43/ zurückgehen. Posts Überlegungen gründeten auf der Idee, logische Systeme durch Manipulation von Symbolketten über einem bestimmten Alphabet darzustellen. Aus dem Vorrat der Zeichen (Alphabet) werden Startwörter (Axiome) gebildet, die sich durch Anwendung von Produktionen (Regeln) in neue Wörter (Theoreme) überführen lassen. Die Produktionen von Post bestehen aus einer "linken Seite", in der die Hypothese angegeben wird, und einer "rechten Seite", mit der beschrieben wird, wie die Zeichen zu verändern sind. Produktionen sind immer dann anwendbar, wenn eine Übereinstimmung eines Teilwortes mit der linken Seite einer Produktion festgestellt wird, was mit einem Mustererkennungsalgorithmus überprüft wird. "Ausführen" der rechten Seite bedeutet bei den Regeln von Post die Ersetzung eines Teilwortes. Diese Idee der Regelsysteme wurde von der "Künstlichen Intelligenz" aufgegriffen und zu der regelorientierten Programmierung ausgebaut.

#### 7.4.2. Systemkomponenten eines regelbasierten Systems

Der Begriff des Produktionensystems (regelbasiertes System) wurde vor allem von A. Newell Anfang der 70er Jahre im Zusammenhang mit Versuchen zur Modellbildung über psychische Gedächtnisphänomene geprägt (/NEWE72/). Die Produktionsregeln (im Gegensatz zu logischen, empirischen,... Regeln) in einem regelbasierten Systemen haben, wie die Regeln in Produktionensystemen, einen Bedingungsteil und einen Ausführungsteil. Ein Produktionensystem (regelbasiertes System) besteht nach /WATE78/ aus folgenden Teilen:

- Kontroll- und Interpretationsmechanismus ("recognize-act-cycle") aus Algorithmen zur Mustererkennung (Vergleich der Bedingungsteile der Regeln mit den Fakten im Arbeitsspeicher) und der Regelverarbeitung.
- Wissensbasis (LTM, Long Ierm Memory) als Vorrat von "Wenn-Dann-Regeln".
- Arbeitsspeicher (STM, Short Ierm Memory), der durch die Ausführung der Regeln modifiziert wird<sup>13</sup>.

##### 7.4.2.1. Recognize-Act-Cycle

Im "recognize-act-cycle" werden mit Hilfe eines Retrieval-Mechanismus die Bedingungsteile der relevanten Regeln überprüft. Dazu werden alle Bedingungsteile ("Wenn-Teile") einer Regel (aus dem LTM) mit dem Inhalt des Arbeitsspeichers verglichen. Diejenigen Regeln, deren Bedingungen erfüllt sind ("gültige Regeln"), gelangen schließlich zur Anwendung ("Feuern einer Regel"). Die gültigen Regeln werden in eine spezielle Liste eingetragen, die sogenannte Konfliktmenge ("conflict set"). Die Konfliktmenge umfaßt eventuell mehrere gültige Regeln, sodaß mittels der Konfliktlösestrategie ("conflict resolution") die Konfliktmenge zugunsten einer Regel aufgelöst werden muß. Der dabei angestoßene Verarbeitungszyklus des Regelinterpreters endet erst, wenn aufgrund der Veränderungen im STM keine Regel mehr erfüllt werden kann,

---

<sup>13</sup> Ein Eintrag in den Arbeitsspeicher wird in OPS83 als Working-Memory-Element bezeichnet.

oder eine explizite Stop-Anweisung erkannt wird.

Zur Konfliktauflösung wurden in der Literatur (z.B. in /BABR83/, /FORG81/ und /TIEL85/) verschiedene Strategien diskutiert:

- Ausführen der Regel, die als letzte (erste) die Konfliktmenge betrat (Aktualität).
- Auswahlkriterien in Form von Regeln als Metawissen (Explizite Auswahl).
- Innerhalb von Regelmengen (z.B. "first rule") sind die Regeln geordnet (globale Ordnung).
- Parallele Abarbeitung aller Regeln.
- Zufällige Auswahl einer Regel.
- Auswahl der Regel, die die meisten Klauseln erfüllt hat.

Die aus der Konfliktmenge ausgewählte Regel darf feuern, dies bedeutet, daß die Anweisungen des "Dann-Teiles" zur Ausführung gelangen, wobei man generell zwischen solchen Operationen unterscheidet, die sich als "normale" Sprachkonstrukte (dies sind Konstrukte, die keine Seiteneffekte haben) erweisen, und solchen, die den Inhalt des Arbeitsspeichers verändern.

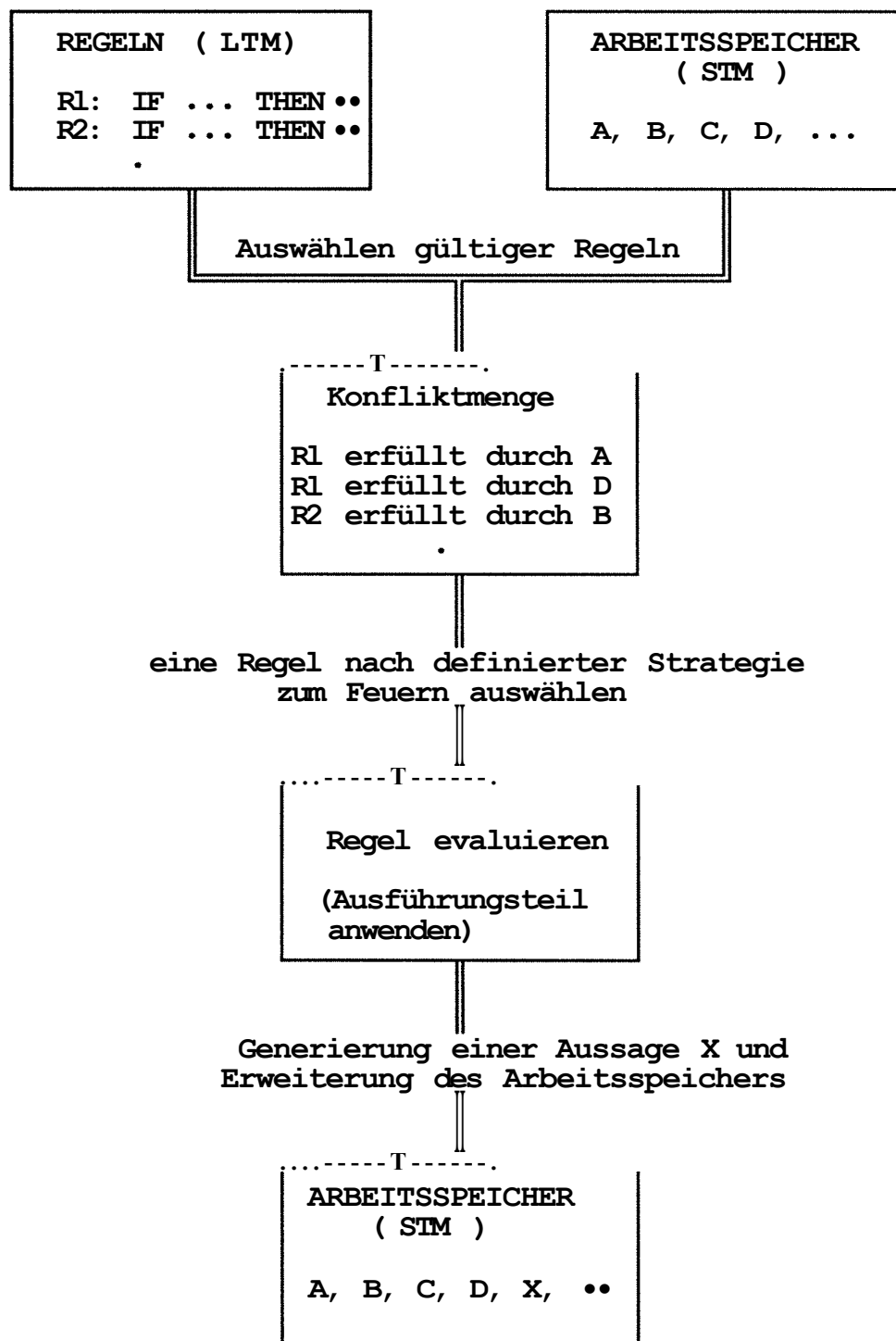


Abb.: 7.4. Der Recognize-Act-Cycle in einem regelbasierten System

#### 7.4.2.2. Aufbau der Regeln

Die Regeln in heutigen Expertensystemen sind ähnlich aufgebaut wie die von Post skizzierten Produktionen. Regeln bestehen syntaktisch aus einer Prämisse, die durch Elemente aus dem Arbeitsspeicher erfüllt werden muß. Wenn alle Prädikate und somit die Prämisse erfüllt sind (und die Regel vom Interpreter ausgewählt wurde), kann mit Hilfe des Ausführungsteiles der Inhalt des Arbeitsspeichers verändert werden. Die Regeln in einem regelbasierten System bestehen aus einer LHS (Left-Hand-Side) und in eine RHS (Right-Hand-Side). Je nach Strategie (Vorwärtsverkettung oder Rückwärtsverkettung) wird eine der beiden Seiten als eine Interpretation der Fakten aus dem Arbeitsspeicher angesehen. Die andere Seite der Regel beschreibt, wie der Arbeitsspeicher geändert werden kann. Aufgrund der Veränderungen im Arbeitsspeicher können folglich wieder neue Bedingungen "wahr" werden und somit neue Regeln anwendbar sein.

#### 7.4.2.3. Strategien

Aus der Unterscheidung zwischen datengesteuerter Regelverarbeitung (Vorwärtsverkettung) und zielorientierter Regelverarbeitung (Rückwärtsverkettung) resultieren zwei unterschiedliche Begriffe der Regelanwendbarkeit und somit auch zwei unterschiedliche Begriffe der Regelselektion:

- **Datengesteuerte Regelverarbeitung (Vorwärtsverkettung)**

Datengesteuerte Regelverarbeitung ("forward-chaining") bedeutet, daß die Prädikate der LHS einer Regel erfüllt sein müssen. Falls die LHS gültig ist, kann die RHS zur Ausführung gelangen. Hier werden - ausgehend von einer bestimmten Belegung des Arbeitsspeichers - so lange Regeln angewendet, bis der Inhalt des Arbeitsspeichers keine Ersetzung mehr ermöglicht.

Beispiel zur datengesteuerte Regelverarbeitung (/WATE75/):

Alphabet: a, b, A, B  
 Variable: x  
 Regeln: R1) aa --> a 14  
 R2) bb --> b  
 R3) xa --> ax  
 R4) a --> A  
 R5) b-->B

Die Prioritäten dieser Regeln entsprechen der Reihenfolge, in der sie hier aufgeführt sind. Mit einer Anfangsbelegung des Arbeitsspeichers "bbaba" ergibt sich folgender Ablauf:

Regel:	STM:	Konfliktmenge:	ausgewählte Regel:
	bbaba	R2, R3, R4, R5	R2
R2 =>	baba	R3, R4, R5	R3
R3 =>	abba	R2, R3, R4, R5	R2
R2 =>	aba	R3, R4, R5	R3
R3 =>	aab	R1, R4, R5	R1
R1 =>	ab	R4, R5	R4
R4 =>	Ab	R5	R5
R5 =>	AB	0	0
0 =>	STOP		

#### - Zielorientierte Regelverarbeitung

Bei der Rückwärtsverkettung der Regeln ("backward-chaining") wird der Inhalt des Arbeitsspeichers als Hypothese aufgefaßt. Durch die Anwendung der Regeln soll versucht werden, diese Hypothese zu beweisen. Hier wird, umgekehrt wie bei der Vorwärtsverkettung, die RHS einer Regel als Prämisse interpretiert. Falls die RHS einer Regel anwendbar ist, kann dann die LHS zur Ausführung gelangen. Dies bedeutet bei der Rückwärtsverket-

---

14 aa --> a soll bedeuten: Falls aa existiert und die Regel ausgewählt wird, kann sie feuern und a erzeugen.



tung, daß die durch die LHS postulierten Teilziele überprüft werden. Die Hypothese gilt als bestätigt, wenn keine Regel mehr anwendbar ist und alle Aussagen im Arbeitsspeicher bestätigt wurden.

Beispiel zur zielorientierte Regelverarbeitung (/WATE75/):

Inhalt des Arbeitsspeichers: A und F

Regeln:

R1)	A & B & C	-->	D
R2)	D & F	-->	G
R3)	A & J	-->	G
R4)	B	-->	C
R5)	F	-->	B
R6)	L	-->	J
R7)	G	-->	H

Im Beispiel sei das Ziel, die Aussage "H" zu überprüfen. Angenommen wird, daß die Aussagen "A" und "F" bereits (als wahr) im Arbeitsspeicher stehen. Es ergibt sich somit folgender Ablauf der Überprüfung:

Regel:	STM: wahr	prüfen	Konfliktmenge:	ausgewählte Regel:
	<b>AF</b>	<b>H</b>	R7	R7
R7	AFH	G	<b>R2, R3,</b>	R2
<b>R2</b>	AFHG	D	R1	R1
<b>R1</b>	<b>AFHGD</b>	BC	R4, R5	<b>R4</b>
<b>R4</b>	AFHGDC	B	<b>R5</b>	<b>R5</b>
<b>R5</b>	AFHGDCB		<b>0</b>	<b>0</b>

==> STOP

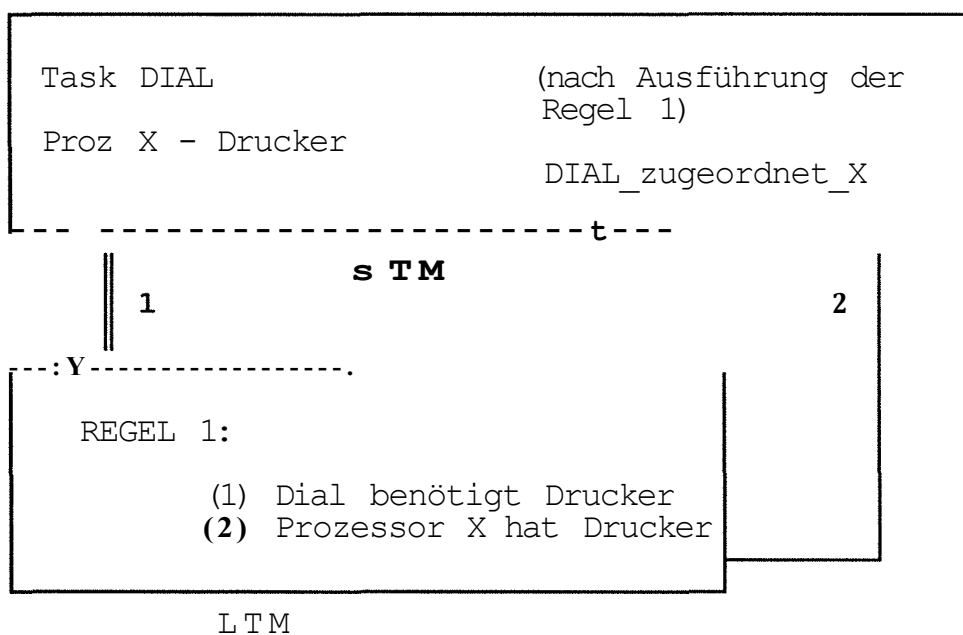
### 7.4.3. Regeln beim Allokationssystem

Die Darstellung der Regeln beim Allokationssystem wird in Kapitel 8 ausführlich behandelt. Deshalb sei hier nur an einem Beispiel gezeigt, wie die (Produktions-) Regeln beim Allokationssystem aussehen können. Für das Allokationssystem wird die datengesteuerte Regelverarbeitung angewandt.

#### REGEL 1:

IF           (1)    die Task DIAL die Hardware DRUCKER benötigt  
              & (2)   und ein Prozessor X existiert, der die gewünschte Hardware  
                          DRUCKER besitzt,  
THEN           Ordne die Task DIAL dem Prozessor X zu

Anhand dieser Regel wird in Abbildung 7.5. gezeigt, wie die Fakten im Arbeitsspeicher vor und nach der Ausführung der Regel aussehen.



1 = Fakten und Regeln in Übereinstimmung bringen

2 = Ausführen der Regel

Abb.: 7.5. Ausführen einer Regel

## 7.5. Expertensysteme

Wie schon eingangs erwähnt, sind Expertensysteme eine Ausprägung der wissensbasierten Systeme. Sie werden als Anwendung von KI-Methoden ("Special-Purpose-Systeme") betrachtet, und gewinnen immer mehr an Bedeutung. Ein Expertensystem soll die Vorgehensweise eines menschlichen Experten (möglichst exakt) simulieren. Das Wissen und die Verarbeitungsstrategie eines Experten werden isoliert und in der Wissensbasis dieser Systeme implementiert. Die wichtigsten Merkmale von Expertensystemen lassen sich wie folgt zusammenfassen:

- Nach Waterman /WATE86/ findet bei Expertensystemen eine Wissensbasis Verwendung, die aus Fakten und Regeln besteht. Die Regeln repräsentieren in dieser Interpretation explizit hinterlegtes "Expertenwissen".
- Die Wissensbasis wird von einer Inferenzmaschine "verarbeitet". Zu dieser Maschine gehören ein Interpreter und ein Scheduler. Der Interpreter wendet kontextabhängig die in der Wissensbasis hinterlegten Regeln (Heuristiken) an, während der Scheduler abwägt, wann und in welcher Ordnung verschiedene Teile der Wissensbasis anzuwenden sind (Metawissen).

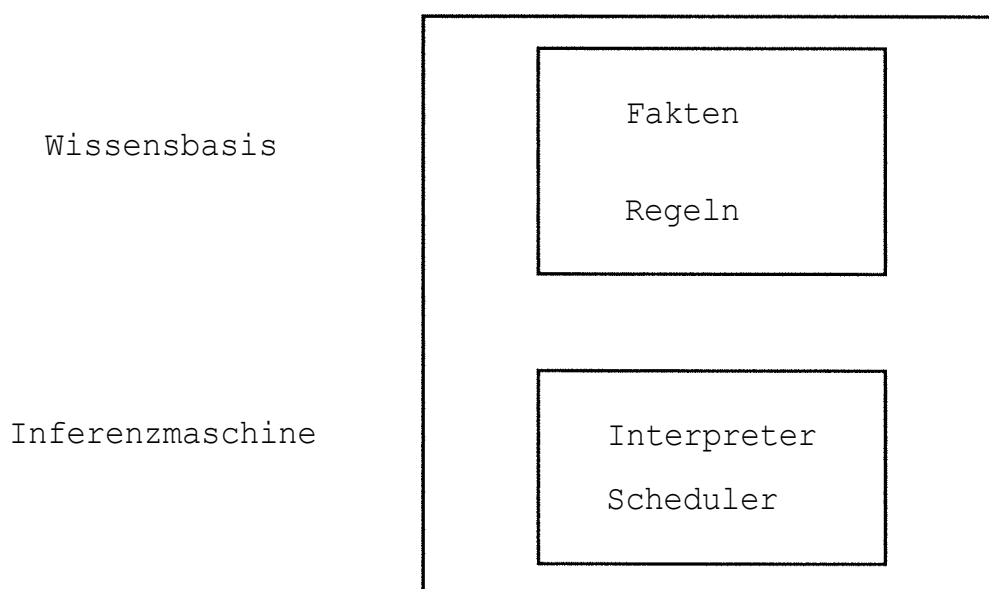


Abb.: 7.6. Systemkern eines Expertensystems

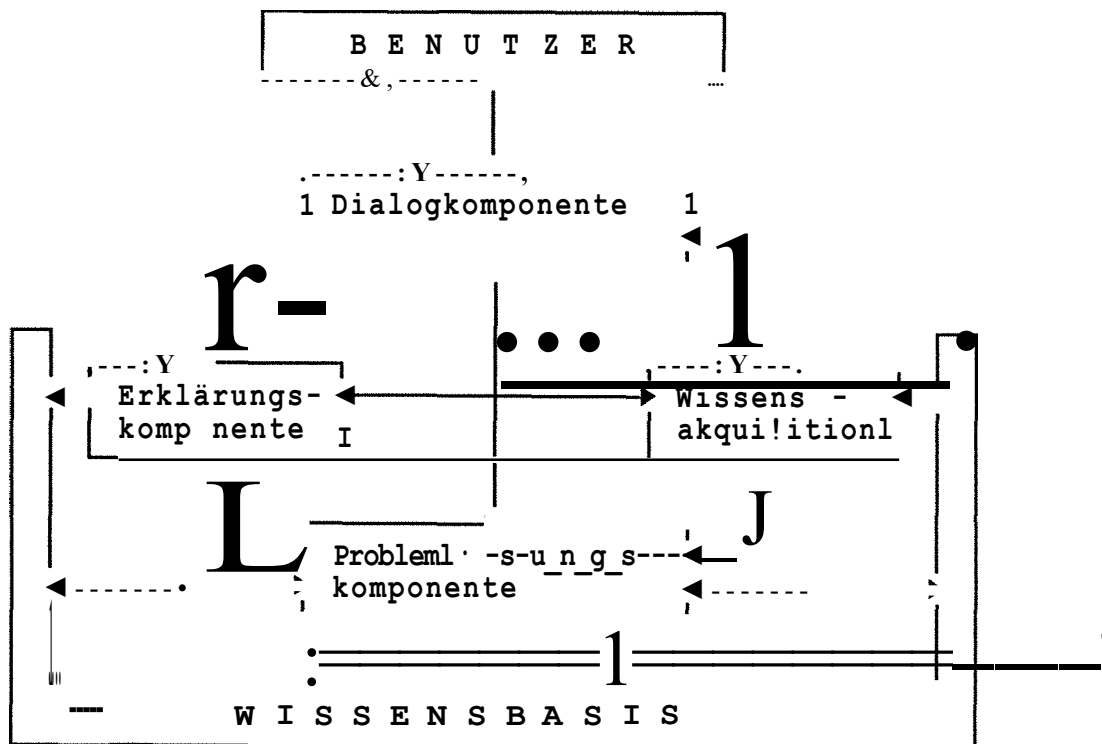


Abb.: 7.7. Schematischer Aufbau eines Expertensystems

Den wichtigsten Teil eines Expertensystems stellt die Problemlösungskomponente (zusammen mit der Wissensbasis) dar. Sie besteht *im* wesentlichen aus dem in Abbildung 7.6. angegebenen Systemkern. Damit das System mit der Außenwelt kommunizieren kann, sind eine Reihe weiterer Komponenten notwendig:

- Die Erklärungskomponente begründet die von der Problemlösungskomponente erarbeitete Lösung. Dieser Teil eines Expertensystems hat die Aufgabe, die gefundene Lösung (bzw. den Lösungsweg) für den Benutzer des Systems nachvollziehbar zu machen. Da das Optimum (meistens) nicht gefunden wird, ist diese Komponente wichtig, um die Ergebnisse transparent zu machen.

- Die Wissensakquisitions-Komponente unterstützt den Systemarchitekten bei seiner Aufgabe, das Wissen des Experten zu erfragen, indem beispielsweise auf Widersprüche in der Wissensbasis aufmerksam gemacht wird.
- Die Dialogkomponente dient als Schnittstelle des Systems zum Benutzer. Aufgabe dieser Komponente ist die Befragung des Benutzers, falls die Problemlösungskomponente zusätzliche Information benötigt.

Um ein bestehendes Problem mit Hilfe eines Expertensystems zu lösen, sollten nach /WATE86/ folgende Bedingungen erfüllt sein:

- nur eine heuristische Lösung führt zum Ziel
- das Problem ist nicht trivial
- es besteht ein praktischer Nutzen
- das Problem hat "machbare" Komplexität

Um die wichtigsten Teile eines Expertensystems verständlicher zu machen, wird ein bestehendes System vorgestellt.

### **7.6. Expertensystem - ein Beispiel**

Beispielhaft wird das System R1 /DERM80/ zur Konfigurierung von Rechenanlagen des Typs VAX-11/780 vorgestellt. Dieses System erstellt aufgrund bestimmter Kundenwünsche Diagramme einer speziellen Rechnerkonfiguration. Nachdem ein Kunde dem System (R1) einen Auftrag übermittelt hat, wird dieser zunächst auf Vollständigkeit überprüft. Falls der Auftrag nicht vollständig ist, werden ggf. Komponenten hinzugefügt.

Das System R1 eignet sich besonders deshalb als Beispiel, weil es als Konfigurationssystem eine ähnliche Aufgabenstellung wie das Allokationssystem bearbeitet. Beim Allokationssystem werden die Tasks auf die Prozessoren verteilt, was man ebenfalls als eine Art Konfigurierung ansehen kann. Für die Konfiguration einer Rechenanlage sind (im vorliegenden Fall) zwei Arten von Wissen notwendig:

- Wissen über jede Komponente, die der Benutzer anfordern kann. Für

jede dieser Komponenten müssen die wichtigsten Spezifikationen (z.B. Größe, Spannung..) bekannt sein.

- Wissen darüber, wie einzelne Komponenten zusammengefügt werden dürfen und welche Randbedingungen einzuhalten sind (z.B. Einhalten der Prioritäten bei der Vergabe von Bus-Steckplätzen).

### Wissensrepräsentation

Das Expertenwissen wurde im System RI mit ca. 800 Regeln (in OPS5 /FORG81/ ) repräsentiert. In der Wissensbasis ist jede der etwa 400 Komponenten beschrieben. In Abbildung 7.8. wird beispielhaft ein Eintrag in der Wissensbasis gezeigt. Abbildung 7.9. beschreibt (umgangssprachlich) eine Regel.

RK611

```

CLASS:          BUNDLE
TYPE:           DISK DRIVE
SUPPORTED:      YES
COMPONENT LIST:3  G727
                  I  M9202
                  I  070-12412-00
                  1  RK611*
```

Abb.: 7.8. Eintrag in der Wissensbasis (im System RI)

```

WENN:           ein MASSBUS-Gerät zugeordnet werden soll
                UND es ein Plattenlaufwerk (mit Anschluß) gibt, das noch keinem
                   MASSBUS zugeordnet wurde
                UND es noch freie Plattenlaufwerke gibt
                UND bekannt ist, wieviele Geräte jeder MASSBUS unterstützen soll
                UND es einen MASSBUS gibt, der bereits (mindestens) ein Platten-
                   laufwerk besitzt und der weitere Laufwerke unterstützen sollte
                UND der Kabeltyp zur Verbindung des Laufwerkes mit dem vorherigen
                   Plattenlaufwerk vorhanden ist
DANN            ordne das betrachtete Laufwerk dem MASSBUS zu
```

Abb.: 7.9. Regel im System RI

Das System RI ist im Vergleich zum entwickelten Allokationssystem ALLOC sehr einfach strukturiert. Es existieren eine Reihe von Regeln, die die Zusammenstellung der Komponenten steuern und weitere Regeln, die sich mit der Einhaltung der Randbedingungen beschäftigen. Die Einteilung des Wissens in Bereiche ist in Abbildung 7.10. dargestellt.

Kontexterzeugung (96 Regeln)	Generierung der Ausgabe (106 Regeln)
Vorbedingungen (127 Regeln)	Kontextabbau (84 Regeln)
Komponentenverbindung (156)	Zählen (54 Regeln)
Retrieval (54 Regeln)	Aufbau (48 Regeln)
Zusammenstellung (47 Regeln)	

Abb.: 7.10. Aufteilung der Regeln im System RI

Das Wissen beim Systems RI zerfällt in vier Klassen:

- Wissen über die verschiedenen Eigenschaften der VAX-Komponenten und darüber, wie man Information über die Komponenten aus der Wissensbasis erhält.
- Wissen, wie man Teilkonfigurationen erkennt und zusammenfügt.
- Wissen darüber, wie man verschiedene Arten der Berechnung erstellt und benutzt.
- Wissen darüber, welche Aktionen innerhalb verschiedener Kontexte angemessen sind und welche Kontexte man von anderen aus ansprechen kann.

Am Beispiel des Systems RI sollte gezeigt werden, wie in einem wissensbasierten System (Expertensystem) isoliertes Wissen in Strukturen für die Wissensrepräsentation umgesetzt werden kann.

Analog zum System RI benutzt das Allokationssystem ALLOC Regeln zur Darstellung von Wissen, jedoch werden zusätzlich eine Reihe weiterer Strukturen notwendig (siehe Kapitel 9).

### 7.7. Zusammenfassung

Wissensbasierte Systeme (Expertensysteme) repräsentieren Wissen über ein Problem, mit dem "intelligentes" Verhalten in bestimmten Situationen simuliert wird. Die Repräsentierung des Wissens erfolgt durch eine klare Trennung in Bereichswissen und dem zur Verarbeitung notwendigen Wissen (Metawissen). Auch in herkömmlichen Systemen ist Wissen in den realisierten Algorithmen repräsentiert. Im Gegensatz dazu sind wissensbasierte Systeme klarer strukturiert (Wissen über das Problem (Fachwissen), Meta-Wissen). Aufgrund dieser (architektonischen) Aufteilung eignen sich wissensbasierte Systeme besonders, um in relativ kurzer Zeit eine (partielle) Simulation des Experten zu besitzen (rapid prototyping). Außerdem ist es leicht, die Wissensbasis einer veränderten Problemstellung anzupassen:

- Um Änderungen an der explizit abgelegten Wissensbasis auszuführen, müssen nur Regeln verändert, neue Regeln zusätzlich in das System eingebracht oder alte Regeln gelöscht werden. Auf diese Art kann das Faktenwissen sehr schnell aktualisiert werden.
- Das Meta-Wissen kann dadurch modifiziert werden, daß die Abarbeitungsstrategien der Inferenzmaschine verändert werden. In einigen Systemen (z.B. KEE als Entwicklung von /INTEL85/, LOOPS von /BOBR83/) können Regeln in Regelmengen unterteilt und kontextabhängig strukturiert werden. Diese Regelmengen können verschiedene Verarbeitungsstrategien aktivieren, so daß die Inferenzmaschine selbst nicht mehr geändert werden muß. Hier kann also über die Repräsentation von Kontrollwissen direkt Einfluß auf die Aktivierung von Regeln (Regelmengen) genommen werden.



Die Technik (prädikatenlogische Wissensrepräsentation, Frames, Regeln...), mit der das vorhandene Wissen repräsentiert wird, ist vom zu lösenden Problem abhängig. Entscheidend für die Qualität eines wissensbasierten Systems ist aber nicht die verwendete Technik; vielmehr ist hier die vollständige Erforschung des (Experten-) Wissens notwendig, die den Grundstein für eine akzeptable Problemlösung darstellt.

## **Kapitel 8**

### **Wissensbasierte Lösung des Allokationsproblems**

#### **8. Wissensbasierte Lösung des Allokationsproblems**

Problemlösungen aus dem Bereich der künstlichen Intelligenz sollen mehr oder weniger "natürliche Vorgehensweisen" besitzen. Dies bedeutet, daß der Lösungsweg dem menschlichen Problemlösungsverhalten angepaßt sein sollte und selbst in schwierigen Situationen zum Ziel führen sollte.

Wie aber geht ein Experte bei der Allokation vor und welches Wissen benutzt er zur Lösung des Problems?

##### **8.1. Das Expertenwissen bei der Allokation**

Um das Expertenwissen bei der Allokation geeignet zur Verarbeitung in einem Rechner aufzubereiten, muß zunächst dieses Wissen dem Ersteller des Allokationssystems verfügbar gemacht werden.

Das Wissen der Experten (/NIEM87/, /BEIE88/, /TRAU87/, /BES085/), das bisher akzeptable Lösungen des Allokationsproblems bei Realzeitaufgaben ermöglichte, wird erfragt und in Form einer Stichpunktsammlung aufgelistet. Danach werden diese "Gedanken" geordnet, und eine Repräsentation vorgeschlagen, die als Wissensbasis für das Allokationssystem dient. Anhand eines Teilschrittes bei der Allokation wird exemplarisch gezeigt, wie das Expertenwissen in die Wissensbasis umgesetzt wird.

Eine (ungeordnete) Ansammlung der geäußerten Expertenmeinungen soll zunächst einen Überblick über das Wissen geben:

- Allgemeine Aussagen über die Verteilung:
  - Die Allokationsinformation soll auf Spezifikationsniveau bereitgestellt werden, da die Verteilung an dieser Stelle der (Realzeitsystem-) Entwicklung am leichtesten zu ersehen und darzustellen ist.
  - Zur Darstellung der Allokation soll nach Möglichkeit eine Methode verwendet werden, mit der auch die Anforderungen beschrieben werden. Da die "Realzeitumgebung" die Allokation beeinflusst, erscheint es sinnvoll, die gesamte Realzeitspezifikation mit einer einheitlichen Methode zu beschreiben.
  - Das Allokationssystem soll nach verschiedenen Kriterien optimieren, da die Aufgabenstellungen aus der Praxis nicht ausschließlich eine Minimierung der IPC-Kosten erfordern.
- Parameter, die eine Verteilung beeinflussen:
  - Die Randbedingungen (z.B. Anzahl der peripheren Geräte an einem Prozessor) müssen bei der Verteilung berücksichtigt werden.
  - Netzwerkparameter (Topologie des Netzwerkes, Leitungsgeschwindigkeit..) sollen berücksichtigt werden.
  - Die Anzahl der zur Verfügung stehenden Prozessoren kann die Verteilung beeinflussen.
  - Eine Vorgabe zur Bindung von Tasks an Prozessoren muß möglich sein.
  - Funktionale Zusammenhänge zwischen Tasks sollen berücksichtigt werden (laut /NIEM87/ erfüllen mehrere Tasks gemeinsam eine bestimmte Aufgabe und sollen deshalb auf einem gemeinsamen Prozessor liegen).
  - Die Art der Prozessoren (homogenes/heterogenes System von Prozessoren) ist für die Verteilung von besonderem Interesse. Das gleiche gilt für die Art des verwendeten Netzwerkes.
  - Die Redundanz von Tasks muß berücksichtigt werden.

- 
- Vorgehensweise bei der Verteilung:
    - Der Lösungsraum ( $n^m$ ) muß in geeigneter Weise eingeschränkt werden. Dazu sind beispielsweise die folgenden Kriterien hilfreich:
      - Tasks, die harte Randbedingungen erfüllen (z.B. Tasks, die fest einem Prozessor zugewiesen sind) werden als erste zugeteilt.
      - "Unmögliche" Zuteilungen werden ausgeschlossen (z.B. Zuteilungen, die aufgrund der Peripherieanforderungen nicht erfüllbar sind).
      - Für manche Tasks kann die Zuteilungsfreiheit eingeschränkt werden (z.B. für Tasks, die eine bestimmte Anforderung haben, die nicht von allen Prozessoren erfüllt werden kann).
    - Für die Zuweisung der Tasks an die Prozessoren haben sich prinzipiell zwei unterschiedliche Verfahren herausgebildet:
      - Die Tasks werden in Cluster eingeteilt, die dann den Prozessoren zugeteilt werden.
      - Die Tasks werden gleich den Prozessoren zugeteilt.
    - Nach Abschluß der Zuweisung wird versucht, die entstandene Lösung noch zu verbessern.
  - Ziele der Verteilung:
    - Ziel der Verteilung ist es, eine gestellte Allokationsaufgabe im Rahmen der Realzeitanforderungen zu lösen. Dazu sollen verschiedene Optimierungskriterien behilflich sein:
      - Optimale Kommunikationskosten,
      - Optimale Auslastung der Prozessoren und
      - Optimale Auslastung des Gesamtsystems aufgrund der relativen Prozessorlaufzeiten.
    - Außerdem wird geprüft, ob mit weniger (mehr) Prozessoren eine "bessere" Verteilung erreicht werden kann.

Die hier aufgeführte Stichpunktsammlung erhebt keinesfalls den Anspruch auf Vollständigkeit; vielmehr sollte versucht werden, einen Überblick über Gedankengänge eines Allokationsexperten zu geben. Aufgrund der Expertenaussagen wurde eine erste Einteilung des Allokationswissens in Verarbeitungsschritte erkennbar:

A) Auffinden von Zuordnungseinschränkungen

Da ein Experte nicht alle Verteilungsmöglichkeiten mit  $n \geq m$  ( dies sind  $n^m$  Möglichkeiten) betrachten kann, sucht er nach geeigneten Einschränkungen. Meist ergeben sich solche Einschränkungen schon aus der Aufgabenstellung. Eine feste Task-Prozessor-Zuordnung ergibt sich beispielsweise dann, wenn eine Task mit einem peripheren Gerät arbeiten soll, das nur an einem Prozessor vorhanden ist. Andere Einschränkungen ergeben sich aus der Spezifikation (z.B. müssen bei Montagestraßen die Steuerprozesse für einzelne Fertigungsroboter an die Steuerungsrechner verteilt werden, während die Überwachungsfunktionen von größeren Rechnern übernommen werden).

Für die Verteilung verbleiben nunmehr  $n^m - k$  Möglichkeiten, wobei  $k$  die Anzahl der bereits fest zugewiesenen Tasks bezeichnet.

B) Tasks auf die Prozessoren verteilen

Mit dieser "Anfangsverteilung" einiger Tasks ist das Allokationsproblem (je nachdem, wie viele Tasks schon eine feste Zuteilung erhalten) bei Realzeitaufgaben bereits reduziert. Die "übriggebliebenen" Tasks werden mit Hilfe geeigneter Verteilungsstrategien auf die Prozessoren aufgeteilt. Dazu werden zunächst die Tasks nach bestimmten Kriterien in Cluster zusammengefaßt (eines der wichtigsten Kriterien zur Zusammenfassung sind die schon erwähnten Kommunikationskosten). Anschließend werden diese Cluster auf die Prozessoren verteilt. Die Verteilungsstrategie, die dazu verwendet wird, ist von entscheidender Bedeutung, da sie den größten Einfluß auf die Qualität der Verteilung hat. Bei der Verteilung der Tasks muß versucht werden, das Optimierungskriterium zu erfüllen (z.B. Optimale Auslastung der Prozessoren), das vom Benutzer vorgegeben wird.

### C) Lösung verbessern

Die gefundene Lösung wird nach Abschluß der Zuteilung noch verbessert. Die Optimierungskriterien, die Kommunikationskosten und die einzuhaltenden Randbedingungen dienen als Hilfsmittel zur Bewertung der Lösung. Von den Experten wurden als Optimierungskriterien, neben einem optimalen Kommunikationsverhalten des verteilten Realzeitsystems, vor allem die gleichmäßige Auslastung der Prozessoren in Bezug auf den Speicherplatz als Kriterium genannt. Von einigen Experten (z.B. /BEIE88/) wurde das Gesamtverhalten des Systems in Bezug auf die Laufzeit als Optimierungskriterium in den Vordergrund gestellt.

Diese Einteilung in drei Abschnitte stellt eine sehr grobe Einteilung in Arbeitsphasen der Experten dar. Ausgehend von dieser Strukturierung wird nun versucht, das Gerüst durch zusätzliches Expertenwissen zu ergänzen. Dazu wird zunächst dieses Wissen weiter erforscht.

## 8.2. Aufbereitung des Wissens

### 8.2.1. Zuordnungseinschränkungen

In verschiedenen Phasen des Allokationsalgorithmus schränken die Randbedingungen, die sich aus den Realzeitanforderungen ableiten, die Entscheidungsfreiheit für die Verteilung ein, bzw. geben wichtige Hinweise für die Zuteilung der Tasks an die Prozessoren.

Bei der Allokation führen folgenden Kriterien zu Einschränkungen:

- Anzahl und Art der peripheren Geräte.
- Hardware-Ausstattung der Prozessoren (z.B. Größe des Hauptspeichers).
- Eigenschaften des Netzwerkes.
- Funktionale Einschränkungen, die sich aus der Spezifikation ergeben.

Bei der Einteilung der Tasks in Gruppen (z.B. durch Clusteranalyse) wird in der Literatur nur eine Zusammenfassung aufgrund der Kommunikationskosten vorgenommen. Die Randbedingungen werden meist überhaupt nicht oder nur in einem gesonderten Teil des Algorithmus berücksichtigt. Ein menschlicher Experte jedoch betrachtet Randbedingungen in jeder Phase der Verteilung.

Die unterschiedliche Wirkung, die Randbedingungen auf die Verteilung haben, wird im wissensbasierten Ansatz durch entsprechende Regeln (in jeder Phase der Verteilung) berücksichtigt.

#### Botschaften zwischen Prozessen

Aufgrund der zwischen Prozessen ausgetauschten "Menge" an Botschaften können wichtige Aussagen über die Güte der Verteilung getroffen werden. Dem Experten dienen diese Kommunikationskosten als Entscheidungsgrundlage, ob Prozesse auf demselben oder auf getrennten Prozessoren installiert werden müssen. Für die vorliegende Arbeit werden die Kommunikationskosten dazu benutzt, um die

- Einteilung der Prozesse in Cluster zu ermöglichen
- die Verteilung der Cluster auf die Prozessoren vorzunehmen,
- die Gesamtkosten des Systems zu minimieren und
- die Optimierung zu unterstützen.

Die Häufigkeit, mit der Tasks in einem System kommunizieren und die daraus resultierenden Kosten für die Botschaften müssen vom Entwerfer eines Realzeitsystems vorgegeben werden. In der Realzeitprogrammierung ist es aufgrund von Ereignissen, die vom zu steuernden System ausgelöst werden können, prinzipiell sehr schwierig, exakte Angaben über das Kommunikationsverhalten zu machen. Für die vorliegende Arbeit können die Kommunikationskosten mit Hilfe der in /ANDRSS/ entworfenen Methode ermittelt werden. Mit dieser Methode ist es möglich, das zeitliche Verhalten des Realzeitsystems zu simulieren. Anhand der Simulationsergebnisse können Rückschlüsse auf das Kommunikationsverhalten gewonnen werden.

### Prozeßperipherie

Wichtige Einschränkungen bei der Verteilung von Realzeitsystemen ergeben sich aus der Kommunikation einer Task mit einem technischen Prozeß, bzw. der Prozeßperipherie. Da Realzeitsysteme technische Prozesse steuern, gibt es Tasks, die mit der Prozeßperipherie (und dadurch mit dem technischen Prozeß) Botschaften austauschen. Solche Tasks unterliegen der Einschränkung, daß sie dem Prozessor zugewiesen werden müssen, an den die Prozeßperipherie angeschlossen ist. Um die Prozeßperipherie geeignet in die Verteilung mit aufnehmen zu können, soll die Kommunikation zwischen "technischen" Prozessen und Tasks, sowie die Kommunikation zwischen Tasks mit der gleichen Spezifikationsmethode beschrieben werden. Dies ist mit der Spezifikationsmethode PASS möglich, wodurch eine geschlossene Darstellung des Hard- und Software-systems erreicht wird. In /BESO85/ wurde die Anwendbarkeit dieser Spezifikationsmethode auf technische Prozesse am Beispiel eines Zählerbausteines demonstriert.

### Redundanz als Strukturierungsmittel für Ausfallsicherheit

Neben dem Zugang zur Prozeßperipherie schränkt die Redundanz von Tasks die Zuteilung weiter ein, da redundante Tasks immer auf verschiedenen Prozessoren installiert werden müssen. Folglich gilt:

- redundante Tasks können nur auf redundant vorhandenen Prozessoren zu liegen kommen
- Inkarnationen einer redundanten Task dürfen nie auf demselben Prozessor liegen.

Redundante Tasks erzwingen also (wie die Prozeßperipherie) eine Einschränkung der Zuweisung von Tasks an die Prozessoren.

### Speicherplatzbedarf

Bei der Verteilung spielt der Speicherplatzbedarf einer Task (Taskgröße) dann eine Rolle, wenn es darum geht, die gleichmäßige Auslastung der Prozessoren (bzw. die Überlastung einzelner Prozessoren) zu überwachen.



Die Taskgröße kann Aufschluß darüber geben,

- ob ein Prozessor mit den ihm zugeordneten Prozessen überlastet ist und
- ob eine gleichmäßige Auslastung aller Prozessoren erreicht wurde.

Für den Experten ist die Größe der Tasks somit ein wichtiges Kriterium, das er bei seiner Entscheidungsfindung über eine Verteilung berücksichtigt.

### Laufzeiten der Tasks

Ebenso wie die gleichmäßige Auslastung der Prozessoren ist auch die Zeit, in der ein Prozessor "beschäftigt" ist, ein Entscheidungskriterium für die Verteilung.

Im Kommunikationsdiagramm wird deshalb (neben der Taskgröße) der Laufzeitbedarf einer Task angegeben. Unter dem Laufzeitbedarf einer Task soll die Zeit verstanden werden, die diese Task tatsächlich aktiv ist. Realzeitsysteme sind Multitasking-Systeme, d.h. ein Prozessor bearbeitet mehrere Tasks. Um die Auslastung des Prozessors zu bestimmen, muß die Rechenzeit (Laufzeitbedarf) der einzelnen Tasks festgestellt werden.

Wie bei der Angabe des Speicherplatzbedarfs der Tasks die gleichmäßige Speicherauslastung der beteiligten Prozessoren überwacht wird, kann mit der Angabe des Laufzeitbedarfes einer Task die gleichmäßige Auslastung der Prozessoren in zeitlicher Hinsicht erreicht werden. Unter der gleichen zeitlichen Auslastung der Prozessoren soll verstanden werden, daß die Prozessoren in einem bestimmten Zeitintervall alle gleich viel beschäftigt sind. Damit soll verhindert werden, daß ein Prozessor mit Arbeit "überlastet" ist, während ein anderer nichts zu tun hat.

### Task-Prozessor-Zuordnungsgebote und -verbote

In einem Realzeitsystem wird es immer Tasks geben, die der Benutzer einem Prozessor fest zuweisen möchte, bzw. solche, denen die Zuteilung zu bestimmten Prozessoren verboten werden soll.

Ein Grund für ein Task-Prozessor-Zuordnungsgebot kann beispielsweise darin bestehen, daß ein Prozessor bestimmte "Fähigkeiten" (z.B. Coprozessor für Graphik) besitzt. Diesem Prozessor wird eine Task dann fest zugewiesen,

wenn sie unbedingt diese Funktion für ihre Berechnungen benötigt.

Analog zu Geboten gibt es auch Task-Prozessor-Verbote:

Eine Task soll mit der Aufgabe Daten vom Netz entgegenzunehmen auf einem Rechner zur Datenaufnahme liegen, nicht aber auf dem Auswertungsrechner, der keinen Zugang zum Netz hat. Für den Auswertungsrechner wird dieser Task ein Zuordnungsverbot erteilt.

Diese Zuordnungskriterien schränken das Problem der Allokation ein. Die Anzahl der Tasks, die frei verteilbar sind, reduziert sich, wodurch weniger Allokationsmöglichkeiten betrachtet werden müssen. Deshalb ist es sinnvoll, Gebote und Verbote bereits zu Beginn der Verteilung zu betrachten.

#### Task-Task-Zuordnungsgebote und -verbote

Ähnlich wie bei Task-Prozessor-Beziehungen kann es für zwei Tasks Zuordnungsgebote bzw. -verbote geben. Anders als die Task-Prozessor-Beziehungen beinhalten die Task-Task-Beziehungen allerdings keine feste Zuordnung (bzw. Verbot der Zuordnung) an einen Prozessor.

Bei der Beziehung zwischen zwei Tasks handelt es sich um eine schwächere Forderung. Es wird nur festgelegt, daß zwei Tasks gemeinsam bzw. getrennt einem Prozessor zugewiesen werden sollen. Eine Zuweisung an einen bestimmten Prozessor wird damit aber noch nicht festgelegt. Es könnte sich jedoch eine Einschränkung der Zuweisung (z. B. durch Peripheriebindung einer der beiden Tasks) ergeben. Für ein Task-Task-Verbot können beispielsweise folgende Gründe vorhanden sein:

- Funktionale Gründe verbieten eine Zuweisung von zwei Tasks an denselben Prozessor (Eine Task soll ununterbrechbar ablaufen, während eine andere Task Interruptverarbeitung benötigt).
- Es handelt sich um redundante Prozesse.

Funktionale Gründe können auch die Ursache eines Task-Task-Gebotes sein (z.B. die Tasks arbeiten "gemeinsam" an einer Aufgabe /NIEM87/).

Aus den Einschränkungen, die durch Task-Task-Beziehungen festgelegt werden, ergeben sich keine festen Zuteilungen von Tasks an Prozessoren. Dennoch wird das Problem der Allokation durch diese zusätzliche Information reduziert, da die Allokationsfreiheit solcher Tasks eingeschränkt wird.

### Netzwerktopologie

Von entscheidender Bedeutung für die Verteilung ist die Topologie des Netzwerkes, das die Inter-Prozessor-Kommunikation realisiert. Die Art der Verbindungen zwischen den Prozessoren beeinflusst die Allokation sehr stark.

- Moderne Netzwerke (lokale Netzwerke -z.B. Ethernet- oder sogenannte Wide Area Netzwerke -z.B. X.25-) benutzen Routing-Verfahren, bei denen jeder Teilnehmer im System adressiert werden kann. In diesem Fall liegt ein vollständig vermaschtes System zugrunde.
- Andererseits werden bei Realzeitsystemen auch Kopplungen über direkte Verbindungen benötigt. Bei direkten Verbindungen kann es Netzwerktopologien geben, bei denen nicht alle Partner miteinander verbunden sind.

Für die Verteilung bedeutet dies, daß sowohl vollständig vermaschte Systeme, als auch Systeme mit Einzelverbindungen berücksichtigt werden müssen.

Vollständig vermaschte Systeme haben in Bezug auf die Topologie keinen Einfluß auf die Verteilung. Bei Einzelverbindungen hingegen muß darauf Rücksicht genommen werden, welche Prozessoren miteinander verbunden sind. Hier wird der Aufwand der Allokationsalgorithmen erheblich vergrößert.

Zusammenfassend läßt sich feststellen, daß die Randbedingungen wichtige Hilfsmittel für den Experten bei der Allokation darstellen. Es ist völlig unzureichend, Tasks in einem homogenen System zu betrachten und aufgrund einer (nur auf einer Metrik zur Einteilung basierenden) Clusteranalyse für diesen Spezialfall der Verteilung ein Ergebnis zu errechnen. Vielmehr müssen für den Einsatz eines Allokationssystems in der Praxis die Randbedingungen geeignet in die Verteilung miteinbezogen werden. Dies bedeutet konkret, daß ein Allokationssystem alle derartigen Restriktionen bei der Verteilung berücksichtigen muß. In jeder Phase eines Allokationssystems müssen die Randbedingungen Einfluß auf die Verteilung nehmen.

Am Beispiel der Clusteranalyse wird im nächsten Abschnitt gezeigt, wie diese Forderung aus der Praxis der Allokation umgesetzt werden kann. Dabei wird deutlich, wie stark die Randbedingungen die Zerlegung eines Allokationssystems beeinflussen.

### 8.2.2. Verteilung der Tasks auf die Prozessoren

Die oben erwähnten Einschränkungen sind gewichtige Einflußfaktoren, die den Aufwand für die Allokation verringern. Dennoch bleibt die Zuteilung (bzw. die Verteilungsstrategie) der "übriggebliebenen" Tasks an die Prozessoren ein wesentlicher Bestandteil der Allokation. Nach den Expertenmeinungen sind unter anderem die folgenden Verteilungsstrategien für die Allokation relevant:

- Einteilung der Tasks in Cluster mit dem Ziel, die Kommunikationskosten zwischen den Clustern zu minimieren. Die Zuweisung der erzeugten Cluster an die Prozessoren erfolgt dann in Abhängigkeit von der Größe der Kommunikationskosten der Cluster.
- Zuweisung der Tasks an die Prozessoren in Abhängigkeit von den Kommunikationskosten oder abhängig vom Speicherbedarf der Tasks.
- Zuweisung der Tasks an die Prozessoren nach der Größe ihres Anteils an den Gesamtkommunikationskosten.
- Zuweisung der Tasks an die Prozessoren (unter Berücksichtigung der Netzwerktopologie) in der Reihenfolge der Anzahl der Verbindungen, die ein Cluster besitzt.

Solche Heuristiken dienen bei /BORR86/ als Zielfunktion. Für eine "gute" Allokation reichen aber "einfache" Heuristiken nicht aus; stattdessen benötigt man kombinierte Vorgehensweisen, bei denen verschiedene Strategien "gleichzeitig" die Verteilung beeinflussen. Abhängig von der aktuellen Situation sollen wechselnde Kriterien zur Verteilung beitragen.

So kann beispielsweise die Zuteilung der Tasks unter Berücksichtigung der Kommunikationskosten nur dann -im Sinne der Realzeitprogrammierung- erfolgreich sein, wenn auch die Randbedingungen erfüllt werden. Zudem kann eine "einfache" Verteilungsstrategie leicht in eine Sackgasse führen: Aufgrund einer falsch getroffenen Entscheidung kann man in eine Situation gelangen, von der aus das Optimum (bzw. ein vertretbares Ziel) nicht mehr erreichbar ist. In solchen Fällen muß das System in der Lage sein, einmal getroffene Entscheidungen rückgängig zu machen (Backtracking). Deshalb ist es von entscheidender Bedeutung, daß situationsgerecht die relevanten Verteilungskriterien berücksichtigt werden.

In der vorliegenden Lösung der Allokation kann die Verteilungsstrategie bei Bedarf vom System gewechselt werden. Dies bedeutet, das Allokationssystem erkennt die Konstellation in der ein Strategiewechsel notwendig wird.

Zu Veranschaulichung dient ein einfaches Beispiel:

Anfangs wird das Ziel verfolgt, alle Prozessoren mit mindestens einem Cluster zu belegen. Wenn dies für alle Prozessoren erfüllt ist, wird eine neue Strategie bestimmt. Die Verteilung der nächsten Cluster erfolgt nun unter dem Aspekt der gleichmäßigen Prozessorauslastung.

Auch hier wird der Unterschied zu herkömmlichen Allokationssystemen deutlich: Während bislang die Optimierung eines Systems nach IPC-Kosten im Vordergrund stand, kann in der vorliegenden Arbeit nach verschiedenen Kriterien optimiert werden. Dies entspricht den Anforderungen aus der Realzeitpraxis. Außerdem wird die Vorgehensweise eines Experten bei der Zuteilung nachgebildet. Dies bedeutet, daß nicht nur nach einem Verteilungsalgorithmus vorgegangen wird, sondern daß unter Berücksichtigung der Randbedingungen und unter gleichzeitiger Einhaltung von unterschiedlichen Optimierungskriterien verschiedene Strategien zum tragen kommen.

### 8.2.3. Verbessern der Lösung

Nach Zuteilung aller Tasks an die Prozessoren soll durch eine abschließende "Bewertung" die Verteilung noch verbessert werden. Für eine Verbesserung der erstellten Lösung gelten die gleichen Bedingungen wie für die Verteilungsstrategien. Auch hier kann keine "einfache" Strategie zur Lösung führen. Betrachtet werden müssen vielmehr alle verfügbaren Randbedingungen, die in einer bestimmten Situation gelten. Als Strategie zur Verbesserung der Lösung dienten folgende Empfehlungen von den Experten:

- Verschieben einzelner (oder aller) Tasks von einem Prozessor A auf einen Prozessor B.
- Verschieben ganzer Cluster (falls solche vorhanden).

#### 8.2.4. Optimieren nach verschiedenen Kriterien

Die aufgezeigte Vorgehensweise bei der Allokation zeigt deutlich, daß im Rahmen der Realzeitprogrammierung stärkere Forderungen zu erfüllen sind, als nur eine optimale Verteilung für ein homogenes System von Tasks und Prozessoren in Bezug auf die IPC-Kosten zu errechnen.

Die Zuteilung der Tasks an die Prozessoren wird abhängig von der aktuellen Situation von verschiedenen Kriterien beeinflusst. Nur unter Beachtung aller Verteilungskriterien entspricht die Allokation der Vorgehensweise eines Experten.

Bei der Verteilung von Tasks auf Prozessoren (und bei der Lösungsverbeserung) müssen, den Expertenmeinungen zufolge, neben den IPC-Kosten auch andere Optimierungskriterien zum tragen kommen:

- Gleichmäßige Speicherauslastung aller Prozessoren.
- Gleichmäßige Laufzeitauslastung aller Prozessoren.
- Kombinationen dieser Kriterien.

Zudem soll nicht grundsätzlich von einer festen Anzahl von Prozessoren ausgegangen werden. Falls eine bessere Lösung des Allokationsproblems mit weniger (oder mehr) als den zur Verfügung gestellten Prozessoren möglich ist, sollte diese Möglichkeit vom Allokationssystem berücksichtigt werden. Dadurch ließe sich eventuell erheblicher Kommunikationsaufwand einsparen. Außerdem muß das zugrundeliegende Hardwaresystem als heterogenes System auslegbar sein, um den Anforderungen der Echtzeitprogrammierung gerecht zu werden. Insbesondere sollen die oben erwähnten unterschiedlichen Netztopologien für die Verteilung betrachtet werden können.

Die Optimierung eines Allokationsproblems nach den hier aufgeführten Kriterien sollte (nach Meinung der Experten) nicht erst in einem abschließenden Abschnitt behandelt werden, vielmehr sollte sie alle Berechnungsschritte beeinflussen.

### 8.3. Repräsentation des Wissens

#### 8.3.1. Verfeinerung der Arbeitsschritte

Nach der Untersuchung und Klassifikation des Expertenwissens muß dieses Wissen "nachgebildet" werden, um einem Expertensystem als Wissensbasis zu dienen. Die ursprüngliche Einteilung des Expertenwissens ist zu grob, daher müssen die oben angegebenen Verarbeitungsschritte nach bestimmten Kriterien geordnet werden:

- Die Zuordnungseinschränkungen werden unterteilt in solche, die eine feste Zuordnung erzwingen (z.B. Zuteilung aufgrund der Prozeßperipherie oder eines Benutzerwunsches) und solche, die "nur" eine Einschränkung der Allokationsfreiheit der Tasks bewirken.
- Die Verteilung der Tasks auf die Prozessoren sollte nach Meinung der Experten nicht in einem Schritt erfolgen. Vielmehr sollten zunächst die Tasks in Cluster eingeteilt werden und diese anschließend den Prozessoren zugeteilt werden. (Diese Vorgehensweise wurde auch experimentell nachgeprüft).
- Ebenso wurde die "Verbesserung der Lösung" in die beiden oben erwähnten Verarbeitungsschritte "Verschieben einzelner Tasks" oder "Verschieben ganzer Cluster" unterteilt.

Die Bearbeitung eines Allokationsproblems geschieht nach folgenden Schritten:

- A) Auswerten der festen Randbedingungen.
- B) Auswerten der Zuordnungseinschränkungen.
- C) Einteilen der Tasks in Cluster.
- D) Verteilen der Cluster auf die Prozessoren.
- E) Verschieben einzelner Tasks.
- F) Verschieben ganzer Cluster.

Diese Einteilung stellt sicher, daß die Bearbeitung des Allokationsproblems dem Vorgehen des Experten entspricht. Ein Experte versucht in allen Schritten, das Optimierungsziel der Allokation zu berücksichtigen. Es existieren also keine eigenen Schritte für die Verarbeitung von Randbedingungen und die Optimierung des Systems nach bestimmten Gesichtspunkten.

### 8.3.2. Forderungen an die Repräsentation

Wie kann die Vorgehensweise eines Experten repräsentiert werden?

Dazu werden folgende Forderungen an die Repräsentation erhoben, um die Expertenaussagen darstellen zu können:

- I. Es müssen alle geforderten Verarbeitungsschritte (A-F) in der vorgegebenen Reihenfolge angewandt werden.
- II. Innerhalb der einzelnen Schritte muß das System aufgrund einer aktuellen Situation unter Berücksichtigung der Randbedingungen, der Optimierungskriterien und der aktuell geltenden Strategie, den jeweils besten Lösungsschritt vollziehen. Dazu müssen folgende Teilaspekte berücksichtigt werden:
  - Die Verfolgung globaler Strategien (z.B. Zuteilen der Cluster aufgrund der Kommunikationskosten) muß sichergestellt sein.
  - Die Randbedingungen müssen eingehalten werden (z.B. kann eine fehlende Verbindung zwischen Prozessoren eine Clusterzuteilung unmöglich machen).
  - Es müssen Entscheidungen aufgrund der Bewertung der aktuellen Situation möglich sein.
  - Falsche Entscheidungen müssen zurückgenommen werden können (Backtracking).
  - In Situationen, in denen keine Entscheidung getroffen werden kann, muß der Benutzer die Möglichkeit erhalten, in den Ablauf einzugreifen (z.B. wenn eine Task keinem der Prozessoren zugeordnet werden kann).

Um die hier aufgestellten Forderungen im Rechner nachvollziehen zu können und die Strategien der Experten exakt einzuhalten, wird das Expertenwissen durch die Umsetzung des problemspezifischen Wissens in Phasen, Regelmengen und Regeln modelliert. Um eine korrekte Verarbeitung zu garantieren, muß das Meta-Wissen in der Lage sein das modellierte Wissen richtig "anzuwenden".

Im folgenden wird nun aufgezeigt, wie das gesammelte Expertenwissen bei der Allokation als Basis eines wissensbasierten Systems dienen soll.



### 8.3.3. Repräsentation des Wissens bei der Allokation

#### Problemspezifisches Wissen

Das problemspezifischen Wissens wird aufgrund der aufgestellten Forderungen folgendermaßen strukturiert:

#### **Phasen**

Die Einteilung des Allokationssystems in Verarbeitungsschritte wird durch "Phasen" realisiert. Unter Phasen sollen Strukturierungseinheiten verstanden werden, die sequentiell verarbeitet werden können. Die Reihenfolge, in der die Phasen ausgeführt werden, ist festgelegt und entspricht den vorgeschlagenen Schritten (A bis F; siehe 8.3.) eines Experten. Innerhalb der Phasen sind Regelmengen und Regeln für die Bearbeitung bestimmter Teilaufgaben zuständig. Die Regeln (und die Regelmengen) der einzelnen Phasen sind nur dann ausführbar, wenn die entsprechende Phase aktiv ist. Die Einteilung in Phasen entspricht der Forderung I.

Die Forderung II wird durch die Regelmengen und die Regeln erfüllt, wobei die Regelmengen die zentralen Aufgaben innerhalb einer Phase wahrnehmen. Die Regeln erfüllen "lokale Schritte" innerhalb der Regelmengen. Da die Regelmengen und die Regeln der eigentliche Kern des wissenbasierten Systems sind, sollen sie näher betrachtet werden:

#### **- Regelmengen**

In den Regelmengen werden Regeln zusammengefaßt, die gemeinsam an einer Teilaufgabe arbeiten (z.B. die Auswahl eines Clusters, das als nächstes den Prozessoren zugeteilt werden soll). Ferner wachen diese "Einheiten" über die Einhaltung der jeweils gültigen Strategie und sorgen für die Zurücknahme falscher Schritte, falls dies notwendig ist. Regelmengen repräsentieren die zentralen Teilaufgaben der Allokation, wie sie in der Forderung II im letzten Abschnitt dargestellt wurden.

Die Regelmengen haben (statisch zugeteilte) Prioritäten. Die Ausführungsreihenfolge der Regelmengen ist abhängig von den Prioritäten, der aktuellen Situation des Systems und der Strategie zur Regelauswahl (Inferenzmechanismus).

### - Regeln

Das Expertenwissen wird schließlich durch Regeln repräsentiert. Die Regeln beschreiben das lokale Vorgehen innerhalb der Regelmengen. Eine systematische Abfolge der Regeln gewährleistet die richtige Entscheidung in einer konkreten Situation. Die Reihenfolge, mit der die Regeln aktiviert werden, wird durch Prioritäten gesteuert.

Innerhalb "ihrer" Regelmenge besorgen die Regeln die eigentliche Arbeit. So wachen Regeln über die Einhaltung der Randbedingungen für die Einteilung der Tasks und erkennen die Notwendigkeit für einen Strategiewechsel.

### Meta-Wissen

Das Meta-Wissen gibt an, wie das problemspezifische Wissen zu bearbeiten ist. Dies geschieht dadurch, daß gewisse Schlüsse über den Repräsentationen des problemspezifischen Wissens zugelassen werden.

Für den konkreten Fall des Allokationssystems muß das Meta-Wissen dafür Sorge tragen, daß folgende Schritte eingehalten werden:

- Mit Hilfe des Metawissens muß das Allokationssystem in der Lage sein, die Reihenfolge der Phasen einzuhalten. Dies bedeutet, daß erkannt werden muß, wann eine Phase ihre Arbeit beendet hat und zur nächsten umgeschaltet werden soll.

- Darüberhinaus müssen die jeweils gültigen Regelmengen aktiviert werden. Dazu ist das Metawissen in einen Regelinterpreter und einen Scheduler unterteilt. Der Interpreter wendet kontextabhängig die in der Wissensbasis hinterlegten Regeln an, während der Scheduler abwägt, wann und in welcher Ordnung verschiedene Teile der Wissensbasis gültig sind.

Das Meta-Wissen sorgt durch die Aktivierung der Regelmengen für die Einhaltung der Strategie der Verteilung. Dies wird dadurch sichergestellt, daß die jeweils "beste" Regel zur Ausführung ausgewählt wird und somit die Vorgehensweise des Experten exakt eingehalten werden kann.

Zusammenfassung

Mit der Aufteilung in Phasen, Regelmengen und Regeln ist das problem-spezifische Wissen eines Experten bei der Allokation korrekt repräsentiert. Um die Strategien eines Allokationsexperten bei der Verteilung einzuhalten, muß das Meta-Wissen in der Inferenzmaschine die Strukturen "kennen" und darüberhinaus in der Lage sein, die Abarbeitung der Strukturen in der gewünschten Weise zu vollziehen. Die folgende Zeichnung soll die Aufteilung in Phasen, Regelmengen und Regeln verdeutlichen.

Phase 1

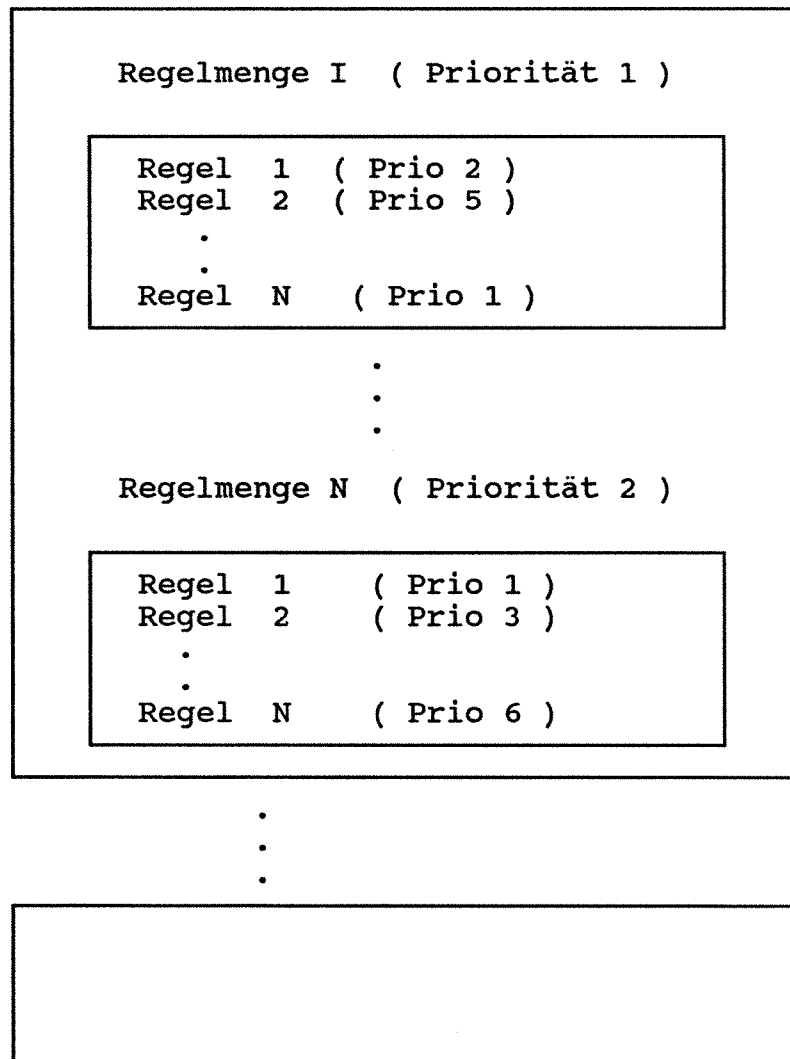


Abb. : 8.1. Repräsentationsschema

### 8.3.4. Wissensrepräsentation am Beispiel der Clusteranalyse

Die Einteilung des Wissens in Phasen folgt der oben erwähnten Gliederung (A-F). Diese prototypisch behandelte Einteilung kann bei Bedarf durch neue Phasen (z.B. konkrete Beschreibung der Hardwarekomponenten) erweitert werden. Innerhalb der einzelnen Phasen werden weitere Teilschritte notwendig, um das Expertenwissen richtig zu interpretieren. Diese Teilschritte werden durch die Regelmengen repräsentiert (siehe Abbildung 8.1.).

Die Zerlegung der Phasen in weitere Teilschritte und die Repräsentation dieser Schritte durch Regeln und Regelmengen wird am Beispiel der Phase C "Einteilen der Tasks in Cluster" (Clusteranalyse) näher erläutert. Die Clusteranalyse bietet sich an, da sie vergleichsweise leicht überschaubar ist. Sie wird näher untersucht, um die Arbeitsweise des Allokationssystems mit seinen Regelmengen und Regeln zu verdeutlichen. Die Wirkungsweise der anderen Phasen kann der Darstellung der Regelmengen und der Regeln in Anhang IV entnommen werden.

#### 8.3.4.1. Kriterien

Algorithmen zur Clusteranalyse /SPÄT75/ gehen normalerweise von der Tatsache aus, daß ein homogenes System zur Einteilung vorliegt und versuchen deshalb eine optimale Einteilung dadurch zu erreichen, daß die Gesamtkosten zwischen den entstehenden Clustern minimal sind.

Die Clusteranalyse hat für die Allokation im Rahmen der Realzeitprogrammierung die Aufgabe, die vorhandenen Tasks aufgrund der Kommunikationsstruktur in zusammenhängende Bereiche (Cluster) zu unterteilen. Dabei ist die Optimierung der Kommunikationskosten allerdings nur ein Kriterium, das zu berücksichtigen ist:

- Die Einteilung soll so erfolgen, daß die Kommunikationskosten zwischen den Clustern minimal werden.
- Darüberhinaus werden die Randbedingungen in die Clusteranalyse mit einbezogen. Die Randbedingungen beeinflussen die Entscheidung, eine Task in ein bestehendes Cluster mit aufzunehmen.
- Die Qualität der Clustereinteilung zeigt sich in der Vorgehensweise, wann Cluster aufgefüllt bzw. neue angelegt werden. Es ist also wichtig zu wissen,

wann ein neues Cluster anzulegen ist. Für die Entscheidung, welchem Cluster eine Task zugewiesen werden soll, können die folgenden Kriterien behilflich sein:

- Allokationsgebot
- Allokationsverbot
- Rechnerüberlast (falls aufgrund der Prozeßperipherie schon in dieser Phase eine Rechnerzuteilung feststeht)
- Task-Prozessor-Zuordnungsverbot
- Task-Prozessor-Zuordnung
- Information aus der Zuordnungseinschränkung
- Kombination aus Allokationsgebot und Allokationsverbot

#### 8.3.4.2. Vorgehensweise

Die Vorgehensweise eines Experten bei der Clustereinteilung wird unter Berücksichtigung der aufgeführten Kriterien in drei Teilaufgaben zerlegt:

##### - Aufgabe I

Falls eine Task aufgrund fester Zuweisung<sup>15</sup> bereits einem Prozessor zugeteilt wurde, wird die entsprechende Task als erste betrachtet. Für solche Tasks sind im allgemeinen neue Cluster anzulegen.

##### - Aufgabe II

Von allen Tasks ohne feste Zuweisung muß eine ausgewählt werden, die als nächste einem Cluster zugewiesen werden soll. Zur Auswahl einer Task können verschiedene Kriterien herangezogen werden (z.B. Kommunikationskosten einer Task mit deren Nachbarn, Anzahl der Verbindungen einer Task).

##### - Aufgabe III

Eine "ausgewählte" Task wird daraufhin untersucht, ob sie einem bereits

---

<sup>15</sup> Eine feste Zuweisung kann eine Task-Prozessor-Zuweisung durch den Benutzer sein, oder eine Zuweisung, die sich aufgrund der Beziehung Task - technischer Prozeß ergibt.

bestehenden Cluster angegliedert werden soll oder ob für die Task ein neues Cluster angelegt werden muß. Dazu sind umfangreiche Prüfungen vorzunehmen, die sich aufgrund der Randbedingungen und der Optimierungskriterien ergeben. Für Tasks, die keine Einschränkungen oder feste Zuteilungen besitzen, muß ein Schwellwert (eine Metrik) berechnet werden, von dem es abhängt, ob ein neues Cluster angelegt wird, oder ein bestehendes erweitert wird. Falls keine Aussage über die Zuteilung gemacht werden kann, wird die betrachtete Task -im Moment- von der Zuteilung zurückgestellt, und eine andere Task vorrangig betrachtet.

Die vorhandenen Tasks sind nach Abschluß der Clusteranalyse in eine Menge von Clustern unterteilt. Die einzelnen Aufgaben (I bis III) müssen bei der Clusteranalyse so zusammenspielen", daß zwischen den erzeugten Clustern minimale Kommunikationskosten entstehen und die Realzeitanforderungen berücksichtigt sind. Die Einteilung in drei Teilaufgaben erwies sich als ausreichend, um eine gute Einteilung der Tasks in Cluster zu erreichen.

#### 8.3.4.3. Umsetzung

Die Vorgehensweise bei der Clustereinteilung wird folgendermaßen umgesetzt:

Jede der vorher aufgezählten Aufgaben (Teilschritte) wird durch eine Regelmengerepräsentiert. Um eine Abarbeitungsreihenfolge (Strategie) einzuhalten, wie sie vom Experten vorgeschlagen wurde, werden die Regelmengen mit Prioritäten versehen. Die Prioritäten ergeben sich bei der Clusteranalyse folgendermaßen:

- Als erstes sollen die Tasks zugewiesen werden, die eine feste Zuteilung haben. (Priorität 1)
- Nach unterschiedlichen Bedingungen (siehe oben) wird eine Task für die Zuweisung ausgewählt (Priorität 2).
- Für die ausgewählte Task wird überprüft, ob der Zuweisungswunsch erfüllt werden kann. (Priorität 3)

Innerhalb der Regelmengen sorgen nun die Regeln dafür, daß die richtige Clustereinteilung "errechnet" wird. Jede Regel verfolgt dabei ein lokales Ziel;

jedoch erst eine systematische Ablauffolge stellt die Clustereinteilung sicher. Diese Folge wird durch die Vergabe von Prioritäten gewährleistet. Die Prioritäten steuern die Reihenfolge der Anwendung der einzelnen Regeln. Soll für eine Task als erstes geprüft werden, ob ein Verbot der Zuteilung vorliegt, muß die Regel, die diesen Sachverhalt repräsentiert, eine hohe Priorität besitzen.

Neben den eigentlichen "Arbeitsaufträgen", die zur Einteilung der Tasks in Cluster notwendig sind, sorgen die Regeln für die Einhaltung der Strategie. Dies soll (für das konkrete Beispiel der Clusteranalyse) an folgendem Beispiel deutlich werden:

- Falls eine Zuteilung (in Regelmenge III) nicht erfolgen kann, muß eventuell in die Regelmenge II "umgeschaltet" werden. Dazu stehen Regeln zur Verfügung, die eine solche Notwendigkeit erkennen.

Mittels der Regelmengen, der Regeln und der jeweils zugeteilten Priorität läßt sich die geforderte Vorgehensweise bei der Clusteranalyse einhalten. Die Tasks werden, unter Berücksichtigung der Kommunikationskosten und der Randbedingungen, in Cluster eingeteilt, wie es der Vorgehensweise des Experten entspricht.

#### **8.3.4.4. Erläuterung der Regelmengen und der Regeln**

Zum besseren Verständnis der Arbeitsweise der Regelmengen und der Regeln werden diese anhand einiger Beispiele präzisiert:

Die angegebenen Regeln sind im folgenden in einer umgangssprachlichen Syntax angegeben und entsprechen der logischen Implikation. Die Bedingungsteile oder Voraussetzungsteile (linken Seiten) der Regeln werden durch das Schlüsselwort *Wenn* eingeleitet. Die Ausführungsteile (rechten Seiten) beginnen mit dem Wort *Dann*. Die logischen Aussagen der linken Seiten sind numeriert und mit "&" verknüpft. Die Prioritäten der Regeln (bzw. der Regelmengen) sind jeweils angegeben. Die Priorität einer Regel ist umso höher, je kleiner die angegebene Zahl ist. Für jede Regelmenge werden beispielhaft einige Regeln angegeben. Die Gesamtheit aller Regeln in den Regelmengen (bzw. den Phasen) ist im Anhang IV aufgelistet.

#### 8.3.4.4.1. Regelmenge I

Die Regeln dieser Regelmenge teilen solche Tasks in Cluster, die vom Benutzer schon eine feste Task-Prozessor Zuteilung erhalten haben.

Aufgrund von Beziehungen zwischen Tasks und technischen Prozessen (der Prozeßperipherie) bestehen feste Zuweisungen von Tasks an Prozessoren, die bei der Einteilung der Tasks in Cluster entsprechend berücksichtigt werden müssen. Feste Beziehungen können dazu führen, daß eine Task in ein neues Cluster bzw. in ein bestimmtes (bereits vorhandenes) Cluster gelegt werden muß (in der Beispielregel existiert bereits eine Task, die schon in einem Cluster liegt und die gleiche feste Zuteilung wie die betrachtete Task hat).

Priorität der Regelmenge: 1

Beispiel: Feste Zuweisung mit bereits zugeworbener Task

*Wenn* (1) Es existiert eine Task A, die eine feste Zuweisung zu einem Prozessor Y hat.  
& (2) Es existiert eine weitere Task B, die ebenfalls eine feste Zuweisung zu dem Prozessor Y hat und bereits einem Cluster C zugeworben ist.  
& (3) Die beiden Tasks sind durch die feste Zuweisung demselben Prozessor zugeworben.

*dann* Die Task A kommt in dasselbe Cluster, in dem die Task B bereits liegt.

*Priorität der Regel:* 1

#### 8.3.4.4.2. Regelmenge II

Die Regeln in dieser Menge bestimmen die Task, die folgende Bedingungen erfüllt:

- a) Für die Task besteht keine feste Zuweisung.
- b) Die Task liegt noch in keinem Cluster.



- c) Die Task hat größte Priorität in Bezug auf
- die Kommunikationskosten, oder
  - die Gesamtkommunikationskosten<sup>16</sup> aller noch nicht zugeteilter Tasks oder
  - die Anzahl ihrer Verbindungen zu anderen Tasks.

Hier werden diejenigen Tasks in Cluster eingeteilt, für die noch keine Einschränkungen aufgrund der Hardwarezuteilung bestehen. Zunächst wählt ein Experte eine Task aus, die er entweder in ein neues Cluster legen oder einem bereits vorhandenem zuteilen kann. Die richtige Auswahl der Task, die er als nächstes betrachtet, ist wichtig, da mit der richtigen Taskreihenfolge die Einteilung in Cluster entscheidend beeinflußt wird. Deshalb existieren verschiedene Regeln, die die Auswahl der geeigneten Task vornehmen. Die Auswahl wird beispielsweise anhand der höchsten Gesamtkommunikationskosten getroffen. Außerdem muß für diese Task eine Partnertask bestehen, die bereits einem Cluster zugewiesen ist und mit der die ausgesuchte Task zusammen in ein gemeinsames Cluster gelegt werden soll. Ansonsten wird ein neues Cluster angelegt.

Außer den exemplarisch aufgeführten Kriterien, die durch die Beispielregeln repräsentiert sind, sind eine Reihe weiterer Kriterien von Bedeutung. Die vorhandenen Regeln haben sich allerdings als hinreichend herausgestellt, so daß in der vorliegenden Arbeit auf weitere Auswahlkriterien verzichtet wurde.

Priorität der Regelmenge: 2

Beispiel: Bestimmen der Task mit größten Kommunikationskosten

Hier werden Tasks zur Clusterzuteilung ausgewählt, wobei vorzugsweise solche behandelt werden, die die größten Gesamtkommunikationskosten verursachen. Außerdem wird gefordert, daß die ausgesuchte Task die höchsten Kommunikationskosten mit einer bereits zugewiesenen Task hat.

---

<sup>16</sup> Gesamtkommunikationskosten einer Task:  
Summe der Kosten, die diese Task mit allen ihren Nachbarn hat.

- Wenn*      (1)    Betrachtet wird eine Task A, die die höchsten Gesamtkommunikationskosten hat.
- & (2)    Die Task A hat unter den betrachteten die größten Kommunikationskosten mit einer schon eingeteilten Partnertask B.

*dann*            Die Task A wird daraufhin überprüft, ob sie dem Cluster der Task B zugewiesen werden soll, oder ob ein neues Cluster zu erstellen ist.

*Priorität der Regel: 1*

Beispiel: Bestimmung einer Task mit größten Gesamtkommunikationskosten

Mit dieser Regel wird ebenfalls eine Task zur Clusterzuteilung ausgewählt. Im Gegensatz zur vorigen Regel existieren hier aber zwei Partnertasks, die mit der ausgewählten Task gleichgroße Kommunikationskosten und gleichgroße Gesamtkommunikationskosten haben. Ausgewählt wird dann die Task, die die größte Anzahl von Verbindungen zu anderen Tasks hat.

- Wenn*      (1)    Betrachtet wird eine Task A, die die höchsten Gesamtkommunikationskosten hat.
- & (2)    Es existieren zwei andere Tasks (B und C), die mit der Task A gleichgroße Kommunikationskosten haben.
- & (3)    Die Tasks B und C haben gleichgroße Gesamtkommunikationskosten.
- & (4)    Die Tasks B und C liegen bereits in Clustern  $C_1$  und  $C_2$ .

*dann*            Nehme die Task (B oder C), die die größte Verbindungsanzahl hat, und prüfe, ob die Task A dem Cluster dieser Task ( $C_1$  bzw. und  $C_2$ ) zugewiesen werden soll, oder ob ein neues Cluster zu erstellen ist.

*Priorität der Regel: 3*

#### 8.3.4.4.3. Regelmenge III

Für eine bestimmte Task wird die Entscheidung getroffen, ob für diese Task ein neues Cluster angelegt wird, oder ob die Task einem bereits bestehenden Cluster zugeteilt werden kann. Dazu stehen Regeln bereit, die nach bestimmten Einschränkungen der Zuteilungsmöglichkeiten suchen. Für den Fall, daß keine solche Einschränkung gefunden werden kann, werden Regeln aktiviert, die die Task abhängig von einer Kostenfunktion einem Cluster zuteilen.

Zu der Strategie aus der Regelmenge II, nach der ein Experte die größten Tasks (größte Task ist diejenige, die die höchsten Kommunikationskosten hat) auswählt, wird mit der Regelmenge III die Strategie realisiert, nach der ein Experte die Clusterzuweisung für eine Task vornimmt. Dazu geben die Randbedingungen wertvolle Hinweise zur Clusterbildung. Neben den Randbedingungen wird die Einteilung in solche Cluster betrachtet, bei denen die Kommunikationskosten zwischen den Tasks minimal werden.

Mit der Regelmenge III wird eine wichtige Vorentscheidung zur Verteilung getroffen. Die Cluster, die in dieser Regelmenge erzeugt werden, sind Ausgangspunkt für spätere Phasen. Die Qualität der Verteilung hängt somit in großem Umfang von der Einteilung der Tasks in Cluster ab. Für die detaillierte Überprüfung der Randbedingungen stehen in dieser Regelmenge 20 Regeln zur Verfügung, von denen hier drei als Beispiel vorgestellt werden:

- A) Überprüfung eines der möglichen Allokationsgebote.
- B) Überprüfung eines Allokationsverbotes.
- C) Wenn keine der Gebots- oder Verbotsregeln zutrifft, wird ein neues Cluster mit Hilfe einer geeigneten Metrik bestimmt.

Weitere Regeln überwachen Task-Prozessor-Verbote, Speicherüberlastung und Zuordnungsgebote. Diese Regeln können im Anhang IV nachgelesen werden.

Priorität der Regelmenge: 3

---

**Beispiel: Task-Allokationsgebot**

Falls ein Benutzer ein Allokationsgebot für zwei Tasks (A und B) angegeben hat, können diese beiden Tasks bereits in der Clusterzuteilung fest "verbunden" werden. Für diese Phase bedeutet dies, daß die beiden Tasks in ein gemeinsames Cluster gelegt werden.

Wenn (1) Für die zur Einteilung ausgewählte Task A besteht ein Allokationsgebot, das vom Benutzer vorgegeben ist.  
& (2) Die Task B, mit der für die ausgewählte Task A ein Allokationsgebot besteht, ist schon einem Cluster C zugeteilt.

dann Die ausgewählte Task A wird dem Cluster C zugewiesen, in der die Task B schon liegt.

*Priorität der Regel: 4*

**Beispiel: Allokationsverbot**

Wenn ein Benutzer ein Allokationsverbot (Task A und Task B) vorgegeben hat, dürfen diese beiden Tasks nicht auf einem gemeinsamen Prozessor zu liegen kommen. Dies bedeutet insbesondere, daß die beiden Tasks nicht gemeinsam in ein Cluster gelegt werden dürfen.

Wenn (1) Für die zur Einteilung ausgewählte Task A besteht ein Allokationsverbot.  
& (2) Die Partnertask B, mit der für die ausgewählte Task A ein Allokationsverbot besteht, ist schon einem Cluster C zugeteilt.

dann Die ausgewählte Task A darf dem Cluster nicht zugewiesen werden. Die Task muß entweder einem -bereits existierenden-Cluster C' zugewiesen werden, oder sie wird einem neuen Cluster  $C_{neu}$ <sup>17</sup> zugeteilt.

*Priorität der Regel: 1*

---

<sup>17</sup>  $C_{neu}$  wird vom Allokationssystem bestimmt.

### Beispiel: Zuteilung ohne Randbedingungen

Die bisher beschriebenen Fälle der Regelmenge III verarbeiten Randbedingungen, die bei der Clusteranalyse eine Rolle spielen können. Tasks, die keinen solchen Randbedingungen unterliegen, müssen mit Hilfe einer geeigneten Metrik auf die Cluster verteilt werden. Die gewählte Metrik hat entscheidenden Einfluß auf die Qualität der Lösung; deshalb soll sie in der vorliegenden Arbeit folgende Aspekte berücksichtigen:

- Gesamtkommunikation eines Clusters mit allen seinen Nachbarclustern
- Kommunikation mit den einzelnen Nachbarn
- Verbindungsanzahl eines Clusters mit seinen Nachbarn.

Die Prioritäten dieser Kriterien sind in dieser Reihenfolge vergeben. Auf dieser Basis wird ein Mittelwert errechnet. Je nach Größe des Abstands zu diesem Mittelwert wird eine Entscheidung zur Zuteilung getroffen.

*Wenn* (1) Die zur Einteilung ausgewählte Task A hat keine besonderen Merkmale<sup>18</sup>.

& (2) Die Partnertask B liegt schon in einem Cluster C.

*dann* Falls die Gesamtkommunikationskosten der betrachteten Task A kleiner sind (in Bezug auf die errechnete Metrik) als der (Gesamtkommunikationskosten-) Mittelwert,  
 oder die Kommunikationskosten der Task mit dem Cluster C kleiner sind als der (Kommunikationskosten-) Mittelwert,  
 oder die Anzahl der Verbindungen dieser Task kleiner ist als der Mittelwert aller Verbindungen,  
 dann wird die Task A in ein neues Cluster  $C_{neu}$  gelegt.  
 Anderenfalls wird die Task A in das Cluster der Partnertask B gelegt.

(Priorität der Regel: 2)

---

<sup>18</sup> Besondere Merkmale sind bestimmte Einschränkungen, die durch die Regeln der Regelmenge III geprüft werden.

#### 8.3.4.4. Vergleich mit anderen Methoden

Anhand der Clusteranalyse sollte gezeigt werden, wie sich Expertenwissen bei der Allokation repräsentieren läßt. Der Unterschied zwischen einer Lösung des Allokationsproblems, die den Anforderungen der Realzeitprogrammierung entspricht und anderen bislang verwendeten heuristischen Methoden wird hier deutlich:

- In herkömmlichen Systemen (wie sie z.B. in /MA82/ beschrieben sind), werden Tasks nur aufgrund der Kommunikationskosten in Cluster zusammengefaßt. Die Unterteilung wird anhand einer fest definierten Metrik verfolgt. Solche Verfahren lösen die Clustereinteilung dadurch, daß sie das größte Cluster als Startcluster definieren. Durch iterative Lösungsverfahren wird danach die Ausgangslage verbessert. Die Verbesserung wird durch die verbesserte Gesamtkommunikation des Systems festgestellt. Unberücksichtigt bleiben dabei Randbedingungen, die in jeder Situation Wissen zur Verteilung beitragen können. Außerdem werden (neben den Kommunikationskosten) keine zusätzlichen Optimierungskriterien berücksichtigt, die für eine Verteilung im Rahmen der Realzeitanwendungen unerlässlich sind.
- Im Gegensatz dazu wird bei der wissensbasierten Lösung die Vorgehensweise eines menschlichen Experten nachvollzogen. Da es bei komplexen Kommunikationsstrukturen sehr schwierig ist, vorab eine Einteilung in Cluster sofort zu erkennen, werden verschiedene Bedingungen bei der Einteilung berücksichtigt. Das Vorgehen nach "globalen" Strategien (z.B. Einteilung der Tasks in Cluster aufgrund von Metriken) wird deshalb abgelöst durch eine situationsgerechte Bearbeitung des Problems, unter Berücksichtigung aller Randbedingungen. Dies erfolgt in der vom Experten dargestellten Vorgehensweise. Die Zusammenhänge zwischen Tasks, sowie zwischen den Tasks und der Prozßperipherie werden also in die Zerlegung miteinbezogen. Entscheidungen werden aufgrund einer Vielzahl von Vorbedingungen gefällt, und die Zuordnungsstrategie wird der aktuellen Situation angepaßt. Diese Arbeitsweise bei der Einteilung der Cluster wird durch Regelmengen und die Regeln formuliert. Die Gesamtheit aller Regeln und die Strategie, mit der die Inferenz durchgeführt wird, repräsentieren das Wissen.

Am Beispiel der Clusteranalyse konnte die Repräsentation des Expertenwissens bei der Allokation verdeutlicht werden. Die Arbeitsweise des Expertensystems (und hier insbesondere die Wahl der geeigneten Regelmenge unter der Berücksichtigung der Strategie und der aktuellen Situation) ließe sich an der Phase "Zuteilung der Cluster an die Prozessoren" besser erkennen. Da dieser Teil der Allokation jedoch sehr umfangreich ist, wurde auf die Darstellung dieser Phase verzichtet. Die einzelnen Regelmengen und Regeln, und die damit verbundenen Strategien können aber dem Anhang IV entnommen werden.

Die Erforschung des Expertenwissens und die Repräsentation dieses Wissens mit den hier vorgestellten Strukturen genügen den Anforderungen, die an ein Allokationssystem im Rahmen der Realzeitprogrammierung gestellt werden. Die Vorgehensweise eines Experten konnte durch die Zerlegung des Wissens in problemspezifisches Wissen und Meta-Wissen in angemessener Weise nachvollzogen werden. Die Einteilung des problemspezifischen Wissens in Verarbeitungsschritte (Phasen), Regelmengen und Regeln wird den Anforderungen gerecht, die sich aus den Expertenbefragungen ergaben.

#### **8.4. Übersicht über die Regeln**

Die Wissensrepräsentation bei der Clusteranalyse wurde stellvertretend für die des gesamten Allokationssystems betrachtet. In der gleichen Weise wurde bei allen anderen Phasen das Wissen dargestellt. Da die Teilaufgaben innerhalb der Phasen unterschiedliche Ziele verfolgen, differiert auch die Anzahl der Regelmengen und der Regeln.

Die Regeln im Allokationssystem realisieren einzelne lokale Verarbeitungsschritte. So existieren beispielsweise in allen Phasen Regeln, die die Einhaltung der Randbedingungen überwachen (z.B. Task-Task-Verbote). Die Semantik dieser Regeln variiert in den verschiedenen Phasen, um die Forderung der aktuellen Strategie zu erfüllen.

Die unterschiedliche Wirkung einer Regel zur Überwachung der Randbedingung in verschiedenen Phasen kann man am Beispiel des Task-Task-Verbotes demonstrieren:

- In der Phase "Auswerten der festen Randbedingungen" wacht beispielsweise eine Regel darüber, daß nicht zwei Tasks trotz eines Task-Task-Verbotes einem gemeinsamen Prozessor zugeteilt werden.
- In der Clusteranalysephase sorgt eine Task-Task-Verbot-Regel dafür, daß die entsprechenden Tasks nicht in eine gemeinsames Cluster gelegt werden.

Außer der Überwachung der Randbedingungen erfüllen die Regeln im wissensbasierten System ALLOC weitere Aufgaben, die sich wie folgt zusammenfassen lassen:

- Einhaltung der Strategie der jeweiligen Regelmenge.
- Berücksichtigung der Randbedingungen innerhalb der Phasen.
- Entscheidung für einen Wechsel der Strategie.
- Verwaltungsaufgaben (z.B. Berechnen der Kommunikationskosten).
- Erkennen von inkonsistenten Zuständen.

Am Allokationssystem sind derzeit 120 Regeln (in umgangssprachlicher Notation bzw. 280 Regeln in der Schreibweise der Sprache OPS83 /FORG86/) beteiligt, die die Allokationsaufgaben wahrnehmen. Weitere 70 Regeln sind für untergeordnete (Verwaltungs-) Aufgaben zuständig.

In Abbildung 8.2. werden die Phasen mit der Anzahl der Regelmengen und der Regeln dargestellt (ohne die Regeln für Verwaltungsaufgaben).



Phase I:	Auswerten der festen Randbedingungen
Anzahl der Regelmengen:	4
Anzahl der Regeln:	30
Phase II:	Auswerten der Zuordnungseinschränkungen
Anzahl der Regelmengen:	4
Anzahl der Regeln:	40
Phase III:	Einteilen der Tasks in Cluster
Anzahl der Regelmengen:	4
Anzahl der Regeln:	32
Phase IV:	Verteilen der Cluster auf Prozessoren
Anzahl der Regelmengen:	14
Anzahl der Regeln:	92
Phase V:	Verschieben einzelner Tasks
Anzahl der Regelmengen:	5
Anzahl der Regeln:	41
Phase VI:	Verschieben ganzer Cluster
Anzahl der Regelmengen:	5
Anzahl der Regeln:	46

Abb. : 8.2. Phasen, Regelmengen und Regeln im System ALLOC

### 8.5. Zusammenfassung

Das Expertenwissen (bei der Allokation) wird durch das problemspezifische Wissen (die Phasen, die Regelmengen und die Regeln) und das Meta-Wissen repräsentiert. Das problemspezifische Wissen wird aufgrund der Anforderungen an ein Allokationssystem in folgende strukturelle Einheiten untergliedert:

#### Problemspezifisches Wissen

##### - Phasen

Mit Hilfe der Phasen wird erreicht, daß die Verarbeitungsschritte (A-F) des Allokationssystems in der festgelegten Reihenfolge abgearbeitet werden können.

Am Beispiel der Clusteranalyse wurde gezeigt, wie ein Teilbereich des Expertenwissens bei der Allokation in adäquate Strukturen umgesetzt wird.

##### - Regelmengen

Regelmengen repräsentieren die zentralen Teilaufgaben innerhalb der Phasen. Die Prioritäten der Regelmengen sind so gewählt, daß die Abarbeitungsstrategie des Experten genau simuliert werden kann.

##### - Regeln

Innerhalb der Regelmengen sind die Regeln für die Nachbildung der "lokalen" Teilschritte vorgesehen. Die Regeln arbeiten gemeinsam daran, das Ziel zu erfüllen, das mit einer Regelmenge verfolgt wird. Die Reihenfolge, mit der die Regeln aktiviert werden, wird durch Prioritäten festgelegt.

#### Meta-Wissen

Mit Hilfe des Metawissens ist das Allokationssystem in der Lage, die Reihenfolge der Phasen einzuhalten. Darüberhinaus werden hier die Entscheidungen über die zu aktivierende Regelmenge (bzw. die Regel) getroffen. Das Meta-Wissen ist also verantwortlich für die Einhaltung der Strategie der Verteilung.

Schließlich muß es dafür Sorge tragen, daß die jeweils "beste" Regel zur Ausführung gelangt.

## **Kapitel 9**

### **Das Allokationssystem**

#### **9. Das Allokationssystem**

##### **9.1. Übersicht**

Nachdem die Konzepte und Strukturen für das wissensbasierte System bekannt sind, wird in diesem Kapitel die Implementierung des Allokationssystems ALLOC vorgestellt.

In der vorliegenden Arbeit wurde eine Lösung des Allokationsproblems entwickelt, bei der folgende Kriterien berücksichtigt werden:

- Die Darstellung des Prozeßsystems und des zugrundeliegenden Hardware-systems erfolgt graphisch.
- Das Allokationsproblem wird so gelöst, wie es von menschlichen Experten vorgeschlagen wurde. Die Repräsentation geschieht durch die im vorangegan-genen Kapitel beschriebenen Strukturierungsmittel. Die Umsetzung dieser Teile in eine Implementation wird durch ein regelbasiertes System, das in der Sprache OPS83 /FORG86/ geschrieben ist, realisiert.

- 
- Zur Verarbeitung des Wissens werden Randbedingungen berücksichtigt, die folgende Bereiche abdecken:
    - Die Prozeßperipherie wird in die Verteilung einbezogen.
    - Es dürfen feste Zuordnungen zwischen Tasks und Prozeßperipherie festgelegt werden.
    - Tasks können mehrfach vorhandener Prozeßperipherie wahlweise zugeteilt werden.
    - Es kann nach unterschiedlichen Schwerpunkten optimiert werden:
      - Optimierung nach Kommunikationskosten
      - Gleichmäßige Auslastung der Prozessoren
      - Optimierung nach Laufzeiten
      - Auslastung aller Prozessoren
    - Der Benutzer kann Task-Prozessor-Gebote und -verbote angeben.
    - Es können Allokationsgebote und -verbote für Tasks definiert werden.
    - Die Überlastung eines Rechners wird erkannt und bei der Verteilung berücksichtigt.
    - Der Einfluß der Netztopologie wird berücksichtigt.
  - Das eigentliche Allokationssystem ALLOC wird darüberhinaus durch weitere Programmteile, die für die Ein- und Ausgabe verantwortlich sind, versorgt:
    - Die Information aus der Spezifikation wird in "Working Memory Elemente" umgesetzt.
    - Die Ergebnisse des Allokationssystems werden in übersichtlicher Form graphisch dargestellt, und darüberhinaus wird dem Benutzer die erzeugte Verteilung erklärt (Erklärungskomponente).

### Spezifikation als Eingabe

Da die vorliegende Arbeit in eine Programmierumgebung zur Erstellung von Realzeitsystemen eingebunden ist, wurden Systemkomponenten aus /Krag86/ für das Allokationssystem mitverwendet. Mit dieser Umgebung kann eine Realzeitaufgabe in PASS spezifiziert, validiert und in (PEARL-) Code umgesetzt werden.

- Bei der vorhandenen Umgebung zur Erstellung einer Spezifikation für eine Realzeitaufgabe wird die Methode PASS /FLEI84/ verwendet. Für diese Spezifikationsmethode sind das sogenannte Kommunikationsdiagramm und die Ablaufsteuerung anzufertigen. In der Ablaufsteuerung wird beschrieben, wann welche Aktionen und Reaktionen ausgeführt werden. In der Kommunikationsstruktur werden die Tasks, die Kommunikation zwischen Tasks und die technischen Prozesse spezifiziert.
- Diese Strukturen werden in einem weiteren Teil der Programmierumgebung /Krag86/ zur automatischen Erzeugung von PEARL-Code benutzt. Es existieren Programme, die diese graphischen Strukturen auswerten.
- Im Rahmen einer weiteren Arbeit /ANDR88/ wurde ein System erstellt, mit dessen Hilfe man eine (in graphischer Form) gegebene Spezifikation validieren kann.
- Die gleiche Spezifikation wird auch für die Verteilung genutzt.
- Ein Vorteil ist hier, daß eine einheitliche graphische Darstellung sowohl zur Spezifikation und der automatischen Codeerzeugung, als auch zur Allokation verwendet wird. Für das Allokationssystem benötigt der Benutzer also keine eigene Eingabemöglichkeit.

Die Darstellung der Hardware erfolgt ebenfalls mit der gleichen Spezifikationsmethode. Dazu wurde die "Hardwarestruktur" eingeführt.

Zur Ausgabe der Allokationsergebnisse wird ebenfalls eine graphische Darstellung gewählt: Der Benutzer des Allokationssystems erhält die Ergebnisse in Form einer Graphik, die die Hardwarekonfiguration, die zugewiesenen Tasks bei den entsprechenden Prozessoren, die Auslastung der Prozessoren und die Kommunikationskosten zwischen den Prozessoren enthält.

### **Gliederung des Allokationssystems**

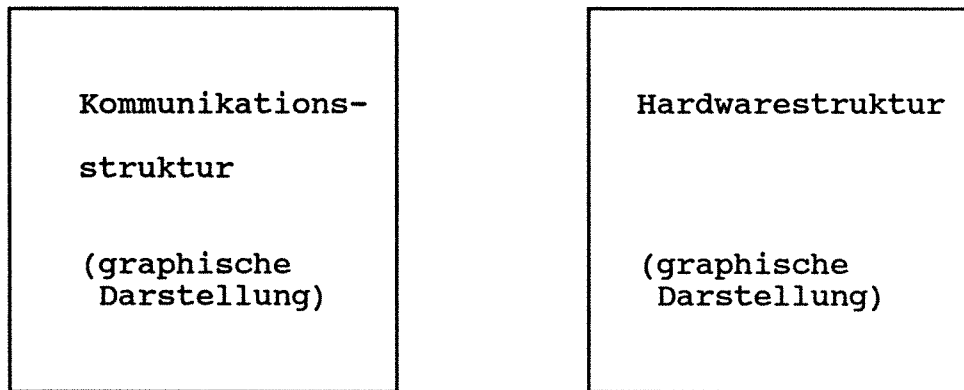
Mit dem wissensbasierten System wird das Allokationsproblem gelöst. Das in der Programmiersprache OPS83 /FORG86/ implementierte Allokationssystem ALLOC erstellt, abhängig von den gewünschten Optimierungskriterien, eine Lösung des Allokationsproblems.

Darüberhinaus gibt es weitere Programmteile, mit deren Hilfe der Benutzer in übersichtlicher Weise das Kommunikations- und das Hardwaresystem darstellen kann. Außerdem werden die Ergebnisse graphisch ausgegeben. Das gesamte Allokationssystem gliedert sich in drei Teile:

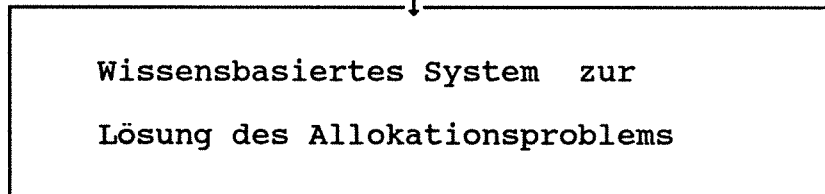
- Im ersten Teil wird die graphische Eingabe ausgewertet. Aus der Information der Kommunikationsstruktur und der Hardwarestruktur werden Listen für die nachfolgende Teile erstellt.
- Der zweite Teil ist das wissensbasierte System, mit dessen Hilfe die Verteilung erstellt und nach verschiedenen Kriterien optimiert wird.
- Im letzten Teil werden die Ergebnisse aufbereitet und mit Hilfe eines graphischen Editors ausgegeben. Dem Benutzer wird das Hardwaresystem (Prozesse und angeschlossene Peripherie) in graphischer Form dargestellt.  
Zudem kann die Erklärungskomponente Aufschluß darüber geben, aufgrund welcher Kriterien das Allokationssystem zu seinen Entscheidungen gekommen ist.

In der folgenden Graphik ist der Ablauf des Allokationssystems dargestellt.

EINGABE:



ALLOC



AUSGABE:

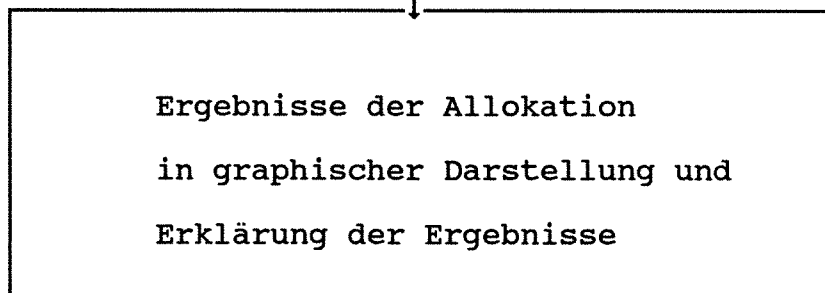


Abb. : 9.1. Ablauf des Allokationssystems

## 9.2. Eingabedaten für das Allokationssystem

### 9.2.1. Umsetzen der graphischen Eingabe

In /Krag86/ wurde ein Programm implementiert, das die graphischen Elemente der Kommunikationsstruktur und der Ablaufsteuerung einer Spezifikation in einen Zwischencode transferiert. Dieser Zwischencode repräsentiert die Information aus der Graphik in einer festgelegten Darstellung. Für die vorliegende Arbeit wurde diese Darstellung (bzw. das Programm, das diese Darstellung erzeugt) übernommen. Aus diesem "Zwischencode" erstellt ein weiteres Programm die Working-Memory-Elemente. Für die graphische Darstellung der Hardwarestruktur gilt die gleiche Vorgehensweise.

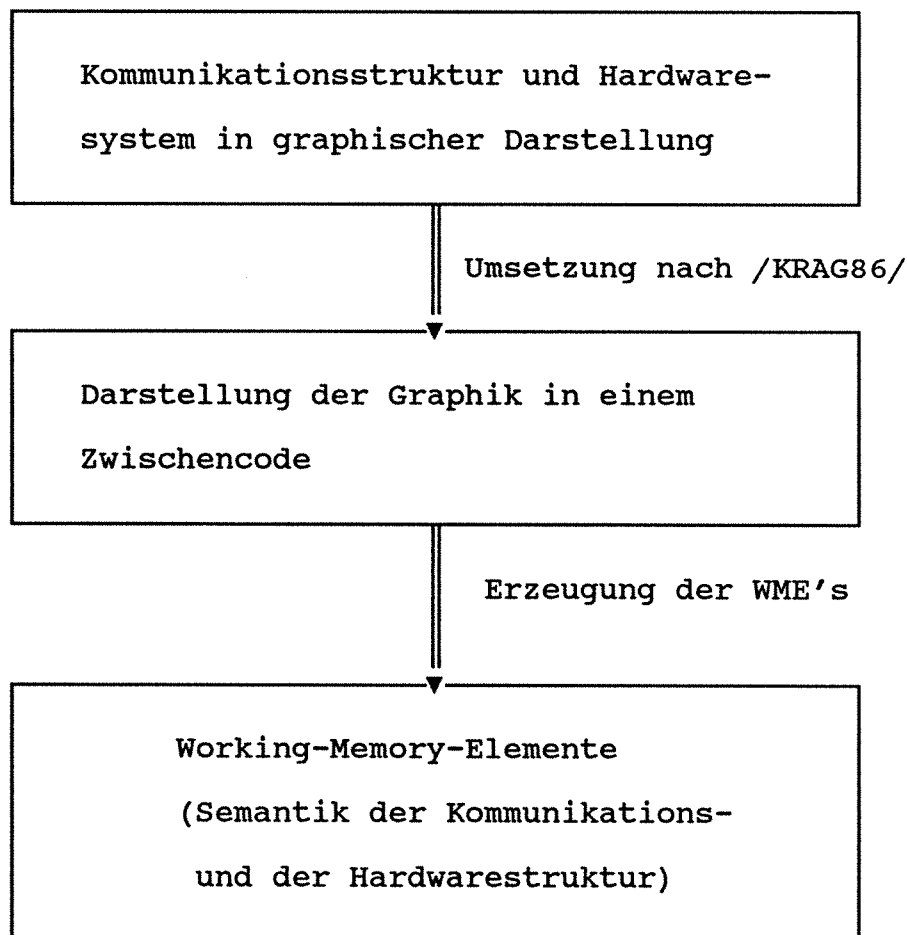


Abb. : 9.2. Umsetzung der Graphik in Working-Memory-Elemente



### 9.2.2. Erzeugen der Working Memory Elemente

Als Working-Memory-Elemente werden die "Fakten" aus der Kommunikationsstruktur und der Hardwarestruktur benutzt.

Die syntaktische Auswertung der Kommunikationsstruktur erfolgt im Programmteil zur Umsetzung der Graphik in den Zwischencode. Daran anschließend wird die semantische Analyse der Kommunikationsstruktur bzw. des Hardwaresystems vorgenommen. Die in der Kommunikationsstruktur (bzw. Hardwarestruktur) vorhandene Semantik wird ausgewertet und in die für das Expertensystem verständlichen Working-Memory-Elemente umgesetzt.

#### Kommunikationsstruktur

Vom Analyseprogramm werden aus der Kommunikationsstruktur folgende Working Memory-Elemente erstellt:

- Botschaften
- Prozesse
- Verbindungen zwischen den Prozessen
- Laufzeitkosten
- Größe der einzelnen Prozesse
- Kommunikationskosten

(Die Kommunikationskosten sind -im Gegensatz zur formalen Darstellung in Kapitel 2- unsymmetrisch; d.h. je eine Verbindung zwischen zwei Tasks wird als ein Working-Memory-Element dargestellt).

- Tasks nach Kommunikationskosten geordnet
- Tasks nach Partneranzahl geordnet
- Topologie (durch die die Kommunikationskosten und die damit angezeigten Task - Task - Verbindungen festgelegt werden)
- Task-Prozessorzuweisungen
- Task-Prozessorverbote
- Peripheriezugehörigkeit einer Task
- Task-Task Gebote
- Task-Task-Verbote

**Erweiterung der Kommunikationsstruktur**

Um alle benötigten Working-Memory-Elemente erstellen zu können, muß die Kommunikationsstruktur erweitert werden<sup>19</sup>. Zusätzlich zu den in /FLEI84/ vorgegebenen graphischen Symbolen werden im Kommunikationsdiagramm weitere Beschriftungen zugelassen:

- Kommunikationskosten zwischen je zwei Tasks
- Kosten zwischen Task und einem peripheren Gerät
- Feste Task-Prozessor-Zuordnung
- Task-Prozessor-Verbote
- Laufzeitbedarf einer Task
- Speicherplatzbedarf einer Task

**Hardware**

Im Hardwarediagramm wird das Hardwaresystem beschrieben, auf das die Tasks verteilt werden sollen. Der Benutzer kann die Hardwarestruktur ähnlich wie die Kommunikationsstruktur darstellen. Die Prozessoren und die Verbindungen zwischen den Prozessoren, sowie die an dem jeweiligen Prozessor vorhandene Peripherie können mit graphischen Symbolen dargestellt werden. Die Prozessorkapazität und die Leistungsfähigkeit (MIPS) des Prozessors, sowie die Kapazität der Verbindungen zwischen den Prozessoren werden vom Benutzer als Beschriftung des "Hardwaregraphen" angegeben.

Aus der Hardwaredarstellung werden zusätzlich folgende Working-Memory-Elemente erzeugt:

- Prozessorgröße
- Leistungsfähigkeit des Prozessors
- Peripherie
- Netzwerktopologie

Eine vollständige Beschreibung der Eingabe an das Allokationsprogramm (bestehend aus Kommunikationsstruktur und der zugrundeliegenden Hardwarestruktur) befindet sich im Anhang.

---

<sup>19</sup> Beispiel siehe Anhang I

### 9.3. Das wissensbasierte System ALLOC

#### 9.3.1. Beschreibung der Sprache OPS

Nachdem die Kommunikationsstruktur und die Hardwarestruktur ausgewertet und die Working-Memory-Elemente (WME) angelegt sind, kann nun das Allokationsprogramm die Verteilung vornehmen. Das Allokationssystem wurde in der Sprache OPS /FORG86/ implementiert. Diese für die Entwicklung von Expertensystemen geschaffene Programmiersprache bietet dem Benutzer die Programmiermöglichkeiten konventioneller Sprachen an (Unterprogramme, Laufschleifen, Rekursion ... ). Zusätzlich zu den "normalen" Sprachkonstrukten können Regeln formuliert werden, die zusammen mit dem Working-Memory die Darstellung der Wissensbasis ermöglichen. Die Regeln haben in OPS folgendes syntaktisches Aussehen:

```
rule Beispiel          -- Name der Regel

{
    LHS                -- "Left-Hand-Side" der Regel
-->
    RHS                -- "Right-Hand-Side" der Regel
};
```

#### LHS

Die LHS (Left-Hand-Side) einer Regel ist der sogenannte Bedingungsteil; er besteht aus bedingten Ausdrücken. Diese Verknüpfungen bestehen (in OPS) ihrerseits aus einzelnen '*Patterns*', die wiederum aus einzelnen '*Termen*' zusammengesetzt sind. Eine Regel ist dann ausführbar, wenn alle '*Patterns*' der LHS ausführbar sind.

---

Ein Element aus dem Working-Memory macht ein 'Pattern' gültig, wenn:

1. das Element vom gleichen Typ ist wie das entsprechende Element in der Regel  
und
2. das Element jeden 'Term' innerhalb des 'Pattern' erfüllt

Eine Regel wird dadurch ausführbar, daß alle ihre Patterns gültig sind. Als **Instantiierung** wird in OPS eine Regel bezeichnet, die aufgrund der aktuellen Belegung des WME's gültig ist. Zu einer Regel kann es aufgrund verschiedener WME-Elemente mehrere unterschiedliche Instantiierungen geben.

### RHS

Die RHS (Right-Hand-Side) einer Regel entspricht exakt dem ausführbaren Teil einer Prozedur. Außer den "normalen" Sprachkonstrukten stehen in OPS Sprachelemente zur Manipulation des Working-Memory (WM) zur Verfügung<sup>20</sup>. Es gibt drei verschiedene Möglichkeiten, Elemente im Working-Memory zu manipulieren:

1. **MAKE**      Mit diesem Statement wird ein WM-Element erzeugt.
2. **MODIFY**    Dieses Statement wird benutzt, um einen oder mehrere Werte in einem existierenden WM-Element zu verändern.
3. **REMOVE**    Mit diesem Statement entfernt man ein WM-Element aus dem Working Memory.

---

<sup>20</sup>. Der Working-Memory besteht aus einer Reihe von "Record-Strukturen" die Working-Memory-Elemente genannt werden.

### Interpreter

Um die Regeln "auszuführen", ist ein Regelinterpreter (siehe Kapitel 7) notwendig, der die Regeln im sogenannten recognize-act-cycle anwendet. In OPS besteht dieser recognize-act-cycle aus folgenden Schritten:

1. Match:                      Aussuchen der Regeln, deren LHS mit dem aktuellen Inhalt des Working-Memory ausführbar sind (Instanzen von Regeln).
2. Conflict Resolution:      Bestimmen einer Regel (nach definierten Strategien), die ausgeführt wird. Falls keine Regel ausführbar ist --> STOP.
3. Act:                         Ausführen der Anweisungen auf der rechten Seite der Regel.
4. Wieder bei Punkt 1 weitermachen

Die Strategien, nach denen eine Regel aus der Menge der ausführbaren Regeln ausgewählt wird, sind in den Expertensystemen unterschiedlich realisiert. In komfortablen Systemen (z.B. FORK /BECK86/, /TIEL85/) können zu verschiedenen Zeitpunkten -abhängig vom Gesamtzustand des Expertensystems- verschiedene Strategien gewählt werden.

Die Auswahlstrategie, die in OPS angeboten wird, besteht aus folgenden Teilen:

1. Ausgeführt werden nur Instanziierungen, die noch nicht ausgeführt wurden.
2. Von diesen werden die ausgesucht, deren erste Elemente die höchsten Nummern im Working-Memory haben.
3. Aus diesen wiederum werden die Regeln mit der größten Anzahl von Patterns in der LHS zugelassen.
4. Aus dieser Menge von ausgewählten Regeln wird diejenige zur Ausführung gebracht, die die Konfliktmenge zuletzt betrat.

Da in OPS keine Phasen und Regelmengen vorgesehen sind, mußten sie zusätzlich in die Sprache eingebracht werden. Der Regelinterpreter wurde erweitert, damit er die neuen Sprachkonstrukte verarbeiten kann. Die Regeln, die sich in der Konfliktmenge befinden, werden in der erweiterten Sprache daraufhin untersucht, ob sie zur gültigen Phase und Regelmenge gehören.

Darüberhinaus wurde mit Hilfe der Rekursion die Möglichkeit geschaffen, alle Instanziierungen einer bestimmten Regel ausführen zu lassen. Damit können beispielsweise für eine Task vorrangig alle Verbote erkannt werden, bevor sie zur Verteilung freigegeben wird.

Am Beispiel einer Regel aus der Clusteranalyse (siehe Kapitel 8, bzw. Anhang IV) soll gezeigt werden, wie die umgangssprachlich formulierte Regel in eine OPS-Regel umgesetzt wird.

#### Feste Zuweisung mit bereits zugeteilter Task

Wenn ( Logische Verknüpfung I )

- |     |  |   |
|-----|--|---|
| (1) | Es existiert eine Task A,<br>die eine feste Zuweisung hat. | ( <u>Pattern I</u> )<br>( <u>Pattern II</u> ) |
|-----|--|---|

( Logische Verknüpfung II )

- |      |   |   |
|------|---|---|
| &(2) | Es existiert eine weitere Task B,<br>die ebenfalls eine feste Zuweisung hat<br>und bereits einem Cluster zugeteilt ist. | ( <u>Pattern I</u> )<br>( <u>Pattern II</u> )<br>( <u>Pattern III</u> ) |
|------|---|---|

( Logische Verknüpfung III )

- |      |   |                      |
|------|---|----------------------|
| &(3) | Die beiden Tasks sind durch die feste Zuweisung demselben<br>Prozessor zugeteilt. | ( <u>Pattern I</u> ) |
|------|---|----------------------|

dann Die Task A kommt in dasselbe Cluster, in dem die Task B bereits liegt.

*Priorität der Regel:* 1

Eine OPS-Regel besteht aus dem Regelnamen, einer linken und einer rechten Seite. Die einzelnen logischen Verknüpfungen sind mit "UND" verbunden. Die Variablen  $V_x$  ( $x = 1, 2, \dots, n$ ) dienen dazu, die Elemente der Patterns der linken Seiten zu verknüpfen und die Elemente an die rechten Seiten weiterzugeben. Kommentare sind durch "--" gekennzeichnet. Das Zeichen "-->" trennt die rechte von der linken Seite. Das Zeichen "<>" steht für ungleich und das Zeichen "\" für das logische "ODER".

Um die Forderungen aus der (umgangssprachlichen) Regel umzusetzen werden die folgenden Working-Memory-Elemente (mit den entsprechenden Komponenten z.B. Taskname) benötigt:

Taskliste:	Taskname, zugeordnet;
Task_Prozessor_Zuordnung:	Taskname, Prozessor(nummer), verbunden;
Cluster:	Clusternummer, Taskname;

Diese WME's reichen aus, um alle Forderungen der Regel zu erfüllen. So wird beispielsweise die Forderung LV III dadurch erfüllt, daß 2 Instantiierungen des WME's Task\_Prozessor\_Zuordnung gefunden werden müssen, für die die Prozessornummern übereinstimmen.

Für das vorliegende Beispiel bedeutet dies, daß vier verschiedene Working-Memory-Elemente (1: Taskliste, 2: Task\_Prozessor\_Zuordnung, 3: Task\_Prozessor\_Zuordnung, 4: Cluster) gefunden werden müssen, die die linke Seite der Regel gültig machen.

- Für das Element 1 wird ein beliebiger Taskname gefordert (LV I, P I).
- Element 2 soll den gleichen Tasknamen wie Element 1 besitzen und eine feste Zuordnung haben (LV I, P II).
- Element 3 darf nicht den gleichen Tasknamen wie die beiden vorangegangenen Elemente besitzen (LV II, P I); soll aber eine feste Zuordnung vorweisen können (LV II, P II). Außerdem wird gefordert, daß die Elemente 2 und 3 beide dem gleichen Prozessor zugeordnet sind (LV III, P I).
- Das vierte Element schließlich soll wiederum den ersten Tasknamen besitzen (LV II, P III).

Darstellung der Regel "Feste Zuweisung mit bereits zugeteilter Task" in OPS:**rule Phase06\_Regel2**

```

    -- Linke Seite der Regel ( "Wenn" )

{
    -- Logische Verknüpfung I
    V1 ( Taskliste zugeordnet    <> nein );           -- Pattern I
    V2 ( Task_Prozessor_Zuordnung
        Task      = V1.Task;           -- Pattern II
        verbunden = fest );

    -- Logische Verknüpfung II
    V3 ( Task_Prozessor_Zuordnung
        Task      <> V1.Task;           -- Pattern I
        verbunden = fest;               -- Pattern II

    -- Logische Verknüpfung III
        Prozessor = V2.Prozessor; );   -- Pattern I

    -- Logische Verknüpfung II
    V4 ( Cluster      Task      = V3.Task );           -- Pattern III

-->

    -- Rechte Seite der Regel ( "Dann" )

    -- Ein neues Cluster-Element wird angelegt. Die Clusternummer wird
    -- dem Clusterelement der bereits zugewiesenen Task entnommen.
    -- Der Taskname wird dem Element V1 entnommen, das der Task A der
    -- Regel (in umgangssprachlicher Formulierung) entspricht.

    make ( Cluster      Nummer = V4.Clusternummer;
          Task      = V1.Task;
          zugeordnet = V2.verbunden );

};

```



### 9.3.2. Implementierung des Allokationssystems

Die in Kapitel 8 vorgestellten Verarbeitungsschritte werden in der Implementierung weiter verfeinert. Dazu werden die Regelmengen näher beschrieben. In jeder Phase stehen neben den hier aufgeführten Regelmengen noch solche zur Verfügung, die sich um organisatorische Belange kümmern (z.B. Sortieren der Tasks nach Kommunikationskosten).

#### 9.3.2.1. Auswerten der festen Randbedingungen

In dieser Phase werden die Tasks an Prozessoren verteilt, für die eine feste, vom Benutzer vorgegebene Verbindung vorliegt. Eine "feste" Verbindung kann bestehen in:

1. fester, vom Benutzer explizit angegebener Zuordnung von Task und Prozessor,
2. der Zuteilung einer Task aufgrund eines peripheren Gerätes, das nur an einem bestimmten Prozessor vorhanden ist.

**Beispiel:**     Voraussetzungen:

Task A fordert Drucker 1 an.

Der Drucker 1 ist an Prozessor X angeschlossen.

Ergebnis:

Das Expertensystem weist die Task A dem Prozessor X fest zu.

Eine Regelmenge in dieser Phase beschäftigt sich ausschließlich mit der Auswertung der festen Zuordnungen (sowohl Zuordnungen aufgrund von Benutzerzuteilungen, als auch aufgrund der Prozeßperipherie).

Die zweite Regelmenge dieser Phase überprüft die Verteilungsvorschrift auf mögliche Inkonsistenzen:

1. Eine Task wurde vom Benutzer einem Prozessor zugewiesen, an dem die benötigte Peripherie nicht zur Verfügung steht.
2. Eine Task hat für n Prozessoren ein Zuordnungsverbot erhalten. Aufgrund der Peripherieanforderung müßte sie aber auf einen

dieser Prozessoren zu liegen kommen.

3. Es fehlt eine benötigte Peripherie an einem bestimmten Prozessor.
4. Zwei Tasks wurden vom Benutzer einem gemeinsamen Prozessor zugeordnet. Gleichzeitig besteht für diese Tasks aber ein Allokationsverbot.
5. Zwei Tasks besitzen Task-Task-Zuordnungsgebot. Andererseits sollen sie aufgrund einer vorliegenden Hardwarezuordnung getrennten Prozessoren zugeordnet werden.
6. Falls aufgrund fester Zuordnungen bereits in dieser Phase ein Prozessor überbelegt ist, wird der Benutzer darüber informiert und gegebenenfalls der Speicherplatz des Prozessors erhöht.
7. Die Namensgebung im Kommunikations- und im Hardwarediagramm stimmt nicht überein.

Jeder dieser Inkonsistenztests wird im Allokationssystem durch eine Regel repräsentiert. Mit den "linken Seiten" der Regeln wird eine Inkonsistenz erkannt, während mit dem Ausführungsteil versucht wird (gegebenenfalls nach "Rücksprache" mit dem Benutzer) den Widerspruch aufzulösen.

#### 9.3.2.2. Auswerten der Zuordnungseinschränkung

Während in der vorangegangenen Phase Tasks aufgrund ihrer Prozeßperipherieanforderungen einem Prozessor fest zugeordnet werden, sollen in dieser Phase Einschränkungen untersucht werden. Unter einer Einschränkung der Zuteilung soll beispielsweise folgendes verstanden werden:

Der Benutzer fordert eine Verbindung einer Task mit einem Drucker. Im Hardwaresystem sind verschiedene Prozessoren vorhanden, die Zugang zu einem Drucker haben. Da der Benutzer keinen bestimmten Drucker (über dessen Namen gekennzeichnet) angegeben hat, wird für die entsprechende Task keine feste Zuordnung nötig. Die Zuteilung der Task wird auf die Prozessoren eingeschränkt, die einen Drucker haben.

Das Expertensystem erkennt diese Situation und reagiert darauf mit einer Einschränkung der Zuordnungsmöglichkeiten für die entsprechende Task.

**Beispiel:**Voraussetzungen:

Task A fordert einen (beliebigen) Drucker an.

Prozessor 1 besitzt Druckerzugang.

Prozessor 2 besitzt Druckerzugang.

Alle anderen N-2 Prozessoren besitzen keinen Druckerzugang.

Ergebnis:

Das Expertensystem schränkt die Zuordnung der Task A auf die Prozessoren 1 und 2 ein. In dieser Phase wird aber keine Zuordnung an einen der beiden Prozessoren vorgenommen.

In der ersten Regelmenge dieser Phase werden die Einschränkungen, die sich aufgrund der Randbedingungen ergeben, ausgewertet. Aufgrund dieser Einschränkungen werden die betroffenen Tasks in ihrer Allokationsfreiheit beschränkt.

Die zweite Regelmenge erfüllt auch hier die Überprüfung der Eingaben auf vorhandene Inkonsistenzen und Auflösung der Widersprüche.

**9.3.2.3. Clusterbildung**

Ein wesentlicher Bestandteil des Systems ALLOC ist die Einteilung der Tasks in Cluster. Die Clusteranalyse wurde bereits in Kapitel 8 ausführlich beschrieben.

**9.3.2.4. Abbildung auf die vorhandenen Prozessoren.**

Nachdem die Tasks in Cluster unterteilt sind, wird in einer weiteren Phase versucht, diese Cluster auf die Prozessoren abzubilden.

Ziel der Verteilung der Cluster auf die Prozessoren ist es, je nach gewünschtem Kriterium,

- möglichst wenig Kommunikationskosten zu erzeugen, oder
- die Laufzeiten zu optimieren, oder

- die Prozessoren gleichmäßig auszulasten und die Kommunikationskosten gering zu halten, oder
- die Laufzeiten zu optimieren und die Kommunikationskosten gering zu halten.

Diese Kriterien können auch kombiniert werden. Außerdem kann der Benutzer angeben, ob alle zur Verfügung gestellten Prozessoren ausgelastet werden sollen. Im anderen Fall bleibt es dem System überlassen zu entscheiden, wieviele der Prozessoren es benutzen will, um eine optimale Verteilung zu erreichen.

Die Verteilungsstrategie richtet sich in der vorliegenden Arbeit vor allem nach der Art der Vernetzung. Erlaubt sind folgende Vernetzungsarten:

- Direkte Verbindungen zwischen den Prozessoren und
- vollständig vermaschte Systeme (lokale Netze).

Die globale Zuteilungsstrategie wird im Allokationssystem (je durch eine Regelmenge vertreten) folgendermaßen eingeteilt:

1. Alle Cluster, die feste Task-Prozessor-Zuordnungen beinhalten, werden zugewiesen (Priorität 3).
2. Es werden solche Cluster zugewiesen, für die eine Zuordnungseinschränkung für bestimmte Prozessoren besteht (Priorität 4).
3. Beim Netzwerktyp "Direkte Verbindungen" werden die Cluster zugeteilt, die die niedrigsten Kommunikationskosten haben (Priorität 5).
4. Für den Netzwerktyp "vermaschtes System" gilt für diesen Fall die bevorzugte Zuordnung für Cluster mit geringer Verbindungsanzahl (Priorität 6).
5. Wenn alle Prozessoren mit mindestens einem Cluster belegt sind, werden die restlichen Cluster auf die Prozessoren gelegt, mit denen sie jeweils die größten Kommunikationskosten haben (Priorität 7).
6. Falls einem Prozessor keine Task zuteilt werden kann, obwohl dies vom Benutzer gefordert wurde, wird durch eine Verschiebung von einzelnen Tasks oder Clustern dieser Prozessor ebenfalls genutzt (Priorität 8).

Weitere vier Regelmengen sind für die Einhaltung der oben aufgeführten Optimierungskriterien zuständig. Die Prioritäten dieser Regelmengen sind höher als die der Regelmengen der Zuteilungsstrategien (Prioritäten 9 - 12). Um die in Kapitel 8 geforderte Kombination aus verschiedenen Strategien zu gewährleisten, existiert eine weitere Regelmenge, die den Zeitpunkt für die Auswahl der entsprechenden Strategie trifft (Priorität 1).

Falls es zu einer Fehlentscheidung gekommen ist (z.B. ein Prozessor ist in unvertretbarer Weise überlastet) muß dafür gesorgt werden, daß solche Schritte rückgängig gemacht werden. Dies wird durch die letzte Regelmenge erledigt (Priorität 2).

Diese Einteilung spiegelt die Strategie zur Zuteilung der Cluster zu den Prozessoren wieder. Tatsächlich ist eine Reihe weiterer Bedingungen zu prüfen; so ist beispielsweise darauf zu achten, daß kein Task-Prozessor-Verbot verletzt wird. Solche Teilaufgaben werden typischerweise von Regeln erfüllt.

#### **9.3.2.5. Verschieben einzelner Tasks**

Mit der Verbesserung der Zuteilung soll versucht werden, das Gesamtverhalten der vorher erreichten Verteilung noch zu verbessern. Die Verbesserung wird durch Verschieben einzelner Tasks (oder Cluster) erreicht.

- Falls die meisten Tasks aufgrund eines einschränkenden Kriteriums an einen der Prozessoren gebunden sind, besteht nur wenig Möglichkeit, die Verteilung noch zu verbessern.
- Bei überwiegend unbeschränkt verteilten Tasks hingegen werden mit dem Umverteilen der Tasks gute Ergebnisse erzielt.

Im allgemeinen wurden von den Experten drei verschiedene Vorgehensweisen zum Verbessern der Lösung angeboten:

- Vertauschen aller Tasks zweier Prozessoren.
- Verschieben von Clustern auf den Prozessoren.
- Verschieben von Tasks auf den Prozessoren.

In der vorliegenden Arbeit wurden nur die Möglichkeiten zur Verschiebung von Tasks und Clustern implementiert, da davon ausgegangen werden kann, daß im Rahmen der Realzeitanwendung immer Tasks vorhanden sind, die fest an Prozessoren gebunden sind. Aufgrund dieser Tatsache erscheint es nicht sinnvoll, Prozessoren zu vertauschen.

In dieser Phasen existieren Regelmengen, die nach Kommunikationsgröße der Task eine Task auswählen, die zur Verschiebung geeignet ist. Eine andere Regelmenge sorgt dafür, daß durch die Verschiebung keine Randbedingungen verletzt werden.

Eine weitere Regelmenge ermittelt den Grad der Verbesserung des Gesamtsystem. Dies ist vor allem bei kombinierten Optimierungskriterien notwendig. Eine Umordnung kann eine Verbesserung der Kommunikationskosten bringen, gleichzeitig jedoch die Auslastung des Speichers in größerem Umfang verschlechtern. Eine "Optimierung nach Kommunikationskosten" würde eine Verbesserung des Gesamtsystems bewirken, im Falle "Optimieren der Kommunikationskosten und gleichmäßiger Speicherauslastung" würde sich eine Verschlechterung ergeben.

Erst nachdem eine Verbesserung anerkannt wurde, verschiebt eine weitere Regelmenge die entsprechende Task und sorgt dafür, daß die Veränderungen, die sich dadurch am Gesamtsystem ergeben, berücksichtigt werden. (beispielsweise müssen die Kommunikationskosten und die Task-Prozessor-Zuweisungen geändert werden).

#### **9.3.2.6. Verschieben einzelner Cluster**

Um das Gesamtsystem zu optimieren, werden (mit der Phase 6) Cluster verschoben. Die Regelmengen sind hier die gleichen, wie bei der Verschiebung von Tasks. Allerdings müssen in diesem Fall Cluster ausgewählt werden, und die Auswirkungen, die die Verschiebung ganzer Cluster mit sich bringen, sind schwieriger zu überblicken. Deshalb besitzen die Regelmengen in dieser Phase insgesamt mehr Regeln als die vorherige Phase.

#### 9.4. Graphische Ausgabe

Um dem Benutzer einen geeigneten Überblick über die erreichte Verteilung zu verschaffen, wird eine graphische Ausgabe gewählt. Dazu wurde ein Programm /SCHI88/ entwickelt, mit dem die errechnete Verteilung in übersichtlicher Form graphisch dargestellt wird. In den Diagrammen wird der Benutzer auch auf Fehler (z.B. fehlende Verbindung zwischen zwei Prozessoren) und "Mängel" (z.B. eine Leitung wird bei der Verteilung überhaupt nicht benötigt) aufmerksam gemacht. Drei unterschiedliche Darstellungen können gewählt werden:

- Task-Hardware-Beziehungen (I)

In diesem Diagramm sind die Tasks den Prozessoren zugeordnet.

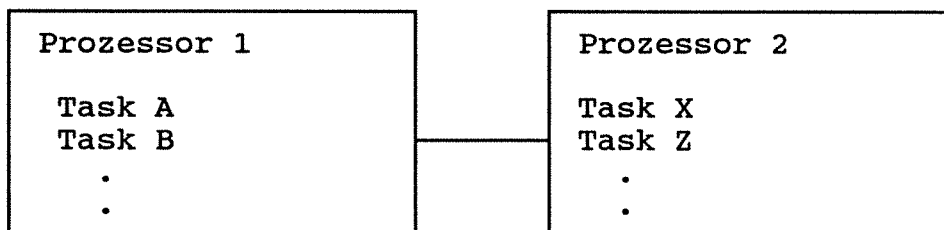
- Hardware-Beziehungen (II)

Hier wird angegeben, welche Peripherie an welchem Prozessor vorhanden ist.

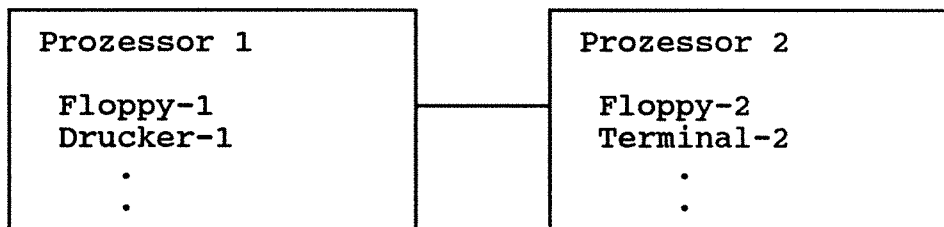
- Auslastung und Kommunikation (III)

In diesem Diagramm werden die Eingabegrößen Speicherplatz und Laufzeit, sowie die errechneten Größen der Speicher- und die Laufzeitauslastung der Prozessoren ausgegeben. Die Kommunikationskosten zwischen den Prozessoren werden ebenfalls in diesem Diagramm angegeben.

##### Darstellung I ( Task-Hardware-Beziehungen)



## Darstellung II ( Hardware-Beziehungen )



## Darstellung III ( Auslastung und Kommunikation )

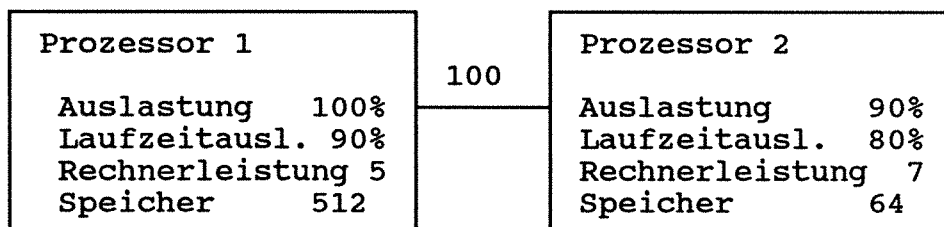


Abb. : 9.3. Graphische Darstellung der Ergebnisse

## 9.5. Erklärungskomponente

Neben einer graphischen Darstellung kann das Ergebnis des Allokationssystems durch eine "Erklärungskomponente" transparent gemacht werden. Dieser Teil eines wissensbasierten Systems soll dem Benutzer die getroffenen Entscheidungen darlegen. In der vorliegenden Arbeit gibt es zu diesem Zweck eine Erklärungskomponente /WISS88/, die im Anschluß an das Allokationssystem die Ergebnisse darstellt. Der Benutzer kann sich darüber informieren lassen, warum eine erreichte Verteilung in der entsprechenden Art erfolgt ist.

Anders als bei Erklärungskomponenten herkömmlicher Systeme, bei denen während des Ablaufs des Systems Fragen beantwortet werden können, sind beim Allokationssystem erst Fragen im Anschluß an die Verteilung erlaubt. Dies erscheint für die Allokation sinnvoll, da ein Interesse an der Begründung für eine errechnete Verteilung besteht und nicht am Ablauf des Systems.



Mit der Erklärungskomponente ist es möglich, die Ergebnisse des Allokationssystems in verschiedenen Detaillierungsstufen nachvollziehbar zu machen:

- 1) Zunächst wird versucht (Stufe I), das Ergebnis in einem Satz zu erläutern.
- 2) Falls der Benutzer zusätzliche Informationen wünscht, können die wichtigsten Einzelschritte für jede Task (Stufe II) abgefragt werden.
- 3) Und schließlich können (Stufe III) alle Schritte für den "Lebenslauf" der Tasks ausgegeben werden.

Beispiel:

Die Task Dialog benötigt einen Zugang zu einem Drucker, der nur von Prozessor 3 zur Verfügung gestellt wird. Aufgrund dieser Anforderung kann die Task nur dem Prozessor 3 zugeteilt werden.

Die Erklärungskomponente gibt deshalb folgenden Satz aus:

"Die Task Dialog wurde dem Prozessor 3 aufgrund der Hardware fest zugeordnet".

Für Tasks ohne feste Zuordnung wird von der Erklärungskomponente das Cluster angegeben in das diese Task gelegt wurde und die Zuteilung dieses Clusters an einen Prozessor angegeben:

"Die Task Proto wurde dem Cluster 5 aufgrund eines Task-Task-Zuordnungsgebotes mit der Task Puffern zugeordnet".

"Das Cluster 5 wurde dem Prozessor 2 aufgrund der der Kommunikationskosten mit dem bereits zugeordneten Cluster 3 zugeordnet".

Falls zusätzliche Informationen benötigt werden, können (mit Stufe II) für die Task Proto im obigen Beispiel weitere Teilschritte abgefragt werden. So wird dann zum Beispiel deutlich, daß in der Phase der Clusteranalyse die Task aufgrund hoher Kommunikationskosten zunächst in ein anderes Cluster gelegt werden sollte.

Dies stellt die Erklärungskomponente mit folgendem Satz fest:

"Es wurde versucht, die Task Proto aufgrund der Kommunikationskosten mit der Task Druck in Cluster 4 zu legen".

Falls keine einzelnen Tasks betrachtet werden sollen, sondern der Gesamtablauf des Systems, werden die Antwortsätze in der Reihenfolge der zugehörigen Regeln aufgelistet.

Zusätzlich kann in der Erklärungskomponente die Historie der Regeln betrachtet werden; d.h. alle Regeln, die im Ablauf des Allokationssystems aktiviert wurden, sind in der Ausführungsreihenfolge aufgelistet.

#### 9.6. Ablauf des Allokationssystems

Das Kommunikationsdiagramm und das Hardwarediagramm werden mit Hilfe des graphischen Editors ZP der Firma S.E.P.P. /SEPP86/ erstellt. Die Auswertung der Diagramme erfolgt in zwei Schritten.

- Zunächst wird die Information der Diagramme mit Hilfe von zwei Programmen (KOMMSTR, PROZ) in eine Zwischensprache abgebildet. Daran anschließend wird mit weiteren Programmen die Semantik der Strukturen ausgewertet.
- Im Programm, das die Semantik der Kommunikationsstruktur auswertet (MAKEWME), werden vom Benutzer die Allokationsgebote und -verbote erfragt. Zu jeder Task kann der Benutzer die Namen der Tasks angeben, mit der sie auf einem gemeinsamen Prozessor liegen soll (bzw. mit der sie nicht auf demselben Prozessor zu liegen kommen darf).

Bei der Analyse der Hardwarestruktur (MAKEPROZ) muß angegeben werden, ob das (vom Benutzer festgelegte) Netzwerk bei der Allokation berücksichtigt werden soll, oder ob von einem vollständig vermaschten System ausgegangen werden kann.

Als Ergebnis dieser beiden Programme werden Working-Memory-Elemente erzeugt.

Nachdem die Eingaben an das Allokationsprogramm erstellt sind, kann das System ALLOC seine Arbeit aufnehmen. Als erstes muß nun angegeben werden, nach welchem Kriterium optimiert werden soll. Angegeben werden kann nun noch, ob das Allokationssystem alle Prozessoren auslasten muß, oder ob nicht unbedingt alle Prozessoren berücksichtigt werden müssen.

Falls das System Inkonsistenzen oder Fehler in der Spezifikation findet, wird der Benutzer davon unterrichtet und kann Änderungen zur Laufzeit angeben. Nachdem eine Verteilung errechnet worden ist, wird das Ergebnis graphisch dargestellt (Programm PARTY). Fehlende oder überflüssige Verbindungen werden besonders gekennzeichnet. Darüberhinaus kann sich der Benutzer die Ergebnisse erklären lassen. Die Erklärungskomponente gibt Aufschluß über die erreichte Verteilung.

In der Zeichnung 9.4. ist das gesamte Allokationssystem abgebildet.

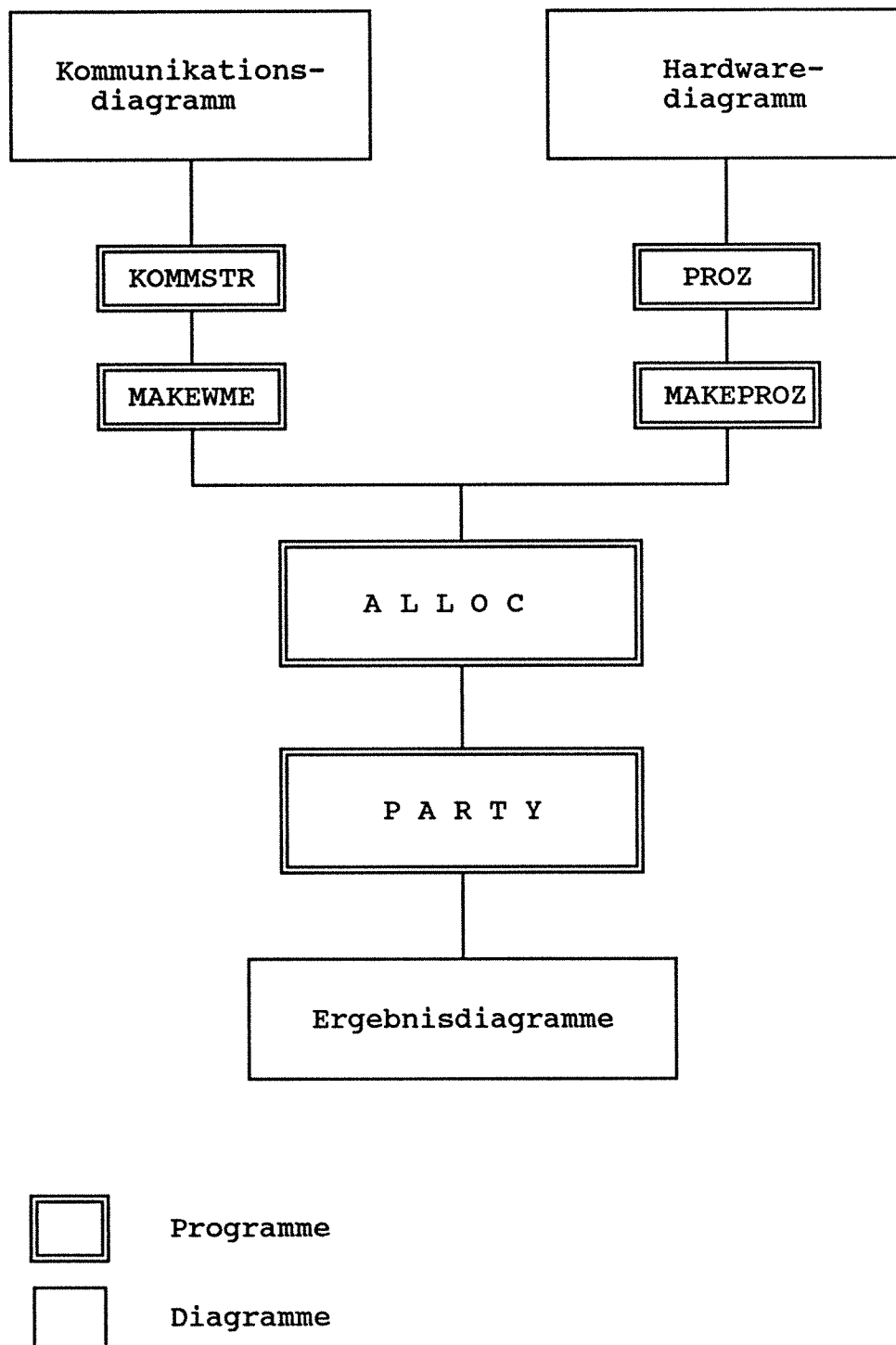


Abb. : 9.4. Übersicht über das gesamte Allokationssystem

### 9.7. Zusammenfassung

Die Umsetzung des Problems der Allokation in ein wissensbasiertes System konnte erfolgreich vorgenommen werden. Dies bedeutet insbesondere, daß die in Kapitel 8 geforderte Repräsentation des gesamten Expertenwissen in der Implementation Eingang gefunden hat. Die korrekte Arbeitsweise, und somit auch die geforderte Konsistenz der Regeln, konnte durch eine Reihe von Beispielen /BAUE89/ getestet werden. In dieser Arbeit wurde sowohl das System ALLOC in seiner Gesamtheit getestet, als auch die kritischen Phasen "Clusterbildung" und "Zuweisung zu den Prozessoren". Die Ergebnisse werden im nächsten Kapitel zusammengefasst.

Die Forderungen, an ein Allokationssystem im Rahmen der Realzeitprogrammierung (siehe Kapitel 6) gestellt werden, werden vom System ALLOC zufriedenstellend erfüllt:

- Die Umsetzung der graphischen Eingabe ermöglicht es dem Benutzer, seine Spezifizierung in PASS als Eingabe an das Allokationssystem zu verwenden.
- Die Auswertung der Kommunikationsstruktur und die Erzeugung der Working-Memory-Elemente erfolgen automatisch. Die erzeugten Working-Memory-Elemente repräsentieren die Kommunikationsstruktur und dienen als "Eingabe" für das wissensbasierte System.
- Das wissensbasierte System zur Lösung des Allokationsproblems wurde in der Sprache OPS implementiert. Durch die Unterteilung des Systems in Phasen, Regelmengen und Regeln, die die einzelnen Schritte der Verteilung simulieren, konnten die Arbeitsschritte, wie sie ein Experte ausführt, in geeigneter Weise im System ALLOC repräsentiert werden.
- Die graphische Ausgabeschnittstelle ermöglicht dem Benutzer eine übersichtliche Darstellung der erreichten Verteilung.
- Die Erklärungskomponente kann darüberhinaus die Ergebnisse der Verteilung näher erläutern.

---

Da alle geforderten Bedingungen (wie z.B. Einbeziehungen der Randbedingungen...) im Allokationssystem ALLOC berücksichtigt sind, ist es mit diesem System möglich, die Probleme aus der Praxis der Realzeitprogrammierung angemessen (im Sinne eines menschlichen Experten) zu lösen.

## **Kapitel 10**

### **Bewertung der Ergebnisse**

#### **10. Bewertung der Ergebnisse**

Bei der Lösung von (NP-vollständigen-) Problemen mittels heuristischer Verfahren gibt es keine Möglichkeit, die Qualität einer Lösung zu beweisen. Ob das Optimum gefunden wurde, kann nicht nachgewiesen werden.

Deshalb müssen die Ergebnisse heuristischer Verfahren experimentell nachgeprüft werden. Durch die Wahl geeigneter Testdaten muß sichergestellt sein, daß alle "wichtigen" Fälle berücksichtigt sind. Für die Bewertung der Ergebnisse muß ein Leistungsmaß gefunden werden, das alle möglichen Lösungen überdeckt.

In /BAUE89/ wird das wissensbasierte System verschiedenen Tests unterzogen, bei denen folgende Möglichkeiten berücksichtigt sind:

- Das Allokationssystem wird anhand der Ergebnisse von optimalen Verfahren bewertet. Dies bedeutet, daß das Allokationssystem (für Problemstellungen, die mit optimalen Verfahren lösbar sind) möglichst die gleichen Ergebnisse liefern sollte, wie die optimalen Methoden.

- 
- In Bereichen, in denen optimale Methoden das Allokationsproblem nicht mehr lösen können, muß das wissensbasierte System durch geeignete Testbeispiele überprüft werden. Um sicherzustellen, daß die Tests die verschiedenen Möglichkeiten des Allokationssystems abdecken, sind folgende Bereiche zu beachten:
    - Randbedingungen sind in geeigneter Weise in die Tests miteinzubeziehen.
    - Homogene und heterogene Hardwarestrukturen des Realzeitsystems müssen berücksichtigt werden.
    - Die Kommunikationsstruktur soll in verschiedene Extremfälle unterteilt sein, d.h. zu testen ist beispielsweise ein homogenes Realzeitsystem, bei dem die Kommunikationskosten zwischen den Tasks "gleichgroß" sind. Ein anderer Extremfall wäre ein Realzeitsystem, bei dem aufgrund der Verteilung der Kommunikationskosten eine klare Trennung der Tasks in Cluster möglich ist.
    - Die verschiedenen Optimierungsmöglichkeiten, die das Allokationssystem bietet, sind zu prüfen. Dabei muß vor allem untersucht werden, ob die einzelnen Kriterien zu unterschiedlichen Verteilungen kommen.
    - Die Auswirkungen des gewählten Netzwerkes auf die Verteilung sind zu beachten.
    - Mit Testfällen, die die oben genannten Forderungen widerspiegeln, wird das Allokationssystem getestet. Zusätzlich sind diese Testfälle kombiniert zu betrachten.

Die hier vorgeschlagenen (optimalen und suboptimalen) Testfälle sind so gewählt, daß die Qualität des Allokationssystems für verschiedene Problemstellungen (/BAUE89/) bewertet werden kann. Zusätzlich zu diesen Tests werden die Ergebnisse des wissensbasierten Systems mit denen eines anderen heuristischen Verfahrens verglichen.

Die Forderung nach Verwendbarkeit des Allokationssystems für die Realzeitprogrammierung wird durch die eigens entwickelten Problemstellungen überprüft. Unterstützt werden diese Tests durch die Einbeziehung eines Beispiels aus der Praxis.



### 10.1. Optimale Methoden

Zunächst wird das wissensbasierte System den optimalen Methoden gegenübergestellt. Dazu wurde ein Programm erstellt /BAUE89/, das nach der Methode der Integer-0/1-Verfahren (siehe Kapitel 5) arbeitet. Wie zu erwarten deckt das wissensbasierte System die Klasse der Verteilungsprobleme ab, die mit optimalen Methoden gelöst werden können.

Anhand eines der Testbeispiele aus /BAUE89/ wird die Behauptung nachgewiesen.

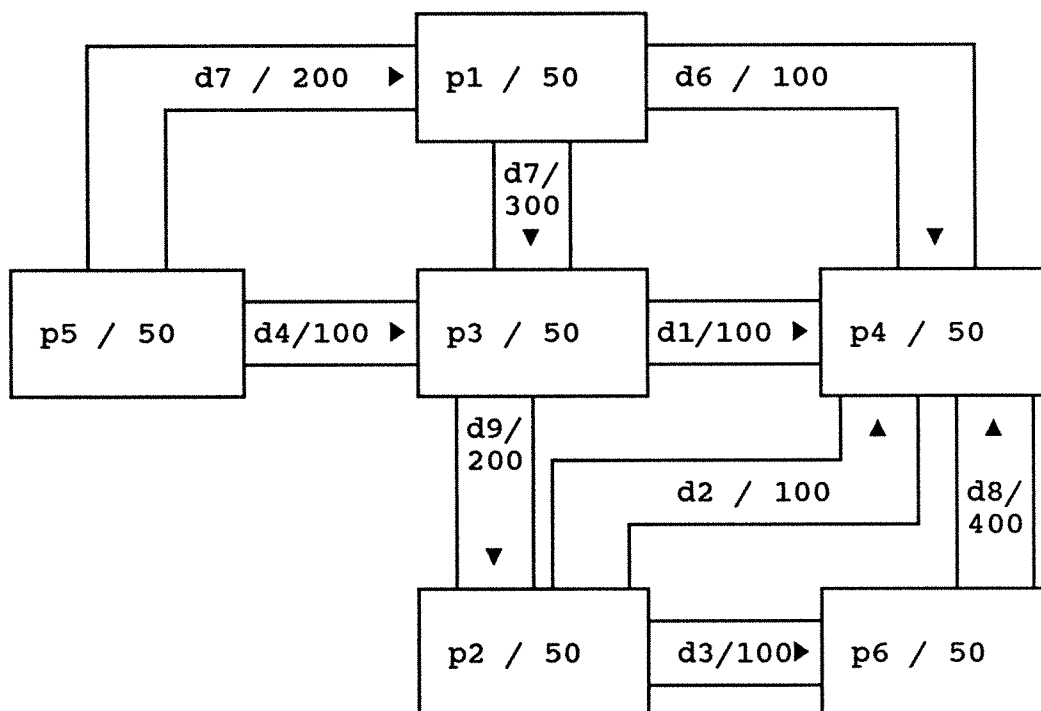


Abb. : 10.1. Kommunikationsstruktur

(Die Kästchen stellen die Tasks dar, die Beschriftung die Tasknamen und die Größe des Speicherplatzbedarfs. In den Kanten werden die Botschaftsnamen und die Größe der Kommunikationskosten beschrieben. Die Speichergrößen in Kbytes angegeben, die Kommunikationskosten in Bytes/s).

Dieses Beispiel, bestehend aus sechs Tasks, soll auf zwei Prozessoren verteilt werden. Die Speichergröße der Prozessoren kann bei der Methode der 0/1-

Programmierung nicht berücksichtigt werden, deshalb wird vorausgesetzt, daß beide Prozessoren (für das wissensbasierte System) gleich groß sind.

Das Integer-0/1-Programm liefert als Ergebnis folgende Verteilung:

Prozessor 1:	Prozessor 2:
Task: p5	Task: p1, p2, p3, p4, p6

Die IPC-kosten betragen: 300 Bytes/s.

Das wissensbasierte System liefert unter der Voraussetzung, daß die Speichergröße der Prozessoren groß genug gewählt wird (der Prozessor 2 muß groß genug sein, um die fünf Tasks aufzunehmen), das gleiche Ergebnis. Im vorliegenden Beispiel muß der Speicherplatz des Prozessors 2 mindestens 250 Kbytes betragen, um das "optimale Ergebnis" zu erreichen.

Wie wichtig die Randbedingungen bei der Allokation sind, wird durch folgendes Beispiel deutlich:

Bei einer Reduzierung der Speichergröße auf 150 Kbytes für jeden der Prozessoren erhält man mit dem wissensbasierten System folgende Verteilung:

Prozessor 1:	Prozessor 2:
Task: p1, p3, p5	Task: p2, p4, p6

Die IPC-Kosten betragen hierbei: 400 Bytes/s.

Diese Verteilung ist im Sinne der Minimierung der IPC-Kosten nicht optimal. Aufgrund des beschränkten Speicherplatzes kann aber keine bessere Verteilung gefunden werden. Die Verteilung ist bezüglich der Speicherplatzbeschränkung optimal. Zu diesem Ergebnis kann das optimale 0/1-Verfahren nicht gelangen, da es die zusätzliche Information über die Größe des Speicherplatzes nicht verarbeitet.

Dieses Beispiel zeigt, daß das wissensbasierte System zu den gleichen Ergebnissen kommt wie optimale Methoden. Zusätzlich wird die Überlegenheit des wissensbasierten Allokationssystems gegenüber optimalen Methoden in Bezug auf die Randbedingungen vorgeführt.

Bei allen betrachteten (optimalen) Testfällen errechnete das Allokationssystem die jeweils beste Lösung.

## 10.2. Heuristische Methoden

Das Allokationssystem soll auch im Bereich der Probleme, die nicht mit optimalen Methoden gelöst werden können, gute Lösungen errechnen. Um Aussagen über die Qualität solcher Lösungen zu treffen, ist es nicht ausreichend, die Ergebnisse des wissensbasierten Systems denen eines anderen Verfahrens gegenüberzustellen, da nicht bewiesen werden kann, daß die Ergebnisse anderer heuristischer Verfahren optimal sind. Damit dennoch die Lösungsqualität in diesem Problembereich getestet werden kann, wurden Teilbereiche der Allokation untersucht, für die die Ergebnisse leichter zu bewerten sind:

- Auswirkungen der verschiedenen Optimierungskriterien.

Das Allokationssystem kann nach unterschiedlichen Kriterien optimieren (siehe Kapitel 8). Die Ergebnisse der verschiedenen Optimierungskriterien werden durch Testbeispiele daraufhin untersucht, inwieweit sie sich voneinander unterscheiden.

Aufgrund der Testergebnisse läßt sich feststellen, daß die Optimierungskriterien die Verteilung erheblich beeinflussen. Es entstehen verschiedene Lösungsvorschläge für eine Verteilung. Aus den Ergebnissen kann der Benutzer des Allokationssystem das für seine Aufgabenstellung am besten geeignete auswählen. Die unterschiedlichen Auswirkungen der Optimierungskriterien sind auch aus den Ergebnissen des Beispiels in Anhang I ersichtlich.

- Auswirkungen kleiner Änderungen.

Hier wurden systematisch die Eingabedaten verändert, um die Auswirkungen auf das Ergebnis zu beobachten. Damit wird untersucht, ob das Allokationssystem auch auf kleine Veränderungen sensibel reagiert.

An einer bereits erprobten Kommunikationsstruktur wurden gezielt "kleine" Änderungen vorgenommen, deren Auswirkungen auf die Verteilung leicht zu übersehen sind. Das Ergebnis der Verteilung wird mit dem verglichen, das die "alte" Verteilung lieferte. Die Ergebnisse waren in allen betrachteten Fällen die vom Benutzer erwarteten.

---

- Auswirkungen der Randbedingungen.

Während in den beiden ersten Testbeispielen Systeme ohne Randbedingungen (z.B. Prozeßperipherie) betrachtet werden, sollen mit diesen Tests die Auswirkungen der Randbedingungen auf die Verteilung näher untersucht werden. Die Beispiele sind so konstruiert, daß die Auswirkungen verschiedener Randbedingungen für eine Task näher untersucht werden können. Auch hier konnte die richtige Arbeitsweise des Allokationssystems, vor allem bei kombinierten Randbedingungen, nachgewiesen werden.

- "Extremfälle" von Taskanordnungen.

Unter "extremen" Taskanordnungen sind (/BAUE89/) solche Fälle zu verstehen, die eine regelmäßige Anordnung der Tasks aufweisen.

So wird beispielsweise eine ringförmige Anordnung von Tasks (bzw. eine sternförmige Anordnung) auf eine Prozessorstruktur bestehend aus vier, ebenfalls ringförmig miteinander verbundenen Prozessoren verteilt. Mit diesen Tests wird die Fähigkeit des Allokationssystem überprüft, regelmäßige Strukturen geeignet aufzubrechen und zu verteilen. Aufgrund der verschiedenen gewählten Testbeispiele, bei denen zusätzliche Einschränkungen betrachtet wurden, kann festgestellt werden, daß das Allokationssystem diese Aufgabe im Sinne eines Experten bewältigt.

- Zusammenwirken verschiedener Zuordnungskriterien.

Die oben aufgeführten Beispiele beschäftigen sich jeweils mit einem bestimmten Teilbereich der Allokation. Eine Kombination aller Bereiche ermöglicht Aussagen über die Auswirkungen der Teilaspekte auf die Allokation. Getestet wird damit auch, wie das Allokationssystem die einzelnen Verteilungskriterien berücksichtigt. So kann beispielsweise beobachtet werden, daß bei verschiedenen Möglichkeiten einer Taskzuteilung im Zweifelsfall die Minimierung der IPC-Kosten ausschlaggebend für die Zuweisung ist. In diesem Bereich wurden umfangreiche Testbeispiele entworfen und mit den verschiedenen Optimierungskriterien ausgewertet.

Wie bereits ausgeführt, besteht keine Möglichkeit, die Optimalität der Lösungen zu beweisen. Die Korrektheit der Ergebnisse kann nur durch Plausibilitätsprüfungen verdeutlicht werden. Die Testbeispiele sind deshalb so gewählt, daß die Ergebnisse nachvollziehbar sind.

Dennoch zeigte sich während der Testphase, daß das Allokationssystem unerwartete Ergebnisse errechnete, die allerdings in den meisten Fällen besser waren, als die vorher "von Hand" erstellten.

Die Beurteilung der Allokationsergebnisse erwies sich für einige der Testbeispiele als ziemlich schwierig. Deshalb wurden die Entscheidungen, die das Allokationssystem im einzelnen getroffen hat, mit Hilfe der Erklärungskomponente kontrolliert. Ob eine Verteilung sinnvoll ist, konnte dadurch verifiziert werden, daß die Aussagen der Erklärungskomponente betrachtet wurden. Anhand der Hinweise, die zum "Lebenslauf" einer Task bzw. zum Gesamtverlauf der Allokation gegeben werden, kann die Beurteilung der Verteilung erleichtert werden. Die einzelnen Schritte, die zu einer Verteilung führen, sind nachvollziehbar, wodurch die Bewertung einfacher wird. Durch das Nachvollziehen der wichtigsten Allokationsschritte mit Hilfe der Erklärungskomponente, kann in jeder Phase die Richtigkeit der Verteilung geprüft werden. Besonders wichtig sind die Entscheidungen, die vom Allokationssystem bei der Clusteranalyse und der Zuteilung der Cluster an die Prozessoren getroffen werden.

Mit Hilfe der Informationen, die die Erklärungskomponente gibt, erleichtert sich die Beurteilung der errechneten Verteilung erheblich.

Mit den umfangreichen Testbeispielen (/BAUE89/) konnte die korrekte Arbeitsweise des Allokationssystem selbst in kritischen Bereichen nachgewiesen werden. Eine Verteilung wird in diesem Fall als korrekt bezeichnet, wenn sie (für ein bestimmtes Optimierungskriterium) der von einem menschlichen Experten erstellten gleich ist, oder nur in dem vom Experten erlaubten Rahmen abweicht.

### 10.3. Vergleich mit einem heuristischen Ansatz

Die Ergebnisse der konstruierten Beispiele zeigen bereits, daß das Allokationssystem die Erwartungen erfüllt. Dennoch soll das Allokationssystem mit einem anderen heuristische Verfahren verglichen werden, um eine realistische Einordnung des Systems zu erhalten.

Als Vergleichsgrundlage wird das in Kapitel 6 vorgestellte heuristische Verfahren von Borrmann /BORR86/ herangezogen. Dort wird ein Beispiel vorgestellt, bei dem zehn Tasks auf 4 Prozessoren verteilt werden sollen.

Die Prozessoren sind durch ein vollständig vermaschtes System miteinander verbunden.

Von den zehn Tasks sind acht in Speicherplatzbedarf und Laufzeitbedarf identisch. Jede dieser acht Tasks kommuniziert mit allen anderen. Die beiden restlichen Tasks (Task neun und zehn) benötigen Prozeßperipherie und haben zu allen andern acht Tasks Verbindung. Die Kommunikationskosten der Tasks können der Abbildung 10.2. entnommen werden. Die Kapazitäten und die daraus resultierende Volumenmatrix können aus der Abbildung entnommen werden.

Der Speicherplatzbedarf der Task beträgt für Task eins bis acht 41 Kbytes, für die Task neun 5 Kbytes und die Task zehn 10 Kbytes (Bedarfsmatrix).

Der Prozessor 1 besitzt die Prozeßperipherie, die die Task 10 anfordert und der Prozessor 4 erfüllt die Anforderungen der Task 9 (Kapazitätsmatrix). Die Kosten für die Leitungen werden mit:

100 bytes/s für die Verbindungen des Prozessors 1 mit allen anderen Prozessoren und mit 20 bytes/s für die übrigen Verbindungen angegeben (Entfernungsmatrix).

Die Speichergröße aller Prozessoren ist mit 128 Kbytes angegeben (Kapazitätsmatrix). (In Borrmann werden kwords verwendet).

Abbildung 10.2. zeigt das Kommunikationsverhalten der 10 Tasks.

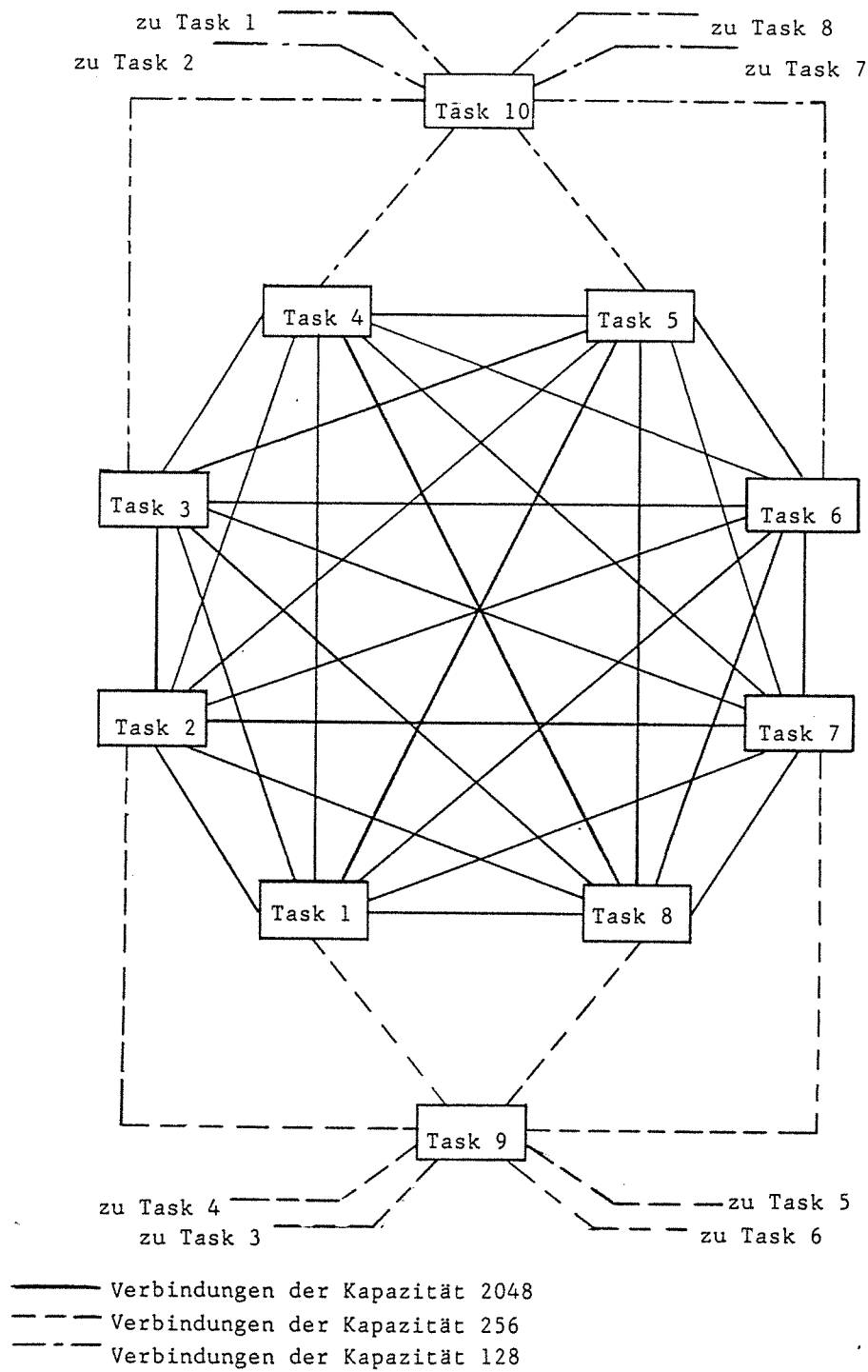


Abb. : 10.2. Kommunikationsstruktur nach /BORR86/

Die Verteilung, die mit dem Verfahren nach Borrmann errechnet wird, stellt sich in folgendermaßen dar:

Prozessor 1:	10
Prozessor 2:	6, 8
Prozessor 3:	1, 4, 7
Prozessor 4:	2, 3, 5, 9

Die IPC-Kosten des System errechnen sich (ohne Berücksichtigung der Leitungskapazitäten) zu 45212 byte/s.

Das Allokationssystem ALLOC kommt mit dem Kriterium "Optimieren nach Kommunikationskosten" zu folgendem Ergebnis:

Prozessor 1:	10
Prozessor 2:	1, 5, 7, 8
Prozessor 3:	2, 3, 4, 6
Prozessor 4:	9

Die IPC-Kosten des Systems errechnen sich zu 35840 byte/s.

Mit dem Kriterium "Optimieren nach Kommunikationskosten und Speicherplatz" wird ein anderes Ergebnis errechnet:

Prozessor 1:	5, 7, 10
Prozessor 2:	3, 8
Prozessor 3:	4, 6
Prozessor 4:	1, 2, 9

Die IPC-Kosten des Systems errechnen sich nun zu 51456 byte/s.

Ein Vergleich der Ergebnisse zeigt, daß das wissensbasierte System für das Optimierungskriterium "Kommunikationskosten" eine in Bezug auf die IPC-Kosten bessere Verteilung errechnet als das Verfahren nach Borrmann.

Bei dieser Verteilung (mit ALLOC) sind allerdings die Prozessoren 2 und 3 überlastet. Dies erkennt das wissensbasierte System und setzt den Benutzer



zur Laufzeit davon in Kenntniss. Der Vorteil, den diese Verteilung in Bezug auf die Kommunikationskosten bringt, wird höher bewertet, als der damit verbundene Nachteil der Überlastung der Prozessoren. Da die Randbedingung "Prozessoren nicht überlasten" verletzt wird, muß das wissensbasierte System den Benutzer davon in Kenntnis setzten.

Beim oben angegebenen Ergebnis wurde dem System signalisiert, eine Verteilung mit einer Überlastung des Speichers fortzuführen. Falls keine Überlastung des Speichers erlaubt wird, kommt das Allokationssystem ALLOC zum gleichen Ergebnis wie das System von Borrmann.

Für das Kriterium "Kommunikationskosten und gleichmäßige Speicherauslastung" ergeben sich höhere Kosten. Die Lösung, die hierbei erreicht wird, ist zwar in Bezug auf die Speicherauslastung optimal, die Kommunikationskosten erreichen hier jedoch das Maximum. Das Allokationssystem hat die Gleichauslastung des Speicherplatzes zugunsten der Kommunikationskosten in den Vordergrund gestellt.

Da die Verbindungen zwischen den Prozessoren nicht homogen sind, müssen bei der Bestimmung der IPC-Kosten die Leitungskapazitäten berücksichtigt werden. Bei Borrmann führt dies zu IPC-Kosten von 1077760, während für das wissensbasierte System 880540 als Kosten entstehen.

Die Forderung nach gleichmäßiger Auslastung der Prozessoren in Bezug auf Speicherplatz, wie auch auf Laufzeit, ist bei allen Optimierungskriterien (bis auf "Optimieren nach Kommunikationskosten") vom wissensbasierten Systems besser erfüllt.

In der folgenden Tabelle ist die Prozessorauslastung nach Borrmann der des wissensbasierten Systems mit Optimierungskriterium "Kommunikationskosten und gleichmäßige Speicherauslastung" gegenübergestellt:

	ALLOC	BORR86
Prozessor 1:	70%	8%
Prozessor 2:	62%	62%
Prozessor 3:	62%	95%
Prozessor 4:	67%	100%

Der Vergleich verschiedener Lösungen des wissensbasierten Systems mit dem Ergebnis nach Borrmann unterstreicht die Forderung nach differenzierten Optimierungsmöglichkeiten. Das Ergebnis, das mit der Methode von Borrmann errechnet wird, ist in Bezug auf die IPC-Kosten gut, berücksichtigt aber nicht gleichmäßige Prozessorauslastung oder gleichmäßige Laufzeiten.

Beim Allokationssystem ALLOC können verschiedene Optimierungskriterien betrachtet werden. Aufgrund der Ergebnisse wird ein Urteil über die Verteilung leichter möglich, als mit einer einzigen Optimierungsmöglichkeit.

Der Vergleich zeigt auch, wie wichtig eine Möglichkeit zur Rückfrage beim Benutzer sein kann. Im Beispiel sind (wie oben gezeigt) zwei verschiedene Verteilungen möglich. Wenn eine Randbedingung (geringfügig) verletzt wird, läßt sich eine bessere Verteilung in Bezug auf die Kommunikationskosten errechnen. Deswegen fragt das System, welcher Verteilung der Vorzug gegeben werden soll. Die Verteilung kann also durch die zusätzliche (Benutzer-) Information verändert werden.

#### 10.4. Beispiel aus der Realzeitpraxis

Um den Einsatz des Allokationssystems auch an praktischen Aufgaben nachvollziehen zu können, wird eine existierende (verteilte) Realzeitaufgabe mit dem Allokationssystem neu bearbeitet.

Am Beispiel eines am RRZE (Regionales Rechenzentrum Erlangen) entwickelten verteilten Rechnersystems /HEIL85/ zur redundanten Erfassung von Telefongesprächsdaten wurde das Allokationssystem getestet.

Das Telefonsystem besteht aus vier Prozessoren, die untereinander gekoppelt sind. Zwei der Prozessoren dienen zur (redundanten) Datenerfassung. Der dritte Prozessor dient hauptsächlich zur Kommunikation mit dem Benutzer, der vierte Prozessor übernimmt die Übertragung der Daten an einen Host-Rechner. Die beiden Erfassungsrechner sind mit dem Dialogrechner verbunden, und dieser hat Verbindung zum Übertragungsrechner (siehe Abbildung 2.2. und /BAUE89/).

Am Telefonsystem sind 15 Tasks beteiligt, die sich mit der Datenaufzeichnung, der Auswertung, Protokollierung und der Übermittlung der Daten an einen Host-Rechner beschäftigen. Aufgrund der Tätigkeiten, die sie ausführen sollen, sind manche Tasks bestimmten Prozessoren unbedingt zuzuordnen. So müssen die (redundant vorhandenen) Tasks, die die Daten von der Leitung entgegennehmen, den Erfassungsrechnern zugeteilt werden, da nur dort die Aufnahmeleitungen vorhanden sind.

Die vom Allokationssystem errechnete Verteilung ist identisch mit der von den Systementwicklern (/HEIL85/, TRAU88/) vorgeschlagenen. Da für einige Tasks bereits feste Zuteilung aufgrund der Hardwareanforderungen bestehen, bleibt dem Allokationssystem wenig Freiheit für seine Verteilung.

Um die Entscheidungsfreiheit für die Verteilung zu vergrößern, werden verschiedene Zusatzbedingungen und Erweiterungen an der Spezifikation des Telefonsystems vorgenommen:

- Die Hardware des Dialogrechners wurde auch am Übertragungsrechner zur Verfügung gestellt.
- Um dem Allokationssystem die Möglichkeit zu geben, jedem Prozessor eine "beliebige" Anzahl von Tasks zuzuordnen, wird die Speicherkapazität vergrößert.

Mit diesen Zusatzbedingungen hat das Allokationssystem eine wesentlich größere Freiheit bei der Verteilung. Die Ergebnisse sind aber für alle Optimierungskriterien ähnlich der vom Entwerfer vorgegebenen Lösung. Lediglich bei der Optimierung nach reinen Kommunikationskosten wird ein abweichendes Ergebnis errechnet, bei dem zwar die IPC-Kosten erwartungsgemäß niedrig sind, dafür der Übertragungsrechner aber nur eine Task (die für die Übertragung an den Host-Rechner) zugeordnet bekommt.

Eine Veränderung bringt die Auswertung des Systems unter der Annahme eines vollständig vermaschten Netzwerkes. Das Allokationssystem verfügt damit über größere Verteilungsfreiheiten. Dies bewirkt, daß die Prozessoren bei gleichbleibenden IPC-Kosten gleichmäßiger ausgelastet sind.

Ein deutlich verbessertes Kommunikationsverhalten wird erreicht, wenn der Übertragungsrechner nicht berücksichtigt wird. Das Verhältnis der Kommunikationskosten zwischen Tasks, die auf demselben Prozessor liegen, und solchen, die auf unterschiedlichen Prozessoren liegen verringert sich von 9:4 auf 6:4. Um diese Verteilung zu ermöglichen, muß die Hardware des Übertragungsrechners am Dialogrechner verfügbar sein.

Ein System von zwei Prozessoren bringt hingegen keine so deutliche Verbesserung des Verhältnisses der Kommunikationskosten. Für die Zwei-Prozessor-Version beträgt das Verhältnis 5:4. Die "übriggebliebenen" Kommunikationskosten zwischen den Prozessoren entstehen hauptsächlich durch den Datentransfer zwischen einer (redundanten) Erfassungstask und der Verarbeitungstask. Der Kommunikationsaufwand läßt sich im Telefonsystem aufgrund der redundant vorhandenen Datenerfassung nicht weiter verringern, sodaß ein Hardwaresystem aus drei Prozessoren unter den gegebenen Umständen die beste Lösung darstellt.

Für die Realisierung muß im konkreten Fall allerdings der Dialogrechner mit mehr Speicherplatz ausgestattet sein.

Da alle Ergebnisse nicht erheblich von der vorgegebenen Lösung abweichen, bestätigt dies die Qualität des Allokationssystems.

### 10.5. Zusammenfassung

Mit Hilfe umfangreicher Testbeispiele konnte gezeigt werden, daß das Allokationssystem vernünftige Ergebnisse liefert.

Insbesondere bei Systemen, die mit optimalen Methoden gelöst werden können, kam das Allokationssystem in allen Fällen zu den gleichen (optimalen) Ergebnissen.

Für Systeme, die nur mittels heuristischer Algorithmen gelöst werden können, wurde durch eine Reihe geeigneter Testdaten gezeigt, daß das Allokationssystem in vernünftiger Zeit gute Ergebnisse liefert. Verschiedene Extremfälle, wie sie in /BAUE89/ getestet werden, führten zu sehr guten Ergebnissen.

Mit den Problemen aus der Praxis konnte festgestellt werden, daß das Allokationssystem zu den gleichen Lösungen kam wie sie von den Experten festgelegt werden (/HEIL85/, /BESO85/, /TRAU88/).

Darüberhinaus erwiesen sich die verschiedenen Optimierungskriterien als geeignetes Mittel, um die Verteilung eines Realzeitsystems geeignet zu untersuchen. Vor allem am Beispiel des Telefonsystems konnten mit den verschiedenen Optimierungskriterien neue Erkenntnisse über eine geeignete Verteilung auf die Hardware gewonnen werden.

Die Zeiten, die vom wissensbasierten System ALLOC zur Erstellung einer Lösung benötigt wurden, liegen bei allen Testbeispielen (/BAUE89/) unter zwei Minuten.

## **Kapitel 11**

### **Zusammenfassung**

#### **11. Zusammenfassung**

Ziel dieser Arbeit war es, ein Allokationssystem zu schaffen, das den Anforderungen der Realzeitprogrammierung gerecht wird. Mit Hilfe der Spezifikation einer Realzeitaufgabe und der gegebenen Hardwarekonfiguration sollte das System die Tasks im Sinne der Realzeitanforderungen optimal auf Prozessoren verteilen, wobei gleichzeitig die Randbedingungen berücksichtigt werden mußten.

Um dieses Ziel zu erreichen, wurden bislang optimale und heuristische Methoden verwendet. Die optimalen Algorithmen können, da das Problem der Allokation NP-vollständig ist, nur für "kleine" Systeme im praktischen Gebrauch (bis zu drei Prozessoren) Anwendung finden. Für größere Systeme können nur die heuristischen Algorithmen in vernünftiger Zeit eine Lösung bringen.

Das vorhandene Expertenwissen bei der Allokation wurde bei bestehenden Allokationssystemen zu wenig berücksichtigt. In der vorliegenden Arbeit wurde dieses Wissen erforscht, in geeignete Strukturen zur Repräsentierung umgesetzt und diese Strukturen in einem wissensbasierten System (ALLOC) realisiert.

Um dem wissensbasierten System ALLOC die Problematik der jeweiligen Realzeitaufgabe auf abstrakter Ebene nahezubringen, wurde zur Eingabe das Spezifikationsniveau gewählt. Da das Allokationssystem ALLOC in eine Entwicklungsumgebung für verteilte Realzeitsysteme /HOLL89/ integriert ist, dient eine Spezifikation nun mehreren Zielen: Sie ist Basis für die Validierung der Realzeitaufgabe, Ausgangspunkt für automatische Codeerzeugung und jetzt auch Eingabe für das Allokationssystem.

Zur Veranschaulichung der Ergebnisse für den Anwender des Allokationssystems wurden eine graphische Ausgabe und darüberhinaus eine Erklärungskomponente geschaffen.

Mit diesem wissensbasierten Ansatz zur Lösung des Allokationsproblems wurde ein System geschaffen, das für den Einsatz in der Realzeitprogrammierung geeignet ist. Da verschiedene Randbedingungen (periphere Geräte, Speichergröße usw.) und unterschiedliche Optimierungskriterien berücksichtigt wurden und die Vorgehensweise eines Experten realisiert ist, können Aufgabenstellungen, wie sie in der Realzeitprogrammierung anfallen, zufriedenstellend gelöst werden.

Anhand einer prototypischen Implementierung konnte gezeigt werden, wie das Expertenwissen bei der Allokation nutzbar gemacht werden kann. Für kleine Allokationsprobleme werden mit dem System ALLOC die gleichen Ergebnisse errechnet wie mit den optimalen Methoden; den untersuchten heuristischen Ansätzen ist das wissensbasierte System überlegen. Mit dem System ALLOC werden Lösungen des Problems erreicht, die den Anforderungen aus der Praxis entsprechen.

Die einbezogenen Randbedingungen und die Verteilungs- und Verfeinerungsstrategien erheben nicht den Anspruch auf Vollständigkeit, es sollte vielmehr beispielhaft gezeigt werden, wie Expertenwissen bei der Allokation repräsentiert werden kann.

Aufgrund der gewählten Umsetzung des Fachwissens in Phasen, Regelmengen und Regeln sowie des Metawissens in die Strategie der Inferenzmaschine kann das System leicht um Randbedingungen und neue Verteilungsstrategien erweitert werden. Im Fall der Randbedingungen sind dazu lediglich neue Regeln, im anderen Fall neue Regelmengen notwendig. Falls weitere Verfahrensschritte eingefügt werden sollen (z.B. eine zusätzliche Verfeinerungsstufe, Erkenntnisse

aus Requirements-Analysen), genügt es, dem System eine weitere Phase einzugliedern. Das System ALLOC kann leicht erweitert werden, ohne daß man die grundlegenden Strukturen verändern muß.

Aufgrund der Erfahrungen mit dem Allokationssystem ergeben sich für weitere Arbeiten auf dem Gebiet der Allokation folgende Forderungen:

- Da bei Realzeitsystemen trotz schneller Verbindungen zwischen den Prozessoren Engpässe auftreten können (/NIEM88/), sollte eine Erweiterung des Allokationssystems die Netzwerkparameter stärker berücksichtigen. Dazu müssen Regeln entwickelt werden, die über die Einhaltung der Netzwerk-anforderungen wachen. Engpässe bei der Datenübertragung zwischen Rechnern können dadurch verringert werden.
- Durch die Implementierung des Systems auf einem Personalcomputer sind dem Allokationssystem (in Bezug auf den zur Verfügung stehenden Speicherplatz) Grenzen gesetzt. Für große Probleme aus der Realzeitprogrammierung kann aufgrund dieser Beschränkung keine Verteilung errechnet werden. Mit einer Erweiterung des Systems auf größeren Computern kann auch ein verbessertes Backtracking implementiert werden. In schwierigen Situationen können damit Entscheidungen rückgängig gemacht werden und aufgrund dieser "Erfahrungen" neue Verteilungsversuche erfolgen.

Die wesentlichen Leistungen des wissensbasierten Allokationssystems (ALLOC) sollen noch einmal überblicksweise zusammengestellt werden:

- Die Arbeit ist in eine bestehende Programmierungsumgebung für verteilte Realzeitanwendungen eingebunden, wobei die Information für die Eingabe an das wissensbasierte System automatisch aus der Spezifikation gewonnen wird.
- Die Randbedingungen, die bei der Verteilung von Tasks eine wichtige Rolle spielen, sind in das Allokationssystem mit einbezogen.
- Es kann nach verschiedenen Gesichtspunkten optimiert werden. Auch kombinierte Optimierungskriterien können verfolgt werden.
- Das Wissen über die Verteilung der Tasks ist in der Wissensbasis festgelegt.
- Die Ergebnisse des Allokationssystems werden in übersichtlicher Form graphisch dargestellt. Die Entscheidungen des Systems werden dem Benutzer von einer Erklärungskomponente dargelegt und er wird auf Fehler und Mängel in der Spezifikation aufmerksam gemacht.



# **Anhang I**

## **I. Beispiel für die Anwendung**

### **A. Beschreibung der Eingabe**

An einem Problem aus der Realzeitprogrammierung wird die komplette Eingabe an das Allokationssystem aufgezeigt. Ausgehend von der Beschreibung, die bereits in einem Kommunikationsdiagramm existiert, wurde für die Allokation die Syntax zur Beschriftung der Knoten und Kanten um die notwendigen Parameter erweitert. Im folgenden werden diese Erweiterungen erklärt.

Im Kommunikationsdiagramm ist für die Botschaften (Kanten) beispielsweise folgende Darstellung erlaubt:

Botschaft\_von\_Prozeß\_A\_an\_Prozeß\_B;

An die vorhandenen Botschaften bzw. Namen für die Tasks wird für das Allokationssystem die entsprechende Information einfach angehängt, so daß der Benutzer seine Spezifikation für die Allokation nur erweitern muß. Zur Beschreibung der Syntax sei auf den Anhang II verwiesen.

Vorgegeben sei ein Kommunikationsdiagramm aus einer in PASS erstellten Realzeitspezifikation (Bild I.1.). In dieses Kommunikationsdiagramm werden die Informationen eingefügt, die für die Allokation notwendig sind (Bild I.2). Zusätzlich zum Kommunikationsdiagramm wird ein Hardwarediagramm erstellt, das die vorhandene Hardware beschreibt (Bild I.3).

(Zur besseren Übersicht wurde auf die Angabe der Benennungen in den Diagrammen verzichtet).

### Kommunikationsdiagramm<sup>1</sup>

Das Kommunikationsdiagramm besteht aus den Tasks (STELLEN, AUSWERT, WACH, DIALOG, und LADE), und den Botschaften zwischen den Tasks. Außerdem wird die Hardware angegeben, mit der eine Task kommunizieren will (DIALOG-Drucker; LADE-Floppy). Die Tasks werden durch einfache Kästchen dargestellt. Die Hardware ist durch Kästchen mit doppelten Linien gekennzeichnet. (Der Einfachheit halber wurde die Richtung der Botschaften nur durch eine Pfeilspitze angegeben).

#### Taskknoten

Die Namen der Tasks bleiben bestehen, die Angaben über den Speicherplatzbedarf der Task, sowie deren Laufzeitbedarf werden angehängt. Bei der Task WACH wurde zusätzlich ein Task-Prozessor-Verbot (-1) angegeben.

Eine Beschreibung der Tasks sieht folgendermaßen aus:

WACH / 50:Kbytes / 1:Mips / -1 #

#### Hardwareknoten

- Die Hardwareknoten enthalten zusätzliche Angaben darüber, ob die Task dem Prozessor zugewiesen wird, der eine ausgezeichnete Prozeßperipherie besitzt (dargestellt durch eine Ziffer, die mit der im Hardwarediagramm übereinstimmen muß), oder ob nur gefordert wird, daß der Prozessor eine bestimmten Prozeßperipherie besitzt (Hardware-\*).

Drucker-1 bzw. Floppy-\*

---

<sup>1</sup> Die Symbole für die Beschreibung der Kommunikationsstruktur und der Hardwarestruktur mit einem graphischen Editor werden in Anhang III erläutert.

### Botschaften

Die Botschaften werden um die Kommunikationskosten erweitert (diese Beschriftung gilt für Botschaften zwischen Tasks, sowie zwischen Task und der Prozeßperipherie). Die Kommunikationskosten werden an den letzten Botschaftsnamen angehängt. Im Gegensatz zu der symmetrischen Darstellung in Kapitel 2 werden hier alle Kommunikationskosten nur "einfach" (unsymmetrisch) verarbeitet. Es wird (in der formalen Darstellung nach Kapitel 2) nur eine der beiden Dreiecksmatrizen betrachtet.

Überwachen / 50:Kbytes/sec

### Hardwarediagramm

Das Hardwarediagramm enthält Angaben über die Prozessoren, deren Leistungsfähigkeit, die Hauptspeichergröße und die Verbindungen zwischen den Prozessoren. Die peripheren Geräte der einzelnen Prozessoren sind ebenfalls dargestellt. Die Prozessoren sind durch einfache Kästchen, die Prozeßperipherie ist durch Kästchen mit doppelten Linien dargestellt.

### Prozessorknoten

Die Beschriftung der Prozessoren enthält die Prozessornummer, die Hauptspeichergröße und die Leistungsfähigkeit:

1 / 256:Kbytes / 6:Mips

### Prozeßperipherie-Knoten

In der Beschriftung der Prozeßperipherie wird deren Name und die Zugehörigkeit zu einem Prozessor angegeben:

Floppy-1

### Verbindungen

Die Kapazitäten der Leitungen zwischen den Prozessoren, und zwischen Prozessoren und der Prozeßperipherie werden folgendermaßen dargestellt:

64 / Kbytes/sec

### Beschreibung des Beispiels

#### Kommunikationsdiagramm

Das Kommunikationsdiagramm besteht aus 5 Tasks STELLEN, AUSWERT, DIALOG, LADE und WACH. Die Kommunikation zwischen den Tasks und die zugehörigen Kosten können der Abbildung I.2 entnommen werden. Für die Task WACH besteht ein Zuordnungsverbot zu Prozessor 1. Die Task DIALOG benötigt einen Drucker, der als Drucker-1 näher spezifiziert ist. Die Task LADE benötigt Zugriff zu einem Floppylaufwerk. Die Floppy wurde nicht näher spezifiziert (Floppy-\*); dies bedeutet, daß die Task einem beliebigen Prozessor zugeordnet werden darf, der über ein Floppylaufwerk verfügt.

Die Angaben über den Speicherplatzbedarf sind in Kbytes, die über den Laufzeitbedarf in Kips zu verstehen. Die Kommunikationskosten sind in Kbytes/sec angegeben.

#### Hardwarediagramm

Das Hardwarediagramm enthält im Beispiel drei Prozessoren, von denen Prozessor 2 mit den Prozessoren 1 und 3 verbunden ist. Zwischen Prozessor 1 und 3 existiert keine Verbindung. Prozessor 1 hat Zugriff zu einer Floppy und einem Drucker, Prozessor 2 hat Floppyzugriff, während Prozessor 3 keine Prozeßperipherie besitzt. Der Speicherplatz des Prozessors 1 beträgt 256 (Kbytes), die der Prozessoren 2 und 3 640 (Kbytes). Die Geschwindigkeiten, mit der Befehle bearbeitet werden können betragen 0.6 (Mips) für Prozessor 1, bzw. 0.1 Mips für die Prozessoren 2 und 3. Die Übertragungsraten des Netzwerkes sind mit 64 Kbytes/sec angegeben.

Zur besseren Übersicht wurden im Beispiel die Benennungen weggelassen.

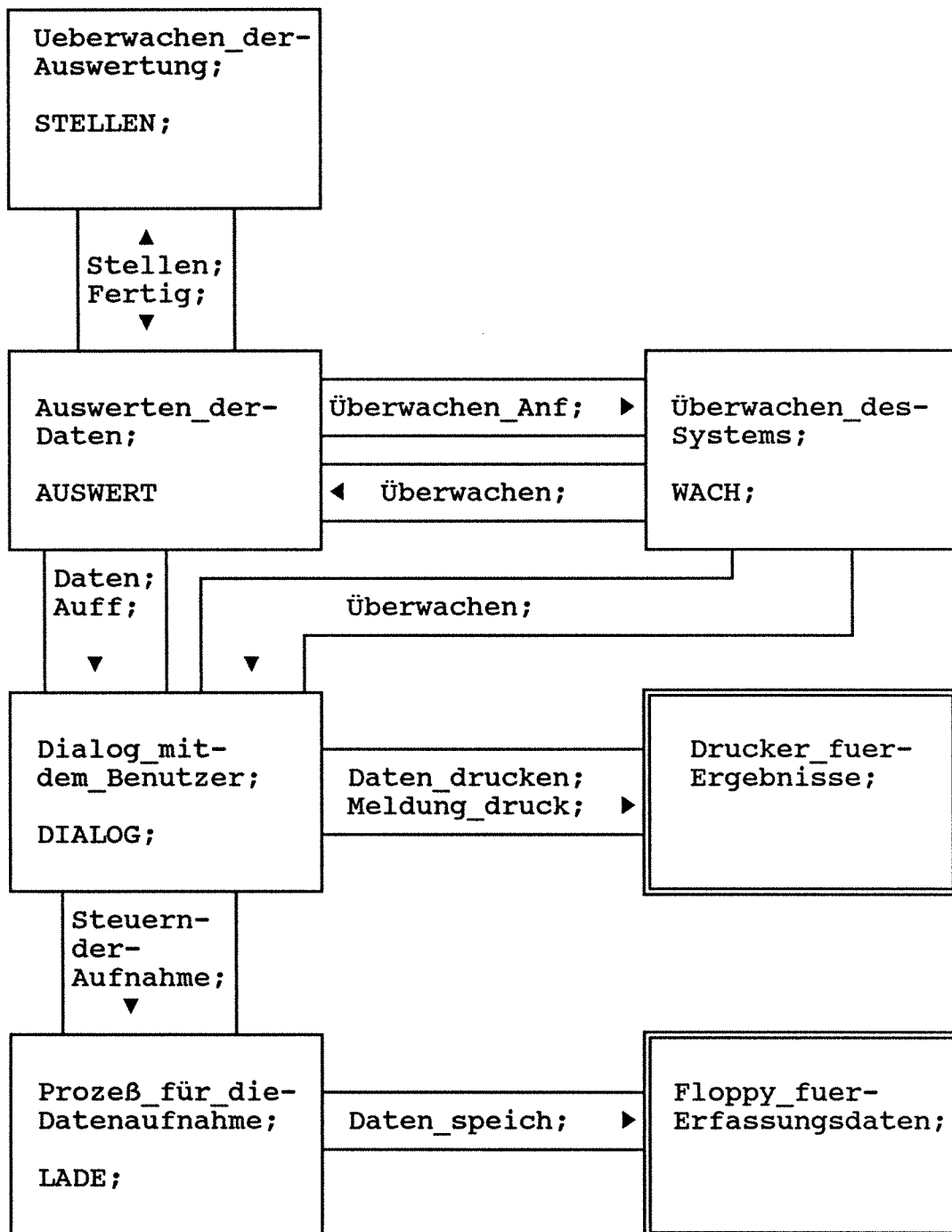


Abb. : I.1. Kommunikationsdiagramm

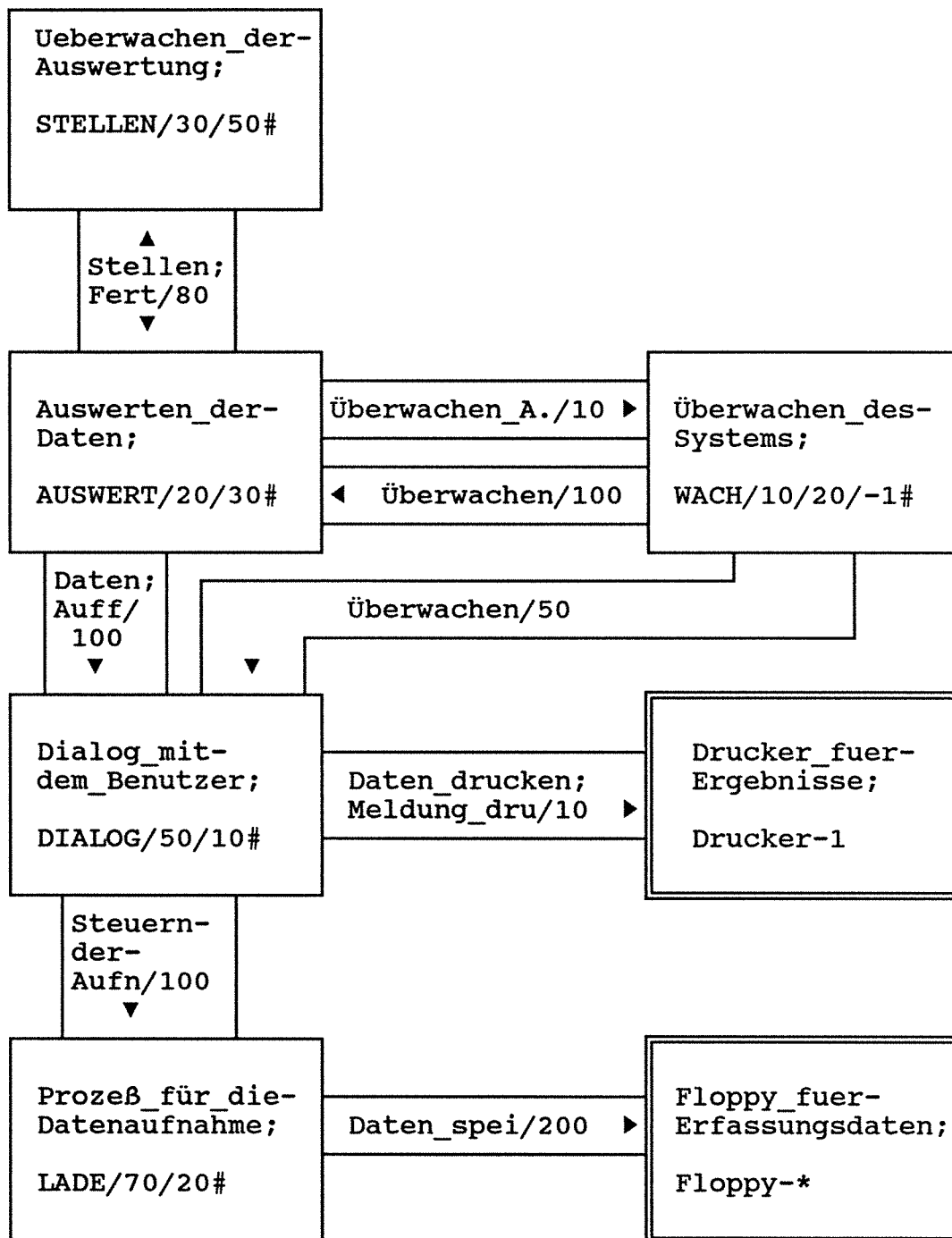


Abb.: I.2. Kommunikationsdiagramm mit Allokationsangaben

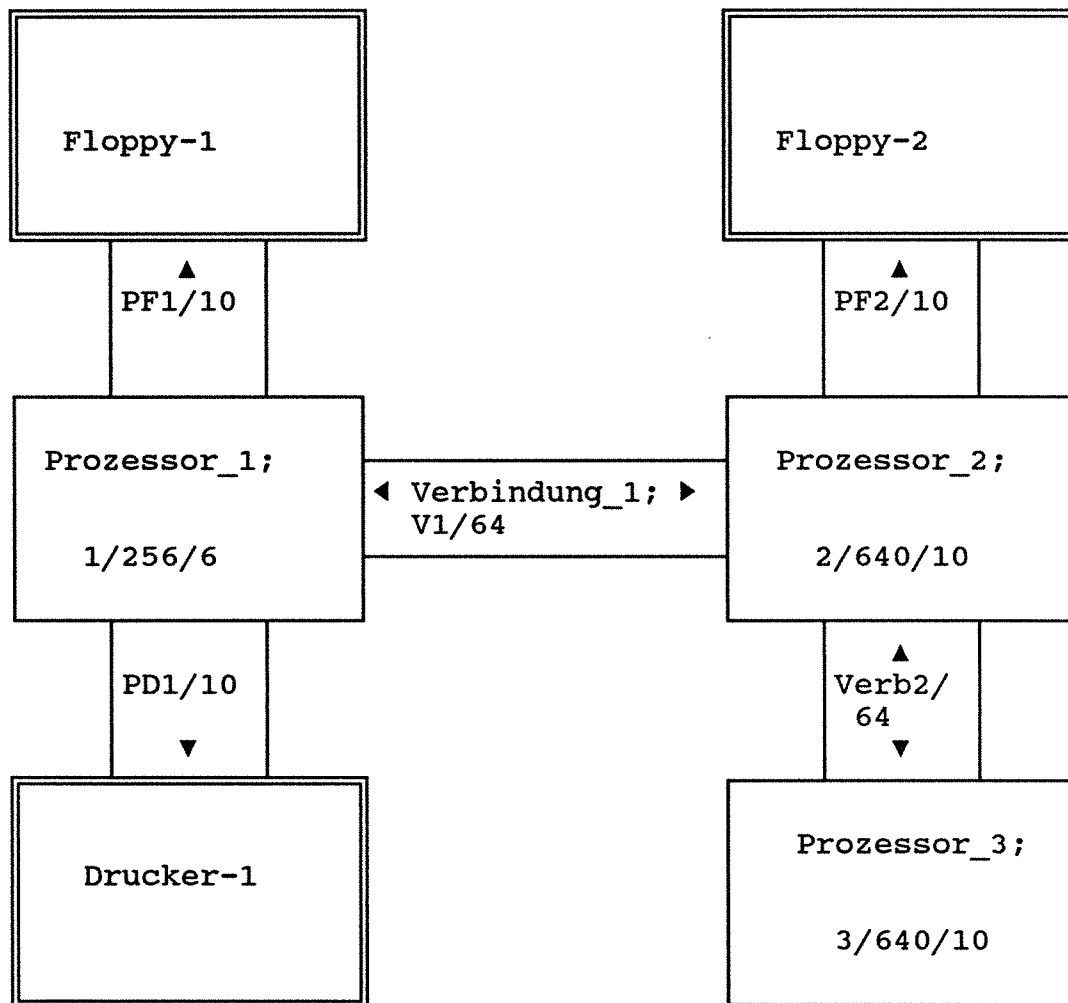


Abb. : I.3. Hardwarediagramm

## **B. Bewertung der Ergebnisse**

Mit dem Beispiel (Abbildung I.1. - I.3.) werden die Arbeitsweise des Allokationssystems verdeutlicht und die Auswirkungen der unterschiedlichen Optimierungskriterien vorgestellt.

### Optimierungskriterien

Die Kriterien sind durchnummeriert. Die Buchstaben A und B geben an, ob der Benutzer die Auslastung aller Prozessoren verlangt.

- 1 Optimieren nach Kommunikationskosten.
  - 2 Optimieren nach Laufzeitauslastung.
  - 3 Optimieren nach Kommunikationskosten und Laufzeitauslastung.
  - 4 Optimieren nach Kommunikationskosten und Speicherplatz.
- 
- A Prozessoren unbedingt alle auslasten.
- B Prozessoren nicht unbedingt alle auslasten.

Beim Netzwerk wird zwischen der Betrachtung der direkten Verbindungen (wie sie im Hardwarediagramm dargestellt sind) und vollständig vermaschten Netzen (lokale Netze) unterschieden.

### Einschränkungen

Für das Beispiel ergeben sich aus der Spezifikation folgende allgemeine Einschränkungen:

- Für die Task DIALOG (DIALOG) wird ein Drucker als peripheres Gerät benötigt. Der Drucker ist mit Drucker-1 angegeben. Die Hardwareanforderung bedeutet, daß die Task DIALOG den Drucker 1 benutzen will. Somit besteht für diese Task eine feste Zuweisung zum Prozessor 1.



- Für die Task Laden (LADE) wird ein Floppyzugang gefordert. Die Floppy, die benutzt werden soll, ist aber nicht festgelegt (Floppy-\*). Da im Hardwarediagramm die Prozessoren 1 (Floppy-1) und 2 (Floppy-2) jeweils eine Floppy besitzen, darf die Task nur diesen beiden Prozessoren zugewiesen werden.
- Für die Tasks AUSWERT und STELLEN entstehen aufgrund der Hardwareanforderungen keine Einschränkungen. Diese Tasks können - je nach Benutzerangabe - vom Allokationssystem aufgrund der Kommunikationskosten, der Laufzeitkosten oder des Speicherbedarfes zugeordnet werden.
- Die Task WACH wird durch ein Zuordnungsverbot zu Prozessor 1 in ihrer Zuordnungsfreiheit eingeschränkt. Für die Zuordnung an die beiden anderen Prozessoren gibt es keine Einschränkungen. Deshalb kann diese Task vom Allokationssystem aufgrund der Kommunikationskosten, der Laufzeitkosten oder des Speicherbedarfes an die Prozessoren 2 oder 3 zugeordnet werden.
- Aufgrund der Verbindungsstruktur der Hardware ergeben sich weitere Einschränkungen. Für je zwei Tasks, die miteinander kommunizieren, darf kein Fall auftreten, in dem eine der Tasks auf Prozessor 1 und gleichzeitig eine andere Task auf Prozessor 3 zu liegen kommt. Da die beiden Prozessoren keine Verbindung untereinander haben, wäre keine Kommunikation zwischen der Task auf Prozessor 1 und der Task auf Prozessor 3 möglich.
- Das vorliegende Beispiel erfordert vom Allokationssystem keine Entscheidung darüber, ob ein Prozessor überlastet wird.

Ergebnisse:

Als erstes werden die Ergebnisse des Allokationssystems die mit dem Netzwerktyp "Direkte Verbindungen" errechnet wurden:

Direkte Verbindungen

TASK      DIALOG      LADE      AUSWERT      WACH      STELLEN

1A	1	1	2	2	3
2A	1	1	2	2	3
3A	1	1	2	2	3
4A	1	1	2	2	3
1B	1	1	2	2	2
2B	1	1	2	2	3
3B	1	1	2	2	3
4B	1	1	2	2	3

Die Tatsache, daß die Prozessoren 1 und 3 keine Verbindung haben, führt zu einer Verteilung der Tasks, die für fast alle Optimierungskriterien gleich ist.

Für den Fall "Optimieren nach Kommunikationskosten" und "Nicht alle Prozessoren müssen ausgelastet werden" wird eine abweichende Lösung erreicht. Vom Allokationssystem wird dann der Prozessor 3 frei gelassen. Dadurch können niedrigere Kommunikationskosten erreicht werden. Für die drei verbleibenden Optimierungskriterien mit Erlaubnis evtl. Prozessoren unbelastet zu lassen, wird die gleiche Verteilung erreicht wie bei der Möglichkeit "alle Prozessoren müssen ausgelastet sein". Dies ist darauf zurückzuführen, daß in diesen Fällen das Verhältnis von Kommunikationskosten zu Laufzeitverhalten oder Speicherplatzauslastung besser ist, wenn alle Prozessoren ausgelastet sind.

### Task-Task-Zuordnungsverbot

Eine weitere Möglichkeit, die das Allokationssystem bietet, sind die Tasks-Task-Zuordnungs-Verbote und die -Gebote. Dazu wurde das Beispiel um folgende Zuordnungsverbote erweitert:

DIALOG - AUSWERT

DIALOG - LADE

Dies bedeutet also, daß die Task DIALOG nicht mit den Tasks AUSWERT und LADE auf einem gemeinsamen Prozessor zu liegen kommen darf.

Mit diesen Einschränkungen ergibt sich beispielsweise für Optimierungskriterium 1A folgende Verteilung:

TASK      DIALOG      LADE              AUSWERT      WACH      STELLEN

1A	1	2	2	2	3
----	---	---	---	---	---

Im Gegensatz zur Verteilung ohne das Task-Task-Zuordnungsverbot wird nun die Task LADE nicht mehr auf den Prozessor 1, sondern auf den Prozessor 2 gelegt. Weitere Veränderungen können hier nicht mehr erreicht werden, da die übrigen Tasks nicht mehr verschoben werden können.

### Task-Task-Zuordnungsgebot

Ähnlich wie ein Tasks-Task-Verbot kann auch ein Gebot die Verteilung entsprechend beeinflussen. Folgendes Task-Task-Gebot wird gemacht:

WACH - DIALOG

Das heißt, daß die Task WACH muß zusammen mit der Task DIALOG gemeinsam auf einem Prozessor liegen muß.

Diese Zuweisung ist jedoch nicht erlaubt, da die Task DIALOG aufgrund der Druckeranforderung auf Prozessor 1 zu liegen kommen muß. Während die Task WACH ein Verbot für diesen Prozessor besitzt.

Daran wird ersichtlich, daß das Allokationssystem das Allokationsgebot höher bewertet als die Einschränkungen der einzelnen Tasks. Die Task WACH und DIALOG werden in ein Cluster zusammengefasst und dem Prozessor 1 zugewiesen. Die Hardwarezuteilung erhält Priorität vor der festen Benutzerzuordnung. Anders als bei den in Kapitel 8 beschriebenen Konsistenztests wird in diesem Fall keine Hilfe vom Benutzer angefordert. Die Verteilung wird aufgrund der vorrangigen Behandlung des Allokationsgebotes vorgenommen.

Wegen der hohen Kommunikationskosten wird die Task AUSWERT ebenfalls auf den Prozessor 1 gezogen. Die Task LADE wird auf Prozessor 2 gelegt. Dies bedeutet somit, daß die Task STELLEN ebenfalls auf Prozessor 1 zu liegen kommen muß, weil sie mit AUSWERT und WACH kommuniziert. Diese Tasks liegen auf Prozessor 1, der mit Prozessor 3 keine Verbindung hat. Deshalb bleibt der Prozessor 3 in diesem Beispiel ungenutzt.

Das Ergebnis lautet in diesem Fall:

TASK	DIALOG	LADE	AUSWERT	WACH	STELLEN
1A	1	2	1	1	1

Die Verbindung zwischen den Prozessoren kann also die Verteilung entscheidend beeinflussen.

Das gleiche Beispiel wird nun ausgewertet, unter zuhelfenahme eines vollständig vermaschten Netzes (lokales Netz), dem ein Routingverfahren zugrundeliegt.

Vollständig vermaschtes Netz

Die Einschränkungen, die dadurch entstanden sind, daß bei den direkten Verbindungen nicht alle Prozessoren miteinander kommunizieren konnten, entfallen bei der Verteilung der Tasks auf ein Hardwaresystem, dem ein Routingverfahren zugrundeliegt. Die Ergebnisse, die mit dem Beispiel erzielt wurden sind bei vollständig vermaschten Netzen folgende:

TASK      DIALOG          LADE          AUSWERT      WACH          STELLEN

1A	1	2	1	1	3
2A	1	1	2	2	3
3A	1	1	2	2	3
4A	1	2	3	3	3
1B	1	1	3	3	3
2B	1	1	2	2	3
3B	1	1	2	2	3
4B	1	2	3	3	3

Da alle Prozessoren mit einander verbunden sind, wird ein Optimum in Bezug auf die Kommunikationskosten gefunden. Für da Optimierungskriterium 1A sind die niedrigsten Kosten erreicht. Diese Verteilung ist nun möglich, da auch die Prozessoren 1 und 3 verbunden sind.

Task-Task-Zuordnungsverbot

Mit den oben angegebenen Task-Task-Verboten für die Task DIALOG ergibt sich für Optimierungskriterium 1A folgende Verteilung:

TASK      DIALOG          LADE          AUSWERT      WACH          STELLEN

1A	1	2	3	3	3
----	---	---	---	---	---

Task-Task-Zuordnungsgebot

Für den Fall, daß ein Task-Task-Gebot zwischen DIALOG und WACH besteht, liefert das Allokationssystem folgendes Ergebnis:

TASK	DIALOG	LADE	AUSWERT	WACH	STELLEN
1A	1	2	1	1	3

Da (im Gegensatz zur Lösung ohne Routingverfahren) die Prozessoren 1 und 3 miteinander kommunizieren können, wird die Task STELLEN auf den Prozessor 3 gelegt.

Speicherplatz

Der Speicherplatz spielte bisher keine Rolle, da jeder der Prozessoren alle Tasks hätte aufnehmen können. Für den praktischen Einsatz eines Allokationssystems ist aber die Größe der Speicher von Bedeutung. Wenn der Speicherplatz eines Prozessors nicht ausreicht, muß eine Task eventuell (trotz höherer Kommunikationskosten) auf einen anderen Prozessor gelegt werden. Im vorliegenden Beispiel wird der Speicherplatz des Prozessors 2 auf 64 (Kbytes) gesetzt. Das Ergebnis der Verteilung mit Optimierungskriterium 1A und vollständig vermaschten Netz ist dann:

TASK	DIALOG	LADE	AUSWERT	WACH	STELLEN
1A	1	1	3 <sup>*</sup>	3	2

Optimalen Kommunikationskosten können hier nicht mehr erreicht werden: Die Task LADE kann in diesem Beispiel aufgrund ihres hohen (70 Kbytes) Speicherplatzbedarfes, nur noch auf den Prozessor 1 gelegt werden. Die Task DIALOG ist ebenfalls auf diesen Prozessor beschränkt.

### Zusammenfassung

Mit einem vollständig vermaschten Netz können Aussagen über das Kommunikationsverhalten der verteilten Tasks gemacht werden, ohne daß die Netzwerktopologie darauf einen Einfluß hat. Für praktische Probleme aus der Realzeitprogrammierung (z.B. Beispiel aus /HEIL85/ in Kapitel 10) ist es aber durchaus sinnvoll, direkte Verbindungen zu berücksichtigen.

Ebenso wie die Netzwerktopologie, die Hardware- und Benutzereinschränkungen und die Laufzeitgrößen, wirkt auch die Größe des Speichers einschränkend auf die Verteilung.

Mit dem gewählten Beispiel sollte gezeigt werden, wie eine Eingabe an das Allokationssystem aussieht.

Darüberhinaus wurden die unterschiedlichen Ergebnisse deutlich, die mit den verschiedenen Optimierungskriterien und Einschränkungen vom Allokationssystem errechnet werden.

Mit Hilfe der verschiedenen Optimierungskriterien können verschiedene Vorschläge erarbeitet werden. Aufgrund der verschiedenen Optimierungsmöglichkeiten und der verschiedenen Netzwerktopologien kann sich der Benutzer einen Überblick über die Lösung seiner Aufgabe verschaffen.

## Anhang II

### II. Syntaxdefinition

Ein Teil der Information über eine Verteilung kann aus der Topologie der Kommunikationsstruktur entnommen werden. Darüberhinaus muß dem Allokationssystem weitere Information über die Verteilung (z.B. Größe der Tasks, Kommunikationskosten, Peripherieanforderungen einer Task...) bekannt sein. Dazu wird die Kommunikationsstruktur um die Angaben für die Verteilung der Tasks erweitert. Die Definition der Syntax der Eingaben für ALLOC erfolgt in einer erweiterten Backus-Naur-Form.

- | Alternative
- [] Option
- { } Iteration (auch nullmalige Wiederholung)

#### A. Zulässige Einheiten

Speichereinheit	::=	Byte   Kbyte   Mbyte
Leistungseinheit	::=	IPS   KIPS   MIPS
Verbindungseinheit	::=	Byte/s   Kbyte/s   Mbyte/s

#### B. Beschreibung der Syntax des Kommunikationsdiagramms

##### Beschreibung der Tasks:

Taskbeschreibung	::=	Taskname '/' Speicherbedarf '/' Laufzeitbedarf '/' Prozessorbeziehungen '#'
Taskname	::=	Symbol /* Maximal 8 Zeichen */
Speicherbedarf	::=	Integer [ ':' Speichereinheit]
Laufzeitbedarf	::=	Integer [ ':' Leistungseinheit]
Prozessorbeziehungen	::=	{ Prozessorverbot }   Prozessorgebot   €
Prozessorverbot	::=	'-' Integer
Prozessorgebot	::=	'+' Integer



Beschreibung der Peripherieanforderungen:

Für die Beschreibung der Peripherieanforderungen (Hardware) gilt folgende Syntax:

```
Hardwarebeschreibung  ::= Hardwarename '-' Hardwarezuordnung
Hardwarename          ::= Symbol                /* Maximal 8 Zeichen */
Hardwarezuordnung     ::= '*' | Integer          /* Maximal 2 Stellen */
```

Beschreibung der Verbindungen zwischen den Tasks und zwischen Tasks und Hardware

```
Kommunikationsbeschreibung ::= Botschaftsname '/' Kommunikationskosten
Botschaftsname              ::= Symbol
Kommunikationskosten        ::= Integer [ ':' Verbindungseinheit]
```

**C. Beschreibung der Syntax des Hardwarediagramms**

Zusätzlich zum Kommunikationsdiagramm wird für die Allokation ein Hardware-diagramm benötigt, in dem folgende Information enthalten ist:

Beschreibung der Prozessoren:

```
Prozessorbeschreibung  ::= Prozessornummer '/' Prozessorgröße '/'
                           Leistung '/' Prozessortyp
Prozessornummer        ::= Integer
Prozessorgröße         ::= Integer [ ':' Speichereinheit]
Leistung               ::= Integer [ ':' Leistungseinheit]
Prozessortyp           ::= Integer
```

Beschreibung der Peripherie:

Für die Peripheriebeschreibung ist folgende Knotenbeschriftung erlaubt.

Peripheriebeschreibung	::=	Peripheriename '-' Zuordner_zu_Prozessor
Peripheriename	::=	Symbol /* Maximal 8 Zeichen */
Zuordner_zu_Prozessor	::=	Integer /* Maximal 2 Stellen */

Beschreibung der Verbindungen zwischen den Prozessoren und zwischen Prozessoren und der Peripherie

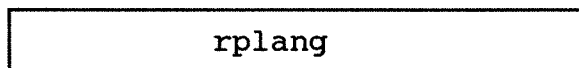
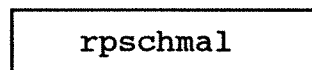
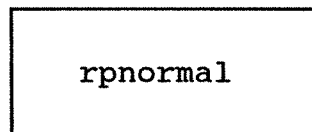
Netzwerkbeschreibung	::=	Verbindungsname '/' Kapazität
Verbindungsname	::=	Symbol [ ':' Verbindungseinheit]
Kapazität	::=	Integer

## Anhang III

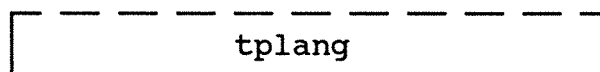
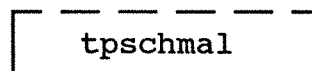
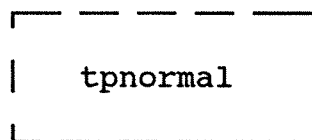
### III. Definition der Symbole für die Eingabe

Die Eingaben (Kommunikationsstruktur, Hardwarestruktur) an das System ALLOC werden mit einem graphischen Editor (ZP der Firma S.E.P.P. /SEPP84/) erstellt. Die verwendeten Symbole sind vordefiniert; sie können aber beliebig verkleinert oder vergrößert werden, und um 90 Grad gedreht werden. Die Verbindungskanten wurden in /BAUE89/ ebenfalls vordefiniert. Falls neue Kantentypen benötigt werden, müssen diese als Polygonzüge gezeichnet werden.

Für die Tasks und die Prozessoren sind folgende Symbole erlaubt:



Für die technischen Prozesse (bzw. die peripheren Geräte) gelten folgende Symbole:



## Anhang IV

### IV. Regeln des Systems ALLOC

Das Allokationssystem ALLOC wurde in der Sprache OPS83 /FORG86/ implementiert. Da die Angabe der Regeln in OPS83 nicht übersichtlich ist, und meist mehrere "OPS-Regeln" notwendig sind um die Funktionalität einer, vom Experten formulierten Teilaussage, im Rechner nachzubilden, werden in diesem Teil des Anhangs die Regeln in einer umgangssprachlichen Form dargelegt. Die Regeln im System ALLOC werden in der vorliegenden Darstellung zu logischen Einheiten zusammengefasst, und die Verwaltungsregeln (siehe Kapitel 8) werden nicht aufgeführt.

Die Reihenfolge der Auflistung der Regeln folgt der in Kapitel 8 angegebenen Phaseneinteilung:

- A) Auswerten der festen Randbedingungen
- B) Auswerten der Zuordnungseinschränkungen
- C) Einteilen der Tasks in Cluster
- D) Verteilen der Cluster auf die Prozessoren
- E) Verschieben einzelner Tasks
- F) Verschieben ganzer Cluster

Die Regeln werden folgendermaßen dargestellt:

Beschreibung der Semantik der Regel (Entsprechende Regelnummern in OPS z.B. R4/R5).

<i>Wenn</i>	(1)	Prädikat 1
	& (2)	Prädikat 2
	.	.
	.	.
	& (n)	Prädikat n

*dann*                      Ausführungsteil der Regel mit Angabe der einzelnen Schritte.

*Priorität der Regel : n*

**A. Auswerten der festen Randbedingungen****Regelmenge I**

Diese Regelmenge sorgt für die Zuteilung einer Task zu einem Prozessor, für den sie eine feste Zuweisung besitzt. Eine Task kann einem Prozessor entweder durch den Benutzer fest zugewiesen werden, oder aufgrund von Hardwareanforderungen.

Priorität der Regelmenge: 1

Feste Task-Prozessor-Zuordnung, wobei aber (mindestens) eine Hardwareanforderung nicht befriedigt werden kann (R4/R5).

Wenn (1) Für eine Task A besteht eine feste Zuordnung zu einem Prozessor Y aufgrund eines benutzerdefinierten Task-Prozessor-Zuordnungsgebotes.

& (2) Die Task A besitzt Hardwareforderungen ( $X_1, \dots, X_n$ ), die an dem betreffenden Prozessor Y nicht zur Verfügung stehen.

dann Der Benutzer wird über die Inkonsistenz informiert. Er muß nun entscheiden, ob die Task A trotz fehlender Hardware ( $X_1, \dots, X_n$ ) zugeteilt werden soll. Falls dies geschehen soll, werden folgende Punkte erledigt:

Die Task A wird dem entsprechenden Prozessor Y zugeordnet. Die Prozessorauslastung und die Laufzeitauslastung für diesen Prozessor werden neu berechnet.

Falls der Prozessor überlastet ist, erfolgt eine Meldung an den Benutzer. Der Speicher kann daraufhin vom Benutzer vergrößert werden. Anderenfalls erfolgt keine Zuordnung der Task A.

Priorität der Regel: 1

Zuteilung einer Task mit fester benutzerdefinierter Zuordnung (R2/R3).

Wenn (1) Für eine Task A besteht eine feste Zuordnung zu einem Prozessor Y aufgrund einer benutzerdefinierten Task-Prozessor-Zuordnung.

& (2) Die Task A besitzt kein Task-Prozessorverbot für den Prozessor Y.

dann Die Task A wird dem Prozessor Y fest zugeordnet. Die Prozessorauslastung und die Laufzeitauslastung für diesen Prozessor werden neu berechnet.

Falls der Prozessor überlastet ist, erfolgt eine Meldung an den Benutzer. Der Speicher kann daraufhin vom Benutzer vergrößert werden. Anderenfalls erfolgt keine Zuordnung der Task A.

Priorität der Regel: 2

Zuordnung einer Task aufgrund von Hardware-Forderungen (R10 bis R13).

Wenn (1) Für eine Task A besteht eine feste Zuordnung zu einem Prozessor Y aufgrund von Hardwareforderungen ( $X_1, \dots, X_n$ ).

& (2) Die Task A besitzt kein Task-Prozessorverbot für den Prozessor Y.

dann Die Task A wird dem Prozessor Y fest zugeordnet. Die Prozessorauslastung und die Laufzeitauslastung für diesen Prozessor werden neu berechnet.

Falls der Prozessor überlastet ist, erfolgt eine Meldung an den Benutzer. Der Speicher kann daraufhin vom Benutzer vergrößert werden. Anderenfalls erfolgt keine Zuordnung der Task A.

Priorität der Regel: 2

**Regelmenge II**

Mit dieser Regelmenge werden mögliche Inkonsistenzen aufgespürt und an den Benutzer gemeldet. Entsprechend der zusätzlichen Information vom Benutzer kann das System ALLOC dann eine Entscheidung treffen.

Priorität der Regelmenge: 2

Task soll trotz bestehenden Inkonsistenz bei der Hardwareforderung zugeteilt werden (R8).

Wenn (1) Für eine Task A besteht eine feste Zuordnung zu einem Prozessor Y aufgrund von Hardwareforderungen ( $X_1, \dots, X_n$ ).

& (2) Die Task A besitzt Hardwareforderungen ( $X_{n+1}, \dots, X_m$ ), die an dem Prozessor Y nicht zur Verfügung stehen.

dann Der Benutzer muß entscheiden, ob die Task dennoch zugeteilt werden soll. Falls dies geschehen soll, werden folgende Punkte erledigt:

- Die Task wird dem entsprechenden Prozessor zugeordnet.
- Die Prozessorauslastung und die Laufzeitauslastung für diesen Prozessor wird neu berechnet.
- Falls der Prozessor überlastet ist, erfolgt eine Meldung an den Benutzer mit Korrekturmöglichkeiten (wie in Regelmenge I beschrieben).

Priorität der Regel: 1

---

Eine Task soll aufgrund der Hardwareforderungen einem Prozessor zugewiesen werden. Gleichzeitig besteht aber ein benutzerdefiniertes Task-Prozessor-Zuordnungsverbot (R30/R31).

Wenn      (1)      Für eine Task A besteht eine feste Zuordnung zu einem Prozessor Y aufgrund von Hardwareforderungen ( $X_1, \dots, X_n$ ).

            & (2)      Die Task A besitzt ein (benutzerdefiniertes) Zuordnungsverbot für den Prozessor Y.

dann              Der Benutzer muß entscheiden, ob die Task trotz des Zuordnungsverbotes zugeteilt werden soll, oder ob die Hardwareforderungen ( $X_1, \dots, X_n$ ) aufgelöst werden sollen.

Fall I (Aufhebung des Verbotes):

- Die Task wird (falls gewünscht) dem Prozessor Y zugeordnet.
- Die Berechnung der Prozessor- und der Laufzeitauslastung (bzw. Speichervergrößerung falls notwendig) wird vorgenommen.

Fall II (Beibehaltung des Verbotes):

- Für den Fall, daß das Verbot nicht aufgehoben wird, muß die Task A ohne die Hardwareforderungen ( $X_1, \dots, X_n$ ) in einer späteren Phase (einem Prozessor  $\neq Y$ ) zugeteilt werden.

Priorität der Regel:              1

Eine Task soll einem Prozessor zugewiesen werden, aber die Hardware fehlt an dem betrachteten Prozessor (R40).

Wenn      (1)      Für eine Task A besteht eine feste Zuordnung zu einem Prozessor Y.

            & (2)      Die Task A fordert Hardware ( $X_1, \dots, X_n$ ), die am Prozessor Y nicht zur Verfügung steht.



*dann* Der Benutzer muß entscheiden, ob die Task A trotz der fehlenden Hardware zugeteilt werden soll, oder ob die feste Zuweisung an den Prozessor Y aufgelöst werden soll.

Fall I (Beibehaltung des Gebotes):

- Die Task A wird dem Prozessor Y zugeordnet.
- Die Berechnung der Prozessor- und der Laufzeitauslastung (bzw. Speichervergrößerung falls notwendig) wird wie in der Regel R8 vorgenommen..

Fall II (Aufhebung des Gebotes):

- Für den Fall, daß das Gebot aufgehoben wird, wird die Task A aufgrund ihrer Hardwareforderungen ( $X_1, \dots, X_n$ ) (einem Prozessor  $\leftrightarrow Y$ ) zugeteilt werden.

*Priorität der Regel:* 2

Eine Task soll einem Prozessor zugewiesen werden. Es besteht aber ein Allokationsverbot mit einer bereits (dem betrachteten Prozessor) zugeordneten Task (R20/21).

*Wenn* (1) Für eine Task A besteht eine feste Zuordnung zu einem Prozessor Y durch den Benutzer.  
 & (2) Es existiert ein Allokationsverbot mit einer bereits dem Prozessor Y zugeordneten Task B.

*dann* Der Benutzer muß entscheiden, ob die Task A trotz des Allokationsverbotes zugeteilt werden soll, oder ob die feste Zuweisung an den Prozessor Y aufgelöst werden soll.

Fall I (Beibehaltung des Zuordnungsgebotes):

- Die Task A wird dem Prozessor Y trotz bestehendem Allokationsverbotes zugeordnet. Die Berechnung der Prozessor- und der Laufzeitauslastung (bzw. Speichervergrößerung falls notwendig) wird wie in der Regel R8 vorgenommen.

## Fall II (Aufhebung des Gebotes):

- Die Task A wird in einer späteren Phase (einem Prozessor  $\leftrightarrow Y$ ) zugeteilt.

Priorität der Regel: 2

Zwei Tasks besitzen ein Allokationsgebot. Aufgrund ihrer Hardwareforderungen sollen sie aber zwei verschiedenen Prozessoren zugeordnet werden (R60/R61).

- Wenn (1) Zwei Tasks (A, B) besitzen ein Allokationsgebot (d.h. sie sollen gemeinsam einem Prozessor  $Y_k$  zugewiesen werden).
- & (2) Die Task sollen aufgrund ihrer Hardwareforderungen  $((X_1, \dots, X_n); (X'_1, \dots, X'_n))$  getrennten Prozessoren  $(Y_i, Y_j)$   $(i, j \neq k)$  zugewiesen werden.

dann Der Benutzer muß entscheiden, ob die Tasks A, B trotz des Allokationsgebotes verschiedenen Prozessoren zugeteilt werden, oder das Gebot höher bewertet werden soll.

## Fall I (Beibehaltung des Allokationsgebotes):

- Die Tasks A und B werden dem Prozessor  $Y_k$  trotz bestehender Hardwareforderungen zugeordnet.

## Fall II (Aufhebung des Gebotes):

- Die Tasks A und B werden aufgrund ihrer Hardwareforderungen den Prozessoren  $Y_i$  bzw.  $Y_j$  zugeteilt.
- Die Berechnung der Prozessor- und der Laufzeitauslastung (bzw. Speichervergrößerung falls notwendig) wird wie in der Regel R8 vorgenommen.

Priorität der Regel: 2

Eine Task wurde zugewiesen und überlastet einen Prozessor (R45).

*Wenn*        (1)        Der Speicherplatz eines Prozessors Y ist überbelegt.

*dann*                Die Größe des Speichers kann vom Benutzer erhöht werden.  
Anderenfalls kann eine Task vom entsprechenden Prozessor  
entfernt werden.

*Priorität der Regel:*                3

Inkonsistenz bei der Namensgebung (R25).

*Wenn*        (1)        Die Namensgebung einer Task stimmt im Hardware- und Kom-  
munikationsdiagramm nicht überein.

*dann*                Der Benutzer wird von dieser Inkonsistenz in Kenntnis  
gesetzt und erhält die Möglichkeit die Namensgebung zu  
korrigieren.

*Priorität der Regel:*                4

### Regelmenge III

Regelmenge zur Berechnung der Fixkosten. Dies sind Kosten, die aufgrund fester Zuweisungen von Tasks an Prozessoren entstehen. Zur besseren Übersicht für einen Benutzer des Allokationssystems werden die Fixkosten berechnet. Sie dienen zur Bewertung der IPC-Kosten.

Priorität der Regelmenge: 2

#### Intern-Fixkosten für fest zugeteilte Tasks berechnen (R70).

*Wenn*      (1)      Eine Task A ist bereits einem Prozessor Y fest zugeordnet.  
             & (2)      Es existiert eine Task B, die ebenfalls dem Prozessor Y  
                         fest zugeordnet ist.  
             & (2)      Die beiden Tasks kommunizieren miteinander.

*dann*                      Die Intern-Fixkosten werden um den Anteil der Kommuni-  
                                 kationskosten zwischen Task A und Task B erhöht.

*Priorität der Regel:*                      1

#### Verbindungs-Fixkosten für fest zugeteilte Tasks berechnen (R71).

*Wenn*      (1)      Eine Task A ist bereits einem Prozessor  $Y_i$  fest zugeordnet.  
             & (2)      Es existiert eine Task B, die einem Prozessor  $Y_j$  fest  
                         zugeordnet ist mit ( $i \neq j$ ).  
             & (2)      Die beiden Tasks kommunizieren miteinander.

*dann*                      Die Verbindungs-Fixkosten werden um den Anteil der  
                                 Kommunikationskosten zwischen Task A und Task B erhöht.

*Priorität der Regel:*                      1

**B. Auswerten der Zuordnungseinschränkungen****Regelmenge I**

Durch Einschränkungen kann die Zuteilungsfreiheit einer Task auf eine bestimmte Gruppe von Prozessoren (z.B. alle Prozessoren, die einen Druckerzugang besitzen) reduziert werden. Diese Hardwareforderung, die von einer ganzen Gruppe von Prozessoren erfüllt werden kann, wird im System ALLOC \*-Hardware genannt. Mit der Regelmenge I wird eine Task ausgesucht, die Zuordnungseinschränkungen besitzt. Für diese Task wird überprüft, ob ein Prozessor ihre Hardwarewünsche befriedigen kann.

Priorität der Regelmenge: 1

Auswahl einer Task zur Zuordnung (R2)

Wenn (1) Eine Task A besitzt \*-Hardwareanforderungen ( $X_1, \dots, X_n$ ).  
& (2) Es existiert ein Prozessor Y, an dem die geforderte Hardware ( $X_1, \dots, X_n$ ) zur Verfügung steht.

dann Diese Task-Prozessor-Kombination (A-Y) wird daraufhin überprüft, ob die Zuteilung der Task A an den Prozessor Y gewährt werden kann.

Priorität der Regel: 1

Ausschluß einer Task von der Zuteilung (R9 und R 10)

Wenn (1) Eine Task A wurde auf ihre Tauglichkeit auf Zuteilung zu einem Prozessor Y überprüft.

dann Diese Task-Prozessor-Kombination wird (in dieser Regelmenge) von der weiteren Zuteilungsüberprüfung ausgeschlossen.

Priorität der Regel: 2

### Regelmenge II

Mit der Regelmenge II wird überprüft, ob eine vorgeschlagene Zuordnung befriedigt werden kann. Die entsprechende Task-Prozessor-Kombination wird daraufhin überprüft, ob alle Anforderungen der Task vom betrachteten Prozessor befriedigt werden können.

Priorität der Regelmenge: 1

#### Ausschluß einer Task von der Zuordnung wegen Allokationsgebotes (R3/R4)

Wenn (1) Task A soll Prozessor Y zugewiesen werden.  
& (2) Auf dem Prozessor Y liegt eine Task B mit der die Task A ein Allokationsverbot hat.

dann Diese Task-Prozessor-Kombination wird von der Zuordnung ausgeschlossen.

Priorität der Regel: 1

#### Task darf zugeordnet werden (R5/R6)

Wenn (1) Task A soll Prozessor Y zugewiesen werden.  
& (2) Alle weiteren Wünsche (Hardwareanforderungen...) können ebenfalls erfüllt werden.

dann Diese Task-Prozessor-Kombination kann als erlaubt vermerkt werden.

Priorität der Regel: 4

Ausschluß einer Task von der Zuordnung wegen unerfüllbarer Hardwareforderungen (R5/R7)

Wenn      (1)      Task A soll Prozessor Y zugewiesen werden.  
            & (2)      Es existiert mindestens eine weitere Hardwareforderung, die nicht erfüllt werden kann.

dann              Diese Task-Prozessor-Kombination wird von der Zuordnung ausgeschlossen.

Priorität der Regel:              2

Ausschluß einer Task von der Zuordnung wegen Zuordnungsverbotes (R5/R8)

Wenn      (1)      Task A soll Prozessor Y zugewiesen werden.  
            & (2)      Die Task A hat für den Prozessor Y ein Zuordnungsverbot.

dann              Diese Task-Prozessor-Kombination wird von der Zuordnung ausgeschlossen.

Priorität der Regel:              3

Eine Task kann bereits zugewiesen werden (R11/R12)

Wenn      (1)      Task A wurde überprüft und kann aufgrund dieser Tests dem Prozessor Y zugewiesen werden.

dann              Ausführen der entsprechenden Verwaltungsoperationen.

Priorität der Regel:              4

### Regelmenge III

Mit dieser Regelmenge werden die Kombinationen für jede Task daraufhin überprüft, ob eine Task bereits in dieser Phase einem Prozessor fest zugeteilt werden kann. Anderenfalls werden die entsprechenden Zuteilungsmöglichkeiten vermerkt und dienen in späteren Phasen für die Entscheidungsfindung.

Priorität der Regelmenge: 3

#### Ausschluß einer Task, für die ein Allokationsverbot besteht (R56/R 57)

Wenn (1) Für eine Task A existiert keine Zuordnungsmöglichkeit<sup>2</sup>.  
& (2) Eine der Task-Prozessor-Kombinationen kann aufgrund eines Task-Task-Allokationsverbotes nicht erfüllt werden.

dann Mit einer Meldung an den Benutzer wird dieser um Stellungnahme gebeten:

- Falls das Allokationsverbot (vom Benutzer) aufgehoben wird, werden die Zuteilungsmöglichkeiten neu überprüft.
- Anderenfalls wird der Benutzer davon in Kenntniss gesetzt, daß diese Task nicht mehr berücksichtigt werden kann.

Priorität der Regel: 1

---

<sup>2</sup> Keine Zuordnungsmöglichkeit bedeutet: keine der betrachteten Task-Prozessor-Kombinationsmöglichkeiten ist zulässig.



Ausschluß einer Task, für die ein Allokationsverbot besteht (R54/R55)

*Wenn*      (1)      Für eine Task A existiert keine Zuordnungsmöglichkeit.  
             & (2)      Eine der Task-Prozessor-Kombinationen kann aufgrund  
                         fehlender Hardware nicht erfüllt werden.

*dann*                      Mit einer Meldung an den Benutzer wird dieser um Stellung-  
                                 nahme gebeten:

- Falls die Task trotz fehlender Hardware zugeteilt werden soll, werden die Zuteilungsmöglichkeiten neu überprüft.
- Anderenfalls wird der Benutzer davon in Kenntniss gesetzt, daß diese Task nicht mehr berücksichtigt werden kann.

*Priorität der Regel:*                      1

Feste Zuteilung einer Task zu einem Prozessor (R50/R51)

*Wenn*      (1)      Für eine Task A existiert (trotz der Zuteilungsmöglichkeiten in Form der \*-Hardware) nur eine Task-Prozessor-Kombination, die ausführbar ist. (Alle anderen Kombinationen sind nicht ausführbar).

*dann*                      Die Task A wird dem Prozessor Y bereits in dieser Phase fest zugewiesen.

Abhängig davon, ob dem entsprechenden Prozessor schon eine Task zugewiesen wurde, werden zusätzliche Verwaltungsarbeiten notwendig.

*Priorität der Regel:*                      2

---

Eine Task kann trotz Hardwareanforderungen mehreren Prozessoren zugeordnet werden (R52)

*Wenn* (1) Für eine Task A existieren mehrere Zuordnungsmöglichkeiten (Prozessoren  $Y_1, \dots, Y_n$ ).

*dann* Die Kombinationen werden vermerkt und dienen in anderen Phasen der Entscheidungsfindung.

*Priorität der Regel:* 3

Eine Task kann aufgrund einer vorliegenden Inkonsistenz (z.B. in der Namensgebung) nicht zugeordnet werden (R53)

*Wenn* (1) Eine Task A kann keinem der vorhandenen Prozessoren zugewiesen werden (z.B. Inkonsistenz der Namensgebung in der Hardwareanforderung).

*dann* Mit einer Meldung an den Benutzer wird dieser um Stellungnahme gebeten:

- Falls die inkonsistente Situation dadurch aufgehoben wird, werden die Zuteilungsmöglichkeiten neu überprüft.
- Anderenfalls wird der Benutzer davon in Kenntnis gesetzt, daß diese Task nicht mehr für die Verteilung zur Verfügung steht.

*Priorität der Regel:* 4

Zuweisung einer Task aufgrund eines Allokationsgebotes (R60)

*Wenn*      (1)      Eine Task A kann keinem der vorhandenen Prozessoren zugewiesen werden (z.B. Inkonsistenz der Parameter).  
              & (2)      Es existiert eine Task B, die mit der Task A ein Allokationsgebot hat und einem Prozessor Y zugewiesen ist.

*dann*                Es muß überprüft werden, ob die Task B dem Prozessor Y ebenfalls zugewiesen werden kann.  
                         Falls dies nicht der Fall ist, muß die Zuteilung der Task A neu vorgenommen werden.

*Priorität der Regel:*                5

Die Regeln zur Bestimmung der Verbindungs- und der Internfixkosten sind denen in der Phase "Auswerten der festen Randbedingungen" ähnlich und werden deshalb nicht gesondert aufgeführt.

### C. Einteilen der Tasks in Cluster

#### Regelmenge I

Die Regeln dieser Regelmenge sollen die Tasks in Cluster einteilen, die vom Benutzer her schon eine feste Task-Prozessor Zuteilung erhalten haben.

Priorität der Regelmenge: 1

#### Einteilung von Tasks mit fester Task-Prozessor-Beziehung (R3).

*Wenn* (1) Es existiert eine Task A, die eine feste Zuweisung<sup>3</sup> zu einem Prozessor Y hat.

& (2) Es existiert eine weitere Task B, die ebenfalls eine feste Zuweisung zu dem Prozessor Y hat und bereits in einem Cluster C liegt.

*dann* Die Task A kommt in dasselbe Cluster, in dem die Task B bereits liegt.

*Priorität der Regel:* 1

#### Einteilung einer Task in ein "neues" Cluster (R2).

*Wenn* (1) Es existiert eine Task A, die eine feste Zuweisung zu einem Prozessor hat.

*dann* Die Task A wird in ein "neues" Cluster ( $C_{neu}$ )<sup>4</sup> gelegt.

*Priorität der Regel:* 2

---

<sup>3</sup> Eine feste Zuweisung kann eine Task-Prozessor-Zuweisung durch den Benutzer sein, oder eine Zuweisung, die sich aufgrund einer Task-Hardware-Beziehung ergibt.

<sup>4</sup>  $C_{neu}$  wird vom Allokationssystem bestimmt.

**Regelmenge II**

Die Regeln in dieser Menge wählen eine Task aus, die folgende Bedingungen erfüllt:

- a) Für die Task besteht keine feste Zuweisung.
- b) Die Task liegt noch in keinem Cluster.
- c) Die Task hat größte Priorität in Bezug auf
  - die Kommunikationskosten, oder
  - die Gesamtkommunikationskosten<sup>5</sup> aller noch nicht zugeteilter Tasks oder
  - die Anzahl ihrer Verbindungen zu anderen Tasks.

Priorität der Regelmenge: 2

Auswahl einer Task A mit den größten Gesamtkommunikationskosten und den größten Kommunikationskosten mit einer Nachbartask, die bereits in einem Cluster liegt (R10).

Wenn      (1)      Task A hat die höchsten Gesamtkommunikationskosten.  
            & (2)      Die Task A hat unter den betrachteten die größten Kommunikationskosten mit einer schon eingeteilten Partnertask B.

dann              Die Task A wird daraufhin überprüft, ob sie dem Cluster der Task B zugewiesen werden soll, oder ob ein neues Cluster zu erstellen ist. (Dazu werden die Regelmenge der Phase III benutzt.)

Priorität der Regel: 1

---

<sup>5</sup> Gesamtkommunikationskosten einer Task:  
Summe der Kosten, die diese Task mit allen ihren Nachbarn hat.

Auswahl einer Task A mit den größten Gesamtkommunikationskosten und gleichgroßen Kommunikationskosten mit zwei Nachbartasks, die bereits in Clustern liegen (R11).

- Wenn*
- (1) Betrachtet wird eine Task A, die die höchsten Gesamtkommunikationskosten hat.
  - & (2) Es existieren zwei andere Tasks B und C, die mit der Task A gleichgroße Kommunikationskosten haben.
  - & (3) Die Tasks B und C liegen bereits in Clustern  $C_1$  und  $C_2$ .
- dann* Nehme die Task (B oder C), die die größten Gesamtkommunikationskosten hat, und prüfe, ob die Task A einem der Cluster ( $C_1$  bzw.  $C_2$ ) zugewiesen werden soll, oder ob ein neues Cluster zu erstellen ist.

*Priorität der Regel:* 2

Auswahl einer Task A mit: den größten Gesamtkommunikationskosten und gleichgroßen Kommunikationskosten mit zwei Nachbartasks, die bereits in Clustern liegen und gleichgroße Gesamtkommunikationskosten haben (R12).

- Wenn*
- (1) Betrachtet wird eine Task A, die die höchsten Gesamtkommunikationskosten hat.
  - & (2) Es existieren zwei andere Tasks (B und C), die mit der Task A gleichgroße Kommunikationskosten haben
  - & (3) Die Tasks B und C haben gleichgroße Gesamtkommunikationskosten.
  - & (4) Die Tasks B und C liegen bereits in Clustern  $C_1$  und  $C_2$ .
- dann* Nehme die Task (B oder C), die die größte Verbindungsanzahl hat, und prüfe, ob die Task A dem Cluster dieser Task ( $C_1$  bzw.  $C_2$ ) zugewiesen werden soll, oder ob ein neues Cluster zu erstellen ist.

*Priorität der Regel:* 3

### Regelmenge III

In dieser Menge wird für eine vorher bestimmte Task die Entscheidung getroffen, ob für diese Task ein neues Cluster angelegt wird, oder ob die Task einem bereits bestehenden Cluster zugeteilt werden kann. Dazu stehen Regeln bereit, die eine Task nach bestimmten Einschränkungen der Zuteilung an ein Cluster untersuchen. Für den Fall, daß keine solche Einschränkung gefunden werden kann, werden Regeln aktiviert, die die Task aufgrund von Berechnungen der Metrik an ein Cluster zuteilen.

Priorität der Regelmenge: 3

Eine Task A hat mit einer Task B ein Allokationsgebot. Die Task B ist bereits einem Cluster zugewiesen. (R42/R43).

Wenn      (1)      Für die zur Einteilung ausgewählte Task A besteht ein Allokationsgebot, das vom Benutzer vorgegeben ist.  
            & (2)      Die Task B, mit der für die ausgewählte ein Allokationsgebot besteht, ist schon einem Cluster C zugeteilt.

dann              Die betrachtete Task A wird dem Cluster C zugewiesen, in der die "Partnertask" B schon liegt.

Priorität der Regel:              1

---

Zu einer Clustereinteilung zweier Tasks A und B existiert eine andere Kombination von Tasks (A und C) mit Verbesserung der Kosten (R15/R16).

*Wenn* (1) Die zur Einteilung ausgewählte Task A hat mit ihrer Partner-task B weniger Kommunikationskosten, als die Partnertask mit einer weiteren noch nicht an ein Cluster zugewiesenen Task C.

*dann* Die betrachtete Task A wird zurückgestellt. Zunächst wird die Task C auf ihre Zuteilungstauglichkeit überprüft.

*Priorität der Regel:* 1

Ein Cluster besitzt (aufgrund einer fest zugeteilten Task) eine feste Zuordnung zu Prozessor. Bei weiteren Zuweisungen von Tasks zu diesem Cluster muß deshalb die Speicherauslastung überprüft werden (R44).

*Wenn* (1) Eine zur Einteilung ausgewählte Task A soll in ein Cluster C gelegt werden, das schon einem Prozessor Y fest zugeordnet ist.

& (2) Das Cluster C enthält eine Task B, die einem Prozessor Y bereits fest zugeordnet ist.

*dann* Der Benutzer wird um Rat befragt, welcher Lösung er den Vorzug geben will:

A) Falls der betreffende Prozessor Y mit dieser zusätzlichen Task A (im Cluster C) überlastet wäre, darf die Task diesem Cluster nicht zugeteilt werden.

B) Die Task muß mit einer anderen "Partnertask" zusammengelegt werden. (Dazu muß ein neuer Partner für die betrachtete Task bestimmt werden.)

*Priorität der Regel:* 1



Eine Task soll einem Cluster zugewiesen werden, in dem eine Task liegt, mit der sie ein Allokationsverbot hat (R45/R46).

Wenn      (1)      Für die zur Einteilung ausgewählten Task A besteht ein Allokationsverbot.  
             & (2)      Die Partnertask B, mit der für die Task A ein Allokationsverbot besteht, ist schon einem Cluster C zugeteilt.

dann                      Die betrachtete Task A darf dem Cluster C nicht zugewiesen werden. (Die Task A muß entweder einem anderen -bereits existierenden- Cluster C' zugewiesen werden, oder sie erhält ein neues Cluster  $C_{neu}$  zugeteilt.)

Priorität der Regel:                      1

Eine Task A soll einem Cluster zugeteilt werden, das einem Prozessor Y fest zugewiesen ist. Gleichzeitig besteht für die Task A ein Prozessorverbot für Prozessor Y (R50).

Wenn      (1)      Für die zur Einteilung an Cluster C ausgewählten Task A besteht ein Task-Prozessor-Verbot für Prozessor Y.  
             & (2)      Das Cluster C (dem die Task A zugeteilt werden soll) ist aufgrund einer festen Zuweisung an den Prozessor Y gekoppelt.

dann                      Die betrachtete Task A darf dem Cluster C nicht zugewiesen werden. (Die entsprechende Task A muß entweder einem anderen (bereits bestehenden) Cluster zugewiesen werden, oder sie erhält ein neues Cluster  $C_{neu}$ .)

Priorität der Regel:                      1

---

Die ausgewählte Task besitzt ein Prozessorverbot. In dem Cluster, dem die Task zugeteilt werden soll, existiert ebenfalls eine Task mit einem Prozessorverbot. Die beiden verbotenen Prozessoren sind nicht identisch (R 51).

*Wenn*      (1)      Für die zur Einteilung an Cluster C ausgewählten Task A besitzt ein Task-Prozessor-Verbot für Prozessor  $Y_1$ .  
             & (2)      Im Cluster C (dem die Task A eingegliedert werden soll) existiert einen Task B die für einen Prozessor  $Y_2$  ein Task-Prozessor-Verbot.

*dann*                      Die betrachtete Task A darf dem Cluster C nicht zugewiesen werden. (Für die entsprechende Task muß entweder ein anderes -bereits existierendes- Cluster C' gefunden werden, oder sie erhält ein neues Cluster  $C_{neu}$  zugeteilt.)

*Priorität der Regel:*                      1

Die ausgewählte Task besitzt eine Prozessorverbot. In dem Cluster, dem die Task zugeteilt werden soll, existiert ebenfalls eine Task mit einem Prozessorverbot. Die beiden verbotenen Prozessoren sind identisch (R52).

*Wenn*      (1)      Für die zur Einteilung ausgewählten Task A besteht ein Task-Prozessor-Verbot für Prozessor  $Y_1$ .  
             & (2)      Im Cluster C (dem die Task A eingegliedert werden soll) existiert eine Task B die ebenfalls für den Prozessor  $Y_1$  ein Task-Prozessor-Verbot.

*dann*                      Die betrachtete Task A wird dem Cluster C zugewiesen, in der die "Partnertask" B bereits liegt.

*Priorität der Regel:*                      1

---

Die ausgewählte Task A besitzt mit Task B im "Zielcluster" ein Allokationsgebot und mit einer Task C eine Allokationsverbot. (R53 bis R56).

*Wenn*      (1)      Für die zur Einteilung an Cluster C ausgewählten Task A besteht ein Allokationsgebot für eine Task B im Zielcluster.  
               & (2)      Für die zur Einteilung ausgewählten Task A besteht ein Allokationsverbot für eine Task C im Zielcluster.

*dann*                      Die betrachtete Task wird einem neuen Cluster  $C_{neu}$  zugewiesen, und die Task B ebenfalls in dieses neue Cluster gelegt.

*Priorität der Regel:*                      1

Eine Task wird aufgrund einer berechneten Metrik einem Cluster zugeteilt (R18/R19).

*Wenn*      (1)      Die zur Einteilung ausgewählte Task A hat keine besonderen Merkmale<sup>6</sup>.  
               & (2)      Die betrachtete Partnertask liegt B schon in einem Cluster.

*dann*                      Falls die Gesamtkommunikationskosten<sup>7</sup> der betrachteten Task A kleiner sind (in Bezug auf die errechnete Metrik) als der (Gesamtkommunikationskosten-) Mittelwert, oder die Kommunikationskosten der Task mit dem Cluster C kleiner sind als der (Kommunikationskosten-) Mittelwert, oder die Anzahl der Verbindungen dieser Task kleiner ist als der Mittelwert aller Verbindungen, dann wird die Task A in ein neues Cluster  $C_{neu}$  gelegt.  
                               Anderenfalls wird die Task A in das Cluster der Partnertask B gelegt.

*Priorität der Regel:*                      3

---

<sup>6</sup> Besondere Merkmale sind bestimmte Einschränkungen, die durch die anderen Regeln in der Regelmenge III übergeprüft werden.

<sup>7</sup> Gesamtkommunikationskosten: Summe der Kosten einer Task (mit allen anderen Tasks)

Falls keine besonderen Merkmale vorliegen (R13/R14).

- Wenn (1) Die zur Einteilung ausgewählte Task A hat keine besonderen Merkmale.  
& (2) Die Partnertask B liegt noch in keinem Cluster.

dann Die betrachtete Task wird einem neuen Cluster  $C_{neu}$  zugewiesen.

Priorität der Regel: 3

Die ausgewählte Task besitzt keine Verbindungskosten (R20/R21).

- Wenn (1) Die zur Einteilung ausgewählte Task A hat keine besonderen Merkmale.  
& (2) Es existiert keine Partnertask B (keine Verbindungskosten).

dann Die betrachtete Task A wird einem neuen Cluster  $C_{neu}$  zugewiesen.

Priorität der Regel: 4

Berechnung der Kosten nachdem eine Task einem Cluster zugewiesen wurde, das eine feste Task-Prozessor-Zuweisung hat (R 32).

- Wenn (1) Die Task A wurde einem Cluster zugewiesen, das schon einem Prozessor Y fest zugeteilt ist (z.B durch Benutzerzuordnung).

dann Die Prozessorauslastung und die Laufzeitkosten für den betrachteten Prozessor Y werden aktualisiert. Bei überschreiten der Speichergrenzen wird der Benutzer um Rat gefragt.

Priorität der Regel: 5

**Regelmenge IV**

Falls ein Cluster neu gebildet wurde, bzw. wenn keine Einigung über eine Zuteilung erreicht wurde, werden die entsprechenden Vorkehrungen für einen weiteren Versuch unternommen.

Priorität der Regelmenge: 4

Kostenberechnung nach Zuweisung einer Task (R30/R31).

*Wenn* (1) Falls eine Clusterzuteilung erfolgte

*dann* Die zugehörigen Kommunikationskosten (bzw. Prozessorkosten) werden entsprechend der Zuweisung verändert.

*Priorität der Regel:* 1

Falls eine Task keinem Cluster zugeteilt werden konnte (R90 bis R93).

*Wenn* (1) Falls keine Clusterzuteilung erfolgte

*dann* Die zugehörigen Kommunikationskosten und die Gesamtkommunikationskosten werden entsprechend reaktiviert, um bei einem weiteren Zuteilungsversuch wieder zur Verfügung zu stehen.

*Priorität der Regel:* 2

**D. Verteilen der Cluster auf die Prozessoren****Regelmenge I**

Die Regeln dieser Regelmenge bestimmen die Strategie, die zu einem bestimmten Zeitpunkt gültig sein soll. Dazu benötigt sie auch Informationen über das ausgewählte Optimierungskriterium (Regelmengen 9 bis 12). Falls keine explizite "Umschaltung" der Strategie erforderlich ist, werden die Regelmengen entsprechend ihrer Prioritäten aktiviert. Diese Prioritäten ergeben sich folgendermaßen:

- A) Auswahl des nächsten Clusters,
- B) Auswahl eines Prozessors, dem dieses Cluster zugewiesen werden soll,
- C) Überprüfung der Optimierungskriterien und
- D) Zuteilung des Clusters an den Prozessor

Priorität der Regelmenge: 1

**Auswahl eines noch unbelegten Prozessors (R78 und R30).**

*Wenn* (1) Es existieren noch freie Prozessoren, die bei der Verteilung nicht berücksichtigt wurden.

*dann* Es wird geprüft, ob diese Prozessoren belegt werden sollen, oder ob andere Verteilungsargumente ("im Moment") wichtiger sind.

*Priorität der Regel:* 1

Es sind bereits alle Prozessoren (mit mindestens einem Cluster) belegt (R78/R32).

*Wenn* (1) Falls alle Prozessoren bereits belegt sind.

*dann* Als Verteilungsstrategie scheidet nun die Bedingung "Prozessoren unbelegt" aus.

*Priorität der Regel:* 2

Einer der Prozessoren ist überlastet (R36).

*Wenn* (1) Der Speicher eines Prozessors ist mit Tasks überlastet.

*dann* Es wird eine Regelmenge (Regelmenge II) aktiviert, die versucht, einige Tasks von diesem Prozessor zu nehmen.

*Priorität der Regel:* 3

Ausbessern von Fehlentscheidungen (R70 bis 75).

*Wenn* (1) Es wurden Fehlentscheidungen rückgängig gemacht (Regelmenge II).

*dann* Es werden die entsprechenden Schritte aktiviert, um eine erneute Verteilung vorzunehmen.

*Priorität der Regel:* 4

---

Der Benutzer fordert eine kombinierte Vorgehensweise bei der Optimierung (R1).

*Wenn* (1) Falls eine Kombination aus verschiedenen Strategien vorliegt.

*dann* Die Regelmengen 9 bis 12 werden aktiviert.

*Priorität der Regel:* 4

### Regelmenge II

Falls es zu Fehlentscheidungen gekommen ist, sorgt diese Regelmenge dafür, daß die entsprechenden Schritte rückgängig gemacht werden und ein weiterer Verteilungsversuch unternommen wird (Backtracking).

*Priorität der Regelmenge:* 3

Es wurde eine Zuteilung trotz Prozessorüberlastung vorgenommen. Aufgrund "neuer" Erkenntnisse kann der entsprechende Prozessor nun entlastet werden. (R200 bis R205).

*Wenn* (1) Wenn ein Prozessor Y bereits überlastet ist, aufgrund der aktuellen Strategie aber das Cluster C unbedingt auf diesen Prozessor zu liegen kommen soll.

*dann* Einige Tasks ( $A_1, \dots, A_n$ ) werden vom entsprechenden Prozessor Y genommen. Dabei muß darauf geachtet werden, daß die Tasks keine Randbedingungen verletzen und die "richtigen" Tasks (im Sinne des gewählten Optimierungskriteriums) entfernt werden.

*Priorität der Regel:* 1



Eine Verbindung zwischen zwei Prozessoren existiert nicht, obwohl sie für eine "richtige" Verteilung benötigt würde (R45 und R200 bis 205).

*Wenn* (1) Ein Cluster fordert eine bestimmte Verbindung zwischen zwei Prozessoren.

& (2) Diese Verbindung existiert nicht.

*dann* Es muß versucht werden eine andere Zuteilung zu erreichen. Andernfalls muß das Cluster "zerlegt" werden (siehe Regelmenge 8).

*Priorität der Regel:* 1

### Regelmenge III

Die Regeln dieser Regelmenge wählen ein Cluster aus, das einem Prozessor zugewiesen werden soll.

Priorität der Regelmenge: 4

Ein Cluster enthält eine Task, die einem Prozessor fest zugewiesen ist (R11/R12).

Wenn (1) Es existiert eine Task A, die eine feste Zuweisung hat.  
& (2) Es existiert ein Cluster C in dem die Task A liegt.

dann Dieses Cluster C wird dem Prozessor zugewiesen, mit dem die Task A fest verbunden ist. ("Feste vom Benutzer vorgegebene Zuweisung").

Priorität der Regel: 1

Ein Cluster enthält eine Task, die einem Prozessor fest zugewiesen ist. Es entstehen Kommunikationswege, die von der Hardware nicht erfüllt werden (bei Optimierungskriterium "Direkte Verbindungen") (R11/R12).

Wenn (1) Es existiert eine Task A, die eine feste Zuweisung hat.  
& (2) Es existiert ein Cluster C in dem die Task A liegt.  
& (3) Das Cluster fordert eine physikalische Verbindungen zwischen zwei Prozessoren  $Y_1$  und  $Y_2$ .  
& (4) Die beiden Prozessoren  $Y_1$  und  $Y_2$  haben keine Verbindung.

dann Es muß versucht werden eine andere Zuteilung zu erreichen. Andernfalls muß das Cluster "zerlegt" werden (siehe Regelmenge 8).

Priorität der Regel: 2

**Regelmenge IV**

Die Regeln dieser Regelmenge wählen ein Cluster aus, das einem Prozessor zugewiesen werden soll. Es werden hier allerdings nur solche Cluster zugewiesen, für die Zuordnungseinschränkungen für mindestens einen Prozessor bestehen.

Priorität der Regelmenge: 4

Für eine Task bestehen eingeschränkte Zuordnungsmöglichkeiten (aufgrund der Berechnungen in der Phase "Auswerten der Zuordnungseinschränkungen") (R20 bis 25).

Wenn (1) Es existiert eine Task A, die nur an bestimmte Prozessoren ( $Y_1, \dots, Y_n$ ) zugeteilt werden darf (Zuordnungseinschränkungen).  
& (2) Es existiert ein Cluster C, in dem die Task A liegt.

dann Das Cluster C wird einem der Prozessoren ( $Y_1, \dots, Y_n$ ) zugewiesen, mit dem die Task A kein Zuordnungsverbot hat. Dazu wird ein geeigneter Prozessor mit Hilfe anderer Regelmengen (z.B. V oder VI) ausgewählt.

Priorität der Regel: 2

Für eine Task bestehen eingeschränkte Zuordnungsmöglichkeiten (aufgrund der Berechnungen in der Phase "Auswerten der Zuordnungseinschränkungen"). Es existiert ein unbelegter Prozessor (R20 bis 25).

Wenn (1) Es existiert eine Task A, die nur an bestimmte Prozessoren ( $Y_1, \dots, Y_n$ ) zugeteilt werden darf (Zuordnungseinschränkungen).  
& (2) Es existiert ein Cluster C, in dem die Task A liegt.  
& (3) Es existiert ein freier Prozessor  $Y_m$  aus der Menge ( $Y_1, \dots, Y_n$ ).

*dann*                    Dieses Cluster C wird bevorzugt dem freien Prozessor  $Y_m$  zugewiesen.

*Priorität der Regel:*    1

### Regelmenge V

Die Regeln dieser Regelmenge wählen ein Cluster aus, das einem Prozessor zugewiesen werden soll. Dazu wird hier ein Cluster betrachtet, das die kleinsten Kommunikationskosten hat. Außerdem wird diese Regelmenge nur aktiv, wenn der Netztyp "Direkte Verbindungen" vorliegt.

Priorität der Regelmenge: 5

Es wird ein Cluster einem Prozessor zugewiesen, das die kleinsten Gesamtkommunikationskosten besitzt. Außerdem wird der Netztyp "Direkte Verbindung" gefordert. (R34, R74/75 und R78).

*Wenn*    (1)    Es existiert ein Cluster C, das noch keinem Prozessor zugewiesen wurde.  
          & (2)    Das Cluster C hat die kleinsten Gesamtkommunikationskosten aller Cluster.  
          & (3)    Es liegt der Netztyp "Direkte Verbindung" vor.  
          & (4)    Es existieren zwei Prozessoren mit den geforderten Verbindung.  
          & (5)    Einer der beiden Prozessoren ist noch unbelegt.

*dann*                    Dieses Cluster wird daraufhin überprüft, ob es dem unbelegten Prozessor zugewiesen werden kann.

*Priorität der Regel:*    3

Es soll ein Cluster einem Prozessor zugewiesen, das die kleinsten Gesamtkommunikationskosten besitzt. Außerdem wird der Netztyp "Direkte Verbindung" gefordert. Die Prozessoren, die in Frage kommen sind aber bereits überlastet (R74/75 und R79).

- Wenn*
- (1) Es existiert ein Cluster C, das noch keinem Prozessor zugewiesen wurde.
  - & (2) Das Cluster C hat die kleinsten Gesamtkommunikationskosten aller Cluster.
  - & (3) Es liegt der Netztyp "Direkte Verbindung" vor.
  - & (4) Es existieren zwei Prozessoren mit der geforderten Verbindung.
  - & (5) Beide Prozessoren sind bereits belegt.

*dann* In diesem Fall muß das Cluster mit Hilfe einer anderen Strategie (z.B. Zuteilung aufgrund minimaler IPC-Kosten) zugewiesen werden. Falls auch dies keinen Erfolg bringt können dann mit Hilfe der Regeln in Regelmenge II einige Tasks von den Prozessoren entfernt werden. (Aufbrechen der Cluster!).

*Priorität der Regel:* 1

Auswahl wie bei der vorigen Regel. Aber in diesem Fall sind beide Prozessoren noch unbelegt. Ausgewählt wird der Prozessor bei dem die geringeren IPC-Kosten entstehen (R44, R74/75 und R89).

- Wenn*
- (1) Es existiert ein Cluster C, das noch keinem Prozessor zugewiesen wurde.
  - & (2) Das Cluster C hat die kleinsten Gesamtkommunikationskosten aller Cluster.
  - & (3) Es liegt der Netztyp "Direkte Verbindung" vor.
  - & (4) Es existieren zwei Prozessoren mit der geforderten Verbindung.
  - & (5) Beide Prozessoren sind noch unbelegt.

*dann* Das Cluster wird daraufhin überprüft, bei welcher Zuteilung die geringeren IPC-Kosten entstehen. Falls kein eindeutiges Ergebnis errechnet wird, muß mit den Regelmengen 9 bis 12 (abhängig von der gewählten Strategie) eine Entscheidung gefunden werden.

*Priorität der Regel:* 1

### Regelmenge VI

Die Regeln dieser Regelmenge wählen ein Cluster aus, das einem Prozessor zugewiesen werden soll. Dazu wird hier ein Cluster betrachtet, das die kleinsten Kommunikationskosten hat. Außerdem wird diese Regelmenge nur aktiv, wenn der Netztyp "Vermaschtes System" vorliegt. Für die Überprüfung des Zuordnungsvorschlages werden -abhängig vom Optimierkriterium- die Regelmengen 9 bis 12 sowie die Regelmenge 13 benutzt.

*Priorität der Regelmenge:* 5

#### Auswahl eines Clusters für die Zuteilung an einen Prozessor (R74B/75B, 78).

*Wenn* (1) Es existiert ein Cluster C, das noch keinem Prozessor zugewiesen wurde.  
& (2) Das Cluster C hat die kleinsten Anzahl von Verbindungen unter allen Clustern.  
& (3) Es liegt das Auswahlkriterium "vermaschtes System" vor.  
& (4) Es ist bereits ein Prozessor Y bestimmt, dem das Cluster C zugewiesen werden soll.

*dann* Das Cluster C wird daraufhin überprüft, ob es dem ausgewählten Prozessor Y zugewiesen werden kann.

*Priorität der Regel:* 2

Auswahl eines Clusters für die Zuteilung an einen unbelegten Prozessor (R74B/75B, 80B).

- Wenn
- (1) Es existiert ein Cluster C, das noch keinem Prozessor zugewiesen wurde.
  - & (2) Dieses Cluster C hat die kleinsten Anzahl von Verbindungen unter allen Clustern.
  - & (3) Es liegt das Auswahlkriterium "vermaschtes System" vor.
  - & (4) Es existiert noch ein unbelegter Prozessor Y, dem das Cluster C zugewiesen werden kann.

dann Das Cluster C wird daraufhin überprüft, ob es dem ausgewählten Prozessor Y zugewiesen werden kann.

Priorität der Regel: 1

Auswahl eines Clusters für die Zuteilung an einen Prozessor (R74B/75B, 82B).

- Wenn
- (1) Es existiert ein Cluster  $C_1$ , das noch keinem Prozessor zugewiesen wurde.
  - & (2) Dieses Cluster  $C_1$  hat die gleiche Anzahl von Verbindungen wie das Cluster  $C_2$ .
  - & (3) Es liegt das Auswahlkriterium "vermaschtes System" vor.
  - & (4) Es existiert ein Prozessor Y, dessen Tasks mit einem der beiden Cluster die größten Kommunikationskosten haben.

dann Dieses Cluster (entweder  $C_1$  oder  $C_2$ ) wird daraufhin überprüft, ob es dem ausgewählten Prozessor Y zugewiesen werden kann.

Priorität der Regel: 3

**Regelmenge VII**

Falls die Regelmengen 3 bis 6 nicht mehr aktiviert werden können, wird nun ein Cluster aufgrund der jeweils höchsten Kommunikationskosten zur Zuteilung ausgewählt. Falls ein Cluster keine Kommunikationskosten mit einem anderen Cluster hat, wird ein beliebiger Prozessor zur Zuteilung ausgewählt.

Priorität der Regelmenge: 6

Auswahl eines Prozessors, mit dem das betrachtete Cluster die höchsten Kommunikationskosten hat (R31/R32/R32-2)

Wenn (1) Es existiert ein zur Zuteilung ausgewähltes Cluster C.  
& (2) Es existiert ein Prozessor Y, mit dem das ausgesuchte Cluster C die höchsten Kommunikationskosten hat.

dann Das Cluster C wird daraufhin überprüft, ob es dem Prozessor Y bekommt zugewiesen werden kann.

Priorität der Regel: 1

Auswahl eines beliebigen Prozessors, für den Fall, daß ein Cluster keine Kommunikationskosten mit anderen Clustern hat (R13).

Wenn (1) Es existiert ein Cluster C, das noch keinem Prozessor zugewiesen wurde.  
& (2) Das Cluster C hat keine Kommunikationskosten mit anderen Clustern.

dann Es wird in diesem Fall ein beliebiger Prozessor ausgewählt. Überprüft wird, ob diese Zuteilung stattfinden soll.

Priorität der Regel: 2



**Regelmenge VIII**

Falls einem Prozessor keine Tasks zugeteilt wurde, (weil beispielsweise die Anzahl der Cluster kleiner ist als die Anzahl der Prozessoren) obwohl dies vom Benutzer gefordert wurde, wird durch Zerlegung eines Clusters versucht, alle Prozessoren auszulasten.

Priorität der Regelmenge: 7

Nach Abschluß der Zuteilung existiert ein Prozessor, der keine Cluster zugeteilt bekommen hat (R80 bis R92).

Wenn (1) Die Zuteilung der Cluster gilt als abgeschlossen.  
& (2) Es existiert (mindestens) ein Prozessor, der keine Task (kein Cluster) zugewiesen bekam.

dann Die zugewiesenen Cluster werden daraufhin überprüft, ob sie einem anderen Prozessor entzogen werden können. Ist dies der Fall werden die Regelmengen 1 bis 7 neu aktiviert.

Priorität der Regel: 1

Wie vorige Regel aber zusätzlich: alle Cluster enthalten feste Zuweisungen (R80B bis R92B).

Wenn (1) Die Zuteilung der Cluster gilt als abgeschlossen.  
& (2) Es existiert (mindestens) ein Prozessor, der keine Task (kein Cluster) zugewiesen bekam.  
& (3) Alle Cluster enthalten mindestens eine feste Zuweisung.

dann Die zugewiesenen Cluster müssen in diesem Fall unter Einbeziehung des Benutzers „aufgebrochen“ werden. Das heißt es müssen einzelne Tasks auf andere Prozessoren verschoben werden.

Priorität der Regel: 1

### Regelmengen IX bis XII

Die Regeln der Regelmengen 9 bis 12 wachen über die Einhaltung der Optimierungskriterien. Neben der Auswahl eines geeigneten Prozessors in den Phasen 4 bis 7 wird hier ebenso ein Prozessor gesucht. Die Bedingungen, die erfüllt sein sollen, richten sich hier aber nach den (vom Benutzer vorgegebenen) Optimierungskriterien.

Da die Regeln der Phasen 9 bis 12 ähnlich aufgebaut sind, werden sie hier auch gemeinsam beschrieben.

Priorität der Regelmenge: 2

Eine Cluster-Prozessor-Zuteilung soll überprüft werden unter dem Gesichtspunkt: "Optimieren der IPC-Kosten" (R40 (20)).

*Wenn*      (1)      Es existiert ein zur Zuteilung an einen Prozessor Y ausgewähltes Cluster C.  
             & (2)      Bei Zuteilung an den Prozessor Y werden die IPC-Kosten des Gesamtsystems optimiert.  
             & (3)      Vom Benutzer wurde die Option "Optimieren nach IPC-Kosten" gewünscht.

*dann*              Das Cluster C wird daraufhin überprüft, ob es dem Prozessor Y zugewiesen werden darf.

*Priorität der Regel:*      2

Eine Cluster-Prozessor-Zuteilung soll überprüft werden unter dem Gesichtspunkt: "Gleichmäßige Laufzeitauslastung" (R40 (30)).

- Wenn*
- (1) Es existiert ein zur Zuteilung an einen Prozessor Y ausgewähltes Cluster.
  - & (2) Bei Zuteilung an diesen Prozessor werden die Laufzeitkosten des Gesamtsystems (zu einem bestimmten Prozentsatz) ausgeglichen.
  - & (3) Vom Benutzer wurde die Option gleichmäßige Laufzeitauslastung gewünscht.
- dann* Das Cluster wird daraufhin überprüft, ob es dem Prozessor zugewiesen werden darf.

*Priorität der Regel:* 2

Eine Cluster-Prozessor-Zuteilung soll überprüft werden unter dem Gesichtspunkt: "Gleichmäßige Laufzeitauslastung und Optimierung der IPC-Kosten" (R40 (40)).

- Wenn*
- (1) Es existiert ein zur Zuteilung an einen Prozessor Y ausgewähltes Cluster.
  - & (2) Es existiert ein Prozessor, mit dem das ausgesuchte Cluster die höchsten Kommunikationskosten hat.
  - & (3) Bei Zuteilung an Prozessor Y werden die Laufzeitkosten des Gesamtsystems zu einem bestimmten Prozentsatz) ausgeglichen.
  - & (4) Vom Benutzer wurde die Option "gleichmäßige Laufzeitauslastung und geringste Kommunikationskosten" gewünscht.
- dann* Das Cluster C wird daraufhin überprüft, ob es diesem Prozessor zugewiesen werden darf.

*Priorität der Regel:* 2

---

Eine Cluster-Prozessor-Zuteilung soll überprüft werden unter dem Gesichtspunkt: "Gleichmäßige Prozessor (Speicher-) auslastung" (R40 (40)).

- Wenn*
- (1) Es existiert ein zur Zuteilung an einen Prozessor Y ausgewähltes Cluster.
  - & (2) Es existiert ein Prozessor, mit dem das ausgesuchte Cluster die höchsten Kommunikationskosten hat.
  - & (3) Bei Zuteilung an diesen Prozessor wird die Prozessorauslastung des Gesamtsystems zu einem bestimmten Prozentsatz) ausgeglichen.
  - & (4) Vom Benutzer wurde die Option "gleichmäßige Prozessorauslastung und geringste Kommunikationskosten" gewünscht.
- dann* Das Cluster wird daraufhin überprüft, ob es dem Prozessor Y zugewiesen werden darf.

*Priorität der Regel:* 2

Überprüfung der Einhaltung der Netzwerktopologie (R40 (50)).

- Wenn*
- (1) Es existiert ein Zuteilungsvorschlag (Cluster C zu Prozessor Y).
  - & (2) Es werden von Cluster C Verbindungen angefordert, die zwischen dem Prozessor Y und seinen "Nachbarprozessoren" nicht bestehen.
  - & (3) Es liegt der Netztyp "Direkte Verbindung" vor.
- dann* Das Cluster C darf dem Prozessor Y nicht zugewiesen werden.

*Priorität der Regel:* 1



Falls die Optimierkriterien nicht greifen (R30).

*Wenn*      (1)      Es existiert ein zur Zuteilung ausgewähltes Cluster C.  
              & (2)      Es existiert ein Prozessor Y.

*dann*              Das Cluster C wird daraufhin überprüft, ob es dem Prozessor  
                         Y zugewiesen werden darf.

*Priorität der Regel:*              3

**Regelmenge XIII**

Die Regeln dieser Regelmenge überprüfen, ob eine Cluster-Prozessor-Zuteilung erlaubt ist.

Priorität der Regelmenge: 8

Ein Cluster kann dem ausgesuchten Prozessor nicht zugewiesen werden, da eine Task (im betrachteten Cluster) ein Prozessorverbot besitzt (R20/R21).

*Wenn*      (1)      Es existiert ein Zuteilungsvorschlag (Cluster C zu Prozessor Y).  
                 & (2)      In dem zur Zuteilung ausgesuchten Cluster existiert eine Task, die für den betreffenden Prozessor Y ein Prozessorverbot besitzt.

*dann*              Für das Cluster C muß eine andere Zuordnung gefunden werden.

*Priorität der Regel:*              1

Überprüfung der Speicherplatzauslastung des ausgesuchten Prozessors (R 40a).

*Wenn*      (1)      Es existiert eine Cluster-Prozessor-Zuweisung.

*dann*              Falls der betreffende Prozessor mit diesem zusätzlichen Cluster (bzw. den im Cluster enthaltenen Tasks) überlastet wäre, darf das Cluster diesem Prozessor nicht zugeteilt werden.  
                      Für das Cluster muß eine andere Zuordnung gefunden werden, falls sich aufgrund einer Nachfrage beim Benutzer keine andere Möglichkeit anbietet.

*Priorität der Regel:*              1

---

Falls ein Cluster keinem der Prozessoren zugewiesen werden kann (R 40b).

*Wenn* (1) Das Cluster kann keinem der getesteten Prozessor zugewiesen werden.

*dann* Falls der betreffende Prozessor mit diesem zusätzlichen Cluster (bzw. den im Cluster enthaltenen Tasks) überlastet wäre, aber keine andere Zuordnung gefunden werden kann, wird das Cluster trotzdem diesem Prozessor zugewiesen.

*Priorität der Regel:* 3

Ein Cluster kann nicht zugeordnet werden aufgrund eines Allokationsverbotes (R41/42).

*Wenn* (1) Für das ausgesuchte Cluster C existiert (für eine Task innerhalb des Clusters) ein Allokationsverbot mit einer Task auf dem zur Zuteilung ausgewählten Prozessor.

*dann* Für das Cluster C muß eine andere Zuordnung gefunden werden.

*Priorität der Regel:* 2

Im Netzwerk fehlt ein Verbindung, die zur korrekten Zuteilung notwendig wäre (R41b/42b).

*Wenn* (1) Für das ausgesuchte Cluster C werden Verbindungen notwendig, die nicht vorhanden sind.

& (2) Es liegt der Netztyp "Direkte Verbindung" vor.

*dann* Für das Cluster C muß eine andere Zuordnung gefunden werden.

*Priorität der Regel:* 2



**Regelmenge XIV**

Die Regeln dieser Regelmenge teilen ein Cluster einem Prozessor zu und erledigen die Verwaltungsarbeit.

Priorität der Regelmenge: 9

Zuteilung eines Clusters an einen Prozessor (R 54).

Wenn (1) Für die ausgesuchte Cluster-Prozessor-Zuweisung (C-Y)  
liegt kein Verbot vor.  
& (2) Das Cluster C hat Kommunikationspartner, die bereits  
zugewiesen sind.

dann Das Cluster C wird dem entsprechenden Prozessor zugeteilt.  
Dazu sind folgende Verwaltungsaufgaben notwendig:

- Berechnung der IPC-Kosten
- Berechnung der Verbindungskosten zwischen den Prozessoren
- Berechnung der Kommunikationskosten innerhalb der  
Prozessoren.

Priorität der Regel: 1

Berechnung der Kosten (R 55).

Wenn (1) Ein Cluster wurde einem Prozessor zugewiesen.

dann Die Prozessorauslastung und die Laufzeitkosten für den  
betrachteten Prozessor werden aktualisiert.

Priorität der Regel: 2

## E. Verschieben einzelner Tasks

### Regelmenge I

Die Regeln dieser Regelmenge wählen eine Task aus, die auf einen anderen Prozessor verschoben werden soll.

Priorität der Regelmenge: 1

Tasks, für die ein Allokationsverbot besteht werden nicht verschoben (R2).

*Wenn* (1) Falls für eine Task ein Allokationsgebot besteht.

*dann* Diese Task soll nicht verschoben werden.

*Priorität der Regel:* 1

Es wird die Task ausgewählt, die dem größten Cluster angehört (R11).

*Wenn* (1) Betrachtet wird die Task, die dem Cluster mit der größten Taskanzahl angehört.

*dann* Die Task wird daraufhin untersucht, ob sie auf einen anderen Prozessor verschoben werden soll.

*Priorität der Regel:* 2

Es wird die Task ausgewählt, die die größten Kommunikationskosten hat und noch nicht verschoben wurde (R12).

*Wenn* (1) Betrachtet wird die Task, die die höchsten Kommunikationskosten hat und noch nicht verschoben wurde.

*dann* Die Task wird daraufhin untersucht, ob sie auf einen anderen Prozessor verschoben werden soll.

*Priorität der Regel:* 2

**Regelmenge II**

In dieser Menge wird für die vorher (in der Regelmenge II) bestimmte Task die Entscheidung getroffen, ob diese Task verschoben werden darf.

Priorität der Regelmenge: 2

Überprüfung, ob der Zielprozessor überlastet ist (R 51).

*Wenn* (1) Der Zielprozessor wäre mit der zusätzlichen Task überlastet.

*dann* Die betrachtete Task darf in dieser Situation nicht verschoben werden. (Für die Task können noch andere Versuche zur Umverteilung angestrengt werden).

*Priorität der Regel:* 1

Überprüfung, ob ein Allokationsverbot am Zielprozessor besteht (R52/R53).

*Wenn* (1) Für die zum Verschieben ausgewählte Task besteht ein Allokationsverbot mit einer anderen Task auf dem Zielprozessor.

*dann* Die betrachtete Task darf in dieser Situation nicht verschoben werden. (Für die Task können noch andere Versuche zur Umverteilung angestrengt werden).

*Priorität der Regel:* 1

Überprüfung, ob ein Task-Prozessor-Verbot am Zielprozessor besteht (R54).

*Wenn* (1) Für den Prozessor, auf den die Task verschoben werden soll, besteht (für die betrachtete Task) ein Task-Prozessor-Verbot.

dann Die betrachtete Task darf in dieser Situation nicht verschoben werden. (Für die Task können noch andere Versuche zur Umverteilung angestrengt werden).

*Priorität der Regel:* 1

Überprüfung, ob der Prozessor "entleert" wird (R55).

Wenn (1) Die Task, die verschoben werden soll, ist die einzige Task auf dem Prozessor, von dem sie weg soll.  
& (2) Es sollen alle Prozessoren belegt sein.

dann Die betrachtete Task darf in dieser Situation nicht verschoben werden. (Für die Task können nur dann noch andere Versuche zur Umverteilung angestrengt werden, wenn vorher eine weitere Task auf den betrachteten Prozessor gelegt wird).

*Priorität der Regel:* 1

Überprüfung, ob die Topologie eine Verschiebung zuläßt (R56).

Wenn (1) Durch die Verschiebung entstehen Verbindungsanforderungen zwischen Tasks, die von der physikalischen Verbindung zwischen den Prozessoren nicht ausgeführt werden können.  
& (2) Es liegt der Netztyp "Direkte Verbindung" vor.

dann Die betrachtete Task darf in dieser Situation nicht verschoben werden. (Für die Task können nur dann noch andere Versuche zur Umverteilung angestrengt werden, wenn sich die Gesamtverteilung aller Tasks ändert).

*Priorität der Regel:* 1

Überprüfung, ob ein Allokationsgebot verletzt wird (R57).

*Wenn*      (1)      Für die zum Verschieben ausgewählte Task besteht ein Allokationsgebot, das vom Benutzer vorgegeben ist.

*dann*              Die betrachtete Task darf nicht verschoben werden.

*Priorität der Regel:*              1

**Regelmenge III**

Falls keine Einwände gegen die Verschiebung vorliegen, wird nun geprüft, ob die Verschiebung eine Verbesserung bringt.

*Priorität der Regelmenge:*      3

Überprüfung, ob die Verschiebung eine Verbesserung der IPC-Kosten mit sich bringt (R40-43).

*Wenn*      (1)      Das Verschieben der ausgewählten Task bringt eine Verbesserung der Gesamtkommunikationskosten.

            & (2)      Der Benutzer hat optimale Kommunikationskosten als Optimierungskriterium gewählt.

*dann*              Die Task wird verschoben. Dazu sind umfangreiche Verwaltungsaufgaben notwendig, die von eigenen Regeln ausgeführt werden.

*Priorität der Regel:*              1

---

Überprüfung, ob die Verschiebung eine Verbesserung der Laufzeitauslastung mit sich bringt (R40-20).

- Wenn*      (1)      Das Verschieben der ausgewählten Task bringt eine Verbesserung der Laufzeitgleichauslastung.
- & (2)      Der Benutzer hat gleichmäßige Laufzeitauslastung als Optimierungskriterium gewählt.

*dann*                      Verschiebung der Task. Dazu sind umfangreiche Verwaltungsaufgaben notwendig, die von eigenen Regeln ausgeführt werden.

*Priorität der Regel:*                      1

Überprüfung, ob die Verschiebung eine Verbesserung der Laufzeitauslastung und gleichzeitig der IPC-Kosten mit sich bringt (R40-30).

- Wenn*      (1)      Das Verschieben der ausgewählten Task bringt eine Verbesserung der Kommunikationskosten und der Laufzeitgleichauslastung.
- & (2)      Der Benutzer hat Kommunikationskosten und gleichmäßige Laufzeitauslastung als Optimierungskriterium gewählt.

*dann*                      Die Task wird verschoben.

*Priorität der Regel:*                      1

Überprüfung, ob die Verschiebung eine Verbesserung der Prozessorauslastung und gleichzeitig der IPC-Kosten mit sich bringt (R40-40).

Wenn      (1)      Das Verschieben der ausgewählten Task bringt eine Verbesserung der Kommunikationskosten oder (und) der gleichmäßigen Prozessorauslastung.  
             & (2)      Der Benutzer hat Kommunikationskosten und gleichmäßige Prozessorauslastung als Optimierungskriterium gewählt.

dann              Die Task wird verschoben.

Priorität der Regel:              1

## **F. Verschieben ganzer Cluster**

Die Vorgehensweise der Verschiebung ganzer Cluster ist der Verschiebung der Tasks sehr ähnlich und braucht deshalb hier nicht näher betrachtet zu werden.

## Literaturangaben

- /AHO74/ Aho, Alfred V., Kohn E. Hopcroft, Jeffrey D. Ullmann :  
"The Design and Analysis of Computer Algorithms",  
Addison-Wesley Publishing Company, London 1974
- /ANDR88/ Andres, Christian :  
"Ein graphentheoretischer Ansatz zur rechnergestützten parallelen  
Komposition von Prozessen für verteilte Echtzeitsysteme",  
Dissertation, Universität Erlangen-Nürnberg 1988
- /AROR80/ Arora, R.K., Rana, S.P. :  
"Heuristic Algorithms for Process Assignment in Distributed  
Computing Systems",  
Information Processing Letters, Vol. 11(4,5), pp. 199-203,  
New Delhi December 1980
- /BANN83/ Bannister, J.A., Trivedi, K.S. :  
"Task Allocation in Fault-Tolerant Distributed Systems",  
Acta Informatica 20, 1983
- /BAUE88/ Bauer, Sabine :  
"Vergleich des wissensbasierten Systems ALLOC mit optimalen  
und heuristischen Verfahren zur Lösung des Allokationsproblems",  
Studienarbeit, Universität Erlangen-Nürnberg, 1985
- /BECK85/ Beckstein, Clemens :  
"Integration objektorientierter Sprachmittel zur Wissensrepräsen-  
tation in LISP",  
Diplomarbeit, Universität Erlangen-Nürnberg, 1985
- /BECK88/ Beckstein, Clemens :  
"Zur Logik der Logik-Programmierung - Ein konstruktiver Ansatz",  
Dissertation, Universität Erlangen-Nürnberg, 1988



- 
- /BEIE88/ Beier, Konrad :  
"Steuerung eines Zentralmastmanipulators (ZMM3)",  
Siemens AG, Mündliche Mitteilungen, Erlangen 1988
- /BESO85/ Besold, Rainer :  
"Ein universelles Überwachungs- und Automatisierungssystem für  
kern- und teilchenphysikalische Experimente",  
Dissertation, Universität Erlangen-Nürnberg 1985
- /BODE80/ Bode, A., Händler, W :  
"Rechnerarchitektur, Grundlagen und Verfahren",  
Springer-Verlag, Berlin Heidelberg New York 1980
- /BOBR83/ Bobrow, D. G., Stefik, M. :  
"The LOOPS Manual",  
Xerox PARC, Palo Alto, CA, 1983
- /BORR86/ Borrmann, Lothar :  
"Allokation von Rechenprozessen in verteilten Realzeitsystemen",  
VDI Verlag, Düsseldorf 1986
- /CHOU82/ Chou, T.C.K., Abraham, J. A. :  
"Load Balancing in Distributed Systems",  
IEEE Transaction on Software Engeneering, Vol. SE-8, No. 4,  
pp. 401-412, July 1982
- /CHOW79/ Chow, Y., Kohler, W. H. :  
"Models for Dynamic Load Balancing in a Heterogenous Multiple  
Processor System",  
IEEE Transactions on Computers, Vol. c-28, No. 5,  
pp. 345-361, May 1979
- /CHU69/ Chu, W. W. :  
"Optimal file allocation in a multiple computing system",  
IEEE Transaction on Computers, Vol c-18,  
pp. 885-889, Oct. 1969

- /CHU80/ Chu, W.W., et al. :  
"Task Allocation in Distributed Data Processing",  
Computer Vol. 13, pp. 57-69, November 1980
- /CHU84/ Chu, W.W., Lan, M., Hellerstein, J. :  
"Estimation of Intermodule Communication (IMC) and its Applications in Distributed Processing Systems",  
IEEE Transactions on Computers, Vol. C-33, No. 8,  
pp. 691-669, August 1984
- /COFF76/ Coffman, Jr. Ed. :  
"Computer and Job-Shop Scheduling Theory",  
New York 1976
- /COLE83/ Coleman, D.M., Jordan jr, B.W., Swamy, S. :  
"A Systematic Approach to the Design of Fault Tolerant Distributed Computer Systems",  
Proc. 2nd anual Phoenix Conf. on Computers and Communication,  
March 1983
- /DERM76/ Mc Dermott, J., Forgy, C. :  
"Production Systems Conflict Resolution Strategies"  
Carnegie-Mellon University, Computer Science Dept.,  
Pittsburg, PA 15213, 1976
- /DERM80/ Mc Dermott, J.  
"R1 : A Rule-Based Configurer of Computer Systems",  
Carnegie-Mellon University, Pittsburg, April 1980
- /DUT82/ Dutta, A., Koehler G., Whinston, A. :  
"On Optimal Allocation in a Distributed Processing Environment",  
Management Science, Vol. 28, no 8, August 1982

- 
- /ECKE77/ Ecker, Klaus :  
"Organisation von parallelen Prozessen, Theorie deterministischer Schedules",  
Zürich 1977
- /EFE82/ Efe, Kemal :  
"Heuristic Models of Task Assignment Scheduling in Distributed Systems",  
Computer, June 1982
- /FEIG81/ Feigenbaum, Edward A., Barr, A. :  
"The Handbook of Artificial Intelligence",  
HeurisTech Press, Stanford, California 1981
- /FLEI84/ Fleischmann, Albert :  
"Ein Konzept zur Darstellung und Realisierung von verteilten Prozeßautomatisierungssystemen",  
Dissertation, Universität Erlangen-Nürnberg 1984
- /FORG81/ Forgy, Charles L. :  
"OPS5 User's Manual",  
Department of Computer Science, Carnegie-Mellon University,  
Pittsburg, Pennsylvania 15213, July 1981
- /FORG84/ Forgy, Charles L. :  
"The OPS83 Report",  
Department of Computer Science, Carnegie-Mellon University,  
Pittsburg, Pennsylvania 15213, May 1984
- /FORG86/ Forgy, Charles L. :  
"The OPS83 User's Manual, System Version 2.2",  
Production System Technologies Inc, July 1986

- /FRIE85/ Friedland, P., Kedes L. :  
"Discovering the Secrets of DNA",  
Computer (IEEE Computer Society), Volume 18,  
Number 11, pp. 49-69, New York, November 1985
- /GAB84/ Gabrelian, A., Tyler D. :  
"Optimal object allocation in distributed computer systems",  
Proc. 4th Int'l Conf. on Distributed Computing Systems, 1984
- /GARF72/ Garfinkel, R.S., Nemhauser, G.L. :  
"Integer Programming",  
John Wiley and Sons, New York - London, 1972
- /GYL76/ Gylys, V.B., Edwards, J.A. :  
"Optimal Partitioning of Workload for Distributed Systems",  
Proc. COMPCON, 1976
- /HÄND84/ Händler, W., Herzog, U., Hofmann, F., Schneider, H.J. :  
"Multiprozessoren für breite Anwendungsgebiete : Erlangen General  
Purpose Array",  
GI/NTG-Fachtagung Architektur und Betrieb von Rechensystemen,  
Informatik Fachberichte Nr. 78, Springer Verlag 1984
- /HAE80/ Haessig, K., Jenny, C. J. :  
"Partitioning and Allocating Computational Objects in Distributed  
Computing Systems",  
Proc. of the IFIP Congress, Melbourne, Australia, Oct. 1980
- /HART75/ Hart, P. E. :  
"Progress on a Computer-Based Consultant",  
Proc. IJCAI-75, 1975, pp 831-841, 1975

- 
- /HASL83/ Hasling, D. W., Clancey, W. J., Rennels, G. :  
"Strategic Explanations for a Diagnostic Consultant System",  
Department of Computer Science, Stanford University CA, 1983
- /HAZE85/ Hayes-Roth, F. :  
"Special Selection : Rule-Based System",  
Communication ACM, Vol.28, Number 9, Sept.1985
- /HEIL85/ Heilmeier, Elfriede :  
"Erfassung von Telefongesprächsdaten mit einem verteilten  
System von Mikrorechnern",  
Diplomarbeit Universität Erlangen-Nürnberg 1985
- /HILB80/ Hilberg, Wolfgang :  
"Partitionierung mit Hilfe der Verbindungsmatrix",  
Institut für Datentechnik, Technische Hochschule Darmstadt,  
ntz-Archiv Bd.3, 1981 H.3
- /HOAR78/ Hoare, C.A.R. :  
"Communicating Sequential Processes",  
Communications of the ACM, Volume 21, Number 8, August 1978
- /HOAR85/ Hoare, C.A.R. :  
"Communicating Sequential Processes",  
Prentice Hall International, London 1985
- /HOFM83/ Hofmann, Fridolin :  
"Koordinierung Asynchroner Prozesse",  
Vorlesung im Wintersemester 83/84, Universität Erlangen-Nürnberg
- /HOFM84/ Hofmann, Fridolin :  
"Betriebssysteme : Grundkonzepte und Modellvorstellungen",  
B. G. Teubner Stuttgart, 1984

- /HOLL89/ Holleccek, P., Andres, C. :  
"A Programming Environment for Distributed Realtime Applications",  
Paper to be submitted to HICSS-22, Erlangen 31.05.88
- /HOPP84/ Hoppe, Beate :  
"Deterministisches Zuordnen vielfach durchlaufender Aufgabensysteme in Multiprozessorsystemen",  
Dissertation, Universität Erlangen-Nürnberg 1984
- /HORO81/ Horowitz, E., Sahni, S. :  
"Algorithmen, Entwurf und Analyse",  
Springer-Verlag, Berlin Heidelberg New York 1981
- /HUAN80/ Huan, W.H, El-Dessouki, O. I. :  
"Distributed Enumeration on Network Computers",  
IEEE Transaction on Computers, vol c-29,  
pp. 818-825 Sept. 1980
- /INTE85/ IntelliCorp :  
"KEE - User's Manual", 1985
- /ISO88/ Nadind, Riboulet :  
"First Working Document Related to Artificial Intelligence and Expert Systems (New Work Item 1.01.28)",  
Information Technology / Vokabulary,  
ISO / CEI JTC 1 / SC 1 N 1108, Septembre 1988
- /JENN77/ Jenny, C. J. :  
"Process Partitioning in Distributed Systems",  
Proc. National Telecom., Conf., Los Angeles, CA, Dec. 1977

- 
- /KAS82/ Kasahra, H., Narita S. :  
"Parallel Processing for Real-Time Control and Simulation of DCCS",  
Proc. 4th IFAC Workshop on Distributed Computer Control Systems,  
Talinn, U.S.S.R., 1982
- /KAR84/ Kar, G., Nikolaou, C.N., Reif, J. :  
"Assigning Processes to Processors : A Fault Tolerant Approach",  
Proc. 14th Int'l Conf. on Fault Tolerant Computing Systems, 1984
- /KERA82/ Keramidis, Sawwas :  
"Eine Methode zur Spezifikation und korrekten Implementierung von asynchronen Systemen",  
Arbeitsbericht des IMMD Erlangen, 1983
- /KRAG86/ Kragl, Gudrun :  
"Eine Programmierungsumgebung für verteiltes PEARL, Codeerzeugung mittels Programmsynthese",  
Dissertation, Universität Erlangen-Nürnberg 1986
- /LAMP83/ Lampson, B.W., Paul, M., Siebert, H.J. :  
"Distributed Systems, An Advanced Course",  
Springer-Verlag, Berlin Heidelberg New York 1983
- /LAUE79/ Lauer, H. C., Needham, R. M. :  
"On the Duality of Operating System Structures",  
ACM Operating Systems Reviews, April 1979
- /LEE77/ Lee, R.P, Muntz, R.R. :  
"On the Task Assignment Problem for Computer Networks",  
Proc. of 10th Hawai Int'l Conf. on System Science 1977

- /LEVI81/    Levi, Paul :  
              "Betriebssysteme für Realzeitanwendungen",  
              Datakontext-Verlag Köln, 1981
- /LO84/        Lo, V.M. :  
              "Heuristic Algorithms for Task Assignment in Distributed Systems",  
              Proc. 4th Int'l Conf. on Distributed Processing, 1984
- /LEH83/      Lehmann, Egbert :  
              "Expertensysteme,  
              Überblick über den aktuellen Entwicklungsstand",  
              Siemens AG, ZT ZTI INF 131, München, 1983
- /MA82/        Ma, P.Y., Lee, E.Y.S., Tsuchiya, M. :  
              "A Task Allocation Model for Distributed Computing Systems",  
              IEEE Transaction on Computers, Vol c-31, no 1,  
              pp. 41-47, January 1982
- /MAKE82/     Mackert, Lothar :  
              "Modellierung, Spezifikation und korrekte Realisierung von asyn-  
              chronen Systemen",  
              Dissertation, Universität Erlangen-Nürnberg 1982
- /MEIE88/     Meier, Werner :  
              "Erklärung in Expertensystemen",  
              Diplomarbeit, Universität Erlangen-Nürnberg 1988
- /MERT83/     Mertens, P., Allgeyer, K. :  
              "Künstliche Intelligenz in der Betriebswirtschaft",  
              Arbeitspapiere Betriebswirtschaftliches Institut der Friedrich-  
              Alexander Universität Erlangen-Nürnberg, 1983



- 
- /MEYE79/ Meyer, Manfred, Hansen, Klaus :  
"Planungsverfahren des Operation Research",  
Band 1, Essen 1972
- /MILN80/ Milner, Robert :  
"A Calculus of Communicating Systems",  
Springer-Verlag, Berlin Heidelberg New York 1980
- /MINS75/ Minsky, M. :  
"A Framework for Representing Knowledge",  
in : P. Winston (ed.) "The Psychology of Computer Vision",  
pp. 211-277, McGraw-Hill, New York 1975
- /MUEL70/ Müller-Merbach, Heiner :  
"Optimale Reihenfolgen",  
Springer-Verlag, Berlin Heidelberg New York 1970
- /NETT79/ Nett, Edgar :  
"Deterministische Strategien zum Scheduling von Taskabhängig-  
keitsstrukturen für Mehrprozessorsysteme",  
R. Oldenburg Verlag, München Wien 1979
- /NEVE63/ Nevell, A., Simon, H.A. :  
"GPS, a Program that Simulates Human Thought",  
In : Feigenbaum, E. Feldmann, J. (Eds.), "Computers and Thought",  
pp. 279-293, McGraw-Hill, New York, 1963
- /NEVE73/ Nevell, A :  
"Production Systems : Modells of Control Structures",  
Visual Information Processing. Acad. Press,  
pp 463 - 526, New York 1973

- /NEVE69/ Newell, A. :  
"Heuristic Programming : Ill-Structured Problems",  
In: Aronofsky, J.S. (Hrsg.) : "Progress in Operations Research",  
Vol. III: Relationship between Operations Research and the  
Computer,  
pp. 361-414, New York 1969
- /NICO89/ Nicol, D. M. :  
"Optimal Partitioning of Random Programs Across Two Processors",  
IEEE Transaction on Software Engineering, Vol. 15, NO 2,  
pp. 134-141, February 1989
- /NIEM88/ Niemann :  
"Prozeßautomatisierung in der Automobilindustrie",  
Siemens AG, Mündliche Mitteilungen, 1988
- /OUST82/ Ousterhout, J. K. :  
"Scheduling Techniques for Concurrent Systems",  
Electrical Engineering and Computer Sciences,  
pp. 22-30, Berkeley, CA 94720, 1982
- /O'SH84/ O'shea, T., Eisenstadt, M. :  
"Artificial Intelligence, Tools, Techniques, and Applications",  
Harper and Row, New York, 1984
- /POST43/ Post, E :  
"Formal Reductions of the General Combinatorial Problem",  
American Journal of Mathematics 65, pp. 127-268, 1943
- /PRI84/ Price, C.C. :  
"The Assignment of Computational Tasks among Processors in a  
Distributed System",  
Proc. NCC, 1981

- 
- /PRI84/ Price, C.C, Krishnaprasad, S. :  
"Software Allocation Models for Distributed Computer Systems",  
Proc. 4th. Int'l Conf on Distributed Processing 1984
- /REIN77/ Reingold, E. M., Nievergeld J. :  
"Combinatorial Algorithms : Theory and Practice",  
Prentice-Hall, Inc., Englewood Cliffs, New York 1977
- /RAULEF/ Raulefs, Peter :  
"Expertensysteme",  
Fachbereich Informatik, Universität Kaiserslautern
- /RETT83/ Retti, J :  
"Artificial Intelligence - Eine Einführung",  
B.G. Teubner Stuttgart, 1984
- /RAO79/ Rao, G.S., Stone, H.S., Hu, T.C. :  
"Assignment of Tasks in a Distributed Processor System with  
Limited Memory",  
IEEE Transaction on Computers, Vol. C-28, no 4,  
pp. 291-299, April 1979
- /RUBI85/ Rubinoff, Robert :  
"Explaining Concepts in Expert Systems : The CLEAR System",  
Computer and Information Science,  
University of Pennsylvania, 1985
- /SCHE86/ Schefe, Peter :  
"Künstliche Intelligenz-Überblick und Grundlagen,  
Grundlegende Konzepte und Methoden zur Realisierung von  
Systemen der künstlichen Intelligenz",  
Mannheim, Wien, Zürich : Bibliographisches Institut, 1986

- /SCHI88/ Schick, Stephan :  
"Graphische Aufbereitung von Ergebnissen eines Allokations-  
systems",  
Laufende Studienarbeit, Universität Erlangen-Nürnberg 1988
- /SEPP86/ S.E.P.P. :  
"Graphischer Editor ZP-86",  
Zeichenprogramm für den IBM-PC/XT/AT,  
Benutzerhandbuch, Version 4.0-D, 1986
- /SHOR76/ Shortliffe, E. H. :  
"Computer-Based Medical Consultations : MYCIN",  
American Elsevier, 1976
- /SHOR76/ Shortliffe, E. H., Buchanan, B.G. :  
"Rule-Based Expert System",  
"The MYCIN Experiments of the Stanford Heuristic Programming  
Project",  
Addison-Wesley Publishing Company, Massachusetts 1984
- /SPÄT75/ Späth, Helmuth :  
"Cluster-Analyse-Algorithmen zur Objektklassifizierung und  
Datenreduktion",  
R. Oldenburg Verlag GmbH, München
- /STAN84/ Stankovic, J.A., Sidhu, J.S. :  
"An Adaptive Bidding Algorithm For Processes, Clusters and Dis-  
tributed Groups",  
Department of Electrical and Computer Engineering, University  
of Massachusetts, Amherst, Mass. 01003, pp. 49-59, 1984

- 
- /STEF84/ Stefik, M., Aikins, J., Balzer, R., Benoit J., Birnbaum, L., Hayes-Roth, F., Sacerdoti, E. :  
"The Architecture of Expert Systems: A Guide to the Organization of Problem-Solving Programs",  
in: Hayes-Roth, F., Waterman, D., Lenat, D., (eds.) :  
"Building Expert Systems", 1984
- /STON77/ Stone, H.S :  
"Multiprocessor Scheduling with Aid of Network Flow Algorithms",  
IEEE Trans. on Software Engineering, Vol. SE-3, no 1, Jan. 1977
- /STONE78/ Stone, H.S., Bokhari, S.H :  
"Control of Distributed Processes",  
IEEE-Computer, pp. 97-106, 1978
- /STOY87/ Stoyan, Herbert :  
"Programmiermethoden der "Künstlichen Intelligenz",  
Universität Konstanz, September 1987
- /STRE75/ Streim, H :  
"Heuristische Lösungsverfahren, Versuch einer Begriffserklärung",  
In: Zeitschrift für Operations Research, Band 19,  
Seite 143-162, Physica-Verlag, Würzburg 1975
- /SUWA82/ Suwa, M., Scott, A. C., Shortliffe, E. H.,  
"An Approach to Verifying Completeness and Consistency in a  
Rule-Based Expert System",  
Heuristic Programming Project,  
Departments of Computer Science and Medicine,  
Standford University California 94305,  
Report No. STAN-CS-82-922, 1982
- /TANE85/ Tanebaum, A.S., Van Renesse, Robbert :  
"Distributed Operating Systems",  
Computer Surveys Vol. 17, No 4,  
pp. 419-470, December 1985

- /TIEL86/ Tielemann, Michael :  
"Eine regelorientierte Erweiterung des Repräsentationssystems  
FORK",  
Diplomarbeit, Universität Erlangen-Nürnberg 1986
- /TRAU87/ Trautner, Martin :  
"Erfassung von Telefongesprächsdaten mit einem verteilten  
System von Mikrorechnern",  
RRZE, Mündliche Mitteilungen, Erlangen 1987
- /TURI63/ Turing, A. M :  
"Computing Machinery and Intelligence",  
in : Feigenbaum, E. A., Feldman, J. (eds),  
"Computers and Thought", New York 1963
- /WATE75/ Waterman, Donald A. :  
"Adaptive Production System",  
Advance Papers of the Fourth IJCAI, Los Altos,  
pp. 296-303, 1975
- /WATE86/ Waterman, Donald A. :  
"A Guide to Expert Sytems",  
Addison-Wesley Publishing Company, Reading, Massachusetts, 1986
- /WECK82/ Weck, Gerhard :  
"Prinzipien und Realisierung von Betriebssystemen",  
B.G. Teubner Stuttgart 1982
- /WISS88/ Wisse, Martin :  
"LABER eine Erklärungskomponente für das wissensbasierte  
Allokationssystem ALLOC",  
Studienarbeit, Universität Erlangen-Nürnberg, 1989

- /WITT80/    Wittie, L.D., Andre van Tilborg :  
              "Micros, A Distributed Operating System for Micronet, A Recon-  
              figurable Network Computer",  
              IEEE Transactions on Computers, Vol. C-29, No. 12,  
              pp. 1133-1144, December 1980
- /WU80/      Wu, S.B., Liu, M.T. :  
              "Assignment of Tasks and Resources for Distributed Processing",  
              Proc. COMPCON, 1980
- /ZAVE82/    Zave, Pamela :  
              "An Operational Approach to Requirements Spezifikation for  
              Embedded Systems",  
              IEEE Transactions on Software Engineering, Vol. SE-8, No. 3,  
              pp. 250269, May 1982
- /ZAVE84/    Zave, Pamela :  
              "The Operational Versus the Conventional Approach to Software  
              Development",  
              Communication of the ACM, Vol. 27, No 2,  
              pp. 104-118, February 1984

## Stichwortverzeichnis

### A

Ablaufsteuerung . . . . .	22-25
ALLOC . . . . .	1, 2, 4, 85, 86, 119, 121, 124, 125, 129, 137, 145, 147-149, 158-160, 164, 166
Approximationsproblem . . . . .	32
Arbeitsspeicher . . . . .	74, 75, 77-80

### B

Benutzermaschine . . . . .	22
----------------------------	----

### C

Cluster . . . . .	55, 90, 91, 93, 98, 99, 101-103, 107-110, 113, 115, 116, 132, 137-140, 143, 144, 150, 155
Cluster-Analyse . . . . .	55
Clusteranalyse . . . . .	93, 97, 106, 115, 116, 120, 132, 137, 143, 155
Clusterbildung . . . . .	112, 147
Clustereinteilung . . . . .	111, 115
Clustern . . . . .	112
Clusternummer . . . . .	134

### D

Dynamische Allokation . . . . .	18
---------------------------------	----

### E

Entfernungsmatrix . . . . .	13
Enumeration . . . . .	33, 34, 57
Enumerationsverfahren . . . . .	35, 42
Erklärungskomponente . . . . .	4, 82, 122, 142-145, 147, 155, 165
Expertensystem . . . . .	61, 63, 77, 81-83, 86, 127, 131, 135-137
Expertenwissen . . . . .	2, 4, 60, 73, 81, 84, 88, 92, 103, 106, 115, 120, 164, 165



**G**

Gesamt-IPC-Kosten . . . . . 56

**H**

Hardwarestruktur . . . . . 23, 123–129, 144, 150

Heuristische Lösung . . . . . 83

Heuristische Methoden . . . . . 2, 59, 115, 153, 164

Heuristische Verfahren . . . . . 4, 28, 29, 37–40, 50, 53, 57, 149, 150, 156

Heuristischer Ansatz . . . . . 156

**I**

Inferenz . . . . . 65, 70, 72, 116

Inferenzmaschine . . . . . 81, 86, 104, 108, 165

Interpreter . . . . . 77, 81, 104, 131

IPC-Kostenfunktion . . . . . 11–13, 27, 42

IPC-Kosten . . . . . 15, 56–58, 89, 100, 152, 154, 158–161

**K**

KI . . . . . 2, 4, 60–64

KI-Methoden . . . . . 81, 115

Kommunikationskosten . . . . . 9, 11–13, 17, 23, 24, 30, 34, 43–47, 50, 51,  
55, 90, 91, 93, 98, 106, 107, 110, 111, 113, 115, 122, 123, 127,  
128, 135, 137, 140, 141, 143, 144, 150, 151, 156, 158–162

Kommunikationsmaschine . . . . . 22

Kommunikationsstruktur . . . . . 22, 23, 106, 116, 123, 124, 126, 127, 128, 129,  
144, 150, 151, 153, 157

Konfliktauflösung . . . . . 75

Konfliktmenge . . . . . 74–76, 78, 79, 131, 132

**L**

Lineare Programmierung . . . . . 26, 33

**N**

Netzwerk . . . . .	13, 45-47, 68, 89, 97, 144, 150, 161
Netzwerkanforderungen . . . . .	166
Netzwerkgraphen . . . . .	46
Netzwerkparameter . . . . .	89, 166
Netzwerktopologie . . . . .	17, 22, 97, 98, 128
Netzwerktyp . . . . .	138
NP-schwer . . . . .	31, 32
NP-vollständig . . . . .	4, 30-32, 34-37, 40

**P**

PASS . . . . .	3, 21, 22, 24, 25, 94, 123, 147
PASS-Spezifikation . . . . .	22
Programmierungsumgebung . . . . .	3, 21, 24, 58, 123, 165, 166
Prozeßautomatisierung . . . . .	1, 6, 21, 40

**R**

Randbedingungen . . . . .	2, 4, 13-15, 28, 32-34, 37, 40, 43, 49, 50, 53-59, 73, 84, 89, 90, 93, 98, 99, 101, 106, 112, 114, 116, 117, 119, 135, 137, 148, 150, 152, 154, 165, 166
Realzeitsystem . . . . .	1, 2, 5-11, 13, 17, 18, 20, 21, 23, 24, 49, 54, 59, 92-95, 97, 123, 150, 163, 166
Regelsysteme . . . . .	73
Repräsentation . . . . .	65-67, 86, 101-103
Repräsentationsmethode . . . . .	67
Repräsentationsschema . . . . .	105
Rückwärtsverkettung . . . . .	77-79

**S**

Scheduler . . . . .	81, 104
Scheduling . . . . .	19, 20, 39
Spezifikationsmethode . . . . .	9, 21, 22, 24, 58, 94, 123

Statische Allokation . . . . .	18
--------------------------------	----

---

**V**

Volumenmatrix . . . . .	12
Vorwärtsverkettung . . . . .	77, 78

**W**

Wissen . . . . .	2, 4, 58, 64-67, 70, 72, 81, 83-86, 88, 89, 102, 103, 116, 164, 166
gebietsspezifisches . . . . .	66
Meta- . . . . .	65, 86, 104
problemspezifisches . . . . .	2, 65, 104
Wissensakquisition . . . . .	83
Wissensbasierter Ansatz . . . . .	4, 60, 73, 93, 165
Wissensbasiertes Allokationssystem . . . . .	88
wissensbasiertes System . . . . .	1, 2, 4, 60, 62, 64-66, 72, 81, 86, 87, 121, 124, 129, 142, 147, 149-152, 158-160, 164, 166
Wissensbasiertes Verfahren . . . . .	165
Wissensbasis . . . . .	65-67, 72, 74, 81-86, 88, 104, 129, 166
Working-Memory . . . . .	130, 131
Working-Memory-Element . . . . .	126-129, 144, 147

**Z**

Zuordnungsmatrix . . . . .	12
Zuordnungsproblem . . . . .	26-28, 32