Potenzial Bayes'scher Netze zur Unterstützung der Produktionsplanung und -steuerung

Dissertation
zum Erlangen des Grades eines

Doctor rerum politicarum

an der Fakultät für Wirtschaftwissenschaften
der Technischen Universität Ilmenau

vorgelegt von
Diplom-Wirtschaftsinformatiker Torsten Munkelt

Eingereicht:	Ilmenau, den 30.06.2008
Verteidigt:	Ilmenau, den 16.12.2008
Erstgutachter:	Univ.-Prof. em. Dr.-Ing. habil. Peter Gmilkowsky
Zweitgutachter:	Univ.-Prof. Dr. rer. pol. habil. Rainer Souren

urn:nbn:de:gbv:ilm1-2009000344

Meinen Eltern

Danksagung

Die vorliegende Doktorarbeit ist vorwiegend während meiner Tätigkeit am Fachgebiet Wirtschaftsinformatik I „Anwendungssysteme für Industriebetriebe“ an der Fakultät für Wirtschaftswissenschaften der Technischen Universität Ilmenau entstanden. Herzlichen Dank an alle, die zum Gelingen der Arbeit beigetragen haben.

Meinem Doktorvater, Herrn Prof. Gmilkowsky, gilt mein besonderer Dank. Er hat mir Freiraum bei der Wahl des Themas und dem Ausgestalten der Arbeit gewährt und mich während des langen Entstehungsprozesses der Arbeit geduldig betreut.

Herrn Prof. Souren, dem Leiter des Fachgebietes Produktionswirtschaft/ Industriebetriebslehre der Technischen Universität Ilmenau, danke ich für das schnelle Erstellen des profunden Zweitgutachtens.

Bei meinen ehemaligen Kollegen, Herr Dr. Völker und Herrn Dr. Döhring, bedanke ich mich für den Wissens- und Erfahrungsaustausch, der mich und somit die vorliegende Arbeit bereichert hat. Zudem bedanke ich mich bei beiden für ungezählte Diskussionen, die immer zu einer besseren Lösung geführt haben. Herrn Dr. Völker danke ich zudem für die Durchsicht des Manuskripts und die zahlreichen, durchdachten Verbesserungsvorschläge.

Herrn Dr. Beyer vom Fachgebiet Wirtschaftsinformatik I danke ich für fachlichen Rat und moralische Unterstützung. Herrn Dr. Vogel vom Fachgebiet „Quantitative Methoden der Wirtschaftswissenschaften“ danke ich für Hinweise auf den Gebieten der Wahrscheinlichkeitslehre und der Statistik.

Frau Dipl.-Ing. Susanne Würfel verdient meinen speziellen Dank. Sie hat mich organisatorisch unterstützt, als ich nicht mehr durchgängig an der TU Ilmenau geweilt habe.

Zuletzt, aber umso mehr danke ich meinen Eltern, mit deren Unterstützung und Zuspruch ich die vorliegende Arbeit vollendet habe.

Torsten Munkelt
Hannover, im November 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Alle Abbildungen ohne explizite Quellenangabe sind eigene Darstellungen.

Tabellenverzeichnis

Alle Tabellen ohne explizite Quellenangabe sind eigene Darstellungen.

Symbolverzeichnis

A	Menge aller Aufträge
$\mathcal{A}$	Borel'sche σ-Algebra auf Ω
$\mathbb{A}$	Menge von Anweisungen
An_v	Vorfahren eines Knotens
a	Auftrag bzw. Arbeitsplan
$\vec{a}$	n-dimensionale Adresse
$\vec{a'}$	reduzierte n-dimensionale Adresse
au_{sp}	Zustand, dass der Rasensprenger sp aus ist
B	Bayes'sches Netz (BN)
$\mathbb{B}$	Menge Bayes'scher Netze
$B_{\rightarrow}$	Übergangsnetz eines DBNes
B_0	Eingangsnetz eines DBNes
B_1	erstes BN
B_2	zweites BN
B_{best}	bestes BN hinsichtlich einer Datenbasis D und eines Gütemaßes
B_U	entrolltes DBN
b	ein Auftrag
bo	Variable „Boden“
C	Clique
C_i	i-te Clique
C_j	i-te Clique
Ch_v	Menge der Kinder des Knotens v
$\vec{c}$	Vektor der Konvertierungsfaktoren für die Adressrechnung
$\vec{c'}$	Vektor der relevanten Konvertierungsfaktoren für die Adressrechnung
ch_v	ein Kind des Knotens v
c_W	wahrscheinlichste Konfiguration der Variablen $w \in W$
c_w	wahrscheinlichste Konfiguration der Variable w
$crd(v)$	Anzahl der Nachbarn eines Knotens v, die bereits numeriert worden sind

$c_{\#}$	Konfiguration eines für einen unvollständigen Datensatz eingefügten Alternativdatensatzes
D	Datenbasis
D'	aus D abgeleitete Datenbasis mit alternativen Datensätzen für Datensätze mit fehlenden Werten aus D
$\mathbb{D}$	Dynamisches Bayes'sches Netz (DBN)
$\overline{D}_A$	mittlere Durchlaufzeit der Aufträge $a \in A$
D_a	Durchlaufzeit des Auftrags a
De_u	Nachfahren eines Knotens
$\vec{d}$	Vektor der Lese- und Schreibindexdekremente
$\vec{d'}$	Vektor der relevanten Lese- und Schreibindexdekremente
d_{max}	maximal betrachtete Simulationsdauer
Δt	Abtastintervalllänge
Δw_i	Wertezuwachs, den der i-te Arbeitsgang zum Wert eines Auftrags beiträgt
E	Kantenmenge
$E_{\rightarrow}$	Kantenmenge des Übergangsnetzes eines DBNes
E_0	Kantenmenge des Eingangsnetzes eines DBNes
E_U	Kantenmenge eines entrollten DBNes
e	Evidenz
ei_{sp}	Zustand, dass der Rasensprenger sp an ist
ε	Epsilon-Umgebung
Fa_v	Familie des Knotens (der Variable) v
f	Potenzialfunktion
$f_{[0;1]}$	Funktion zum Errechnen eines Wertes aus dem Intervall $[0;1]$ aus dem Zustand einer Variable
$f(C)$	Potenzialfunktion der Clique C
fe_{bo}	Zustand, dass der Boden bo feucht ist
f_{high}	sehr hohe Abtastfrequenz
f_i	i-te Potenzialfunktion
$f_{\mathbb{N}}$	Funktion zum Errechnen des Index eines Zustandes einer Variable
f_{max}	maximale Frequenz
$f \triangleleft e$	Absorbtion der Evidenz e durch eine Potenzialfunktion f
G	ein gerichteter, azyklischer Graph
G_1	erster gerichteter, azyklischer Graph
G_2	zweiter gerichteter, azyklischer Graph
g	Potenzialfunktion
$g(n)$	Anzahl aller möglichen gerichteten, azyklischen Graphen bei n Knoten
gu_{ro}	Zustand, dass es den Rosen ro gutgeht

$H[t]$	Variable „Handlungsalternative“ für die Zeitscheibe t
h	Wahrheitswert
h_{ij}	Häufigkeit der Überganges eines Auftrages von Maschine i zu Maschine j
I	Betrachtungszeitraum $[t_s, t_e)$
I_k	Zeitraum, in dem Kapazitätseinheit k zur Verfügung steht
I_x	Indexmenge zum Ordnen der Menge der Ausprägungen einer Variable x
I_Y	Indexmenge zum Ordnen der Menge Y von Variablen
I_Z	Indexmenge zum Ordnen der Menge Z von Variablen
J	Indexmenge zum Ordnen der Menge X der Variablen
ja_{re}	Zustand, dass es regnet
K	Menge der Kapazitätseinheiten
$\mathbb{K}$	Länge der Beschreibung der Verteilung, die ein BN kodiert
k	zeitlicher Versatz bei der Autokorrelation
k_{best}	günstigster zeitlicher Versatz
L	„linke“ Hilfsmatrix beim Generieren einer Maschinenübergangsmatrix
L_k	Auslastung der Kapazitätseinheit k
l	Anzahl der Aufträge im System
$\bar{l}_w$	mittlere Warteschlangenlänge
λ	Ankunftsrate der Aufträge
$\vec{\lambda}$	Vektor der arbeitsplanspezifischen Ankunftsraten
M	Menge der Maschinen
M_i	Maschine i
M_j	Maschine j
$\mathrm{Max}_{Y\setminus X}\, f$	Maximumrand einer Potenzialfunktion f
m	Stichprobenumfang
$\vec{m}$	Vektor der maximalen Zustandsindizes
$\vec{m'}$	Vektor der relevanten maximalen Zustandsindizes
$\bar{m}_{\mathrm{abs}}$	Mittelwert der absoluten Beträge der Messwerte
$\bar{m}_{\mathrm{abs},1}$	Mittelwert der absoluten Beträge der Messwerte der ersten Messreihe
$\bar{m}_{\mathrm{abs},2}$	Mittelwert der absoluten Beträge der Messwerte der zweiten Messreihe
m_d	Anzahl der Datensätze mit $re = ne_{re}$ in einer Datenbasis
m_e	Anzahl der Datensätze mit $re = ne_{re}$, die einer Expertenschätzung entsprechen würden
m_{max}	Engpassmaschine

$msg[(C_1, C_2)]$	Nachricht zum Nachrichtenaustausch entlang der Kante zwischen den Cliquen C_1 und C_2
μ	Erwartungswert
μ_L	Erwartungswert-Parameter der logarithmischen Normalverteilung
N	Anzahl der Fälle in der Datenbasis D
Ne_v	Menge der Nachbarn des Knotens v
N_{ij}	Anzahl der Fälle in D, in denen $\pi_i = x_{\pi_i j}$
N_{ijk}	Anzahl der Fälle in D, in denen $v_i = x_{ik}$ und $\pi_i = x_{\pi_i j}$
n	$\|V\|$
na_{st}	Zustand, dass die Straße st nass ist
ne_{re}	Zustand, dass es nicht regnet
n_d	Anzahl der Datensätze mit $re = ne_{re}$ und $st = na_{st}$ in einer Datenbasis
n_e	Anzahl der Datensätze mit $re = ne_{re}$ und $st = na_{st}$, die einer Expertenschätzung entsprechen würden
ne_v	ein Nachbar des Knotens v
$num(v)$	Nummer des Knotens v
o	Organisationsgrad einer Fertigung
Ω	Menge der Elementarereignisse
P	Menge von Bewertungsfunktionen
$P_{\rightarrow}$	Menge der Bewertungsfunktionen des Übergangsnetzes eines DBNes
P_0	Menge der Bewertungsfunktionen des Eingangsnetzes eines DBNes
Pa_v	Elternmenge des Knotens v
Pa_{v_i}	Elternmenge des i-ten Knotens v_i
$\|Pa\|_{max}$	maximal zulässige Anzahl Eltern pro Knoten
Pr	Wahrscheinlichkeitsmaß auf dem Meßraum $(\Omega, \mathcal{A})$
$P[t]$	Menge der Bewertungsfunktionen der t-ten Zeitscheibe
P_U	Menge der Bewertungsfunktionen eines entrollten DBNes
p	Bewertungsfunktion
$\vec{p}$	Vektor der relativen Häufigkeiten der Arbeitspläne
pa_1	ein erstes Elter eines Knotens
pa_2	ein zweites Elter eines Knotens
pa_v	ein Elter des Knotens v
p_{ij}	Wahrscheinlichkeit der Überganges eines Auftrages von Maschine i zu Maschine j
$p_{t_{a_i}}$	Eintrittswahrscheinlichkeit der Zwischenankunftszeit t_{a_i}
π_E	Projektion von $S_{X \cup E}$ auf S_E
π_i	Pa_{v_i}
π_X	Projektion von $S_{X \cup Y}$ auf S_X

π_Y	Projektion von $S_{X\cup Y}$ auf S_Y
q	Gütemaß für eine Familie eines Knotens
q_i	$\lvert S_{\pi_i}\rvert$ bzw. $\lvert S_{Pa_{v_i}}\rvert$
R	„rechte“ Hilfsmatrix beim Generieren einer Maschinenübergangsmatrix
R_W	Randwahrscheinlichkeitsverteilungen der Variablen $w \in W$
R_w	Randwahrscheinlichkeitsverteilung der Variable w
$\mathbb{R}^{\geq 0}$	Menge der nicht-negativen reellen Zahlen
r	ein Skalar
re	Variable „Regen“
r_i	$\lvert S_{v_i}\rvert$
$\bar{r}\langle k\rangle$	mittlere Autokorrelation mehrerer Variablen mit dem Versatz k
$r_v\langle k\rangle$	Autokorrelation einer Variable v mit dem Versatz k
ro	Variable „Rosen“
S	Menge der Zustände einer Variable
$S_1[t]$	erste intermediäre Variable für die Zeitscheibe t
$S_2[t]$	zweite intermediäre Variable für die Zeitscheibe t
S_a	Menge der Zustände einer Variable a
S_b	Menge der Zustände einer Variable b
S_c	Menge der Zustände einer Variable c
S_E	Zustandsraum der evidenten Variablen
S_i	Menge aller Zustände der i-ten Variable
S_j	Menge aller Zustände der j-ten Variable
S_k	Menge aller Zustände der k-ten Variable
S_{Pa_v}	Zustandsraum der Eltern der Variable v
$S_{Pa_{v_i}}$	Zustandsraum der Eltern der i-ten Variable v_i
S_{π_i}	Zustandsraum der Eltern der i-ten Variable
S_U	Zustandsraum der Variablen $u \in U$
S_u	Menge der Zustände einer Variable u
$S_{V[t]}$	Zustandsraum der Variablen $v \in V[t]$
S_v	Menge der Zustände einer Variable v
S_{v_i}	Menge der Zustände der i-ten Variable v_i
S_X	Zustandsraum der Variablen $x \in X$
$S_{X\cup E}$	Zustandsraum der Variablen $x \in X \cup E$
$S_{X\cup Y}$	Zustandsraum der Variablen $x \in X \cup Y$
S_Y	Zustandsraum der Variablen $x \in Y$
S_Z	Zustandsraum der Variablen $x \in Z$
s	Zustand bzw. Konfiguration einer Variable
$\bar{s}$	zu s komplementärer Zustand einer binären Variable
s_0	erster Zustand einer Variable

s_a	Zustand einer binären Variable a
$\bar{s}_a$	komplementärer Zustand einer binären Variable a
s_b	Zustand einer binären Variable b
$\bar{s}_b$	komplementärer Zustand einer binären Variable b
s_c	Zustand einer binären Variable c
$\bar{s}_c$	komplementärer Zustand einer binären Variable c
sc_{ro}	Zustand, dass es den Rosen ro schlecht geht
s_i	i-ter Zustand einer Variable
$\bar{s}_i$	zu s_i komplementärer Zustand einer binären Variable
s_i^k	Zustand der Variable v_i zum Zeitpunkt k
s_{Pa}	Zustand (Konfiguration) der Eltern einer Variable
s_{Pa_v}	Zustand (Konfiguration) der Eltern der Variable v
sp	Variable „Rasensprenger“
st	Variable „Straße“
s_t	Zustand einer Variable zum Zeitpunkt t
s_U	Zustand (Konfiguration) der Variablen $v \in U$
s_v	Zustand der Variable v
s_X	Zustand (Konfiguration) der Variablen $x \in X$
s_Y	Zustand (Konfiguration) der Variablen $x \in Y$
$\sum_{Y \setminus X} f$	Summenrand einer Potenzialfunktion f
σ	Standardabweichung
σ_L	Standardabweichungs-Parameter der logarithmischen Normalverteilung
T	Indexmenge der Zeit
T_A	kumulierte Termintreue der Aufträge $a \in A$
T_a	Termintreue des Auftrags a
$\vec{T_b}$	Matrix der Bearbeitungszeiten der Arbeitspläne auf den Maschinen
t	Zeitscheibenindex
t_a	Zwischenankunftszeit der Aufträge
$\overline{t_a}$	mittlere Zwischenankunftszeit der Aufträge
t_{a_i}	i-te Zwischenankunftszeit einer stufenförmigen Zwischenankunftszeitverteilung
$t_{a,*}$	ein Termin des Auftrags a zum Zeitpunkt $*$
t_b	Bearbeitungszeit der Aufträge
$\bar{t}_b$	mittlere Bearbeitungszeit der Aufträge
t_{b_i}	Bearbeitungszeit des i-ten Arbeitsganges eines Auftrags
$t_{b,\vec{m}_{max}}$	m_{max}-ter Spaltenvektor der Matrix $\vec{T_b}$
$t_{b,n,m}$	Bearbeitungszeit eines Auftrags mit dem Arbeitsplan n bei Maschine m
t_{cause}	Zeitpunkt, zu dem die Ursache geschieht

t_{effect}	Zeitpunkt, zu dem die Wirkung eintritt
$\bar{t}_d$	mittlere Durchlaufzeit der Aufträge
t_e	Endtermin des Betrachtungszeitraumes I
t_k	ein Zeitpunkt
t_{now}	aktueller Zeitpunkt
t_r	Fertigungsrestzeit eines Auftrags
tr_{bo}	Zustand, dass der Boden *bo* trocken ist
tr_{st}	Zustand, dass die Straße *st* trocken ist
t_s	Starttermin des Betrachtungszeitraumes I
$\bar{t}_w$	mittlere Wartezeit der Aufträge
U	Menge von Variablen, (mehrdimensionale) Zufallsvariable
u	Variable, (eindimensionale) Zufallsvariable
$\vec{u_b}$	Vektor der mittleren Auslastungen der Maschinen
$u_{b,m}$	mittlere Auslastung der Maschine m
$u_{b,m_{max}}$	mittlere Auslastung der Engpassmaschine m_{max}
u_i	i-te Variable bzw. (eindimensionale) Zufallsvariable
$u[t]$	Variable, (eindimensionale) Zufallsvariable in Zeitscheibe t
V	Menge von Variablen, (mehrdimensionale) Zufallsvariable
V'	Menge der Variablen, die nicht auf einen ihrer Zustände fixiert worden sind
$V_{\rightarrow}$	Knotenmenge des Übergangsnetzes eines DBNes
V_0	Knotenmenge des Eingangsnetzes eines DBNes
V_A	kumulierte Verspätung der Aufträge $a \in A$
V_a	Verspätung des Auftrags a
V_{pot}	potenziell relevante Variablen
V_{rel}	relevante Variablen
V_T	stochastischer Prozess
V_t	Menge von Zufallsvariablen zum Zeitpunkt t
$V[t]$	Menge von Zufallsvariablen eines DBNes in Zeitscheibe t
V_U	Knotenmenge eines entrollten DBNes
v	Variable, (eindimensionale) Zufallsvariable
$\vec{v}$	Repräsentation der Potenzialfunktionswerte als Vektor
v_i	i-te (Zufalls)variable
$v[t]$	Variable, (eindimensionale) Zufallsvariable in Zeitscheibe t
W	Menge von Variablen, (mehrdimensionale) Zufallsvariable
W_A	Wert aller Aufträge $a \in A$
W_a	Wert des Auftrags a
w	Variable, (eindimensionale) Zufallsvariable
$w_{a,t}$	Fortschritt bzw. Wert des Auftrags a zu einem Termin t
w_s	Wert eines Auftrags zu seinem Start

$w[t]$	Variable, (eindimensionale) Zufallsvariable in Zeitscheibe t
$w_{\#}$	Wichtung eines für einen unvollständigen Datensatz eingefügten Alternativdatensatzes
X	Menge von Zustandsvariablen
$\|X\|$	Mächtigkeit (Anzahl der Elemente) einer Menge (hier z. B. X)
x	Zustandsvariable
x_i	i-te Zustandsvariable
x_{ik}	k-ter Zustand der i-ten Variable
x_{max}	maximaler Wert einer eindimensionalen Stichprobe
x_{min}	minimaler Wert einer eindimensionalen Stichprobe
$x_{\pi_i j}$	j-te Konfiguration (j-ter Zustand) der Eltern π_i der i-ten Variable
Y	Menge von Zustandsvariablen
Z	Menge von Zustandsvariablen
Z_k	Dauer, die Kapazitätseinheit k produziert
$Z[t]$	Variable „Ziel“ für die Zeitscheibe t
$\vec{z}$	Vektor der Zeiger auf Indizes von Zuständen, auf welche die Variablen fixiert worden sind
z_k	Anzahl der Zeiger, die bei der Repräsentation einer Potenzialfunktion als Baumfunktion zusätzlich benötigt werden
z_{max}	maximale Anzahl zu generierender Aufträge

Abkürzungsverzeichnis

ACO	Ant Colony Optimization
AG	gerichteter, azyklischer Graph
ARFF	Attribute-Relation File Format
BDE	Betriebsdatenerfassung
BN	Bayes'sches Netz
CG	Cliquengraph
CPM	Critical Path Method
CP	Constraint Programming
CPU	Central Processing Unit, Hauptprozessor
CSP	Constraint Satisfaction Problem
d	Tag
DAG	Directed Acyclic Graph (gerichteter, azyklischer Graph)
DBN	Dynamisches Bayes'sches Netz
DLL	Dynamic Link Library
DM	Data Mining
DIN	Deutsche Industrienorm
DOM	Document Object Model
DTD	Document Type Definition
DW	Data-Warehouse
e. V.	eingetragener Verein
EA	evolutionärer Algorithmus
EOS	ereignisorientierte Simulation
ES	Evolutionsstrategie
FCFS	First-Come, First-Serve(d)
FFT	Frühester Fertigstellungstermin
FRS	Fuzzy-Regelsystem
GA	genetischer Algorithmus
GB	Gigabyte
GE	Geldeinheit
GG	gerichteter Graph
ggf.	gegebenenfalls

GUI	Graphical User Interface, grafische Schnittstelle zum Benutzer
h	Stunde
IM	Inferenzmaschine
KDD	Knowledge Discovery in Databases
KNN	künstliches neuronales Netz
KOZ	Kürzeste Operationszeit
KRBZ	Kürzeste Restbearbeitungszeit
KSZ	Kürzeste Schlupfzeit
LRBZ	Längste Restbearbeitungszeit
MAS	Multi-Agenten-System
MB	Megabyte
MDE	Maschinendatenerfassung
MDI	Multiple Document Interface
MDL	Minimum Description Length (minimale Beschreibungslänge)
MH	Metaheuristik
MHz	Megahertz
min	Minute
MRP II	Manufacturing Resource Planning
NP	nichtdeterministisch-polynomial
o. B. d. A.	ohne Beschränkung der Allgemeinheit
PAP	Programmablaufplan
PC	Personalcomputer
PI	Probabilistische Inferenz
PP	Planungsperiode
PPS	Produktionsplanung und -steuerung
R&R	Ruin&Recreate
RIP	Eigenschaft des fortlaufenden Schnittes, Running Intersection Property
RT	retrograde Terminierung
s	Sekunde
SCM	Supply Chain Management
SLX	Simulation Language with eXtensibilities
TU	Technische Universität
UG	ungerichteter Graph
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieure
XML	Extensible Markup Language
XPS	Expertensystem
ZE	Zeiteinheit

Kapitel 1

Einleitung

1.1 Motivation zur Arbeit

Die Produktionsplanung und -steuerung (PPS) wartet mit einer Reihe ungelöster beziehungsweise nicht exakt oder zumindest nicht zur Zufriedenheit der Anwender gelöster Probleme auf. Zu diesen Problemen zählen unter anderem:

- die Primärbedarfsprognose im Rahmen der Primärbedarfsplanung,
- die verbrauchsgesteuerte Materialbedarfsermittlung im Rahmen der Mengenplanung,
- die mehrstufige Bestellmengen-, Auftrags- beziehungsweise Losgrößenplanung ebenfalls im Kontext der Mengenplanung,
- die simultane Termin- und Kapazitätsplanung,
- die Auftragsfreigabe als Bindeglied zwischen Produktionsplanung und Produktionssteuerung
- und nicht zuletzt die Reihenfolgeplanung im Rahmen der Fertigungssteuerung.

Die Betriebswirtschaftslehre teilt die Teilaufgaben der PPS in schlechtstrukturierte und wohlstrukturierte Probleme ein. Die wichtigsten schlechtstrukturierten Probleme der PPS sind bereits oben angeführt worden. Schlechtstrukturiert sei ein Problem immer dann, wenn ihm mindestens einer der folgenden Strukturdefekte anhaftet:

- ein Bewertungsdefekt,

- ein Wirkungsdefekt,
- ein Zielsetzungsdefekt
- oder ein Lösungsdefekt.

Verursacht werden die aufgeführten Strukturdefekte durch stochastische und informationelle Unsicherheit in der Produktion und der PPS. Hervorgerufen werden die Strukturdefekte auch dadurch, dass es sich bei der Produktion in der Regel um einen stochastischen Prozess handelt, der schwer zu regeln ist. Die Komplexität der Produktion und der PPS stellt eine weitere bedeutende Ursache für die aufgezählten Strukturdefekte dar. Die Komplexität äußert sich zum Beispiel in der Vielzahl der an der Produktion und der PPS beteiligten Entitäten, die zur Lösung der schlechtstrukturierten Probleme der PPS mannigfaltig zueinander in Beziehung gesetzt werden können. Insbesondere die (ganzzahligen) kombinatorischen Optimierungsprobleme, von denen die PPS einige aufweist, leiden unter der kombinatorischen Explosion ihres jeweiligen Lösungsraumes. Zur Lösung vieler dieser Probleme, der NP-schweren Probleme, existieren (derzeit) keine Algorithmen, welche die Probleme auf einer deterministischen Maschine mit einem in Abhängigkeit von der Problemgröße polynomialen Rechenaufwand global optimal lösen.

In den vergangenen Jahrzehnten – spätestens seit dem rasanten Aufschwung und verbreiteten Einsatz der Rechentechnik in den siebziger Jahren des vergangenen Jahrhunderts – wurden viele Versuche unternommen, die den Problemen der PPS anhaftenden Strukturdefekte zu beheben und mittels neuer Verfahren und Modelle der künstlichen Intelligenz und des Softcomputing die korrespondierenden schlechtstrukturierten Probleme der PPS exakt zu lösen. Die meisten der untersuchten Verfahren lösen die schlechtstrukturierten Probleme der PPS besser oder schneller als klassische Verfahren und Heuristiken des Operations Research. Besondere Erwähnung verdienen an dieser Stelle evolutionäre Algorithmen (EA), künstliche neuronale Netze (KNN) und nicht zuletzt wissensbasierte Systeme (WBS). Allerdings weisen auch diese Verfahren etliche Nachteile auf, und sie sind noch weit davon entfernt, die den Problemen der PPS anhaftenden Strukturdefekte auszumerzen und die korrespondierenden schlechtstrukturierten Probleme der PPS exakt zu lösen. Als Nachteile seien an dieser Stelle nur die folgenden genannt. Einige dieser Verfahren und Modelle

1. erlauben nicht, sowohl Wissen aus Daten als auch Expertenwissen zu integrieren,
2. vernachlässigen die verschiedenen Formen der Unsicherheit,

3. bilden keine Kausalität ab, sondern suchen nahezu blind,
4. verursachen hohen Rechenaufwand (durch iterative Optimierung),
5. sind weder in der Lage, ihre Lösungswege noch ihre Lösung zu erklären.

Verfahren und Modelle der Wissensverarbeitung im Kontext Bayes'scher Netze (BNe) weisen viele dieser Nachteile nicht auf und warten darüber hinaus mit Eigenschaften und Fähigkeiten auf, die sie dazu prädestinieren, Strukturdefekte zu beheben und schlechtstrukturierte Probleme der PPS zu lösen. Einige der wichtigsten Eigenschaften und Fähigkeiten BNe sind die folgenden. BNe

1. modellieren komplexe stochastische Zusammenhänge zwischen Variablen,
2. entdecken Kausalität zwischen Variablen,
3. verringern den Rechenaufwand durch „gemischtes" Schließen,
4. können sowohl aus Daten erstellt als auch aufgrund von Expertenwissen aufgebaut werden,
5. akzeptieren unvollständige Datensätze sowohl bei ihrer Anwendung als auch bei ihrer Erstellung aus Daten,
6. entdecken „versteckte" Variablen, die bisher noch nicht zur Modellierung herangezogen worden sind, und
7. sind in ihrem Verhalten und ihren Lösungen interpretierbar.

Dynamische BNe (DBNe), eine Sonderform allgemeiner BNe, bilden sogar komplexe stochastische Prozesse ab. Die meisten der aufgeführten Eigenschaften und Fähigkeiten sprechen für sich selbst. Lediglich Punkt 3 bedarf einer näheren Erläuterung: Bei iterativen Optimierungsverfahren, zu denen auch die simulationsbasierte Optimierung zählt, werden iterativ erstens Handlungsalternativen erzeugt respektive verändert, und zweitens wird prognostiziert, wie sich die Handlungsalternativen auf die verfolgten Ziele auswirken. Sind die verfolgten Ziele erreicht worden, steht keine Zeit mehr zur Verfügung oder tritt eine andere Abbruchbedingung auf, wird die bisher beste Handlungsalternative umgesetzt, ansonsten beginnt der Kreislauf von neuem. Im Gegensatz dazu können bei BNen die Ziele auf die erwünschten Ausprägungen gesetzt, und es kann entgegen der kausalen Kette auf die

Handlungsalternative geschlossen bzw. die Handlungsalternative „diagnostiziert“ werden, die mit höchster Wahrscheinlichkeit die gesteckten Ziele erreicht. Es werden also *nicht* mehrfach Handlungsalternativen generiert, und es wird *nicht* mehrfach prognostiziert, was jeweils sehr aufwendig wäre, sondern lediglich einmal „gemischt geschlossen“, was mit einer erheblichen Reduktion des Rechenaufwandes einhergeht. Die Fähigkeit BNe zum gemischten Schließen[1] wird im Rahmen der Arbeit als eine der bedeutendsten erachtet.

Durch ihre Eigenschaften und Fähigkeiten wirken BNe den Strukturdefekten in der PPS entgegen – auch, indem sie ihre Ursachen angreifen –, sie besitzen nicht die Nachteile anderer Verfahren und weisen somit ein Potenzial zur Unterstützung der PPS auf.

1.2 Ziele der Arbeit

Im Rahmen der vorliegenden Arbeit soll das Potenzial BNe zur Unterstützung der Produktionsplanung und -steuerung identifiziert, untersucht und aufgezeigt werden. Das Potenzial BNe zur PPS ist sowohl theoretisch als auch praktisch zu bestimmen. Im theoretisch geprägten Teil der Arbeit ist zu bestimmen, welche Eigenschaften und Fähigkeiten BNe besonders gut mit den Problemen der PPS korrespondieren, das heißt, zur Lösung welcher schlechtstrukturierten Probleme der PPS sich BNe besonders gut eignen. Im praktisch orientierten Teil der Arbeit, den im wesentlichen eine empirische Studie ausmacht, sind die theoretischen Erkenntnisse empirisch zu untermauern: Es ist ein geeignetes Problem der PPS auszuwählen, zu konkretisieren und mittels BNe zu lösen. Das Problem soll nicht eine einzige Probleminstanz repräsentieren, sondern eher eine Problemklasse, die verschiedene gleichartige Probleminstanzen hervorbringt, auf welche dann verschieden parametrisierte BNe angewandt werden. So sollen sich für ein Problem der PPS die betrachteten Produktionssysteme z. B. hinsichtlich ihrer Größe unterscheiden, hinsichtlich ihrer Auslastung und hinsichtlich ihres Organisationsgrades. Nach der Auswertung und Interpretation der Ergebnisse der Experimente soll die Arbeit in der Aussage gipfeln, ob BNe ein Potenzial zur Unterstützung der PPS aufweisen und welche BNe sich besonders gut zu Lösung welcher Instanzen von Problemen der PPS eignen.

[1] Hier und in Punkt 3 ist das Schließen als gemischtes Schließen bezeichnet worden. Dieser Umstand resultiert daraus, dass kein reines diagnostisches Schließen von den Zielen auf die Handlungsalternativen erfolgt, sondern auch die (derzeitigen) Zustandsgrößen interkausal auf Handlungsalternativen bzw. die Stellgrößen Einfluss nehmen.

1.3 Vorgehensweise und Aufbau der Arbeit

Die vorliegende Arbeit kann nicht das Potenzial BNe zur Lösung jeder einzelnen Teilaufgabe der PPS im Detail untersuchen. Deshalb arbeitet sie Gemeinsamkeiten der Teilaufgaben der PPS, wie z. B. Unsicherheit, Komplexität und Strukturdefekte, heraus und abstrahiert allgemeine Aufgaben von den Teilaufgaben der PPS. Im Anschluss ermittelt sie das Potenzial BNe zur Lösung dieser allgemeinen Aufgaben, wie z. B. der schlechtstrukturierten Probleme. Alsdann zieht sie folgenden Schluss: Wenn BNe ein Potenzial zur Lösung der allgemeinen Aufgaben aufweisen, so sollten sie auch das Potenzial besitzen, Teilaufgaben der PPS zu lösen und somit die PPS zu unterstützen.

Der Aufbau der Arbeit orientiert sich grob an der eben geschilderten Vorgehensweise: Kapitel 1, die Einleitung, motiviert zur vorliegenden Arbeit, steckt die Ziele der Arbeit ab und legt die Vorgehensweise im Rahmen der Arbeit und den Aufbau der Arbeit dar.

Kapitel 2 geht auf die Aufgaben der PPS ein: Es beschreibt in Abschnitt 2.1 die Ziele und Nebenbedingungen der PPS, gliedert in Abschnitt 2.2 die PPS in ihre Teilaufgaben, ordnet diese Teilaufgaben an und stellt ihr Zusammenwirken dar. Nachdem die Aufgaben der PPS beschrieben worden sind, wendet sich Kapitel 2 den Problemen der PPS zu: Abschnitt 2.3 widmet sich der Unsicherheit in Produktion und PPS: Er unterscheidet zwischen informationeller und stochastischer Unsicherheit im Kontext von Produktion und PPS. Auf der Beschreibung der stochastischen Unsicherheit aufbauend, stellt Abschnitt 2.4 die Produktion als komplexen stochastischen Prozess dar. Abschnitt 2.5 geht auf die Komplexität der Aufgaben der PPS ein: Er unterteilt die Aufgaben der PPS in mit polynomialem Rechenaufwand lösbare Aufgaben und NP-schwere Probleme der PPS. Abschnitt 2.6 stellt die Eigenschaften und Probleme der Produktion und der PPS, welche in den Abschnitten 2.3 bis 2.5 beschrieben worden sind, zusammen, identifiziert ihre jeweiligen Strukturdefekte und teilt sie aufgrund der identifizierten Strukturdefekte in wohlstrukturierte und schlechtsturkturierte Probleme der PPS ein. Bei den in Abschnitt 2.6 identifizierten schlechtstrukturierten Problemen handelt es sich um diejenigen, die daraufhin untersucht werden sollten, inwieweit BNe zu ihrer Lösung eingesetzt werden bzw. beitragen könnten. Der Einsatz BNe zur Unterstützung wohlstrukturierter Aufgaben der PPS bietet sich nicht an, weil sie bereits mittels anderer – und vermutlich einfacherer – Verfahren effizient exakt gelöst werden.

Nachdem in Kapitel 2 die schlechtstrukturierten Probleme der PPS identifiziert worden sind, beantwortet Kapitel 3 die Frage, wie gut sich zu BNen alternative Verfahren eignen, besagte schlechtstrukturierte Probleme der PPS zu lösen. Es stellt den aktuellen Stand auf dem Gebiet der Unterstützung der

PPS durch alternative Verfahren dar. Es analysiert unter anderem evolutionäre Algorithmen, künstliche neuronale Netze und Fuzzy-Ansätze, die drei Hauptströmungen des Softcomputing. Abschnitt 3.7 fasst zusammen, wie gut sich alternative Verfahren zur Unterstützung der PPS eignen, indem es die Verfahren bezüglich ihrer Vorzüge und Nachteile einander gegenüberstellt.

Kapitel 4 stellt BNe als ein alternatives Verfahren zu Unterstützung der PPS vor. BNe stellen kein Verfahren im eigentlichen Sinne dar, sondern sind eine Form der Wissensverarbeitung, die sich in Wissensrepräsentation, -akquisition und -anwendung unterteilt. BNe im engeren Sinne repräsentieren Wissen, und es sind Vorgehensweisen, Algorithmen und Datenstrukturen bekannt, mittels derer man BNe akquirieren und anwenden kann. Abschnitt 4.1 stellt BNe im engeren Sinne als Wissensrepräsentationsform vor, Abschnitt 4.2 legt dar, wie BNe manuell, automatisch oder kombiniert akquiriert werden, und Abschnitt 4.3 geht auf die Anwendung BNe mittels exakter und probabilistischer Inferenz ein. Die Darstellung der Wissensverarbeitung im Kontext BNe erfolgt anhand eines durchgängigen Beispieles.

Abschnitt 4.4 geht auf die Datenvor- und -nachverarbeitung bei der Akquisition und Anwendung BNe ein, die gerade im Rahmen der PPS von Bedeutung ist.

Der folgende Abschnitt 4.5 führt die Eigenschaften und Fähigkeiten BNe auf, welche sie vor anderen Modellen und Verfahren auszeichnen und zur PPS prädestinieren, wie z. B. das Identifizieren und Quantifizieren kausaler stochastischer Zusammenhänge, das Verringern des Rechenaufwandes durch gemischtes Schließen, die Akquisition BNe aus einer Kombination von Expertenwissen über die PPS und von Wissen aus PPS-Daten, das Verarbeiten von Datensätzen mit fehlenden Werten, das Entdecken „versteckter Variablen“, die Interpretierbarkeit und die Plausibilität.

Eine kurze Darstellung der Gebiete, auf denen BNe bisher erfolgreich eingesetzt worden sind, in Abschnitt 4.6 rundet das Kapitel 4 ab.

Nachdem BNe ausführlich vorgestellt worden sind, erfolgt in Kapitel 5 die theoretische Betrachtung ihres Potenzials zur Unterstützung der PPS. Wie bereits erwähnt, sind Produktion und PPS besonders durch informationelle und stochastische Unsicherheit geprägt. Abschnitt 5.1 stellt dar, wie BNe diese Formen der Unsicherheit abbilden und aktiv dazu nutzen, die PPS zu unterstützen.

Wie bereits festgestellt, handelt es sich bei der Produktion in der Regel um einen komplexen stochastischen Prozess. Wie BNe – oder besser DBNe – diesen Prozess abbilden und wie er mit ihrer Hilfe prognostiziert, diagnostiziert und geregelt werden kann, beantwortet Abschnitt 5.2. Der folgende Abschnitt 5.3 legt dar, wie BNe die Komplexität handhaben und eindämmen, die der Produktion und der PPS innewohnt.

Die Strukturdefekte der PPS erwachsen aus der Unsicherheit, dem Prozesscharakter und der Komplexität der Produktion. Wie nun BNe den einzelnen Strukturdefekten in der PPS entgegenwirken und sie zum Teil beheben, beschreibt Abschnitt 5.4. Abschnitt 5.5 beschließt das Kapitel 5 mit einer zusammenfassenden Beurteilung des Potenzials BNe zur Unterstützung der PPS.

Kapitel 6 stellt das Softwaresytem vor, welches zur Unterstützung der PPS mittels BNe entworfen und implementiert worden ist. Abschnitt 6.1 legt dar, warum ein eigenes Softwaresystem erstellt worden ist, und Abschnitt 6.2 gibt einen Überblick über die Architektur und die Funktionalität des Softwaresystems. Der folgende Abschnitt 6.3 baut ein Vorgehensmodell auf, welches beschreibt, wie das Softwaresystem angewandt wird, um die PPS mittels BNe zu unterstützen. Der folgende Abschnitt 6.4 beschreibt die Komponenten zur Akquisition, Repräsentation und Anwendung BNe, die den Kern des Softwaresystems repräsentieren, im Detail. Abschnitt 6.5 geht auf die PPS-spezifischen Komponenten des Softwaresystems ein: die Komponente zum Management der PPS-Daten, die Komponente zur Generierung der PPS-Daten und die Komponente zur datengetriebenen Simulation der Produktion. Abschnitt 6.6 macht mit den wichtigsten weiteren Komponenten des Softwaresystems bekannt. Eine dieser Komponenten ist die graphische Benutzeroberfläche, die für die manuelle Akquisition des BNes aus Expertenwissen, das manuelle Editieren des BNes und das Interpretieren des BNes und seiner Arbeitsweise unerlässlich ist. Eine weitere Komponente dient dem Management von Experimenten: Sie erlaubt es, multifaktorielle Experimente zu definieren und automatisch durchzuführen. Die letzte Komponente, bei der es sich eher um eine Komponentensammlung handelt, zielt auf den Test des Softwaresystems ab. Kapitel 6 endet in Abschnitt 6.7 mit einem Ausblick darauf, welche potenziellen Einsatzgebiete dem im Rahmen der Arbeit und zur PPS entwickelten Softwaresystem noch offenstehen.

Das Potenzial BNe zur Unterstützung der PPS, das ihm Rahmen der Arbeit bisher nur theoretisch beleuchtet worden ist, soll das nun folgende Kapitel 7 empirisch belegen. Nachdem in Abschnitt 7.1 die Ziele der empirischen Untersuchung vorgestellt worden sind, charakterisiert Abschnitt 7.2 die situationsabhängige Auswahl von Prioritätsregeln mittels DBNe zur Reihenfolgeplanung als das im Rahmen der empirischen Untersuchung zu lösende Problem der PPS, und Abschnitt 7.2 geht ebenfalls darauf ein, wie das Problem prinzipiell mittels BNe gelöst werden soll. Abschnitt 7.3 beschreibt die konkrete, im Rahmen der empirischen Untersuchung betrachtete Aufgabe der PPS. Anschließend geht Abschnitt 7.4 darauf ein, welche Methoden zur Wissensakquisition, -repräsentation und -anwendung im Kontext BNe zur Lösung der im Rahmen der empirischen Untersuchung gestellten Aufgabe

ausgewählt und wie sie parametrisiert werden. Abschnitt 7.5 fasst schließlich die Ergebnisse der empirischen Untersuchung, bezogen auf die Güte der Reihenfolgeplanung, zusammen. Er zeigt z. B. auf, wie sich der Organisationstyp bzw. -grad der Produktion auf die Prognose und Diagnose mittels DBNe auswirkt und wie die Güte der Reihenfolgeplanung mittels DBNe von der Auslastung des Produktionssystems abhängt. Abschnitt 7.6 befasst sich mit dem Rechenaufwand, den DBNe verursachen. Er stellt u. a. dar, wie hoch der fixe und der variable Anteil an Rechenzeit und Speicherbedarf für verschieden ausgestaltete DBNe bei der Reihenfolgeplanung ausfällt. Kapitel 7 schließt mit Abschnitt 7.7, der die Ergebnisse der empirischen Untersuchung hinsichtlich der mit ihr verfolgten Ziele, die in Abschnitt 7.1 gesteckt worden sind, zusammenfasst.

Die Arbeit endet in Kapitel 8 mit einer Zusammenfassung der aus der Arbeit gewonnenen Erkenntnisse und einem Ausblick auf zukünftige Möglichkeiten, BNe zur Unterstützung der PPS in der Praxis einzusetzen, und auf weiterhin empirisch und in der Praxis zu untersuchende Möglichkeiten, die PPS mittels BNe zu unterstützen.

Es wird empfohlen, die Kapitel der Arbeit in der in diesem Abschnitt und im Inhaltsverzeichnis angegebenen Reihenfolge zu lesen, da das Verständnis aller vorausgegangenen Kapitel für das Verständnis des aktuellen und der darauffolgenden vonnöten ist.

Kapitel 2

Aufgaben und Probleme der Produktionsplanung und -steuerung

2.1 Ziele und Nebenbedingungen der PPS

Der Produktionsplanung und -steuerung (PPS) obliegt es, die Produktion so zu planen und zu regeln[1], dass gesteckte Ziele erreicht und vorliegende Nebenbedingungen eingehalten werden. Bei den Zielen der PPS handelt es sich idealerweise um globale Ziele oder Zielfunktionen aus dem Controlling. Es genügt im Allgemeinen nicht, ausschließlich die Kosten zu minimieren, weil dann die Produktion unter Umständen nicht aufrechterhalten werden kann und nicht zwangsläufig rentabel produziert. Als Oberziel bietet sich deshalb eher der Return on Investment (ROI) [Hor02, S. 571 ff.], [Zie98, S. 33 f.] an, der die Rentabilität einbezieht und dessen Anwendung bestandserhaltend wirkt. Nichtsdestotrotz stellen die Kosten das Bindeglied zwischen der Produktion und den globalen Zielen des Controlling dar. Leider gelingt es oft nicht, die Variablen, welche die Produktion beschreiben, auf die Kosten abzubilden bzw. besagte Variablen so zu erfassen, dass sie auf die Kosten abgebildet werden können. [Kur03, S. 20], [Ada98, S. 548] Aus diesem Grunde werden in der PPS oft vier „technizitäre Ersatzziele“ verwandt:

1. optimale Termintreue,
2. minimale Durchlaufzeiten,

[1] Bei der Produktions- oder auch Fertigungs*steuerung* handelt es sich – entgegen ihrem Namen – nicht um eine Steuerung bzw. eine Steuerstrecke, sondern um eine Regelung bzw. einen Regelkreis.

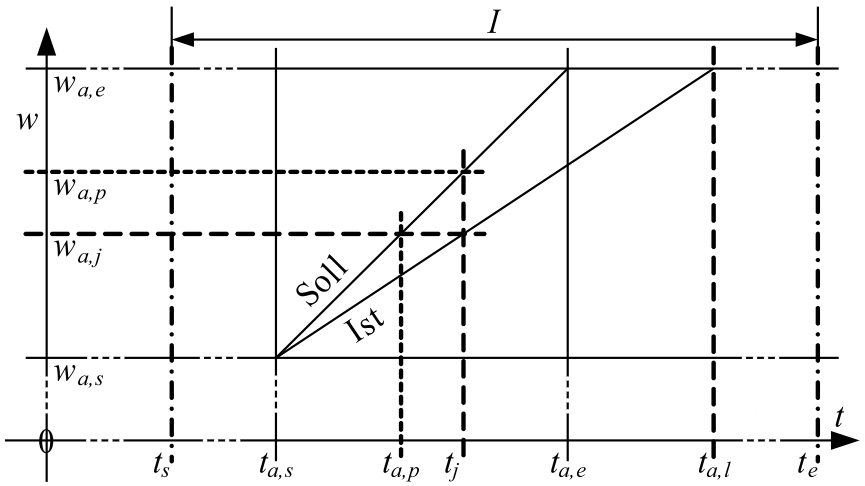

Abbildung 2.1: Vereinfachter geplanter Soll- und tatsächlicher Ist-Verlauf eines Auftrages $a \in A$ bezüglich seines Fortschritts w über der Zeit t

3. minimaler Lagerbestand und

4. maximale Kapazitätsauslastung.

Um die technizitären Ersatzziele näher zu erläutern und sie zu operationalisieren, werden einige Bezeichner eingeführt: Es sei A die Menge aller *A*ufträge, und $a \in A$ sei ein *A*uftrag. Es seien $t_{a,s}$ der *S*tarttermin, zu dem der Auftrag a eingeht, $t_{a,e}$ der *geplante E*ndtermin des Auftrags a und $t_{a,l}$ der *tatsächliche* Endtermin, zu dem der Auftrag a ausge*l*iefert wird. Zum Zeitpunkt $t_{a,s}$ weise der Auftrag a einen Fortschritt von $w_{a,s}$ auf, und zum Zeitpunkt $t_{a,l}$ einen Fortschritt von $w_{a,e}$. Abbildung 2.1 stellt die Größen in einem Diagramm dar.[2] Sie zeigt vereinfacht den geplanten Soll-Verlauf des Auftrages $a \in A$ und seinen tatsächlichen Ist-Verlauf bezüglich des Auftragsfortschritts w über der Zeit t. Zum Zeitpunkt t_j weist der Auftrag a einen tatsächlichen Fortschritt von $w_{a,j}$ auf und sollte laut Plan einen Fortschritt von $w_{a,p}$ aufweisen. Den Fortschritt $w_{a,j}$, den der Auftrag a zum Zeitpunkt t_j tatsächlich aufweist, sollte er laut Plan bereits zum Zeitpunkt $t_{a,p}$ aufweisen. Zudem sei $I = [t_s, t_e)$ ein rechtsoffenes Zeitintervall.

Das erste und derzeit wohl wichtigste technizitäre Ersatzziel am vom Kunden dominierten Markt ist die *optimale Termintreue*. [Kur03, S. 21 f.], [Sch95, S. 116.] In ihrem Rahmen wird angestrebt, Aufträge genau zum vereinbarten

[2] Um die Darstellung nicht zu verkomplizieren, wird vereinfachend davon ausgegangen, dass sich Soll- und Ist-Start des Fertigungsauftrages gleichen und dem Freigabezeitpunkt dieses Auftrages entsprechen. Aus demselben Grunde werden Soll- und Plantermin gleichgesetzt, die ebenfalls noch unterschieden werden könnten.

Liefertermin fertigzustellen. Werden alle Aufträge rechtzeitig fertiggestellt, entfallen zum einen Konventionalstrafen, und zum anderen bleibt das Ansehen des Unternehmens erhalten oder steigt, und weitere Aufträge folgen. Werden die Aufträge nicht früher als geplant fertiggestellt, verringert sich zusätzlich die Kapitalbindung. Optimale Termintreue senkt ergo die Kosten und führt langfristig zu höherem Umsatz.

Die Verspätung V_a eines Auftrags $a \in A$ zu einem Zeitpunkt t_j ist $t_{a,p} - t_j$. Die über den Aufträgen A kumulierte absolute Verspätung V_A zum Zeitpunkt t_j ist

$$V_A = \sum_{a \in B} |V_a| \quad \text{mit} \quad B = \{b \in A | (t_{b,s} \leq t_j) \wedge (w_{b,j} < w_{b,e})\}, \tag{2.1}$$

wobei die beiden UND-verknüpften Bedingungen für den Auftrag $b \in A$ aussagen, dass der Auftrag b bereits eingegangen sein muss und noch nicht fertiggestellt bzw. noch nicht ausgeliefert worden sein darf. Im Kontext der optimalen Termintreue ist V_A zu minimieren: $V_A \rightarrow \min$.

Die Termintreue T_a eines Auftrages $a \in A$ ist $t_{a,e} - t_{a,l}$.[3] Die über den Aufträgen A kumulierte absolute Termintreue T_A im Zeitintervall $I = [t_s, t_e)$ lässt sich wie folgt bestimmen:

$$T_A = \sum_{a \in B} |T_a| \quad \text{mit} \quad B = \{b \in A | t_s \leq t_{b,l} < t_e\}, \tag{2.2}$$

wobei die Bedingung für den Auftrag $b \in A$ aussagt, dass der Auftrag b im Intervall I fertiggestellt bzw. ausgeliefert worden sein muss. Die kumulierte absolute Termintreue T_A ist zu minimieren: $T_A \rightarrow \min$.

Die Verspätung V_a und die Termintreue T_a ließen sich zusätzlich wichten [DSV97, S. 294], wobei sich z. B. der Auftragsumfang $w_{a,e} - w_{a,s}$ oder die geplante Durchlaufzeit $t_{a,e} - t_{a,s}$ als Divisor anböten. Die Verspätung V_a und die Termintreue T_a des Auftrages $a \in B$ in den Formeln 2.1 und 2.2 ließen sich als Alternative zum absoluten Betrag auch quadrieren, wodurch stärkere Abweichungen vom geplanten Verlauf stärker gewichtet würden [Sch02a, S. 77]. Eine positive Verspätung (Verfrühung) schwächer zu wichten als eine negative, kommt ebenfalls als Alternative in Frage.

Das zweite technizitäre Ersatzziel sind *minimale Durchlaufzeiten* der Aufträge. Aus geringen Durchlaufzeiten resultieren kurze Lieferfristen, was wiederum das Ansehen des Unternehmens stärkt, Folgeaufträge nach sich zieht und somit den Umsatz erhöht. Geringe Durchlaufzeiten senken zudem die Kosten, weil die Produktion das Kapital nur kurzzeitig bindet.

[3] Die Termintreue des Auftrags a ist negativ (schlecht) wenn sein tatsächlicher Fertigstellungstermin $t_{a,l}$ zeitlich hinter dem geplanten Fertigstellungstermin $t_{a,e}$ liegt

Die Durchlaufzeit D_a eines Auftrages $a \in A$ ist $t_{a,l} - t_{a,s}$. Die mittlere Durchlaufzeit $\overline{D}_A$ über den Aufträgen A im Zeitintervall $I = [t_s, t_e)$ lässt sich wie folgt bestimmen:

$$\overline{D}_A = \frac{\sum_{a \in B} D_a}{|B|} \quad \text{mit} \quad B = \{b \in A | t_s \leq t_{b,l} < t_e\} \quad \text{und} \quad |B| > 0. \tag{2.3}$$

Die mittlere Durchlaufzeit $\overline{D}_A$ ist zu minimieren: $\overline{D}_A \rightarrow \min$.

Als drittes gilt es, *minimale Lagerbestände* zu erzielen. Niedrige Lagerbestände implizieren geringe Kapitalbindungskosten und wirken sich somit ebenfalls positiv auf übergeordnete Ziele des Controlling aus. Der Wert der Materialien in der Produktion drückt den Lagerbestand aus, wobei der Wert eines Materials steigt, je weiter es bearbeitet worden ist.

Vereinfachend sei jedes Material mit einem Auftrag assoziiert, und der Auftragsfortschritt drücke den Wert der mit dem Auftrag assoziierten Materialien und somit den Wert des Auftrages aus. Der Wert W_a eines Auftrages $a \in A$ zum Zeitpunkt t_j beträgt $w_{a,j}$. Der über den Aufträgen A kumulierte Wert W_A der Aufträge A lautet:

$$W_A = \sum_{a \in B} W_a \quad \text{mit} \quad B = \{b \in A | (t_{b,s} \leq t_j) \wedge (w_{b,j} < w_{b,e})\}. \tag{2.4}$$

Im Sinne minimaler Lagerbestände ist der kumulierte Wert W_A zu minimieren: $W_A \rightarrow \min$.

Das vierte und letzte technizitäre Ersatzziel besteht darin, die Kapazität der Produktion stark und gleichmäßig auszulasten. Im Hinblick auf die globalen Ziele des Controlling führt eine *maximale Kapazitätsauslastung* dazu, dass fixe Kosten, die für die Kapazitäteinheiten und für die Aufrechterhaltung der Produktion anfallen, auf die Aufträge bzw. auf die Produkte umgelegt werden.

Es sei K die Menge der Kapazitätseinheiten, und $k \in K$ sei eine Kapazitätseinheit. Es sei $Z_k \subseteq I_k \subseteq I$ mit $I = [t_s, t_e)$. Es sei $|I_k|$ die Zeit, zu der die Kapazitätseinheit k zur Produktion zur Verfügung steht, und $|Z_k|$ sei die Zeit, zu der die Kapazitätseinheit k tatsächlich produziert bzw. Aufträge aus A bearbeitet. Die Kapazitätsauslastung L_k der Kapazitätseinheit k im Zeitintervall I ist $|Z_k|/|I_k|$ mit $|I_k| > 0$. Die globale Kapazitätsauslastung L_K über den Kapazitätseinheiten K bzw. über der gesamten Produktion im Zeitintervall I lässt sich wie folgt berechnen:

$$L_K = \frac{\sum_{k \in K} |Z_k|}{\sum_{k \in K} |I_k|} \quad \text{mit} \quad \sum_{k \in K} |I_k| > 0. \tag{2.5}$$

Die globale Kapazitätsauslastung L_K ist zu minimieren: $L_K \rightarrow \max$.

Die technizitären Ersatzziele der PPS interdependieren. Zum Teil gehen sie miteinander konform, zum Teil wirken sie einander entgegen. So sind kurze Durchlaufzeiten oft nur dann zu erzielen, wenn die Kapazität der Produktion niedrig ausgelastet ist und hohe Lagerbestände vorliegen, weil so Produkte produziert werden können, ohne dass Aufträge lange auf Materialien oder Kapazitätseinheiten warten müssen. Es wird angestrebt, die vier Ziele miteinander in Einklang zu bringen. Zu diesem Zweck sind sie auf globale Ziele des Controlling abzubilden, was jedoch – wie bereits erwähnt – nicht immer gelingt.

Nachdem die Ziele der PPS behandelt worden sind, obliegt es, die Nebenbedingungen zu erörtern: Es liegen drei Kategorien von Nebenbedingungen vor; Nebenbedingungen bezüglich

1. der Menge,
2. der Zeit und
3. der Kapazität.

Die Nebenbedingung 1 sagt z. B. aus, dass ein Arbeitsgang eines Auftrages nur dann bearbeitet werden kann, wenn alle für seine Bearbeitung notwendigen Materialien zur Verfügung stehen bzw. alle Aufträge, welche diese Materialien erzeugen, bereits abgearbeitet worden sind. Die Nebenbedingung 2 besagt z. B., dass ein Arbeitsgang eines Auftrages erst dann begonnen werden kann, wenn der ihm im Arbeitsplan direkt vorangehende Arbeitsgang bereits vollendet worden ist. Die Nebenbedingung 3 konstatiert, dass ein Arbeitsgang eines Auftrages erst dann gestartet werden kann, wenn die für seine Bearbeitung notwendigen Kapazitätseinheiten zur Verfügung stehen.

2.2 Teilaufgaben der PPS

Um die gesteckten Ziele der PPS unter den vorliegenden Nebenbedingungen zu erreichen, bestimmt die PPS Werte von Entscheidungsvariablen bzw. von Stellgrößen. Bei diesen Variablen bzw. Größen handelt es sich wiederum um Mengen, Zeiten bzw. Termine und Kapazitäten der Entitäten – insbesondere der Aufträge – der Produktion.[4] Die PPS legt fest und stellt sicher, welche Mengen zu welchen Zeiten an welchen Orten zu produzieren oder zu bestellen sind bzw. vorliegen müssen, wobei sie gehalten ist, ihre Ziele zu erreichen und ihre Nebenbedingungen einzuhalten. [CF99, S. 41] Weil eine simultane PPS – obwohl erstrebenswert – aufgrund ihrer Komplexität nicht möglich ist

[4]Die Kapazität der Produktion ist jedoch oft unveränderlich.

[Kur03, S. 45 ff.], werden besagte Entscheidungsvariablen sukzessive festgelegt [DSV97, S. 18]. Abbildung 2.2 stellt die aufeinanderfolgenden Schritte bzw. Teilaufgaben der PPS graphisch dar.

- *Primärbedarfsplanung*: Die Primärbedarfsplanung bestimmt die Mengen der Endprodukte, die im Planungszeitraum zu fertigen sind. In ihrem Rahmen ermittelt die *Primärbedarfsprognose* den Bedarf, der nicht in Form von Kundenaufträgen vorliegt, und die *Grobplanung* legt fest, in welchem Maße und unter welchem Verhältnis der Endprodukte die Produktion den Bedarf befriedigen wird. Die Grobplanung berücksichtigt grob die Kapazität der Produktion.

- *Mengenplanung*: Die Mengenplanung legt im Rahmen der *Materialbedarfsplanung* den Sekundär- und Tertiärbedarf fest und ermittelt im Rahmen der *Auftragsplanung Losgrößen* der Aufträge und *Bestellmengen* für einzukaufende Materialien.[5] Die Materialbedarfsplanung ermittelt den Sekundär- und Tertiärbedarf – entweder (primär) *bedarfsgesteuert* durch Stücklistenauflösung oder *verbrauchsgesteuert* mittels Bedarfsprognose – und berücksichtigt dabei Lagerbestände. Die Auftragsplanung bestimmt Losgrößen und Bestellmengen zumeist mittels mathematischer Modelle, die allerdings von stark vereinfachenden Annahmen ausgehen.

- *Terminplanung*: Die Terminplanung nimmt die *Arbeitsplanauflösung* vor, indem sie die Aufträge anhand ihrer Arbeitspläne in Arbeitsgänge zerlegt, und vollzieht die *Durchlaufterminierung*, indem sie den Arbeitsgängen Start- und Endtermine zuordnet. Sie berücksichtigt Nebenbedingungen bezüglich der Zeit und der Menge, aber nicht bezüglich der Kapazität.

- *Kapazitätsplanung*: Die Kapazitätsplanung führt zuerst die *Kapazitätsbedarfsrechnung* durch, welche anhand der Start- und Endtermine der Arbeitsgänge den Kapazitätsbedarf über der Zeit ermittelt. Die sich im Rahmen der Kapazitätsplanung anschließende *Kapazitätsterminierung* gleicht Kapazitätsangebot und -nachfrage über der Zeit ab. Zu diesem Zweck werden Arbeitsgänge und somit Aufträge verschoben(, oder die Kapazität wird kurzfristig variiert).

[5] Aufträge und Bestellungen sowie Losgrößen und Bestellmengen lassen sich in vielerlei Hinsicht analog behandeln, weil sie gleiche Eigenschaften bezüglich Menge, Zeit und Kapazität aufweisen. [Cor04, S. 445] Die Ausführungen zu Aufträgen in Abschnitt 2.1 lassen sich demzufolge auch auf Bestellungen anwenden.

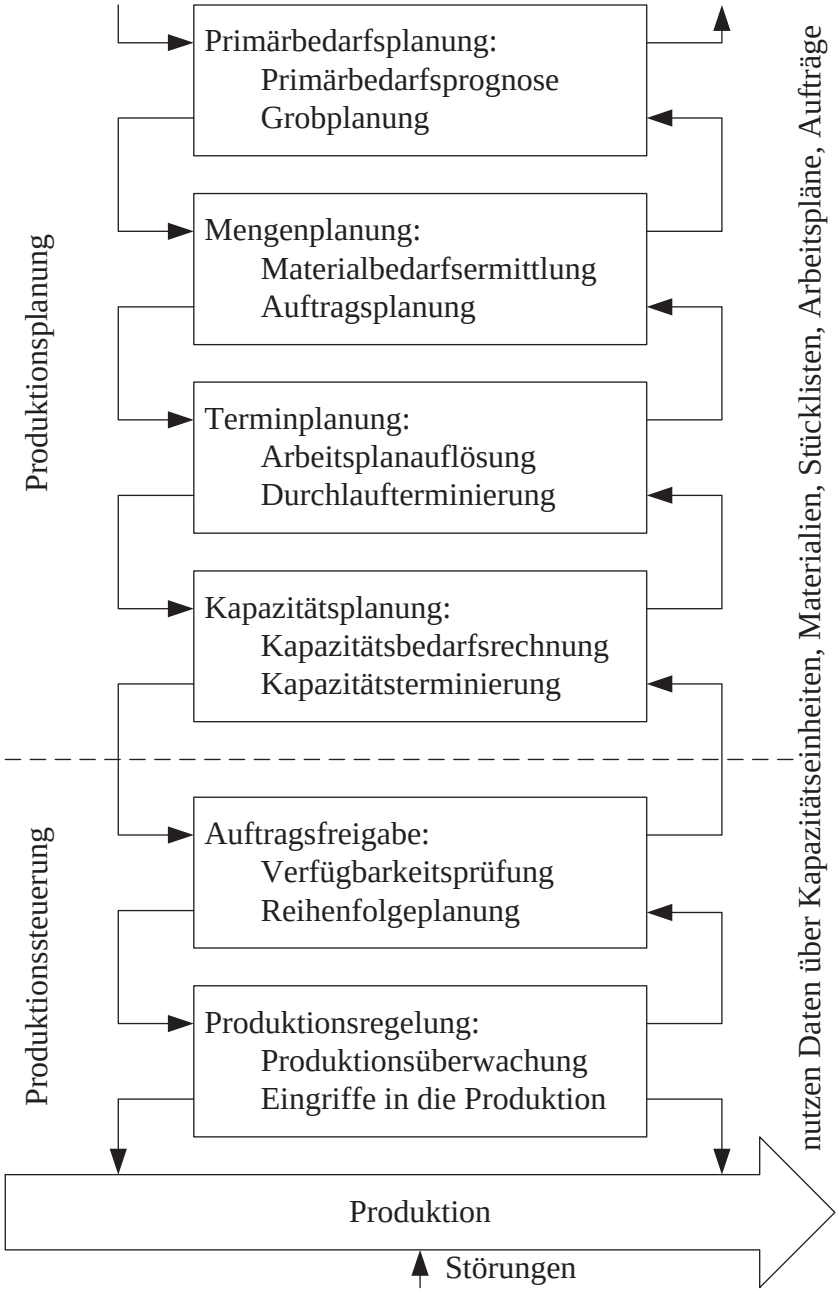

Abbildung 2.2: Schritte bzw. Teilaufgaben der PPS, angelehnt an [Cor04, S. 520 ff.] und [Kur05, S. 137 ff.]

An dieser Stelle geht der Übergang von der Produktionsplanung zur Produktionssteuerung vonstatten.

- *Auftragsfreigabe*: Anhand der Ergebnisse der Termin- und Kapazitätsplanung oder anhand geschätzter Durchlaufzeiten erfolgt die Auftragsfreigabe. In ihrem Kontext ermittelt die *Verfügbarkeitsprüfung*, ob die für die freizugebenden Aufträge nötigen Kapazitäten und Materialmengen zur Verfügung stehen (werden). Aufträge, für die das nicht der Fall ist, stellt die Verfügbarkeitsprüfung zurück. Für die freigegebenen Aufträge und die Aufträge, welche sich noch in der Produktion befinden, erfolgt eine *Reihenfolgeplanung*, aus der ein Maschinenbelegungsplan resultiert.

- *Produktionsregelung*: Die Produktion produziert im Anschluss die freigegebenen Aufträge nach dem Maschinenbelegungsplan. Interne und externe Faktoren – z. B. Maschinenausfälle oder Eilaufträge – stören die Produktion. Störungen erfordern die Produktionsregelung, welche die *Produktionsüberwachung* vornimmt und – sollte die Produktion vom Plan abweichen – regelnde *Eingriffe in die Produktion* vollführt.

Zwischen den dargestellten Schritten der PPS sollte eine Rückkopplung erfolgen: Falls ein Schritt i einem nachfolgenden Schritt $i+1$ Vorgaben unterbreitet, die Schritt $i+1$ nicht einhalten kann, ohne die Nebenbedingungen zu verletzen, sollte ein Rücksprung von Schritt $i+1$ zu Schritt i erfolgen und Schritt i seine Vorgaben so anpassen, dass Schritt $i+1$ seine Teilaufgabe lösen kann. Die Rücksprünge können sich auch über mehrere Schritte fortpflanzen.

Die meisten herkömmlichen PPS-Systeme basieren auf dem hier dargestellten Konzept zur PPS. [Mer04, S. 127 ff.], [Sch95, S. 49] Das Konzept lässt sich aber nur unter bestimmten Voraussetzungen [Cor04, S. 442 f.] erfolgreich anwenden. Andernfalls weist es Mängel und Schwächen auf (siehe z. B. [Ada98, S. 608 ff.]). Aus diesem Grund hat das Konzept viele Anpassungen und Verbesserungen erfahren (siehe z. B. [Cor04, S. 446 ff.] und [Zäp82, S. 308 ff.]). Die im Konzept enthaltenen Schritte bzw. Teilaufgaben der PPS treten jedoch einzeln oder kombiniert auch in abgewandelten oder anderen Konzepten zur PPS auf. Deshalb werden sich die Betrachtungen im weiteren an diesen Teilaufgaben orientieren.

2.3 Unsicherheit in Produktion und PPS

2.3.1 Stochastische, informationelle und linguistische Unsicherheit

Ein Problem der PPS besteht in der Unsicherheit der Produktion. Laut [Zim93, S. 3 ff.] existieren drei voneinander verschiedenen Formen der Unsicherheit: stochastische, informationelle und linguistische.

- *Stochastische Unsicherheit* liegt vor, wenn Werte einer Variable – gegebenenfalls bedingt durch Werte anderer Variablen – zufällig auftreten. Mit Hilfe der Wahrscheinlichkeitstheorie kann stochastische Unsicherheit adäquat abgebildet werden. [Sac99, S. 33], [BKI03, S. 27] Die Variable nimmt ihre Zustände mit bestimmten Wahrscheinlichkeiten an, bzw. eine Wahrscheinlichkeitsdichte über dem Wertebereich einer Variablen existiert. Die Wahrscheinlichkeiten oder die Wahrscheinlichkeitsdichte sind zumeist nicht bekannt, so dass sie aufgrund vorliegender Stichproben und/oder durch einen Experten geschätzt werden. In der Produktion kann z. B. die Rüstzeit stochastisch unsicher sein – ggf. bedingt durch den rüstenden Werker bzw. seine Erfahrung.

- *Informationelle Unsicherheit* tritt dann auf, wenn das Informationsangebot nicht der Informationsnachfrage entspricht. Das ist zum einen der Fall, wenn benötigte Information nicht oder nur zum Teil vorliegt, zum anderen, wenn mehr Information als benötigt anfällt. Liegt zu wenig Information vor, können Entscheidungen nicht oder nur mit unzureichender Güte gefällt werden. Ein Überangebot an Information hingegen überfordert und verwirrt den menschlichen Entscheider, weil er im Durchschnitt nur sieben Fakten gleichzeitig ins Kalkül ziehen kann. [Mil56] Informationeller Unsicherheit kann ebenfalls mittels Wahrscheinlichkeitstheorie begegnet werden. Fälle falscher Information, die z. B. aus Messfehlern oder Fehlern bei der Datenerfassung oder -übertragung resultieren, zählen ebenfalls zu informationeller Unsicherheit.

- Die dritte Form der Unsicherheit, die *linguistische Unsicherheit* betrifft die menschliche Sprache, die Sachverhalte ungenau beschreibt und in ihrer Bedeutung stark vom Kontext abhängt. Der Ausdruck „umfangreicher Auftrag“ sagt nicht aus, um welche genaue Stückzahl oder um welchen exakten materiellen Wert es sich bei besagtem Auftrag handelt, und bedeutet für einen Getränkeproduzenten etwas anderes als

für einen Einzelfertiger von Spezialmaschinen. Tritt linguistische Unsicherheit auf, kann sie rechentechnisch z. B. mittels wissensbasierter Fuzzy-Systeme abgebildet werden. [KKW96, S. 93 ff.], [Zim95, S. 24 ff.] Aber laut [Lin87] kann und sollte jegliche Form von Unsicherheit mittels Wahrscheinlichkeiten abgebildet werden.

Nachfolgend wird dargelegt, inwiefern stochastische, informationelle und linguistische Unsicherheit die Teilaufgaben der PPS betrifft. Eine Teilaufgabe der PPS ist immer dann von Unsicherheit betroffen, wenn der Wert mindestens einer zu ihrer Lösung notwendigen Variable nicht sicher ist.

2.3.2 Unsicherheit in den Teilaufgaben der PPS

Linguistische Unsicherheit ist im Kontext der PPS nahezu omnipräsent, da sie auftreten kann, wenn der Mensch involviert ist und der Mensch stark mit der PPS interagiert. Aussagen in natürlicher Sprache über Termine, Zeiten, Mengen, Kosten oder Kapazitäten können für jede Teilaufgabe der PPS getroffen werden.

Inwiefern stochastische und informationelle Unsicherheit die Teilaufgaben der PPS betrifft, wird nun im Detail dargelegt. Die Darlegung orientiert sich an den in Abschnitt 2.2 und Abbildung 2.2 dargestellten Teilaufgaben der PPS.

- *Primärbedarfsplanung*: Sofern im Rahmen der Primärbedarfsplanung der Bedarf aufgrund von Kundenaufträgen nicht vollständig bekannt ist, liegt informationelle Unsicherheit vor: Es ist nicht bekannt, welche Produkte in welcher Menge nachgefragt und somit abgesetzt werden können. Stochastische Unsicherheit prägt die *Primärbedarfsprognose*, welche oft aufgrund von Vergangenheitsdaten erfolgt. Die bei der *Grobplanung* berücksichtigten Kapazitäten sind ebenfalls und zumindest mit stochastischer Unsicherheit behaftet.

- *Mengenplanung*: Im Rahmen der Mengenplanung tritt bei der *bedarfsgesteuerten Materialbedarfsplanung* keine Unsicherheit auf – sofern keine stochastischen Fehlmengen berücksichtigt werden und die Ergebnisse der Primärbedarfsplanung sicher sind. Erfolgt allerdings *verbrauchsgesteuerte Materialbedarfsplanung*, sind tatsächliche Bedarfe unbekannt und werden prognostiziert. Somit liegt informationelle und stochastische Unsicherheit in der Materialbedarfsplanung vor. Stochastische Unsicherheit prägt auch die *Auftragsplanung*: Schon stochastische Fehlmengen, schwankende Beschaffungskosten und stochastischer Verbrauch führen dazu.

- *Terminplanung*: Im Rahmen der Terminplanung ist die *Arbeitsplanauflösung* frei von Unsicherheit. Die *Durchlaufterminierung* unterliegt jedoch ebenfalls stochastischer Unsicherheit, da die Rüst- und Bearbeitungszeiten für die Arbeitsgänge im Allgemeinen stochastisch schwanken. Wird die stochastische Unsicherheit bei der Durchlaufterminierung berücksichtigt, sind auch die aus ihr resultierenden Arbeitsgangtermine unsicher.

- *Kapazitätsbedarfsrechnung*: Unsichere Termine für Arbeitsgänge vorausgesetzt, unterliegt auch die Kapazitätsbedarfsrechnung im Rahmen der *Kapazitätsplanung* stochastischer Unsicherheit; sonst wäre sie deterministisch. Die *Kapazitätsterminierung* ist schon durch stochastische Maschinenausfälle von Unsicherheit geprägt.

- *Auftragsfreigabe*: Die *Verfügbarkeitsprüfung* im Rahmen der Auftragsfreigabe läuft unter Sicherheit ab, es sei denn, sie wird vorausschauend vorgenommen: Dann sind Zeiten, Mengen und Kapazitäten zumindest stochastisch unsicher. Kann die Verfügbarkeit der Materialien nicht geprüft werden, liegt informationelle Unsicherheit vor. Geht man von stochastischen Rüst- und Bearbeitungszeiten, stochastischem Kapazitätsangebot und stochastischer Materialverfügbarkeit aus, ist die *Reihenfolgeplanung* am stärksten von stochastischer Unsicherheit betroffen. Das verwundert nicht, da sie Zeiten, Mengen und Kapazitäten simultan plant bzw. berücksichtigt.

- *Produktionsregelung*: Wie nicht anders zu erwarten, ist auch die Produktionsregelung von Unsicherheit betroffen: Erfolgt keine vollständige oder unregelmäßige Betriebsdatenerfassung, leidet die *Produktionsüberwachung* unter informationeller Unsicherheit. *Eingriffe in die Produktion* können hinsichtlich ihrer Wirkung nicht sicher vorhergesagt werden, da die Produktion an sich stochastisch unsicher ist.

Tabelle 2.1 stellt zusammengefasst dar, bei welchen Teilaufgaben der PPS welche Form der Unsicherheit auftritt. Informationelle Unsicherheit kann noch wesentlich stärker ins Gewicht fallen, wenn relevante PPS-Daten im PPS- bzw. im ERP-System nicht (ordnungsgemäß) gepflegt und/oder keine Vergangenheitsdaten zur Quantifizierung stochastischer Unsicherheit aufbewahrt werden.

Wie Tabelle 2.1 darstellt, betrifft stochastische Unsicherheit alle Teilaufgaben der PPS – bis auf die Arbeitsplanauflösung. Deshalb scheint es geboten, zumindest stochastische Unsicherheit in die Modelle zu integrieren,

Tabelle 2.1: Arten der Unsicherheit in den Teilaufgaben der PPS: X bedeutet, dass die entsprechende Unsicherheit für die Teilaufgabe der PPS im Allgemeinen vorliegt, und (X) bedeutet, dass die entsprechende Unsicherheit für die Teilaufgabe der PPS unter Umständen vorliegen kann. Die Spalte „gesamt“ sagt aus, ob die entsprechende Teilaufgabe der PPS überhaupt mit Unsicherheit behaftet ist, was dann der Fall ist, wenn ihr mindestens eine Art von Unsicherheit anhaftet.

	Teilaufgabe der PPS			Unsicherheit			
				stochastisch	informationell	linguistisch	gesamt
	Primärbedarfsplanung	Primärbedarfsprognose		X	X	(X)	X
		Grobplanung		X		(X)	X
	Mengenplanung	Materialbedarfsplanung	bedarfsgesteuert	(X)		(X)	(X)
			verbrauchsgesteuert	X	X	(X)	X
		Auftragsplanung		X		(X)	X
	Terminplanung	Arbeitsplanauflösung					
		Durchlaufterminierung		X		(X)	X
	Kapazitätsplanung	Kapazitätsbedarfsrechnung		(X)		(X)	(X)
		Kapazitätsterminierung		X		(X)	X
	Auftragsfreigabe	Verfügbarkeitsprüfung		(X)	(X)	(X)	(X)
		Reihenfolgeplanung		X		(X)	X
	Produktionsregelung	Produktionsüberwachung			(X)	(X)	(X)
		Eingriffe in die Produktion		X	(X)	(X)	X

welche zum Lösen der Teilaufgaben der PPS eingesetzt werden. Das Anwenden solcher Modelle würde zu fundierten Entscheidungen und Plänen führen, deren Sicherheit bzw. Unsicherheit quantifiziert werden könnte.

Zum Thema Unsicherheit in der Produktion sei noch angemerkt, dass der unsichere Wert einer Variablen kein einzelnes und alleinstehendes Phänomen ist, sondern dass dieser Wert von ebenfalls unsicheren Werten weiterer Variablen abhängt. Werte letzterer Variablen werden wiederum von Werten anderer Variablen bedingt, usw. Die Abhängigkeiten zwischen den Variablen sind ebenfalls unsicher. [KR84, S. 23] Die vielfältigen und unsicheren Abhängigkeiten erschweren die Modellierung und die Verarbeitung der Unsicherheit in Produktion und PPS enorm.

2.4 Produktion als komplexer stochastischer Prozess

Die Produktion kann einerseits als System und andererseits als Prozess betrachtet werden. Laut DIN ist ein System „eine in einem betrachteten Zusammenhang gegebene Anordnung von Gebilden, die miteinander in Beziehung stehen." [DIN98, S. 3] In der Produktion handelt es sich bei besagten Gebilden z. B. um Kapazitätseinheiten, welche beispielsweise virtuell über Arbeitspläne oder materiell über Transportwege miteinander in Beziehung stehen.

Laut DIN ist ein Prozess eine „Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Materie, Energie oder auch Information umgeformt, transportiert oder auch gespeichert wird." [DIN98, S. 3] Ein Prozess läuft also in einem System ab. Während der Produktion finden im allgemeinen Arbeits- und Transport*vorgänge* statt, die aufeinander einwirken und Materie, Energie und Information umformen bzw. transportieren. Ob der Begriff Produktion das System, den Prozess oder beide bezeichnet, geht im folgenden zumeist aus dem Kontext hervor. Soll explizit entweder auf den System- oder auf den Prozesscharakter der Produktion hingewiesen werden, so werden die Begriffe Produktions*system* bzw. Produktions*prozess* verwandt.

Um die Produktion als stochastischen Prozess zu beschreiben, erfolgen einige Vorbemerkungen und Definitionen:

Definition 2.1 (Zustandsvariable)
Es sei $S = \{s_i | i \in I_x\}$ eine Menge von Werten, und die Indexmenge I_x stelle S dar. Eine Variable x, welche S aufweist, heißt Zustandsvariable. Ein Element $s_i \in S$ heißt Zustand von x. Wenn S abzählbar ist, dann heißt x

diskret; wenn S ein Intervall von $\mathbb{R}$ und somit überabzählbar ist, dann heißt x kontinuierlich.[6]

Definition 2.2 (Zustandsraum)
Es sei $X = \{x_i | i \in I_X\}$ die Menge aller Zustandsvariablen, und die abzählbar endliche Indexmenge I_X stelle X dar. Es sei S_i die Wertemenge von x_i mit $i \in I_X$. Es sei $I_Y \subseteq I_X$ und $Y = \{x_i \in X | i \in I_Y\}$. Der Zustandsraum S_Y von Y ist das kartesische Produkt $\underset{i \in I_Y}{\times} S_i$. Die Dimension von S_Y ist $d = |I_Y|$, und ein d-Tupel $s \in S_Y$ heißt Zustand von Y.[7]

Jedes System weist einen Zustandsraum auf und befindet sich immer in einem bestimmten Zustand – so auch die Produktion; und Zustandsvariablen beschreiben (mit ihren Zuständen) den Zustand eines jeden Systems und somit auch den der Produktion. Die Zustände der Produktion verkörpern zufällige Ereignisse, da ihr Zutreffen bzw. Eintreten unter gleichen Bedingungen unsicher ist. Die Unsicherheit des Zustandes der Produktion beschreiben Zufallsvariablen:

Definition 2.3 (Zufallsvariable)
Es sei Ω die Menge der Elementarereignisse. Es sei $\mathcal{A}$ eine Borel'sche σ-Algebra auf Ω.[8] *Es sei* Pr *ein Wahrscheinlichkeitsmaß auf dem Meßraum $(\Omega, \mathcal{A})$, und $(\Omega, \mathcal{A}, \Pr)$ sei ein Wahrscheinlichkeitsraum. Eine Zufallsvariable ist eine messbare Funktion, welche auf Ω des Wahrscheinlichkeitsraumes definiert ist.*[9]

Wenn im folgenden Zufallsvariablen verwandt werden, beziehen sie sich stets auf einen passenden Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, \Pr)$.

Zwei Beispiele folgen, wie Zufallsvariablen im Rahmen dieser Arbeit vorwiegend festgelegt und verwandt werden:

1. Es seien I_X, I_Y, X, Y, S_Y, x_i und S_i wie in Definition 2.2 vereinbart. Zudem sei $\Omega = S_Y$. Dann ist die Projektion $v : S_Y \to S_i$ eine (eindimensionale) Zufallsvariable mit dem Wertebereich S_i.

2. Es sei zusätzlich $I_Z \subseteq I_Y$ die $Z \subseteq Y$ darstellende Indexmenge mit $d = |I_Z| = |Z| \geq 1$, und S_Z sei der Zustandsraum von Z. Dann ist die

[6] angelehnt an [Bou95, S. 19]
[7] angelehnt an [Bou95, S. 19]
[8] Für $\mathcal{A}$ kann bei abzählbarem (diskretem) Ω die Potenzmenge $\mathbb{P}(\Omega)$ angenommen werden [Kre02, S. 42], die immer eine Borel'sche σ-Algebra auf S ist.
[9] Zu (Elementar)ereignissen, (Borel'schen σ-)Algebren, Messbarkeit, Messräumen, (Wahrscheinlichkeits)maßen und ihren Eigenschaften siehe z. B. [Kre02, S. 128 ff.], [Bau91, S. 1 ff.] und [Chu78, S. 116 f.].

Tabelle 2.2: Die vier grundlegenden Typen stochastischer Prozesse

		Zustands- bzw. Zufallsvariable	
		diskret	kontinuierlich
Zeit	diskret	1	2
	kontinuierlich	3	4

Projektion $V : S_Y \rightarrow S_Z$ eine (mehr- bzw. d-dimensionale) Zufallsvariable.

Der Wertebereich einer Zufallsvariable ist demzufolge im eindimensionalen Fall der Wertebereich einer Zustandsvariable und im mehrdimensionalen Fall der Zustandsraum des Prozesses. Abhängig davon, ob die Zustandsvariablen diskret bzw. kontinuierlich sind, heißen auch die Zufallsvariablen diskret bzw. kontinuierlich.

Bis jetzt ist der Zustand der Produktion zu einem (impliziten) Zeitpunkt betrachtet worden. Betrachtet man die Entwicklung (der Zustände) der Produktion über der Zeit, gelangt man zu einem stochastischen Prozess:

Definition 2.4 (Stochastischer Prozess)
Es sei $(\Omega, \mathcal{A}, \Pr)$ *ein Wahrscheinlichkeitsraum mit* Ω, $\mathcal{A}$ *und* $\Pr$ *wie in Definition 2.1 vereinbart. Ein stochastischer Prozess ist eine Menge* $\{V_t | t \in T\}$ *von Zufallsvariablen, wobei* T *die Indexmenge verkörpere, welche den Prozess darstellt. Wenn* T *abzählbar ist, dann heißt der Prozess zeitdiskret; wenn* T *ein Intervall von* $\mathbb{R}$ *(und überabzählbar) ist, dann heißt der Prozess zeitkontinuierlich.*[10]

Die Menge T verkörpert die Zeit, welche in geordneten Zeitpunkten oder Zeiträumen angegeben wird. Bei der Produktion handelt es sich um einen stochastischen Prozess [KR84, S. 46] im Sinne der Definition 2.4. Komplex – im Sinne der Überschrift dieses Kapitels – wird der Produktionsprozess dadurch, dass er im Allgemeinen durch mehrere Zufallsvariablen bzw. durch eine mehrdimensionale Zufallsvariable beschrieben werden muss und dass seine Zustandsvariablen stark voneinander abhängen.

Es liegen vier grundlegende Typen von stochastischen Prozessen vor, die Tabelle 2.2 darstellt. Viele der Zustandsvariablen einer Produktion – wie z. B. der Auftragsfortschritt, der Energieverbrauch oder der Arbeitsinhalt im Produktionssystem – sind kontinuierlich und ändern sich kontinuierlich über der Zeit, so dass eine möglichst genaue Beschreibung der Produktion über einem reellen Zeitintervall erfolgen sollte. Viele Ansätze, welche die Produktion beschreiben, – so auch die ereignisdiskrete Simulation [Fis01, S. 5 ff.]

[10] angelehnt an [Win00, S. 12]

–, abstrahieren jedoch von „kontinuierlicher Zeit“ und gelangen so von Typ 4 zu Typ 2. Die Abstraktion ist zulässig, weil erstens die Zustandsänderungen bzw. -übergänge in der Produktion von Ereignissen ausgelöst werden, welche zu diskreten Zeitpunkten erfolgen, und weil zweitens die Veränderung einer kontinuierlichen Variable zwischen zwei Zeitpunkten bzw. Ereignissen oft als linear angenommen und interpoliert werden kann, ohne dass daraus ein Verlust an Genauigkeit resultiert. Eine weitere Abstraktion im Rahmen des Typs 2 besteht darin, jeweils zwei aufeinanderfolgende diskrete Zeitpunkte als äquidistant zu unterstellen. Die Δt-Simulation (System Dynamics [For71], [For61]) z. B. setzt aufeinanderfolgende Zeitpunkte als äquidistant voraus. [Völ03, S. 57] Geht man darüber hinaus von diskreten Variablen aus – wie z. B. der Warteschlangenlänge, der Anzahl der Aufträge im Produktionssystem oder dem Status einer Kapazitätseinheit (frei, rüstend, belegt, außer Betrieb) – so trifft Typ 1 auf den Produktionsprozess und seine Abbildung zu. Sollten nicht alle betrachteten Variablen diskret sein, kann man sie gegebenenfalls diskretisieren (siehe Abschnitt 4.4.3). Der Typ 3 – kontinuierliche Zeit und diskrete Variablen – tritt eher selten auf.

Zum Abschluss des Abschnitts sei noch auf zwei spezielle Eigenschaften zeitdiskreter stochastischer Prozesse hingewiesen, die im Rahmen der vorliegenden Arbeit Bedeutung erlangen werden.

Definition 2.5 (Stationärer Prozess)
Es sei $T = \{0, 1, \ldots, n\}$. Es sei $\{V_t | t \in T\}$ ein diskreter stochastischer Prozess, dessen Zufallsvariablen V_t mit $t \in T$ jeweils denselben Zustandsraum S aufweisen. Der Prozess ist stationär, wenn gilt

$$\Pr(V_t = s) = \Pr(V_{t+\Delta t} = s) = \Pr(V = s) \tag{2.6}$$

für alle $t, (t + \Delta t) \in T$ und für alle $s \in S$.[11]

Das heißt, dass ein Prozess dann stationär ist, wenn die Wahrscheinlichkeitsfunktionen über seinen Zufallsvariablen V_t mit $t \in T$ nicht von der Zeit t abhängen bzw. nicht über der Zeit t variieren, sondern konstant bleiben. Im Allgemeinen ist ein Produktionsprozess nicht zwangsläufig stationär. Er kann vorkommen, dass er z. B. Trends, periodischen bzw. saisonalen Schwankungen oder einer Anlaufphase unterliegt. Viele Verfahren zur Abbildung, Steuerung und Regelung von Prozessen setzen jedoch Stationarität voraus, so z. B. die Bedientheorie [GH98], [Pra97]. Stationarität kann man z. B. künstlich hervorrufen, indem man erkannte Trends durch Zufallsvariablen abbildet, die Differenzen ausdrücken, Zeitpunkte periodisch günstig wählt und Anlaufphasen

[11] angelehnt an [BD03]

erkennt und eliminiert. Nach Beendigung der Anlaufphase – nachdem sich der Produktionsprozess also eingeschwungen hat – kann man in der Praxis von Stationarität ausgehen. [Pag91, S. 122 ff.]

Definition 2.6 (Markowscher Prozess)
Es sei $T = \{0, 1, \ldots, n\}$. Es sei $\{V_t | t \in T\}$ ein diskreter stochastischer Prozess, dessen Zufallsvariablen V_t mit $t \in T$ jeweils denselben Zustandsraum S aufweisen. Der Prozess ist Markowsch, wenn gilt

$$\Pr(V_t = s_t | V_{t-1} = s_{t-1}, \ldots, V_0 = s_0) = \Pr(V_t = s_t | V_{t-1} = s_{t-1}) \qquad (2.7)$$

für alle $t \in T$ und für alle $s_t \in S$.[12]

Laut der Definition 2.6 ist ein Prozess dann Markowsch, wenn lediglich der Zustand s_{t-1} des Prozesses zum Zeitpunkt $t-1$ die Wahrscheinlichkeit des Zustandes s_t des Prozesses zum Zeitpunkt t bedingt und die Zustände $s_{t-2}, \ldots, s_0$ zu den Zeitpunkten $t-2, \ldots, 0$ irrelevant sind für die Wahrscheinlichkeit des Zustandes s_t des Prozesses zum Zeitpunkt t. Ein Produktionsprozess ist nicht zwangsläufig Markowsch; allerdings lässt er sich durch das Einführen eines sogenannten „Gedächtnisses“ Markowsch machen: Es werden zusätzliche Zufallsvariablen bzw. Dimensionen eingeführt, welche die Zustände des Prozesses zu vergangenen Zeitpunkten abbilden.

2.5 Komplexität der Aufgaben der Produktionsplanung und -steuerung

Ein Problem wird im Allgemeinen dann als komplex angesehen, wenn kein Algorithmus existiert bzw. bekannt ist, welcher besagtes Problem mit – in Abhängigkeit von der Problemgröße – höchstens polynomialem Rechenaufwand löst. Unter der Problemgröße versteht man die Länge der Eingabedaten.[13] Für die Reihenfolge- bzw. Maschinenbelegungsplanung verkörpern z. B. die Anzahl der Maschinen, der Aufträge oder der Arbeitsgänge die Länge der Eingabedaten. Der Rechenaufwand kann in der Anzahl elementarer Rechenoperationen bzw. -schritte oder in Speicherbedarf ausgedrückt werden. [Sch01, S. 151] Ein Algorithmus, der höchstens polynomialen Rechenaufwand verursacht, löst ein Problem im Allgemeinen effizient, wobei man einen Algorithmus, der nur logarithmischen oder linearen Rechenaufwand verursacht, vorziehen sollte – sofern denn einer bekannt ist.

[12] angelehnt an [Str05]

[13] Wenn im weiteren von logarithmischem, linearem, polynomialem oder exponentiellem Rechenaufwand die Rede ist, so beziehe er sich jeweils auf die Problemgröße bzw. auf die Länge der Eingabedaten.

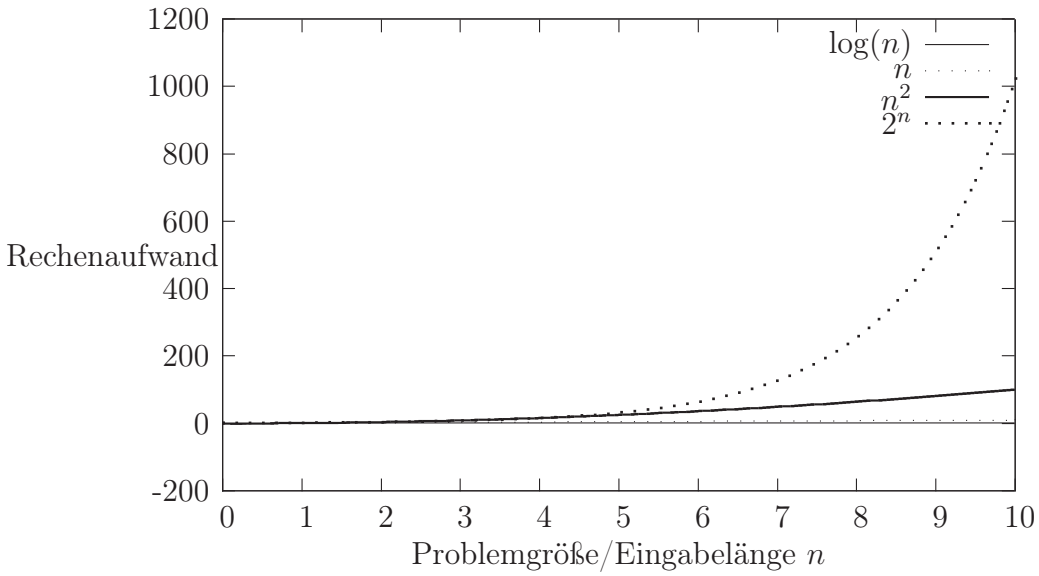

Abbildung 2.3: Beispiele für Verläufe von logarithmischem $\log(n)$, linearem n, polynomialem n^2 und exponentiellem 2^n Rechenaufwand in Abhängigkeit von der Problemgröße/Eingabelänge n

Laut [DSV97, S. 55 ff.] lassen sich Probleme in zwei grundlegende Klassen einteilen:

1. Ein Problem ist *polynomial lösbar*, wenn ein Algorithmus vorliegt, welcher das Problem mit polynomialem Rechenaufwand löst.

2. Ein Problem ist hingegen *NP-schwer*[14], wenn *kein* Algorithmus bekannt ist, welcher das Problem mit polynomialem Rechenaufwand löst.

Polynomial lösbare Probleme (1) können und sollten effizient mit den vorliegenden Algorithmen gelöst werden und bedürfen somit keiner weiteren Betrachtung. Die Klasse der NP-schweren Probleme (2) umfasst die „komplexen“ Probleme, welche sich meist nur mit exponentiellem oder sogar noch höherem Rechenaufwand lösen lassen. Abbildung 2.3 stellt die Verläufe von polynomialem oder niedrigerem und exponentiellem Rechenaufwand einander an Beispielen gegenüber. Steigt der Rechenaufwand so rapide, wie in Abbildung 2.3 ersichtlich, können NP-schwere Probleme realistischer Größenordnung oft nicht oder zumindest nicht effizient exakt bzw. optimal gelöst

[14]Das NP steht für nichtdeterministisch-polynomial und sagt aus, dass ein nichtdeterministischer Rechner die Lösung des Problems mit polynomialem Rechenaufwand „erraten“ könnte. [DSV97, S. 55] Die vorliegende Arbeit betrachtet und verwendet aber ausschließlich deterministische Rechner.

werden. Nichtsdestotrotz liegen NP-schwere Probleme vor und sind bestmöglich und mit akzeptablen Rechenaufwand zu lösen. Zu diesem Zwecke werden problemspezifische Heuristiken entworfen oder zur mathematischen Optimierung alternative Verfahren angewandt, welche keine exakte bzw. (global) optimale Lösung garantieren, aber eine relativ gute liefern und geringeren Rechenaufwand verursachen. Für NP-schwere Probleme der PPS lohnt es, neue Heuristiken zu entwerfen bzw. neue Verfahren bezüglich ihres Potenzials zur Lösung besagter Probleme zu untersuchen.

Welche Teilaufgaben bzw. Probleme der PPS komplex im Sinne obiger Ausführungen sind, stellt Tabelle 2.3 dar. Sie klassifiziert die Teilaufgaben der PPS lediglich grob bezüglich ihrer Komplexität. Erstens weisen spezielle Teilaufgaben unterschiedliche Komplexität auf, so dass man oft nicht eindeutig entscheiden kann, welche Komplexität eine hier angeführte Teilaufgabe aufweist. So existieren spezielle Probleme der Reihenfolge- bzw. Maschinenbelegungsplanung, welche polynomial lösbar sind, aber alle realistischen Probleme der Reihenfolgeplanung sind NP-schwer. [DSV97, S. 396] Zweitens kommt es darauf an, wie das Problem bzw. ob es als (kombinatorisches) Optimierungsproblem aufgefasst und formuliert worden ist. Möchte man bei „Eingriffen in die Produktion“ lediglich erfahren, wie sie sich auswirken, so ist das Problem mittels ereignisdiskreter Simulation im Durchschnitt polynomial lösbar; möchte man den Eingriff allerdings so vornehmen, dass wiederum ein global optimaler Maschinenbelegungsplan entsteht, so ist das Problem im allgemeinen Falle NP-schwer, da die Maschinenbelegungsplanung zu den ganzzahligen, kombinatorischen Optimierungsproblemen gehört, welche wiederum meist zu den NP-schweren Problemen zählen [DD91, S. 112]. Drittens spielt auch die Komplexität des zur Lösung der Teilaufgabe gewählten Algorithmus' eine Rolle: Arbeitet man – z. B. bei Prognosen im Rahmen der Bedarfsplanung – mit Modellen, deren Erstellung und Anwendung exponentiellen Rechenaufwand verursacht, gilt die Teilaufgabe der PPS natürlich auch als NP-schwer. Tabelle 2.3 beschreibt somit lediglich eine Tendenz. Für spezielle Probleme der PPS gibt z. B. [DSV97, S. 135, 396 ff., 427] die Komplexität an. Ist in der Tabelle eine Teilaufgabe der PPS als NP-schwer markiert worden, so bedarf sie weiterer Untersuchungen: Neue (Optimierungs)verfahren können an ihr erprobt und neue problemspezifische Heuristiken für sie entworfen werden.

Tabelle 2.3: Komplexität der Teilaufgaben der PPS: X bedeutet, dass der *allgemeine* Fall der entsprechenden Teilaufgabe der PPS die angegebene Komplexität aufweist.

	Teilaufgabe der PPS			Komplexität	
				polynomial	NP-schwer
	Primärbedarfsplanung	Primärbedarfsprognose		X	
		Grobplanung		X	
	Mengenplanung	Materialbedarfsplanung	bedarfsgesteuert	X	
			verbrauchsgesteuert	X	
		Auftragsplanung			X
	Terminplanung	Arbeitsplanauflösung		X	
		Durchlaufterminierung		X	
	Kapazitätsplanung	Kapazitätsbedarfsrechnung		X	
		Kapazitätsterminierung			X
	Auftragsfreigabe	Verfügbarkeitsprüfung		X	
		Reihenfolgeplanung			X
	Produktionsregelung	Produktionsüberwachung		X	
		Eingriffe in die Produktion			X

2.6 Strukturdefekte und schlechtstrukturierte Probleme in der PPS

2.6.1 Strukturdefekte zur Charakterisierung schlechtstrukturierter Probleme

In der Betriebswirtschaftslehre werden (Entscheidungs)Probleme grundlegend in wohlstrukturierte und schlechtstrukturierte unterteilt. Ein Problem ist wohlstrukturiert, wenn es keinen der folgenden vier Strukturdefekte aufweist [Ada96, S. 10 ff.]:

1. Bewertungsdefekt,
2. Wirkungsdefekt,
3. Zielsetzungsdefekt und
4. Lösungsdefekt.

Ein *Bewertungsdefekt* (1) liegt vor, wenn die zur Beschreibung der Produktion relevanten Zustandsvariablen oder deren zur Beschreibung des *derzeitigen* Zustandes der Produktion relevanten Werte unbekannt oder unsicher sind. Stell- und Regelgrößen bzw. Handlungsalternativen und Ziele sowie intermediäre Größen sind also selbst oder in ihren Ausprägungen nicht determiniert. Es gelingt ergo nicht, den *aktuellen* Zustand der Produktion hinreichend genau zu *messen* und/oder zu *beschreiben*. Eine Facette des Bewertungsdefektes macht die informationelle Unsicherheit aus: Für die PPS bzw. für das Abbilden der Produktion relevante Daten sind nicht erfasst worden oder haben nicht erfasst werden können. Ist nicht bekannt, welche Zustandsvariablen aus einer gegebenen Menge relevant sind oder ob die identifizierten Variablen die Produktion erschöpfend abbilden, liegt ebenfalls ein Bewertungsdefekt vor.

Ein *Wirkungsdefekt* (2) tritt auf, wenn unbekannt oder unsicher ist, welche Wirkung die Zustände von Zustandsvariablen der Produktion auf andere Zustandsvariablen ausüben – und ob überhaupt. So kann beispielsweise nicht festgestellt werden, ob und wie eine Änderung der Stellgrößen die Regelgrößen beeinflusst. Es folgen Beispiele: Angenommen, die Kundenzufriedenheit lasse sich operationalisieren; es liege in Bezug auf sie also kein Bewertungsdefekt vor. Dann ist die Wirkung der Kundenzufriedenheit auf die Nachfrage und den Umsatz in zukünftigen Perioden unbekannt oder zumindest unsicher. Zudem dürften auch Servicegrad und Qualität als Stellgrößen in ihrer Wirkung auf die Kundenzufriedenheit zumindest unsicher, wenn nicht unbekannt sein. Der Umstand, dass oft unbekannt ist, ob und wie Prioritätsregeln

auf technizitäre Ersatzziele der PPS (siehe Abschnitt 2.1) wirken, rangiert ebenfalls unter wirkungsdefekt.

Ein *Zielsetzungsdefekt* (3) besteht, wenn für die Lösung eines Problems keine operationale Zielfunktion vorliegt. [Ada96, S. 10 ff.] Eine Zielfunktion ist dann operational, wenn mit ihrer Hilfe für jede Handlungsalternative bzw. für jede Konfiguration von Stellgrößen ein Funktionswert *errechnet* bzw. *für die Zukunft prognostiziert* werden kann, die Handlungsalternativen anhand der für sie progostizierten Funktionswerte miteinander verglichen und in eine Präferenzreihenfolge gebracht werden können.[15] Im Gegensatz zum Bewertungsdefekt geht es beim Zielsetzungsdefekt nicht um das Messen oder Bestimmen des aktuellen Zustandes der Produktion, sondern um das Errechnen bzw. Prognostizieren der Zielfunktionswerte für die Zukunft. Nicht operational ist eine Zielfunktion auch dann, wenn mehrere konträre oder indifferente Unterziele vorliegen, welche nicht unter einem globalen Oberziel vereint werden können. Ein Beispiel für einen Zielsetzungsdefekt liefern die konträren technizitären Ersatzziele „minimale Durchlaufzeit“ und „minimaler Lagerbestand“: Um geringere Durchlaufzeiten zu erzielen, ist im Allgemeinen ein höherer Lagerbestand vonnöten, weil so kürzer auf Materialien gewartet wird; und ein geringerer Lagerbestand führt im Allgemeinen zu längeren Durchlaufzeiten, weil so länger auf Materialien gewartet wird.

Bestehen die Strukturdefekte 1 bis 3 nicht, kann immer noch ein Lösungsdefekt auftreten: Ein *Lösungsdefekt* (4) ist dann gegeben, wenn kein effizientes Lösungsverfahren existiert, das heißt, wenn kein Verfahren für Probleme realer Größenordnung erfolgversprechende Handlungsalternativen mit akzeptablem Aufwand und in akzeptabler Zeit identifizieren kann. Lösungsdefekte Probleme sind meist diskret und hochdimensional, leiden somit unter der kombinatorischen Explosion der Anzahl möglicher Handlungsalternativen bei zunehmender Problemgröße und weisen extrem „zerklüftete“ Zielfunktionen auf. Lösungsdefekte Probleme entsprechen den NP-schweren Problemen aus Abschnitt 2.5.

Die vier Strukturdefekte sollten in der Reihenfolge, in welcher sie hier angeführt worden sind, behoben, gemildert oder umgangen werden. Tabelle 2.4 listet die Strukturdefekte noch einmal auf und ordnet ihnen ihre grundlegenden Charakteristika zu.

Der folgende Abschnitt legt nun dar, welche Teilaufgaben der PPS wohlstrukturiert sind und welche Teilaufgaben schlechtstrukturierte Probleme verkörpern, und weist den schlechtstrukturierten Problemen der PPS die

[15] Treten Handlungsalternativen mit gleichem Funktionswert auf, so ist ihre Präferenzreihenfolge nicht eindeutig. Im mathematischen Sinne liegt also eine Halbordnung über den Handlungsalternativen vor.

Tabelle 2.4: Strukturdefekte und ihre Charakteristika

#	Strukturdefekt	Charakteristikum
1	Bewertungsdefekt	Variablen oder Ausprägungen unbekannt
2	Wirkungsdefekt	Wirkungszusammenhänge unbekannt
3	Zielsetzungsdefekt	keine operationale Zielfunktion
4	Lösungsdefekt	kein effizientes Lösungsverfahren

Strukturdefekte zu, die ihnen anhaften.

2.6.2 Übersicht über schlechtstrukturierte Probleme der PPS

Tabelle 2.5 zeigt auf, welche Teilaufgaben der PPS wohlstrukturiert sind und welche Teilaufgaben Strukturdefekte aufweisen und somit schlechtstrukturierte Probleme darstellen. Zur Vereinfachung werde angenommen, dass keine stochastische Unsicherheit vorliege und dass das Ergebnis einer Teilaufgabe der PPS die Eingabe für eine direkt darauffolgende Teilaufgabe eindeutig determiniere. Würde stochastische Unsicherheit einbezogen, läge für nahezu jedes Teilproblem der PPS zumindest ein Bewertungsdefekt vor (siehe Tabelle 2.1), und fast alle Teilaufgaben der PPS wären schlechtstrukturierte Probleme.

Die folgenden Anstriche legen dar, weshalb die einzelnen Teilaufgaben der PPS Strukturdefekten unterliegen oder nicht, und begründen somit, ob eine Teilaufgabe der PPS ein schlechtstrukturiertes Problem verkörpert oder wohlstrukturiert ist:

- *Primärbedarfsplanung*: Im Rahmen der Primärbedarfsplanung unterliegt die *Primärbedarfsprognose* einem Bewertungs- und Wirkungsdefekt, weil zum einen nicht alle Variablen, die den Primärbedarf beeinflussen, (in ihren (zukünftigen) Ausprägungen) bekannt sind und weil zum anderen die Wirkung (der Ausprägungen) dieser Variablen – sofern bekannt – oftmals ebenfalls unbekannt ist. Die *Grobplanung* weist hingegen keinen Strukturdefekt auf und ist somit wohlstrukturiert. Sie hat meist maximalen Gewinn bzw. maximalen Deckungsbeitrag zum Ziel [Sch00, S. 227, S. 237], [Ada98, S. 221 ff.] und kann mittels linearer Optimierung vorgenommen werden [Mer04, S. 135], [Ada96, S. 460 ff.].

- *Mengenplanung*: Im Kontext der Mengenplanung ist die *bedarfsgesteuerte Materialbedarfsplanung* wohlstrukturiert. Sie erfolgt hauptsächlich

Tabelle 2.5: Wohlstrukturierte Teilaufgaben und strukturdefekte Probleme der PPS, angelehnt an [CM96, S. 249]: X bedeutet, dass der entsprechende Strukturdefekt für die Teilaufgabe der PPS im Allgemeinen vorliegt. Liegt kein Strukturdefekt pro Teilaufgabe vor, wird sie als wohlstrukturiert charakterisiert.

	Teilaufgabe der PPS			Strukturdefekt			
				Bewertung	Wirkung	Zielsetzung	Lösung
	Primärbedarfsplanung	Primärbedarfsprognose		X	X		
		Grobplanung		wohlstrukturiert			
	Mengenplanung	Materialbedarfsplanung	bedarfsgesteuert	wohlstrukturiert			
			verbrauchsgesteuert	X	X		
		Auftragsplanung					X
	Terminplanung	Arbeitsplanauflösung		wohlstrukturiert			
		Durchlaufterminierung					
	Kapazitätsplanung	Kapazitätsbedarfsrechnung					
		Kapazitätsterminierung				X	X
	Auftragsfreigabe	Verfügbarkeitsprüfung		wohlstrukturiert			
		Reihenfolgeplanung				X	X
	Produktionsregelung	Produktionsüberwachung		wohlstrukturiert			
		Eingriffe in die Produktion				X	X

mittels Stücklistenauflösung, für welche effiziente Algorithmen vorliegen. [Kur03, S. 128], [Sch95, S. 139] Die *verbrauchsgesteuerte Materialbedarfsplanung* weist hingegen in Analogie zur Primärbedarfsprognose einen Bewertungs- und Wirkungsdefekt auf: Variablen, ihre Ausprägungen und Zusammenhänge zwischen ihnen sind nicht vollständig bekannt. Die *Auftragsplanung* bzw. *Losgrößen- und Bestellmengenplanung* kann in ihrer allgemeinen Form – bei mehrstufiger Produktion, mehreren Produkte und dynamischer Nachfrage – als lösungsdefekt angesehen werden. [CM96, S. 249] Bereits für die einstufige Zweiproduktfertigung bei dynamischer Nachfrage ist die Auftragsplanung NP-schwer. [DSV97, S. 135 f.]

- *Terminplanung*: Für die *Arbeitsplanauflösung* und die *Durchlaufterminierung* im Rahmen der Terminplanung sind in der Regel alle relevanten Daten – Arbeitspläne, Rüst- und Bearbeitungszeiten pro Arbeitsgang und Abhängigkeiten zwischen den Aufträgen – bekannt. Bei der Durchlaufterminierung handelt es sich zudem *nicht* um ein (kombinatorisches) Optimierungsproblem, und es liegen effiziente Algorithmen für sie vor [Kur03, S. 141 ff.], [Sch00, S. 252 f.], weshalb die gesamte Terminplanung als wohlstrukturiert gilt. [CM96, S. 250]

- *Kapazitätsplanung*: Im Kontext der Kapazitätsplanung handelt es sich bei der *Kapazitätsbedarfsrechnung* um eine wohlstrukturierte Teilaufgabe der PPS [CM96, S. 250], da sie sich lediglich auf die Summation des laut Terminplan pro Kapazitätseinheit und pro Periode benötigten Kapazitätsbedarfes beläuft. Nimmt die *Kapazitätsterminierung* nicht nur einen einfachen Kapazitätsabgleich vor, sondern wird als Optimierungsproblem mit Zielfunktion und Nebenbedingungen aufgefasst, ist sie zielsetzungs- und lösungsdefekt [CM96, S. 250], da für sie keine ad hoc operationale Zielfunktion existiert und sie im Allgemeinen als Zuordnungs- bzw. Maschinenbelegungsproblem [Kur03, S. 166], [Ada98, S. 605] aufzufassen ist, welches für die meisten realen Probleme NP-schwer ist [Sch01, S. 155], [DSV97, S. 372, S. 397, S. 427].

- *Auftragsfreigabe*: Geht man von einem zulässigen Termin- und Kapazitätsplan aus, stellt die der Auftragsfreigabe untergeordnete *Verfügbarkeitsprüfung* eine wohlstrukturierte Teilaufgabe der PPS dar [CM96, S. 250] – auch dann, wenn sie vorausschaut [Mer04, S. 156 f.], [Ada98, S. 605]. Die *Reihenfolgeplanung* hingegen ist – als Optimierungsproblem betrachtet – strukturdefekt. Sie weist als Maschinenbelegungsproblem in Analogie zur Kapazitätsterminierung (siehe oben) einen Zielsetzungs- und Lösungsdefekt auf.

- *Produktionsregelung*: Eine umfassende und funktionierende BDE vorausgesetzt, ist die *Produktionsüberwachung*, die im Rahmen der Produktionsregelung stattfindet, wohlstrukturiert. Wenn über die Produktion alle relevanten Daten vorliegen, können kurzfristig vorgenommene *Eingriffe in die Produktion* im Hinblick auf ihre Wirkung effizient prognostiziert werden – z. B. mittels ereignisdiskreter Simulation.[16] Strebt man aber einen optimalen Maschinenbelegungsplan an, der die Eingriffe einschließt, treten wiederum Zielsetzungs- und Lösungsdefekt auf.

Die Teilaufgaben der PPS und vor allem die Probleme, die in der PPS auftreten, sind relativ umfassend und unter verschiedenen Aspekten beleuchtet worden. Kapitel 3 geht nun darauf ein, wie bisher versucht worden ist, die Probleme der PPS mittels alternativer Verfahren zu lösen.

[16] Im Gegensatz zu [CM96, S. 249 ff.] wird also kein Wirkungsdefekt attestiert.

Kapitel 3

Analyse alternativer Verfahren zur Unterstützung der PPS

Bei der Analyse alternativer Verfahren zur Unterstützung der PPS soll nicht auf solche Verfahren eingegangen werden, welche von Grund auf und direkt für die PPS entwickelt worden sind, sondern auf universelle Verfahren, die man auch auf Probleme anwenden kann, welche nicht der PPS entstammen, aber gleichen Problemklassen angehören. Zum einen sind PPS-spezifische Verfahren bereits oft besprochen worden[1], zum anderen ist ein Vergleich universeller Verfahren, wie z. B. BNe, und problemspezifischer Verfahren oft nicht sinnvoll und nur anhand von Beispielen punktuell möglich. Darüber hinaus ist keineswegs sicher, dass problemspezifische Verfahren bessere Ergebnisse zeitigen als universelle, welche an das zu lösende Problem angepasst worden sind.

Die für dieses Kapitel zur Analyse ausgewählten Verfahren zur Unterstützung der PPS entstammen dem Softcomputing und der Wissensverarbeitung und sind auf diesen Gebieten die renommiertesten Verfahren.

3.1 Universelle Heuristiken

Laut [MM92, S. 290] handelt es sich bei Heuristiken um Verfahren, die „Vorgehensregeln“ anwenden, um ein Problem zu lösen. Die Regeln seien hinsichtlich der Struktur des Problems und bezüglich der Zielfunktion „sinnvoll,

[1] Für einen Überblick über PPS-spezifische Verfahren, wie z. B. Heuristiken zur Bedarfs-, Losgrößen- und Bestellmengenplanung, belastungsorientierte Auftragsfreigabe, retrograde Terminierung, Optimized Production Technology (OPT) oder Kanban, sei z. B. auf [Sch00, S. 246 ff.], [Cor04, S. 407 ff., S. 547 ff.] und [Ada98, S. 476 ff., S. 621 ff.] verwiesen.

zweckmäßig und erfolgversprechend“. Heuristiken garantieren keine optimalen Lösungen, und der mit ihnen verbundene Rechenaufwand sei eher gering. Letzterer Aussage wird – vor allem im Lichte der nachfolgenden Ausführungen – zumindest nicht uneingeschränkt zugestimmt.

Heuristiken lassen sich laut [DSV97, S. 44 ff.] in Eröffnungs- und Verbesserungsverfahren[2] unterteilen.[3] Die *Eröffnungsverfahren* erstellen eine zulässige Lösung, indem sie schrittweise Entscheidungsvariablen einbeziehen und auf geeignete Werte setzen. Bei jedem Schritt sucht das Eröffnungsverfahren diejenigen Entscheidungsvariablen und setzt sie auf diejenigen Werte, die zum besten Zielfunktionswert führen, wobei die Zielfunktion nur die bisher einbezogenen Entscheidungsvariablen berücksichtigt. Eröffnungsverfahren gehen prinzipiell nie einen Schritt zurück, das heißt, dass sie eine Lösung nur auf- und nie wieder abbauen – auch nicht partiell. [GKK04, S. 23 f.] Die Eröffnungsverfahren blicken deshalb oft mehr als einen Schritt voraus[4], um nicht Gefahr zu laufen, durch Kurzsichtigkeit eine Lösung zu verhindern, welche sich im weiteren Verlauf des Verfahrens als besser herausstellen würde. Die bekanntesten Eröffnungsverfahren sind das „Verfahren des besten Nachfolgers“ und das „Verfahren der sukzessiven Einbeziehung“ [MM92, S. 294 ff.], welche vorrangig auf das „Problem des Handlungsreisenden“ angewandt werden, aber – in modifizierter Form – auch auf Reihenfolge- und Zuordnungsprobleme, wie sie z. B. bei der Maschinenbelegungsplanung auftreten.

Die *Verbesserungsverfahren* basieren zumeist auf einer zufällig oder mittels eines Eröffnungsverfahrens erstellten, zulässigen Initiallösung. [DSV97, S. 45] Sie verbessern diese Lösung iterativ, indem sie logisch zusammengehörende Entscheidungsvariablen auswählen und ihre Werte – innerhalb eher enger Grenzen – so modifizieren, dass eine neue zulässige Lösung entsteht, welche tendenziell einen besseren Zielfunktionswert aufweist als die vorangegangene. Zu diesem Zwecke untersuchen Verbesserungsverfahren mehrere, der vorangegangenen Lösung räumlich nahe[5] potenzielle Lösungen und wählen deren beste oder zumindest die erste bessere aus. Um sich bei stark

[2] Letztere werden auch als „suboptimierende Iterationsverfahren“ bezeichnet. [MM92, S. 298 ff.]

[3] Auf unvollständig ausgeführte exakte Verfahren und auf Verfahren, welche auf Relaxation der Nebenbedingungen basieren, soll nicht näher eingegangen werden.

[4] Eröffnungsverfahren, die nur einen Schritt vorausblicken, werden als gierig (englisch: greedy) bezeichnet. Verfahren, welche k Schritte vorausblicken, werden auf englisch als k-step-look-ahead bezeichnet. [Alm95, S. 169 ff.]

[5] Aufgrund der Suche in der Nähe der vorangegangenen Lösung werden die Verbesserungsverfahren im deutschen Sprachraum auch als *lokale Suchverfahren* bezeichnet [Dör04, ca. S. 65], wobei die Bezeichnung irreführend ist, da nicht die Verfahren lokal sind, sondern die Suche lokal erfolgt, so dass sie besser als *Verfahren der lokalen Suche* bezeichnet würden.

zerklüfteten Zielfunktionen nicht in einem lokalen Optimum zu verfangen, akzeptieren Verbesserungsverfahren – im Gegensatz zu den Eröffnungsverfahren – z. T. auch eine temporäre Verschlechterung des Zielfunktionswertes, merken sich aber immer die bisher beste Lösung. Verbesserungsverfahren enthalten oft eine stochastische Komponente, welche dazu dient, Entscheidungsvariablen zur Modifikation zu identifizieren, und vor allem dazu, ein lokales Optimum zu verlassen. [LMS03] Die bekanntesten Verbesserungsverfahren sind *simuliertes Abkühlen*[6] [Wen95], [Brü95, S. 26 ff., 165 ff., 270 ff.][7], *Tabu-Suche* [GL01], Algorithmus der *Schwellwertakzeptanz*[8] [Pla96], [DS90] sowie der *Sintflutalgorithmus*[9] [Due93], [DSW93]. Den grundlegenden Ablauf von Verbesserungsverfahren stellen [Stü99] und [AL98] dar – z. T. auch in Pseudoquelltext auf hohem Abstraktionsniveau. [AKL98] und [HTd98] gehen besonders auf simuliertes Abkühlen respektive Tabu-Suche ein. Für Eröffnungs- und Verbesserungsverfahren liegt eine Vielzahl von Anwendungen im Rahmen der PPS vor. Zwei interessante stellen [Sch03] und [HH97] vor, die zwar gute Lösungen, aber sehr lange Laufzeiten verursachen.

Eröffnungs- und Verbesserungsverfahren weisen Vor- und Nachteile auf: Eröffnungsverfahren verursachen geringen Rechenaufwand, weil sie Rückschritte nicht erlauben und meist nur einen oder wenige Schritte vorausschauen [DSV97, S. 45], wodurch sie den Suchraum stark einschränken. Verbesserungsverfahren verursachen ebenfalls relativ geringen Rechenaufwand. Dieser Umstand liegt darin begründet, dass Verbesserungsverfahren im Gegensatz zu Evolutionären Algorithmen (EAen, siehe Abschnitt 3.2) lediglich eine Lösung (ein Individuum) durch den Suchraum bewegen und keine Population von Lösungen verwalten müssen [Wei02, S. 156], weshalb sie auch nur Speicherplatz für eine Lösung und Rechenzeit für die Modifikation und die Bewertung einer Lösung beanspruchen. Dass sich Verbesserungsverfahren bei ihrer Suche größtenteils lokal orientieren, schränkt den Suchraum ein und trägt ebenfalls zur Reduktion des Rechenaufwandes bei.

Der größte Nachteil von Eröffnungs- und Verbesserungsverfahren besteht darin, dass bei stark zerklüfteter Zielfunktion mit vielen lokalen Optima ihre Lösungsgüte eher gering ist. [Wei02, S. 162] Dieser Umstand liegt im lokalen Charakter ihrer Suche begründet. Es besteht die Gefahr, dass sie sich in einem (dem als ersten gefundenen) lokalen Optimum verfangen, welches oft empfindlich vom globalen abweicht. Zudem hängt ihre Lösungsgüte stark von der Initiallösung ab: Gelangt man von der Initiallösung mehr oder minder direkt

[6] in englischer Sprache: Simulated Annealing

[7] [Brü95, S. 165 ff., 270 ff.] beschreibt simuliertes Abkühlen zur Werkstattfertigung und wendet es zur simultanen Losgrößen- und -reihenfolgeplanung an.

[8] in englischer Sprache: Threshold Accepting

[9] in englischer Sprache: Great Deluge Algorithm

zum globalen Optimum oder zu einem „guten“ lokalen Optimum, erzielt man eine gute Lösung; startet man jedoch im Einzugsgebiet eines „schlechten“ lokalen Optimums, fällt die Lösung schlecht aus. Dass Verbesserungsverfahren stark von ihren Verfahrensparametern abhängen, ist per se noch kein Nachteil. Die aufwendige und schwierige Wahl der Verfahrensparameter zählt hingegen schon zu den Nachteilen von Verbesserungsverfahren. Die Parameter sollen so gewählt werden, dass das Verfahren einerseits schnell, aber andererseits nicht bereits in einem „schlechten“ lokalen Optimum konvergiert. So sind gute Werte für z. B. die Länge und die Art der Tabu-Liste bei der Tabu-Suche, das Abkühlungsschema beim simulierten Abkühlen, die Schrittweite beim Sintflutalgorithmus oder ganz allgemein für ein gutes Akzeptanz- und ein gutes Abbruchkriterium nur aufwendig zu bestimmen. Zudem hängen die Verfahrensparameter stark vom Problem oder sogar von der speziellen Probleminstanz ab [Dör04, S. 3, S. 18 ff.], so dass sie für jedes neue Problem oder in letzterem Falle sogar für jedes Anwenden des Verfahrens auf eine weitere Probleminstanz neu zu ermitteln bzw. anzupassen sind. Simuliertes Abkühlen erhält sogar das Attribut „launisch“ [Wei02, S. 156] bezüglich der Abhängigkeit seiner Lösungsgüte vom Problem und von seinen Parametern.

3.2 Evolutionäre Algorithmen

Evolutionäre Algorithmen (EAen) dienen der Lösung von Optimierungsproblemen. Sie orientieren sich in ihrer Arbeitsweise an den Mechanismen der natürlichen Evolution [Kur03, S. 360] und fußen hauptsächlich auf den folgenden drei Operatoren: der *Selektion*, der *Rekombination* und der *Mutation* [GKK04, S. 3]. Datenstrukturen für EAen und den Ablauf EAen beschreiben [Rot02], [Mic96] und [SHF94].

Die grundlegenden Vertreter der EAen sind die *genetischen Algorithmen* (GAen) und die *Evolutionsstrategien* (ESen). [Ort02, S. 10], [Kur99, S. 28] GAen gehen auf Holland und Goldberg zurück, siehe z. B. [Hol75], [Gol89], und ESen haben Rechenberg und Schwefel entwickelt, siehe z. B. [Rec65], [Sch75]. GAen lösen vorwiegend diskrete bzw. ganzzahlige kombinatorische Optimierungsprobleme, wie sie in der PPS vorliegen, und verwenden vordergründig die Rekombination von Individuen. ESen hingegen suchen, reellwertige Optimierungsprobleme zu lösen, und basieren auf der Mutation von Individuen. Ausgewählte Anwendungen EAen in der PPS stellen [Ala95] und [Nis94, S. 295 ff.] zusammen. Zwei aktuelle Anwendungen präsentieren [Dud05] und [OV04].

Zu den Vorteilen EAen zählt, dass sie nahezu universell auf Optimierungsprobleme anwendbar sind, welche sich aufgrund stark zerklüfteter Zielfunk-

tionen oder einer Vielzahl komplexer Nebenbedingungen einer erfolgreichen mathematischen Optimierung entziehen. [SHF94, S. 13], [Gol89, S. 7 ff.] Zudem ist ihr grundlegender Ablauf plausibel, wenn auch für einen Experten nicht im Detail nachzuvollziehen [Wei02, S. 163]. Zum Nachteil gereicht den EAen, dass sie nicht zwangsläufig ein globales Optimum finden, sondern sich meist in lokalen Optima verfangen, deren Funktionswerte von denen des globalen Optimums oft empfindlich abweichen. [GKK04, S. 61], [Ort02, S. 14] Haben EAen für ein bestimmtes Problem eine Lösung hoher Qualität erzielt, so kann dieselbe Lösungsqualität auch bei ähnlichen Problemen derselben Problemklasse nicht garantiert bzw. nicht zwangsläufig reproduziert werden. Die Praxis akzeptiert EAen deshalb nur schwer. [Wei02, S. 163] EAen bilden zudem nicht die der Produktion inhärente Unsicherheit (siehe Abschnitt 2.3) ab und lassen keine Aussage darüber zu, ob eine erzielte Lösung robust gegenüber geringfügigen Änderungen der Eingabedaten ist. Außerdem akzeptieren sie keine (informationelle) Unsicherheit in den Eingabedaten. Darüber hinaus verursacht jedes neue Problem, welches man mit EAen lösen möchte, den Aufwand, eine passende Datenstruktur für die Repräsentation eines Individuums zu entwerfen bzw. ein Argument der Zielfunktion so zu kodieren, dass es in eine gegebene Datenstruktur passt. [Wei02, S. 169 f.], [Rot02, S. 9 ff.] EAen haften außerdem viele Parameter an, welche für neue Probleme in einen aufwendigen Trial&Error-Prozess eingestellt werden müssen [GKK04, S. 63], [ES03, S. 129 ff.] – vor allem dann, wenn man in ihrer Endphase noch Verbesserungen erreichen möchte [Wei02, S. 163]. Nicht zuletzt verursachen EAen einen sehr hohen Rechenaufwand und somit lange Rechenzeiten [Wei02, S. 163], [Ort02, S. 6] – vor allem dann, wenn die Rekombination, die Mutation, das Überprüfen des Einhaltens der Nebenbedingungen, die Reparatur oder das Errechnen des Zielfunktionswertes der Individuen aufwendige Operationen sind und große Populationen stark erneuert werden und das über viele Generationen hinweg. [RR05] Angenommen, ein EA solle einen Maschinenbelegungsplan erstellen, die Mächtigkeit der Population betrage 100 Individuen (Maschinenbelegungspläne), der Algorithmus laufe über 1.000 Iterationen (Generationen), und pro Iteration werde die Hälfte der Population neu erzeugt. Dann wäre 50.000 Mal zu selektieren, zu rekombinieren, zu mutieren, das Einhalten der Nebenbedingungen zu überprüfen, gegebenenfalls zu reparieren und der Zielfunktionswert zu ermitteln.

3.3 Künstliche neuronale Netze

Künstliche Neuronale Netze (KNNe) orientieren sich in Aufbau und Arbeitsweise am Gehirn. Ihr Einsatz kommt in Frage, wenn komplexe Probleme mit

herkömmlichen Mitteln nicht oder nur unzureichend gelöst werden können, wenn aber die Daten vorliegen, von denen man erwartet, dass sie zur Lösung besagter Probleme notwendig und ausreichend sind. [WG90, S. 152] Da in der PPS viele solcher Probleme – seien es Prognose-, Diagnose-, reine Mustererkennungs-, Optimierungs- oder Planungsprobleme – auftreten und entsprechende Daten in PPS-, ERP- oder BDE-Systemen vorliegen, bieten sich KNNe grundsätzlich als Lösungsverfahren an. Zu Aufbau und Funktionsweise KNNe liegen zahlreiche Quellen vor. Auf einige, ausgewählte sei an dieser Stelle verwiesen: [Cal03], [Vid03], [Zel00], [Hay99], [AB99], [SHG90].

Einen Überblick über den Einsatz KNNe in der PPS geben [CM96]. Eine Literaturstudie zum Einsatz KNNe in Betriebswirtschaft und Operations Research stammt von [SR98]. Zwei neuere Anwendungen sogenannter *Constraint Satisfaction Adaptive Neural Networks* (CSANNs) stellen [YW01] und [WAS04] vor. Die praktische Relevanz KNNe in der PPS ist zwar eher gering, aber die Qualität der Prognose mittels KNNe in anderen Zweigen der Betriebswirtschaft überzeugt durchaus. [Lan04, S. 233 ff.]

KNNe adaptieren Beispieldaten, was einen Vorteil darstellt. Im Idealfall generalisieren KNNe sogar, dass heißt, sie extrahieren Wissen aus Beispieldaten, verallgemeinern es, stellen es implizit dar und wenden es erfolgreich auf ihnen bisher unbekannte Daten an. [Loh94, S. 1] Unter letzterem Aspekt stellen auch verrauschte Eingangsdaten für KNNe ein lösbares Problem dar. [Rig94, S. 7 f.] Die Eigenschaft, aus Beispieldaten zu lernen, haben KNNe allerdings mit anderen Modellen der Wissensverarbeitung gemein; so werden Entscheidungsbäume und Regeln aus Daten induziert [Qui93] und BNe aus Daten erstellt. Als weiterer Vorteil KNNe wird ihre massive Parallelität angeführt, welche – sofern von der Hardware unterstützt – KNNe im Gegensatz zu sequentiell arbeitenden von Neumann-Rechnern beschleunigt. [SHG90, S. 13 f.] Aber selbst wenn man den mit zunehmender Größe des KNNes überproportional steigenden Aufwand zur Kommunikation zwischen den Neuronen außer acht lässt [BL93, S. 77 f.], lohnt der Aufwand zum Entwurf spezieller Hardware für KNNe im Allgemeinen nicht[10], da Einprozessorrechner immer schneller werden und obwohl [Kra93, S. 16] gegenteiliges behauptet. KNNe verhalten sich auch robust gegenüber dem Ausfall einzelner Neuronen oder Verbindungen. [BL93, S. 95] Im Kontext betriebswirtschaftlicher Anwendungen kommen die Parallelität KNNe und ihre Robustheit ge-

[10]So vertritt Teuvo Kohonen die Ansicht, dass der Aufwand für das Entwickeln spezieller KNNe-Hardware nicht gerechtfertigt sei, da Ein-Prozessor-Rechner aufgrund ihrer rasanten Entwicklung spezielle KNNe-Hardware jeweils innerhalb kurzer Zeit überholen. Herr Kohonen hat die *Selbstorganisierenden Karten* [Koh01] entwickelt, einen speziellen Typ KNNe, und obige Aussage bereits im Jahre 1998 während einer Podiumsdiskussion auf der EUFIT in Aachen getroffen.

genüber Ausfällen ihrer Komponenten jedoch nicht zum Tragen.

Den Vorteilen KNNe stehen nicht unerhebliche Nachteile gegenüber: Zur Auswahl des für die zu lösende Aufgabe günstigsten Typs eines KNNes stehen – wenn überhaupt – nur Richtlinien und Präzedenzfälle zur Verfügung. [Cal03, S. 155 f.] Die Frage, welcher KNN-Typ sich für ein bestimmtes Problem am besten eignet, wird wohl auch in näherer Zukunft nicht umfassend beantwortet werden. [Zel00, S. 418] Viele Parameter KNNe können ebenfalls nicht a priori optimal bestimmt werden, so z. B. die Anzahl der Schichten, die Art und die Anzahl der Neuronen (pro Schicht [Cal03, S. 156]), ob eine Verbindung zwischen zwei Neuronen existiert, die initialen Gewichte der Verbindungen, die Lernregeln und die Lernrate. Man ist gezwungen, eine aufwendige Trial-And-Error-Methode anzuwenden [CM96, S. 245], [Loh94, S. 17] und die Parameter zufällig zu initialisieren [Zel00, S. 419]. Funktionsbildende KNNe bedürfen eines aufwendigen und mehrfachen Trainings, um beim späteren Einsatz ein erwünschtes Verhalten aufzuweisen. [Hay99, S. 64], [Kra93, S. 165] KNNe konvergieren zum Teil nur sehr langsam [Hay99, S. 190], [Zel00, S. 28] oder oszillieren [Zel00, S. 112], [Koh01, S. 183], [Kra93, S. 166]. Es besteht die Gefahr, dass sich KNNe beim Lernen in einem lokalen Minimum der Energiefunktion verfangen [Hay99, S. 191], [Zel00, S. 112, S. 207], [Mun95, S. 41], was zu einer Lösung führt, deren Zielfunktionswert z. T. erheblich vom globalen Optimum abweicht. Wendet man Algorithmen an, die es ermöglichen, lokale Minima wieder zu verlassen, steigt der Rechenaufwand drastisch. [CM96, S. 247] KNNe unterliegen zudem dem Phänomen des „Overfitting“. Sie lernen die Trainingsdaten auswendig und assoziieren bekannte Eingaben perfekt mit den Ausgaben. Bei unbekannten Eingaben versagen sie jedoch. [Vid03, S. 416 ff.], [Kra93, S. 148]

Die Kritik an KNNen fokussiert bisher auf Lösungsgüte und Rechenaufwand. Gerade im Rahmen der PPS ist aber auch von Bedeutung, ob der Anwender ein Modell bzw. eine Methode akzeptiert. Im Allgemeinen ist schwer zu beschreiben und kaum nachzuvollziehen, wie KNNe Wissen aus Daten akquirieren, wie sie das Wissen repräsentieren [BL93, S. 95], und wie sie das Wissen anwenden, um zu Lösungen zu gelangen. [Zel00, S. 27 ff.] Demzufolge stoßen KNNe oft auf Akzeptanzprobleme. [CM96, S. 237]. Dass KNNe im Allgemeinen keine Aussage darüber zulassen, wie gut die von ihnen dargebotenen Lösungen in Relation zur optimalen Lösung sind und ob sie ähnliche Probleme ähnlich gut lösen, trägt nicht zu ihrer Akzeptanz im Kontext der PPS bei.

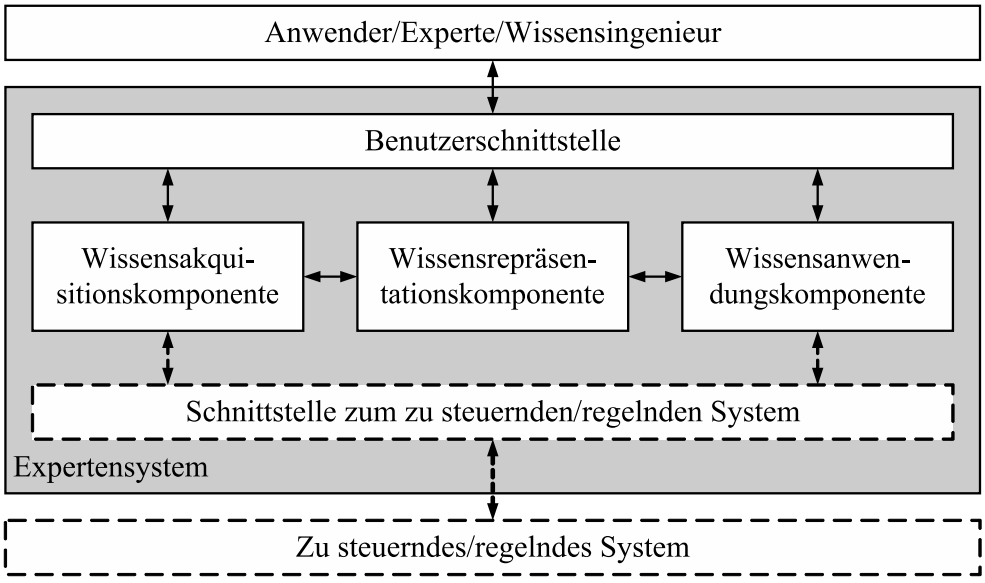

Abbildung 3.1: Vereinfachte Komponentenarchitektur eines Expertensystems

3.4 Regelbasierte Expertenysteme

Expertensysteme (XPSe) akquirieren das Wissen von Experten, repräsentieren es so, dass es maschinell verarbeitet werden kann, und wenden es zur Lösung von Problemen an. Die drei Grundaufgaben von XPSen – Wissensakquisition, -repräsentation und -anwendung – induzieren die drei Hauptkomponenten von XPSen – eine Komponente zur Akquisition, eine Basis zur Repräsentation und eine Inferenzmaschine zur Anwendung des Wissens. Abbildung 3.1 stellt nicht nur die Architektur eines XPS dar, sondern anhand ihr können auch das Erstellen und das Anwenden von XPSen erklärt werden: Zu Beginn übergeben der Anwender und/oder der Wissensingenieur der Wissensrepräsentationskomponente Wissen – bei XPSen in Form von Regeln. Die Wissensakquisitionskomponente gewinnt die Regeln alternativ aus Daten des zu steuernden/regelnden Systems und übergibt sie der Wissensrepräsentationskomponente. Der Anwender versorgt nun die Wissensanwendungskomponente mit Fakten und fragt unbekannte Fakten oder auszuführende Aktionen nach. Die Wissensanwendungskomponente schlussfolgert aufgrund der Regeln und aufgrund der eingegebenen Fakten und antwortet dem Anwender. Optional versorgt das zu steuernde/regelnde System die Wissensanwendungskomponente mit Fakten und führt geschlussfolgerte Aktionen aus.

Im Gegensatz zu herkömmlichen Programmen, die das Wissen in Algorithmen und Datenstrukturen fest kodieren, trennen XPSe das Wissen strikt

vom Programm zu seiner Verarbeitung, so dass das Wissen erweitert oder editiert werden kann, ohne das XPS zu ändern. [Fei92, S. 8]

Die meisten XPSe basieren auf Regeln. [HSA99, S. 121], [Pup91, S. 21] Das Wissen wird in der Wissensbasis ergo in Form von Regeln repräsentiert. Regeln weisen die folgende Form auf:

Wenn <Prämisse>, **dann** <Konklusion>. (3.1)

Prämisse und Konklusion entsprechen jeweils einem logischen Ausdruck. Ein logischer Ausdruck besteht aus Aussagen und logischen Operatoren. Die Aussagen ordnen Variablen Werte zu, und die logischen Operatoren verknüpfen die Aussagen miteinander. Die folgende Gleichung sei ein Beispiel für eine Regel:

Wenn Lagerbestand=hoch, **dann** Durchlaufzeit=hoch. (3.2)

Sei diese Regel auch umstritten: Ein Experte möge sie so angegeben haben. Die Aussage in der Konklusion einer Regel kann auch eine Aktion sein, eine Anweisung, wie gehandelt werden soll:

Wenn Verspätung=stark, **dann** Prioritätsregel=KOZ. (3.3)

„Wenn starke Verspätung vorliegt, dann wähle die Prioritätsregel ‚Kürzeste Operationszeit' (KOZ)". – Eine Regel in der Wissensbasis des Expertensystems wird auch als abstraktes Wissen bezeichnet, wohingegen ein Fakt, dass eine bestimmte Variable einen bestimmten Zustand bzw. Wert annimmt, als konkretes Wissen bezeichnet wird. [CGH97, S. 22]

Die Wissensakquisition erfolgt manuell oder automatisch. Bei manueller Wissensakquisition formulieren Experten selbst Regeln und geben sie in ein XPS ein. „Wissensingenieure" unterstützen die Experten bei der Wissensakquisition. [Tim94, S. 20] Bei automatischer Wissensakquisition werden Regeln algorithmisch aus Beispielfällen generiert, welche Experten explizit angeben oder implizit erzeugen. Die bekanntesten Algorithmen zur automatischen Wissensakquisition heißen ID3[11] [HMS66] und C4.5[12] [Qui93]. Einen allgemeinen Algorithmus zum automatischen Erstellen eines Entscheidungsbaumes aus Beispieldaten stellt [BKI03] dar. Es ist ersichtlich, dass dieser Algorithmus keine gleichen Fälle mit unterschiedlichen Klassen zulässt; treten solche Fälle auf, scheitert er. Er akzeptiert ergo keine stochastische und

[11] ID steht für Iterative Dichotomizer (iterativer „Zweiteiler"): Der ID3 „zweiteilt" wiederholt die Menge(n) der Beispielfälle und erzeugt so peu à peu einen binären Entscheidungsbaum.

[12] Mittlerweile liegt eine Version C5.0 vor. [Int97, S. 63 ff.]

keine informationelle Unsicherheit (siehe Abschnitt 2.3.1). Selten gelingt es, den Attributen weitere hinzuzufügen und die Beispielfälle so bezüglich der ihnen zugeordneten Klassen eindeutig zu gestalten. Zudem dient der Algorithmus von [BKI03] dem Aufbau eines Klassifikators, welcher Fälle lediglich klassifiziert. Außerdem bedarf dieser Algorithmus explizit vorgegebener Klassen. Er bildet keine Kausalität ab und verwendet unbedingte Häufigkeiten bei der Auswahl des Attributes zum Aufteilen der Beispieldaten.

Die Methoden der automatischen Wissenakquisition haben den Vorteil, dass durch ihren Einsatz die Effektivität bei der Generierung von Wissen wesentlich höher ist als bei der direkten Akquisition von Wissen aus Expertenaussagen. In der Praxis besitzen sie jedoch eher untergeordnete Bedeutung [Pup88, S. 23 f.], da die verwendeten Fallbeispieldatensätze einen Problemraum meist nur unvollständig abdecken, die verwendeten Datensätze oft nicht korrekt [Len91, S. 199] und die generierten Regeln sehr komplex und demzufolge nur schwer nachvollziehbar sind [Mer90, S. 36]. Außerdem lässt sich mit Hilfe der automatischen Wissensakquisition bisher nur Regelwissen ableiten. [Len91, S. 199]

Die Modi und Algorithmen zur Anwendung des akquirierten Wissens in regelbasierten Expertensystemen beschreiben [BKI03], [HSA99], [CGH97] und [Pup91].

Der Vollständigkeit halber seien noch zwei weitere Komponenten von XPSen erwähnt: Die Erklärungskomponente legt dar, warum, auf welchen Wegen bzw. mittels welcher Regeln eine Lösung geschlussfolgert worden ist, und die Kohärenz- bzw. Konsistenzkontrolle stellt sicher, dass dem regelbasierten XPS keine einander widersprechenden Regeln und Fakten eingegeben werden. Kohärenz- bzw. Konsistenz können bei der Eingabe der Regeln von der Wissensakquisitionskomponente und bei der Eingabe der Fakten von der Komponente zur Wissensanwendung kontrolliert werden, und grundlegende Erklärungen entspringen der Komponente zur Wissenanwendung, so dass nicht zwingend zwei extra Komponenten vorliegen müssen [BKI03, S. 86].

Die meisten Beispiele für den Einsatz von XPSen in der bzw. zur PPS stammen aus den späten 80er und frühen 90er Jahren des vergangenen Jahrhunderts und werden vorwiegend zur Planung (siehe z. B. [Möl95], [Mon94] und [Cre91]) und auch zur Diagnose (siehe z. B. [Kom93, S. 12]) eingesetzt. [Ste93] erwägt Einsatzmöglichkeiten von XPSen in PPS-Systemen. In [Mer90] finden sich Beispiele für den Einsatz von XPSen in der Produktion im deutschsprachigen Raum. Einen guten Überblick über den Einsatz von XPSen im internationalen Raum gibt [MAP02].

Regeln sind von Experten relativ leicht anzugeben und zu verstehen. Allerdings weisen regelbasierte XPSe – zusätzlich zu den bereits aufgeführten – eine Reihe gravierender Nachteile auf:

- Es fällt dem Experten, der kein Experte für Wissensakquisition, sondern „lediglich“ Experte auf seinem Problemgebiet ist, in der Regel schwer, konsistente, widerspruchsfreie Regeln anzugeben. Die Wissenakquisition gilt als Engpass/Flaschenhals beim Einsatz von XPSen [Tim94, S. 20 f.], ist also sehr schwierig und aufwendig.

- Sind während der Wissensanwendung mehrere Regeln anwendbar, tritt also ein sogenannter Regel*konflikt* auf, fällt es schwer, automatisch zu entscheiden, welche der Regeln bzw. in welcher Reihenfolge die Regeln anzuwenden sind. [HSA99, S. 129 ff.]

- Regeln können sich widersprechen und somit nicht induziert oder verarbeitet werden, oder die Inferenzmaschine schlussfolgert widersprüchlich [BKI03, S. 85], auch wenn in der Realität für alle Regeln Beispiele vorliegen.

- Gegebene Fakten können sich unter Berücksichtigung der vorliegenden Regeln widersprechen und somit nicht verarbeitet/eingegeben werden, auch wenn die Kombination von Fakten in der Realität vorkommt.

- Regelbasierte XPSe verarbeiten nicht per se und nicht theoretisch fundiert Unsicherheit [BKI03, S. 93], woraus auch die zwei vorangehenden Nachteile resultieren. Da sie lediglich deterministisches Wissen verarbeiten, können sie für viele potenzielle Anwendungsfälle in der Praxis nicht eingesetzt werden, da diese Anwendungsfälle stochastischen Einflüssen unterliegen. [CGH97, S. 65]

Da insbesondere in der PPS komplexe Anwendungsfälle auftreten, die auch Experten nicht oder nur unzureichend überblicken, und diese Anwendungsfälle auch noch mit Stochastik behaftet sind, bietet sich der Einsatz regelbasierter XPSe in der PPS nicht grundsätzlich an. [Kur03, S. 359] empfiehlt, das Wissen beim Experten zu belassen und ihn das Problem lösen zu lassen, wenn sein Wissen nicht akquiriert werden kann.

3.5 Fuzzy-Logik und Fuzzy-Regelsysteme

Fuzzy-Logik bzw. Fuzzy-Set-Theorie und Fuzzy-Regelsysteme (FRSe) dienen dazu, diejenigen Sachverhalte und Zusammenhänge berechen- und anwendbar sowie robust zu modellieren, welche in natürlicher Sprache vorliegen [HSA99, S. 247 ff.] und denen somit die in Abschnitt 2.3.1 beschriebene linguistische Unsicherheit anhaftet [BKI03, S. 395]. Die grundlegenden Konstrukte der Fuzzy-Logik, *Zugehörigkeitsfunktion* und *linguistische Variable*,

beschreiben z. B. [Zim93, S. 8 ff.] und [Sib98, S. 113 ff]. Das Schlussfolgern in Fuzzy-Regelsystem, das sich in das *Fuzzifizieren* reeller Eingabewerte, in die *„Fuzzy-Inferenz“* mit ihren drei Schritten, *Aggregation*, *Implikation* und *Akkumulation*, und in das *Defuzzifizieren* unterteilt, erläutert z. B. [Zim93, S. 96 ff.].

Zwei jüngere Ansätze zur Unterstützung der PPS durch Fuzzy-Technologie stammen von [Keu99] und [Sib98]. Diese Ansätze sind insbesondere durch hohen manuellen Aufwand für die Modellierung, durch starke subjektive Einflüsse und durch Ergebnisse gekennzeichnet, die sich nur unwesentlich von den angegebenen Referenzverfahren unterscheiden.

Der Vorteil der Fuzzy-Technologie besteht darin, dass sich mit ihr Sachverhalte abbilden lassen, die nur in natürlicher Sprache adäquat ausgedrückt werden können und somit linguistische Unsicherheit aufweisen. FRSe können linguistische Unsicherheit abbilden. Beim Einsatz von Fuzzy-Logik und FRSen treten aber unter anderem die folgenden grundsätzlichen Probleme auf:

- Der Rechenaufwand für das Anwenden einer Regel ist in FRSen um ein Vielfaches höher als der Rechenaufwand für das Anwenden einer Regel in einem klassischen Regelsystem.

- Bei Wissensanwendung in FRSen („Fuzzy-Inferenz“) sind im Gegensatz zu klassischen Regelsystemen nicht nur die Regeln auf einem Pfade anzuwenden, sondern *alle* Regeln, und die Ergebnisse der Anwendung sind geeignet zu akkumulieren [Zim93, S. 52], woraus ein Vielfaches an Rechenaufwand resultiert.

- FRSe verfügen nicht über (Online-)Lernfähigkeit, wodurch sie sich z. B. von KNNen (siehe Abschnitt unterscheiden. [Sib98, S. 159] Sie sind demzufolge meist subjektiv geprägt. Es liegen aber Lernverfahren, wie z. B. der Fuzzy-ID3 [Web92], vor, welche zumindest Teile von FRSen aus Daten erstellen.

- Zusätzlich zu der grundsätzlich schwierigen und aufwendigen Akquisition der Regeln sind bei FRSen viele Parameter einzustellen (Art, Anzahl und Form mehrerer Zugehörigkeitsfunktionen pro linguistischer Variable; Operatoren und ihre Parameter für Aggregation, Implikation und Akkumulation; Regelgewichte für jede Regel; Entscheidung über (De)fuzzifizierung zwischen Regelsystemen; ...), was viel Erfahrung oder empirische Studien erfordert [Zim93, S. 212 f.] und ihre Erstellung weiter erschwert.

- FRSe benötigen vollständige Eingabedaten, die im Kontext der PPS nur selten vorliegen. Stehen nicht alle Eingabedaten zur Verfügung, kann keine Fuzzy-Inferenz vollzogen werden.

- FRSe eignen sich gut zur Abbildung reellwertiger Basisvariablen, aber können keine Basisvariablen mit nominalen oder rein ordinalen Ausprägungen abbilden.

[Kur03, S. 362] räumt ein, dass mit Fuzzy-Technologie noch keine Erfolge bei der betriebswirtschaftlichen Planung und Steuerung erzielt worden sind, die mit den Erfolgen der Fuzzy-Technologie bei der Regelung technischer Systeme vergleichbar wären. Es seien nur Teilprobleme „eventuell besser gelöst“ worden, und ein „gewaltiger mathematischer Aufwand“ verschrecke unter Umständen die Anwender.

3.6 Weitere Verfahren

Die vorangegangenen Abschnitte geben einen Überblick über ausgewählte alternative Verfahren zur Unterstützung zur PPS. Über diese Verfahren hinaus kommen noch weitere alternative Verfahren zur Unterstützung der PPS zum Einsatz, die an dieser Stelle lediglich vorgestellt, aber nicht intensiv untersucht werden.

Der „Optimierung mit Ameisenkolonien“ (Ant Colony Optimization, ACO) liegt der Sachverhalt zugrunde, dass Ameisen ihre Wege mit Pheromonen benetzen, sich mit höherer Wahrscheinlichkeit für Wege entscheiden, die eine höhere Pheromonkonzentration aufweisen, kürzere Wege höher frequentieren und demzufolge nach einem gewissen Zeitraum alle den kürzesten Weg wählen, womit der kürzeste Weg gefunden worden wäre. ACO ist im Rahmen der PPS bisher vorwiegend zur Maschinenbelegungs- und Feinplanung eingesetzt worden. Probleme bei der ACO bereiten das manuelle Finden eines passenden Modelles, das Konvergenzverhalten und die Laufzeit. Zur ACO, ihren Vorzügen und Nachteilen sowie zu ihrem Einsatz in der PPS siehe z. B. [DS04].

Beim „Optimieren mit Bomben“ [Pöp00] (Ruin&Recreate, R&R) wird eine Lösung mittels einer einfacheren lokalen Suche (siehe Abschnitt 3.1) erzeugt, wiederholt an zufälligen Stellen in unterschiedlichem Ausmaße zerstört (Ruin) und dann mit aufwendigeren Suchverfahren wiederaufgebaut (Recreate), so dass sich die Lösung nach und nach einem Optimum nähert. Bisher ist R&R vorwiegend zur Tourenplanung eingesetzt worden, aber es liegen auch erfolgreiche Ansätze zur Optimierung der Rüstreihenfolge im Rahmen

der PPS vor. [Hep04] Wie auch ACO kämpft R&R mit dem manuellen Finden eine passenden Modelles, dem Konvergenzverhalten und der Laufzeit. Zu R&R, seinen Vor- und Nachteilen und seinen Einsatzgebieten siehe unter anderem [SSSWD00].

Multi-Agenten-Systeme (MASe) trachten, durch das Zusammenwirken von Agenten Probleme der PPS zu lösen. Im allgemeinen Falle arbeiten Agenten weitgehend autonom, besitzen lokales Wissen, nehmen ihre Umwelt wahr, sind lernfähig, agieren und reagieren zielgerichtet, kommunizieren und kooperieren mit anderen Agenten. Ein möglicher Ansatz, ein MAS für die PPS zu modellieren, besteht darin, für Objekte in der Produktion, wie z. B. Maschinen und Aufträge, Agenten anzulegen, welche ihre Objekte einander bestmöglich zuordnen. Die Kommunikation kann direkt oder über eine Zentrale, ein sogenanntes schwarzes Brett, erfolgen, um die Komplexität zu verringern. Der Aufwand für den Aufbau eines individuellen MAS zur Unterstützung der PPS ist immens, auch wenn mittlerweile Multi-Agenten-Plattformen existieren [BCG07], MASe verursachen – insbesondere bei einer suboptimalen Systemarchitektur – hohen Rechenaufwand, und MASe garantieren keinen Erfolg. Spezielle Einsätze von MASen zur Unterstützung der PPS schildern z. B. [Mön06] und [BB97]; [CC04] gibt einen umfassenden Überblick über den Einsatz von MASen in der PPS, und [Lid07] führt allgemein in MASe ein.

Constraint Programming (CP) löst sogenannte Constraint Satisfaction Problems (CSPs). Die nachfolgende Darstellung zu CP und CSPs orientiert sich an [Bar99a] und [FA97b]. Ein CSP besteht aus einer Menge von Variablen mit jeweils einer abzählbar endlichen Menge von Zuständen und aus einer Menge von Constraints, welche die Kombinationen von Zuständen beschränken, welche die Variablen gleichzeitig annehmen dürfen. Optional enthält ein CSP eine Zielfunktion. Mittels systematischer Suche nach zulässigen Lösungen und/oder dem Ausschluss unzulässiger Lösungen aus dem Lösungsraum findet CP gültige – und gegebenenfalls entsprechend einer Zielfunktion gute – Lösungen. Zum Einsatz kommen auch Verfahren der lokalen Suche und EAen sowie Branch&Bound-Algorithmen [Dak65], [LD60]. CP kämpft mit mehreren Problemen [Bar99a]: Die Lösungsgüte und der Rechenaufwand sind unvorhersagbar – besonders deshalb, weil Lösungsgüte und Rechenaufwand äußerst empfindlich auf Änderungen der Constraints und/oder der Eingaben reagieren. Es ist schwer, die Constraints so zu identifizieren, dass der Rechenaufwand erträglich bleibt. Für die Wahl des Lösungsverfahrens gibt es keinerlei allgemeingültige Anhaltspunkte. Die Lösungen verhalten sich nicht robust gegenüber einer sich ändernden Umgebung und stochastischen Einflüssen, und die Lösungsverfahren eignen sich nicht dazu, Lösungen in Echtzeit anzupassen. Nichtsdestotrotz berichtet [FA97a] von erfolgreichen Anwendun-

gen der CP zur PPS in der Praxis – so unter anderem vom Einsatz der CP zur Feinplanung in der Fertigung.

Nicht zuletzt unterstützt die ereignisorientierte Simulation (EOS) die PPS. Sie modelliert für die PPS relevante Objekte durch statische und bewegliche Instanzen, wie z. B. Maschinen und Aufträge, welche komplexe Zustände aufweisen. Jedes Ereignis enthält einen Zeitpunkt, zu dem es stattfindet, und eine Routine, welche ausgeführt wird, wenn das Ereignis stattfindet. Sämtliche Ereignisse liegen in einer zentralen Ereignisliste vor, sortiert nach dem Zeitpunkt ihres jeweiligen Auftretens (und einer Priorität). Eine zentrale Routine arbeitet diese Ereignisse ihrer Reihenfolge nach ab, indem sie die Routine des jeweils anstehenden Ereignisses ausführt. Eine solche Routine liest und schreibt Zustände der statischen und beweglichen Objekte, erzeugt neue Ereignisse für zukünftige Zeitpunkte und reiht diese Ereignisse in die zentralen Datenstrukturen ein. Beispiel für Ereignisse sind z. B. die Freigabe eines Fertigungsauftrages, das Belegen einer Maschine durch einen Fertigungsauftrag und das Fertigstellen eines solchen Auftrages auf einer Maschine. Wenn nun z. B. ein Auftrag eine Maschine belegt, ändert sich unter anderem ihr Zustand auf „belegt“, und ein neues Ereignis wird erzeugt für den Simulationszeitpunkt plus die Bearbeitungszeit des Auftrages auf der Maschine. Das neue Ereignis beschreibt das Fertigstellen des Auftrages auf der Maschine und wird in die zentrale Ereignisliste eingereiht. Erreicht die zentrale Routine dieses Ereignis, verarbeitet sie es ebenfalls, usw. Die Zeiten, die zwischen jeweils zwei aufeinanderfolgenden Ereignissen liegen, wartet die EOS nicht, sondern überspringt sie, so dass sie nur die Laufzeit für das Einsortieren und Abarbeiten der Ereignisse in Anspruch nimmt. Diese Laufzeit ist aber bei der EOS einer komplexen Fertigung über einen längeren Zeitraum durchaus erheblich. [Völ03, S. 1] EOS selbst optimiert nicht, sondern prognostiziert ausschließlich den Verlauf der Fertigung. In Verbindung mit den Verfahren aus den Abschnitten 3.1 und 3.2 dient sie der simulationsbasierten, iterativen Optimierung (siehe unter anderem [Völ03] und [Kru97]). Der Nachteil der simulationsbasierten, iterativen Optimierung besteht darin, dass Simulationsläufe für jeden alternativen Plan durchgeführt werden müssen, was sehr hohe Laufzeiten für die gesamte Optimierung nach sich zieht. EOS ist so genau wie das ihr zugrundeliegende Modell. Ein Simulationsmodell kann automatisch aus Daten eines PPS-Systems erstellt werden (siehe z. B. [Eck02], [KGW00] sowie Abschnitt 6.5.2), was allerdings nur sinnvoll ist, wenn die Daten vollständig und korrekt im PPS-System vorliegen. Das manuelle Erfassen der Daten und der manuelle Aufbau eines Simulationsmodelles aus ihnen ist hingegen äußerst aufwendig. EOS berücksichtigt die Stochastik der Produktion nicht per se, kann aber um stochastische Komponenten erweitert werden. [Pag91] Allerdings sind dann Simulationsläufe mehrfach

durchzuführen, um statistische Sicherheit zu erlangen, was die Laufzeit drastisch erhöht. Mit fehlenden Daten kann EOS nicht umgehen.

Der Einsatz der EOS zur PPS gestaltet sich vielfältig. Einige Beispiele – z. T. aus dem persönlichen Umfeld des Autors – sind die folgenden: [Sch02b], [Thi01], [GR99], [SGH97], [Bar94] und [Gün93]. Zu den Grundlagen der EOS, dem Aufbau entsprechender Simulationsmodelle und der Programmierung ereignisorientierter Simulatoren siehe z. B. [Fis01] und [BCNN01].

3.7 Zusammenfassende Beurteilung alternativer Verfahren zur Unterstützung der PPS

Sollen die bisher in Kapitel 3 vorgestellten Verfahren zusammenfassend beurteilt und miteinander verglichen werden, so bietet es sich an, Kriterien zu formulieren, anhand denen man sie beurteilen und vergleichen kann. Folgende Kriterien sind identifiziert worden, wobei kein Anspruch auf Vollständigkeit an sie erhoben wird:

1. hohe Lösungsgüte, bezogen auf die verfolgten Ziele,
2. niedriger Rechenaufwand hinsichtlich Speicherbedarf und Laufzeit,
3. gute Nachvollziehbarkeit bzw. Erklärbarkeit und
4. hohe Akzeptanz von Unsicherheit bzw. ihre adäquate Abbildung.

Diese Kriterien werden angewandt auf die drei Aspekte der Wissensverarbeitung, die jedem Verfahren bzw. jedem Modell zumindest implizit innewohnen:

1. die Wissensrepräsentation bzw. die Repräsentation des Modells,
2. die Wissensakquisition bzw. die Modellierung und
3. die Wissensanwendung bzw. das Anwenden des Modells.

Dem Aspekt der Wissensrepräsentation wird noch das Kriterium hinzugefügt, wie gut die Modelle Kausalität abbilden, und dem Aspekt der Wissensanwendung, inwieweit die Verfahren optimierenden Charakter tragen. Nicht alle Kriterien können auf alle Aspekte angewandt werden, und für einige der Verfahren kann nicht für jede Kombination aus Kriterium und Aspekt eine allgemeingültige Aussage getroffen werden.

Tabelle 3.1 stellt dar, wie gut die Verfahren und Modelle den Kriterien genügen. Die Beurteilung der Verfahren erfolgt sehr grob, geht aber mit den Beurteilungen der Verfahren in den Abschnitten 3.1 bis 3.6 konform. Die

Tabelle 3.1: Vergleich alternativer Verfahren zur Unterstützung der PPS

			Alternative Verfahren zur Unterstützung der PPS							
		Kriterium	MHen	KNNe	XPSe	FRSe	MASe	EOS	CP	BNe
Aspekt	Repräsentation	Geringer Speicherbedarf	++	0	+	-	-	-	++	-
		Gute Erklärbarkeit	0	--	++	+	++	+	0	++
		Adäquate Abbildung von Unsicherheit	--	--	--	+	-	+	--	++
		Abbilden von Kausalität	0	--	0	0	+	++	-	++
	Modellierung	Hohe Lösungsgüte	0	+	0	0	0	+	+	+
		Geringer Aufwand	--	--	0	-	-	0	--	++
		Nachvollziehbarkeit	-	--	0	+	0	++	0	0
		Berücksichtigung von Unsicherheit	--	--	--	+	--	+	--	++
	Anwendung	Hohe Lösungsgüte	++	0	0	0	+	++	++	0
		Geringer Rechenaufwand	--	++	0	0	--	--	--	+
		Nachvollziehbarkeit/ Erklärbarkeit	--	--	++	+	+	+	--	+
		Integration von Unsicherheit	--	--	--	+	--	+	--	++
		Optimierender Charakter	++	0	--	--	+	--	++	+

Legende: -- sehr schlecht, - schlecht, 0 neutral, + gut, ++ sehr gut

letzte Spalte der Tabelle stellt den Verfahren BNe gegenüber. Mit ihr erfolgt ein Vorgriff auf die Eigenschaften und Fähigkeiten BNe, die erst Abschnitt 4.5 darstellt. Die folgenden Kapitel belegen die Bewertungen in dieser Spalte.

Die erste Spalte der Tabelle fasst Verfahren der lokalen Suche und EAen unter der Bezeichnung Metaheuristiken (MHen) zusammen, da die lokale Suche einem EA mit nur einem Individuum entspricht. Zu den MHen zählen auch ACO sowie R&R, da sie ähnliche Charakteristika besitzen.

Die Bewertungen in den Zellen der Tabelle 3.1 geben nur eine Tendenz wieder und sind durch konkrete Beispiele sicherlich sowohl zu belegen als auch zu widerlegen. Tabelle 3.1 bringt aber zum Ausdruck, dass die alternativen Verfahren zur Unterstützung der PPS unter verschiedenen Aspekten und bezogen auf verschiedene Kriterien unterschiedliche Vor- und Nachteile aufweisen, einander insgesamt aber durchaus ebenbürtig sind. BNe stehen den anderen Verfahren im Mittel nicht nach und sind somit durchaus eine Untersuchung ihres Potenzials zur Unterstützung der PPS wert.

Kapitel 4

BNe - ein alternatives Verfahren zur Unterstützung der PPS[1]

4.1 Wissensrepräsentation: Abbilden von Wissen mittels BNe

4.1.1 Allgemeine BNe

Definition 4.1 (Bayes'sches Netz)
Es sei V eine Menge diskreter Variablen $v \in V$. Ein Bayes'sches Netz (BN) $B = (G, P)$ über der Menge der diskreten Variablen V ist ein geordnetes Paar (G, P) aus einem gerichteten, azyklischen Graphen[2] $G = (V, E)$ und einer Menge P von Bewertungsfunktionen[3] $p(v|Pa_v) \in P$ mit $v \in V$. Es existiert eine bijektive Abbildung $V \rightarrow P$, die jeder Variable $v \in V$ eine Bewertungsfunktion $p(v|Pa_v) \in P$ zuordnet.

Abbildung 4.1 stellt den Graphen eines BNes dar. Eine gerichtete Kante $(u, v) \in E$ bzw. $u \longrightarrow v$ eines BNes $B = ((V, E), P)$ stellt eine direkte kausale Beziehung zwischen der Variable u und der Variable v mit $u, v \in V$ dar [Nea04, S. 48 f.] [Hec96, S. 13. f.], das heißt, die Variable u beeinflusst die Variable v direkt und kausal.[4] Eine gerichtete Kante gibt den direkten kausalen Zusammenhang zwischen zwei Variablen allerdings nur qualitativ wieder.

[1] Die in diesem Kapitel verwendeten Begriffe werden in Anhang A.1 definiert und erläutert. Sofern kein expliziter Verweis von einem Begriff auf eine Definition in Anhang A.1 vorliegt, befindet sich die entsprechende Definition in diesem Anhang.

[2] siehe Definition A.32

[3] siehe Definition A.5

[4] Die Kanten müssen nicht als Kausalität interpretiert werden. [Jen01, S. 19] Aus Gründen der Einfachheit und Verständlichkeit soll das aber geschehen.

Erst die Bewertungsfunktion $p(v|Pa_v)$ quantifiziert, wie die Eltern $Pa_v \subset V$ die Variable $v \in V$ beeinflussen.

Theorem 4.1 (BN repräsentiert $\Pr(V)$)
Es sei V eine Menge diskreter Variablen. Ein BN $B = ((V, E), P)$ über V repräsentiert die gemeinsame Wahrscheinlichkeitsfunktion $\Pr(V)$ wie folgt:

$$\Pr(V) = \prod_{v \in V} p(v|Pa_v) \tag{4.1}$$

Ein BN faktorisiert demzufolge die gemeinsame Wahrscheinlichkeitsfunktion von V und kodiert sie so auf eine effiziente Art und Weise.

Nachdem BNe formal dargestellt worden sind, soll ein einfaches Beispiel geringen Umfanges BNe veranschaulichen. Das Beispiel heißt „Marie und ihre Rosen“. Von Marie sei vorerst noch nicht die Rede. Die Menge $I = \{1, \ldots, 5\}$ stelle die Indexmenge der diskreten Variablen V dar, welche fünf binäre Variablen enthalte:

1. Die Variable Regen re sagt aus, ob es regnet, und weist die Zustände $S_1 = \{ne_{re}, ja_{re}\}$ auf für nein, es regnet nicht, $re = ne_{re}$, und ja, es regnet, $re = ja_{re}$.

2. Die Variable Rasensprenger sp gibt Auskunft darüber, ob der Rasensprenger an ist, und kann sich in den Zuständen $S_2 = \{au_{sp}, ei_{sp}\}$ befinden für nein, der Rasensprenger ist aus, $sp = au_{sp}$, und ja, der Rasensprenger ist an, $sp = ei_{sp}$.

3. Die Variable Straße st informiert darüber, ob die Straße nass ist, und befindet sich in den Zuständen $S_3 = \{tr_{st}, na_{st}\}$ für nein, die Straße ist trocken, $st = tr_{st}$, und ja, die Straße ist nass, $st = na_{st}$.

4. Die Variable Boden bo gibt an, wie gut der Boden gewässert ist, und nimmt die Zustände $S_4 = \{tr_{bo}, fe_{bo}\}$ an mit nein, der Boden ist trocken, $bo = tr_{bo}$, und ja, der Boden ist feucht, $bo = fe_{bo}$.

5. Die Variable Rosen ro sagt aus, ob es den Rosen gutgeht, und nimmt die Zustände $S_5 = \{sc_{ro}, gu_{ro}\}$ an für nein, es geht den Rosen schlecht, $ro = sc_{ro}$, oder ja, es geht den Rosen gut, $ro = gu_{ro}$.

Es sei $B = ((V, E), P)$ ein BN über $V = \{re, sp, st, bo, ro\}$. Zwischen den Variablen in V liegen folgende direkte kausale Beziehungen – und nur diese Beziehungen – vor:

1. Ob es regnet, beeinflusst, ob die Straße nass ist, das heißt, $re \longrightarrow st$ bzw. $(re, st) \in E$.

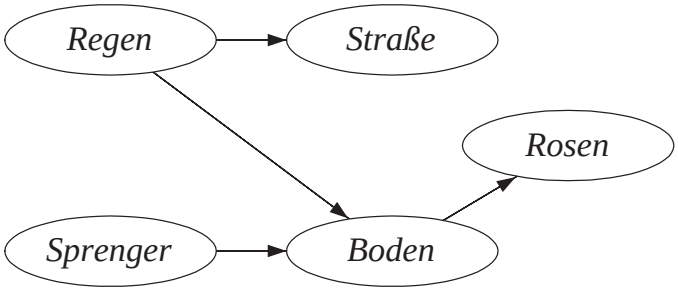

Abbildung 4.1: Graph des BNes zu „Marie und ihre Rosen“

2. Ob es regnet, wirkt sich darauf aus, ob der Boden feucht ist, das heißt, $re \longrightarrow bo$ bzw. $(re, bo) \in E$.

3. Ob der Rasensprenger eingeschaltet ist, beeinflusst, ob der Boden feucht ist, das heißt, $sp \longrightarrow bo$ bzw. $(sp, bo) \in E$.

4. Ob der Boden feucht ist, wirkt sich darauf aus, ob es den Rosen gutgeht, das heißt, $bo \longrightarrow ro$ bzw. $(bo, ro) \in E$

Der Rasensprenger sprengt also z. B. nicht die Straße, das heißt, $(sp, st) \notin E$. Es liegen in $B = ((V, E), P)$ also die folgenden gerichteten Kanten vor:

$$E = \{(re, st), (re, bo), (sp, bo), (bo, ro)\}$$

Abbildung 4.1 stellt den Graphen des BNes zu „Marie und ihre Rosen“ graphisch dar. Eine graphische Repräsentation eines BNes ist, da besser zu überblicken, einer textuellen meist vorzuziehen. [BETT98, S. vii] Nachdem die qualitativen Bestandteile des BNes, der Graph bzw. seine Knoten und Kanten, beschrieben worden sind, die auch als die Struktur des BNes bezeichnet werden [Nea04, S. 285], verbleibt der quantitative Teil, die Bewertungsfunktionen P. Für die Variablen Regen re und Sprenger sp , die keine Eltern aufweisen, das heißt, im Kontext von V nicht von anderen Variablen beeinflusst werden, entsprechen die Bewertungsfunktionen den unbedingten Wahrscheinlichkeitsfunktionen dieser Variablen (siehe Tabelle 4.1 a) und b)). Das heißt, $p(re|Pa_{re}) = p(re|\emptyset) = p(re) = \Pr(re)$ und $p(sp|Pa_{sp}) = p(sp|\emptyset) = p(sp) = \Pr(sp)$. Die Variable Straße st wird durch die Variable Regen re direkt kausal beeinflusst. Ihre Bewertungsfunktion stellt Tabelle 4.1 c) dar. Die Bewertungsfunktion $p(st|\{re\})$ umfasst zwei bedingte Wahrscheinlichkeitsfunktionen: $Pr(st|re = ne_{re})$ und $Pr(st|re = ja_{re})$. Die Variable Boden bo wird von zwei Variablen, Regen re und Sprenger sp, direkt kausal beeinflusst, und ihre Bewertungsfunktion $p(bo|\{re, sp)\}$ umfasst

Tabelle 4.1: Bewertungsfunktionen der Variablen a) Regen re, b) Sprenger sp und c) Straße st

a)

re	$p(re\|\emptyset)$
ne_{re}	0,8
ja_{re}	0,2

b)

sp	$p(sp\|\emptyset)$
au_{sp}	0,95
ei_{sp}	0,05

c)

re	st	$p(st\|\{re\})$
ne_{re}	tr_{st}	0,90
ne_{re}	na_{st}	0,10
ja_{re}	tr_{st}	0,01
ja_{re}	na_{st}	0,99

Tabelle 4.2: Bewertungsfunktionen der Variablen a) Boden bo und b) Rosen ro

a)

re	sp	bo	$p(\{bo\}\|\{re, sp\})$
ne_{re}	au_{sp}	tr_{bo}	0,85
ne_{re}	au_{sp}	fe_{bo}	0,15
ne_{re}	ei_{sp}	tr_{bo}	0,05
ne_{re}	ei_{sp}	fe_{bo}	0,95
ja_{re}	au_{sp}	tr_{bo}	0,10
ja_{re}	au_{sp}	fe_{bo}	0,90
ja_{re}	ei_{sp}	tr_{bo}	0,01
ja_{re}	ei_{sp}	fe_{bo}	0,99

b)

bo	ro	$p(\{ro\}\|\{bo\})$
tr_{bo}	sc_{ro}	0,75
tr_{bo}	gu_{ro}	0,25
fe_{bo}	sc_{ro}	0,05
fe_{bo}	gu_{ro}	0,95

vier bedingte Wahrscheinlichkeitsfunktionen und füllt Tabelle 4.2 a). Die Variable Rosen ro wird durch die Variable Boden bo direkt kausal beeinflusst. Ihre Bewertungsfunktion stellt Tabelle 4.2 b) dar. Die Bewertungsfunktion $p(ro|\{bo\})$ umfasst zwei bedingte Wahrscheinlichkeitsfunktionen. – Bevor darauf eingegangen wird, wie BNe erstellt und angewandt werden, soll zunächst auf spezielle BNe, die sogenannten dynamischen Bayes'schen Netze, eingegangen werden.

4.1.2 Dynamische BNe

Dynamische BNe (DBNe) sind für die Abbildung stochastischer Prozesse entwickelt worden. Bei der Produktion, wie sie in Abschnitt 2.4 beschrieben worden ist und in der vorliegenden Arbeit geplant und geregelt werden soll, handelt es sich im Allgemeinen um einen stochastischen Prozess [KR84, S. 46 f.], der seinen Zustand über der Zeit ändert. Ein solcher Prozess kann mittels Zufallsvariablen beschrieben werden. Eine Zustandsvariable ist z. B. die Warteschlangenlänge.

DBNe werden über Zufallsvektoren (Vektoren von Zufallsvariablen) eingeführt: Es seien $v_1, \ldots, v_n$ Zufallsvariablen. Der Spaltenvektor $V = (v_1, \ldots, v_n)^{\mathrm{T}}$ heißt Zufallsvektor, wobei V auch die Menge $\{v_1, \ldots, v_n\}$ der Zufallsvariablen bezeichne. Ob V eine Menge oder einen Vektor von Zufallsvariablen bezeichnet, geht im Einzelfall aus dem Kontext hervor. Ein d-dimensionaler Zufallsvektor entspricht einer d-dimensionalen Zufallsvariable.

Es sei $V_T = \{V[t]|t \in T\}$ ein zeitdiskreter stochastischer Prozess. Es seien $t \in T$ die diskreten Zeitpunkte, zu denen der Zustand des Prozesses V_T betrachtet wird. Die Menge von Zufallsvariablen bzw. die Zufallsvektoren $V[t]$ charakterisieren den Zustand des Prozesses V_T zum Zeitpunkt $t \in T$, und eine Zufallsvariable $v[t] \in V[t]$ beschreibe den Zustand einer Zustandsvariable $v \in V$ zum Zeitpunkt t. Sollten die Variablen in der Realität nicht diskret sein, müssen sie diskretisiert werden, um den Prozess mittels DBNe abzubilden.

Ein BN $B = (G(V_T, E), P)$ über T repräsentiert die gemeinsame Wahrscheinlichkeitsfunktion $\prod_{v \in V_T} p(v|Pa_v)$ von V_T. Das BN B repräsentiert also nicht eine gemeinsame Wahrscheinlichkeitsfunktion von V (siehe Theorem 4.1), sondern die gemeinsame Wahrscheinlichkeitsfunktion von V_T. Im allgemeinen Fall, wenn keine Annahmen über den zugrundeliegenden Prozess V_T getroffen werden, können der Graph G und die Bewertungsfunktionen P von B sehr komplex werden. [DD04, S. 291] [FMR98, S. 140] Deshalb werden meist drei vereinfachende Annahmen über den Prozess V_T getroffen:

Annahme 4.1 (Äquidistante Zeitpunkte)
Der Prozess V_T erstrecke sich über die Zeitpunkte $0, \ldots, |T|$. Für drei direkt

aufeinanderfolgende Zeitpunkte t_{k-1}, t_k und t_{k+1} gelte

$$t_k - t_{k-1} = t_{k+1} - t_k \tag{4.2}$$

mit $k = 1, \ldots, |T| - 1$.

Das heißt, dass zwischen zwei aufeinanderfolgenden Zeitpunkten immer der gleiche Zeitraum liegt respektive die gleiche Zeit vergeht.

Annahme 4.2 (Markowscher Prozess)
Bei V_T handele es sich um einen Markowschen Prozess (erster Ordnung):

$$p\left(V[t] \left| \bigcup_{k=0}^{t-1} V[k] \right.\right) = p(V[t]|V[t-1]) \tag{4.3}$$

mit $0 < t < m$.[5]

Das heißt, dass ein Zustand $V[t]$ von V_T zum Zeitpunkt t ausschließlich vom ihm direkt vorangegangen Zustand $V[t-1]$ von V_T zum Zeitpunkt $t-1$ abhängt.

Annahme 4.3 (Stationärer Prozess)
Der Prozess sei stationär:

$$p(V[t]|V[t-1]) = p(V[t+1]|V[t]) \tag{4.4}$$

mit $0 < t < m - 1$.[6]

Das heißt, dass die Bewertungsfunktionen $p(V[t]|V[t-1])$ der Übergänge vom Zustand $V[t-1]$ zum Zustand $V[t]$ für $t = 1, \ldots, m-1$ identisch sind (und nicht von der Zeit t abhängen).

Sollte ein Prozess den Annahmen 4.1, 4.2 oder 4.3 nicht genügen, können sie natürlich sukzessive gelockert werden. Damit unter den getroffenen Annahmen auch ein Prozess abgebildet werden kann, der nicht markowsch ist, bietet es sich ebenfalls an, ein „Gedächtnis“ einzuführen: Dazu wird die Menge V um Variablen ergänzt, welche den Zustand des Prozesses zu Zeitpunkten in der Vergangenheit charakterisieren. Gelten die Annahmen 4.1, 4.2 und 4.3 jedoch, kann V_T durch zwei relativ einfache BNe, das Eingangsnetz und das Übergangsnetz, vollständig beschrieben werden.

[5] Die Annahme entspricht der Definition 2.6. Sie ist aus Gründen der leicht abweichenden Notation an dieser Stelle noch einmal aufgeführt worden.

[6] Die Annahme entspricht der Definition 2.5. Sie ist aus Gründen der leicht abweichenden Notation an dieser Stelle noch einmal aufgeführt worden.

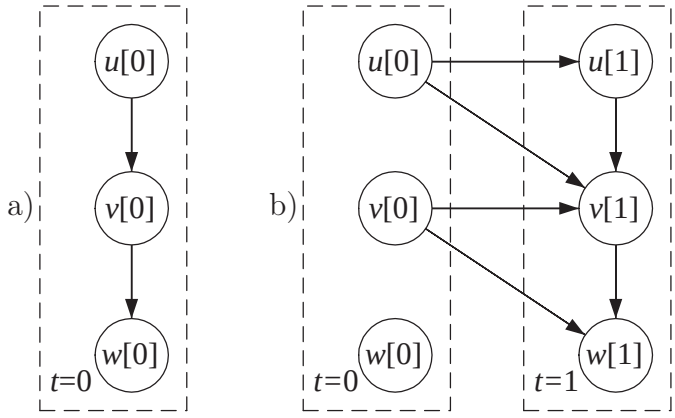

Abbildung 4.2: Graphen eines DBNes über $V = \{u, v, w\}$: a) Graph des Eingangsnetzes über $V[0]$, b) Graph des Übergangsnetzes über $V[0] \cup V[1]$

Definition 4.2 (Eingangsnetz)
Es seien V_T und V wie vorab vereinbart. Es sei $V_0 = V[0]$ die Menge der Variablen, die den Zustand von V_T zum Zeitpunkt 0 darstellen. Ein Eingangsnetz $B_0 = ((V_0, E_0), P_0)$ ist ein BN über V_0.

Abbildung 4.2 a) stellt den Graphen eines Eingangsnetzes dar. Das Eingangsnetz B_0 repräsentiert mit $\prod_{v \in V_0} p(v|Pa_v)$ die gemeinsame Wahrscheinlichkeitsfunktion von V_0. Es spiegelt also den Zustand von V_T zum Zeitpunkt 0 (den Anfangszustand von V_T) wider. Das Eingangsnetz B_0 repräsentiert darüber hinaus den Prozess V_T vor dem Zeitpunkt 0. [FMR98, S. 140]

Definition 4.3 (Übergangsnetz)
Es seien V_T und V wie vorab vereinbart. Es seien $V[0]$ und $V[1]$ die Mengen der Variablen, welche die Zustände von V_T zu den Zeitpunkten 0 und 1 darstellen. Es sei $V_\rightarrow = V[0] \cup V[1]$. Ein Übergangsnetz $B_\rightarrow = ((V_\rightarrow, E_\rightarrow)P_\rightarrow)$ ist ein spezielles BN über $V_\rightarrow$, für das gilt

$$\forall v \in V[0](Pa_v = \emptyset). \tag{4.5}$$

Abbildung 4.2 b) stellt den Graphen eines Übergangsnetzes dar. Das Übergangsnetz $B_\rightarrow$ repräsentiert eine durch $V[0]$ bedingte Bewertungsfunktion $\prod_{v \in V[1]} p(v|Pa_v)$ von $V_\rightarrow$. Durch die Bedingung in Formel 4.5 impliziert, liegen in $B_\rightarrow$ also keine Kanten zwischen den Variablen in $V[0]$ und keine Kanten von einer Variable in $V[1]$ zu einer Variable in $V[0]$ vor. Das Übergangsnetz $B_\rightarrow$ stellt aufgrund der Annahmen 4.1, 4.2 und 4.3 die bedingte Wahrscheinlichkeitsfunktion $\Pr(V[k+1]|V[k] = s)$ von $V[k+1]$ bedingt durch $V[k] = s$ für alle $s \in S_{V[k]}$ mit $0 \leq k < m - 1$ dar. Im Gegensatz zum Eingangsnetz

B_0 charakterisiert das Übergangsnetz $B_{\rightarrow}$ keinen Zustand von V_T zu einem Zeitpunkt, sondern den Zustandsübergang zwischen zwei aufeinanderfolgenden Zuständen $V[k]$ und $V[k+1]$ von V_T zu den Zeitpunkten k und $k+1$.

Definition 4.4 (Dynamisches Bayes'sches Netz)
Es seien T, B_0 und $B_{\rightarrow}$ wie vorab vereinbart. Ein DBN $\mathbb{D}$ über T ist ein geordnetes Paar $(B_0, B_{\rightarrow})$.

Ein DBN $\mathbb{D}$ repräsentiert die gemeinsame Wahrscheinlichkeitsfunktion über T in einer kompakten Art und Weise. Um später probabilistische Inferenz (PI) über einem DBN vollziehen zu können, wird das DBN „entrollt", indem – informal ausgedrückt – sein Eingangsnetz und $m-1$ mal sein Übergangsnetz aneinandergefügt werden.

Definition 4.5 (Entrolltes DBN)
Es seien V, m, B_0, $B_{\rightarrow}$ und $\mathbb{D}$ wie vorab vereinbart. Ein zu $\mathbb{D}$ äquivalentes entrolltes DBN $B_U = ((V_U, E_U), P_U)$ über T ist ein BN mit

$$V_U = \bigcup_{t=0}^{m-1} V[t] \quad \textit{und} \tag{4.6}$$

$$\begin{aligned} E_U = E_0 \cup \bigcup_{t=1}^{m-1} & (\{(u[t-1], v[t]) | (u[0], v[1]) \in E_{\rightarrow}\} \\ & \cup \{(u[t], v[t]) | (u[1], v[1]) \in E_{\rightarrow}\}) , \end{aligned} \tag{4.7}$$

für das gilt

$$\forall P[x] \in P_U \left(P[x] = \left\{ \begin{array}{l} P[0] | x = 0 \\ P[1] | x > 0 \end{array} \right. \right) . \tag{4.8}$$

Abbildung 4.3 stellt den Graphen eines entrollten DBNes dar. Formel 4.6 stellt die Knoten bzw. Variablen des entrollten DBNes über allen Zeitpunkten des Prozesses zusammen. Formel 4.7 fügt Kanten zwischen Knoten gleicher und direkt aufeinanderfolgender Zeitpunkte in das entrollte DBN ein, so wie es B_0 und $B_{\rightarrow}$ vorgeben. Formel 4.8 versieht die Knoten des entrollten DBNes mit Bewertungsfunktionen entsprechend der Bewertungsfunktionen von B_0 und $B_{\rightarrow}$.

Definition 4.6 (Zeitscheibe)
Es seien V, T, V_T und B_U wie vorab vereinbart. Die Zufallsvektoren $V[t]$ mit $t = 0, \ldots, [|T|]$ der Variablen im entrollten DBN B_U heißen auch Zeitscheiben. Der Index t heißt Zeitscheibenindex.

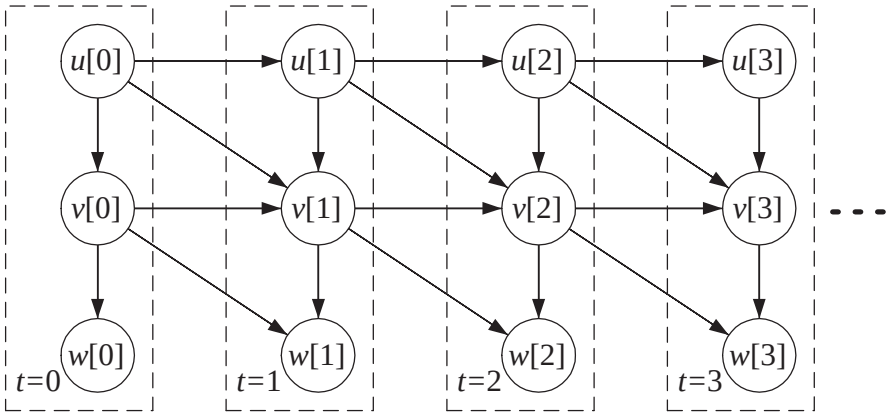

Abbildung 4.3: Graph eines entrollten DBNes über $V_T = \bigcup_{t=0}^{3} V[t]$ mit $V = \{u, v, w\}$

Die Rechtecke mit gestricheltem Rand in Abbildung 4.3 symbolisieren die Zeitscheiben. Die Variablen innerhalb eines Rechteckes sind jeweils eine Zeitscheibe. Wenn man ein DBN entrollt, so entrollt man es über $|T|$ Zeitscheiben.

Aufgrund der Annahmen 4.1, 4.2 und 4.3 sind die Übergangsnetze identisch, so dass es legitim ist, ein DBN zu entrollen. Ein entrolltes DBN repräsentiert ein DBN auf eine andere Art und Weise. Ein DBN kann demzufolge als ein Paar spezieller BNe, des Eingangsnetzes und des Übergangsnetzes, aufgefasst werden, und zum anderen als ein einzelnes BN, das entrollte DBN. Abbildung 4.2 stellt die Graphen des Eingangsnetzes und des Übergangsnetzes eines Beispiel-DBNes dar, Abbildung 4.3 die des äquivalenten entrollten DBNes.

Nachdem BNe vorgestellt worden sind, wird im folgenden darauf eingegangen, wie sie erstellt werden.

4.2 Wissensakquisition: Erstellen BNe

4.2.1 Ablauf der Wissensakquisition

Die Wissensakquisition zum Erstellen BNe geht wie folgt vonstatten:

1. Zuerst werden die für den Problembereich relevanten Variablen $v \in V$ identifiziert und ihre Zustände $s \in S_v$ ermittelt. Unter Umständen können ähnliche oder benachbarte Zustände einer Variable zu einem Zustand verschmolzen, oder ein Zustand kann in mehrere aufgespaltet werden.

2. Danach werden die Kanten $(u, v) \in E$ des Graphen G festgelegt, das heißt, es werden direkte kausale Beziehungen zwischen jeweils zwei Knoten u und v bestimmt. Alternativ können auch die Eltern Pa_v einer jeden Variable v bestimmt werden, da sich direkte Kausalität oft erst im Zusammenwirken mehrerer Variablen $\subseteq Pa_v$ äußert.

3. Im Anschluss erfolgt für jede Variable $v \in V$ das Schätzen einer bedingten Wahrscheinlichkeit für jede Kombination aus $s_{Pa} \in S_{Pa_v}$ und $s_v \in S_v$.

4. Abschließend wird das BN evaluiert, und eventuell erfolgt ein Rücksprung zu einem der vorangegangenen Schritte.

Prinzipiell ist ein Rücksprung zu einem vorangegangenen Schritt jederzeit möglich. Der Programmablaufplan (PAP) in Abbildung 4.4 stellt die Schritte zum Erstellen eines BNes noch einmal graphisch dar. Die Funktionen in den Anweisungen 4, 6, 8 und 10 editieren das BN, und ihr Rückgabewert gibt an, welche von ihnen als nächste ausgeführt werden soll. Die ganzzahligen Variablen F und N bestimmen, welche Funktion ausgeführt wird bzw. als nächstes ausgeführt werden soll. Anweisung 12 stellt sicher, dass keine Funktion zum Editieren des BNes übersprungen wird. Sollte ein Netz bereits zum Teil bestehen, können Variablen hinzugefügt oder entfernt werden, und Zustände der Variablen können hinzugefügt, entfernt, verschmolzen oder getrennt werden. Es besteht die Möglichkeit, Kanten hinzuzufügen, zu entfernen oder zu invertieren[7], sofern der Graph gerichtet und azyklisch bleibt. Die bedingten Wahrscheinlichkeiten in den Bewertungsfunktionen können ebenfalls editiert werden. Veränderungen am BN ziehen Veränderungen der Bewertungsfunktionen nach sich: Letztere werden entsprechend umverteilt, gestreckt oder gestaucht (siehe Anhang B.1.2).

Die Wissensakquisition zum Erstellen BNe kann manuell oder automatisch erfolgen. Es besteht die Möglichkeit, ein BN automatisch aus Daten zu erstellen und es dann manuell aufgrund von Expertenwissen zu editieren, und es ist umgekehrt möglich, ein aus Expertenwissen manuell erstelltes BN automatisch aufgrund von Daten nachzubearbeiten. Manuelle und automatische Wissensakquisition kann in beliebiger Folge mehrfach hintereinander erfolgen.

[7]Das Invertieren einer Kante (u, v) lässt sich zurückführen auf das Entfernen von (u, v) und das Hinzufügen der zu ihr inversen Kante (v, u).

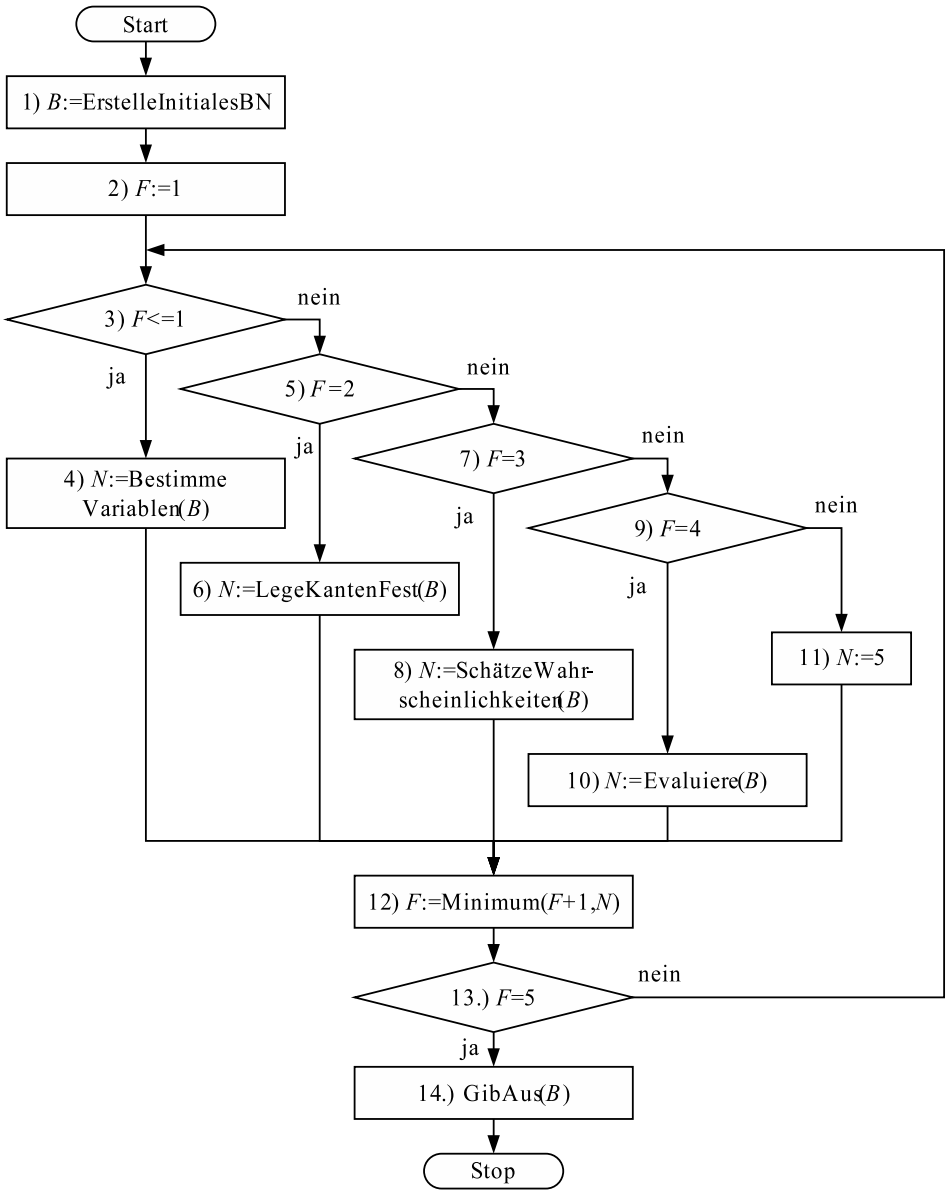

Abbildung 4.4: PAP zum Erstellen eines BNes B: Der PAP führt aufgrund des Wertes der ganzzahligen Variable F eine der Anweisungen $\mathbb{A} = \{4, 6, 8, 10, 11\}$ aus. Jede Funktion in $\mathbb{A} \setminus \{11\}$ modifiziert B, entscheidet daraufhin intern, welche der Anweisungen in $\mathbb{A}$ als nächste ausgeführt werden soll, gibt die Entscheidung in Form einer ganzen Zahl an die ganzzahlige Variable N zurück und bestimmt somit indirekt, ob das Erstellen von B voranschreitet oder zurückspringt. Anweisung 12 stellt sicher, dass der PAP im Kontext der Anweisungen $(4, 6, 8, 10, 11)$ maximal eine voranschreitet.

4.2.2 Manuelle Wissensakquisition

Das manuelle Erstellen BNe erfolgt nach dem in Abbildung 4.4 dargestellten Schema durch Experten. Es wird unterstellt, dass es dem Experten gelingt, die relevanten Variablen $v \in V$ zu identifizieren. Oft gelingt es ihm auch, die direkten kausalen Beziehungen $(u, v) \in E$ anzugeben. Korrelieren jedoch zwei Variablen u und v miteinander, und der Experte kann nicht ad hoc angeben, ob u Ursache von v ist, so sollte er folgendes Schema anwenden: Er sollte gedanklich den Zustand von u durch einen externen Einfluss setzen lassen und einschätzen, ob sich dadurch der Zustand von v änderte. Wenn nicht, ist u keine Ursache von v. Betrachten wir die Variablen Straße und Regen aus dem Beispiel „Marie und ihre Rosen“: Wenn jemand die nasse Straße mit Heißluft trocknete, würde das nicht dazu führen, dass es aufhört zu regnen, und wenn er einen Eimer Wasser auf die trockene Straße gösse, würde es nicht beginnen zu regnen, woraus zu schließen ist, dass der Zustand der Straße nicht den Regen verursacht. Wenn allerdings umgekehrt jemand Silberjodid in Wolken injizierte und so veranlasste, dass es regnet [Czy95], [BSS92], [WD80, S. 0] [Den80, S. 96 ff.], dann würde die Straße nass werden, und wenn er den Regen mit einem Dach von der Straße abhielte, dann würde die Straße trocknen. Daraus folgt, dass Regen den Zustand der Straße kausal beeinflusst. Korrelieren zwei Variablen miteinander, und der Experte kann nicht auf die beschriebene Art und Weise Ursache und Wirkung bestimmen, so sollte er eine oder mehrere Variablen identifizieren, welche die gemeinsame Ursache für die zwei korrelierten Variablen sind, und keine Kante zwischen den lediglich korrelierten Variablen einfügen, sondern nur Kanten zwischen den ursächlichen und den korrelierten Variablen. Nehmen wir z. B. die Variablen Straße und Boden. Sie korrelieren stark miteinander: Wenn die Straße nass ist, ist meist auch der Boden feucht (und umgekehrt), und wenn der Boden trocken ist, ist meist auch die Straße trocken (und umgekehrt). Allerdings kann man nach dem oben beschriebenen Schema nicht bestimmen, welche Variable Ursache und welche Wirkung ist. Auf der Suche nach der gemeinsamen Ursache identifiziert man den Regen. Demzufolge wird keine Kante zwischen Straße und Boden eingefügt, sondern zwei Kanten: eine zwischen Regen und Straße und eine zwischen Regen und Boden. Es ist allerdings auch möglich, dass zwei Variablen über eine gemeinsame Wirkung korreliert sind. Betrachten wir die Variablen Regen und (Rasen)sprenger. Wenn es regnet, ist der Sprenger meist aus (und umgekehrt), und wenn der Sprenger an ist, regnet es meist nicht (und umgekehrt). Wenn wir allerdings postulieren, dass der Boden (die gemeinsame Wirkung) einen seiner Zustände feucht oder trocken annimmt, kann man vom Zustand der Variable Regen auf den Zustand der Variable Sprenger schließen und die Korrelation zwischen Regen

und Sprenger erklären.

Können Kanten zwischen zwei korrelierten Variablen nicht eindeutig gerichtet werden, deutet das auf gemeinsame Ursachen oder gemeinsame Wirkungen hin. Sollten die gemeinsamen Ursachen oder Wirkungen nicht unter den bereits identifizierten Variablen gefunden werden, so ist das ein Indiz dafür, dass noch nicht alle für den Problembereich relevanten Variablen identifiziert worden sind, sogenannte versteckte Variablen vorliegen und beim Erstellen eines BNes ein Rücksprung zum ersten Schritt, der Identifikation der Variablen, erfolgen sollte.

Sollte es nicht gelingen, den Graphen des BNes zu erstellen, liegen aber Daten über das Problemfeld vor, so kann man die in den Abschnitten 4.2.3 und 4.2.4 beschriebenen Gütemaße und Suchalgorithmen anwenden, um den Graphen des BNes aus Daten zu erstellen oder alternative Graphen zu vergleichen.

Ist die Struktur (der Graph) des BNes erstellt worden, gilt es, die bedingten Wahrscheinlichkeiten zu schätzen. Das fällt Experten bei komplexen Bewertungsfunktionen schwer. [Tol98, S. 16] Zum einen wächst die Anzahl der für eine Variable v bzw. eine Bewertungsfunktion p zu schätzenden bedingten Wahrscheinlichkeiten exponentiell mit der Anzahl $|Pa_v|$ der Eltern Pa_v der Variable v: $|S_{\{v\}\cup Pa_v}| = |S_v| \prod_{u \in Pa_v} |S_u|$.[8] Wenn $|Pa_v|$ hoch ist, müssten Experten ergo sehr viele bedingte Wahrscheinlichkeiten schätzen. Zum anderen fällt es Experten schwer, konkrete bedingte Wahrscheinlichkeiten zu schätzen, wenn $|Pa_v|$ hoch ist, weil sie $|Pa_v|$ Einflussgrößen gleichzeitig ins Kalkül ziehen müssten. – Sollte es einem Experten nicht gelingen, die bedingten Wahrscheinlichkeiten vollständig zu schätzen, so kann er die im Abschnitt 4.2.5 beschriebenen Methoden zu ihrer Schätzung anwenden.

Angenommen, es sei dem Experten gelungen, das BN – wie beschrieben – zu erstellen. Dann stellt sich die Frage, wie gut das BN die Realität abbildet. Diese Frage beantwortet der folgende Abschnitt.

4.2.3 Gütemaße für BNe

Hat man ein BN erstellt, kann man seine Güte anhand von Daten bestimmen; oder besser ausgedrückt: Liegen mehrere alternative BNe $\mathbb{B}$ für eine Datenbasis D vor, so kann man jedem BN $B \in \mathbb{B}$ mittels eines Gütemaßes q eine reelle Zahl (Güte) zuordnen, zwei BNe $B_1, B_2 \in \mathbb{B}$ anhand ihrer Güte miteinander vergleichen und so das BN[9] $B_{best} \in \mathbb{B}$ bestimmen, welches die

[8] zum Zustandsraum S_X der Variablen X siehe Definition A.1 und zu den Eltern Pa_v einer Variable bzw. eines Knotens v siehe Definition A.17

[9] Der Einfachheit halber werde angenommen, dass immer genau ein bestes BN vorliegt. Natürlich können in der Realität auch mehrere BNe die gleiche Güte aufweisen und somit

Daten D am besten widerspiegelt.

Bisher sind zwei grundlegende Arten von Gütemaßen für BNe vorgeschlagen worden [CGH97, S. 509]: Bayes'sche und informationstheoretische. Das bekannteste Gütemaß einer jeden Spezies soll hier vorgestellt werden. Zu Beginn sei allerdings eine spezielle Notation eingeführt [Bou95, S. 60]: Es sei v_i eine Variable in V, r_i sei die Anzahl der Zustände von v_i ($r_i = |S_{v_i}|$), und n sei die Anzahl der Variablen ($n = |V|$). Es bezeichne π_i die Menge der Eltern von v_i ($\pi_i = Pa_{v_i}$) und q_i die Anzahl möglicher Konfigurationen[10] von π_i ($q_i = |S_{\pi_i}|$). Wenn $\pi_i = \emptyset$ ist, dann sei $q_i = 1$. Die Elemente von S_{v_i} und S_{π_i} werden wie folgt indiziert: $x_{i1}, \ldots, x_{ir_i}$ respektive $x_{\pi_i 1}, \ldots, x_{\pi_i q_i}$. Es bezeichne x_{ik} den k-ten Zustand der i-ten Variable v_i und $x_{\pi_i j}$ die j-te Konfiguration der Eltern π_i der i-ten Variable v_i. Es sei N die Anzahl der Fälle bzw. Datensätze in der Datenbasis D. Es sei N_{ijk} die Anzahl der Fälle in D, in denen $v_i = x_{ik}$ und $\pi_i = x_{\pi_i j}$ ist, und N_{ij} sei die Anzahl der Fälle in D, in denen $\pi_i = x_{\pi_i j}$ ist. (Demzufolge ist $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. [CH92, S. 315])

Cooper und Herskovits [CH92, S. 316] schlagen ein Bayes'sches Gütemaß vor, welches als Güte die gemeinsame Wahrscheinlichkeit über dem BN B bzw. über seiner Struktur und über der Datenbasis D errechnet:

$$\Pr(B, D) = \Pr(B) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \left[\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right] \tag{4.9}$$

Der erste Faktor, $\Pr(B)$, kann eine von einem Experten vorab geschätzte Wahrscheinlichkeit sein, mit der er bestimmte BNe aufgrund struktureller Eigenschaften höher oder niedriger wichtet als andere. Liegt kein Expertenwissen vor, sei $\Pr(B) = 1/|\mathbb{B}|$ für alle $B \in \mathbb{B}$. Die folgenden Faktoren berechnen, wie gut die Struktur des BNes B zur Datenbasis D passt. Die Berechnung zieht die Anzahl der Fälle in D heran, in denen bestimmte Eltern-(Kind-)Konfigurationen vorkommen. Für eine Herleitung der Formel 4.9 sei auf [CH92, S. 338 ff.] verwiesen.

Ein informationstheoretisches Gütemaß schlägt Bouckaert [Bou94] vor. Er bezeichnet das Gütemaß als minimale Beschreibungslänge (Minimum Description Length, MDL):

$$\mathrm{MDL}(B, D) = \log \Pr(B) - N \cdot H(B, D) - \frac{1}{2} \mathbb{K} \cdot \log N. \tag{4.10}$$

Die MDL setzt sich aus drei Bestandteilen zusammen:

auch mehrere beste BNe existieren. Dann wird, liegen keine anderen Kriterien vor, ein beliebiges BN ausgewählt.

[10] zur Konfiguration s_X der Variablen X siehe Definition A.2

1. Die vorab durch Experten geschätzte Wahrscheinlichkeit für das BN B lautet wie beim Bayes'schen Gütemaß $\Pr(B)$.

2. Die Länge der Beschreibung der Datenbasis D durch die Verteilung, die das BN B kodiert, ist

$$H(B,D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} -\frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}. \tag{4.11}$$

3. Die Länge der Beschreibung der Verteilung, die das BN B kodiert, ist

$$\mathbb{K} = \sum_{i=1}^{n} (r_i - 1) q_i \tag{4.12}$$

   und entspricht der Anzahl der bedingten Wahrscheinlichkeiten der Bewertungsfunktionen im BN.

Abbildung 4.5 zeigt, wie die drei Bestandteile der MDL qualitativ zusammen- und von der Beschreibungslänge $\mathbb{K}$ der Verteilung abhängen, die das BN B kodiert. Das Maximum von $\mathrm{MDL}(B,D)$ entspricht dem BN mit der optimalen Struktur. Für die Herleitung der MDL und für einen theoretischen und praktischen Vergleich beider Gütemaße sei auf [Bou95, S. 65 ff., S. 68 ff. bzw. S. 114 ff.] verwiesen.

Ein Rechenbeispiel verdeutliche die Anwendung der Gütemaße. Es liege eine Datenbasis D mit $N = 10$ Fällen über den drei binären Variablen $V = \{re, sp, bo\}$ aus dem Beispiel „Marie und ihre Rosen“ vor. Marie habe die Fälle in D gesammelt. Tabelle 4.3 stellt die Datenbasis dar. Es seien weiterhin zwei alternative BNe $\mathbb{B} = \{B_1, B_2\}$ über den Variablen V erstellt worden. Die Graphen G_1 und G_2 der BNe stellt Abbildung 4.6 in a) und b) dar. Mittels eines Gütemaßes kann man nun bestimmen, welches BN $B \in \mathbb{B}$ der Datenbasis D am besten entspricht. Die Berechnung der Güte $\Pr(B_1, D)$ für

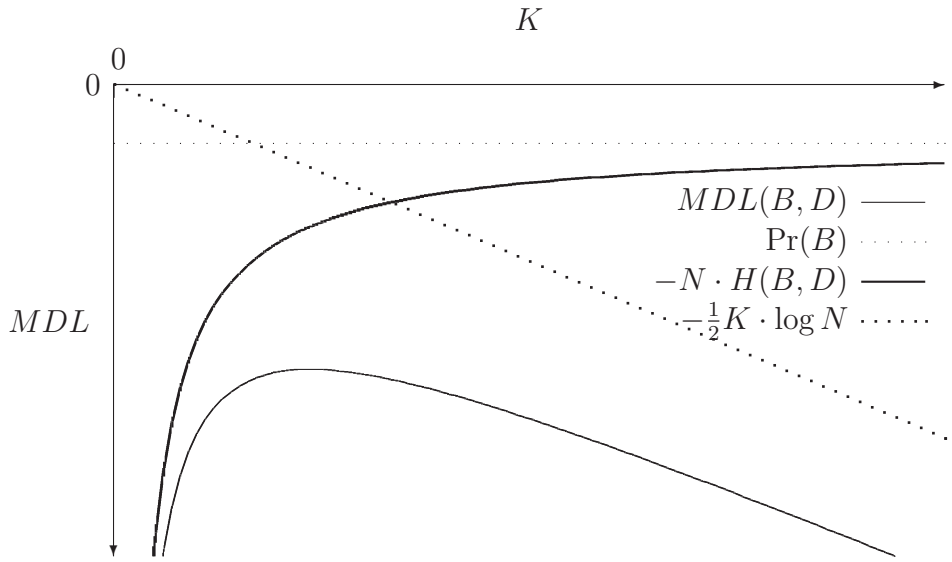

Abbildung 4.5: Qualitativer Zusammenhang zwischen der Beschreibungslänge $\mathbb{K}$ des BNes (entspricht laut Formel 4.12 der Anzahl der bedingten Wahrscheinlichkeiten im BN), den Bestandteilen des MDL-Gütemaßes und dem MDL-Gütemaß selbst [Bou95, S. 65]

Tabelle 4.3: Datenbasis D mit $N = 10$ Fällen über den drei binären Variablen $V = \{re, sp, bo\}$ aus dem Beispiel „Marie und ihre Rosen“, Abschnitt 4.1.1

#	re	sp	bo
1	ne_{re}	au_{sp}	tr_{bo}
2	ja_{re}	au_{sp}	fe_{bo}
3	ja_{re}	au_{sp}	fe_{bo}
4	ja_{re}	au_{sp}	fe_{bo}
5	ne_{re}	au_{sp}	fe_{bo}
6	ne_{re}	au_{sp}	tr_{bo}
7	ja_{re}	au_{sp}	fe_{bo}
8	ne_{re}	ei_{sp}	fe_{bo}
9	ne_{re}	au_{sp}	tr_{bo}
10	ne_{re}	au_{sp}	tr_{bo}

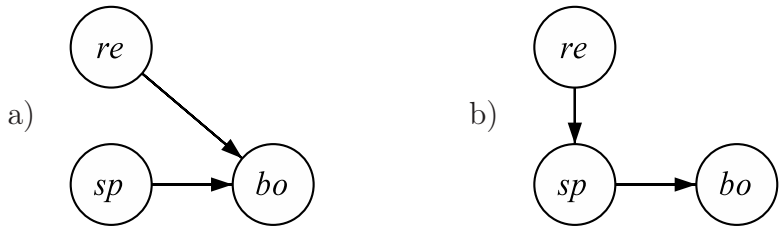

Abbildung 4.6: Graphen zweier alternativer BNe $\mathbb{B} = \{B_1, B_2\}$ über den Variablen $V = \{re, sp, bo\}$ aus dem Beispiel „Marie und ihre Rosen“: a) Graph G_1 des BNes B_1 und b) Graph G_2 des BNes B_2

das BN B_1 mittels des Bayes'schen Gütemaßes lautet wie folgt:

$$
\begin{aligned}
\Pr(B_1, D) &= \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \\
&= \prod_{i=1}^{3} \prod_{j=1}^{q_i} \frac{(2 - 1)!}{(N_{ij} + 2 - 1)!} \prod_{k=1}^{2} N_{ijk}! \\
&= \left(\prod_{j=1}^{1} \frac{1}{(N_{1j} + 1)!} \prod_{k=1}^{2} N_{1jk}! \right) \left(\prod_{j=1}^{1} \frac{1}{(N_{2j} + 1)!} \prod_{k=1}^{2} N_{2jk}! \right) \\
&\quad \left(\prod_{j=1}^{4} \frac{1}{(N_{3j} + 1)!} \prod_{k=1}^{2} N_{3jk}! \right) \\
&= \left(\frac{1}{(N_{11} + 1)!} N_{111}! N_{112}! \right) \left(\frac{1}{(N_{21} + 1)!} N_{211}! N_{212}! \right) \\
&\quad \left(\frac{1}{(N_{31} + 1)!} N_{311}! N_{312}! \quad \frac{1}{(N_{32} + 1)!} N_{321}! N_{322}! \right. \\
&\quad \left. \frac{1}{(N_{33} + 1)!} N_{331}! N_{332}! \quad \frac{1}{(N_{34} + 1)!} N_{341}! N_{342}! \right) \\
&= \left(\frac{1}{(10 + 1)!} 6!4! \right) \left(\frac{1}{(10 + 1)!} 9!1! \right) \left(\frac{1}{(5 + 1)!} 4!1! \right. \\
&\quad \left. \frac{1}{(1 + 1)!} 0!1! \quad \frac{1}{(4 + 1)!} 0!4! \quad \frac{1}{(0 + 1)!} 0!0! \right) \\
&\approx 1{,}31 \cdot 10^{-8}
\end{aligned}
$$

Da für kein BN $B \in \mathbb{B}$ eine Wahrscheinlichkeit $\Pr(B)$ vorgelegen hat, ist sie – da für alle $B \in \mathbb{B}$ konstant – aus der Berechnung ausgenommen worden. Die Güte $\Pr(B_2, D)$ ist für das BN B_2 analog mittels des Bayes'schen Gütemaßes berechnet worden. Sie beträgt rund $8{,}18 \cdot 10^{-10}$. Da es sich bei den berechneten

Güten um Wahrscheinlichkeiten handelt, kann man sagen, dass das BN B_1

$$\frac{\Pr(B_1, D)}{\Pr(B_2, D)} = \frac{1{,}31 \cdot 10^{-8}}{8{,}18 \cdot 10^{-10}} \approx 16{,}04$$

mal wahrscheinlicher ist als das BN B_2. Das soll nicht verwundern, da die Datenbasis D in Tabelle 4.3 in Wahrheit nicht von Marie zusammengetragen, sondern mittels stochastischer Vorwärtssimulation (siehe Algorithmus A.22) aus einem BN erzeugt worden ist, dem der Graph aus Abbildung 4.6 zugrunde liegt. Bei dem BN, das zur Simulation verwandt worden ist, hat es sich um einen Ausschnitt aus dem BN zu „Marie und ihre Rosen“ gehandelt.

Wendet man die MDL auf die zwei BNe B_1 und B_2 und auf die Datenbasis D an, erhält man $\mathrm{MDL}(B_1, D) \approx -19{,}39$ und $\mathrm{MDL}(B_2, D) \approx -21.37$. Ergo präferiert auch die MDL das BN B_1. Ein sogenanntes Voting, eine Vorgehensweise im Rahmen des Data-Mining, bei der verschiedene Verfahren über alternative Lösungen abstimmen [CS95, S. 90 f.], wäre einstimmig zugunsten des BNes B_1 ausgegangen.

Die vorgestellten Gütemaße können natürlich nicht nur auf gewöhnliche BNe und unabhängige Fälle in einer Datenbasis, sondern auch auf DBNe und multivariate Zeitreihen angewandt werden. [FMR98] [Mur02, S. 102 ff.] Für gewöhnlich werden das Eingangs- und das Übergangsnetz separat bewertet. Als Datenbasen liegen mehrere multivariate Zeitreihen über einer Menge V von Variablen vor. Die jeweils ersten Datensätze bilden die Datenbasis zum Berechnen der Güte des Eingangsnetzes. Zum Berechnen der Güte des Übergangsnetzes wird aus den multivariaten Zeitreihen eine neue Datenbasis gebildet, indem jeweils zwei Datensätze, die in einer multivariaten Zeitreihe aufeinanderfolgen, in der Reihenfolge ihres Auftretens zu einem neuen Datensatz verkettet und der Datenbasis hinzugefügt werden. Tabelle 4.4 zeigt ein Beispiel für eine multivariate Zeitreihe. Es bezeichne s_i^k den Zustand der Variable v_i zum Zeitpunkt k mit $1 \leq i \leq n$ und $0 \leq k \leq N - 1$. Tabelle 4.5 zeigt eine Datenbasis zum Berechnen der Güte eines Übergangsnetzes, die aus der multivariaten Zeitreihe aus Tabelle 4.4 erstellt worden ist. Natürlich ist es nicht nötig, die Zeitreihen derart redundant zu halten. Beim rechnergestützten Messen der Güte eines Übergangsnetzes bezüglich einer Datenbasis sollten geeignete Datenstrukturen und eine geschickte Indizierung angewandt werden, um darauf zu verzichten und den z. T. nicht unerheblichen Speicherbedarf für die Datenbasis nicht noch weiter zu erhöhen.

Bisher ist davon ausgegangen worden, dass alternative BNe bzw. ihre Graphen vorliegen und man mittels eines Gütemaßes dasjenige auswählt, welches die Datenbasis am besten widerspiegelt. Wie man automatisch zu alternativen BNen gelangt, beschreibt der folgende Abschnitt.

Tabelle 4.4: Beispiel für eine multivariate Zeitreihe über $v_1, \ldots, v_n$ Variablen und $0, \ldots, N-1$ Zeitpunkte.

t	v_1	$\ldots$	v_n
0	s_1^1	$\ldots$	s_n^1
1	s_1^2	$\ldots$	s_n^2
2	s_1^3	$\ldots$	s_n^3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$N-2$	s_1^{N-2}	$\ldots$	s_n^{N-2}
$N-1$	s_1^{N-1}	$\ldots$	s_n^{N-1}

Tabelle 4.5: Beispiel für eine Datenbasis über den Variablen $v_1, \ldots, v_n, v_1, \ldots, v_n$ und $|\{(0,1),(1,2),\ldots,(N-2,N-1)\}| = N-2$ aufeinanderfolgenden Zustandsübergängen

$(t, t+1)$	v_1	$\ldots$	v_n	v_1	$\ldots$	v_n
$(0,1)$	s_1^1	$\ldots$	s_n^1	s_1^2	$\ldots$	s_n^2
$(1,2)$	s_1^2	$\ldots$	s_n^2	s_1^3	$\ldots$	s_n^3
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(N-2,N-1)$	s_1^{N-2}	$\ldots$	s_n^{N-2}	s_1^{N-1}	$\ldots$	s_n^{N-1}

4.2.4 Algorithmen zum automatischen Erstellen BNe aus Daten

Wenn kein oder nicht genug Expertenwissen über die Produktion oder die PPS zum manuellen Erstellen eines BNes vorliegt, bestehen Möglichkeiten, BNe aus PPS-Daten D zu erstellen. BNe, die aus PPS-Daten erstellt worden sind, können wie manuell erstellte BNe zur probabilistischen Inferenz verwandt werden. [CH92, S. 307] Sie gewähren aber darüber hinaus (einen besseren) Einblick in den Problembereich [Hec99, S. 302] – hier in den Bereich der PPS – und sind sehr gut als Erklärungsmodelle zu verwenden [Nea04, S. 216 ff.].

Eine Möglichkeit, BNe automatisch aus Daten D zu erstellen, bestünde darin, alle für eine gegebene Menge V möglichen BNe bzw. gerichteten, azyklischen Graphen zu enumerieren, für jedes BN die Güte anhand D zu berechnen und das beste BN bzw. die besten BNe – es können auch mehrere Netze gleicher respektive höchster Güte auftreten – auszuwählen. Die Anzahl g aller für $|V| = n$ Knoten möglichen gerichteten, azyklischen Graphen ergibt

sich allerdings aus folgender rekursiver Formel[11] [Har00, S. 200]

$$g(n) = \begin{cases} 1 & \text{für} \quad n = 0, \\ \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} g(n-i) & \text{für} \quad n > 0, \end{cases} \tag{4.13}$$

und wächst stärker als exponentiell mit steigendem n [Hec96, S. 24 f.], weshalb die Möglichkeit ausscheidet, alle möglichen BNe zu enumerieren. Bereits bei $n = 5$ wären knapp $g(5) \approx 30.000$ Netze zu untersuchen; bei $n = 10$ wären es bereits über vier Trillionen ($g(10) > 4 \cdot 10^{18}$).

Da die PPS ein sehr komplexes Aufgabengebiet ist und meist viele Variablen berücksichtigt werden müssen, ist eine vollständige Enumeration aller möglichen BNe erst recht nicht praktikabel. Es sind allerdings mehrere heuristische Suchalgorithmen entworfen worden, die zwar nicht garantieren, das BN zu finden, das die Daten am besten widerspiegelt, die aber ein BN finden, das nah am Optimum liegt. [Nea04, S. 503] Diese Algorithmen tasten zum einen nicht den gesamten Suchraum ab. Zum anderen berechnen sie auch nicht die Gesamtgüte des BNes, sondern lediglich, wie sich die Gesamtgüte des BNes durch das Hinzufügen von Kanten oder das Entfernen von Kanten verändert. Letzteres wird dadurch ermöglicht, dass sich die Gesamtgüte des BNes als Summe der Beiträge ausdrücken lässt, die jede Variable des BNes leistet. Die Summe wird noch um einen konstanten Summanden ergänzt. [Bou95, S. 92 f.] Wie gut ein Knoten inklusive seiner Eltern die Datenbasis D widerspiegelt bzw. welchen Beitrag er zur Gesamtgüte des BNes leistet, berechnen die Formeln 4.14 und 4.15 für das Bayes'sche Gütemaß respektive für die MDL.[12]

$$q(v_i, \pi_i, D) = \sum_{j=1}^{q_i} \left(\log \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} + \sum_{k=1}^{r_i} \log N_{ijk}! \right) \tag{4.14}$$

$$q(v_i, \pi_i, D) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{1}{2} q_i (r_i - 1) \log N \tag{4.15}$$

Beide Maße ermöglichen es Suchalgorithmen, effizient BNe mit hoher Güte zu finden. Zwei alternative effiziente Algorithmen stellt Anhang A.2 vor: den K2-Algorithmus A.1 und den B-Algorithmus A.2. Modifikationen von $K2$ und B erlauben es, vorab

[11] Die Formel ist von [Rob73] entwickelt worden. In [Bou95, S. 90] liegt ein (Tipp)fehler vor: Die 1 im zweiten Exponenten ist durch ein i zu ersetzen.

[12] Für den Beweis der Formeln 4.14 und 4.15 sei auf [Bou95, S. 93] verwiesen.

1. Kanten im BN bzw. Eltern eines Knotens vorzugeben,

2. Kanten im BN bzw. Eltern eines Knotens zu verbieten und

3. die Anzahl der Eltern pro Variable nach oben zu begrenzen.

Die Punkte 1 und 2 ermöglichen es, partiell vorhandenes Expertenwissen über direkte kausale Beziehungen zwischen Variablen bzw. über das Fehlen solcher Beziehungen vorzugeben. Punkt 2 ist besonders geeignet, um beim Erstellen des Übergangsnetzes eines DBNes verbotene Kanten auszuschließen. Die vorgestellten Algorithmen können demzufolge nicht nur gewöhnliche BNe aus unabhängigen Fällen einer Datenbasis erstellen, sondern auch DBNe aus multivariaten Zeitreihen. [FMR98], [Mur02, S. 102 ff.] Die Algorithmen erstellen lediglich separat das Eingangs- und das Übergangsnetz und nicht direkt das entrollte DBN. Daraus resultiert ein sehr stark reduzierter Rechenaufwand. Das entrollte DBN wird anschließend aus einem Eingangs- und mehreren identischen Übergangsnetzen zusammengesetzt. Das algorithmische Erstellen der Netze benötigt Datenbasen. Wie selbige aus den multivariaten Zeitreihen gewonnen werden, beschreibt bereits Abschnitt 4.2.3 an seinem Ende. Punkt 3 kann angewandt werden, wenn das BN zu komplex werden würde, um bei akzeptablem Rechenaufwand probabilistische Inferenz über ihm zu vollziehen.

Natürlich besteht auch die Möglichkeit, universelle Metaheuristiken (siehe z. B. [MF00] und [Fel99]) zur Suche nach BNen hoher Güte zu verwenden. Diese Verfahren entfernen oder invertieren auch eine oder mehrere Kanten und lassen eine temporäre Verschlechterung des BNes zu, um lokalen Optima zu entkommen. Mehrere Metaheuristiken sind bereits erfolgreich erprobt worden (siehe z. B. [Bou95, S. 120 ff.]). Allerdings haben sie im Vergleich mit K2 und B nur einen geringen Gütezuwachs verzeichnet, verursachen einen wesentlichen höheren Rechenaufwand und werden deshalb nicht uneingeschränkt empfohlen. [Nea04, S. 519 f.], [Bou95, S. 124] Die diskrete Struktur des Problems legt es nahe, auch parallele genetische Algorithmen [Mic96], [Nis94] [Gol89] oder einen Ruin&Recreate-Algorithmus [Hep04], [SSSWD00], [Pöp00] auf ihre Eignung zu untersuchen, das Optimierungsproblem zu lösen. Genetische Algorithmen haben [KN04, S. 211] bereits mit mäßigem Erfolg angewandt. Das Potenzial dieser Verfahrensgruppe kann jedoch noch nicht als ausgeschöpft betrachtet werden.

Ein anderer Ansatz zum Erstellen der Struktur eines BNes aus einer Datenbasis D arbeitet constraint-basiert. [Nea04, S. 533 ff.] Er geht auf [SGS93] zurück und ist in [SSGM94] umgesetzt worden. Nach diesem Ansatz werden in einem ersten Schritt aus einem vollständig verbundenen ungerichteten Graphen über V sukzessive ungerichtete Kanten $(u, v), (v, u) \in V$ dann entfernt,

wenn sie, bedingt durch Nachbarn von u oder v, voneinander unabhängig sind. Die bedingte Unabhängigkeit von u und v wird aufgrund der Datenbasis ermittelt. In einem zweiten Schritt werden adjazente Kanten des Graphen nach plausiblen Annahmen gerichtet. Das constraint-basierte Erstellen BNe aus Daten soll hier nicht im Detail dargestellt werden.

4.2.5 Schätzen bedingter Wahrscheinlichkeiten für BNe

Nachdem die Struktur des BNes bestimmt worden ist, sind seine bedingten Wahrscheinlichkeiten[13] zu schätzen, die aussagen, wie wahrscheinlich sich eine Variable v in einem Zustand s_v befindet, wenn sich ihre Eltern Pa_v, die sie direkt und kausal beeinflussen, im Zustand bzw. in der Konfiguration[14] s_{Pa_v} befinden. Das Schätzen der Wahrscheinlichkeiten kann manuell oder anhand einer Datenbasis erfolgen. Wenn keine Daten vorliegen, schätzt ein Experte die bedingten Wahrscheinlichkeiten. Da ihm das unter Umständen schwerfällt, sind mehrere Operatoren entwickelt worden, die das Schätzen der bedingten Wahrscheinlichkeiten vereinfachen, so z. B. das verrauschte Oder (Noisy-OR), das verrauschte Maximum (Noisy-MAX), das verrauschte Und (Noisy-AND) und das verrauschte Minimum (Noisy-MIN). [Tol98, S. 17 ff.] Noisy-MAX und Noisy-MIN sind Verallgemeinerungen von Noisy-OR bzw. Noisy-AND. Diese Operatoren senken die Anzahl der zu schätzenden bedingten Wahrscheinlichkeiten drastisch, indem sie es erlauben, bedingte Wahrscheinlichkeiten nur in Abhängigkeit von jeweils einer Bedingung zu schätzen, und aus diesen einfach bedingten Wahrscheinlichkeiten später mehrfach bedingte zu ermitteln. Allerdings können die Operatoren nur für binäre bzw. ordinale Variablen angewandt werden. Das stellt im Kontext der PPS kein Problem dar, da in der PPS viele binäre und ordinale Variablen auftreten und auch kontinuierliche Variablen diskretisiert und wie rein ordinale Variablen behandelt werden können. Eine binäre Variable ist z. B. der Belegungsstatus eines Berarbeitungskanals einer Maschine mit den Zuständen frei und belegt, eine ordinale Variable ist z. B. die Länge der Warteschlange vor einer Maschine mit den Zuständen leer, kurz, mittel und lang, und eine kontinuierliche Variable ist z. B. die Kapazitätsauslastung mit dem Wertebereich $[0; 1]$, die in drei ordinale Zustände diskretisiert werden könnte: $[0; 0{,}25]$ – niedrig, $(0{,}25; 0{,}6]$ – mittel und $(0{,}6; 1]$ – hoch.

Als ein Operator zur effizienten Wissensakquisition soll Noisy-OR exemplarisch en detail vorgestellt werden. Noisy-OR ist eine verallgemeinerte

[13] Eine bedingte Wahrscheinlichkeit ist die Wahrscheinlichkeit dafür, dass ein Ereignisses B eintritt, wenn bereits ein Ereignis A eingetreten ist, und wird als als $\Pr(B|A)$ geschrieben. [BSMM99, S. 625]

[14] zur Konfiguration s_X der Variablen X siehe Definition A.2

logische Oder-Verknüpfung, bei der das Vorliegen mindestens eines Eingangssignales nicht zwangsläufig ein Ausgangssignal nach sich zieht. Um Noisy-OR anwenden zu können müssen zwei Voraussetzungen erfüllt sein: Die Zustände der bedingenden Variablen treten voneinander unabhängig auf, und sämtliche beteiligte Variablen seien binäre Variablen, deren Zustände Boole'schen Charakter tragen. Es sei v eine durch ihre Eltern $Pa = \{u_1, \ldots, u_k\}$ bedingte Variable. Die Variable v kann die Zustände $\{\bar{s}, s\}$ annehmen und die Variable u_i die Zustände $\{\bar{s}_i, s_i\}$ mit $i = 1, \ldots, k$. Um Noisy-OR anzuwenden, schätzt man lediglich die bedingte Wahrscheinlichkeit $\Pr(v = s|u_i = s_i)$ für $i = 1, \ldots, k$, wodurch der Aufwand zum Schätzen der bedingten Wahrscheinlichkeiten nur noch linear mit $|Pa_v|$ steigt. Da sämtliche Variablen binär sind, ergibt sich folgende komplementäre bedingte Wahrscheinlichkeit $\Pr(v = \bar{s}|u_i = s_i) = 1 - \Pr(v = s|u_i = s_i)$ für $i = 1, \ldots, k$. Mittels Noisy-OR ermittelt man die bedingten Wahrscheinlichkeiten

$$\Pr(v = \bar{s}|u_1 = x_1, \ldots, u_k = x_k) = \prod_{\{u_i \in Pa | x_i = s_i\}} \Pr(v = \bar{s}|u_i = s_i) \quad (4.16)$$

und

$$\Pr(v = s|u_1 = x_1, \ldots, u_k = x_k) = 1 - \Pr(v = \bar{s}|u_1 = x_1, \ldots, u_k = x_k) \quad (4.17)$$

für alle $(x_1, \ldots, x_k) \in S_{Pa}$. Ist kein $x_i = s_i$ bzw. sind alle $x_i = \bar{s}_i$ für $i = 1, \ldots, k$, ist die bedingte Wahrscheinlichkeit $\Pr(v = \bar{s}|u_1 = x_1, \ldots, u_k = x_k) = 1$, und die bedingte Wahrscheinlichkeit $\Pr(v = s|u_1 = x_1, \ldots, u_k = x_k) = 0$. Das würde bedeuten, dass man im Falle $u_i = \bar{s}_i$ mit $i = 1, \ldots, k$ sämtliche anderen Ursachen für $v = s$ ausschließen könnte und kein zufälliger Einfluss vorläge. Um die 0 zu vermeiden, kann man eine sogenannte Hintergrundvariable u_{k+1} mit den Zuständen $\{\bar{s}_{k+1}, s_{k+1}\}$ temporär einführen, $Pa \leftarrow Pa \cup \{u_{k+1}\}$, die sich immer im Zustand s_{k+1} befindet.

Ein Beispiel soll die Anwendung des Noisy-OR verdeutlichen. Eine Variable c mit $S_c = \{\bar{s}_c, s_c\}$ habe die Eltern a und b mit $S_a = \{\bar{s}_a, s_a\}$ respektive $S_b = \{\bar{s}_b, s_b\}$. Die geschätzten Wahrscheinlichkeiten lauten $\Pr(c = s_c|a = s_a) = 0{,}8$ und $\Pr(c = s_c|b = s_b) = 0{,}7$, woraus sich die komplementären bedingten Wahrscheinlichkeiten $\Pr(c = \bar{s}_c|a = s_a) = 0{,}2$ und $\Pr(c = \bar{s}_c|b = s_b) = 0{,}3$ ergeben. Tabelle 4.6 zeigt die mittels Noisy-OR ermittelten bedingten Wahrscheinlichkeiten, und wie sie ermittelt worden sind. Im Gegensatz zu Noisy-OR und -AND sind Noisy-MAX und -MIN nicht nur auf binäre Variablen mit wahr/falsch-Charakter anzuwenden, sondern auch auf ordinale Variablen, deren Zustände man ordnen kann und wie sie in der PPS des öfteren vorliegen. Bezüglich einer detaillierten Beschreibung der Noisy-Operatoren sei auf die Literatur verwiesen. Noisy-MAX ist z. B. in [Dź93] und [PPMH94] unabhängig voneinander vorgestellt worden.

Tabelle 4.6: Beispiel für eine mittels Noisy-OR erstellte Bewertungsfunktion

a	b	c	$\Pr(c = \cdot \mid a = \cdot, b = \cdot)$
$\bar{s}_a$	$\bar{s}_b$	$\bar{s}_c$	1=1
$\bar{s}_a$	$\bar{s}_b$	s_c	$1 - \Pr(\bar{s}_c \mid \bar{s}_a, \bar{s}_b) = 1 - 1$=0
$\bar{s}_a$	s_b	$\bar{s}_c$	$1 \cdot \Pr(\bar{s}_c \mid s_b) = 1 \cdot 0{,}3$=0,3
$\bar{s}_a$	s_b	s_c	$1 - \Pr(\bar{s}_c \mid \bar{s}_a, s_b) = 1 - 0{,}3$=0,7
s_a	$\bar{s}_b$	$\bar{s}_c$	$1 \cdot \Pr(\bar{s}_c \mid s_a) = 1 \cdot 0{,}2$=0,2
s_a	$\bar{s}_b$	s_c	$1 - \Pr(\bar{s}_c \mid s_a, \bar{s}_b) = 1 - 0{,}2$=0,8
s_a	s_b	$\bar{s}_c$	$1 \cdot \Pr(\bar{s}_c \mid s_a) \cdot \Pr(\bar{s}_c \mid s_b) = 1 \cdot 0{,}2 \cdot 0{,}3$=0,06
s_a	s_b	s_c	$1 - \Pr(\bar{s}_c \mid s_a, s_b) = 1 - 0{,}06$=0,94

Liegt eine adäquate Datenbasis D vor, so können die bedingten Wahrscheinlichkeiten der Bewertungsfunktionen des BNes nach verschiedenen Formeln geschätzt werden. Eine *offensichtliche* Formel zum Schätzen einer bedingten Wahrscheinlichkeit lautet

$$\Pr(v_i = x_{ik} \mid \pi_i = x_{\pi_i j}) = \frac{N_{ijk}}{N_{ij}}. \tag{4.18}$$

Die Notation entspricht der im Abschnitt 4.2.3 vereinbarten. Es soll also die Wahrscheinlichkeit dafür geschätzt werden, dass sich die Variable v_i in ihrem k-ten Zustand x_{ik} befindet, wenn sich die Eltern π_i dieser Variable v_i in ihrem j-ten Zustand $x_{\pi_i j}$ befinden. Zwei Anzahlen N_{ij} und N_{ijk} an Fällen werden in der Datenbasis D ermittelt: Die Anzahl N_{ij} ist die Anzahl der Fälle, in denen die Eltern π_i der i-ten Variable v_i sich in ihrem j-ten Zustand $x_{\pi_i j}$ befinden. Die Anzahl N_{ijk} ist die Anzahl der Fälle, in denen die Eltern π_i der i-ten Variable v_i sich in ihrem j-ten Zustand $x_{\pi_i j}$ befinden und in denen sich *zusätzlich* die i-te Variable v_i in ihrem k-ten Zustand x_{ik} befindet. Der Quotient $\frac{N_{ijk}}{N_{ij}}$ schätzt die besagte bedingte Wahrscheinlichkeit. Formel 4.18 ist zur Herleitung des MDL-Gütemaßes verwandt worden.

Eine weitere Formel zum Schätzen der bedingten Wahrscheinlichkeiten der Bewertungsfunktionen des BNes, die zur Herleitung des Bayes'schen Gütemaßes verwandt worden ist, lautet

$$\Pr(v_i = x_{ik} \mid \pi_i = x_{\pi_i j}) = \frac{N_{ijk} + 1}{N_{ij} + r_i}, \tag{4.19}$$

wobei r_i die Anzahl der Zustände $|S_{v_i}|$ der i-ten Variable v_i bezeichnet. Für die Herleitung und Interpretation dieser Formel sei auf [CH92, S. 342 ff.] verwiesen. Formel 4.18 weist gegenüber der Formel 4.19 den Vorteil auf, dass

keine Division durch 0 auftreten kann und die resultierende bedingte Wahrscheinlichkeitsfunktion gleichverteilt ist, wenn keine passenden Fälle vorliegen. Sind die Anzahlen N_{ijk} und N_{ij} der Fälle in der Datenbasis sehr hoch, nähern sich die Ergebnisse der Formeln 4.18 und 4.19 zudem einander an.

Beim Schätzen der Wahrscheinlichkeiten aus Daten können natürlich auch die oben beschriebenen Noisy-Operatoren angewandt werden, wenn nicht genug Datensätze für Ausprägungskombinationen der Eltern einer Variable vorliegen.

Nachdem das BN nun erstellt worden ist, kann man es zumindest als qualitatives Erklärungsmodell verwenden. Wie über dem BN wahrscheinlichkeitsbasiert geschlussfolgert wird, beschreibt der folgende Abschnitt.

4.3 Wissensanwendung: probabilistische Inferenz über BNen

4.3.1 Grundlagen und Aufgaben der probabilistischen Inferenz

Probabilistische Inferenz (PI) bedeutet soviel wie wahrscheinlichkeitsbasiertes Schließen oder auch Schlussfolgern. Mittels PI werden Randwahrscheinlichkeitsfunktionen oder wahrscheinlichste Konfigurationen der Variablen V eines BNes ermittelt. Ein zentraler Begriff im Rahmen der PI ist der Begriff der Evidenz:

Definition 4.7 (Evidenz)
Es sei $U \subseteq V$, und S_U sei der Zustandsraum von U. Eine Konfiguration $s_U \in S_U$, von der man weiß, dass sich die Variablen U in ihr befinden, wird als Evidenz bezeichnet. Die Variablen in U werden als evidente Variablen bezeichnet.

Sofern Evidenz s_U vorliegt, können die evidenten Variablen U des BNes auf s_U gesetzt werden. Das Setzen von U auf s_U heißt Instanziieren. Sind Variablen $U \subseteq V$ instanziiert worden, werden mittels PI Randwahrscheinlichkeitsfunktionen oder wahrscheinlichste Konfigurationen der Variablen $V \setminus U$ des BNes unter der Bedingung $U = s_U$ ermittelt.

An dieser Stelle seien am Beispiel „Marie und ihre Rosen“ einige Fragen gestellt und zugehörige Aufgaben aufgeführt, die mittels PI über dem BN beantwortet respektive gelöst werden können. Tabelle 4.7 stellt die Fragen und die zugehörigen Aufgaben vor. Die geklammerten Teilsätze in der Fragenspalte symbolisieren die Evidenz, die gesetzt werden kann, sofern sie denn

Tabelle 4.7: Aufgaben der PI über BNen und zugehörige Fragen im Kontext von „Marie und ihre Rosen“, die PI über BNen beantworten kann

Nr.	Frage	Aufgabe
1.	Wie wird es den Rosen gehen(, wenn die Straße nass und/oder der Sprenger an ist)?	Prognose
2.	Woran hat es gelegen, dass es den Rosen schlecht geht(, wenn die Straße trocken ist)?	Diagnose
3.	In welchen Zustand sollte der Sprenger versetzt werden, damit es den Rosen gutgeht(, wenn die Straße nass ist)?	Entscheidungs-unterstützung
4.	Kann es denn sein, dass es regnet, wenn die Straße trocken (und der Sprenger an) ist?	Erkennen von Messfehlern bzw. Fehlern in Daten
5.	Wie wirkt sich der Zustand des Sprengers darauf aus, wie es den Rosen geht(, wenn es regnet)?	Sensitivitäts-analyse
6.	Wie ist der Zustand der Straße gewesen, als es den Rosen gutgegangen und der Sprenger angewesen ist?	Bestimmen fehlender Werte
7.	Welche Konfiguration von Regen und Sprenger tritt vermutlich auf(, wenn es den Rosen gutgeht)?	Bestimmen der wahrscheinlichsten Konfiguration

vorliegt. Nachdem die Aufgaben der PI über BNen am Beispiel vorgestellt worden sind, sollen sie nun noch einmal allgemeiner gefasst werden:

1. Die *Prognose* ermittelt Wirkungen bzw. – zeitlich betrachtet – zukünftige Zustände von Variablen. Evidente Ursachen können instanziiert bzw. Variablen auf ihre gegenwärtigen Zustände gesetzt werden.

2. Die *Diagnose* ermittelt umgekehrt Ursachen bzw. – zeitlich betrachtet – vergangene Zustände von Variablen. Evidente Wirkungen (Symptome) können instanziiert bzw. Variablen auf gegenwärtige Zustände gesetzt werden.

3. Die *Entscheidungsunterstützung* ermittelt Konfigurationen für Stellgrößen bzw. Handlungsalternativen, die zu erwünschten Zuständen von Regelgrößen bzw. Zielen führen. Regelgrößen bzw. Ziele müssen mit erwünschten Zuständen instanziiert, und Zustandsgrößen können auf ihre derzeitigen Zustände gesetzt werden.

4. Zum *Erkennen von Messfehlern oder Fehlern in Daten* wird die Wahrscheinlichkeit von Werten in einem Datensatz bestimmt und anhand dieser Wahrscheinlichkeit der Datensatz als fehlerhaft erkannt. Sichere Werte im Datensatz können vorgegeben werden.

5. Die *Sensitivitätsanalyse* bestimmt, ob und – wenn ja – wie stark sich eine Variable ändert, wenn eine andere geändert wird. Evidente Zustände anderer Variablen können gesetzt werden.

6. Zum *Bestimmen fehlender Werte* in einem Datensatz werden die bekannten Variablen auf die vorhandenen Werte gesetzt, und die wahrscheinlichste Konfiguration für die unbekannten Variablen wird ermittelt. Diese Konfiguration umfasst die fehlenden Werte.

7. Zum *Bestimmen der wahrscheinlichsten Konfiguration* werden evidente Variablen instanziiert, und für ausgewählte andere Variablen wird die wahrscheinlichste Konfiguration bestimmt.[15]

Im Rahmen der PPS treten die vorgestellten Aufgaben, die mittels PI über BNen gelöst werden können, oft und an vielen verschieden Stellen auf. Im Folgenden soll deshalb en detail dargestellt werden, wie PI über BNen vollzogen wird.

[15]Bei den Aufgaben 1, 2, 3 und 6 werden wie bei Aufgabe 7 wahrscheinlichste Konfigurationen bestimmt. Diese Aufgaben können somit als Spezialfälle der Aufgabe 7 aufgefasst werden.

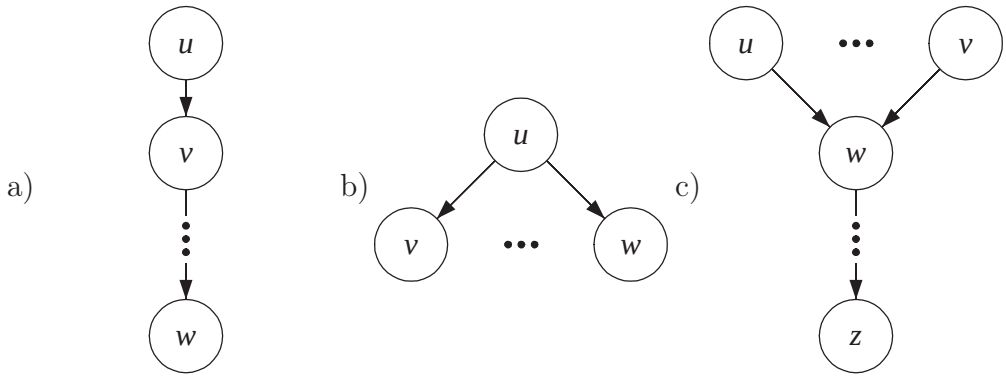

Abbildung 4.7: a) Serielle, b) divergierende und c) konvergierende Verbindungen zwischen Variablen in BNen

Entscheidend für die Anwendung BNe im Allgemeinen und für die PI im Besonderen ist es, ob vorliegende Evidenz nicht-evidente Variablen im BN beeinflusst. Das kann man bestimmen, bevor PI durchgeführt wird: Zwischen Variablen BNe können drei verschiedene Arten von Verbindungen vorliegen: serielle, divergierende und konvergierende. Abbildung 4.7 stellt die drei Verbindungsarten graphisch dar.

Im Falle der a) *seriellen Verbindung* wirkt u auf v, und v wirkt auf w. Für u vorliegende Evidenz beeinflusst v und via v auch w. Umgekehrt beeinflusst für w vorliegende Evidenz v und via v auch u. Wird v nun instanziiert, blockiert es den Einfluss der für u vorliegenden Evidenz auf w bzw. der für w vorliegenden Evidenz auf u. Wenn Evidenz für v vorliegt, sind die Variablen u und w voneinander unabhängig. Betrachten wir die Variablen Regen re, Boden bo und Rosen ro aus dem Beispiel „Marie und ihre Rosen“. (Ein Rasensprenger sp existiere nicht.): Wenn man weiß, dass es regnet, kann man darauf schließen, dass der Boden feucht ist und es den Rosen demzufolge gutgeht. Weiß man umgekehrt, dass es den Rosen gutgeht, kann man darauf schließen, dass der Boden feucht ist und es geregnet hat. Weiß man allerdings vorab, dass der Boden feucht ist, lässt erstens das Wissen, ob es geregnet hat, keinen (zusätzlichen) Schluss auf den Zustand der Rosen zu, und lässt zweitens das Wissen um den Zustand der Rosen keinen (zusätzlichen) Schluss darauf zu, ob es geregnet hat.

Im Falle der b) *divergierenden Verbindung* wirkt u auf v und w. Für v vorliegende Evidenz beeinflusst u und via u auch w, und für w vorliegende Evidenz beeinflusst u und via u auch v. Wird u nun instanziiert, blockiert es den Einfluss der für v vorliegenden Evidenz auf w bzw. der für w vorliegenden Evidenz auf v. Wenn Evidenz für u vorliegt, sind die Variablen v und w voneinander unabhängig. Betrachten wir die Variablen Regen re, Straße

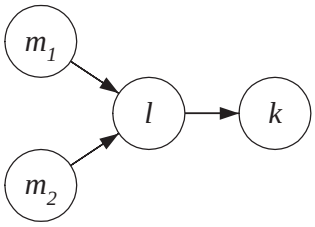

Abbildung 4.8: Kausale Zusammenhänge zwischen zwei Münzen, m_1 und m_2, und einer Lampe, l, (und einer Klingel, k) aus einem Beispiel

st und Boden *bo* aus dem Beispiel „Marie und ihre Rosen“. (Ein Rasensprenger *sp* existiere wiederum nicht.): Wenn man weiß, dass die Straße nass ist, kann man darauf schließen, dass es geregnet hat und der Boden demzufolge feucht ist. Weiß man umgekehrt, dass der Boden feucht ist, kann man darauf schließen, dass es geregnet hat und die Straße demzufolge nass ist. Weiß man allerdings vorab, dass es geregnet hat, lässt erstens das Wissen um den Zustand der Straße keinen (zusätzlichen) Schluss auf den Zustand des Bodens und zweitens das Wissen um den Zustand des Bodens keinen (zusätzlichen) Schluss auf den Zustand der Straße zu.

Im Falle der c) *konvergierenden Verbindung* wirken u und v auf w(, und w wirkt auf z). Liegt keine Evidenz für w (oder z) vor, sind u und v voneinander unabhängig. Wird w (oder z) nun instanziiert, beeinflusst für u vorliegende Evidenz v, und für v vorliegende Evidenz beeinflusst u. Für Fall c) böten sich die Variablen Regen *re*, Sprenger *sp* und Boden *bo* (und Rosen *ro*) aus Beispiel „Marie und ihre Rosen“ an. Allerdings ist die Unabhängigkeit zwischen den Variablen *re* und *sp* nur eine vereinfachende Annahme, da *sp* sicher in Abhängigkeit von *re* ein- bzw. ausgeschaltet wird. Aus diesem Grunde betrachten wir ein anderes Beispiel: Es liegen vier Variablen vor: zwei Münzen, m_1 und m_2, und eine Lampe, l(, und eine Klingel, k). Es werden m_1 und m_2 geworfen. Nur in dem Fall, dass beide Kopf oder beide Zahl zeigen, leuchtet l auf. (Und nur in dem Fall, dass l aufleuchtet, ertönt k.) Abbildung 4.8 veranschaulicht die Zusammenhänge zwischen den Variablen. Weiß man nun nichts über den Zustand von l (oder k), kann man nicht vom Zustand von m_1 auf den Zustand von m_2 schließen und umgekehrt. Kennt man allerdings den Zustand von l (oder den von k und demzufolge den von l), kann man vom Zustand von m_1 auf den Zustand von m_2 schließen und umgekehrt.

Die Regeln, ob vorliegende Evidenz nicht-evidente Variablen im BN beeinflusst, lassen sich im sogenannten d-Separationskriterium zusammenfassen:

Definition 4.8 (d-Separation)
Es sei B ein BN. Es sei $G = (V, A)$ der (gerichtete) Graph von B. Es sei $U = (V, E)$ der dem Graphen G unterliegende Graph. Es seien $u, v \in V$. Es sind u und v d-separiert, wenn für jeden einfachen Pfad p zwischen u und v in U gilt: Es liegt eine von u und v verschiedene Variable w auf p, so dass entweder gilt, dass die Verbindung zwischen u, w und v seriell oder divergierend und w instanziiert ist, oder dass gilt, dass die Verbindung zwischen u, w und v konvergierend ist und weder w noch seine Nachfahren instanziiert sind.

Will man mittels eines BNes z. B. die Produktion regeln, kann man mit Hilfe des d-Separationskriteriums bestimmen, ob (bei evidenten Zustandsgrößen) die Stellgrößen überhaupt Einfluss auf die Regelgrößen ausüben: Sind Stell- und Regelgrößen d-separiert, beeinflussen die Stellgrößen die Regelgrößen nicht, und das spezielle BN bzw. der spezielle Ansatz zur Regelung der Produktion ist nicht tragbar.

4.3.2 Exakte probabilistische Inferenz

Es sind bereits mehrere Verfahren zur exakten PI über allgemeinen BNen entwickelt worden. Bei diesen Verfahren handelt es sich im wesentlichen um das Clustering [JLO90], [Jen96, S. 69 ff.], das Conditioning [DÓ96], [SAS94] und die Optimale Faktorisierung [LD94], [D'A95]. All diese Verfahren nutzen die speziellen strukturellen Eigenschaften BNe, um niedrigen Rechenaufwand zu gewährleisten. [MVD01, S. 99] Im folgenden soll nun das Clustering als ein Verfahren beschrieben werden, da es mit den anderen Verfahren äquivalent ist, das heißt, dass es ebenfalls exakte Ergebnisse liefert und den gleichen Rechenaufwand verursacht. [Tol98, S. 62] PI wird nicht direkt über einem BN vollführt, sondern über einer sogenannten Inferenzmaschine. Demzufolge gliedert sich auch das Clustering in zwei Schritte: den Bau der Inferenzmaschine über dem BN und der PI im engeren Sinne über der Inferenzmaschine. Abbildung 4.9 stellt den groben Ablauf des Clustering dar.

Das Clustering liegt dem in Kapitel 6 beschriebenen Softwaresystem und insbesondere seiner Komponente zur Anwendung BNe aus Abschnitt 6.4.3 zugrunde. Zudem ist das Clustering in der deutschen und englischen Literatur nach dem Wissen des Autors noch nicht detailliert und durchgängig beschrieben worden. Allerdings ist die detaillierte Kenntnis des Clustering für das Verständnis des weiteren Inhaltes der Arbeit nicht von entscheidender Bedeutung. Aus den genannten Gründen erfolgt die detaillierte Darstellung des Clustering als eine Form der exakten PI in Anhang A.3. Nachdem das Clustering als ein mögliches Verfahren zur exakten PI abstrakt dargestellt worden ist, veranschaulicht der folgende Abschnitt 4.3.3 das Clustering

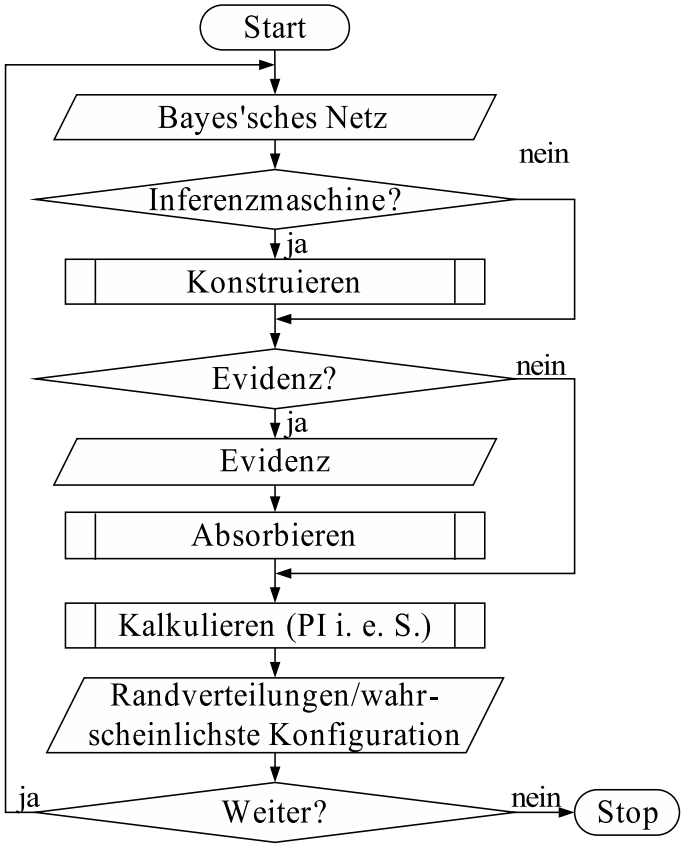

Abbildung 4.9: PAP für das Clustering (siehe Anhang A.3) als eine Form der PI: Als Eingabe dient ein BN. Über dem BN wird eine Inferenzmaschine konstruiert, sofern noch keine existiert. Sofern Evidenz vorliegt, wird sie eingegeben und von der Inferenzmaschine absorbiert. Die Inferenzmaschine vollzieht PI i. e. S., indem sie Nachrichten zwischen ihren Potenzialfunktionen austauscht, bis sich letztere im Gleichgewicht befinden. Randverteilungen oder wahrscheinlichste Konfigurationen werden aus der Inferenzmaschine herausintegriert und als Ergebnis der PI (i. e. S.) ausgegeben. Der Ablauf kann mit der Eingabe eines neuen oder veränderten BNes oder der Eingabe neuer oder veränderter Evidenz in die bestehende Inferenzmaschine von Neuem beginnen.

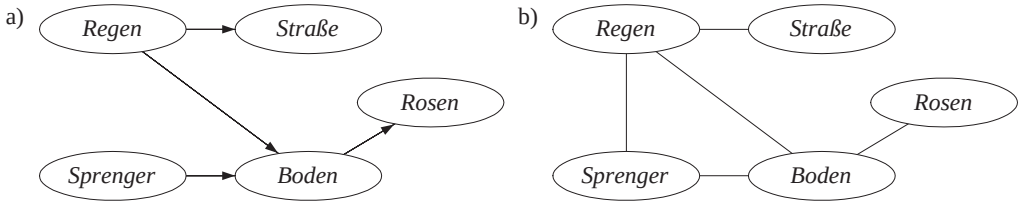

Abbildung 4.10: a) Graph D des BNes B aus dem Beispiel „Marie und ihre Rosen“ und b) der zu ihm moralische Graph U

anhand des Beispiels „Marie und ihre Rosen“.

4.3.3 Beispiel für exakte PI mittels Clustering

Vom Moralisieren bis zum Erstellen des Cliquenbaumes

Das BN $B = ((V, A), P)$, über welchem PI vollzogen werden soll, ist bereits aus dem Beispiel „Marie und ihre Rosen“ bekannt. Zu Beginn wird die Inferenzmaschine über B erstellt. Dazu wird ein zum Graphen $D = (V, A)$ des BNes moralischer Graph $U = (V, E)$ erstellt. Das sogenannte Moralisieren erfolgt mittels Algorithmus A.4, der intern auch Algorithmus A.3 zum Erstellen des unterliegenden Graphen aufruft. Faktisch wird den Kanten E lediglich die ungerichtete Kante re$\longleftrightarrow$sp hinzugefügt. Abbildung 4.10 zeigt a) noch einmal den Graphen D und b) den zu ihm moralischen Graphen U. Nachdem der moralische Graph U erstellt worden ist, wird er trianguliert. Da der Graph U im Beispiel bereits triangulär ist (siehe Definition A.33), entfällt der Schritt des Triangulierens. Nun werden im triangulären Graphen U die maximalen Cliquen $c \in C$ bestimmt. Folgende maximale Cliquen $c \in C$ sind mittels des Algorithmus A.8 bestimmt worden: die Clique $C_1 = \{re, sp, bo\}$, die Clique $C_2 = \{re, st\}$ und die Clique $C_3 = \{bo, ro\}$. Nachdem die Knoten bzw. Cliquen $c \in C$ des Cliquengraphen $T = (C, S)$ bestimmt worden sind, sind Kanten der Kantenmenge S des Cliquengraphen T so hinzuzufügen, dass T zum Cliquenbaum wird. Das geschieht mittels Algorithmus A.9, der die Kanten zusätzlich mit den Separatoren attributiert, welche die Schnittmengen der Variablen der den Kanten adjazenten Cliquen enthalten. Den erhaltenen Cliquenbaum T stellt Abbildung 4.11 dar. Die Separatoren werden durch Rechtecke dargestellt, welche besagte Schnittmengen enthalten. Im Anschluss ordnet Algorithmus A.10 den Cliquen $c \in C$ die Knoten $v \in V$ des Graphen D des BNes so zu, dass ein Knoten $v \in V$ einer der Cliquen $d \in C$ zugeordnet wird, welche die Familie Fa_v des Knotens v enthalten und zusätzlich minimale Mächtigkeit aufweisen. Somit werden der Clique C_1 die Knoten sp und bo zugeordnet, der Clique C_2 die Knoten re und st und der

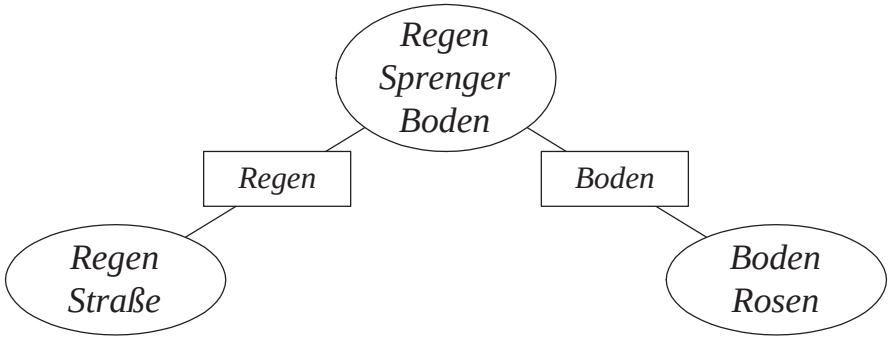

Abbildung 4.11: Der Cliquenbaum T zum Graphen D des BNes B aus dem Beispiel „Marie und ihre Rosen“

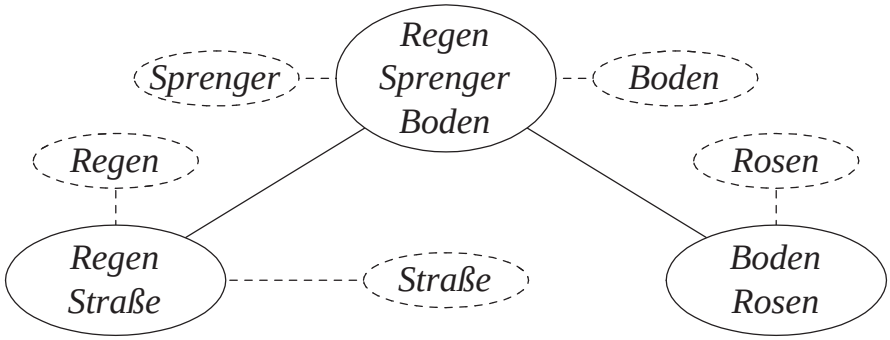

Abbildung 4.12: Die Variablen V des Graphen D des BNes B aus dem Beispiel „Marie und ihre Rosen“, den Cliquen C des Cliquenbaumes T zugeordnet

Clique C_3 der Knoten ro (siehe Abbildung 4.12).

Aufspannen und Füllen der Potenzialfunktionen

In einem letzten Schritt zum Erstellen der Inferenzmaschine M mittels Algorithmus A.12 werden die Potenzialfunktionen über den Cliquen C aufgespannt und gefüllt. Das geschieht mittels Algorithmus A.11: Zu Beginn werden sämtliche Funktionswerte der Potenzialfunktion einer Clique auf 1 gesetzt, und im Anschluss wird die Potenzialfunktion mit den Bewertungsfunktionen der Knoten multipliziert, welche der Clique zugeordnet worden sind. Die Tabellen 4.8, 4.9, 4.10, 4.11 und 4.12 stellen die zu vollführenden Multiplikationen dar.

Tabelle 4.8: Multiplikation der initialen Potenzialfunktion $f(C_1) = f(\{re, sp, bo\})$ mit der Bewertungsfunktion $p(\{sp\}|\emptyset)$

re	sp	bo	$f(C_1)$
ne_{re}	au_{sp}	tr_{bo}	0,95
ne_{re}	au_{sp}	fe_{bo}	0,95
ne_{re}	ei_{sp}	tr_{bo}	0,05
ne_{re}	ei_{sp}	fe_{bo}	0,05
ja_{re}	au_{sp}	tr_{bo}	0,95
ja_{re}	au_{sp}	fe_{bo}	0,95
ja_{re}	ei_{sp}	tr_{bo}	0,05
ja_{re}	ei_{sp}	fe_{bo}	0,05

$\leftarrow$

$f(C_1)$
1
1
1
1
1
1
1
1

$\cdot$

sp	$p(\{sp\}\|\emptyset)$
au_{sp}	0,95
ei_{sp}	0,05

Tabelle 4.9: Multiplikation der resultierenden Potenzialfunktion $f(C_1) = f(\{re, sp, bo\})$ aus Tabelle 4.8 mit der Bewertungsfunktion $p(\{bo\}|\{re, sp\})$

re	sp	bo	$f(C_1)$
ne_{re}	au_{sp}	tr_{bo}	0,8075
ne_{re}	au_{sp}	fe_{bo}	0,1425
ne_{re}	ei_{sp}	tr_{bo}	0,0025
ne_{re}	ei_{sp}	fe_{bo}	0,0475
ja_{re}	au_{sp}	tr_{bo}	0,095
ja_{re}	au_{sp}	fe_{bo}	0,855
ja_{re}	ei_{sp}	tr_{bo}	0,0005
ja_{re}	ei_{sp}	fe_{bo}	0,0495

$\leftarrow$

$f(C_1)$
0,95
0,95
0,05
0,05
0,95
0,95
0,05
0,05

$\cdot$

$p(\{bo\}\|\{re, sp\})$
0,85
0,15
0,05
0,95
0,1
0,9
0,01
0,99

Tabelle 4.10: Multiplikation der initialen Potenzialfunktion $f(C_2) = f(\{re, st\})$ mit der Bewertungsfunktion $p(\{re\}|\emptyset)$

re	st	$f(C_2)$
ne_{re}	tr_{st}	0,8
ne_{re}	na_{st}	0,8
ja_{re}	tr_{st}	0,2
ja_{re}	na_{st}	0,2

$\leftarrow$

$f(C_2)$
1
1
1
1

$\cdot$

re	$p(\{re\}\|\emptyset)$
ne_{re}	0,8
ja_{re}	0,2

Tabelle 4.11: Multiplikation der resultierenden Potenzialfunktion $f(C_2) = f(\{re, st\})$ aus Tabelle 4.10 mit der Bewertungsfunktion $p(\{st\}|\{re\})$

re	st	$f(C_2)$		$f(C_2)$		$p(\{st\}\|\{re\})$
ne_{re}	tr_{st}	0,72		0,8		0,9
ne_{re}	na_{st}	0,08	$\leftarrow$	0,8	$\cdot$	0,1
ja_{re}	tr_{st}	0,002		0,2		0,01
ja_{re}	na_{st}	0,198		0,2		0,99

Tabelle 4.12: Multiplikation der initialen Potenzialfunktion $f(C_3) = f(\{bo, ro\})$ mit der Bewertungsfunktion $p(\{ro\}|\{bo\})$

bo	ro	$f(C_3)$		$f(C_3)$		$p(\{ro\}\|\{bo\})$
tr_{bo}	sc_{ro}	0,75		1		0,75
tr_{bo}	sc_{ro}	0,25	$\leftarrow$	1	$\cdot$	0,25
fe_{bo}	gu_{ro}	0,05		1		0,05
fe_{bo}	gu_{ro}	0,95		1		0,95

Absorbieren der Evidenz

Die nunmehr gefüllten Potenzialfunktionen absorbieren Evidenz (siehe Algorithmus A.13). Nehmen wir an, Marie sehe aus dem Fenster auf die Straße und stelle fest, dass die Straße nass ist. Also sind die evidenten Variablen $E = \{st\}$, und die Evidenz ist $e = (na_{st})$. Lediglich die Potenzialfunktion der Clique C_2 ist von e betroffen, da nur C_2 die einzig evidente Variable st enthält. Demzufolge wirkt sich Algorithmus A.13 nur auf die Potenzialfunktion von $C_2 = \{re, st\}$ aus. Alle Funktionswerte von $f(C_2)$, welche nicht mit der Evidenz $e = (na_{st})$ übereinstimmen, werden auf 0 gesetzt (siehe Tabelle 4.13).

Tabelle 4.13: Absorption der Evidenz $e = (na_{st})$ durch die Potenzialfunktion $f(C_2) = f(\{re, st\})$

re	st	$f(C_2)$		$f(C_2)$	
ne_{re}	tr_{st}	0		0,72	
ne_{re}	na_{st}	0,08	$\leftarrow$	0,08	$\triangleleft(na_{st})$
ja_{re}	tr_{st}	0		0,002	
ja_{re}	na_{st}	0,198		0,198	

Tabelle 4.14: Herausintegrieren der Nachricht $msg[(C_2, C_1)]$ aus der Potenzialfunktion $f(C_2)$ aus Tabelle 4.13

re	$msg[(C_2, C_1)]$
ne_{re}	0,08
ja_{re}	0,198

$\leftarrow \sum_{C_2 \setminus \{re\}}$

re	st	$f(C_2)$
ne_{re}	tr_{st}	0
ne_{re}	na_{st}	0,08
ja_{re}	tr_{st}	0
ja_{re}	na_{st}	0,198

Austauschen der Nachrichten zwischen den Potenzialfunktionen

Nach der Absorption der Evidenz $e = (na_{st}) \in S_E$ durch die Potenzialfunktionen F werden Nachrichten zwischen ihnen bzw. zwischen den ihnen unterliegenden Cliquen ausgetauscht, bis sich die Potenzialfunktionen im Gleichgewicht befinden. Das Austauschen der Nachrichten erfolgt wie im Algorithmus A.16 beschrieben: Zu Beginn wird der rekursive Algorithmus A.14 zum Sammeln der Nachrichten aufgerufen. Sein erster Parameter ist die Clique, von deren Potenzialfunktion Evidenz gesammelt werden soll. Sein zweiter Parameter ist die Clique, deren Potenzialfunktion die Evidenz sammelt. Im Beispiel ruft Algorithmus A.16 den Algorithmus A.14 mit den Parametern $C_1 = \{re, sp, bo\}$ und NULL auf: Die Evidenz der Potenzialfunktion der Clique C_1 soll gesammelt werden, und es liegt beim ersten Aufruf des rekursiven Algorithmus' keine Clique bzw. Potenzialfunktion vor, welche diese Evidenz sammelt bzw. benötigt. Im Algorithmus A.14 in der ersten Rekursionsebene wird alsdann für jeden Nachbarn der Clique C_1 der Algorithmus A.14 noch einmal (rekursiv) aufgerufen. Zuerst wird er mit den Cliquen $C_2 = \{re, st\}$ und C_1 als Parametern aufgerufen: Die Potenzialfunktion der Clique C_1 sammelt die Evidenz der Potenzialfunktion der Clique C_2. Nunmehr in der zweiten Rekursionsebene, stellt Algorithmus A.14 fest, dass die Clique C_2 keine weiteren Nachbarn besitzt, für die er sich selbst aufrufen könnte, denn von Clique C_1 ist er ja aufgerufen worden. Er integriert somit die Nachricht $msg[(C_2, C_1)]$ aus der Clique C_2 heraus (siehe Tabelle 4.14). Ob es sich bei der Nachricht um einen Summenrand oder einen Maximumrand handelt, hängt von den globalen Einstellungen zur PI ab. Zurück in der ersten Rekursionsebene, multipliziert Algorithmus A.14 die von der Potenzialfunktion der Clique C_2 gesammelte Nachricht $msg[(C_2, C_1)]$ mit der Potenzialfunktion der Clique C_1 (siehe Tabelle 4.15). Alsdann ruft sich Algorithmus A.14 rekursiv mit den Cliquen $C_3 = \{bo, ro\}$ und C_1 als Parametern auf: Die Evidenz von der Clique C_3 wird gesammelt, und Clique C_1 sammelt die Evidenz. Abermals in der zweiten Rekursionsebene, stellt Algorithmus A.14 fest, dass die Clique C_3 keine weiteren Nachbarn besitzt, für die er sich selbst aufrufen

Tabelle 4.15: Multiplizieren der Potenzialfunktion $f(C_1)$ aus Tabelle 4.9 mit der resultierenden Nachricht $msg[(C_2, C_1)]$ aus Tabelle 4.14 von Clique C_2 bzw. von Potenzialfunktion $f(C_2)$

re	sp	bo	$f(C_1)$
ne_{re}	au_{sp}	tr_{bo}	0,0646
ne_{re}	au_{sp}	fe_{bo}	0,0114
ne_{re}	ei_{sp}	tr_{bo}	0,0002
ne_{re}	ei_{sp}	fe_{bo}	0,0038
ja_{re}	au_{sp}	tr_{bo}	0,01881
ja_{re}	au_{sp}	fe_{bo}	0,16929
ja_{re}	ei_{sp}	tr_{bo}	9,9E-05
ja_{re}	ei_{sp}	fe_{bo}	0,009801

$\leftarrow$

$f(C_1)$
0,8075
0,1425
0,0025
0,0475
0,095
0,855
0,0005
0,0495

$\cdot$

re	$msg[(C_2, C_1)]$
ne_{re}	0,08
ja_{re}	0,198

Tabelle 4.16: Herausintegrieren der Nachricht $msg[(C_3, C_1)]$ aus der Potenzialfunktion $f(C_3)$ aus Tabelle 4.12

bo	$msg[(C_3, C_1)]$
tr_{bo}	1
fe_{bo}	1

$\leftarrow \sum_{C_3 \setminus \{bo\}}$

bo	ro	$f(C_3)$
tr_{bo}	sc_{ro}	0,75
tr_{bo}	sc_{ro}	0,25
fe_{bo}	gu_{ro}	0,05
fe_{bo}	gu_{ro}	0,95

könnte. Er integriert somit die Nachricht $msg[(C_3, C_1)]$ aus der Potenzialfunktion der Clique C_3 heraus (siehe Tabelle 4.16). Dass die Funktionswerte von $msg[(C_3, C_1)]$ 1 betragen, ist nicht die Regel. Zurück auf erster Rekursionsebene, multipliziert Algorithmus A.14 die Potenzialfunktion der Clique C_1 mit der von Clique C_3 gesammelten Nachricht $msg[(C_3, C_1)]$ (siehe Tabelle 4.17), sämtliche Evidenz ist gesammelt worden, und Algorithmus A.14 terminiert. Der Algorithmus A.16 ruft nun den rekursiven Algorithmus A.15 zum Verteilen der Nachrichten auf. Bei den Parametern des Algorithmus' handelt es sich um die Clique, deren Potenzialfunktion Evidenz verteilt, und die Clique, an deren Potenzialfunktion Evidenz verteilt wird. Im Beispiel ruft Algorithmus A.16 den Algorithmus A.15 mit den Parametern NULL und $C_1 = \{re, sp, bo\}$ auf: Beim ersten Aufruf des rekursiven Algorithmus' liegt keine Clique bzw. Potenzialfunktion vor, welche Evidenz verteilen könnte, und die Potenzialfunktion der Clique C_1 soll ihre Evidenz an ihre Nachbarn verteilen. Im Algorithmus A.15 auf erster Rekursionsebene kann keine Nachricht mit der Potenzialfunktion der Clique C_1 multipliziert werden, da der erste Parameter NULL ist. Nun wird für die Nachbarclique $C_2 = \{re, st\}$ der

Tabelle 4.17: Multiplizieren der resultierenden Potenzialfunktion $f(C_1)$ aus Tabelle 4.15 mit der Nachricht $msg[(C_3, C_1)]$ von Clique C_3 bzw. von Potenzialfunktion $f(C_3)$ aus Tabelle 4.16

re	sp	bo	$f(C_1)$
ne_{re}	au_{sp}	tr_{bo}	0,0646
ne_{re}	au_{sp}	fe_{bo}	0,0114
ne_{re}	ei_{sp}	tr_{bo}	0,0002
ne_{re}	ei_{sp}	fe_{bo}	0,0038
ja_{re}	au_{sp}	tr_{bo}	0,01881
ja_{re}	au_{sp}	fe_{bo}	0,16929
ja_{re}	ei_{sp}	tr_{bo}	9,9E-05
ja_{re}	ei_{sp}	fe_{bo}	0,009801

$\leftarrow$

$f(C_1)$
0,0646
0,0114
0,0002
0,0038
0,01881
0,16929
9,9E-05
0,009801

$\cdot$

bo	$msg[(C_3, C_1)]$
tr_{bo}	1
fe_{bo}	1

Tabelle 4.18: Herausintegrieren der Nachricht $msg[(C_1, C_2)]$ aus der resultierenden Potenzialfunktion $f(C_1)$ aus Tabelle 4.17

re	$msg[(C_1, C_2)]$
ne_{re}	0,08
ja_{re}	0,198

$\leftarrow \sum_{C_1 \setminus \{re\}}$

re	sp	bo	$f(C_1)$
ne_{re}	au_{sp}	tr_{bo}	0,0646
ne_{re}	au_{sp}	fe_{bo}	0,0114
ne_{re}	ei_{sp}	tr_{bo}	0,0002
ne_{re}	ei_{sp}	fe_{bo}	0,0038
ja_{re}	au_{sp}	tr_{bo}	0,01881
ja_{re}	au_{sp}	fe_{bo}	0,16929
ja_{re}	ei_{sp}	tr_{bo}	9,9E-05
ja_{re}	ei_{sp}	fe_{bo}	0,009801

Clique C_1 die Nachricht $msg[(C_1, C_2)]$ herausintegriert (siehe Tabelle 4.18). Die Nachricht $msg[(C_1, C_2)]$ muss nun noch um die ihr entgegengesetzt gesendete Nachricht $msg[(C_2, C_1)]$ bereinigt und somit durch letztere dividiert werden, wie es Tabelle 4.19 zeigt.

Der Algorithmus A.15 wird nun rekursiv aufgerufen mit den Cliquen C_1 und C_2 als Parameter. In der zweiten Rekursionsebene angekommen, wird die Potenzialfunktion der Clique C_2 mit der Nachricht $msg[(C_1, C_2)]$ multipliziert (siehe Tabelle 4.20). Da Clique C_2 keinen weiteren Nachbarn aufweist außer der Clique C_1, welche die Nachricht $msg[(C_1, C_2]$ an die Clique C_2 verteilt hat, läuft der Algorithmus aus, und es wird zur ersten Rekursionsebene zurückgekehrt: Die Nachricht $msg[(C_1, C_3)]$ wird aus der Potenzialfunktion der Clique C_1 herausintegriert (siehe Tabelle 4.21) und durch die ihr entgegengesetzte Nachricht $msg[(C_3, C_1)]$ dividiert (siehe Tabelle 4.22). Alsdann

Tabelle 4.19: Dividieren der resultierenden Nachricht $msg[(C_1, C_2)]$ aus Tabelle 4.18 durch die ihr entgegengesetzte Nachricht $msg[(C_2, C_1)]$ aus Tabelle 4.14

re	$msg[(C_1, C_2)]$
ne_{re}	1
ja_{re}	1

$\leftarrow$

$msg[(C_1, C_2)]$
0,08
0,198

/

$msg[(C_2, C_1)]$
0,08
0,198

Tabelle 4.20: Multiplizieren der resultierenden Potenzialfunktion $f(C_2)$ aus Tabelle 4.11 mit der resultierenden Nachricht $msg[(C_1, C_2)]$ aus Tabelle 4.19 von der der Clique C_1 bzw. ihrer Potenzialfunktion

re	st	$f(C_2)$
ne_{re}	tr_{st}	0
ne_{re}	na_{st}	0,08
ja_{re}	tr_{st}	0
ja_{re}	na_{st}	0,198

$\leftarrow$

$f(C_2)$
0
0,08
0
0,198

$\cdot$

re	$msg[(C_1, C_2)]$
ne_{re}	1
ja_{re}	1

Tabelle 4.21: Herausintegrieren der Nachricht $msg[(C_1, C_3)]$ aus der resultierenden Potenzialfunktion $f(C_1)$ aus Tabelle 4.17

bo	$msg[(C_1, C_3)]$
tr_{bo}	0,083709
fe_{bo}	0,194291

$\leftarrow \sum_{C_1 \setminus \{bo\}}$

re	sp	bo	$f(C_1)$
ne_{re}	au_{sp}	tr_{bo}	0,0646
ne_{re}	au_{sp}	fe_{bo}	0,0114
ne_{re}	ei_{sp}	tr_{bo}	0,0002
ne_{re}	ei_{sp}	fe_{bo}	0,0038
ja_{re}	au_{sp}	tr_{bo}	0,01881
ja_{re}	au_{sp}	fe_{bo}	0,16929
ja_{re}	ei_{sp}	tr_{bo}	9,9E-05
ja_{re}	ei_{sp}	fe_{bo}	0,009801

Tabelle 4.22: Dividieren der resultierenden Nachricht $msg[(C_1, C_3)]$ aus Tabelle 4.21 durch die ihr entgegengesetzte Nachricht $msg[(C_3, C_1)]$ aus Tabelle 4.16

bo	$msg[(C_1, C_3)]$
tr_{bo}	0,083709
fe_{bo}	0,194291

$\leftarrow$

$msg[(C_1, C_3)]$
0,083709
0,194291

/

$msg[(C_3, C_1)]$
1
1

Tabelle 4.23: Multiplizieren der resultierenden Potenzialfunktion $f(C_3)$ aus Tabelle 4.12 mit der resultierenden Nachricht $msg[(C_1, C_2)]$ aus Tabelle 4.22 von der der Clique C_1 bzw. ihrer Potenzialfunktion

bo	ro	$f(C_3)$
tr_{bo}	sc_{ro}	0,06278175
tr_{bo}	gu_{ro}	0,02092725
fe_{bo}	sc_{ro}	0,00971455
fe_{bo}	gu_{ro}	0,18457645

$\leftarrow$

$f(C_3)$
0,75
0,25
0,05
0,95

$\cdot$

bo	$msg[(C_1, C_3)]$
tr_{bo}	0,083709
fe_{bo}	0,194291

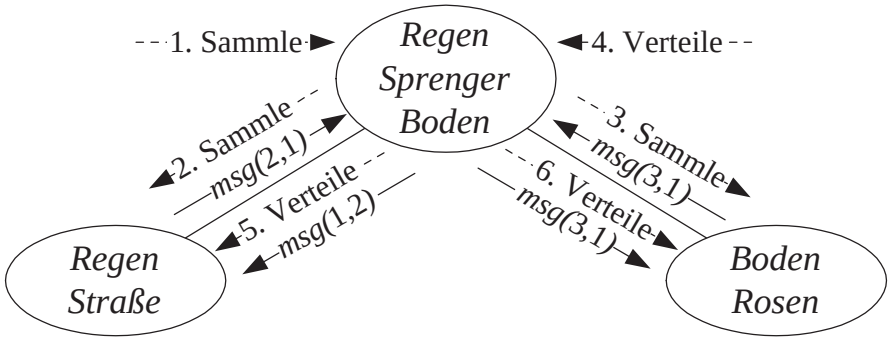

Abbildung 4.13: Austausch der Nachrichten zwischen den Cliquen C bzw. ihren Potenzialfunktionen F im Cliquenbaum T

ruft sich Algorithmus A.15 rekursiv mit den Parametern C_1 und C_3 auf. In der zweiten Rekursionsebene angelangt, wird die Nachricht $msg[(C_1, C_3)]$ mit der Potenzialfunktion der Clique C_3 multipliziert (siehe Tabelle 4.23). Die Clique C_3 hat keine weiteren Nachbarn außer Clique C_1, von der die Nachricht $msg[(C_1, C_3)]$ gekommen ist, so dass der Algorithmus A.15 zur ersten Rekursionsebene zurückkehrt. Auch Clique C_1 hat keine weiteren Nachbarn, so dass der Algorithmus A.15 auch die erste Rekursionsebene verlässt und terminiert. Nach dem Sammeln und dem Verteilen der Evidenz ist entlang jeder Kante des Cliquenbaumes T eine Nachricht ausgetauscht worden, und die Inferenzmaschine M befindet sich im Gleichgewicht. Abbildung 4.13 stellt zusammenfassend dar, in welcher Reihenfolge die Algorithmen A.14 und A.15 aufgerufen und Nachrichten zwischen den Cliquen bzw. ihren Potenzialfunktionen ausgetauscht worden sind. Wie die Potenzialfunktionen $f(C_1)$ bis $f(C_3)$ im Anschluss normalisiert werden zeigen die Tabellen 4.24 bis 4.26.

Tabelle 4.24: Normieren der resultierenden Potenzialfunktion $f(C_1)$ aus Tabelle 4.17

re	sp	bo	$f(C_1)$		$f(C_1)$	
ne_{re}	au_{sp}	tr_{bo}	0,232374101		0,0646	
ne_{re}	au_{sp}	fe_{bo}	0,041007194		0,0114	
ne_{re}	ei_{sp}	tr_{bo}	0,000719424		0,0002	
ne_{re}	ei_{sp}	fe_{bo}	0,013669065	$\leftarrow$	0,0038	$/ \sum \frac{\sum f(C_1)}{0{,}278}$
ja_{re}	au_{sp}	tr_{bo}	0,067661871		0,01881	
ja_{re}	au_{sp}	fe_{bo}	0,608956835		0,16929	
ja_{re}	ei_{sp}	tr_{bo}	0,000356115		9,9E-05	
ja_{re}	ei_{sp}	fe_{bo}	0,035255396		0,009801	

Tabelle 4.25: Normieren der resultierenden Potenzialfunktion $f(C_2)$ aus Tabelle 4.20

re	st	$f(C_2)$		$f(C_2)$	
ne_{re}	tr_{st}	0		0	
ne_{re}	na_{st}	0,287769784	$\leftarrow$	0,08	$/ \sum \frac{\sum f(C_2)}{0{,}278}$
ja_{re}	tr_{st}	0		0	
ja_{re}	na_{st}	0,712230216		0,198	

Tabelle 4.26: Normieren der resultierenden Potenzialfunktion $f(C_3)$ aus Tabelle 4.23

bo	ro	$f(C_3)$		$f(C_3)$	
tr_{bo}	sc_{ro}	0,225833633		0,06278175	
tr_{bo}	gu_{ro}	0,075277878	$\leftarrow$	0,02092725	$/ \sum \frac{\sum f(C_3)}{0{,}278}$
fe_{bo}	sc_{ro}	0,034944424		0,00971455	
fe_{bo}	gu_{ro}	0,663944065		0,18457645	

Tabelle 4.27: Herausintegrieren der Wahrscheinlichkeitsfunktion $\Pr(ro)$ aus der resultierenden Potenzialfunktion $f(C_3)$ aus Tabelle 4.26

ro	$\Pr(ro)$
sc_{ro}	0.260778058
gu_{ro}	0.739221942

$$\leftarrow \sum_{C_3 \setminus \{ro\}}$$

bo	ro	$f(C_3)$
tr_{bo}	sc_{ro}	0.22583363
tr_{bo}	gu_{ro}	0.07527788
fe_{bo}	sc_{ro}	0.03494442
fe_{bo}	gu_{ro}	0.66394406

Herausintegrieren der Randverteilungen

Nun verbleibt lediglich, Randwahrscheinlichkeitsfunktionen aus den Potenzialfunktionen, welche gemeinsame Wahrscheinlichkeitsfunktionen darstellen, herauszuintegrieren. Marie möchte natürlich wissen, wie es ihren Rosen geht bzw. ob es ihnen gutgeht. Tabelle 4.27 stellt das Herausintegrieren der Randwahrscheinlichkeitsfunktion $\Pr(ro)$ aus der Potenzialfunktion $f(C_3)$ dar. Das Herausintegrieren der Randwahrscheinlichkeitsfunktion für die anderen vier Variablen erfolgt analog und soll hier nicht explizit beschrieben werden.

Marie weiß nun also, dass die Wahrscheinlichkeit, dass es ihren Rosen gutgeht, rund 74 Prozent beträgt, wenn die Straße nass ist. Sie könnte sich mit diesem Wissen zufriedengeben. Als Alternative könnte sie vor die Haustüre treten, um zu spüren, ob es wahrhaftig regnet. Sie könnte in den Garten gehen, um zu fühlen, ob der Boden tatsächlich feucht ist. Oder sie könnte im Keller nachsehen, ob der Rasensprenger eingeschaltet ist und ihn eventuell ein- oder ausschalten. Mit der neu hinzugekommenen Evidenz könnte Marie eine weitere PI im engeren Sinne mit Algorithmus A.20 vollziehen und eine bessere Aussage über den Zustand ihrer Rosen treffen.

4.3.4 Approximative probabilistische Inferenz

Exakte PI im Kontext BNe verursacht hohen Rechenaufwand. Vor allem der Speicherbedarf für eine oder mehrere Cliquen in einem großen und stark vernetzten Graphen ist oftmals sehr hoch. Selbst bei einem optimal triangulierten Graphen kann er so hoch sein, dass er nicht von derzeit gebräuchlichen Rechnern gedeckt werden kann. [Jen01, S. 189], [CDLS99, S. 271] Da alle Verfahren exakter PI im Kontext BNe NP-vollständig sind und unter kombinatorischer Explosion leiden [CGH97, S. 393], werden auch zukünftige Rechnergenerationen den Bedarf an Speicher und Rechenleistung für PI im Kontext BNe nicht decken. Wenn man schnell oder sogar in Echtzeit probabilistisch inferieren möchte, wie es auch im Rahmen der PPS oft der Fall ist, spielt auch die nicht unerhebliche Rechenzeit für die exakte PI, die

im ungünstigsten Falle auch exponentiell wächst, eine entscheidende Rolle. Um den Forderungen nach geringerem Rechenaufwand nachzukommen, sind Verfahren der approximativen PI entwickelt worden, die bei eingeschränkter, aber vorab festgelegter Genauigkeit geringeren Rechenaufwand verursachen. Nach [CGH97, S. 393] lassen sich die Verfahren der approximativen PI in zwei Gruppen einteilen, die stochastische Simulation und die deterministische Suche. Anhang A.4 stellt die stochastische Simulation anhand des Beispiels „Marie und ihre Rosen“ vor und führt einen Algorithmus zur stochastischen Simulation auf.

Allerdings sind noch zwei weitere Verfahren zur approximativen PI bekannt. Das erste basiert auf einem Verfahren, das in [AOJJ89] und [AOJJ90] vorgestellt wird: Potenzialfunktionen, bei denen Funktionswerte gleich 0 auftreten, können z. T. stark komprimiert und auch komprimiert verarbeitet werden, woraus geringerer Speicherbedarf und geringere Rechenzeit resultieren. Schneidet man nun die Nachkommastellen sehr kleiner Funktionswerte ab bzw. rundet diese Funktionswerte auf 0, wie es in [JA90] vorgeschlagen wird, nehmen Speicherbedarf und Rechenzeit weiter ab, aber die PI wird approximativ.

Das zweite approximative Verfahren stammt aus [Kjæ94] und entfernt unbedeutende Kanten[16] aus dem moralischen Graphen, wodurch beim Triangulieren weniger Kanten eingefügt werden müssen. Es entstehen dann zwar mehr Cliquen und Potenzialfunktionen, aber der Speicherbedarf letzterer ist insgesamt geringer. Natürlich verursacht das Entfernen von Kanten ein gewissen Fehler bei der Inferenz.

Da für die im Rahmen der Arbeit durchgeführten Experimente sowohl der Speicher als auch die Rechenzeit ausgereicht haben, ist es nicht nötig gewesen, Verfahren der approximativen PI zu implementieren und anzuwenden.

[16] Um zu bestimmen, ob Kanten unbedeutend sind, werden zwei Inferenzmaschinen erstellt: eine über dem vollständigen moralischen Graphen und eine über dem moralischen Graphen *ohne* besagte Kanten. Liegt der Unterschied zwischen den Potenzialfunktionen beider Inferenzmaschinen unterhalb eines einzustellenden Parameters, so sind die entfernten Kanten unbedeutend. [Kjæ94]

4.4 Datenvor- und -nachverarbeitung bei Wissensakquisition und -anwendung im Kontext BNe

4.4.1 Bestimmen relevanter Variablen

In vielen Fällen in der Praxis ist nicht bekannt, welche Variablen relevant für das Problem sind, welches mittels eines BNes gelöst werden soll. Diese Variablen sind ergo zu ermitteln, um ein BN zu erstellen, das korrekt dimensioniert ist und effizient arbeitet. Steht bereits eine große Anzahl potenziell relevanter Variablen V_{pot} zur Verfügung, sind die tatsächlich relevanten Variablen $V_{rel} \subseteq V_{pot}$ zu identifizieren. Die Identifikation von V_{rel} wird auch als Merkmalsextraktion bezeichnet. [KW00, S. 373 ff.]

Im Rahmen der (statistischen) Datenanalyse [Zim95], [Sta00], des Data Mining (DM) [Kan03] und des Knowledge Discovery in Databases (KDD) [ES00] werden zur Vorauswahl relevanter Variablen verschiedene Verfahren verwendet. Zu den einfacheren unter ihnen zählen die graphische Darstellung der Daten in Form niedrigdimensionaler Punktwolken (Streudiagramme) und ihre visuelle Analyse [Sac99, S. 490 f.], [Zim95, S. 14 f.] sowie die (partielle und multiple) Korrelationsanalyse [Bös99, S. 143 ff.], [Sac99, S. 570 ff.]. Bei ihr wird z. B. auf sogenannte $n \times n$-Korrelationsmatrizen zurückgegriffen [Zim95, S. 132], deren Elemente jeweils mit dem Korrelationskoeffizienten zwischen zwei Variablen besetzt sind.

Beim Erstellen BNe ist es aber nicht nötig, externe Verfahren anzuwenden, um die relevanten Variablen V_{rel} zu bestimmen. BNe stellen per se Eigenschaften und Fähigkeiten bereit, welche es erlauben, die relevanten Variablen V_{rel} in den potenziell relevanten V_{pot} zu identifizieren bzw. die irrelevanten Variablen $V_{pot} \setminus V_{rel}$ aus V_{pot} zu entfernen.

Laut Barton [Bar02, S. 47] gibt es vier grundlegende Arten relevanter Variablen: Stellgrößen, Regelgrößen, Störgrößen und intermediäre[17] Größen, die zwischen den Variablen der ersten drei Arten vermitteln.[18] Dem sind – zumindest im Kontext BNe – noch zwei weitere Arten von Variablen hinzuzufügen: Die Zustandsgrößen, welche den Zustand eines Systems zu einem

[17] Das Wort intermediär bedeutet, sich zwischen zwei Dingen befindend und zwischen ihnen vermittelnd, entstammt dem Lateinischen und wird in Fachsprachen verwandt. [Dud89, S. 224]

[18] Barton [Bar99b, S. 32] bezeichnet Stell- und Regelgrößen als unabhängige und abhängige Variablen. Da die Begriffe Stell- und Regelgröße aber gängig und aussagekräftiger sind, werden sie hier verwandt. In der Betriebswirtschaft, werden Stell- und Regelgrößen auch als Parameter und Ziele bezeichnet. Die Volkswirtschaft bezeichnet Störgrößen als externe Effekte.

bestimmten bzw. zum derzeitigen Zeitpunkt charakterisieren und die Größen, die von den Regelgrößen beeinflusst werden.

Unterstellt, es liegen die Variablen V_{pot} vor, und man könne die Variablen aller vier Arten eindeutig identifizieren, wird zu Beginn das BN erstellt, das noch sämtliche Variablen V_{pot} enthält. Um die Variablen $V_{rel} \subseteq V_{pot}$ zu identifizieren, werden die Variablen im BN instanziiert bzw. mit hypothetischer Evidenz versehen, wie sie auch beim späteren Einsatz des BNes mit Evidenz versehen werden würden. Im Kontext einer Regelung mittels BNe wären das die derzeitigen Zustandsgrößen und die zukünftigen Regelgrößen. Letztere werden auf die erwünschten Führungswerte gesetzt. (Von den Zustands- und Regelgrößen wird dann interkausal auf die Werte oder Verteilungen der Stellgrößen geschlossen. Bei einer Prognose oder einer Diagnose würden lediglich die derzeitigen Zustandsgrößen gesetzt, um von ihnen auf zukünftige oder vergangene Größen zu schließen. Nachfolgend werde aber weiterhin die Regelung betrachtet.) Sind die Variablen gesetzt worden, werden mittels des Kriteriums der d-Separation die relevanten Variablen V_{rel} bestimmt. Zu den Variablen V_{rel} zählen

1. die Stellgrößen, die von mindestens einer Regelgröße nicht d-separiert sind,

2. die Regelgrößen, die von mindestens einer Stellgröße nicht d-separiert sind, und

3. die Zustands- und Störgrößen, welche von den bereits identifizierten Stell- und Regelgrößen nicht d-separiert sind.

Die in der Menge $V_{pot} \setminus V_{rel}$ enthaltenen Variablen werden – zuzüglich der zu ihnen inzidenten Kanten – aus dem BN entfernt.

Sollte sich das resultierende BN als immer noch zu umfangreich herausstellen, um schnelle PI durchzuführen, können beim erstmaligen Erstellen desselben noch Parameter vorgegeben werden, welche die Anzahl der eingefügten Kanten nach oben beschränken. Bei diesen Parametern handelt es sich im wesentlichen um die maximale Anzahl der für eine Variable im Graphen zugelassenen Eltern $|Pa|_{max}$. Die Parameter werden im Abschnitt 4.2.4 näher behandelt. Beschränkt man die Anzahl der Eltern im BN nach oben, sinkt die Anzahl der relevanten Variablen $|V_{rel}|$ tendenziell, irrelevante Variablen werden nach dem obigen Schema entfernt, und die PI verursacht im Allgemeinen weniger Rechenaufwand. Aber es besteht auch die Gefahr, dass ein zu stark beschränktes BN die Realität nicht mehr korrekt abbildet und die PI somit systematische Fehler begeht. Obwohl nicht unumstritten [Hec99, S. 303], bietet sich in einem solchen Fall das Testen des BNes anhand von

Testdaten an. Im Rahmen der vorliegenden Arbeit sind solche Tests erfolgreich durchgeführt worden.

Bis jetzt ist davon ausgegangen worden, dass $V_{rel} \subseteq V_{pot}$. Wenn aber $V_{rel} \setminus V_{pot} \neq \emptyset$ und V_{rel} unbekannt ist, dann kann man die Teilmenge $V_{rel} \cap V_{pot}$ der relevanten Variablen wie oben dargestellt ermitteln, aber nicht die Teilmenge $V_{rel} \setminus V_{pot}$ der relevanten Variablen. Um die Teilmenge $V_{rel} \setminus V_{pot}$ ebenfalls zu ermitteln, bietet sich im Kontext BNe folgende Vorgehensweise an: Kann eine Kante zwischen zwei Variablen $(u, v) \in V_{rel} \cap V_{pot}$ nicht eindeutig orientiert werden, so deutet das darauf hin, dass eine oder mehrere versteckte Variablen[19] $w \in V_{rel} \setminus V_{pot}$ vorliegen, welche die (gemeinsame) Ursache für u und v oder auch ihre (gemeinsame) Wirkung verkörpern, und dass zwischen den Variablen u und v kein Kausalzusammenhang besteht, sondern lediglich eine Korrelation vorliegt. Es sollte ergo nicht eine Kante zwischen u und v eingefügt werden, sondern z. B. die gerichteten Kanten (w, u) und (w, v). Ob eine Kante nicht eindeutig zu orientieren ist, findet der Experte oder Wissensingenieur heraus, der das BN manuell erstellt. Abschnitt 4.2.2 zeigt die Vorgehensweise auf. Aber es lässt sich ebenfalls automatisch aufgrund von Daten bestimmen, ob sich eine Kante nicht eindeutig orientieren lässt: Bewertet anhand einer Datenbasis das Gütemaß zwei BNe gleich, die sich lediglich in der Orientierung einer ihrer Kanten unterscheiden, kann die Kante ebenfalls nicht eindeutig orientiert werden, und es wird auf versteckte Variablen $w \in w \in V_{rel} \setminus V_{pot}$ hingewiesen. Es sei explizit vermerkt, dass auf die beschriebene Art und Weise lediglich indiziert wird, dass versteckte Variablen vorliegen; um welche Variablen es sich im Detail handelt, kann nicht automatisch, sondern nur mittels Expertenwissens über das Problemfeld bestimmt werden.

4.4.2 Festlegen des Abtastintervalls zur Erhebung multivariater Zeitreihen

Um mittels eines DBNes die Produktion zu modellieren, benötigt man multivariate Zeitreihen über den Verlauf des Produktionsprozesses. Diese Zeitreihen werden gewonnen, indem man äquifrequent multivariate Datensätze vom Produktionsprozess abtastet. Ein Problem, welches sich beim Abtasten stellt, ist das Festlegen des Abtastintervalls. Das Problem tritt nicht auf, wenn

1. technisch bedingt nur in einem bestimmten Intervall abgetastet werden kann,

[19] Versteckte Variablen ist eine Übersetzung des Begriffes Hidden Variables, der in der englischsprachigen Fachliteratur verwandt wird (siehe z. B. [Ram98] und [SGS93, S. 38 f.]).

2. der Prozess sich bereits in feste Intervalle gliedert oder

3. eine multivariate Zeitreihe bereits unabänderlich vorliegt und so verwendet werden soll.

Im Rahmen der PPS treten diese Fälle z. B. dann auf, wenn

1. keine gute und durchgängige Betriebsdatenerfassung (BDE) existiert bzw. vorgenommen wird oder nur an bestimmten spatio-temporalen Stellen der Prozess abgetastet werden kann,

2. eine getaktete Fließfertigung vorliegt und der Takt das Abtastintervall vorgibt oder

3. eine starke Ausrichtung der Produktion an Schichten, Tagen, Wochen oder anderen zeitlichen Perioden erfolgt, welche als Abtastintervall verwendet werden sollen.

Ist all das nicht der Fall, so muss das Abtastintervall so festgelegt werden, dass multivariate Zeitreihen abgetastet werden, aus denen ein DBN erstellt werden kann, das einerseits die Produktion für die zu lösende Aufgabe ausreichend genau abbildet, das aber andererseits nicht so komplex wird, dass die PI über ihm bezüglich des Rechenaufwandes untragbar wird, wenn es einen längeren Zeitraum abbilden soll. In dem Dilemma zwischen Abbildungsgenauigkeit und Rechenaufwand ist das Bestimmen des Abtastintervalls angesiedelt. [Gün97, S. 258]

Vier prinzipielle Vorgehensweisen sind denkbar, um das Dilemma zu lösen:

1. die graphische Darstellung der multivariaten Zeitreihen und das visuelle bzw. manuelle Festlegen des Abtastintervalles,

2. das Testen mehrerer BNe, die aus multivariaten Zeitreihen erstellt worden sind, welche wiederum durch Abtasten der Produktion in verschiedenen Frequenzen entstanden sind, und die Auswahl der Frequenz und somit des Abtastintervalles, die bzw. das zum besten BN führt – bezüglich der Genauigkeit der Abbildung der Produktion und bezüglich des beim Anwenden des BNes verursachten Rechenaufwandes,

3. das hochfrequente Abtasten des Produktionsprozesses, das Bestimmen der Autokorrelationen [Wer89, S. 81 ff.] bei unterschiedlichem zeitlichen Versatz und die Auswahl desjenigen Versatzes und somit desjenigen Abtastintervalles, der bzw. das zu maximaler Autokorrelation führt, und

4. das hochfrequente Abtasten des Produktionsprozesses, die Fourier-Transformation [Gün97, S. 18 ff.] der multivariaten Zeitreihe, das Bestimmen der maximalen Frequenz, der Abtastfrequenz nach dem Abtasttheorem [Wer89, S. 77] und somit des Abtastintervalles.

Punkt 1 stellt eine gängige Vorgehensweise zum Schätzen des Abtastintervalls dar. Allerdings ist die Schätzung stark subjektiv geprägt. Mehrere visuelle Schätzungen, Tests derselben und Auswahl der besten Schätzung – wie in Punkt 2 dargelegt – zeitigen jedoch gute Resultate. Problematisch ist auch, dass Punkt 1 manuell durchgeführt werden muss. Vollautomatisches Erstellen mehrerer DBNe in Folge ist demzufolge nicht möglich.

Die Vorgehensweise in Punkt 2 – das systematische Testen verschiedener Abtastintervalle – ist universell. Sie weist allerdings zwei Nachteile auf: Erstens müssen erfolgversprechende Abtastintervalle vorgegeben werden, und zweitens verursacht das Erstellen und vor allem das Testen mehrerer DBNe hohen Rechenaufwand. Nichtsdestotrotz wird die Vorgehensweise in Punkt 2 – unter Umständen in Verbindung mit Punkt 1 – favorisiert. Auch in der Literatur wird empfohlen, die adäquate Wahl des Abtastintervalles durch Tests sicherzustellen. [Gün97, S. 259]

Punkt 3 basiert auf der Autokorrelationsanalyse [Sta00, S. 323]. Es sei m der Stichprobenumfang bzw. die Anzahl der Datensätze in der Zeitreihe, und k sei der betrachtete zeitliche Verzug. Es bezeichne s_t die t-te Repräsentation der Variable $v \in V$ in der Zeitreihe. Die Formel zum Berechnen der Autokorrelation $r_v\langle k\rangle$ in einer univariaten Zeitreihe über v lautet in Anlehnung an [Sta00, S. 52]

$$r_v\langle k\rangle = \frac{\sum_{t=1}^{m-k}(s_t - \bar{s})(s_{t+k} - \bar{s})}{\sum_{t=1}^{m}(s_t - \bar{s})^2} \tag{4.20}$$

mit $\bar{s} = \frac{1}{m}\sum_{j=1}^{m} s_j$. Für eine multivariate Zeitreihe über $n = |V|$ Variablen könnte man nun eine durchschnittliche Autokorrelation wie folgt definieren:

$$\bar{r}\langle k\rangle = \frac{1}{n}\sum_{v\in V} r_v\langle k\rangle, \tag{4.21}$$

wobei $r_v\langle k\rangle$ der Autokorrelation nach Formel 4.20 entspricht.

Die n Variablen V des Produktionsprozesses seien nun mit einer sehr hohen Frequenz f_{high} m mal abgetastet worden, woraus eine m-variate Zeitreihe über n Zeitpunkten resultiert. Nun ist ein Zeitverzug

$$k_{best} = \underset{k\in\{1,\ldots,m-1\}}{\operatorname{argmax}} \bar{r}\langle k\rangle \tag{4.22}$$

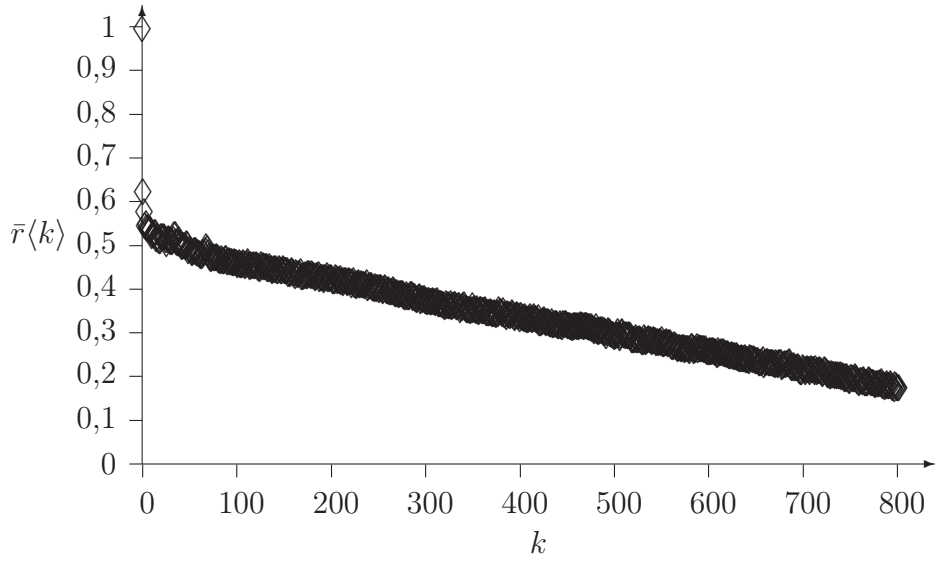

Abbildung 4.14: Verlauf der durchschnittlichen Autokorrelation $\bar{r}\langle k\rangle$ über einer 23-variaten Zeitreihe mit 20.000 Datensätzen in Abhängigkeit vom Zeitverzug k (eigene Messung)

zu bestimmen, für den die mittlere Autokorrelation $\bar{r}\langle k_{best}\rangle$ maximal wird, das heißt $\bar{r}\langle k_{best}\rangle = \max_{\{k|k\geq 1\}} \bar{r}\langle k\rangle$. Aus k_{best} kann dann in Verbindung mit f_{high} das Abtastintervall Δt wie folgt bestimmt werden:

$$\Delta t = \frac{k_{best}}{f_{high}} \tag{4.23}$$

Das Problem besteht nun lediglich darin, dass die durchschnittliche Autokorrelation $\bar{r}\langle k\rangle$ mit steigendem k tendenziell sinkt, wie es Abbildung 4.14 an einem Beispiel zeigt. Die maximale durchschnittliche Autokorrelation $\bar{r}\langle k\rangle$ wird ergo bei $k_{best} = 1$ erzielt, was zu einem untragbar hohen Rechenaufwand bei der PI über einem DBN führt, das über mehrere Intervalle entrollt worden ist. Zudem misst die Autokorrelation lediglich lineare Zusammenhänge und ist bei nicht-periodischen Zeitreihen nur bedingt aussagekräftig. Aus den genannten Gründen ist die Vorgehensweise aus Punkt 3 nicht anwendbar, um das Abtastintervall festzulegen.[20] Ein Schluss lässt sich aus der Analyse der Autokorrelation jedoch ziehen: Vermutlich bildet ein DBN den Produktionsprozess umso genauer ab, je kleiner das Abtastintervall gewählt worden ist.

[20]Man könnte zusätzlich die Kreuzkorrelation [Wer89, S. 83 f.] heranziehen, um einen günstigen Verzug k zwischen zwischen zwei und mehreren Variablen zu ermitteln. Allerdings sind ähnliche Ergebnisse wie bei der Autokorrelation zu erwarten.

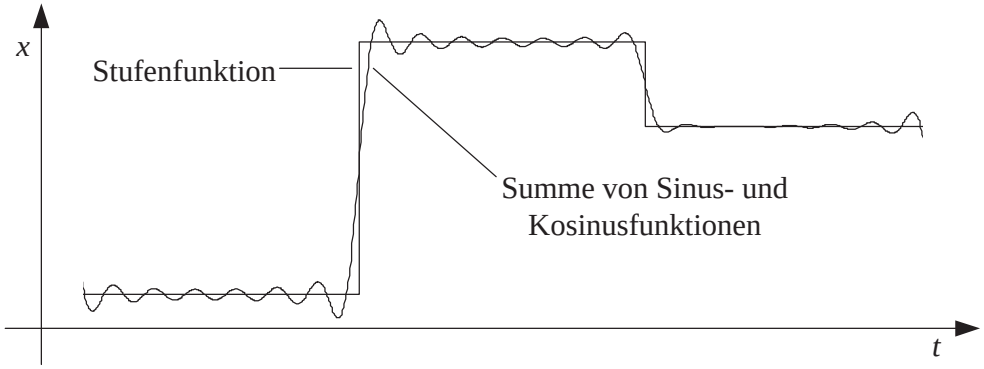

Abbildung 4.15: Eine Stufenfunktion aus einem Produktionsprozess und ihre Approximation durch eine Summe von Sinus- und Kosinusfunktionen, welche mittels Fourier-Transformation ermittelt worden sind

Versucht man – wie in Punkt 4 beschrieben – den Produktionsprozesses hochfrequent abzutasten und mittels einer Fourier-Transformation [Wer89, S. 77 ff.] Sinus- und Kosinusfunktionen zu bestimmen, deren Summe die multivariate Zeitreihe bzw. die Stufenfunktionen approximiert, erhält man eine sehr geringe minimale Wellenlänge bzw. eine sehr hohe maximale Frequenz f_{max}. Eine solche Approximation stellt Abbildung 4.15 dar. Nach dem Abtasttheorem [Lud95, S. 285] müsste man den Produktionsprozess nun mit einer Frequenz $f > 2f_{max}$ abtasten. Das heißt, dass das Abtastintervall sehr gering wäre. Da die Weite einer Stufe der Stufenfunktion z. B. der Länge eines einzelnen Arbeitsganges entspricht und während eines solchen mehrmals abgetastet werden müsste, wäre ein DBN über sehr viele Intervalle zu entrollen, um mit ihm die Produktion über einen längeren Zeitraum zu modellieren. Die PI über einem solchen DBN wäre vom Rechenaufwand her wiederum untragbar. Die Vorgehensweise aus Punkt 4 ist demzufolge ebenfalls nicht dazu geeignet, ein adäquates Abtastintervall zu bestimmen.[21] Man sollte das Abtastintervall aber möglichst klein wählen, um den Produktionsprozess gut mittels eines DBNes zu modellieren. Eine Grenze (Nebenbedingung) setzt dabei die PI bzw. der für sie erforderliche Rechenaufwand.

[21] Genügte die Wahl des Abtastintervalles dem Abtasttheorem, könnte man aufgrund der resultierenden multivariaten Zeitreihe den Produktionsprozess detailliert abbilden. Quasi könnte jeder Zustandsübergang identifiziert und reproduziert werden. Ein DBN soll den Produktionsprozess jedoch nicht en detail, sondern eher auf einer makroskopischen Ebene abbilden. Demzufolge muss die Wahl des Abtastintervalles nicht dem Abtasttheorem genügen.

Zusammenfassend muss festgestellt werden, dass es keinen analytisch fundierten Weg gibt, im Kontext DBNe ein optimales Abtastintervall zu bestimmen. Aber mittels systematischen Testens mehrerer Alternativen eventuell in Verbindung mit einer visuellen Analyse bzw. manuellen Vorauswahl kann ein akzeptables Abtastintervall festgelegt werden.

4.4.3 Diskretisieren reellwertiger Variablen

Da BNe nur diskrete Variablen behandeln, ist es vonnöten, reellwertige Variablen zu diskretisieren. In [Sac99, S. 107] wird vorgeschlagen, in sieben bis 20 Intervalle gleicher Breite zu diskretisieren – abhängig vom Stichprobenumfang. [BSMM99, S. 765] schlägt zehn bis 20 Intervalle vor. Eine Faustregel, die auf [Stu26] zurückgeht, lautet

$$k = \lfloor 1 + 3{,}322 \lg m \rfloor, \tag{4.24}$$

wobei k die Anzahl der Intervalle sei und m der Stichprobenumfang.[22] Diese Vorschläge und Formel 4.24 sind allerdings für das Erstellen von Histogrammen und für eindimensionale Stichproben gedacht und können nur bedingt zur Diskretisierung reellwertiger Variablen im Kontext BN angewandt werden: Diskretisiert man Variablen einer Familie Fa_v der Variable v jeweils in relativ viele Intervalle, so kann es erstens vorkommen, dass in der multivariaten Stichprobe keine Repräsentation oder nur sehr wenige Repräsentationen für eine Kombination von Intervallen der Variablen Fa_v vorliegen und die bedingten Wahrscheinlichkeitsfunktionen gleichverteilt sind oder zumindest viele gleiche Werte enthalten und somit nicht aussagekräftig sind. Zweitens führt es zu umfangreichen Bewertungsfunktionen im BN, im Rahmen der PI zu noch umfangreicheren Potenzialfunktionen und somit zu einer sehr aufwendigen PI. Erfahrungen, die im Rahmen der Arbeit gesammelt worden sind, legen nahe, im Kontext BNe so zu diskretisieren, dass pro Variable zwei bis sieben Intervalle der gleichen Breite b entstehen. Fällt eine Repräsentation einer Stichprobe auf eine Intervallgrenze, so muss eindeutig festgelegt werden, zu welchem Intervall sie gehört. Wenn nicht explizit anders angegeben, werden die Intervalle deshalb als rechtsoffen angenommen.

Nimmt ein Experte die Diskretisierung vor, so kann er ein Randintervall oder beide Randintervalle verbreitern.

[22] In [Sac99, S. 134] werden weitere Vorschläge unterbreitet: $k = [6 + m/50]$, $k = [5 \lg m]$ oder $b = [\sqrt{x_{max} - x_{min}}]$ für die Intervallbreite, wobei x_{max} und x_{min} den größten bzw. geringsten Wert in der Stichprobe bezeichnen.

Der Zusammenhang zwischen Intervallbreite und -anzahl lautet $b = (x_{max} - x_{min})/k$.

Sollten theoretisch reellwertige Variablen in der Praxis in nur wenigen, bekannten und immer wiederkehrenden Ausprägungen auftreten, ist eine Diskretisierung nicht unbedingt erforderlich. Sind Werkstücke z. B. nur entweder 317 mm, 473 mm oder 529 mm lang, so kann die eigentlich reellwertige Variable als ordinale Variable aufgefasst und verarbeitet werden.

Wenn ganzzahlige Variablen auftreten, könnte theoretisch jede auftretende ganze Zahl im BN als Zustand verwendet werden. In der Praxis würde das bei einem großen Wertebereich wiederum zu schlecht besetzten und umfangreichen Bewertungsfunktionen bzw. zu noch umfangreicheren Potenzialfunktionen führen, die eine PI bei akzeptablem Rechenaufwand verhindern. Demzufolge werden auch ganzzahlige Variablen wie reellwertige Variablen diskretisiert.

Liegen ordinale Variablen vor, so sind im Kontext BNe ihre Zustände so zu gruppieren, dass wiederum zwei bis sieben Gruppen aufeinanderfolgender Zustände entstehen. Wird automatisch gruppiert, so sollten die Größen zweier Gruppen um nicht mehr als eins differieren. Aufgrund von Expertenwissen kann natürlich wieder von dieser Regel abweichend gruppiert werden.

Sollten nominale Variablen sehr viele Zustände aufweisen, sollten auch sie gruppiert werden, um niedrigen Rechenaufwand bei der PI zu garantieren. Sinnvolles automatisches Gruppieren gestaltet sich bei nominalen Variablen allerdings schwieriger und kann mittels Verfahren der multivariaten Statistik, wie z. B. der Clusteranalyse [HE99, S. 443 ff.], vorgenommen werden.

Keines der beschriebenen Diskretisierungsverfahren kann einem anderen grundsätzlich vorgezogen werden. Es muss das Verfahren ausgewählt und angewandt werden, das zum Skalenniveau der zu diskretisierenden Variable passt. Liegt Expertenwissen über die Variable vor, bietet es sich zudem an, die Variable aufgrund dieses Wissens zu diskretisieren. Lediglich die Empfehlung, den Wertebereich einer Variable in zwei bis maximal sieben Intervalle bzw. Mengen möglichst gleicher Mächtigkeit zu unterteilen, die den Wertebereich vollständig abdecken, aber sich gegenseitig ausschließen, kann verallgemeinert werden.

4.5 Eigenschaften und Fähigkeiten BNe

Die vorangegangenen Abschnitte dieses Kapitels sind bereits kurz auf markante Eigenschaften und Fähigkeiten BNe eingegangen. Der hiesige Abschnitt stellt die Eigenschaften und Fähigkeiten BNe noch einmal zusammen.

4.5.1 Identifizieren und Quantifizieren kausaler stochastischer Zusammenhänge

Eine Kombination aus einem Suchverfahren und einem Gütemaß ermöglicht es, die Struktur bzw. den Graphen es BNes automatisch aus einer Datenbasis zu erstellen. [KN04, S. 197 ff.] Es existieren darüber hinaus alternative Ansätze, z. B. [Nea04, S. 533 ff.], [KN04, S. 147 ff.] und [SGS93], die auf Constraints basieren und äquivalente Ergebnisse zeitigen. Die direkten kausalen Zusammenhänge, welche zwischen den betrachteten Variablen in der Datenbasis vorliegen, können demzufolge als qualifiziert bzw. identifiziert betrachtet werden. Es verbleibt, die kausalen stochastischen Zusammenhänge automatisch zu quantifizieren. Zu diesem Zweck existieren Schätzer (siehe z. B. [KN04, S. 179 ff.]). Zwei von ihnen stellt Abschnitt 4.2.5 vor. Die Schätzer ermitteln automatisch die bedingten Wahrscheinlichkeiten der Bewertungsfunktionen des BNes aus der Datenbasis. Diese Schätzer quantifizieren allerdings nicht jede stochastische kausale Beziehung (jede Kante des BNes) einzeln, sondern sämtliche kausale Beziehungen zwischen einer Variable und ihren Eltern. Noisy-Operatoren (siehe Abschnitt 4.2.2) erlauben allerdings, in einem Zwischenschritt die kausalen Beziehungen erst einzeln automatisch zu quantifizieren und danach ebenfalls automatisch zu Bewertungsfunktionen zu aggregieren. BNe sind demzufolge in der Lage, kausale stochastische Zusammenhänge sowohl automatisch zu identifizieren als auch automatisch zu quantifizieren und somit sich selbst zu erstellen.

Nicht vergessen werden soll an dieser Stelle, dass BNe in Form DBNe in der Lage sind, automatisch stochastische Prozesse zu modellieren. [FMR98, S. 139] Erstellte BNe oder DBNe dienen nun als Erklärungsmodelle, zur Prognose, zur Diagnose und zur Entscheidungsunterstützung.

4.5.2 Prognose und Diagnose mittels BNe

Prognose und Diagnose können sowohl kausal als auch temporal interpretiert werden. Oft überwiegt bei der Prognose der temporale Aspekt und bei der Diagnose der kausale. Die Übergänge sind jedoch fließend. In der vorliegenden Arbeit soll unter Prognose das Schließen von Ursachen auf Wirkungen entlang der kausalen Kette verstanden werden.[23] Da die Wirkungen ihren Ursachen zeitlich immer folgen, handelt es sich bei der Prognose ebenso um ein Schließen von Zuständen zu einem Zeitpunkt t_{cause} auf Zustände zu einem Zeitpunkt t_{effect} mit $t_{cause} \leq t_{effect}$. Prognose erfolgt nicht zwangsläufig vom gegenwärtigen Zeitpunkt aus in die Zukunft. Die Zeitpunkte t_{cause} oder

[23] Das Schließen von Ursachen auf Wirkungen wird auch als kausales Schließen bezeichnet. [KN04, S. 34 f.]

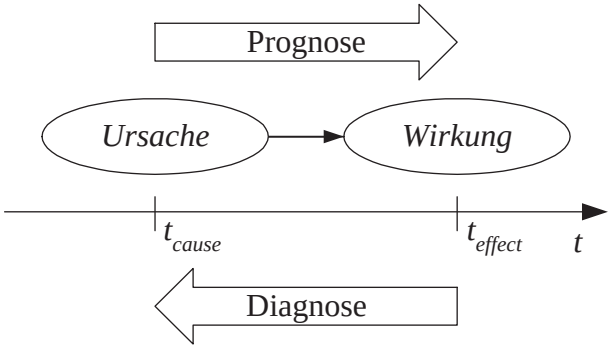

Abbildung 4.16: Prognose und Diagnose im Kontext BNe

t_{effect} können auch (beide) in der Vergangenheit oder in der Zukunft liegen, so dass z. B. auch analysiert werden kann, wie die durchschnittliche Termintreue der Produktion in der Gegenwart wäre, hätte man in der Vergangenheit eine andere Prioritätsregel angewandt.

Im Gegensatz zur Prognose sei unter Diagnose das Schließen von Wirkungen – auch Symptome genannt – auf Ursachen entgegen der kausalen Kette zu verstehen. Da Ursachen zeitlich immer vor ihren Wirkungen liegen, handelt es sich bei der Diagnose ebenso um ein Schließen von Zuständen zu einem Zeitpunkt t_{effect} auf Zustände zu einem Zeitpunkt t_{cause} mit $t_{effect} \geq t_{cause}$. Die Diagnose schließt nicht zwangsläufig vom gegenwärtigen Zeitpunkt in die Vergangenheit. Die Zeitpunkte t_{cause} oder t_{effect} können auch (beide) in der Vergangenheit oder in der Zukunft liegen, so dass z. B. auch bestimmt werden kann, welche Prioritätsregel man in der Gegenwart anwenden müsste, um in der Zukunft eine durchschnittliche Terminabweichung der Produktion von null zu erreichen. Abbildung 4.16 stellt Prognose und Diagnose noch einmal graphisch dar.

BNe und DBNe sind nun in der Lage, mittels PI Prognose und Diagnose sowohl in kausaler als auch in temporaler Hinsicht zu vollziehen. Prognose und Diagnose können und sollten sogar über ein und demselben BN vollzogen werden. Beide sind möglich, weil mittels PI von Zuständen oder von Randwahrscheinlichkeitsfunktionen beliebiger Variablen des BNes auf die wahrscheinlichste Konfiguration oder auf Randwahrscheinlichkeitsfunktionen beliebiger anderer Variablen des BNes geschlossen werden kann. Aus demselben Grunde erlauben BNe nicht nur Prognose und Diagnose, sondern auch gemischtes Schließen, das im folgenden Abschnitt behandelt wird.

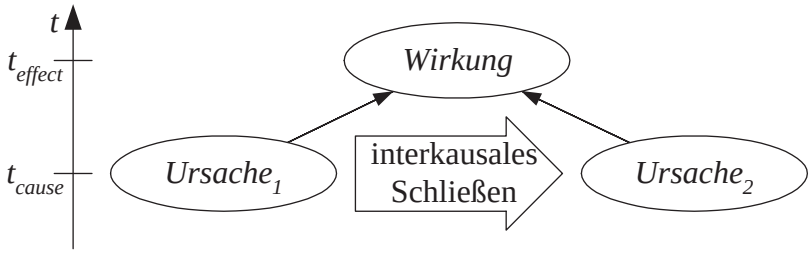

Abbildung 4.17: Interkausales Schließen im Kontext BNe

4.5.3 Verringern des Rechenaufwandes durch gemischtes Schließen

Gemischtes Schließen [KN04, S. 34 f.] sei das Schließen von Zuständen oder von Randwahrscheinlichkeitsfunktionen beliebiger Variablen eines BNes auf wahrscheinlichste Konfigurationen oder auf Randwahrscheinlichkeitsfunktionen beliebiger anderer Variablen des BNes. Prognose und Diagnose sind Sonderfälle des gemischten Schließens. Ein weiterer Sonderfall des gemischten Schließens ist das interkausale Schließen [KN04, S. 34 f.], bei dem von Ursachen über ihre Wirkungen auf andere Ursachen geschlossen wird (siehe Abbildung 4.17). Interkausales Schließen lässt sich ebenfalls im Rahmen der PPS verwenden. Folgende Frage könnte mittels interkausalen Schließens beantwortet werden: „Welche Menge an Rohstoffen sollte bestellt werden, um geringe Bestände beizubehalten, wenn die Auftragslage gut ist?“ Es würde *von* der Auftragslage als einer Ursache für die Bestände *über* die Bestände als Wirkung *auf* die Bestellmenge als eine weitere Ursache für die Bestände geschlossen.

Gemischtes Schließen entspricht faktisch der PI über BNen, und BNe können mittels PI gemischtes Schließen über sich selbst vollziehen. Gemischtes Schließen kombiniert Prognose, Diagnose und interkausales Schließen. Unter anderem die Fähigkeit, gemischt zu schließen, unterscheidet BNe z. B. von KNNen und Fuzzy-Regelsystemen und zeichnet sie vor ihnen aus. Die Fähigkeit BNe, gemischt zu schließen, wird im Allgemeinen und vor allem im Kontext der PPS als die bedeutendste erachtet, da mittels gemischten Schließens der Rechenaufwand zur Lösung von Problemen der Entscheidungsunterstützung und Regelungstechnik spürbar reduziert wird.

Im folgenden werde unter einer Handlungsalternative eine Konfigurationen von Stellgrößen verstanden, und erwünschte Zielfunktionswerte bezeichnen Führungswerte für Regelgrößen.

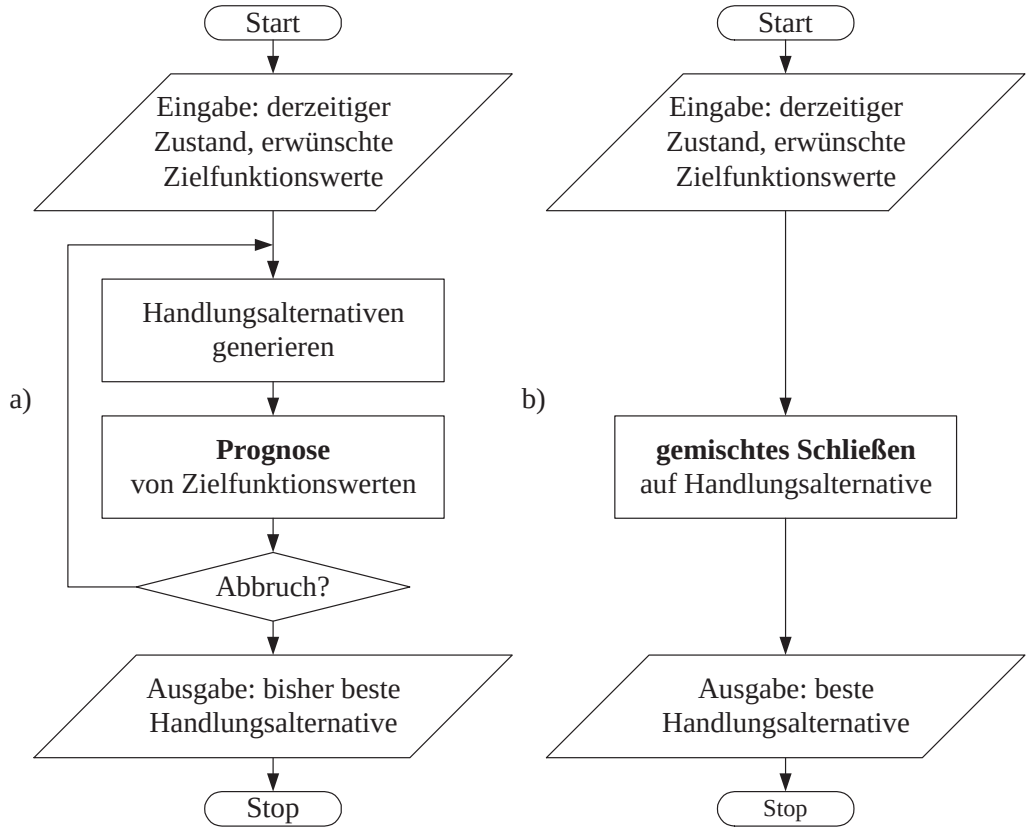

Abbildung 4.18: a) Iterative Optimierung versus b) gemischtes Schließen im Kontext BNe

Um Probleme der PPS zu lösen, wird zumeist iterativ optimiert, wie Abbildung 4.18 a) darstellt. Die iterative Optimierung – hier stark vereinfacht dargestellt – gibt erwünschte Zielfunktionswerte und derzeitige Werte von Zustandsgrößen vor, generiert erfolgversprechende Handlungsalternativen, prognostiziert Zielfunktionswerte aufgrund der generierten Handlungsalternativen, prüft eine Abbruchbedingung und beginnt, sofern nicht abgebrochen worden ist, wieder damit, weitere Handlungsalternativen zu generieren. Als Abbruchbedingung kann z. B. dienen, ob oder zu welchem Grade der erwünschte Zielfunktionswert erreicht worden ist und/oder ob noch Rechenzeit zur Verfügung steht. Oft erfolgt auch ein Abbruch, wenn über eine bestimmte Anzahl von Iterationen hinweg keine Verbesserung des Zielfunktionswertes stattgefunden hat. Aus der iterativen Vorgehensweise bei der Optimierung resultiert ein enorm hoher Rechenaufwand [Völ03, S. 25], da sowohl die Generierung zulässiger und erfolgversprechender Handlungsalternativen als auch die Prognose von Zielfunktionswerten sehr aufwendig ist.

Mittels gemischten Schließens über BNen kann im Gegensatz zur iterativen Optimierung sofort von erwünschten Zielfunktionswerten und von den gegenwärtigen Werten der Zustandsgrößen auf die Handlungsalternative geschlossen werden, aus welcher die vorgegebenen Zielfunktionswerte resultieren. Abbildung 4.18 b) stellt diesen Sachverhalt dar. Aus dem gemischten Schließen über BNen und der sequentiellen Vorgehensweise folgt deshalb ein wesentlich geringerer Rechenaufwand als aus der iterativen Optimierung. Zudem kann man mittels einer Prognose über dem BN überprüfen, ob die Handlungsalternative bei gegebenen Zustandsgrößen tatsächlich zu den erwünschten Zielfunktionswerten führt. Zum Überprüfen der ermittelten Handlungsalternative lässt sich auch die ereignisdiskrete Simulation einsetzen, da sie – nur einmal durchgeführt – lediglich moderaten Rechenaufwand verursacht.

Viele Verfahren, die zur Lösung der Probleme der PPS eingesetzt werden, arbeiten auf Basis der iterativen Optimierung, so z. B. die simulationsbasierte Optimierung [NB99, S. 114 ff.], [GR99, S. 223 f.], die auf der ereignisdiskreten Simulation beruht, lokale Suchverfahren, wie z. B. Simulated Annealing [AKL98] und Tabu-Search [GL01], und evolutionäre Algorithmen [Nis94], zu denen auch die genetischen Algorithmen zählen. BNe ermitteln aufgrund ihrer Fähigkeit zum gemischten Schließen die beste Handlungsalternative also mit weniger Rechenaufwand als die aufgeführten Verfahren der iterativen Optimierung.

Allerdings ist der Rechenaufwand zum Erstellen eines BNes, welches die Produktion ausreichend gut abbildet, nicht zu unterschätzen: Das Bestimmen relevanter Variablen und das experimentelle Festlegen des Abtastintervalles, verursachen hohen Rechenaufwand. Allerdings fällt der Rechenaufwand nicht in dem Zeitraum an, in dem eine Entscheidung im Rahmen der PPS getroffen werden soll, sondern vorab. Der Rechenaufwand, ein BN an eine Produktion anzupassen, die sich geändert hat, ist mit dem Aufwand zum Anpassen anderer Verfahren und Modelle zu vergleichen.

Auch stellen BNe – wie die anderen Verfahren ebenfalls – nicht sicher, dass wahrhaftig die beste Handlungsalternative ermittelt wird. Sie können ergo nicht als das Mittel zur Lösung aller Probleme der PPS angesehen werden; BNe stellen allerdings eine begründete Alternative zu den anderen Verfahren dar und lassen sich auch in Kombination mit letzteren einsetzen: So kann ein BN z. B. eine gegebene Anzahl von Handlungsalternativen bestimmen, welche mit den höchsten Wahrscheinlichkeiten zu den erwünschten Zielfunktionswerten führen – absteigend nach den Wahrscheinlichkeiten sortiert –, und ereignisdiskrete Simulation könnte diese Handlungsalternativen evaluieren.

4.5.4 Automatisches Erstellen BNe aus Daten

Mittels Gütemaßen und Suchverfahren können sowohl BNe als auch DBNe aus Datenbasen erstellt werden. Die Abschnitte 4.2.3 und 4.2.4 stellen angemessene Gütemaße und Suchverfahren vor. Im Kontext der PPS liegen große Mengen von Daten vor. PPS-Systeme speichern viele dieser Daten in internen Datenbanken über lange Zeiträume. Betriebsdatenerfassungen (BDEen) sind oft nicht in der Lage, die großen in der Produktion anfallenden Datenmengen über einen längeren Zeitraum zu speichern, so dass man die Daten gegebenenfalls in Data-Warehouses (DWs) und auch aggregiert auslagern kann. [Meh03], [Muc00] PPS-Systeme, BDEen und DWs stellen also die Daten zur Verfügung, aus denen BNe erstellt werden können. Die Vorstellung ist jedoch irrig, dass der Weg von den PPS-Daten zu einem aussagekräftigen oder effizienten BN kurz oder einfach sei. Zum einen sind die relevanten Daten zu identifizieren und zu extrahieren. Die Extraktion der Daten geht zumeist mittels Auswahlabfragen über mehreren verknüpften Tabellen der meist relationalen Datenbanken vonstatten. Die Identifikation relevanter Daten bzw. Variablen hingegen ist wesentlich schwieriger und aufwendiger. Sie ist bereits im Abschnitt 4.4.1 beschrieben worden und kann ebenfalls unter Zuhilfenahme BNe vorgenommen werden. In die Identifikation relevanter Daten sollte natürlich auch Expertenwissen einfließen – sofern vorhanden. Zum anderen sind die in den beschriebenen Datenbanken vorhandenen Daten mit verschiedenen Formen der Unsicherheit behaftet (siehe Abschnitt 2.3.1). Linguistische Unsicherheit liegt in den betrachteten Datenbanken nur selten vor. Gerade mit stochastischer Unsicherheit in Datenbasen können BNe hervorragend umgehen. Sie sind aber auch in der Lage, informationeller Unsicherheit zu begegnen, wie Abschnitt 4.5.6 noch näher darlegen wird. BNe können falsche Werte identifizieren und fehlende Werte bestimmen. BNe sind demzufolge fähig, sich automatisch aus Daten zu erstellen. Zur Datenvor- und -nachverarbeitung sind jedoch unter Umständen Eingriffe durch den Menschen vonnöten.

4.5.5 Integration von Expertenwissen

Wie bereits Abschnitt 4.2.2 beschreibt, lassen sich BNe vollständig aus Expertenwissen erstellen. Experten können also manuell Variablen identifizieren, ihre Zustände festgelegen, direkte kausale Beziehungen zwischen den Variablen qualifizieren und bedingte Wahrscheinlichkeiten schätzen. Gerade in großen Problembereichen oder in welchen, über die nur partiell Expertenwissen vorliegt, ist das jedoch nicht möglich. Dann bietet es sich an, BNe automatisch aus Daten zu erstellen und lediglich Expertenwissen zu integrie-

ren. Die Integration von Expertenwissen ist in allen Phasen zum Erstellen eines BNes (siehe Abbildung 4.4) möglich. Experten können Variablen und ihre Zustände ex ante vorgegeben und ex post verändern. Gegebenenfalls sind die nachfolgenden Schritte zum Erstellen eines BNes zu wiederholen. Gleiches gilt für das Schätzen der bedingten Wahrscheinlichkeiten: Auch sie können von Experten vorgegeben und aufgrund vorliegender Datensätze angepasst werden, wobei Daten und Expertenwissen zu wichten sind, oder der Experte passt die automatisch aus Daten geschätzten bedingten Wahrscheinlichkeiten im nachhinein manuell an.

Das folgende Beispiel illustriere das nachträgliche Anpassen einer von einem Experten geschätzten bedingten Wahrscheinlichkeit aufgrund von Daten: Es liege das BN aus dem Beispiel „Marie und ihre Rosen“ vor. Ein Experte habe die Wahrscheinlichkeit, dass die Straße nass ist, wenn es nicht regnet, mit $\Pr(st = na_{st}|re = ne_{re}) = 0{,}15$ geschätzt. In der Datenbasis liegen $m_d = 793$ Datensätze vor, welche den Zustand $re = ne_{re}$ enthalten. Von diesen m_d Datensätzen enthalten $n_d = 39$ zusätzlich den Zustand $st = na_{st}$ (und die verbleibenden 754 den Zustand $st = tr_{st}$). Es liegen also m_d relevante Datensätze vor. Die Daten und das Expertenwissen mit $m_e = m_d$ gewichtet, ergibt sich aus der vom Experten geschätzten bedingten Wahrscheinlichkeit

$$\begin{aligned}
\frac{n_e}{m_e} &= \Pr(st = na_{st}|re = ne_{re})|m_e \leftarrow m_d \\
\frac{n_e}{m_d} &= \Pr(st = na_{st}|re = ne_{re})| \cdot m_d \\
n_e &= \Pr(st = na_{st}|re = ne_{re}) \cdot m_d \\
n_e &= 0{,}15 \cdot 793 \\
n_e &\approx 119,
\end{aligned}$$

wobei m_e und n_e als die Anzahlen der Fälle interpretiert werden, welche der Experte beobachtet hat. Nun ergibt sich die folgende angepasste bedingte Wahrscheinlichkeit[24]:

$$\Pr(st = na_{st}|re = ne_{re}) = \frac{n_d + n_e}{m_d + m_e} = \frac{39 + 119}{793 + 793} = 0{,}09962169 \approx 0{,}1.$$

[24]Eine „angepasste“ bedingte Wahrscheinlichkeit ist quasi ein neuer Wert für eine bestimmte bedingte Wahrscheinlichkeit, die durch die Zustände ihrer bedingten und der sie bedingenden Variablen eindeutig charakterisiert wird. Die angepasste bedingte Wahrscheinlichkeit errechnet sich im Allgemeinen aus dem alten Wert der bedingten Wahrscheinlichkeit und einem für diese bedingte Wahrscheinlichkeit aus neuen Daten oder erneut von einem Experten geschätzten Wert, wobei eine Wichtung des alten oder des erneut geschätzten Wertes vorgenommen werden kann. Sofern möglich, bricht man das Errechnen der angepassten bedingten Wahrscheinlichkeit wie im Beispiel auf Fälle herunter.

Natürlich kann die bedingte Wahrscheinlichkeit auch effizienter adaptiert werden, aber der vorgestellte Weg ist erstens plausibel und leicht nachvollziehbar, zweitens stellt er dar, wie sich die vom Experten geschätzte Wahrscheinlichkeit in Fällen ausdrücken lässt, und drittens zeigt der Weg auch, wie man eine bereits aus Daten gewonnene bedingte Wahrscheinlichkeit mittels weiterer Daten anpassen könnte.

Am interessantesten ist die Integration des Expertenwissens allerdings bei der Identifikation direkter kausaler Beziehungen bzw. der Kanten im BN. Der Experte kann natürlich Kanten hinzufügen, entfernen oder invertieren, nachdem das BN automatisch aus Daten erstellt worden ist. Gegebenenfalls sind danach die bedingten Wahrscheinlichkeiten anzupassen. Es ist dem Experten allerdings auch möglich, Kanten vorzugeben oder zu verbieten, *bevor* die Struktur des BNes aus Daten erstellt wird. Die in Abschnitt 4.2.4 vorgestellten Algorithmen stellen sicher, dass die Vorgaben des Experten eingehalten werden, dass heißt, dass nach dem automatischen Erstellen des BNes aus Daten sämtliche vorgegebenen Kanten im BN enthalten und die vorab verbotenen Kanten nicht im BN enthalten sein werden.

BNe erlauben es also, in jeder Phase ihres Erstellens in beliebigem Umfang und auch mehrfach Expertenwissen zu integrieren. Das Erstellen BNe aus Expertenwissen und die Verfahren zum automatischen Erstellen BNe aus Daten gehen faktisch Hand in Hand.

4.5.6 Verarbeiten von Datensätzen mit fehlenden Werten

BNe sind in der Lage, sowohl im Rahmen ihres Erstellens als auch im Kontext der PI Datensätze mit fehlenden Werten[25] zu verarbeiten. Sie benötigen zu diesem Zwecke keine anderen als die ihnen inhärenten Eigenschaften und Fähigkeiten. Natürlich könnte man auch externe Methoden anwenden, um fehlende Werte zu behandeln, aber das ist im Kontext BNe nicht nötig.

Im folgenden werde vereinfachend angenommen, dass fehlende Werte rein zufällig in der Datenbasis auftreten und ihr Auftreten nicht von den Zuständen anderer Variablen im selben Datensatz abhänge. Fehlen z. B. in manchen Auftragsdatensätzen Werte für die Durchlaufzeit, so darf dieses Fehlen unter anderem nicht vom Auftragsumfang oder vom gefertigten Produkt abhängen. Allerdings darf das Gerät zum Erfassen der Durchlaufzeit zufällig ausfallen und unabhängig vom Auftrag, dessen Durchlaufzeit gerade zu erfassen ist.

[25] Auf englisch werden fehlende Werte als Missing Values bezeichnet. [CH92, S. 323], [SGS93, S. 270]

Verfahren, die auch Werte verarbeiten, welche systematisch fehlen, behandeln z. B. [RS01] und [RS99]. [Hec99, S. 321] und [Nea04, S. 354] verweisen auf weitere Verfahren. Eine zwar einfache, aber meist unzureichende Möglichkeit besteht darin, systematisch fehlende Werte als extra Zustände der entsprechenden Variablen zu behandeln.

Betrachten wir zuerst zufällig fehlende Werte beim automatischen Erstellen BNe aus Daten. Fehlende Werte sind beim Bestimmen der Güte eines BNes und beim Schätzen der bedingten Wahrscheinlichkeiten anhand einer unvollständigen Datenbasis zu verarbeiten. Die Güte und die bedingten Wahrscheinlichkeiten werden nur für eine Familie von Variablen berechnet und nicht für ein gesamtes BN (siehe z. B. Formeln 4.14 und 4.19). Der einfachste Weg, fehlende Werte zu handhaben, besteht darin, diejenigen Fälle auszuschließen, in denen für die Eltern und/oder für das Kind fehlende Werte auftreten. Dieser Weg wird auch praktiziert. [CH92, S. 324] So werden nicht alle Datensätze ausgeschlossen, die fehlende Werte aufweisen, sondern nur zeitweilig diejenigen, in welchen für die Variablen der aktuellen Familie Werte fehlen. Da es sich bei letzteren Datensätzen um relativ wenige handelt, ist dieser Weg durchaus akzeptabel.

Eine exakte Methode, einen unvollständigen Fall zu verarbeiten, basiert darauf, aufgrund der vollständigen Fälle eine Wahrscheinlichkeitsfunktion über den Alternativen für den unvollständigen Fall zu schätzen. Ein Beispiel soll diese Methode illustrieren: Tabelle 4.28 a) enthält die unvollständige Datenbasis. Ihr liegt die Datenbasis aus Tabelle 4.3 zugrunde. Lediglich in Fall 3 fehlen Werte für die Variablen Regen *re* und Sprenger *sp*. Der Fall 3 könnte in vier verschiedenen Alternativen aufgetreten sein – eine für jede mögliche Kombination aus Zuständen der Variablen, deren Werte fehlen. In der Datenbasis wird der Fall 3 demzufolge durch vier Fälle ersetzt, welche diesen Alternativen entsprechen. Tabelle 4.28 b) bezeichnet diese Fälle mit 3a bis 3d. Für jeden dieser Fälle wird die Wahrscheinlichkeit anhand der übrigen, vollständigen Fälle $\{1, \ldots, 10\} \setminus \{3\}$ in D geschätzt. Für die Konfiguration $(ne_{re}, au_{sp}, fe_{bo})$ beträgt sie z. B.

$$\Pr(ne_{re}, au_{sp}, fe_{bo}) = \frac{n_{(ne_{re}, au_{sp}, fe_{bo})}}{n_{(*,*,fe_{bo})}} = \frac{1}{5} = 0{,}2,$$

wobei n_c die absolute Häufigkeit der Konfiguration c in der Datenbasis D ohne Fall 3 bezeichne und $*$ einen Platzhalter für einen Wert bzw. Zustand. Für die übrigen Fälle in D' beträgt die Wahrscheinlichkeit 1,0, da sie sicher in D vorliegen. Tabelle 4.28 b) führt die Wahrscheinlichkeiten als „Wichtung“ w in der letzten Spalte auf. Eine bedingte Wahrscheinlichkeit ließe sich nun

Tabelle 4.28: a) Datenbasis D mit $N = 10$ Fällen über den Variablen $V = \{re, sp, bo\}$ aus dem Beispiel „Marie und ihre Rosen“ und fehlenden Werten für re und sp in Fall 3 – entspricht bis auf die fehlenden Werte der Datenbasis aus Tabelle 4.3 b) Datenbasis D' mit den Alternativen 3a bis 3d für Fall 3 und mit Datensätzen, gewichtet nach der Wahrscheinlichkeit ihres Auftretens

a)

#	re	sp	bo
1	ne_{re}	au_{sp}	tr_{bo}
2	ja_{re}	au_{sp}	fe_{bo}
3	?	?	fe_{bo}
4	ja_{re}	au_{sp}	fe_{bo}
5	ne_{re}	au_{sp}	fe_{bo}
6	ne_{re}	au_{sp}	tr_{bo}
7	ja_{re}	au_{sp}	fe_{bo}
8	ne_{re}	ei_{sp}	fe_{bo}
9	ne_{re}	au_{sp}	tr_{bo}
10	ne_{re}	au_{sp}	tr_{bo}

b)

#	re	sp	bo	w
1	ne_{re}	au_{sp}	tr_{bo}	1,0
2	ja_{re}	au_{sp}	fe_{bo}	1,0
3a	ne_{re}	au_{sp}	fe_{bo}	0,2
3b	ne_{re}	ei_{sp}	fe_{bo}	0,2
3c	ja_{re}	au_{sp}	fe_{bo}	0,6
3d	ja_{re}	ei_{sp}	fe_{bo}	0,0
4	ja_{re}	au_{sp}	fe_{bo}	1,0
5	ne_{re}	au_{sp}	fe_{bo}	1,0
6	ne_{re}	au_{sp}	tr_{bo}	1,0
7	ja_{re}	au_{sp}	fe_{bo}	1,0
8	ne_{re}	ei_{sp}	fe_{bo}	1,0
9	ne_{re}	au_{sp}	tr_{bo}	1,0
10	ne_{re}	au_{sp}	tr_{bo}	1,0

angelehnt an Formel 4.18 wie folgt schätzen:

$$\Pr(fe_{bo}|ja_{re}, au_{sp}) = \frac{\sum\limits_{\{\#\in D'|c_\#=(ja_{re},au_{sp},fe_{bo})\}} w_\#}{\sum\limits_{\{\#\in D'|c_\#=(*,*,fe_{bo})\}} w_\#} = \frac{3{,}6}{6} = 0{,}6,$$

wobei # den Datensatz darstelle und $c_\#$ und $w_\#$ die Konfiguration respektive die Wichtung des Datensatzes #. Die Schätzung erlaubt es, vorab Fälle mit identischen Konfigurationen zusammenzufassen und ihre Wichtungen zu addieren und so insgesamt Rechenaufwand zu verringern. Die eben beschriebene Vorgehensweise ist die bestmögliche. Sie kann ebenfalls genutzt werden, um bereits vorliegende Bewertungsfunktionen an neue, unvollständige Daten anzupassen. [Nea04, S. 349 ff.] Allerdings wächst der Rechenaufwand mit steigendem Anteil fehlender Werte exponentiell, weshalb die Vorgehensweise in der Praxis meist nicht tragbar ist. [CH92, S. 324], [KN04, S. 182] Es sind jedoch Verfahren entwickelt worden, welche diese Vorgehensweise approximieren: [MK97]. Eine andere Approximation fehlender Werte geht auf das Gibbs-Sampling zurück (siehe Anhang A.4 und [KN04, S. 182 ff.]). Es sind noch weitere approximierende Verfahren entworfen worden, über die [CDLS99, S. 200 ff.] und [Hec99, S. 320 ff.] eine guten Überblick geben.

Ein weiteres Vorgehen, fehlende Werte zu verarbeiten, läuft in drei aufeinander aufbauenden Schritten ab:

1. Das BN wird – wie oben beschrieben – aus einer Datenbasis mit fehlenden Werten erstellt: Beim Ermitteln der Güte einer Familie werden also die Datensätze nicht verwandt, bei denen für Variablen der Familie Werte fehlen.

2. Das aus unvollständigen Daten erstellte BN wird verwendet, um die fehlenden Werte zu bestimmen: Für jeden Datensatz, der fehlende Werte enthält, werden die Variablen, für die Werte im Datensatz vorliegen, mit diesen Werten instanziiert. Für die verbleibenden Variablen, für die Werte fehlen, wird mittels PI über dem BN die wahrscheinlichste Konfiguration ermittelt.[26] Die fehlenden Werte im Datensatz werden durch die Zustände in der wahrscheinlichsten Konfiguration ersetzt. Die Datenbasis weist nun keine fehlenden Werte mehr auf und ist somit vollständig.

3. Im Anschluss wird das BN noch einmal erstellt - aus einer nunmehr vollständigen Datenbasis, die keine fehlenden Werte mehr enthält.

[26] Die wahrscheinlichste Konfiguration kann auch ermittelt werden, wenn alle Werte im Datensatz fehlen, wobei sich dann die Frage stellt, ob der Datensatz nicht ausgesondert werden sollte.

Der Schritt 2 soll anhand eines Beispiels erläutert werden: In der Datenbasis für „Marie und ihre Rosen“ liege der unvollständige Fall

$$(re =?, sp = au_{sp}, st = tr_{st}, bo =?, ro = sc_{ro})$$

vor. Die Werte für die Variablen Regen re und Boden bo fehlen. Die Fragezeichen symbolisieren die fehlenden Werte. Im BN werden die Variablen Sprenger sp, Straße st und Rosen ro mit den Zuständen aus au_{sp}, nein ne_{re} respektive schlecht sc_{ro} instanziiert. PI über dem BN bestimmt nun die wahrscheinlichste Konfiguration der Variablen re und bo. Sie lautet $(re = ne_{re}, bo = tr_{bo})$ mit einer Wahrscheinlichkeit $\Pr(ne_{re}, tr_{bo} | au_{sp}, tr_{st}, sc_{ro}) \approx 0{,}44$. Diese Konfiguration vervollständigt nun den Fall um die Werte ne_{re} und tr_{bo} für die Variablen re und bo mit folgendem Ergebnis:

$$(re = ne_{re}, sp = au_{sp}, st = tr_{st}, bo = tr_{bo}, ro = sc_{ro}).$$

Bisher ist dargelegt worden, wie beim automatischen Erstellen BNe aus Daten fehlende Werte verarbeitet werden. Aber auch und gerade bei der PI über BNen werden fehlende Werte verarbeitet. Das Verarbeiten fehlender Werte ist im Kontext der PI ein natürliches Konzept: Es stehe eine Aufgabe an, welche mittels PI über einem vorliegenden BN B gelöst werden soll. Es bezeichne V die Variablen von B. Bestimmte Variablen $U \subset V$ sollten im Rahmen der Aufgabe instanziiert werden, und auf Randwahrscheinlichkeitsfunktionen R_W oder auf die wahrscheinlichste Konfiguration c_W anderer Variablen $W \subseteq V \setminus U$ des BNes sei probabilistisch zu schließen. Liegt nur für die Variablen $E \subset U$ Evidenz e_E vor, so dass nicht alle Variablen in U instanziiert werden können, ist trotzdem PI möglich. Das heißt, dass trotzdem R_w oder c_w ermittelt werden können. Anstelle von Evidenz gehen für die Variablen $U \setminus E$ – und auch für die intermediären Variablen $V \setminus (U \cup W)$ – deren Bewertungsfunktionen in die PI ein. Die Bewertungsfunktionen schlagen sich in den Potenzialfunktionen F der Cliquen C der Inferenzmaschine M nieder.

Wie bereits zu Beginn des Abschnittes 4.3 dargelegt, schließt PI auf Randwahrscheinlichkeitsfunktionen R_W oder die wahrscheinlichste Konfiguration c_W der Variablen W. Es wird ergo nicht vorausgesetzt, dass eine der Variablen $V \setminus W$ mit Evidenz behaftet ist.

Ein Beispiel verdeutliche das Dargelegte und zusätzlich, wie man dem BN sukzessive Evidenz zuführen und seine Variablen instanziieren kann. Es basiere wiederum auf „Marie und ihre Rosen“. Marie möchte wissen, ob es den Rosen ro gutgeht. Sie weiß nichts über die Variablen $V \setminus \{ro\}$. PI ohne jegliche Evidenz errechnet die Wahrscheinlichkeit, dass es den Rosen gutgeht: $\Pr(ro = gu_{ro}) \approx 0{,}48$. Marie genügt diese Wahrscheinlichkeit nicht. Sie sieht aus dem Fenster und stellt fest, dass die Straße st trocken tr_{st} ist. Sie instanziiert

$st = tr_{st}$ und vollzieht abermals PI. Die Wahrscheinlichkeit dafür, dass es den Rosen gutgeht, sinkt auf $\Pr(ro = gu_{ro} | st = tr_{st}) \approx 0{,}38$. Das gefällt Marie nicht, so dass sie in den Keller hinabsteigt, um nachzusehen, ob der Sprenger sp eingeschaltet worden ist. Der Sprenger sp befindet sich jedoch im Zustand au_{sp}. Marie instanziiert zusätzlich den Sprenger $sp = au_{sp}$ und vollzieht PI. Die Wahrscheinlichkeit fällt auf $\Pr(ro = gu_{ro} | st = tr_{st}, sp = au_{sp}) \approx 0{,}36$. Marie instanziiert den Sprenger sp mit ei_{sp}, um auszuprobieren, wie sich die Zustandsänderung des Sprengers auf die Rosen auswirkt. Nach erneuter PI steigt die Wahrscheinlichkeit für $ro = gu_{ro}$ auf $\Pr(ro = gu_{ro} | st = tr_{st}, sp = ei_{sp}) \approx 0{,}92$. Eine Wahrscheinlichkeit von rund 0,92 dafür, dass es den Rosen gutgeht, genügt Marie. Sie schaltet den Sprenger auch in der Realität ein und kann sich relativ sicher sein, dass es den Rosen gutgehen wird. (Sie hätte natürlich auch noch das Haus verlassen können, um zu spüren ob es regnet, oder in den Garten gehen können, um zu fühlen, ob der Boden feucht ist.)

Das Beispiel belegt, dass BNe bei der Wissensanwendung fehlende Werte akzeptieren und trotz fehlender Werte plausibel schließen bzw. plausible Aussagen treffen.

Im Gegensatz zu BNen akzeptieren KNNe und Fuzzy-Regelsysteme keine fehlenden Werte bei der Wissensanwendung, was ein großes Manko von KNNen und Fuzzy-Regelsystemen und einen entscheidenden Nachteil gegenüber BNen darstellt. Sofern Eingabewerte fehlen, können bei der Wissensanwendung im Kontext von KNNen oder Fuzzy-Regelsystemen zwar Standardwerte herangezogen werden, aber in der Regel bilden mehrdimensionale bedingte Wahrscheinlichkeitsfunktionen die Realität wesentlich besser ab als z. B. Mittelwerte eindimensionaler Wahrscheinlichkeitsfunktionen.

Bisher haben lediglich Werte in einer Datenbasis oder für die PI gefehlt. Der folgende Abschnitt 4.5.7 legt dar, wie Wissensverarbeitung im Kontext BNe versteckte Variablen entdeckt.

4.5.7 Entdecken versteckter Variablen

Werden BNe aus Daten erstellt, kann man nicht zwangsläufig davon ausgehen, dass die Datenbasis D alle Variablen V umfasst, welche für den betrachteten Problembereich relevant sind. Auch einem Experten gelingt es nicht immer, alle relevanten Variablen zu identifizieren. Es gilt, weitere relevante Variablen $U \subset V$ zu entdecken, die nicht in D enthalten sind bzw. die noch kein Experte identifiziert hat. Die Variablen U heißen versteckte

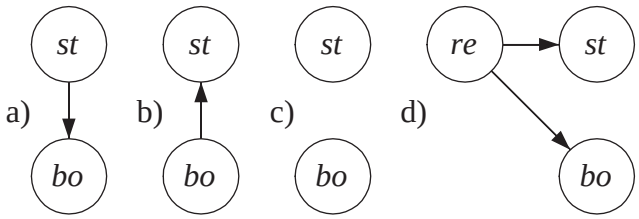

Abbildung 4.19: Vier gerichtete Graphen, von denen a), b) und c) die alternativen Graphen über den Variablen Straße *st* und Boden *bo* verkörpern und d) um die „versteckte“ Variable Regen *re* als Ursache für Straße *st* und Boden *bo* ergänzt worden ist

Variablen.[27] Wie man versteckte Variablen manuell identifiziert, beschreibt Abschnitt 4.2.2. Das Problem beim automatischen Entdecken versteckter Variablen besteht darin, dass potenziell unendlich viele solcher Variablen existieren und demzufolge auch unendlich viele Graphen, welche diese Variablen enthalten, untersucht werden müssten. [CH92, S. 326] Demzufolge beschränkt man sich im Allgemeinen darauf, mittels statistischer Verfahren einzelne versteckte Variablen zu identifizieren.

Das Identifizieren versteckter Variablen geht – grob vereinfachend – wie folgt vonstatten: Ein BN enthalte zwei Variablen u und v. Gütemaße oder statistische Tests von Annahmen über bedingte Unabhängigkeiten ermitteln, welche Kante zwischen den Variablen vorliegt: entweder die Kante (u, v) oder die Kante (v, u) oder *keine Kante*. Gelingt es ihnen nicht, sich eindeutig zwischen den Kanten (u, v) und (v, u) zu entscheiden, führte aber *keine Kante* zwischen u und v zu einem BN geringerer Güte bezüglich D, liegt ein Indiz dafür vor, dass eine versteckte Variable w existiert, welche die Variablen u und v über die Kanten (w, u) und (w, v) beeinflusst. In diesem Fall berücksichtigt das automatische Erstellen des BNes aus D zusätzlich die Variable w und die Kanten (w, u) und (w, v), aber keine Kante zwischen u und v.

Ein Beispiel, welches wiederum auf „Marie und ihre Rosen“ basiert, verdeutliche das Entdecken einer versteckten Variable: Zu Beginn liegen in D lediglich Spalten und Werte für die Variablen Straße *st* und Boden *bo* vor. Abbildung 4.19 stellt dar, was zwischen den Variablen *st* und *bo* vorliegen kann: entweder a) die Kante (st, bo) oder b) die Kante (bo, st) oder c) *keine Kante*. Die Variablen *st* und *bo* korrelieren offensichtlich miteinander: Die Zustände $st = tr_{st}$ und $bo = tr_{bo}$ sowie die Zustände $st = na_{st}$ und $bo = fe_{bo}$ treten in D meist gemeinsam auf. Welche der Variablen aber die Ursache

[27] Auf Englisch werden versteckte Variablen als Hidden, Latent oder Unobserved Variables bezeichnet. [Nea04, S. 469], [SGS00, S. 253] und [SRM96]

und welche die Wirkung verkörpert, kann nicht eindeutig bestimmt werden, weil zwischen den Variablen keine kausale Beziehung vorliegt. Bezüglich D seien die Güte des Graphen aus Abbildung 4.19 a) und die Güte des Graphen aus Abbildung 4.19 b) identisch, und die Güte des Graphen aus Abbildung 4.19 c) sei geringer. Demzufolge liegt ein Indiz dafür vor, dass eine versteckte Variable existiert. Im Beispiel wird die „versteckte“ Variable Regen re als gemeinsame Ursache für die Variablen Straße st und Boden bo „entdeckt“. Das automatische Erstellen des BNes aus D berücksichtigt nun zusätzlich die Variable Regen re und fügt die Kanten (re, st) und (re, bo) ein, aber keine zwischen st und bo. Abbildung 4.19 d) stellt den resultierenden Graphen dar.

[SGS00, S. 253 ff.], [SRM96] und [SSGM94, S. 29, 36] beschreiben im Detail Verfahren zur automatischen Identifikation versteckter Variablen in Daten.

Sind die versteckten Variablen identifiziert worden, werden sie der Datenbasis D als Spalten und dem BN als Knoten hinzugefügt. Die Suche nach der Struktur des BNes behandelt die versteckten Variablen wie gewöhnliche Variablen, für die *sämtliche* Werte in D fehlen. [Nea04, S. 470] Das heißt, dass in allen Spalten in D, die mit versteckten Variablen korrespondierenden, keine Werte vorliegen. Somit ist das Problem zurückgeführt worden auf das Verarbeiten fehlender Werte beim automatischen Erstellen eines BNes aus einer Datenbasis D (siehe Abschnitt 4.5.6).

Versteckte Variablen werden aus Gründen des Rechenaufwandes oft als zweiwertig angenommen, wobei auch mehrwertige Variablen möglich sind [Nea04, S. 469] und die Anzahl der Zustände versteckter Variablen optimiert werden kann [CH92, S. 326]

[Nea04, S. 469 ff.] und [SRM96] stellen ausführlich dar, wie versteckte Variablen entdeckt und in das automatische Erstellen BNe aus Daten integriert werden, und [Ram98] schlägt einen alternativen Ansatz zum hier geschilderten vor.

4.5.8 Akzeptanz, Plausibilität und Interpretierbarkeit BNe

Ein Problem der Wissensverarbeitung, zu welcher die Wissensrepräsentation, -akquisition und -anwendung gehören, besteht im Allgemeinen darin, dass ihre potenziellen Anwender sie nur schwer akzeptieren – und das nicht nur im Rahmen der PPS. Aus diesem Grunde werden Systeme zur Wissensverarbeitung oft nicht eingesetzt, obwohl sie einen wirtschaftlichen Nutzen stiften würden. Für das Symptom der fehlenden Akzeptanz der Wissensverarbeitung

in der PPS werden zwei grundlegende Ursachen diagnostiziert:

1. Der Mitarbeiter in der PPS fürchtet, dass ihn ein System zur Wissensverarbeitung ersetzt. Aus diesem Grund gibt er oft auch sein Expertenwissen über den von ihm bearbeiteten Problembereich nicht, spärlich oder nur ungern preis.

2. Der Mitarbeiter in der PPS kann die Wissensverarbeitung nicht interpretieren und erst recht nicht nachvollziehen. Ihre Ergebnisse erscheinen ihm – z. T. deshalb – nicht plausibel. Das führt dazu, dass der Mitarbeiter der Wissensverarbeitung misstraut.

Der Ursache 1 begegnet die Wissensverarbeitung im Kontext BNe damit, dass sie *nicht* den Experten und sein Verhalten modelliert – so wie es z. B. im Kontext regelbasierter Expertensysteme geschieht –, *sondern* den Problembereich. [Jen96, S. 3] Der Mitarbeiter verfügt ergo über ein Modell seines Problembereiches in Form eines BNes, mit dem er experimentieren kann. BNe trachten nicht danach, den Mitarbeiter zu ersetzen, sondern ihn zu unterstützen. [Jen96, S. 4] Sie nehmen dem Mitarbeiter in der PPS keine Entscheidung ab, sondern liefern ihm Entscheidungs*vorschläge*, die er umsetzen oder verwerfen kann. BNe erlauben es dem Mitarbeiter zudem, Entscheidungsvorschläge zu überprüfen, bevor er sie umsetzt.

Wie die Wissensverarbeitung im Kontext BNe der Ursache 2 entgegenwirkt, beschreibt der folgende Teil des vorliegenden Abschnittes.

Ein BN und insbesondere die graphische Repräsentation seines gerichteten, azyklischen Graphen sind als Wissensrepräsentationsform bereits sehr gut zu interpretieren. Direkte kausale Zusammenhänge zwischen einer (oder mehreren) Ursache(n) und einer Wirkung werden von Mitarbeitern in der Produktion(splanung und -steuerung) intuitiv erfasst – zumal sie diese Zusammenhänge oft sogar selbst angeben oder widerlegen. Man beachte nur das Beispiel „Marie und ihre Rosen“ und betrachte den dazugehörenden Graphen aus Abbildung 4.1. Auch die bedingten Wahrscheinlichkeiten aus den Bewertungsfunktionen sind selbst für Laien relativ leicht zu interpretieren. Man betrachte z. B. die Bewertungsfunktionen zum BN aus dem Beispiel „Marie und ihre Rosen“ in den Tabellen 4.1 und 4.2.

Schwerer als die Repräsentation des Wissens als BN lässt sich die *automatische* Akquisition eines BNes mittels Gütemaßen und Suchalgorithmen interpretieren. Es wird jedoch bezweifelt, dass im Rahmen der PPS das *Erstellen* eines Modelles derjenige Mitarbeiter interpretieren können muss, der das Modell später anwendet; zumal dem Mitarbeiter das *Modell* per se plausibel ist und er es interpretieren kann. Die *manuelle* Akquisition eines BNes

aus Expertenwissen kann auf jeden Fall als plausibel angesehen werden, zumal der Experte sein eigenes Wissen einbringt und mit einer interpretierbaren Wissensrepräsentationsform – dem BN – arbeitet. Sollte wider Erwarten auch die Interpretierbarkeit der automatischen Akquisition des BNes gefordert worden sein, kann auf andere Algorithmen zurückgegriffen werden, auf welche zum Ende des Abschnittes 4.2.4 kurz eingegangen worden ist und deren Arbeitsweise – im Gegensatz zu der von Gütemaßen und Suchalgorithmen – relativ leicht interpretiert werden kann.

Von Mitarbeitern in der Produktion bzw. in der PPS kann sicherlich nicht verlangt werden, dass sie die *Wissensanwendung* im Kontext BNe interpretieren oder gar im Detail nachvollziehen können, zumal z. B. der Aufbau einer Inferenzmaschine und das Message Passing im Rahmen der exakten PI (siehe Anhang A.3) nur schwer zu interpretieren sind. Qualitativ lässt sich das Verhalten eines BNes bei der PI aber am BN selbst erklären. Zu erklären ist, wie vorliegende Evidenz interessierende Variablen qualitativ beeinflusst. Einige abgeleitete Fragen, die relativ leicht zu beantworten sind, lauten wie folgt:

1. Beeinflusst die vorliegende Evidenz die interessierenden Variablen überhaupt?

2. Auf welchen Wegen beeinflusst die vorliegende Evidenz die interessierenden Variablen?

3. In welche Richtung beeinflusst die vorliegende Evidenz die interessierenden Variablen; positiv oder negativ?

4. Wie stark beeinflusst die vorliegende Evidenz die interessierenden Variablen?

Die Frage 1 lässt sich mittels des Kriteriums der d-Separation erklären, dass in Abschnitt 4.3.1 und in Definition 4.8 behandelt worden ist: Ist eine evidente Variable $u = e$ unter Berücksichtigung weiterer Evidenz *nicht* d-separiert von einer interessierenden Variable v, so übt e Einfluss auf v aus.

Die d-Separation beantwortet auch Frage 2. Rekursiv werden im Graphen, welcher dem Graphen des BNes unterliegt, diejenigen einfachen Pfade gesucht, welche u und v miteinander verbinden – also quasi dafür sorgen, dass u und v nicht d-separiert worden sind. Die Suche berücksichtigt die gerichteten Kanten des Graphen des BNes und weitere vorliegende Evidenz.

Im Beispiel „Marie und ihre Rosen“ liege die Evidenz nass na_{st} für die Variable Straße st vor. Die verbleibenden Variablen seien nicht evident. Von Interesse ist die Variable ro. Da st, re und bo divergierend miteinander verbunden sind (und für re keine Evidenz vorliegt), und da re, bo und ro seriell

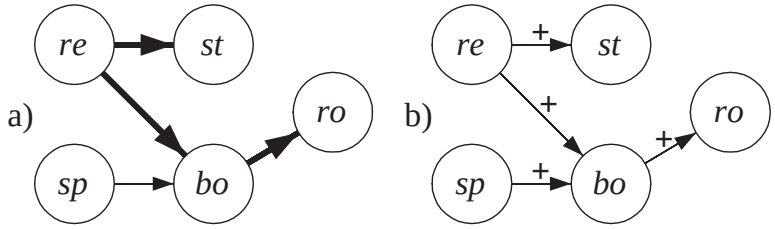

Abbildung 4.20: Der Graph des BNes aus dem Beispiel „Marie und ihre Rosen“ a) mit hervorgehobenen Kanten, entlang denen Evidenz von *st* zu *ro* fließt, und b) mit qualitativ attributierten Kanten

miteinander verbunden sind (und für *bo* keine Evidenz vorliegt), sind *st* und *ro nicht d*-separiert. Es kann Evidenz von *st* zu *ro* „fließen“, und (st, re, bo, ro) ist der einfache Pfad im unterliegenden Graphen U, entlang dem die Evidenz fließt.

Die gefundenen Pfade lassen sich z. B. in der visuellen Repräsentation des Graphen des BNes hervorheben und können so dem Mitarbeiter in der PPS präsentiert werden. Abbildung 4.20 a) hebt die über dem Pfad liegenden Kanten des BNes hervor. Allerdings bleibt fraglich, ob ein BN mit hunderten oder gar tausenden von Knoten, wie z. B. in [AWFA87], sinnvoll visualisiert und in ihm hervorgehobene Kanten erkannt und in ihrer Gesamtheit interpretiert werden können und sollten.

Es ist ebenfalls möglich, den Weg textuell als „Kausalzusammenhang“ wiederzugeben: Die Frage laute: Auf welchem Wege beeinflusst die Variable *Straße* die Variable *Rosen*? Die interpretierende, textuelle Antwort könnte wie folgt lauten: Die Variable *Straße* befindet sich im Zustand *feucht*. Da die Variable *Regen* die Variable *Straße* beeinflusst (und für die Variable *Regen* keine Evidenz vorliegt), ändert das Wissen um den Zustand der Variable *Straße* die Vermutung über den Zustand der Variable *Regen*. Da die Variable *Regen* die Variable *Boden* beeinflusst (und für die Variable *Boden* keine Evidenz vorliegt), ändert die Vermutung über den Zustand der Variable *Regen* die Vermutung über den Zustand der Variable *Boden*. Da die Variable *Boden* die Variable *Rosen* beeinflusst (und für die Variable *Rosen* keine Evidenz vorliegt), ändert die Vermutung über den Zustand der Variable *Boden* die Vermutung über den Zustand der Variable *Rosen*.[28, 29]

[28] Damit die textuelle Antwort – auch von einem Mitarbeiter in der PPS – leichter gelesen werden kann, sind absichtlich die Variablen*namen* verwendet und *kursiv* gesetzt worden.

[29] Die Worte Variable und Zustand sind mit Absicht verwendet worden. Beim automatischen Erstellen einer textuellen Antwort braucht so z. B. nicht auf das grammatikalische Geschlecht der Variablennamen Rücksicht genommen oder der Zusammenhang zwischen

Die Frage 3, in welche Richtung vorliegende Evidenz die interessierenden Variablen beeinflusst, beantworten z. B. [DH93a], [DH93b] und [Dru92]. Sie nennen ihr Verfahren qualitatives Schließen. Jede gerichtete Kante (u, v) eines BNes wird bezüglich ihrer Wirkung mit einem (Vor)zeichen attributiert: Das Zeichen $(+)$ bedeutet, dass ein Erhöhen/Verringern des Wertes der beeinflussenden Variable u wahrscheinlich ein Erhöhen/Verringern des Wertes der beeinflussten Variable v verursacht. Das Zeichen $(-)$ steht dafür, dass ein Erhöhen/Verringern von u wahrscheinlich ein Verringern/Erhöhen von v nach sich zieht. Entlang der bereits ermittelten Pfade kann nun nicht nur geschlossen werden, dass Evidenz eine bestimmte Variable beeinflusst, sondern auch, in welche Richtung.[30] Um auf das Beispiel „Marie und ihre Rosen“ zurückzukommen: Es liege die Evidenz $st = na_{st}$ vor. Die interessierende Variable sei ro. Abbildung 4.20 b) stellt den Graphen des BNes mit attributierten Kanten dar: Wenn die Straße nass ist, steigt die Wahrscheinlichkeit dafür, dass es regnet. Wenn es regnet, steigt die Wahrscheinlichkeit dafür, dass der Boden nass ist. Wenn der Boden nass ist, steigt die Wahrscheinlichkeit dafür, dass es den Rosen gutgeht. Die Evidenz beeinflusst den Zustand der Rosen ergo positiv.

Die Frage 4 nach der Stärke des Einflusses der Evidenz auf interessierende Variablen beantwortet [Sue92]. Er analysiert die Stärke, mit welcher die Evidenz auf interessierende Variablen wirkt, mittels einer Kostenfunktion über den Wegen, auf welchen die Evidenz besagte Variablen beeinflusst.

Eine ausführliche Übersicht über Ansätze und Algorithmen zur Interpretation der Wissensverarbeitung im Kontext BNe gibt [LD02].

4.5.9 Repräsentation von Wahrscheinlichkeitsverteilungen

Ein BN repräsentiert äußerst kompakt die gemeinsame Wahrscheinlichkeitsfunktion $\Pr(V)$ über V (siehe Theorem 4.1). Spannte man zur Repräsentation von $\Pr(V)$ den gesamten Zustandsraum S_V auf, würde die gemeinsame Wahrscheinlichkeitsverteilung $\Pr(V)$ zwar explizit repräsentiert, aber ihre Platzbe-

dem Variablen- und dem Zustandsnamen sprachlich erfasst werden: Die Aussagen „die Straße ist nass“, „den Rosen geht es gut“ oder „es regnet“ wären wesentlich schwieriger und nur unter zusätzlicher grammatikalischer und inhaltlicher Information zu treffen als die Aussagen „die Variable *Straße* befindet sich im Zustand *nass*“, „die Variable *Rosen* befindet sich im Zustand *gut*“ und „die Variable *Regen* befindet sich im Zustand *ja*“.

[30] Das qualitative Schließen wird hier stark vereinfacht dargestellt. Das Original bezieht auch (0) für neutralen und (?) für unbekannten Einfluss ein. Des weiteren attributiert und verrechnet das Original nicht nur einzelne Beziehungen zwischen zwei Variablen, sondern auch mehrere Beziehungen zwischen Eltern und ihrem Kinde.

darf wäre enorm, da er mit $\prod_{v \in V} |S_v|$ stark exponentiell wüchse - unabhängig davon, ob zwischen den Variablen (bedingte) Unabhängigkeiten vorliegen. Eine gemeinsame Wahrscheinlichkeitsverteilung über z. B. 27 zweiwertigen Variablen würde bei expliziter Repräsentation demzufolge ein GB Speicher beanspruchen, wenn man davon ausgeht, dass eine einzelne Wahrscheinlichkeit durch eine Gleitkommazahl dargestellt wird, die acht Byte Speicher belegt. Da der Platzbedarf für die Repräsentation einer gemeinsamen Wahrscheinlichkeitsverteilung durch ein BN nur mit $\sum_{v \in V}(|S_v| \prod_{u \in Pa_v} |S_u|)$ linear wächst, benötigte ein BN für die Repräsentation einer gemeinsamen Wahrscheinlichkeitsverteilung über 27 binären Variablen nur zirka 55 kByte, oder es könnte die gemeinsame Wahrscheinlichkeitsverteilung über mehr als 500.000 binären Variablen in einem GB Speicher repräsentieren, vorausgesetzt, eine Variable werde durch sieben andere Variablen direkt kausal beeinflusst (weise genau sieben Eltern auf), was relativ hoch angesetzt ist. Da Rechenanlagen direkt in der Produktion meist mit nur wenig Speicher ausgerüstet sind, bietet sich gerade dort eine Repräsentation gemeinsamer Wahrscheinlichkeitsverteilungen mittels BNe an.[31]

4.6 Bisherige Anwendungen BNe

Bisherige Anwendungen BNe lassen sich grob in drei Anwendungsgebiete einordnen: die Diagnose, die Prognose und die Entscheidungsunterstützung. BNe haben sich unter anderem aus dem Bedürfnis nach rechnergestützter medizinischer *Diagnose* entwickelt. Die medizinische Diagnose ist durch stochastische und informationelle Unsicherheit geprägt, welche herkömmliche regelbasierte Expertensysteme nicht abbilden können. Zudem liegt nicht sämtliche Evidenz zu Beginn der Diagnose vor, sondern die Evidenz wird bei Bedarf nach und nach eruiert, bis eine Diagnose mit akzeptabler Sicherheit gestellt werden kann. Da BNe für die rechnergestützte medizinische Diagnose entwickelt worden sind, decken sie die Anforderungen, die an die Diagnose gestellt werden, sehr gut ab. Einige der ersten Anwendungen BNe stammen demzufolge aus dem Gebiet der medizinischen Diagnose. Die bekannteste Anwendung ist wohl MUNIN [JAKA87], [AWFA87], das Ursachen für motorische Fehlfunktionen bei Menschen diagnostiziert. Das BN in MUNIN enthält zirka 1.000 Variablen. Eine weitere bekannte Anwendung ist PATHFINDER [Hec88], [HHN89], das Erkrankungen der Lymphknoten bei Menschen diagnostiziert. In PATHFINDER ist ein BN eingesetzt worden, weil die Ergebnisse der Inferenz über einem regelbasierten Expertensystem

[31] Der Speicherbedarf einer Inferenzmaschine übersteigt den eines BNes im Allgemeinen, kann jedoch durch bestimmte Verfahren stark verringert werden (siehe Abschnitt 4.3.4).

den Anwendern nicht vertrauenswürdig erschienen sind – im Gegensatz zu den Ergebnissen der PI über einem BN. Kurz darauf ist die Diagnose mittels BNe auch schnell und erfolgreich auf andere Gebiete übertragen worden. So werden Kreditkartenbetrug [MTVM02], [MTVM93], Betrug in der Telekommunikation [Hol99], [ES95] und Fehler in Halbleiterbauelementen [AG04] diagnostiziert. Die „Ratgeber“ unter Windows^{TM} basieren auf BNen und diagnostizieren Ursachen für fehlerhaftes Verhalten von Hard- und Software.[32] [Hed98] Einer von ihnen ist z. B. der bekannte Druck-Ratgeber. Die ebenfalls bekannten Microsoft Office^{TM}- und Antwort-Assistenten basieren auch auf BNen. [HBH^{+}98], [HH98] Die Diagnose mittels BNe ließe sich leicht in die PPS überführen. So könnten z. B. Ursachen für Materialverluste, für zu geringe Qualität oder für verspätete Aufträge diagnostiziert und anschließend bekämpft bzw. behoben werden.

Die *Prognose* ist ein weiteres Anwendungsgebiet BNe. Sind im BN auch diejenigen Variablen enthalten, welche die Variablen beeinflussen, die mittels Diagnose über einem BN als Ursachen für ein Fehlverhalten identifiziert worden sind, lässt sich mittels Prognose über demselben BN bestimmen, welche Wirkung ausgewählte Aktionen auf die Ursachen von Fehlverhalten haben. Im BN im Beispiel „Marie und ihre Rosen“ würden als Ursachen für das Symptom, dass es den Rosen schlechtgeht, trockener Boden und fehlender Regen diagnostiziert. Nun könnte man prognostizieren, wie sich das Anschalten des Sprengers auf das Befinden der Rosen auswirkt.

Prognose wird aber auch losgelöst von möglichen Aktionen vollzogen. So werden Hagelschläge [ABE^{+}96] und spatio-temporal das Wetter [CSG04] vorhergesagt sowie Ölpreise [AF91] prognostiziert. Insbesondere für die Prognose unter temporalem Aspekt bieten sich DBNe an. BNe oder DBNe können auch im Kontext der PPS zur Prognose angewandt werden. So ließen sich mittels BNe beispielsweise Primärbedarfe prognostizieren[33] oder Liefertermine (potenzieller) Kundenaufträge vorhersagen.

Neben Prognose und Diagnose werden BNe auch zur *Entscheidungsunterstützung* und zur *Regelung* eingesetzt. Eine der bekanntesten Anwendungen ist ein automatisches Taxi. [FHKR95] Ein DBN lenkt, beschleunigt und verzögert das Taxi, wobei eine Fahrtroute als möglichst einzuhaltende Trajektorie vorgegeben wird. Ein weiteres BN unterstützt das Manövrieren und das Operieren eines unbemannten Unterseebotes. [Exp04a] In [DV99, S. 297 ff., S. 325 ff.] wird die Anwendung BNe und DBNe zur Steuerung von Robotern beschrieben: Ein BN unterstützt einen Roboter bei der Auswahl alternativer

[32] Auf Englisch heißen diese Ratgeber Troubleshooter.

[33] Die dänische Firma HUGIN EXPERT A/S hat bereits ein BN für die Prognose des Teilebedarfs bei der Auftragsfertigung in der Automobilindustrie erstellt. [Exp04b]

Routen, und ein DBN hilft einem Roboter, bewegliche Objekte zu identifizieren und gegebenenfalls zu verfolgen.

Auf Basis BNe ist ein Entscheidungsunterstützungssystem entwickelt worden, welches Telemetriedaten des Space Shuttle, die in Echtzeit eintreffen, analysiert, interpretiert und auf ihrer Basis – ebenfalls in Echtzeit – Entscheidungsvorschläge unterbreitet. [HB95] Auch im Rahmen der PPS bietet sich der Einsatz BNe und DBNe zur Entscheidungsunterstützung oder gar zur Regelung an. So könnte mit Hilfe BNe über die Annahme oder Ablehnung zeitkritischer Kundenaufträge entschieden werden, BNe könnten vorausschauend über die Fertigung bestimmter Baugruppen auf Lager entscheiden oder Strategien zur Fertigungssteuerung auswählen und parametrisieren.

An dieser Stelle sind nur wenige, ausgewählte Anwendungen BNe vorgestellt worden, die zu den drei hauptsächlichen Anwendungsgebieten BNe – Prognose, Diagnose und Entscheidungsunterstützung – passen. Weitere Anwendungen widmen sich z. B. der Spracherkennung [DD04], dem Wiederauffinden von Informationen [CFLH04], oder sie sind auf dem Gebiet der Genetik angesiedelt [FLNP00]. Eine aktuelle Übersicht über einige der hier vorgestellten und viele weitere Anwendungen BNe geben [Nea04, S. 369 ff.] und [KN04, S. 117. ff.].

Kapitel 5

Theoretische Betrachtung des Potenzials BNe zur Unterstützung der PPS

Die theoretische Betrachtung des Potenzials BNe zur Unterstützung der PPS stellt den Problemen der PPS aus Kapitel 2 die Wissensverarbeitung im Kontext BNe aus Kapitel 4 gegenüber und legt dar, wie Eigenschaften und Fähigkeiten der Methoden der Wissensverarbeitung im Kontext BNe aus Kapitel 4 und insbesondere aus Abschnitt 4.5 dazu genutzt werden, die Probleme der PPS aus Kapitel 2 zu beheben. Das vorliegende Kapitel stellt somit das Bindeglied dar zwischen den Problemen der PPS auf der einen und ihrer Lösung mittels Methoden der Wissensverarbeitung im Kontext BNe auf der anderen Seite.

5.1 Potenzial BNe zur Abbildung und aktiven Nutzung von Unsicherheit

5.1.1 Potenzial BNe zur Abbildung und aktiven Nutzung informationeller Unsicherheit

Informationelle Unsicherheit liegt im Allgemeinen und in der PPS im Speziellen immer dann vor, wenn zu wenig Information, zu viel Information oder falsche Information zur Verfügung steht (siehe Abschnitt 2.3.1).

Methoden der automatischen Wissensakquisition im Kontext BNe sind insofern in der Lage, mit zu wenig Information umzugehen, als dass sie erstens fehlende Werte in den Daten approximieren, die zum automatischen Erstellen der BNe herangezogen werden (siehe Abschnitt 4.5.6). Zweitens

entdecken Methoden der automatischen Wissensakquisition im Kontext BNe „versteckte“ Variablen und beheben so ein weiteres Informationsdefizit (siehe Abschnitt 4.5.7).

Einem Informationsüberschuss werden Methoden der automatischen Wissensakquisition im Kontext BNe insofern gerecht, als dass sie erstens zu viele gleiche Fälle in den Daten nicht auswendig lernen, wie es z. B. KNNe tun würden (siehe „Overfitting“ in Abschnitt 3.3), sondern lediglich bedingte Wahrscheinlichkeiten anpassen und gegebenenfalls die Struktur des BNes verändern. Zweitens sortiert die automatische Wissensakquisition im Kontext BNe Variablen aus, die nicht oder zu wenig für das Problem relevant sind, indem sie sie entweder nicht durch Kanten mit anderen Variablen des BNes verbindet oder sie nicht direkt mit den relevanten Variablen verbindet, sondern nur über den Umweg über andere Variablen, wodurch sie die Informationsflut ebenfalls eindämmt.

Methoden der Wissensanwendung im Kontext BNe begegnen einem Informationsdefizit dadurch, dass sie fehlende Werte für Variablen durch Verteilungen bedingter Wahrscheinlichkeiten ersetzen oder gemeinsame Wahrscheinlichkeitsverteilungen für sie ermitteln (siehe Abschnitt 4.5.6), so dass BNe auch bei fehlenden Werten anwendbar bleiben und trotz der fehlenden Werte Randverteilungen oder wahrscheinlichste Konfigurationen ermitteln. Einem Informationsüberschuss begegnen Methoden der Wissensanwendung im Kontext BNe, indem sie für Evidenz, die das BNe überbestimmt, eine negative bzw. sehr geringe Plausibilit errechnen. [Jen96, S. 107 ff.], [Jen01, S. 208 ff.] Auf dieselbe Art und Weise erkennen Methoden der Wissensverarbeitung im Kontext BNe auch falsche und/oder unplausible Eingaben.

Methoden der Wissensrepräsentation im Kontext BNe bzw. BNe selbst kanalisieren einen Informationsüberfluss bereits dadurch, dass sie sehr viele Datensätze, einen komplexen Sachverhalt bzw. eine hochdimensionale gemeinsame Wahrscheinlichkeitsverteilung äußerst kompakt repräsentieren, indem sie (bedingte) Unabhängigkeiten zwischen den Variablen ausnutzen (siehe Abschnitt 4.5.9).

5.1.2 Potenzial BNe zur Abbildung und aktiven Nutzung stochastischer Unsicherheit

Wenn Werte einer Variable – gegebenenfalls bedingt durch Werte oder Zustände anderer Variablen – zufällig auftreten, spricht man von stochastischer Unsicherheit, die insbesondere in der PPS auftritt (siehe Abschnitt 2.3.1). Als Wissensrepräsentationsform bilden BNe stochastische Unsicherheit bereits adäquat mittels bedingter Wahrscheinlichkeiten ab.

Methoden der automatischen Wissensakquisition im Kontext BNe errechnen über stochastischen Daten Güten für BNe alternativer Strukturen und schätzen für Bewertungsfunktionen bedingte Wahrscheinlichkeiten, welche stochastische Unsicherheit bekanntlich sehr gut wiedergeben.

Methoden der Wissensanwendung im Kontext BNe nehmen stochastische Unsicherheit nicht nur billigend in Kauf, sondern nutzen sie darüber hinaus aktiv, indem sie nicht nur einzelne Werte bzw. Erwartungswerte propagieren, wie z. B. Fuzzy-Regelsysteme oder KNN, sondern Wahrscheinlichkeitsverteilungen. Sämtliche Ergebnisse der Wissensanwendung sind entweder selbst Wahrscheinlichkeiten oder zumindest mit Wahrscheinlichkeiten behaftet, seien es wahrscheinlichste Konfigurationen oder Summen- und Maximumränder, so dass auch die Ausgabe der PI bzw. die Ergebnisse der Wissensanwendung im Kontext BNe der stochastischen Unsicherheit Rechnung tragen.

5.2 Potenzial BNe zur Abbildung und Regelung komplexer stochastischer Prozesse

5.2.1 Potenzial BNe zur Prognose und Diagnose komplexer stochastischer Prozesse

Bei der Produktion handelt es sich um einen komplexen stochastischen Prozess (siehe Abschnitt 2.4). Um ihre bzw. seine Entwicklung vorherzusagen oder in der Vergangenheit begangene Fehler zu erkennen und in der Zukunft nicht zu wiederholen, bedarf es der Prognose und der Diagnose.

Wie bereits in Abschnitt 4.5.2 dargelegt, eignen sich BNe bzw. Methoden der Wissensanwendung in ihrem Kontext zur Prognose und zur Diagnose – und das jeweils sowohl in kausaler als auch in temporaler Hinsicht. DBNe und PI über ihnen eignen sich darüber hinaus zur Prognose und zur Diagnose stochastischer Prozesse, wie es die Produktion einer ist. Abbildung 5.1 stellt die Prognose mittels DBNe an einem abstrakten Beispiel dar.

Die Abbildung zeigt ein DBN mit sechs Zeitscheiben. In ihm liegen vier Variablen über der Zeit vor: Die Variable H verkörpert die Handlungsalternative, die auch als Entscheidungsvariable oder Stellgröße bezeichnet werden soll. Die Variablen S_1 und S_2 sind intermediäre Variablen und stellen gemeinsam den aktuellen Zustand der Fertigung dar. Die Variable Z gibt den Zielerreichungsgrad an. In der Gegenwart bzw. in der 0-ten Zeitscheibe werden die Variablen $H[0]$, $S_1[0]$, $S_2[0]$ und $Z[0]$ auf ihre aktuell anliegenden Werte gesetzt. Die Ausrufezeichen rechts oberhalb der jeweiligen Knoten bzw. Variablen in Abbildung 5.1 geben an, dass die zugehörigen Variablen

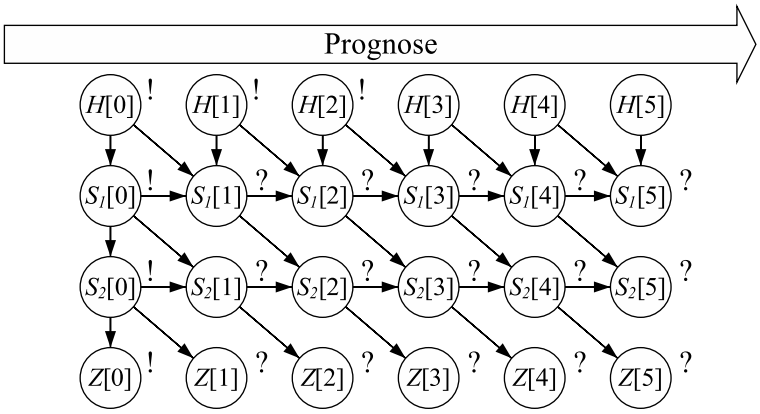

Abbildung 5.1: Prognose mittels DBNe

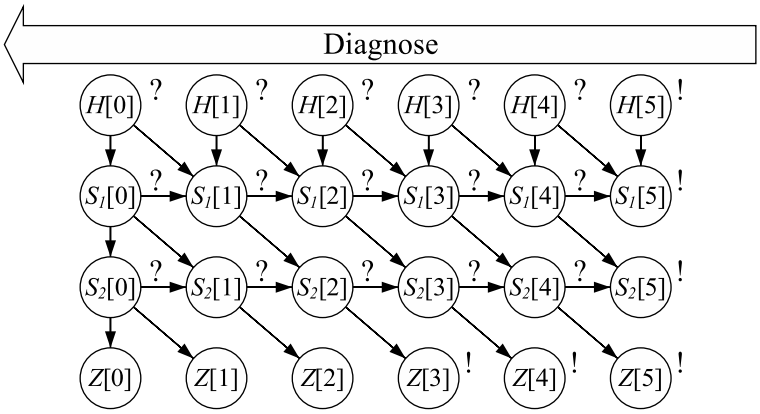

Abbildung 5.2: Diagnose mittels DBNe

gesetzt worden sind. PI prognostiziert nun in Richtung des Prognosepfeiles über dem DBN die zukünftige Konfiguration oder die zukünftigen Randverteilungen der Variablen $S_1[1], \ldots, S_1[5]$, $S_2[1], \ldots, S_2[5]$ und $Z[1], \ldots, Z[5]$. Die Fragezeichen rechts oberhalb der jeweiligen Knoten bzw. Variablen in Abbildung 5.1 geben an, dass es sich bei diesen Variablen um die gesuchten Variablen handelt. Möchte man ermitteln, welchen Einfluss für die Zukunft ausgewählte Handlungsalternativen auf (noch weiter) in der Zukunft liegende Zustände und Ziele ausüben, so können auch weitere Handlungsalternativen gesetzt werden, was die Ausrufezeichen an den Handlungsalternativen $H[1]$ und $H[2]$ andeuten.

Abbildung 5.2 veranschaulicht die Diagnose mittels des DBNes, das bereits in Abbildung 5.1 Verwendung gefunden hat. Diesmal verkörpern die Variablen $H[5]$, $S_1[5]$, $S_2[5]$ und $Z[5]$, die gegenwärtig anliegende Handlungs-

alternative, den gegenwärtigen Zustand respektive das gegenwärtig erreichte Ziel. Sie werden gesetzt, was wiederum die Ausrufezeichen rechts oberhalb der Knoten in Abbildung 5.2 angeben. Sämtliche anderen Variablen gelten für die Vergangenheit. Je geringer der Zeitscheibenindex, desto weiter liegen sie in der Vergangenheit. PI diagnostiziert nun in Richtung des Diagnosepfeiles und entgegen dem Zeitverlauf die Ursachen für den gegenwärtigen Zustand der Produktion: die Konfiguration oder die Randverteilungen der Variablen $S_1[0], \ldots, S_1[4]$, $S_2[0], \ldots, S_2[4]$ und insbesondere der Handlungsalternativen $H[0], \ldots, H[4]$. Die Fragezeichen in 5.2 kennzeichnen wiederum die gesuchten Variablen. Sollten Zielerreichungsgrade aus der Vergangenheit bekannt sein, so können sie ebenfalls gesetzt werden, was die Ausrufezeichen an den Zielen $Z[3]$ und $Z[4]$ andeuten, und die Diagnose mittels DBNe verbessern.

5.2.2 Potenzial BNe zur Regelung komplexer stochastischer Prozesse

Für die Produktion als komplexen stochastischen Prozess ist es besonders schwer, herauszufinden, welche Handlungsalternative gegenwärtig zu ergreifen ist, um in der Zukunft die gesteckten Ziele bestmöglich zu erreichen. Anders ausgedrückt, ist es kompliziert, die Konfiguration der Stellgrößen zu ermitteln, welche dazu führt, dass die Regelgrößen zukünftig die gewünschten Werte aufweisen.

Wie bereits in Abschnitt 4.5.3 dargelegt, eignen sich BNe und insbesondere DBNe besonders gut dazu, komplexe stochastische Prozesse zu regeln bzw. die Handlungsalternative zu ermitteln, welche den bestmöglichen Zielerreichungsgrad nach sich zieht. Sie vermeiden nämlich erstens die mehrfachen Prognosen im Rahmen der iterativen Optimierung (siehe Abbildung 4.18), und zweitens bezieht die PI die Stochastik des Prozesses ein und quantifiziert ihre Ergebnisse mit Wahrscheinlichkeiten.

Abbildung 5.3 stellt dar, wie die Entscheidungsunterstützung zur Regelung der Produktion als komplexer stochastischer Prozess vonstatten geht. Bei dem DBN handelt es sich um dasjenige, anhand dessen bereits Prognose und Diagnose in Abschnitt 5.2.1 erläutert worden sind. Die Variablen $S_1[0]$ und $S_2[0]$ der 0-ten Zeitscheibe werden auf ihre gegenwärtigen Werte gesetzt. Zusätzlich werden die Ziele $Z[0], \ldots, Z[5]$ auf die für sie angestrebten Ausprägungen gesetzt. Die Ausrufezeichen rechts oberhalb der Knoten/Variablen in Abbildung 5.3 kennzeichnen die evidenten (gesetzten) Variablen. PI schließt nun *in einem Schritt* vom gegenwärtigen Zustand der Produktion und von den für die Zukunft erwünschten Zielerreichungsgraden auf die Konfiguration der Handlungsalternativen $H[0], \ldots, H[5]$, welche zu diesen Zielerreichungs-

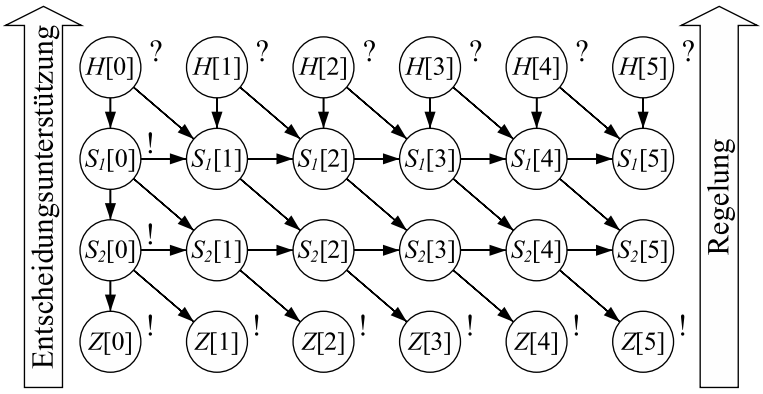

Abbildung 5.3: Regelung bzw. Entscheidungsunterstützung mittels DBNe

graden führen, und trifft Aussagen über die Wahrscheinlichkeit des Ergebnisses. Die Fragezeichen in Abbildung 5.3 kennzeichnen wiederum die gesuchten Variablen.

Möchte man zusätzlich überprüfen, ob die ermittelte Konfiguration der Handlungsalternativen tatsächlich zu den erwünschten Zielerreichungsgraden führt, so kann man sich einmalig der Prognose aus Abschnitt 5.2.1 bedienen.

Es folgen zwei Anmerkungen zu diesem und dem vorangegangenen Abschnitt zur Prognose und Diagnose der Produktion mittels BNe: Erstens kann und sollte immer dasselbe entrollte DBN zur Prognose, zur Diagnose und zur Entscheidungsunterstützung verwandt werden. Es bedarf ergo nicht dreier Modelle, sondern nur eines einzigen DBNes zur Lösung aller drei Aufgaben. Schließlich ist das DBN ja auch ein kausales Modell eines komplexen stochastischen Prozesses i. w. S. und der Fertigung i. e. S.

Zweitens können nicht nur mehrere intermediäre Variablen für den Zustand der Fertigung in einer Zeitscheibe eines DBNes vorliegen, sondern auch mehrere Variablen zum Abbilden der Handlungsalternative und mehrere Variablen zur Charakterisierung der technizitären Ersatzziele der PPS sind in einer Zeitscheibe zulässig. Natürlich sind diese Variablen wiederum untereinander und mit anderen Variablen über Kanten verknüpft, so dass die bedingten (Un)abhängigkeiten zwischen ihnen zum Ausdruck kommen.

5.3 Potenzial BNe zur Bewältigung der Komplexität

Die Produktion selbst und die PPS zu ihrer Planung und Regelung weisen im Allgemeinen eine sehr hohe Komplexität auf (siehe Abschnitt 2.5). Wie

Methoden der Wissensverarbeitung im Kontext BNe dieser Komplexität begegnen und sie letztendlich bewältigen, erläutert der vorliegende Abschnitt.

5.3.1 Komplexitätsreduktion durch Wissensrepräsentation

Obwohl BNe im Gegensatz zu vielen anderen Modellen, die in der PPS zur Abbildung, Planung und Regelung der Fertigung verwendet werden, der Stochastik der Produktion Rechnung tragen, repräsentieren sie äußerst kompakt die gemeinsame Wahrscheinlichkeitsverteilung über einer Menge von Variablen (siehe Abschnitt 4.5.9 sowie Theorem 4.1) in Form eines Graphen und bedingter Wahrscheinlichkeiten. DBNe repräsentieren darüber hinaus – und ebenfalls äußerst kompakt – sogar die gemeinsamsame Wahrscheinlichkeitsverteilung über der Entwicklung eines stochastischen Prozesses, wie z. B. der Produktion, über der Zeit.

Indem BNe die Teilaufgaben der PPS nicht als Optimierungsprobleme mit variablen Anzahlen von Variablen und Nebenbedingungen auffassen, wie sie der Betrachtung in Abschnitt 2.5 zugrundeliegen, sondern den Problembereich mit einer fixen Anzahl von Variablen modellieren und Prognose, Diagnose und/oder Entscheidungsunterstützung über ihm betreiben, lösen sie die Probleme der PPS indirekt auf einer höheren Abstraktionsstufe und reduzieren so die den Problemen der PPS inhärente Komplexität. Ob diese grundsätzliche Komplexitätsreduktion gerechtfertigt ist, muss im Einzelfall entschieden werden.

BNe begegnen der Komplexität der Produktion und der PPS, indem sie bedingte Unabhängigkeiten kodieren, die, würden sie nicht beachtet, das Modell zur Abbildung der Produktion und ihrer Regelung übermäßig komplex werden ließen. Variablen, die durch die Eingabe von Evidenz oder per se von den Variablen d-separiert worden sind (siehe Definition 4.8), die für das Problem direkt relevant sind – nämlich den Zielen und den Handlungsalternativen –, brauchen nicht mit ins BN einbezogen werden, da sie bei der PI weder die Ziele noch die Handlungsalternativen beeinflussen und auch nicht von ihnen beeinflusst werden. Wenn nur die tatsächlich für das jeweilige Problem der PPS relevanten Variablen im BN berücksichtigt werden, reduziert das die Komplexität und erleichtert die PI über dem BN.

DBNe begegnen der Komplexität der Produktion zusätzlich dadurch, dass sie drei Annahmen treffen, welche die Komplexität des DBNes als Modell reduzieren: Zustandswechsel gehen periodisch vonstatten, die Produktion ist ein markowscher Prozess, und die Produktion befindet sich in einem stationären Zustand. Ob diese Annahmen gerechtfertigt sind, muss allerdings im

Einzelfall entschieden werden. Falls das aber nicht der Fall sein sollte, können die Annahmen auch sukzessive gelockert und gar vollständig aufgegeben werden, was jedoch in einem überproportional starken Anstieg des Rechenaufwandes für die Wissensanwendung im Kontext BNe bzw. für die PI über BNen resultiert.

5.3.2 Komplexitätsreduktion durch Diskretisierung

Die Produktion und die PPS werden durch eine Vielzahl reellwertiger Variablen gekennzeichnet. Diskretisiert man die Wertebereiche reellwertiger Variablen oder fasst die Werte ganzzahliger, ordinaler oder gar nominaler Variablen zu sich gegenseitig ausschließenden, aber gemeinsam den Wertebereich abdeckenden Gruppen zusammen (siehe Abschnitt 4.4.3), geht das zwingend mit einem Informationsverlust einher, was aber nicht grundsätzlich von Nachteil ist: Der positive Nebeneffekt einer Diskretisierung, wie sie BNe nicht grundsätzlich, aber im Allgemeinen benötigen, ist eine Komplexitätsreduktion.

Ob das Diskretisieren der Wertebereiche reellwertiger Variablen oder das Zusammenfassen der Werte ganzzahliger, ordinaler oder nominaler Variablen im Kontext des zu lösenden Problems der PPS zulässig ist und das Modell bzw. das Problem nicht so stark vereinfacht, so dass die Wissensanwendung im Kontext BNe keine zufriedenstellenden Ergebnisse mehr liefert, muss im Einzelfall und nötigenfalls durch Experimente entschieden werden, aber vielleicht bedarf eine Variable, wie z. B. die Warteschlangenlänge nicht eines Wertebereiches von $0, 1, \ldots, n$, sondern lediglich dreier Zustände: 1.) „leer“, 2.) „zwei bis fünf Aufträge“ und 3.) „mehr als fünf Aufträge“.

An dieser Stelle soll nicht verschwiegen werden, dass die Diskretisierung bzw. die Gruppierung mit einem Aufwand einhergeht, der erheblich sein kann, wenn die betroffenen Variablen rein nominal sind, wenn die Intervalle bzw. Gruppen nicht gleich groß sein sollen oder wenn die Intervall- bzw. Gruppengrenzen von n Variablen so gewählt werden sollen, dass sie einen n-dimensionalen Raum möglichst elegant in wiederum n-dimensionale Bereiche aufteilen (Clustering). Dieser Aufwand fällt allerdings nur einmal an, ist somit fix und amortisiert sich durch die PI, welche über einem BN mit sinnvoll diskretisierten Variablen schneller abläuft und bessere Ergebnisse zeitigt.

5.3.3 Komplexitätsreduktion bei der Wissensakquisition

Die Produktion und die PPS werden durch eine Vielzahl von Variablen beschrieben, die nicht alle für ein zu lösendes Problem relevant sind. Würde

man alle Variablen in das BN einbeziehen, welches die Lösung des Problems der PPS unterstützen soll, so würde das BN übermäßig komplex – insbesondere, was seine Kanten angeht –, und der Aufwand für die PI über dem BN wüchse inakzeptabel.

Die Gütemaße und Suchalgorithmen zur automatischen Akquisition BNe sortieren nun nicht oder nur wenig für das Problem relevante Variablen aus der Menge der verfügbaren Variablen aus. Sie fügen entweder keine Kanten zwischen diesen Variablen und den Variablen ein, die bereits als für das Problem relevant identifiziert worden sind, oder sie verbinden sie nicht direkt mit den relevanten Variablen, so dass das BN später einem „Pruning“ unterzogen werden kann, welches die Variablen „wegschneidet“, die von den relevanten Variablen unter Berücksichtigung der Eingabe von Evidenz *d*-separiert sind (siehe Definition 4.8). Die Komplexität des Modells bzw. des BNes sinkt, ohne dass darunter die Abbildungsgenauigkeit leidet, und die PI über dem BN verursacht wesentlich weniger Rechenaufwand.

Produktion und PPS werden nicht nur durch sehr viele Variablen, beschrieben, sondern es fallen auch sehr viele Daten an, welche vergangene Zustände der Produktion beschreiben und in denen komplexe Zusammenhänge vorliegen, die nicht ad hoc zu erkennen sind. Um aus diesen Daten und den komplexen Zusammenhängen in ihnen Schlüsse für die Zukunft zu ziehen, bedürfen diese Daten und die in ihnen verborgenen komplexen Abhängigkeiten einer Analyse und einer Synthese zu einem Modell. Die automatische Wissensakquisition im Kontext BNe nimmt eine solche Analyse vor und synthetisiert ein BN, das die Daten und insbesondere die komplexen Zusammenhänge zwischen ihnen kompakt widerspiegelt (siehe Abschnitt 4.5.9). Somit ist die Wissensakquisition im Kontext BNe auch in der Lage, Komplexität zu beherrschen, die sich aus der schieren Menge anfallender Daten und den in ihnen verborgenen Zusammenhängen ergibt.

Doch nicht nur die automatische Wissensakquisition im Kontext BNe ist durch Komplexität gekennzeichnet, sondern auch die manuelle. Soll der Einfluss mehrerer gleichzeitig wirkender beeinflussender Variablen auf eine beeinflusste Variable manuell quantifiziert werden, sind also im Rahmen der Akquisition BNe Wahrscheinlichkeiten, die mehrfach bedingt sind, manuell zu schätzen, so fällt das einem Menschen gemeinhin sehr schwer, auch wenn er Experte im Problembereich der PPS ist. Zudem wächst die Anzahl zu schätzender bedingter Wahrscheinlichkeiten exponentiell mit der Anzahl der beeinflussenden Variablen, was das Problem verschärft und ihm Komplexität verleiht. Diese Komplexität wird durch „verrauschte Operatoren“ verringert (siehe Abschnitt 4.2.5). Diese Operatoren erlauben es, Wahrscheinlichkeiten zu schätzen, die nur durch eine beeinflussende Variable bedingt sind, was dem Menschen wesentlich leichter fällt. Diese einfach bedingten Wahrscheinlich-

keiten müssen allerdings für jede beeinflussende Variable geschätzt werden. Die verrauschten Operatoren aggregieren im Anschluss die einfach bedingten Wahrscheinlichkeiten zu mehrfach bedingten Wahrscheinlichkeiten und erzeugen so letztlich eine Bewertungsfunktion für die beeinflusste Variable.

Durch den Einsatz der sogenannten Noisy-Operatoren wächst die Anzahl der zu schätzenden bedingten Wahrscheinlichkeiten zudem nur noch linear mit der Anzahl der beeinflussenden Variablen, was die eigentliche Komplexitätsreduktion beim manuellen Schätzen der bedingten Wahrscheinlichkeiten bewirkt.

5.3.4 Komplexitätsreduktion bei der probabilistischen Inferenz

Zur Lösung der Teilaufgaben der PPS werden oft Verfahren der „iterativen Optimierung“ angewendet, welche zyklisch eine Handlungsalternative auswählen, ihre Auswirkungen auf die Ziele der PPS prognostizieren und erst nach dem Eintreten eines bestimmten Kriteriums abbrechen [siehe Abbildung 4.18 a)]. Die Verfahren benötigen oft mehrere tausend Iterationen, bis sie ein Abbruchkriterium erreichen bzw. eine „optimale“ Handlungsalternative hervorbringen. Die iterative Optimierung weist den Nachteil auf, dass die vielfache Suche nach erfolgversprechenden Handlungsalternativen und die vielfache Prognose ihrer Konsequenzen sehr aufwendig ist. Hält sich der Aufwand für die mehrfache Prognose mittels KNNe noch in Grenzen, so ruft die Prognose mittels ereignisdiskreter Simulation einen nicht unerheblichen Aufwand hervor – besonders dann, wenn sie stochastische Komponenten enthält und zum Erreichen einer statistischen Sicherheit für jede Prognose mehrfach ausgeführt werden muss.

Aufgrund ihrer Fähigkeit zum „gemischten Schließen“ (siehe Abschnitt 4.5.3) bedarf PI nicht einer vielfachen Auswahl von Handlungsalternativen und einer vielfachen Prognose ihrer Auswirkungen auf die technizitären Ersatzziele der PPS. Im Gegensatz zu den Verfahren der iterativen Optimierung schließt PI einmalig von den erwünschten Zielerreichungsgraden und dem gegenwärtigen Zustand der Fertigung auf die Handlungsalternativen, die zu diesen Zielerreichungsgraden führen [siehe Abbildung 4.18 b)] – und das über einem kausalen Modell und unter Berücksichtigung der Stochastik der Produktion durch Wahrscheinlichkeiten. Diese Fähigkeit BNe zum gemischten Schließen wird als die herausragende Fähigkeit BNe angesehen und prädestiniert BNe zur Unterstützung der PPS.

PI verursacht zwar Aufwand beim Erstellen der Inferenzmaschine, aber der Aufwand z. B. für das Training eines KNNes bewegt sich in denselben

Größenordnungen. Geht das gemischte Schließen mittels BNe auch langsamer vonstatten als z. B. eine Prognose mittels KNNe, die ausgesprochen schnell ist, so muss es nicht wiederholt, sondern nur einmal durchgeführt werden. Ist die Komplexität der PI immer noch zu hoch, so bieten sich Verfahren der approximativen PI an (siehe Abschnitt 4.3.4).

5.4 Potenzial BNe zur Behandlung der Strukturdefekte

Die meisten Teilaufgaben der PPS sind von Strukturdefekten geprägt (siehe Abschnitt 2.6.1), welche diese Teilaufgaben der PPS zu schlechtstrukturierten Problemen machen. Eine Übersicht über die schlechtstrukturierten Probleme bzw. Teilaufgaben der PPS und darüber, welche Strukturdefekte ihnen anhaften, geben Abschnitt 2.6.2 und Tabelle 2.5.

Die folgenden Unterkapitel legen dar, wie Methoden der Wissensverarbeitung im Kontext BNe Strukturdefekte mildern oder beheben und so schlechtstrukturierte Probleme lösen. Sind BNe in der Lage, schlechtstrukturierte Probleme grundsätzlich zu lösen, so lösen sie auch die schlechtstrukturierten Probleme der PPS und eignen sich somit auch zur Unterstützung der PPS.

5.4.1 Potenzial BNe zur Behandlung des Bewertungsdefektes

Ein Bewertungsdefekt liegt vor, wenn die Variablen, die das Problem beschreiben sollten, oder deren Ausprägungen unbekannt sind. BNe beheben den Bewertungsdefekt, indem sie „versteckte“ bzw. unbekannte Variablen entdecken. Abschnitt 4.5.7 stellt dar, wie das Entdecken versteckter Variablen abläuft. Unbekannte Variablen können allerdings nur entdeckt werden, wenn andere Variablen bereits bekannt sind, die potenziell mit den unbekannten Variablen in Beziehung stehen. Sind noch gar keine Variablen bekannt, so gelingt es, BNe automatisch aus Daten zu akquirieren und so relevante Variablen zu identifizieren und irrelevante zu verwerfen.

Sind jedoch keine Variablen bekannt und liegen auch keine Daten über das schlechtstrukturierte Problem vor, so sind auch Methoden der Wissensverarbeitung im Kontext BNe nicht in der Lage, Bewertungsdefekte zu beheben. In diesem pathologischen Falle besteht aber zumindest im Umfeld der Produktion und der PPS zumeist die Möglichkeit, Daten zu erheben, so dass BNe aus ihnen automatisch akquiriert und relevante Variablen identifiziert werden können.

Sind die Ausprägungen von Variablen unbekannt, so ermöglichen sowohl die automatische Akquisition BNe aus Daten als auch die PI über BNen, Datensätze mit fehlenden Werten zu verarbeiten. Abschnitt 4.5.6 legt sowohl abstrakt als auch anhand von Beispielen dar, wie Datensätze mit fehlenden Werten problemlos und kohärent von Methoden der Wissensakquisition und -anwendung im Kontext BNe verarbeitet werden.

5.4.2 Potenzial BNe zur Behandlung des Wirkungsdefektes

Ein Wirkungsdefekt liegt dann vor, wenn die Wirkungszusammenhänge zwischen den Variablen, die das schlechtstrukturierte Problem beschreiben, unbekannt oder unsicher sind. Methoden der automatischen Akquisition BNe aus Daten beheben den Wirkungsdefekt, indem sie sukzessive gerichtete Kanten in ein leeres BN einfügen. Die Kanten repräsentieren die direkten kausalen Beziehungen zwischen den Variablen, die in den Daten implizit vorliegen, die sogenannten Wirkungszusammenhänge. Die Kanten geben die Wirkungszusammenhänge allerdings nur qualitativ an. Bedingte Wahrscheinlichkeiten quantifizieren die Wirkungszusammenhänge und bilden gemeinsam sogenannte Bewertungsfunktionen (siehe Definition A.5). Die bedingten Wahrscheinlichkeiten können ebenfalls automatisch aus Daten geschätzt werden. Die Quantifizierung der Wirkungszusammenhänge mit Methoden der Akquisition BNe geht ergo auch automatisch vonstatten.

Wirkungszusammenhänge sind nicht immer sicher bzw. nicht immer deterministisch. Mittels der bedingten Wahrscheinlichkeiten gelingt es BNen, stochastische Wirkungszusammenhänge adäquat abzubilden. Im Anschluss können Methoden der Wissensanwendung im Kontext BNe über diesen stochastischen Wirkungszusammenhängen probabilistisch schließen.

BNe bzw. Methoden ihrer automatischen Akquisition aus Daten sind zum Ermitteln von Wirkungszusammenhängen geradezu prädestiniert. Selbst wenn BNe später nicht zur Prognose, zur Diagnose oder zur Entscheidungsunterstützung eingesetzt werden sollten, können die durch die automatische Akquisition BNe aus Daten gewonnenen qualitativen Wirkungszusammenhänge zum Bau anderer Wissensrepräsentationsformen, wie z. B. Fuzzy-Regelsystemen, verwandt werden.

5.4.3 Potenzial BNe zur Behandlung des Zielsetzungsdefektes

Ein Zielsetzungsdefekt besteht immer dann, wenn für die Lösung eines Problems keine operationale Zielfunktion vorliegt. Gerade in der PPS liegen oft keine operationalen Zielfunktionen vor, weil mehrere interdependierende und zum Teil konträre Ziele verfolgt werden, die nicht auf ein gemeinsames Oberziel, wie z. B. die Kosten, abgebildet werden können (siehe Abschnitt 2.1).

BNe sind in der Lage, den Zielsetzungsdefekt auf zweierlei Arten zu beheben. Entweder beziehen sie das übergeordnete Ziel, wie z. B. die Kosten, mit in die automatische Akquisition BNe aus Daten ein und versuchen (unsichere) Wirkungszusammenhänge zwischen den Unterzielen, wie z. B. den technizitären Ersatzzielen der PPS, und dem übergeordneten Ziel herzustellen. Gelingt es, diese Wirkungszusammenhänge herzustellen, kann der Zielsetzungsdefekt als behoben betrachtet werden. Oder die automatische Akquisition BNe aus Daten lässt das übergeordnete Ziel unberücksichtigt und ermittelt nur die Wirkungszusammenhänge zwischen den Unterzielen und zwischen anderen, das schlechtstrukturierte Problem beschreibenden Variablen und den Unterzielen. Erkennt die automatische Akquisition BNe aus Daten diese unsicheren Wirkungszusammenhänge, so behebt sie den Zielsetzungsdefekt ebenfalls.

Sind besagte Wirkungszusammenhänge ermittelt worden, liegt im BN eine Art „stochastische Zielfunktion“ vor, welche die PI im Rahmen einer Entscheidungsunterstützung mittels BNe verwendet. Wie Entscheidungsunterstützung mittels BNe ein schlechtstrukturiertes Problem löst, beschreibt der folgende Abschnitt.

5.4.4 Potenzial BNe zur Behandlung des Lösungsdefektes

Ein Lösungsdefekt ist dann gegeben, wenn kein effizientes Lösungsverfahren für das betrachtete Problem existiert. Viele Teilaufgaben der PPS weisen Lösungsdefekte auf (siehe Tabelle 2.5). Entscheidungsunterstützung mittels BNe behebt den Lösungsdefekt insofern, als dass sie in einem Schritt die Handlungsalternative ermittelt, welche die gesteckten Ziele unter Berücksichtigung der aktuellen Situation am besten erreicht (siehe Abbildung 4.18). Eine iterative Optimierung ist nicht vonnöten. Wie die Entscheidungsunterstützung mittels BNe im Allgemeinen abläuft, beschreiben bereits die Abschnitte 4.5.3 und 5.2.2.

Sofern das übergeordnete Ziel aus dem vorangegangenen Abschnitt in das BN integriert worden ist, wird dieses Ziel „gesetzt“, und die PI ermittelt die

wahrscheinlichste Konfiguration der Variablen, welche die Handlungsalternativen abbilden. Diese Konfiguration bzw. diese Handlungsalternative führt – umgesetzt – zu dem Wert, auf den das übergeordnete Ziel gesetzt worden ist, was mittels einer Prognose über dem BN überprüft werden kann.

Ist das übergeordnete Ziel aus dem vorangegangenen Abschnitt nicht in das BN integriert worden, werden alle untergeordneten Ziele auf die gewünschten Werte „gesetzt“, und die PI ermittelt wiederum die wahrscheinlichste Konfiguration der Variablen, welche die Handlungsalternativen abbilden. Diese Konfiguration bzw. diese Handlungsalternative führt – umgesetzt – dazu, dass alle untergeordneten Ziele erreicht werden. Bei dieser Vorgehensweise besteht allerdings die Gefahr, dass die Wahrscheinlichkeit für die gewählte Konfiguration der untergeordneten Ziele bereits sehr gering ist, und die Wahrscheinlichkeit dafür, dass die ermittelte Handlungsalternative zu dieser Zielkonfiguration führt, ist noch geringer. Aus diesem Grunde sollte die Handlungsalternative mittels einer Prognose über dem BN überprüft werden. Wie eine Prognose mittels BNe vonstatten geht, beschreibt bereits Abschnitt 5.2.1.

5.5 Zusammenfassende Beurteilung des Potenzials BNe zur Unterstützung der PPS

Aus den folgenden vier Gründen weist die Wissensverarbeitung im Kontext BNe ein sehr breites und sehr tiefes Potenzial zur Unterstützung der PPS auf:

- Die Fertigung und die PPS sind mit informationeller und stochastischer Unsicherheit behaftet, die abgebildet und adäquat verarbeitet werden muss. Die Wissensverarbeitung im Kontext BNe bildet stochastische und informationelle Unsicherheit ab und nutzt sie aktiv.

- Bei der Fertigung handelt es sich um einen komplexen stochastischen Prozess, welcher modelliert werden muss und einer Regelung bedarf. DBNe bilden komplexe stochastische Prozesse ab, und PI vollführt Prognose, Diagnose und Entscheidungsunterstützung über diesen DBNen und regelt somit den komplexen stochastischen Prozess.

- Die Produktion und die PPS weisen eine hohe Komplexität auf, welche reduziert und/oder bewältigt werden will. Eigenschaften und Fähigkeiten der Wissensverarbeitung im Kontext BNe bewältigen die Komplexität der Produktion und der PPS.

- Nicht zuletzt sind die meisten Teilaufgaben der PPS von Strukturdefekten geprägt und stellen somit schlechtstrukturierte Probleme dar, von denen erstere behoben und letztere gelöst werden müssen. Methoden der Wissensverarbeitung im Kontext BNe merzen Strukturdefekte aus und lösen schlechtstrukturierte Probleme der PPS.

Die Wissensverarbeitung im Kontext BNe scheint ergo geeignet, alle Probleme der PPS sehr gut und umfassend zu lösen.

Darüber, wie sich BNe gegenüber anderen universellen und zur PPS eingesetzten Verfahren positionieren, gibt Tabelle 3.1 einen Überblick.

Kapitel 6

Ein Softwaresystem zur Unterstützung der PPS mittels BNe

6.1 Motivation zum Erstellen des Softwaresystems

Als mit der empirischen Untersuchung des Potenzials BNe zur PPS begonnen werden sollte, ist nicht geplant gewesen, eine Software zur Wissensverarbeitung im Kontext BNe zu erstellen, sondern es sind die folgenden acht grundlegenden Anforderungen zusammengestellt worden, welche eine solche Software zur Unterstützung der PPS zwingend erfüllen muss. Eine solche Software sollte:

1. eine (Programmier)schnittstelle aufweisen, die es erlaubt, das Softwaresystem in PPS-Systeme zu integrieren,

2. in der Lage sein, BNe automatisch aus Daten zu erstellen, wie es Abschnitt 4.2 beschreibt,

3. automatische PI über BNen vollziehen können, wie es Abschnitt 4.3 darlegt,

4. sowohl automatische PI als auch Wissensakquisition im Kontext BNe in *Stapelverarbeitung* vollziehen können, damit mehrere Experimente sequentiell und *ohne* manuellen Eingriff durch den Experimentator vollzogen werden können,

5. sowohl bei der Wissensakquisition als auch bei der automatischen PI fehlende Werte so behandeln, wie es Abschnitt 4.5.6 darstellt,

6. kontinuierliche Größen, die in der PPS auftreten, diskretisieren, wie es Abschnitt 4.4.3 vorgibt,

7. es erlauben, der automatischen Wissensakquisition Expertenwissen vorzugeben *und* das automatisch akquirierte BN nachträglich aufgrund von Expertenwissen zu modifizieren, wie es Abschnitt 4.5.5 gebietet, und

8. eine graphische Benutzerschnittstelle aufweisen, die das Interpretieren BNe erlaubt oder zumindest vereinfacht und somit ihre Plausibilität und ihre Akzeptanz erhöht (siehe Abschnitt 4.5.8).

Folgende fünf Softwaresysteme zur Wissensverarbeitung im Kontext BNe sind identifiziert worden und in eine engere Auswahl gelangt:

1. Hugin^{TM} 5.2, eine Software der dänischen Firma HUGIN Expert[1] A/S,

2. MSBN^{TM}, eine Software der Decision Theory Group der Microsoft Corporation[2],

3. Bayesware Discoverer^{TM}1.0, eine Software der Firma Bayesware[3] Ltd.,

4. BayesNet Toolbox[4], ein Softwarepaket, das Kevin Murphy an der Universität von Berkeley entwickelt hat, und

5. TETRAD[5] II [SSGM94], ein kommandozeilenorientiertes Programm, das von den Psychologen Peter Spirtes, Richard Scheines, Christopher Meek und Clark Glymour entwickelt worden ist.

Tabelle 6.1 stellt stark vereinfacht dar, wie gut die aufgeführten Softwaresysteme die gestellten Anforderungen erfüllen.[6] Da sämtliche Anforderun-

[1] `http://www.hugin.dk/`, Abrufzeitpunkt: 18.04.2006, 15:56 Uhr

[2] `http://research.microsoft.com/adapt/MSBNx/`, Abrufzeitpunkt: 18.04.2006, 16:00 Uhr

[3] `http://bayesware.com`, Abrufzeitpunkt: 18.04.2006, 15:55 Uhr

[4] `http://bnt.sourceforge.net/`, Abrufzeitpunkt: 18.04.2006, 15:59 Uhr

[5] `http://www.phil.cmu.edu/projects/tetrad/`, Abrufzeitpunkt: 18.04.2006, 15:57 Uhr

[6] Es ist anzumerken, dass sich die Aussagen in der Tabelle auf den Entwicklungsstand der aufgeführten Softwaresysteme zum Ende des Jahres 1998 beziehen und alle bis auf den Bayesware Discoverer weiterentwickelt worden sind. Die Fähigkeiten heutiger Softwaresysteme zur Wissensverarbeitung im Kontext BNe bzw. die Fähigkeiten dieser Softwaresy-

Tabelle 6.1: Vor- und Nachteile verfügbarer Softwaresysteme

Anforderung	Softwaresystem				
	1.) Hugin^{TM} 5.2	2.) MSBN^{TM} 1.0	3.) Bayesware Discoverer^{TM} 1.0	4.) BayesNet Toolbox	5.) TETRAD II
1.) (Programmier-) schnittstelle	++	+	−−	0	−−
2.) Automatisches Erstellen BNe	−−	−−	++	+	+
3.) Probabilistische Inferenz	++	+	−	−	−−
4.) Stapel-verarbeitung	−−	−−	−	+	−−
5.) Behandeln fehlender Werte	−−	−−	+	−−	−−
6.) Diskretisieren reellwertiger Variablen	−−	−−	+	−−	−−
7.) Integrieren von Expertenwissen	0	0	0	0	++
8.) Graphische Benutzerschnittstelle	++	++	+	−−	−−
Spalten-minimum	−−	−−	−−	−−	−−

Legende: ++ sehr gut, + gut, 0 neutral, − schlecht, −− sehr schlecht

gen unbedingt erfüllt werden müssen, gibt das Spaltenminimum (letzte Zeile) darüber Auskunft, ob das betrachtete Softwaresystem sämtlichen Anforderungen genügt. Wenn dem so sein sollte, müsste das Spaltenminimum mindestens 0 betragen. Da dass Spaltenminimum für jede Spalte jedoch „−−“ („Anforderungen faktisch nicht erfüllt“) beträgt, ist keines der verfügbaren/untersuchten Softwaresysteme zur Unterstützung der PPS und für eine empirische Untersuchung geeignet. Aus diesem Grund ist entschieden worden, eine neue Software zur Wissensverarbeitung im Kontext BNe zur Unterstützung der PPS zu erstellen.

Es ist nun überflüssig, jede Tabellenzelle separat zu diskutieren. Stattdessen soll nur jeweils *eine* der Zellen diskutiert werden, die zum *jeweiligen* Spaltenminimum geführt haben:

1. Hugin hat keine Möglichkeit geboten, BNe automatisch aus Daten zu generieren (Anforderung 2).

2. Das MSBN hat ebenfalls keine Möglichkeit geboten, BNe automatisch aus Daten zu generieren (Anforderung 2).

3. Der Bayesware Discoverer hat – obwohl angekündigt – keine Programmierschnittstelle geboten (Anforderung 1), die man auch zur Stapelverarbeitung hätte verwenden können.

4. Die BayesNet Toolbox hat keine grafische Benutzerschnittstelle (Anforderung 8) aufgewiesen.

5. TETRAD II hat ebenfalls keine grafische Benutzerschnittstelle aufgewiesen (Anforderung 8).

Es ist anzumerken, dass die hier analysierten Softwaresysteme z. T. von mehreren Personen über mehrere Jahre hinweg entwickelt worden sind und die bestehenden Softwaresysteme in ähnlichem Umfang weiterentwickelt werden. Im Gegensatz dazu ist das Softwaresystem, das in diesem Kapitel vorgestellt wird, von einer Person in Teilzeit entwickelt worden. So hat die Entwicklung erstgenannter Softwaresysteme die des letztgenannten z. T. ein- oder sogar überholt. Immer dann, wenn eine neue und *benötigte* Funktionalität und/oder Datenstruktur in die in diesem Kapitel vorgestellte Software eingebaut worden ist, ist für das eine oder andere (kommerzielle) Softwaresystem bekanntgegeben worden, dass diese Funktionalität/Datenstruktur

steme zum jetzigen Zeitpunkt sind für den hiesigen Abschnitt allerdings irrelevant, da die Entscheidung, ein Softwaresystem zur Unterstützung der PPS mittels BNe zu erstellen, damals und basierend auf dem damaligen Entwicklungsstand gefällt worden ist. Der Autor hat das besagte Softwaresystem entwickelt, und es liegt heute vor.

ebenfalls zur Verfügung gestellt worden ist. Ein Beispiel stammt aus dem „Hugin Newsletter“ vom März des Jahres 2004: „The Hugin Graphical User Interface now includes a tool for analysing dependence and independence relations between nodes given evidence (d-separation).“ Der Newsletter ist genau neun Tage, nachdem die *d*-Separatation zur Eliminierung irrelevanter Variablen in die hier vorgestellte Software eingebaut worden ist, erschienen, was durchaus demotivierend gewesen ist. – Wenn eine solche Entwicklung 1998 vorausgesehen worden wäre, hätte man versucht, die unzulänglichen Softwaresysteme mit pragmatischen Mitteln zu koppeln und zu verwenden. Die Aufwandsersparnis hätte die Nachteile einer solchen Lösung vermutlich mehr als aufgewogen. Nichtsdestotrotz liegt nun ein konsistentes Softwaresystem zur Unterstützung der PPS durch BNe vor, welches die in diesem Abschnitt aufgeführten Anforderungen und noch weitere erfüllt.

6.2 Überblick über Architektur und Funktionalität des Softwaresystems

Abbildung 6.1 stellt die Komponentenarchitektur des Softwaresystems dar. Das System erstellt das BN und wendet es an. Während der Akquisition wird das BN erstellt. Zur Akquisition stellt das Datenmanagement der Akquisitionskomponente Vergangenheitsdaten aus der Produktion zur Verfügung, und der Nutzer versorgt die Akquisitionskomponente via Benutzerschnittstelle mit zusätzlichem Expertenwissen, sofern solches verfügbar ist. Die Akquisitionskomponente erstellt aus den Daten und dem Expertenwissen automatisch das BN und übergibt es der Repräsentationskomponente. Während der Anwendung des BNes wird ein schlechtstrukturiertes Problem der PPS gelöst. Zur Anwendung stellt die Repräsentations- der Anwendungskomponente das bereits erstellte BN zur Verfügung. Danach liefert das Datenmanagement der Anwendungskomponente Zustandsdaten aus der Produktion und der Anwender gibt der Anwendungskomponente via Benutzerschnittstelle Zielausprägungen bzw. Führungsgrößen vor. Die Anwendungskomponente errechnet aus den Zustandsdaten und den Zielvorgaben über dem BN eine Lösung, welche sie dem Anwender über die Benutzerschnittstelle präsentiert und – sofern vom Anwender bestätigt – via Datenmanagement an die Produktion übergibt. Die Anwendung des BNes läuft immer dann ab, wenn eine Entscheidung im Rahmen der PPS ansteht. Sie kann zeit- oder ereignisgesteuert erfolgen. Die Akquisition des BNes wird in Abhängigkeit von Änderungen in der Produktion ebenfalls zeit- oder ereignisgesteuert wiederholt und kann partiell erfolgen.

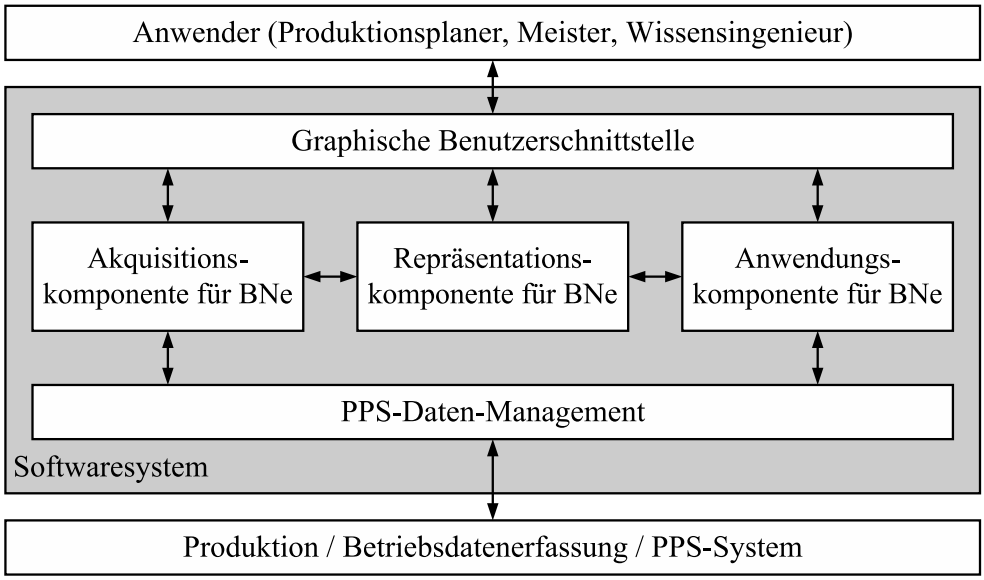

Abbildung 6.1: Komponentenarchitektur des Softwaresystems zur Unterstützung der PPS mittels BNe

Das erstellte Softwaresystem zur Unterstützung der PPS mittels BNe weist nicht die Nachteile anderer Systeme (siehe Tabelle 6.1) auf:

1. Es stellt mit einer Dynamic Link Library (DLL) eine definierte Schnittstelle zur Verfügung, die es gängigen ERP-Systemen ermöglicht, das System anzusprechen. Die DLL veröffentlicht alle wichtigen Routinen der Akquisitions-, Repräsentations- und Anwendungskomponente, das heißt zum Editieren des BNes aufgrund von Expertenwissen über PPS und Produktion, zum automatischen Erstellen und Aktualisieren der Schätzfunktionen des Netzes aus PPS-Daten und zur Anwendung des Netzes zu Prognose, Diagnose und Entscheidungsunterstützung in der PPS.

2. Zu Aufbau, Repräsentation und Anwendung der Inferenzmaschine wurden Algorithmen und Datenstrukturen entwickelt, die es erlauben, auch zur Lösung von PPS-Problemen praxisrelevanter Größenordnung die PI speicher- und laufzeiteffizient zu vollführen und zu protokollieren.

3. Das Softwaresytem kann sowohl automatische PI als auch Wissensakquisition im Kontext BNe in Stapelverarbeitung vollziehen und erlaubt so, mehrere Experimente sequentiell und *ohne* manuellen Eingriff durch den Experimentator durchzuführen.

4. Zwei Verfahren zur Behandlung fehlender Werte beim Erstellen BNe machen das System besonders für die PPS praktikabel, da sie oft unvollständiges Datenmaterial aufweist; und die eingesetzten Inferenzalgorithmen erlauben fehlende Eingabewerte per se.

5. Zwei Verfahren zum Diskretisieren kontinuierlicher Größen, die in der PPS auftreten, ermöglichen die Integration reellwertiger Variablen bzw. die Eingabe reeller Werte sowohl beim Erstellen BNe als auch bei ihrer Anwendung.

6. Das Softwaresystem erlaubt es, der automatischen Wissensakquisition Expertenwissen in Form eines vorab manuell erstellten BNes oder in Form von Strukturrestriktionen vorzugeben *und* das automatisch akquirierte BN nachträglich aufgrund von Expertenwissen zu modifizieren.

7. Besonderes Augenmerk wurde auf ein intuitiv zu bedienendes GUI mit Multiple Document Interface (MDI) gelegt. Via GUI können BNe dem Anwender auch als Erklärungsmodell dienen. Außerdem wurden zwei Algorithmen zum Graphenzeichnen eingebaut, welche BNe bzw. deren Graphen automatisch übersichtlich darstellen. Die GUI erlaubt und vereinfacht das Interpretieren BNe und erhöht somit ihre Plausibilität und ihre Akzeptanz.

Darüber hinaus bietet das System die Möglichkeit, dynamische BNe zu erstellen und anzuwenden, die sich aufgrund ihrer Eigenschaften und Fähigkeiten besonders gut dazu eignen, Produktionsprozesse abzubilden und zu regeln [MVD01].

Das System ist in der Programmiersprache Object Pascal unter Borland Delphi für das Betriebssystem Microsoft Windows entwickelt worden und kann durch den Einsatz eines Cross-Platform-Compilers auch für das Betriebssystem Linux kompiliert werden.

6.3 Vorgehensmodell zur Anwendung des Softwaresystems zur PPS

Hat sich Abschnitt 6.2 eher mit dem Aufbau und den Fähigkeiten des Softwaresystems befasst, so wird nun ein grobes Vorgehensmodell zur Anwendung des Softwaresystems zur Lösung einer Teilaufgabe der PPS entworfen. Das Vorgehensmodell nimmt Bezug auf Abbildung 6.1 und die in ihr aufgeführten Komponenten und Beteiligten:

1. *automatische Wissensakquisition*: Soll die Akquisitionskomponente das BN automatisch aus Daten akquirieren, so stellt ihr das PPS-Daten-Management Vergangenheitsdaten in Form von Datensätzen zur Verfügung. Die Vergangenheitsdaten liegen entweder bereits als Fälle im PPS-System vor, oder sie müssen extra vom Produktionsprozess abgetastet werden. Soll das Softwaresystem bzw. das BN zur Entscheidungsunterstützung eingesetzt werden, so müssen die Datensätze explizit Zielfunktionswerte und Handlungsalternativen enthalten. Die automatische Wissensakquisition muss nicht vom Anwender, sondern kann zeit- oder ereignisgesteuert vom PPS-Daten-Management angestoßen werden.

2. *manuelle Wissensakquisition*: Soll das BN aus Expertenwissen erstellt werden, so stellt ein Experte – ggf. unterstützt durch einen „Wissensingenieur“ – das Wissen der Repräsentationskomponente via grafische Benutzerschnittstelle zur Verfügung. Das Wissen umfasst die für die Teilaufgabe der PPS relevanten Variablen, direkte kausale Abhängigkeiten zwischen den Variablen sowie die bedingten Wahrscheinlichkeiten, die für den Aufbau des BNes vonnöten sind.

3. *Wissensrepräsentation*: Über die grafische Benutzerschnittstelle erhält der Experte bzw. der Wissensingenieur eine Rückkopplung über das BN, das aus Daten und/oder Expertenwissen erstellt worden ist, und kann ggf. nachträglich Änderungen am BN vornehmen oder die automatische Akquisition des BNes aus Daten unter vorheriger Eingabe weiteren und/oder alternativen Expertenwissens wiederholen.

4. *Erstellen der Inferenzmaschine*: Ist das BN erstellt worden, so generiert die Anwendungskomponente eine Inferenzmaschine über dem BN. Das Erstellen der Inferenzmaschine kann implizit nach dem Aufbau oder einer Änderung des BNes erfolgen oder durch den Experten bzw. den Wissensingenieur via grafische Benutzerschnittstelle angestoßen werden.

5. *Eingabe von Evidenz*: Nachdem die Inferenzmaschine generiert worden ist, erhält die Anwendungskomponente Evidenz (siehe Definition 4.7) in Form des Zustandes der Produktion, erwünschter Zielfunktionswerte oder erfolgversprechender Handlungsalternativen[7], und die Inferenzmaschine absorbiert die Evidenz (siehe Anhang A.3.7). Die Evidenz

[7]Für Entscheidungsunterstützung müssen erwünschte Zielfunktionswerte und der gegenwärtige Zustand der Produktion angegeben werden. Für Prognose und Diagnose genügt der gegenwärtige Zustand der Produktion

stellen das PPS-System oder die Produktion selbst via PPS-Daten-Management und/oder der Anwender via grafische Benutzeroberfläche und Repräsentationskomponente zur Verfügung.

6. *Wissensanwendung*: Die Anwendungskomponente vollzieht nun die Anwendung des BNes bzw. der Inferenzmaschine und integriert Randverteilungen aus der Inferenzmaschine heraus oder bestimmt die wahrscheinlichste Konfiguration (siehe Anhang A.3.9).[8] Die Wissensanwendung kann vom Anwender via grafische Benutzeroberfläche sowie zeit- oder ereignisgesteuert vom PPS-Daten-Management angestoßen werden.

7. *Ergebnisanzeige*: Die Ergebnisse der Wissensanwendung zeigt die Anwendungskomponente über die Repräsentationskomponente und die grafische Benutzeroberfläche dem Anwender an. Je nach Wunsch zeigt sie dem Anwender die wahrscheinlichste Konfiguration oder die Randverteilungen der gewünschten Variablen an. Die Ergebnisanzeige muss bei Stapelverarbeitung bzw. im Automatikbetrieb nicht zwangsläufig erfolgen.

8. *Ergebnisübergabe*: Sind die Ergebnisse der Wissensanwendung dem Anwender angezeigt worden, so kann er sie über die grafische Benutzeroberfläche, die Repräsentationskomponente, die Anwendungskomponente und das PPS-Daten-Management an die Produktion übergeben. Die Ergebnisübergabe kann auch erfolgen, ohne dass der Anwender einbezogen wird, indem die Anwendungskomponente die ermittelten Ergebnisse der Produktion via PPS-Daten-Management direkt nach der Wissensanwendung übergibt. Die Ergebnisübergabe ist meist nur bei der Entscheidungsunterstützung bzw. Regelung sinnvoll, bei der die beste Handlungsalternative bzw. Ausprägungen der Stellgrößen der Produktion zum Umsetzen übergeben werden.

Das dargelegte Vorgehensmodell kann auf das PPS-Daten-Management und die unter ihm liegende Produktion verzichten, indem ausschließlich der Anwender mit der Repräsentations- und der Anwendungskomponente interagiert. Das Vorgehensmodell lässt auch einen Betrieb des Softwaresystems

[8] Für welche Variablen die Konfiguration oder die Randverteilungen ermittelt werden, hängt davon ab, ob Prognose, Diagnose oder Entscheidungsunterstützung vollzogen werden und kann vom Anwender über die grafische Benutzeroberfläche vorgegeben werden. Bei der Entscheidungsunterstützung handelt es sich um die Konfiguration der Variablen, welche die Handlungsalternativen kodieren, und bei der Prognose und der Diagnose um Randverteilungen oder Konfigurationen der Variablen, die zukünftige oder vergangene Zustände repräsentieren.

ohne (Interaktion mit dem) Anwender und ohne grafische Benutzeroberfläche zu, indem sowohl die Akquisitionskomponente als auch die Anwendungskomponente ausschließlich von der Datenverwaltungskomponente mit Daten versorgt und gesteuert werden.

Wie bereits angemerkt, können sämtliche Schritte im Vorgehensmodell manuell oder automatisch erfolgen und ereignis- oder zeitgesteuert vom Anwender oder vom PPS-Daten-Management angestoßen werden.

6.4 Komponenten des Softwaresystems zur Wissensverarbeitung im Kontext BNe

6.4.1 Komponente zur Repräsentation BNe

Da die physischen Repräsentationen eines BNes und seiner Inferenzmaschine auf denselben Datenstrukturen beruhen, stellt sie dieser Abschnitt gemeinsam dar. Die physischen Repräsentationen basieren auf den logischen Repräsentationen aus Kapitel 4 und aus Anhang siehe A.1.

Ein BN und seine Inferenzmaschine setzen sich aus speziellen Graphen und speziellen n-dimensionalen, diskreten, reellwertigen Funktionen[9] zusammen. Abbildung B.1 zeigt das UML-Klassendiagramm der Graphen und der Funktionen, und Anhang B.1.1 erläutert das Klassendiagramm.

Die Komponente zur Repräsentation BNe sieht zwei verschiedene Datenstrukturen zur Repräsentation von Potenzial- und Bewertungsfunktionen (siehe Definition A.5) vor: Die *Vektorfunktion* beinhaltet lediglich einen „$\mathbb{R}$-Vektor“, einen eindimensionalen Vektor reeller Zahlen $\mathbb{R}$. Die *Baumfunktion* hingegen hält einen abstrakten Vektor, der im Spezialfall einer eindimensionalen Funktion ein $\mathbb{R}$-Vektor ist und sonst ein Vektor von Zeigern auf Vektoren, die wiederum Zeigervektoren oder $\mathbb{R}$-Vektoren sein können. [GHJV04, S. 239 ff.] bezeichnet eine Struktur wie die Baumfunktion als Kompositum, das dazu dient, einzelne und zusammengesetzte Teile in Hierarchien einheitlich zu behandeln. Zum Verdeutlichen der alternativen Funktionsrepräsentationen stellt Abbildung 6.2 die resultierende Potenzialfunktion aus Tabelle 4.24 einmal als Vektorfunktion und einmal als Baumfunktion dar.

Die Vektorfunktion bestimmt mittels Adressrechnung den Funktionswert für eine Kombination von Argumenten. Die Adressrechnung verursacht relativ hohen Rechenaufwand, da sie multipliziert. Die alternative Baumfunktion hingegen bestimmt den Funktionswert für eine Kombination von Argumenten durch Dereferenzieren, das wesentlich geringeren Rechenaufwand verursacht

[9] siehe Anhang A.1.1

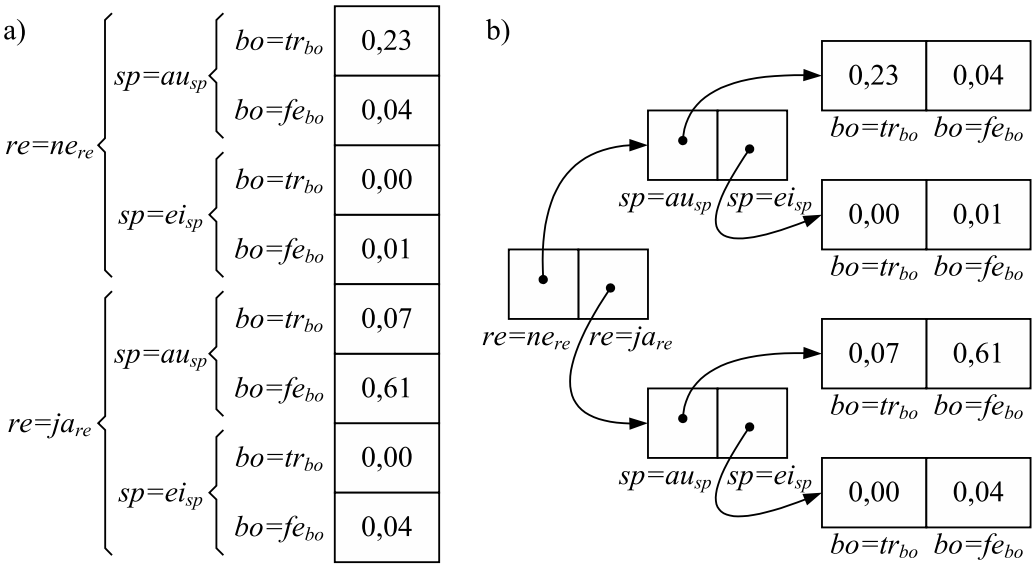

Abbildung 6.2: Die resultierende Potenzialfunktion aus Tabelle 4.24 alternativ repräsentiert als a) Vektorfunktion und als b) Baumfunktion

als Multiplizieren, was für den Einsatz der Baumfunktion spricht.

Anhang A.1.1 entsprechend, seien $|S_1|, \ldots, |S_k|$ die Mächtigkeiten der Zustandsmengen k diskreter Variablen. Zur Repräsentation einer Potenzialfunktion über diesen Variablen benötigt eine Vektorfunktion lediglich Speicherplatz für $\prod_{i=1}^{k} |S_i|$ reelle Zahlen. Eine Baumfunktion benötigt zur Repräsentation derselben Potenzialfunktion hingegen zusätzlich Speicherplatz für $z_k = \sum_{i=2}^{k} \prod_{j=i}^{k} |S_j|$ Zeiger[10], der mit der Anzahl der Variablen exponentiell wächst, was gegen den Einsatz der Baumfunktion und für den Einsatz der Vektorfunktion spricht. Da Hauptspeicher bei der im Softwaresystem eingesetzten PI den kritischen Faktor darstellt und eine Möglichkeit besteht, über Vektorfunktionen effizient Adressrechnung zu vollziehen, wird die Baumfunktion als Repräsentationsform für Potenzialfunktionen nicht weiter verfolgt. Wie Adressrechnung über Potenzialfunktionen effizient vollzogen wird, beschreibt Abschnitt 6.4.3.

Um einmal erstelle BNe dauerhaft zu speichern, legt eine Subkomponente sie in einem XML-Austauschformat für BNe ab. Selbige Komponente lädt die BNe auch wieder aus XML-Dateien dieses Formates in den Hauptspeicher. Eine Document Type Definition (DTD) namens `xbn.dtd`[11] beschreibt das

[10] Die explizite Berechnungsvorschrift ist aus ihrer rekursiven Darstellung, $z_1 = 0$ und $z_i = |S_i|(z_{i-1} + 1)$, entwickelt worden.

[11] `http://research.microsoft.com/dtas/bnformat/xbn_dtd.html`, abgerufen am 31.08.2007 11.15 Uhr

XML-Austauschformat für BNe [Dec07], das die Decision Theory & Adaptive Systems Group von Microsoft Research entworfen hat. Zum Erstellen der XML-Dokumente aus BNen, zu ihrem Speichern, ihrem Parsen und zu ihrem Validieren gegen die DTD, dient der XML-Parser OpenXML[12] von Dieter Köhler in der stabilen Version 2.4.0.

Die Datenstruktur für ein DBN ist lediglich eine Hülle für drei BNe, das obligatorische Eingangsnetz, das obligatorische Übergangsnetz und das fakultative entrollte DBN. Die Datenstruktur für ein DBN bietet einzig die Methode zum Entrollen des DBNes nach Definition 4.5 an. Das Austauschformat für BNe erlaubt es, mehrere BNe in einem XML-Dokument zu speichern. Die Subkomponente zum Speichern und Laden BNe beutet diese Möglichkeit dahingehend aus, dass sie die drei Komponenten DBNe, Eingangs-, Übergangs- und entrolltes Netz, gemeinsam in einem XML-Dokument speichert.

Das Erstellen von Inferenzmaschinen, die möglichst wenig Hauptspeicher benötigen und geringe Laufzeiten bei der PI aufweisen, verursacht sehr hohen Rechenaufwand. [CDLS99, S. 58] Aus diesem Grunde scheint es geboten, Inferenzmaschinen ebenfalls dauerhaft zu speichern und sie beim Neustart des Softwaresystems nicht neu zu erstellen, sondern sie lediglich zu laden. Die Software Hugin^{TM} (siehe Tabelle 6.1) bietet diese Option an. Da das vorliegende Softwaresystem Inferenzmaschinen mittels einer effizienten Heuristik aufbaut und sie während der empirischen Studie in Abschnitt 7 nur jeweils einmal erstellt, ist vorerst davon abgesehen worden, Inferenzmaschinen zu persistieren.

Zur Repräsentation BNe gehört ebenfalls die Möglichkeit, sie zu modifizieren. Das Modifizieren BNe ist dringend vonnöten, da insbesondere das manuelle Erstellen BNe evolutionär abläuft (siehe Abbildung 4.4) und dem Anwender leichtfallen sollte. Den Graphen und die Variablen des BNes algorithmisch zu modifizieren, fällt verhältnismäßig leicht. Kompliziert gestaltet sich jedoch das Modifizieren, wenn bereits Bewertungsfunktionen für die Knoten des BNes vorhanden bzw. gefüllt worden sind und das BN auch nach der Modifikation noch einen konsistenten und erwartungskonformen Zustand aufweisen soll. Anhang B.1.2 beschreibt Operationen, welche die Bewertungsfunktionen modifizieren, abstrakt und anhand zweier Beispiele.

Das Softwaresystem ermöglicht über ein bloßes Modifizieren hinaus ein mehrstufiges Rückgängigmachen für reversible und irreversible Operationen beim Editieren BNe, indem es jede dieser Operationen und Zusatzinformation über den vorherigen Zustand nach dem Entwurfsmuster „Befehl“ [GHJV04, S. 273 ff.] in einer Befehlsklasse kapselt und beim Editieren eines BNes Instanzen dieser Klassen zur sukzessiven Bearbeitung für den Fall des Rück-

[12] `http://www.philo.de/xml/`, abgerufen am 02.03.2007 14:05 Uhr

gängigmachens aneinanderreiht.

Nachdem nun die Repräsentation und Verwaltung BNe im Softwaresystem vorgestellt worden ist, beschreibt der folgende Abschnitt, wie das Softwaresystem die (automatische) Akquisition BNe (aus Daten) vornimmt.

6.4.2 Komponente zur Akquisition BNe[13]

Zum Erstellen BNe sind zwei alternative Gütemaße (siehe Formeln 4.14 und 4.15) sowie drei Suchalgorithmen implementiert und in dem Softwaresystem zur Unterstützung der PPS mittels BNe verankert worden. Zusätzlich zu den gebräuchlichen Suchalgorithmen K2 und B (siehe Algorithmen A.1 und A.2) ist ein Rejection-Free Annealing [GS86] implementiert worden, wie es sich laut [Bou94, S. 103 ff.] empfiehlt. Jeder Suchalgorithmus kann mit jedem Gütemaß kombiniert werden, und sowohl neue Suchalgorithmen als auch neue Gütemaße können problemlos in das System integriert werden. Zur Berechnung der bedingten Wahrscheinlichkeiten ist ausschließlich Formel 4.19 verwendet worden.

Da Rejection-Free Annealing im Rahmen der vorliegenden Arbeit noch nicht vorgestellt worden ist, folgen an dieser Stelle ein paar kurze Anmerkungen zu ihm: Rejection-Free Annealing [GS86] ist ein universeller Algorithmus zur kombinatorischen Optimierung, der als eine Weiterentwicklung des Simulated Annealing angesehen wird. Es bzw. er ist ausgewählt und in der vorliegenden Software umgesetzt worden, um nicht nur „gierige“ Suchalgorithmen zu verwenden, die den „Einzugsbereich“ eines Optimums nicht verlassen können und bei denen somit große Gefahr besteht, sich bei Optimierungsproblemen, die Zielfunktionen mit mehreren Extrema aufweisen, in einem lokalen Optimum zu verfangen. Im Gegensatz zum Simulated Annealing vermeidet Rejection-Free Annealing, Lösungen in Betracht zu ziehen, die es dann doch wieder verwirft, was zu Laufzeitvorteilen führt. Zudem benötigt Rejection-Free Annealing außer einem Abbruchkriterium keine weiteren Parameter. Beim Rejection-Free Annealing handelt es sich um ein abgewandeltes Verfahren der lokalen Suche, das Abschnitt 3.1 beschreibt, so dass hier nicht näher auf den Algorithmus eingegangen werden soll.

Für die vorliegende Komponente zur Akquisition BNe sind zwei Abbruchkriterien entwickelt worden: Das erste bricht nach einer gegebenen und konstanten Anzahl von Schritten ab, das zweite, wenn sich nach einer gegebenen und konstanten Anzahl von Schritten keine Verbesserung ergeben hat.

Die Schwierigkeit beim Erstellen der Komponente zur Akquistion BNe

[13]Die Komponente zur (automatischen) Akquisition BNe ist zu großen Teilen von Herrn Torsten Hildebrandt implementiert worden, wofür ihm herzlicher Dank gilt.

aus Daten hat nicht in den Suchalgorithmen und den Modifikationen des Graphen des BNes bestanden, sondern im Speichern der bedingten Häufigkeiten und dem Errechnen der Werte für die Gütemaße für Knotenfamilien (siehe Definition A.20) aus diesen bedingten Häufigkeiten. Aus diesem Grunde geht Anhang B.2.1 auf die drei alternativen Datenstrukturen ein, welche zum schnellen Speichern der bedingten Häufigkeiten und zum schnellen Zugriff auf die bedingten Häufigkeiten beim Errechnen der Gütemaße entwickelt worden sind.

Von großem Interesse bei einer Komponente zur Akquisition BNe sind die Möglichkeiten, die Struktur des automatisch zu erstellenden BNes vorab partiell vorzugeben und/oder einzuschränken. Die vorliegende Komponente bietet die folgenden:

- Die maximal zulässige Anzahl von Eltern kann pro Knoten oder global für alle Knoten vorgegeben werden. So wird der Rechenaufwand für die PI über dem resultierenden BN verringert.

- Kanten dürfen explizit verboten oder erzwungen werden, was es erlaubt, Basisstrukturen vorab festzulegen.

- Knoten können als reine Quellknoten ausgewiesen werden, indem man die maximal zulässige Anzahl von Eltern für sie auf 0 setzt.

- Knoten können als reine Zielknoten ausgewiesen werden, indem man Kanten von ihnen zu allen anderen Knoten verbietet.

- Die Richtung einer potenziellen Kante kann vorab festgelegt werden, indem man die Kante in die entgegengesetzte Richtung vorab verbietet.

Mittels der hier vorgestellten Möglichkeiten kann a priori Expertenwissen über kausale Zusammenhänge zwischen den Variablen des Problemraums der PPS *vollständig* in das BN eingebracht werden.

Die Komponente zur Akquisition BNe weist mehrere fachliche und softwaretechnische Vorteile auf:

- Die Komponente zur Akquisition BNe erstellt nicht nur einfache BNe, sondern auf Wunsch auch Eingangs- und Übergangsnetze für DBNe. Das „Entrollen“ dieser DBNe nimmt allerdings die Komponente zur Anwendung BNe (siehe Abschnitt 6.4.3) vor.

- Die Suchalgorithmen der Komponente zur Akquisition BNe sind so gestaltet worden, dass sie durch eine Benutzerinteraktion abgebrochen werden können, auch wenn ihr Abbruchkriterium noch nicht erfüllt worden ist.

- Die Suchalgorithmen der Komponente zur Akquisition BNe weisen die „anytime“-Eigenschaft auf, die bedeutet, dass die Suchalgorithmen, sofern sie sich bereits in der Iterationsphase befinden, jederzeit die bis dato beste gefundene Lösung (das beste gefundene BN) kennen und in der Lage sind, sie bzw. es auszugeben.

- Die Komponente zur Akquisition BNe ist dynamisch um neue Gütemaße, Suchalgorithmen und sonstige Bestandteile, wie z. B. *Häufigkeitenstrukturen*, erweiterbar. Da diese Bestandteile oft spezifische Parameter aufweisen, enthält die Komponente zur Akquisition BNe einen Mechanismus, der es erlaubt, zu den Bestandteilen passende Parameter(hierarchien) zu definieren, die sich selbst auf ihre Bestandteile anwenden. Vom Hinzufügen weiterer Bestandteile und ihrer Parameter bleibt vorhandener Quelltext unberührt.

- Um eine einfache und gute Erweiterbarkeit der Komponente zur Akquisition BNe zu gewährleisten, verfügt sie über eine Registratur für ihre und neue Bestandteile, wie z. B. Gütemaße, Suchalgorithmen und *Häufigkeitenstrukturen*. Sind neue Bestandteile entwickelt worden, müssen sie sich lediglich registrieren und stehen nach einem erneuten Kompilieren der Anwendung zur Verfügung.

- Die Komponente zur Akquisition BNe behandelt fehlende Werte. Allerdings behandelt sie fehlende Werte derzeit nur auf zwei einfache Arten: Wenn ein Datensatz nicht für alle Eltern der betrachteten Familie Werte enthält, wird er entweder ignoriert, oder für die von fehlenden Werten betroffenen Variablen wird jeweils eine weitere Ausprägung angelegt.

- Die Komponente zur Akquisition BNe kann nicht nur via Hauptspeicher mit Datensätzen versorgt werden, sondern verfügt auch über eine Schnittstelle zum Einlesen des Attribute-Relation File Format[14] (ARFF) [Wek07].

- Die Komponente zur Akquisition BNe stellt ermittelte BNe nicht nur via Hauptspeicher zur Verfügung, sondern auch im Interchange Format for Bayesian Networks (XMLBIF) [Coz98], einem standardisierten Format zum Austausch BNe zwischen verschiedenen Programmen.

[14]Das ARFF ist ein Format zum strukturierten Speichern homogener Datensätze in (Form) einer Textdatei. Der Kopf einer ARFF-Datei beschreibt die Namen und die Typen der Felder der Datensätze, und der Rumpf der ARFF-Datei enthält die Datensätze, deren Felder Kommata separieren. Das ARFF erlaubt Kommentarzeilen.

- Die Komponente zur Akquisition BNe ist separat als Kommandozeilenprogramm lauffähig und relativ komfortabel über Kommandozeilenparameter steuerbar.

Über die angegebenen Vorteile hinaus bietet die Komponente zur Akquisition BNe geeignete Ansatzpunkte, die Suchalgorithmen zu parallelisieren oder zu verteilen. Da die Laufzeiten der Algorithmen aber bereits relativ gering sind – vor allem im Verhältnis zur mehrfach auszuführenden PI –, ist von einer Parallelisierung der Suchalgorithmen vorerst abgesehen worden.

6.4.3 Komponente zur Anwendung BNe

Das Anwenden BNe bzw. die PI über BNen verursacht im Rahmen der Wissensverarbeitung im Kontext BNe den mit Abstand größten Rechen- und Speicheraufwand. [CGH97, S. 393]) Im Rahmen der vorliegenden Arbeit ist es gelungen, Algorithmen und Datenstrukturen zu entwerfen und umzusetzen, die eine effiziente exakte PI durchführen. Diese Algorithmen und Datenstrukturen werden als ein wesentliches Ergebnis der Arbeit betrachtet. Sie ermöglichen es überhaupt erst, die Experimente aus Kapitel 7 in akzeptabler Zeit durchzuführen. Schließlich ist im Rahmen der Experimente vier Millionen Mal PI i. e. S. vollzogen worden. Nicht zuletzt machen diese Algorithmen und Datenstrukturen BNe für die PPS erst interessant, da eine zu langsame PI für die PPS inakzeptabel wäre.

Da die Algorithmen für das weitere Verständnis der Arbeit aber nicht von entscheidender Bedeutung sind, geht lediglich Anhang B.3 detailliert auf die entwickelten Algorithmen zur PI ein. Er beschreibt vier Algorithmen zur exakten PI, die im Rahmen der vorliegenden Arbeit entwickelt worden sind, en detail:

- Algorithmus B.5 zum *Multiplizieren (Dividieren)* zweier Potenzialfunktionen,
- Algorithmus B.6 zum *Absorbieren von Evidenz* durch eine Potenzialfunktion und
- Algorithmus B.7 zum *Herausintegrieren eines Summenrandes (Maximumrandes)* aus einer Potenzialfunktion sowie
- Algorithmus B.2 zum *Inkrementieren der Lese- und Schreibposition* einer Potenzialfunktion, der den drei erstgenannten Algorithmen zugrundeliegt.

Um die vier Algorithmen besser zu verstehen und ihre Effizienz besser zu erfassen, wird empfohlen, vorab zweierlei zu lesen: den Anhang B.3, der in Operationen über Potenzialfunktionen einführt, und den Anhang B.3.1, der auf die Adressrechnung in Vektoren eingeht.

An dieser Stelle sei nochmals betont, dass die entworfenen und im Softwaresystem implementierten Algorithmen und Datenstrukturen zur exakten PI über BNen als ein wesentliches Ergebnis der Arbeit angesehen werden, weil sie aufgrund ihrer Effizienz die Anwendung BNe zur Unterstüzung der PPS überhaupt erst ermöglichen.

6.5 PPS-spezifische Komponenten des Softwaresystems

6.5.1 Komponente zur Generierung von PPS-Daten

Um die Leistungsfähigkeit von Heuristiken zur Unterstützung der PPS zu untersuchen, bestehen zwei grundlegende Möglichkeiten:

1. die Analyse der Heuristiken auf Grundlage der Algorithmen- und Komplexitätstheorie sowie

2. die empirische Untersuchung der Heuristiken mit Hilfe statistischer Verfahren.

Erstere ist zum Teil nicht möglich und weist nur begrenzte Aussagekraft auf [Völ03, S. 102]; letztere hingegen ist universell anwendbar, und die Sicherheit ihrer Aussagen kann quantifiziert werden. Will man nun letztere Untersuchung durchführen, so geht man vor, wie in Abbildung 6.3 beschrieben: Auf mehrere Probleme bzw. auf n Instanzen dieser Probleme werden verschiedene Verfahren bzw. m Ausprägungen dieser Verfahren angewandt.[15] Eine Verfahrensausprägung wird mehrfach auf eine Probleminstanz angewandt, wenn das Problem und/oder das Verfahren mit Stochastik behaftet ist, so dass mehrere Ergebnisse aus einer Kombination von Probleminstanz und Verfahrensausprägung resultieren. Ein Ergebnis wird immer hinsichtlich Lösungsgüte und Rechenaufwand beurteilt, wobei sich der Rechenaufwand wiederum aus Speicherbedarf und Rechenzeit zusammensetzt. Alle Ergebnisse einer Kombination aus Probleminstanz und Verfahrensausprägung ergeben

[15] Bei den „Verfahrensausprägungen“ handelt es sich um Kombinationen von Parameterausprägungen, mit denen diese Verfahren parametrisiert und angewendet werden. Diese Verfahrensausprägungen werden auch als Konfigurationen bezeichnet. [Dör04, S. 7]

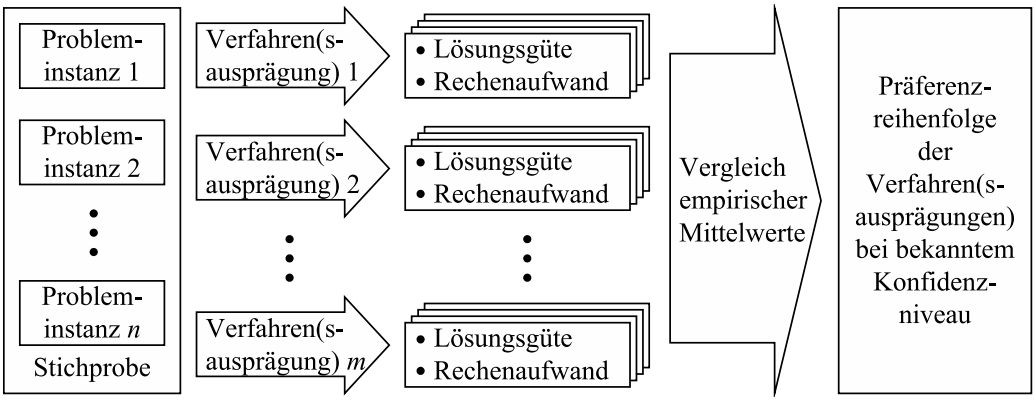

Abbildung 6.3: Vorgehensweise zum empirischen Vergleich alternativer Heuristiken; in Anlehnung an [VDM01]

nun empirische Verteilungen bzw. Mittelwerte bezüglich Lösungsgüte und Rechenaufwand, die es ermöglichen,

- die Verfahren miteinander zu vergleichen und somit
- die beste Verfahrensausprägung für eine konkrete Probleminstanz zu ermitteln sowie
- eine Präferenzreihenfolge für Verfahrensausprägungen zu bestimmen und
- Zusammenhänge zwischen Probleminstanzen, Verfahrensausprägungen, Lösungsgüte und Rechenaufwand mittels multivariater Regression herzustellen.

Die Verfahren zur Unterstützung der PPS, wie z. B. BNe, seien bekannt und mit ihren Parametern implementiert worden. Um diese Verfahren zu testen, müssen nun Probleminstanzen respektive PPS-Testdaten vorliegen, die folgende Eigenschaften aufweisen: Die Probleminstanzen sollten [VDM01]

1. vollständig,
2. korrekt,
3. von hinreichendem Umfange,
4. realitätsnah und
5. in ihren relevanten Eigenschaften vorab von extern determinierbar sein,

Tabelle 6.2: Erfüllungsgrad, den verschiedene Quellen für Probleminstanzen, die zum Testen von Heuristiken verwendet werden, hinsichtlich der Anforderungen, die an diese Probleminstanzen gestellt werden, aufweisen; in Anlehnung an [DMV00]

Anforderung an Probleminstanzen	Quellen für Probleminstanzen		
	Praxis	Bibliotheken	Generieren
1. vollständig	0	+	+
2. korrekt	0	+	+
3. Umfang	0	-	+
4. realitätsnah	+	-	0
5. determinierbar	-	-	+
6. Abdeckung	0	0	+
7. geringer Aufwand	0	+	0

Legende: Anforderung gut +, eingeschränkt 0, schlecht oder nicht − erfüllt

6. den relevanten Problemraum möglichst gut abdecken und

7. einen möglichst niedrigen Bereitstellungsaufwand aufweisen.

Als potenzielle Quellen für Testdaten bieten sich Probleminstanzen aus der Praxis, Bibliotheken mit Probleminstanzen (siehe z. B. [FPA+99] und [Bea96]) und das Generieren von Probleminstanzen (siehe z. B. [Hil98]) an. Wie die verschiedenen Quellen bzw. Probleminstanzen aus ihnen obige Anforderungen erfüllen, sagt Tabelle 6.2 aus.

Probleminstanzen aus der Praxis sind zwar realitätsnah, aber oft unvollständig und nicht korrekt. Sie weisen einen großen Umfang auf, der sich aber auf einige wenige Problemdimensionen beschränkt. Sie sind nicht vorab von extern determinierbar und decken den relevanten Problemraum nur punktuell ab. Der Aufwand, sie zu erlangen und in eine für Tests verwendbare Form zu transformieren, ist mitunter erheblich, wie [MS98] belegt.

Probleminstanzen aus Bibliotheken, wie z. B. die 10·10-Fisher-Thompson-Probleminstanz[16], sind vollständig, korrekt, aber von geringem Umfang und nicht realitätsnah. Sie sind zudem statisch und nicht vorab von extern determinierbar, und sie decken den Problemraum ebenfalls nur punktuell ab. Ihr großer Vorteil besteht darin, dass sie zur Verfügung stehen und direkt und ohne zusätzlichen Aufwand verwendet werden können.

[16]Besagte Probleminstanz stammt von [FT63]: Es liegen zehn Maschinen und zehn Aufträge vor, von denen jeder auf jeder Maschine genau einmal aber in unterschiedlicher Folge und mit unterschiedlichem Kapazitätsbedarf bearbeitet werden soll. Das Problem ist wohlbekannt (siehe z. B. [Sie95, S. 4], [BB97] und [TTL99, S. 5]).

Das Generieren von Probleminstanzen lässt einzig Realitätsnähe missen, die aber ex ante injiziert werden kann, indem plausible Annahmen über Verteilungen und Zusammenhänge in den Probleminstanzen getroffen werden und das Generieren mit Parameterausprägungen parametrisiert wird, die aus Praxisdaten gewonnen worden sind. Der Aufwand für das Generieren von Probleminstanzen resultiert aus einem hohen fixen Anteil für das Implementieren eines Generators und aus einem sehr geringen variablen Anteil für das wiederholte, automatische Generieren von Probleminstanzen mittels des Generators.

Die Aussagen, welche Tabelle 6.2 trifft, sind diskussionswürdig, implizieren aber, einen Generator zu erstellen und mit seiner Hilfe Probleminstanzen automatisch zu generieren.

Um einen Generator zu erstellen, der plausible Probleminstanzen erzeugt, ist unter anderem die bereits erwähnte $10 \cdot 10$-Fisher-Thompson-Probleminstanz in Anhang B.4.1 untersucht worden, um auf allgemeingültige Zusammenhänge zu schließen.

Aus den in Anhang B.4.1 geschilderten Phänomenen, Erwägungen und plausiblen Annahmen ist der Organisationsgrad o hergeleitet worden, der den Organisationstyp [Cor04, S. 30] der Fertigung quantitativ beschreibt. Er variiert zwischen 0 und 1, wobei 0 aussagt, dass reine Werkstattfertigung vorliegt, und 1, dass es sich um reine Fließfertigung handelt. Der Organisationsgrad o berechnet sich wie folgt[17] [VDM01],[DMV00] und [DMV99]:

$$o = \frac{1}{|M|} \sum_{i=0}^{|M|} \sum_{j=0}^{|M|} \left(p_{ij} - \frac{1}{|M|+1} \right)^2 . \tag{6.1}$$

Um nun eine Maschinenübergangsmatrix für einen gegebenen Organisationsgrad zu erstellen, die zusätzlich den in Anhang B.4.1 getroffenen Annahmen entspricht, ist der Algorithmus B.8 entworfen, implementiert und in Anhang B.4.2 beschrieben worden.

Die mittels Algorithmus B.8 mit einem gewünschten Organisationsgrad o generierte Maschinenübergangsmatrix U kann nun verwendet werden, um mittels Algorithmus B.9 Maschinenfolgen zu generieren.

Bisher ist beschrieben worden, wie Maschinenfolgen für Arbeitspläne bzw. Fertigungsaufträge generiert werden. Um plausibel und zufällig Zeitpunkte, Zeiträume und Dauern für die Arbeitspläne und ihre Positionen sowie für die Fertigungsaufträge und ihre Arbeitsgänge zu generieren, werden gleich-,

[17] Der Organisationsgrad der $10 \cdot 10$-Fisher-Thompson-Probleminstanz beträgt nur $o \approx 0{,}14$, was Werkstattfertigung impliziert.

normal- oder logarithmisch normalverteilte Zufallszahlen erzeugt, deren Generierung Anhang B.4.4 beschreibt.

Die Komponente zur Generierung von PPS-Daten arbeitet nun in zweierlei Modi: Entweder generiert sie eine Menge A von Aufträgen, oder sie generiert einen Auftragsstrom, indem sie auf Anforderung eine neue Zwischenankunftszeit t und einen neuen Auftrag a_i zurückgibt. Algorithmus B.10 verkörpert den ersten Modus, Algorithmus B.11 den zweiten.

Die im folgenden Abschnitt 6.5.2 beschriebene Komponente zur Simulation der Produktion nutzt die Generierung von PPS-Daten im zweiten Modus.

6.5.2 Komponente zur Simulation der Produktion und der PPS

Für die im Rahmen der vorliegenden Arbeit durchzuführende empirische Studie hat keine reale Fertigung zur Verfügung gestanden. Aus diesem Grunde ist ein flexibles, ereignisdiskretes Simulationsmodell erstellt worden, welches Fertigungen abbildet. Das Simulationsmodell weist – im Gegensatz zu einer realen Fertigung – den Vorteil auf, dass es verschiedene Fertigungen bzw. verschieden parametrisierte Fertigungen abbilden kann. Zudem wirken sich schlechte Ergebnisse von Experimenten, die mit Hilfe eines Simulationsmodells durchgeführt worden sind, nicht negativ auf eine reale Fertigung aus. Bei dem Simulationsmodell handelt es sich nicht um ein fixes Simulationsmodell im engeren Sinne, sondern um ein Programm für eine datengetriebene, ereignisdiskrete Simulation. Das Simulationsprogramm weist allerdings eine Besonderheit auf: Funktionsübergreifend gesprochen, ist das Simulationsprogramm der führende Prozess. Es stößt aktiv sämtliche anderen passiven Komponenten synchron und auch indirekt an und lässt sich gegebenenfalls ihre Berechnungsergebnisse zurückgeben. Die anderen Komponenten befinden sich in oder hinter einer DLL. Abbildung 6.4 stellt die Kommunikation der wesentlichen Komponenten über die DLL dar. Die Komponenten nehmen die folgenden Aufgaben wahr:

- Bei der *Registratur* registrieren sich alle Komponenten, um von den jeweils anderen Komponenten und vom Simulator bei Bedarf angesprochen zu werden.

- Der *CPU-Zeit-Messer* misst und protokolliert für beliebig viele Aktivitäten, die sich auch wiederholen dürfen, nicht die Rechen-, sondern die CPU-Zeit, so dass andere Prozesse auf demselben Rechner die Messergebnisse nicht verfälschen. Ein *Speicherdifferenzmesser*, den Abbildung 6.4 *nicht* mit darstellt, misst zusätzlich den Speicherbedarf bzw. die

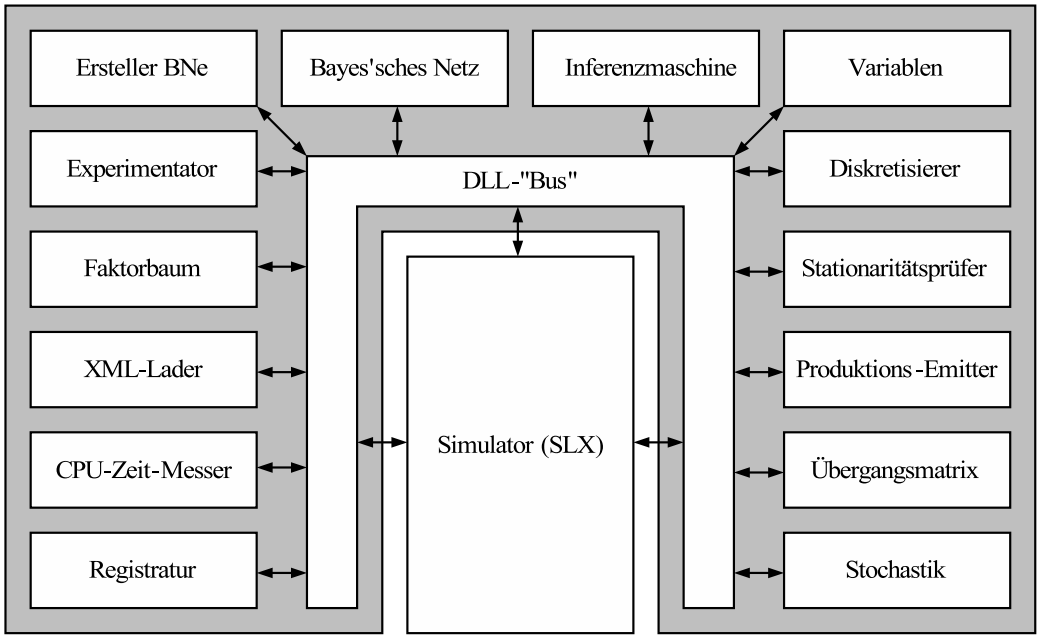

Abbildung 6.4: Komponentenarchitektur zum Test und zur Anwendung des Softwaresystems zur Unterstützung der PPS mittels BNe sowie zu Experimenten mit dem Softwaresystem

Speicherbedarfsdifferenz, welche andere Komponenten, wie z. B. die *Inferenzmaschine*, zwischen zwei Zeitpunkten verursachen.

- Der *XML-Lader* liest XML-Dateien zuerst in DOMs und aus denen in proprietäre Datenstrukturen, wie z. B. in BNe oder in *Faktorbäume*. Er schreibt diese Datenstrukturen auf dem Rückweg zuerst in DOMs und aus denen zurück in XML-Dateien.

- Der *Faktorbaum* repräsentiert eine geordnete Menge von Experimenten. Die Experimente sind nicht nur hinsichtlich ihrer Parameterausprägungen heterogen, sondern auch hinsichtlich ihrer Parameter. Der *Faktorbaum* erlaubt es, effizient durch die Experimente zu iterieren (siehe Abschnitt 6.6.2).

- Der *Experimentator* schaltet auf Anforderung des Simulators den Faktorbaum weiter und stellt ein neues Experiment zur Verfügung bzw. parametrisiert die anderen Komponenten entsprechend des aktuellen Zustandes des Faktorbaumes. Er parametrisiert nur die Komponenten neu, deren Parameter sich tatsächlich geändert haben, und nimmt demzufolge keine überflüssige Parametrisierung vor.

- Der *Ersteller BNe* erstellt nach den in den Abschnitten 4.2.3 und 4.2.4 beschriebenen Gütemaßen und Algorithmen und unter Nutzung der in Abschnitt 6.4.2 beschriebenen Komponente für jedes Experiment ein dynamisches BN aus multivariaten Zeitreihen, die durch das Abtasten der *Variablen* entstanden sind.

- Das *Bayes'sche Netz* wird durch die in Abschnitt 6.4.1 beschriebene Komponente repräsentiert. Der Ersteller BNe erstellt dieses BN.

- Die *Inferenzmaschine* vollzieht PI i. w. S. über dem BN, indem sie sich selbst einmalig initial erstellt und dann wiederholt PI i. e. S. über der erstellten Inferenzmaschine vollführt, wie Anhang A.3 beschreibt. Inhaltliche Besonderheiten der Inferenzmaschine stellt Abschnitt 6.4.3 dar.

- Die *Variablen*-Komponente verwaltet die Stell-, Regel-, Führungs- und intermediären Größen hinsichtlich ihrer Wertebereiche und gegebenenfalls hinsichtlich der Intervalle, in denen sie diskretisiert werden. Zusätzlich verwaltet sie die multivariaten Zeitreihen, die im Rahmen der Experimente durch das Abtasten der Variablen entstehen.

- Der *Diskretisierer* legt fest, in welchen Intervallen und nach welchen Verfahren reellwertige und ganzzahlige *Variablen* diskretisiert werden, und nimmt das Diskretisieren selbst vor. Er setzt darüber hinaus diskrete Werte wieder in reellwertige oder ganzzahlige um, sofern die Stellgrößen reellwertig oder ganzzahlig sein sollten. Abschnitt 4.4.3 diskutiert das äquidistante und äquifrequente Diskretisieren.[18]

- Der *Stationaritätsprüfer* stellt mittels einer Heuristik fest, ob sich die abgetasteten Variablen der simulierten Fertigung und somit auch die gesamte Fertigung bereits in einem eingeschwungenen Zustand befinden. Er ermittelt für die bisher abgetasteten Werte $x_1, \ldots, x_n$ einer *jeden* Variable die Anzahl

$$c = \sum_{i=1}^{n-1} \begin{cases} 1| \ (x_i \leq m < x_{i+1}) \vee (x_i \geq m > x_{i+1}) \\ 0| \ \text{sonst} \end{cases} , \tag{6.2}$$

  mit

$$m = \frac{\sum_{j=1}^{n} x_j}{n},$$

[18] Die Anzahl der Intervalle im während des Experimentes „überstrichenen“ Wertebereiches ist ein Faktor im multifaktoriellen Experiment. Die Intervallgröße ist der Quotient aus der Länge des Wertebereiches und der gegebenen Anzahl der Intervalle.

die ausdrückt, wie oft der Mittelwert m der Werte zwischen zwei aufeinanderfolgenden Werten x_i und x_{i+1} liegt. Überschreitet die Anzahl c für *jede* Variable eine vorgegebene Grenze k, so betrachtet der Stationaritätsprüfer das System als eingeschwungen bzw. als in einem stationären Zustand befindlich.[19]

- Der *Produktions-Emitter* stellt auf Anforderung des Simulators Maschinen und vor allem Aufträge mit Arbeitsgängen, Zeiten und Terminen zur Verfügung. Er verfährt nach Algorithmus B.11 und nutzt die *Übergangsmatrix* und die *Stochastik*-Komponente.

- Die *Übergangsmatrix* entspricht der mittels Algorithmus B.8 erzeugten und dient dem *Produktions-Emitter* als Eingabe. Ihre Parameter M und o entstammen dem aktuellen Zustand des *Faktorbaumes*. Der *Experimentator* stellt die Parameter zum Erzeugen bereit.

- Die *Stochastik*-Komponente stellt auf Anforderung $[0; 1)$-gleichverteilte, λ-exponentialverteilte und μ, σ-lognormalverteilte Zufallszahlen bereit. Vorwiegend der *Produktions-Emitter*, aber auch der Simulator sprechen sie direkt an.

- Der *DLL-„Bus“* stellt erstens die einheitliche Schnittstelle der DLL gegenüber dem Simulator zur Verfügung. Zweitens vermittelt er Anfragen und Antworten zwischen den anderen Komponenten, die sich bei der *Registratur* angemeldet haben.

- Beim *Simulator (SLX)*[20] handelt es sich um das bereits erwähnte Simulationsprogramm, das ein Modell für die ereignisdiskrete Simulation datengetrieben erstellt und im Anschluss simuliert. Sowohl zum Erstellen des Modells als auch zum Simulieren ruft der Simulator Funktionen des *DLL-„Busses“* auf.

Den komponentenübergreifenden Ablauf einer Simulationsstudie stellt Anhang B.5 anhand eines UML-Kollaborationsdiagrammes dar.

Die Komponente zur Simulation der Produktion und der PPS, die hier dargestellt worden ist, spielt mit einem Regler zusammen. Sie kann nicht nur

[19] Das Verfahren ist „Allgemeingut“ und trägt in der englischen Sprache den Namen „Crossing [of] the Mean“. [Lie95, S. 160 f.], [Fis73a, S. 275] Es erkennt zuverlässig stationäre Prozesse. Weist es bei genügend großem k, z. B. $k \geq 10$, Stationarität aus, so trifft seine Aussage zu, wenn der Prozess nicht periodisch schwingt.

[20] SLX kürzt Simulation Language with eXtensibilities ab. Die Wolverine Software Corporation entwickelt SLX. Es ist lediglich „objektbasiert“, kennt also keine Vererbung und lehnt sich in der Syntax an die Programmiersprache C an. SLX umfasst einen Simulationskern sowie eine rudimentäre Entwicklungsumgebung.

gemeinsam mit einem BN als Regler eingesetzt werden, sondern auch mit gänzlich anderen Reglern, wie z. B. KNNen oder Fuzzy-Regelsystemen, sofern diese nur die erforderlichen Schnittstellen implementieren. Im Rahmen der vorliegenden Arbeit ist die Simulationskomponente allerdings ausschließlich für die Durchführung einer empirischen Studie eingesetzt worden.

6.5.3 Komponente zum Management der PPS-Daten

Ein erster Plan für das Erstellen des Softwaresystems zur Unterstützung der PPS mittels BNe hat noch vorgesehen, Datenstrukturen und Algorithmen zu konzipieren, zu entwerfen und zu implementieren, die dem Anlegen, Bereitstellen, Modifizieren und Löschen von Stamm- und Bewegungsdaten aus dem PPS-Umfeld dienen. Bei den vorgesehenen Datenstrukturen hat es sich insbesondere um

- Materialbedarf und Materialien,
- Arbeitspläne mit Arbeitsplanpositionen,
- Stücklisten mit Stücklistenpositionen,
- Kapazitätseinheiten mit Kapazitätsangebot,
- Aufträge mit Arbeitsgängen,
- Auftragsstücklisten mit Auftragsstücklistenpositionen,
- Rückmeldungen und Teilmengen,
- Lager, Materialbewegungen und Bestand

gehandelt. Es ist jedoch davon abgesehen worden, ein solches Management für PPS-Daten bereitzustellen, da nicht der Anspruch besteht, das Softwaresystem mit dem vollständigen Datenmanagement eines PPS-Systems zu versehen. Ein solches Datenmanagement ginge zudem über Sinn und Zweck des Softwaresystems hinaus, wenn nicht gar an ihnen vorbei, und eine empirische Untersuchung bedarf nicht des Managements der PPS-Daten in diesem Umfange.

Das vorliegende Softwaresystem geht nunmehr davon aus, dass ein PPS-System mit einer entsprechenden Datenverwaltung existiert, von dem das Softwaresystem bei Bedarf Daten erhält und an das es bei Bedarf Daten übergibt. Eine BDE oder gar eine MDE eignen sich als Partnersysteme für das Softwaresystem ebenso wie ein PPS-System. Wenn das in diesem Kapitel beschriebene Softwaresystem die PPS tatsächlich nur unterstützt und nicht

die Aufgaben eines PPS-Systems, einer BDE oder einer MDE übernimmt, tritt es nicht in Konkurrenz zu diesen Systemen und erfährt so eine bessere Akzeptanz bei Anwendern und Softwarehäusern.

6.6 Weitere Komponenten des Softwaresystems

6.6.1 Graphische Schnittstelle zum Benutzer

Bei BNen handelt es sich nicht um Black-Boxes, was sie z. B. gegenüber KNNen und EAen auszeichnet. BNe können manuell erstellt und/oder modifiziert werden, sie ermöglichen die manuelle Eingabe von Expertenwissen, und ihre Struktur, ihre Parameter und ihre Arbeitsweise können manuell beurteilt und nachvollzogen werden. Die aufgeführten Eigenschaften BNe stellen wesentliche Vorteile gegenüber anderen Verfahren der Wissensverarbeitung dar. Sie müssen maximal ausgeschöpft werden, um das Potenzial BNe zur Unterstützung der PPS vollständig zu erschließen. Nur eine komfortable und ergonomische Schnittstelle zum Benutzer erlaubt einfaches und schnelles manuelles Erstellen und Modifizieren BNe, reibungsloses Integrieren von Expertenwissen in BNe sowie die Interpretation BNe und das Nachvollziehen ihrer Arbeitsweise. Eine solche grafische Schnittstelle zum Benutzer ist als Komponente für das Softwaresystem zur Unterstützung der PPS mittels BNe erstellt worden, und dieser Abschnitt stellt diese Komponente vor.

Abbildung 6.5 zeigt einen Bildschirmabzug des grafischen Multi-Document-Interface (MDI) zum Benutzer. Im einzelnen spiegelt der Bildschirmabzug von oben nach unten die folgenden Dinge wider:

- ein Pull-Down-Menü für das aktuelle Dokument („Notizbuch“[21]) und für das im aktuellen Notizbuch aktuelle BN, das unter anderem ein Öffnen und Speichern von Notizbüchern anstößt,
- zwei Notizbücher für jeweils potenziell mehrere BNe, von denen das eine Notizbuch ein BN und das andere zwei BNe in Registerkarten enthält,
- die grafische Repräsentation des Graphen eines BNes in der Registerkarte eines Notizbuches,

[21] Mehrere logisch zusammengehörende BNe können im Softwaresystem zu einem sogenannten Notizbuch zusammengefasst werden, welches dann als Standard im XBN-Format [Dec07] (siehe auch Abschnitt 6.4.1) gespeichert wird. Das Öffnen von Notizbüchern, welche in diesem Format gespeichert worden sind, ermöglicht das Softwaresystem natürlich ebenso.

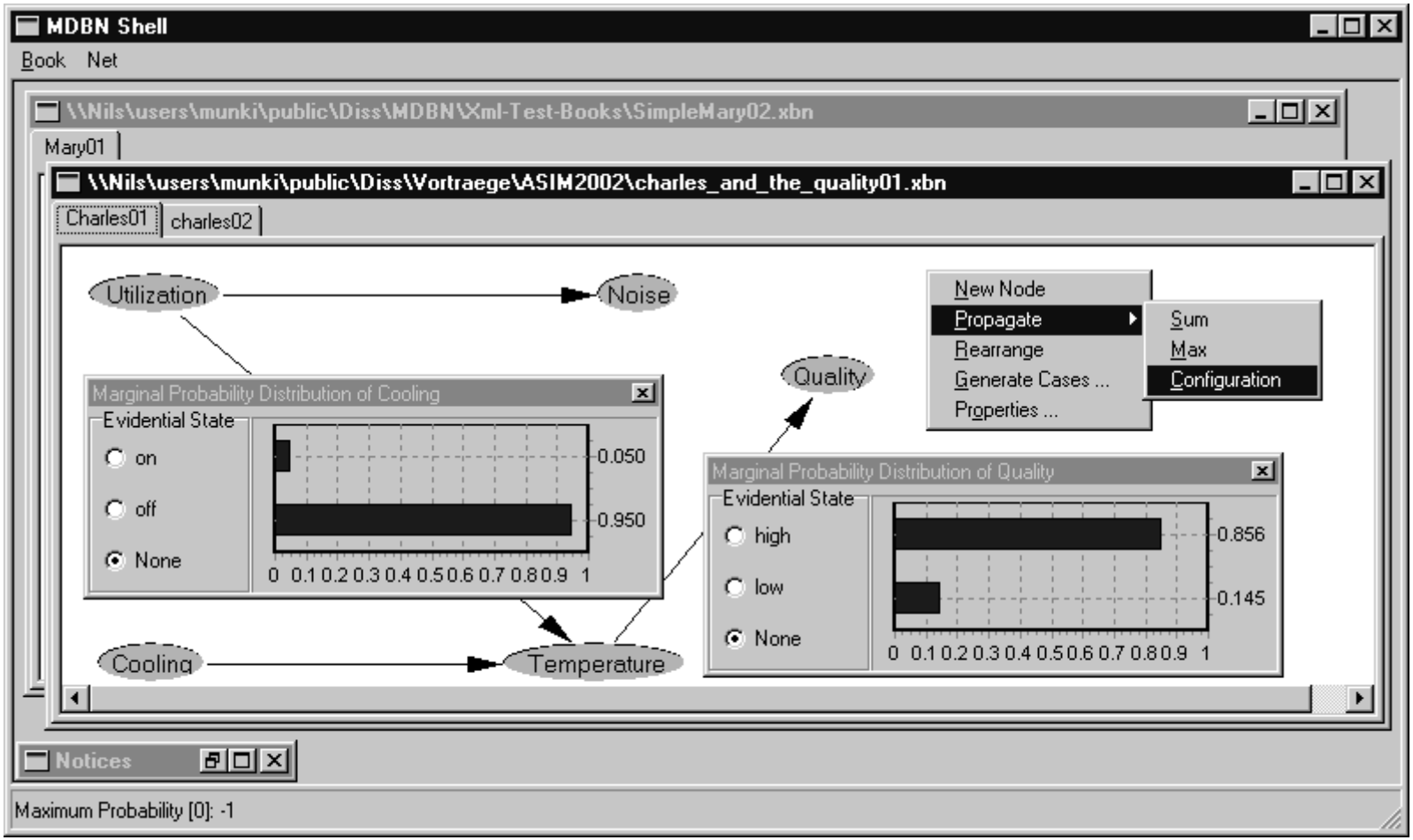

Abbildung 6.5: Graphisches MDI zum Benutzer

- zwei Dialoge mit Balkendiagrammen für Randverteilungen zweier Variablen des dargestellten BNes, die auch erlauben, Evidenz für die Variablen über Radio-Buttons zu setzen,
- ein Kontextmenü, das die Operationen offeriert, welche über dem dargestellten BN ausgeführt werden können:
  - das Hinzufügen eines Knotens zum BN,
  - drei Arten exakter PI über dem BN,
  - das Generieren einer Datenbasis auf Grundlage des BNes und
  - das Setzen von Eigenschaften des BNes,
- ein minimiertes Fenster, das die Logging- und gegebenenfalls die Debugging-Ausgaben enthält, und eine Statusleiste.

Die grafische Schnittstelle zum Benutzer weist einige Vorzüge, Besonderheiten und Bedienmöglichkeiten auf, die im folgenden aufgeführt werden:

- Anlegen von Kanten mittels „Drag and Drop“ vom Eltern- auf den Kindknoten,
- Löschen und Invertieren von Kanten mittels rechter Maustaste und Kontextmenü,
- Verschieben von Knoten mittels Maus und kontinuierliches Nachführen der zu den Knoten inzidenten Kanten während des Verschiebens,
- automatisches Anpassen der Größe der Ellipsen, welche die Knoten repräsentieren, an ihre Beschriftung,
- automatisches Anpassen der Größe der Pfeilspitzen, welche die Richtung der Kanten repräsentieren, an den Abstand zwischen den zwei ihrer Kante inzidenten Knoten und
- Öffnen der Dialoge für die Diagramme mit den Randverteilungen über Doppelklick auf die Knoten.

Abbildung 6.6 zeigt zwei modale Dialoge. Der Dialog im Hintergrund dient dem Anlegen, Anzeigen und Ändern eines Knotens eines BNes sowie dem Anlegen, Ändern, Entfernen und Vertauschen der Reihenfolge seiner Zustände. Der Dialog im Vordergrund wird aus dem Dialog im Hintergrund heraus aufgerufen und dient lediglich dem Ändern des ausgewählten Zustandes des

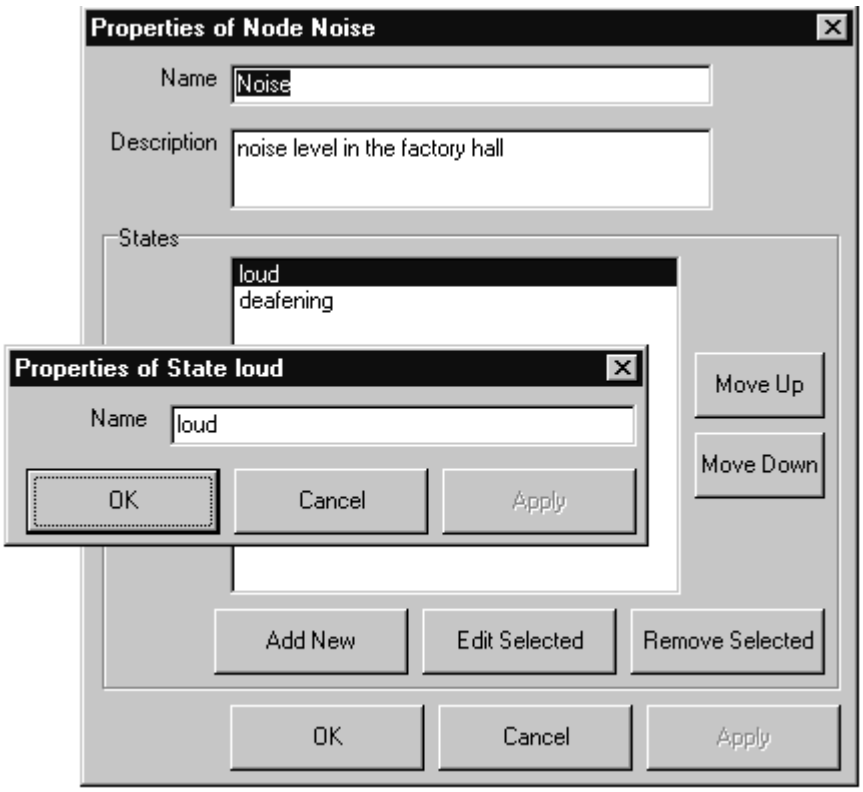

Abbildung 6.6: Modale Dialoge zum Editieren eines Knotens eines BNes sowie zum Editieren und zum Vertauschen der Reihenfolge der Zustände des Knotens

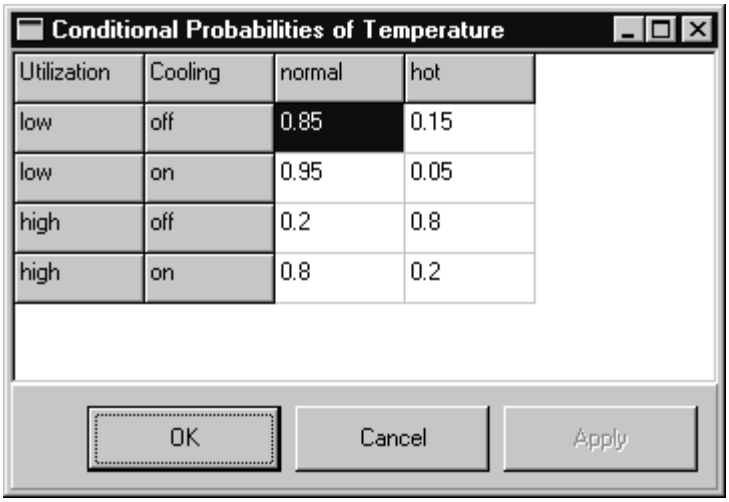

Abbildung 6.7: Modaler Dialog zum Editieren einer Bewertungsfunktion bzw. ihrer bedingten Wahrscheinlichkeiten

Knotens, was sich bei diesem Knotentyp auf das Ändern des Namens des Zustandes beschränkt.

Abbildung 6.7 zeigt einen modalen Dialog, der dem Editieren der bedingten Wahrscheinlichkeiten einer Bewertungsfunktion eines Knotens eines BNes dient. Der Benutzer kann die bedingten Wahrscheinlichkeiten direkt in den Zellen der Tabelle editieren. Der Dialog normalisiert die Verteilungen bedingter Wahrscheinlichkeiten beim Bestätigen der Änderungen und fängt Fehleingaben ab.

Sechs Operationen zur Modifikation eines BNes wirken sich auf die Bewertungsfunktionen der betroffenen Knoten des BNes aus:

1. Das Hinzufügen einer Kante,
2. das Löschen einer Kante,
3. das Invertieren einer Kante,
4. das Einfügen eines Zustandes in eine Variable,
5. das Entfernen eines Zustandes aus einer Variable und
6. das Vertauschen zweier (aufeinanderfolgender) Zustände einer Variable,

wobei die dritte Operation auf das Ausführen der zweiten und der ersten Operation direkt hintereinander zurückgeführt werden kann. Abschnitt 6.4.1 beschreibt an seinem Ende bereits ausführlich, wie die Bewertungsfunktionen

aufgrund der obigen Operationen modifiziert werden, so dass sich das BN auch nach besagten Operationen erwartungskonform verhält.

Das Softwaresystem zur Unterstützung der PPS mittels BNe erlaubt weiterhin, aufeinanderfolgende Operationen rückgängig zu machen. Nach welchem Prinzip das Rückgängigmachen vonstatten geht, beschreibt bereits Abschnitt 6.4.1 an seinem Ende.

Bis hierher erlaubt es die grafische Schnittstelle zum Benutzer, ein BN komfortabel anzulegen und zu editieren. Die grafische Schnittstelle adressiert allerdings noch ein weiteres Problem: Wenn man BNe automatisch aus einer Datenbasis erstellt, liegen für die Knoten des BNes, die zumeist mit mehreren anderen Knoten über Kanten verbunden sind, keine Positionen für ihre grafischen Repräsentation vor. Die grafischen Repräsentationen der Knoten, ab jetzt nur noch Knoten genannt, müssten ergo per Hand so verschoben werden, dass das BN bezüglich seiner Knoten und Kanten gut zu überblicken ist. Diese Arbeit ist bei großen und stark verbundenen BNen sehr aufwendig. [MMHT05] Aus diesem Grunde sind mehrere Algorithmen implementiert worden, welche die Knoten eines BNes automatisch so anordnen, dass es gut zu überblicken ist. Diese Algorithmen stellt Anhang B.6 vor.

Die in diesem Abschnitt vorgestellte grafische Schnittstelle zum Benutzer erlaubt es, ein BN komfortabel manuell zu editieren und seine Struktur und seine Arbeitsweise zu interpretieren. Einzig eine Zoomfunktion wäre für einen Überblick über BNe mit sehr vielen Knoten noch angebracht, wobei in der grafischen Repräsentation eines BNes aber gescrollt werden kann.

6.6.2 Komponente zum Management von Experimenten

Um das Potenzial BNe zur PPS empirisch zu untersuchen und zu allgemeingültigen Aussagen darüber zu gelangen, ob sich BNe zur PPS eignen, ist es nötig, eine Vielzahl von Experimenten durchzuführen.

Eine Komponente zum Management von Experimenten muss

1. es erlauben, mehrere Experimente vorab so kompakt zu definieren, dass keine Redundanz in der Definition der Experimente vorliegt, die sich weitgehend gleichen,

2. in der Lage sein, sämtliche Komponenten und Subkomponenten automatisch und sukzessive aufgrund der Experimentendefinition zu parametrisieren,

3. sämtliche definierten Experimente in Stapelverarbeitung durchführen,

so dass kein Eingriff seitens des Benutzers beim Durchführen der Experimente notwendig ist, und

4. die Ergebnisse der Experimente aufbereiten und sie in einer Form zur Verfügung stellen, welche eine anschließende Auswertung vereinfacht.

Für jedes der besagten Experimente müssen mehrere der im bisherigen Verlaufe dieses Kapitels beschriebene Komponenten umfangreich parametrisiert werden. Bei diesen Komponenten handelt es sich insbesondere um:

- die Komponente zur Akquisition BNe,
- die Komponente zur Anwendung BNe,
- die Komponente zur Generierung von PPS-Daten und
- die Komponente zur Simulation der Produktion.

Diese Komponenten setzen sich hierarchisch aus Subkomponenten zusammen, welche wiederum eigene Parameter besitzen. Von besonderem Belang ist der Umstand, dass im Rahmen aufeinanderfolgender Experimente Subkomponenten ausgetauscht werden müssen, die bezüglich der ihnen übergeordneten Komponente dieselbe Aufgabe erfüllen, aber andere Parameter(mengen) besitzen. Soll z. B. in einem Experiment eine Exponentialverteilung mit nur einem Parameter zur Generierung zufälliger Bearbeitungszeiten herangezogen werden und in einem darauffolgenden eine logarithmische Normalverteilung mit zwei Parametern, so weisen sie verschiedene Parameter(mengen) auf. Dasselbe gilt beispielsweise auch für die Algorithmen zur automatischen Akquisition BNe, die z. T. sogar disjunkte Parametermengen besitzen. Dem Umstand heterogener Parameter(mengen) muss die Komponente zum Management von Experimenten Rechnung tragen.

Zudem liegen verschiedene Typen von Parametern vor. Die einfachsten entsprechen elementaren Datentypen. Darüber hinaus existieren Parameterlisten sowie komplexe Parameter, die sich wiederum aus elementaren und/oder komplexen Parametern zusammensetzen. Auch treten Parameter z. T. in Kombination (parallel) und z. T. alternativ auf. Die Komponente zum Management von Experimenten muss sämtliche Typen von Parametern abbilden.

Die Parametrisierung der Komponenten und Subkomponenten ist relativ aufwendig, weil nicht nur elementare Parameter gesetzt, sondern oft auch Subkomponenten und sogar ganze Komponenten vollständig zerstört und zu ihnen alternative (Sub)komponenten neu aufgebaut werden. Aus diesem Grunde darf beim Wechsel von einem Experiment zum nächsten keine vollständige Neuparametrisierung aller Komponenten erfolgen, sondern es dürfen

nur diejenigen Parameter verändert werden, die sich zwischen zwei aufeinanderfolgenden Experimenten unterscheiden.

Die bisher genannten Anforderungen soll die Komponente zum Management von Experimenten erfüllen. Darüber hinaus soll die Komponente nicht von den Komponenten abhängen, die sie verwaltet, und somit auch für Experimente mit anderen Verfahren zur Unterstützung der PPS zur Verfügung stehen.

Die Komponente zum Management von Experimenten behandelt eine Menge von Experimenten als sogenannten Faktorbaum. Abbildung 6.8 stellt ein Exemplar eines Faktorbaumes dar. Jedes Blatt des Baumes verkörpert einen Faktor. Ein Faktor beinhaltet entweder weitere Faktoren und strukturiert so die Menge der Experimente, oder er ist ein „elementarer“ Faktor, der Bereiche für elementare Werte des Faktors definiert. Zur Durchführung der Experimente wird über dem Faktorbaum iteriert. Zu Beginn jeder Iteration parametrisieren nur die sich ändernden Faktoren die ihnen zugeordneten Parameter der Komponenten des Softwaresystems zur Unterstützung der PPS mittels BNe, und nach jeder Iteration erfolgt ein Experiment.

Anhang B.7 beschreibt den Aufbau und die Funktionsweise des Faktorbaumes ausführlich. So legt Anhang B.7.1 dar, wie ein Faktorbaum prinzipiell strukturiert ist und wie er erstellt wird; Anhang B.7.2 beschreibt kurz das Iterieren über einem Faktorbaum; und Anhang B.7.3 geht auf das effiziente und universelle Parametrisieren des Softwaresystems für Experimente ein.

Nach dem Initialisieren und nach jedem Inkrementieren des Faktorbaumes führt die Komponente zum Management von Experimenten ein Experiment durch. Für jedes Experiment legt sie eine persistente Datenstruktur an und versieht die Datenstruktur mit der für dieses Experiment geltenden Konfiguration. Die parametrisierten Komponenten legen die Ergebnisse des Experimentes ebenfalls in dieser Datenstruktur ab. Nachdem alle Experimente durchgeführt worden sind, lassen sich die Daten in der Gesamtheit aller angelegten Strukturen leicht auswerten.

Sollte eine Folge von Experimenten versehentlich unterbrochen werden, so erlauben es die persistenten Datenstrukturen zudem, die Experimente wieder aufzunehmen, ohne bereits abgeschlossene Experimente zu wiederholen.

Zusammengefasst erfüllt die Komponente zum Management von Experimenten sämtliche Anforderungen, die zu Anfang dieses Abschnittes an sie gestellt worden sind. Da von den im vorliegenden Kapitel vorgestellten Komponenten unabhängig, ist sie darüber hinaus zum Management nahezu beliebiger Experimente im Softwarebereich fähig.

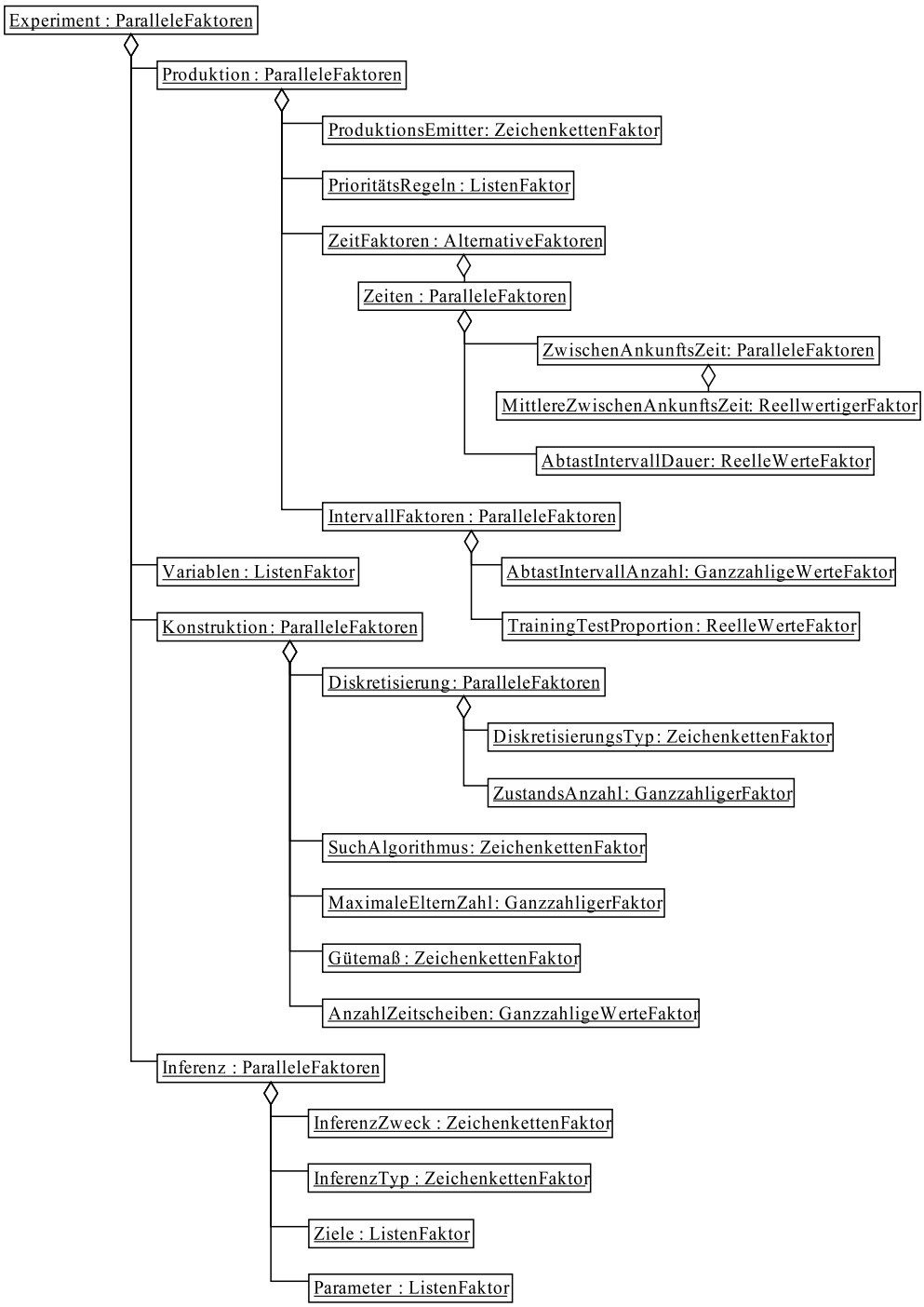

Abbildung 6.8: Beispiel für einen Faktorbaum als UML-Instanzdiagramm. Aus Platzgründen ist auf die Angabe der möglichen Werte der Faktoren verzichtet worden.

6.6.3 Komponenten zum Test des Softwaresystems

Die Komponente zur Akquisition BNe und die Komponente zur Anwendung BNe bedürfen umfangreicher, übergreifender Tests, die über einfache Unit-Tests hinausgehen. Die Komponenten zum Test des Softwaresystems adressieren den Test genau dieser beiden Komponenten. Die Tests der übrigen Komponenten des Softwaresystems zur Unterstützung der PPS mittels BNe sind manuell oder mittels Unit-Tests auf herkömmliche Art und Weise erfolgt.

Das Testen der Komponenten zur Akquisition und zur Anwendung BNe stellt Abbildung 6.9 dar: Die Komponente zum Management von Experimenten parametrisiert zu Beginn einen Generator (1), der daraufhin ein BN zum Test erstellt, das in seinen Eigenschaften den Werten der Parameter entspricht, und es der Repräsentationskomponente übergibt (2). Die Parameter drücken z. B. die Anzahl der Knoten, die Anzahl der Zustände pro Knoten, die Anzahl der eingehenden und/oder ausgehenden Kanten pro Knoten oder die Wirkungsweise der Bewertungsfunktionen aus. Die Komponente zum Management von Experimenten führt daraufhin zweierlei, voneinander unabhängige Experimente durch: Zum einen vollzieht sie mittels der Komponente zur Anwendung BNe PI über dem BN (3, 4 und 5) und wertet deren Ergebnis im Anschluss im Hinblick auf Laufzeit, Speicherbedarf und Korrektheit aus (6). Zum anderen erstellt die Komponenente zum Management von Experimenten mittels stochastischer Vorwärtssimulation (siehe Algorithmus A.22) über dem BN eine Menge von Fällen (7, 8 und 9), deren Mächtigkeit einem weiteren Parameter entspricht, den die Komponente zum Management von Experimenten vorgibt. Die Komponente zum Management von Experimenten stößt dann die Komponente zur Akquisition BNe an (10), die aus den Fällen (11) ein BN erstellt (12). Es erfolgt im Anschluss ein Vergleich (13) des ursprünglich generierten BNes mit dem, das aus den Fällen erstellt worden ist, welche die stochastische Vorwärtssimulation über dem ursprünglich generierten BN ermittelt hat. Das Ergebnis des Vergleichs gibt Aufschluss über die Arbeitsweise der Komponente zur Akquisition BNe: Die Arbeitsweise ist korrekt, wenn das BN, das aus den Fällen erstellt worden ist, dem ursprünglichen BN hinsichtlich seiner Kanten und seiner Bewertungsfunktionen gleicht. Laufzeit und Speicherbedarf zum Erstellen des BNes werden natürlich ebenfalls gemessen und ausgewertet.

Die Komponente zum Management von Experimenten ermöglicht es per se, mehrere Experimente mit verschiedenen Konfigurationen (Kombinationen von Parameterwerten) automatisch nacheinander auszuführen. Auf die beschriebene Art und Weise gelingt es, die Komponente zur automatischen Akquisition BNe und die Komponente zur Anwendung BNe parallel zu testen. Die Komponente zum Test des Softwaresystems testet beide Komponenten

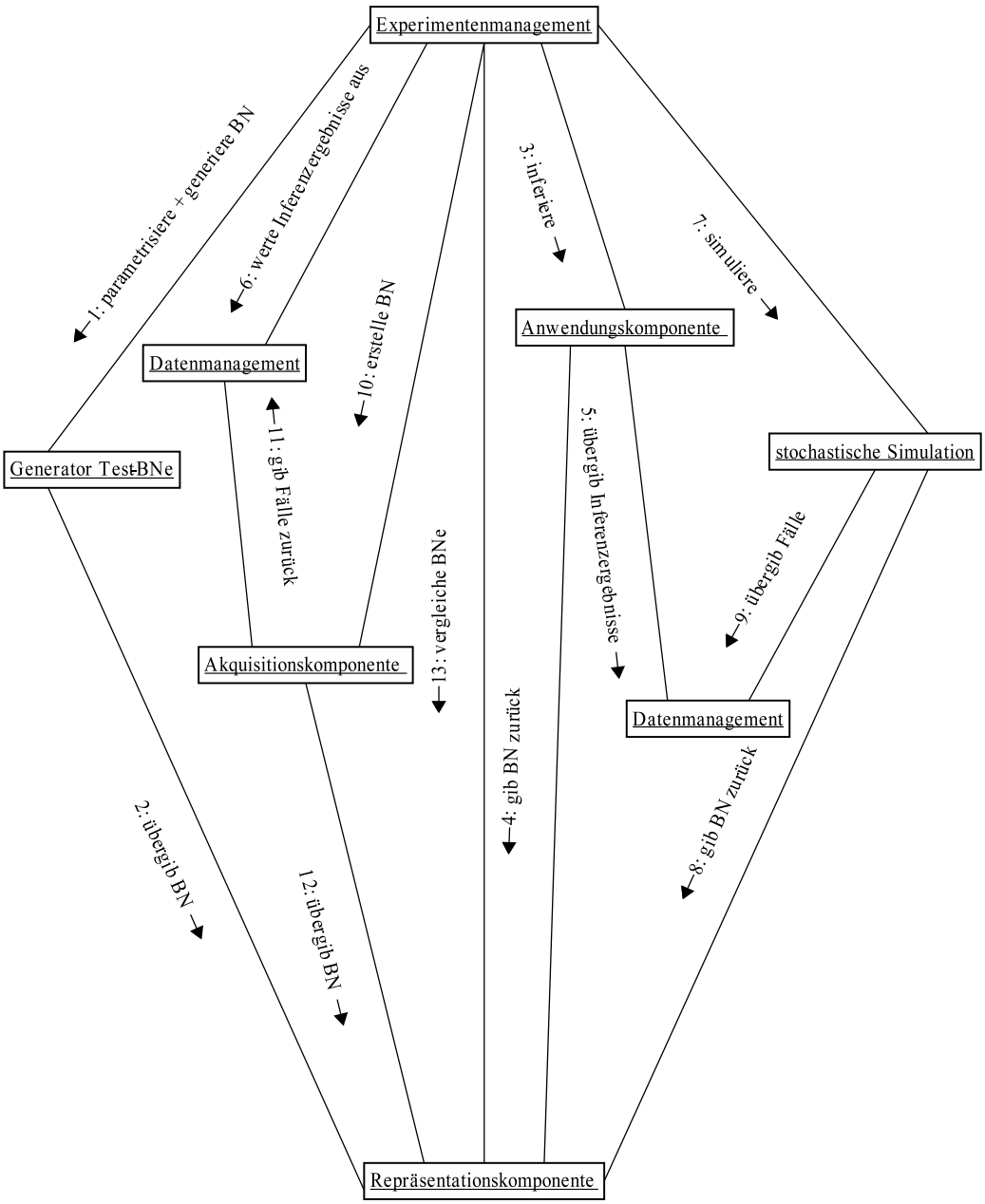

Abbildung 6.9: Ablauf des Tests der Komponenten zur Akquisition und zur Anwendung BNe in Form eines UML-Kollaborationsdiagrammes

auf ihre korrekte Arbeitsweise und zudem auf ihre Laufzeiten und auf ihren Speicherbedarf.

Anhang B.8 beschreibt zentrale Algorithmen, Datenstrukturen und Formeln, welche die Komponente zum Test des Softwaresystems anwendet. Sie dienen insbesondere dem zielgerichteten Erzeugen BNe für Testzwecke, das der „Generator Test-BNe“ ausführt, den bereits Abbildung 6.9 auf der linken Seite aufführt.

Anhang B.8.1 legt dar, wie ein zufälliger, gerichteter Baum aufgespannt wird, der dem Graphen des Test-BNes als Grundlage dient.

Relativ leicht wäre es nun, die Bewertungsfunktionen des in Anhang B.8.1 generierten BNes mit gleichverteilten Zufallszahlen zu belegen und die bedingten Wahrscheinlichkeitsverteilungen in ihnen zu normalisieren. Diese Vorgehensweise, die z. B. [SSGM94] gewählt haben, führt dazu, dass sich das BN völlig willkürlich verhält, dass die PI zu völlig unvorhersehbaren Ergebnissen führt und dass die Eingabe von Evidenz die Ergebnisse der PI nicht zielgerichtet beeinflusst. Gerade für Tests müssen die Ergebnisse der PI aber vorhersagbar und durch die Eingabe von Evidenz zielgerichtet beeinflussbar sein. Deshalb erhält jeder Knoten des BNes einen Typ zugewiesen, der aussagt, wie die Zustände der Eltern des Knotens seine bedingten Wahrscheinlichkeiten tendenziell beeinflussen.

Anhang B.8.2 beschreibt, wie Bewertungsfunktionen verschiedener Typen erstellt bzw. mit bedingten Wahrscheinlichkeiten gefüllt werden. Als Typen von Bewertungsfunktionen stehen derzeit „Summe“, „inverse Summe“, „Maximum“ und „Minimum“ zur Disposition, wobei weitere Typen denkbar wären.

Erhält nun die Bewertungsfunktion einer jeden Variable des BNes einen Typ zugewiesen, der aussagt, wie die Zustände der Eltern des Knotens seine bedingten Wahrscheinlichkeiten tendenziell beeinflussen, so verhält sich das BN nicht völlig willkürlich, die PI führt zu vorhersehbaren Ergebnissen und die Eingabe von Evidenz beeinflusst die Ergebnisse der PI zielgerichtet. Gerade für Tests sind diese Eigenschaften eines BNes unabdingbar.

Auf die in diesem Abschnitt beschriebene Art und Weise gelingt es, die Komponente zur automatischen Akquisition BNe und die Komponente zur Anwendung BNe parallel zu testen. Die Komponente zum Test des Softwaresystems testet beide Komponenten auf ihre korrekte Arbeitsweise und zudem auf ihre Laufzeiten und auf ihren Speicherbedarf.

Die Komponente zum Management von Experimenten erlaubt es, die Komponenten zum Test des Softwaresystems sowie zur Anwendung und zur automatischen Akquisition BNe sukzessive verschieden zu parametrisieren und für jede Konfiguration einen Test auszuführen. Aus den Ergebnissen multipler Tests erwächst die Möglichkeit, Laufzeiten und Speicherbedarf für BNe, für Algorithmen und Gütemaße zur automatischen Akquisition BNe

und für PI über BNen zu inter- oder extrapolieren, woraufhin Empfehlungen für ihren Einsatz gegeben werden können.

Ein vollautomatischer Integrationstest des Softwaresystems zur Unterstützung der PPS mittels BNe im Hinblick auf seine korrekte Arbeitsweise ist aber aufgrund der hohen Komplexität des Softwaresystems faktisch nicht möglich gewesen.

6.7 Potenzielle Einsatzgebiete des Softwaresystems

Das Softwaresystem zur Unterstützung der PPS mittels BNe ist einerseits ein vollständiges und umfassendes Managementsystem zur automatischen und manuellen Akquisition, Repräsentation und Anwendung allgemeiner BNe und DBNe. Deshalb eignet es sich prinzipiell für jedes Einsatzgebiet, für das BNe und DBNe geeignet sind. Zu den hauptsächlichen Einsatzgebieten BNe und DBNe zählen

- Prognose,
- Diagnose und
- Entscheidungsunterstützung.

BNe und DBNe werden auch als komplexe (technische) Regler oder als Erklärungsmodell eingesetzt. Spezielle Aufgaben der PI über BNen und DBNen gibt Tabelle 4.7 an. Das hier vorgestellte Softwaresystem kann grundsätzlich auf all den genannten Gebieten eingesetzt werden und eignet sich zur Lösung all der aufgeführten Aufgaben.

Beispiele für bisherige Anwendungen der Wissensverarbeitung im Kontext BNe hat bereits Abschnitt 4.6 zusammengestellt. Die Gebiete, auf denen der Einsatz BNe erfolgt ist, sind auch potenzielle Einsatzgebiete für das vorliegende Softwaresystem zur Unterstützung der PPS mittels BNe.

Da das Softwaresystem andererseits speziell auf die PPS ausgerichtet worden ist und auch mehrere PPS-spezifische Komponenten aufweist, ist es besonders geeignet, schlechtstrukturierte Probleme der PPS zu lösen. Die Gebiete der PPS, auf denen schlechtstrukturierte Probleme vorliegen, und die schlechtstrukturierten Probleme, um welche es sich im Speziellen handelt, beschreibt Abschnitt 2.6.2.

BNe arbeiten grundsätzlich nur mit Variablen, welche diskrete Wertebereiche mit nur wenigen möglichen Ausprägungen aufweisen. In Anwendungsfällen, in denen reellwertige Variablen auftreten oder ganzzahlige Variablen

mit sehr vielen tatsächlichen Werten vorkommen, könnte dieser Umstand als Ausschlusskriterium für BNe verwendet werden, was ihr Einsatzgebiet drastisch einschränken würde. Da auch der Problembereich der PPS durch reellwertige oder ganzzahlige Variablen gekennzeichnet ist, sind Wege zum Diskretisieren reellwertiger Variablen und zur Aggregation diskreter Wahrscheinlichkeitsverteilungen zu reellen Werten in Abschnitt 4.4.3 aufgezeigt worden, und diese Wege sind im Softwaresystem zur Unterstützung der PPS mittels BNe beschritten bzw. umgesetzt worden. Somit ist der Einsatz des vorliegenden Softwaresystems grundsätzlich auch auf Gebieten zu empfehlen, auf denen reellwertige oder ganzzahlige Variablen vorkommen.

Kapitel 7

Empirische Untersuchung des Potenzials BNe zur Unterstützung der Reihenfolgeplanung

7.1 Ziele der empirischen Untersuchung

Die vorangegangenen Kapitel, insbesondere die Kapitel 4 und 5, haben das Potenzial BNe zur Unterstützung der PPS vorwiegend unter dem Aspekt der Theorie betrachtet. Die vorliegende empirische Untersuchung soll nun erlauben, das Potenzial BNe zur Unterstützung der PPS aus Sicht der Praxis einzuschätzen. Sie untersucht besagtes Potenzial also nicht analytisch, sondern ermittelt es empirisch anhand komplexer Beispiele für die Anwendung BNe zur PPS.

Die empirische Untersuchung ist hinsichtlich mehrerer „Dimensionsbündel“ variabel:

1. Es können die verschiedenen Teilaufgaben der PPS aus Abschnitt 2.2 einzeln oder in Kombination simultan gelöst werden, wobei sie sich zusätzlich in den konkreten Definitionen ihrer Eingaben, Ausgaben und Ziele unterscheiden.

2. Die Eingaben für die Instanzen der (Kombinationen von) Teilaufgaben der PPS, wie z. B. die betrachteten Fertigungsaufträge, die Rabattstaffelung eines Lieferanten oder der verfügbare Bestand, variieren darüber hinaus in Art und Umfang.

3. Es variieren die Eigenschaften der Produktion, wie z. B. die Anzahl der betrachteten Kapazitätseinheiten in der Produktion, oder der Organisationstyp, ob es sich also um Fließ- oder um Werkstattfertigung handelt.

4. Zudem können für die Wissensakquisition, -repräsentation und -anwendung im Kontext BNe verschiedene Algorithmen und Datenstrukturen eingesetzt werden (siehe Abschnitt 6.4), die sich hinsichtlich ihrer Ausgaben und Eigenschaften voneinander unterscheiden.

5. Besagten Algorithmen und Datenstrukturen haften darüber hinaus viele Parameter an, die ebenfalls veränderlich sind. So kann man beispielsweise beim Erstellen BNe aus Daten die Anzahl der Elternknoten begrenzen und bei der approximativen PI die Genauigkeit der berechneten Wahrscheinlichkeiten vorgeben.

Um die kombinatorische Vielfalt der möglichen Experimente zu begrenzen, soll während der empirischen Untersuchung eine der komplexesten und kompliziertesten Teilaufgaben der PPS herausgegriffen werden, wobei davon ausgegangen wird, dass, wenn BNe zur Lösung dieser schwierigen Teilaufgabe beitragen, sie auch das Potenzial aufweisen, die Lösung anderer und einfacherer Teilaufgaben der PPS zu unterstützen.

Eine externe Vorgabe lautet, eine Fertigung und eine Probleminstanz zu untersuchen, die bereits in vielen Arbeiten untersucht worden sind, und die Probleminstanz auch nur geringfügig zu verändern. So greift die empirische Untersuchung vorerst *nicht* auf die in Abschnitt 6.5.1 beschriebene Vorgehensweise zur Generierung von PPS-Daten zurück, sondern widmet sich nur besagter Probleminstanz, die sie aber, um eine gewisse statistische Sicherheit hinsichtlich der Qualität der Lösungen zu gewährleisten, vervielfältigt und deren Kopien sie stochastisch modifiziert. Die gewählte Probleminstanz gibt die Eigenschaften der Produktion ebenfalls fest vor.

Des weiteren soll die empirische Untersuchung nur ausgewählte Verfahren zur Akquisition, Repräsentation und Anwendung BNe herausgreifen und deren Parameter mit Standard- bzw. Vorgabewerten versehen. Verfahren zur Wissensverarbeitung im Kontext BNe sind nur insofern auszutauschen oder hinsichtlich ihrer Parameter zu verändern, als dass sie nach ihrer ersten Verwendung noch einer Feineinstellung bedürfen. So sollen beispielsweise keine Verfahren der approximativen PI angewendet werden, wenn die Rechenzeiten für die exakte PI im Kontext BNe bei der Lösung der ausgewählten Teilaufgabe der PPS akzeptabel sind.

Es wird explizit darauf geachtet, dass das jeweilige Verfahren der Wissensverarbeitung im Kontext BNe nur dann ausgewechselt und/oder seine

Konfiguration nur dann variiert wird, wenn das Resultat seines Einsatzes den Qualitätsansprüchen nicht genügt. Dass die Verfahren der Wissensverarbeitung im Kontext BNe und ihre möglichen Konfigurationen nicht in voller Breite empirisch untersucht werden, dient nämlich nicht ausschließlich dazu, die kombinatorische Explosion der möglichen Experimente einzudämmen, sondern auch dazu, zu belegen, dass kein Feineinstellen und keine mehrfachen Experimente nötig sind, um BNe erfolgreich zur Unterstützung der PPS einzusetzen. Nichtsdestotrotz erfolgt eine Variation der Parameter der Verfahren der Wissensverarbeitung im Kontext BNe, und es werden mehrere Experimente zu jeder vorliegenden Konfiguration der Verfahren durchgeführt, um gegebenenfalls Abhängigkeiten zwischen den Verfahren und den Resultaten der Experimente aufzuzeigen.

Die Resultate der Experimente sind hinsichtlich ihrer Güte und hinsichtlich des fixen und des variablen Aufwandes zu beurteilen, der getrieben worden ist, um sie zu erlangen. Beim fixen und variablen Aufwand handelt es sich um den Aufwand, der für das Erstellen des BNes, respektive den Aufwand, der für das Anwenden des BNes zur Lösung der ausgewählten Teilaufgabe der PPS angefallen ist. Anhand der Lösungsgüte und des Lösungsaufwandes muss die empirische Untersuchung es nun erlauben, das Potenzial BNe zur Lösung dieser Teilaufgabe zu beurteilen und es bezüglich der Lösung der anderen Teilaufgaben der PPS einzuschätzen.

7.2 Im Rahmen der empirischen Untersuchung zu lösendes Problem der PPS

7.2.1 Motivation zur Reihenfolgeplanung als im Rahmen der empirischen Untersuchung zu lösendes Problem

Wie bereits Abschnitt 4.6 zusammenfasst, eignen sich BNe und das Softwaresystem zur Unterstützung der PPS mittels BNe aus Kapitel 6 prinzipiell zur Unterstützung aller schlechtstrukturierten Teilaufgaben der PPS aus Abschnitt 2.6.2. BNe eignen sich besonders gut zur Prognose, zur Diagnose und zur Entscheidungsunterstützung, und alle strukturdefekten Probleme der PPS lassen sich als Probleme der Prognose, der Diagnose oder der Entscheidungsunterstützung auffassen. Somit könnte das Potenzial BNe zur Unterstützung der PPS anhand jedes schlechtstrukturierten Problems der PPS empirisch untersucht werden. Für eine empirische Untersuchung sind allerdings Daten vonnöten, welche nicht, in unzureichender Qualität, in zu

geringer Quantität oder nur für sehr spezielle Probleme zur Verfügung stehen. Wollte man BNe manuell für ausgewählte Probleminstanzen erstellen, fehlte zudem das notwendige Expertenwissen, und das manuelle Erstellen BNe für eine Vielzahl von Probleminstanzen wäre darüber hinaus sehr aufwendig, so intuitiv und einfach es im Einzelfall auch ist.

Die Reihenfolgeplanung sticht aus den schlechtstrukturierten Problemen der PPS aufgrund vierer Eigenschaften hervor, welche sie als das im Rahmen der empirischen Studie mittels BNe zu unterstützende Problem der PPS prädestinieren:

1. Für die Reihenfolgeplanung liegt bereits eine große Anzahl von Testprobleminstanzen vor, wie z. B. die Probleme von [FT63], an deren Struktur man sich orientieren kann. Allerdings tragen diese Instanzen meist statischen Charakter.

2. Für die Reihenfolgeplanung kann man unter plausiblen und allseits akzeptierten Annahmen relativ leicht Testdaten bzw. Testprobleminstanzen generieren. Probleminstanzen für beispielsweise die Primärbedarfsprognose zu generieren, scheint hingegen angreifbar. Abschnitt 6.5.1 beschreibt ein mögliches Konzept zur Generierung von Testdaten zur Reihenfolgeplanung und die Umsetzung dieses Konzeptes im Softwaresystem zur Unterstützung der PPS mittels BNe.

3. Es gelingt relativ leicht, aus Probleminstanzen der Reihenfolgeplanung Modelle für die ereignisdiskrete Simulation zu generieren, die ereignisdiskrete Simulation gegebenenfalls wiederholt ablaufen zu lassen und während der Simulationsläufe Prozessdaten vom Produktionssystem abzutasten. Die Komponente zur Akquisition BNe erstellt aus den abgetasteten Prozessdaten automatisch BNe, so dass kein Expertenwissen vorliegen muss, um die BNe für die empirische Studie zu erstellen, und die BNe auch nicht manuell erstellt werden müssen.

4. Nicht zuletzt ist für die in diesem Kapitel vorliegende empirische Untersuchung ein Problem der Reihenfolgeplanung extern vorgegeben worden.

Die vier aufgeführten Punkte lassen die Wahl auf die Reihenfolgeplanung als das schlechtstrukturierte Problem der PPS fallen, das im Rahmen der empirischen Untersuchung zu lösen bzw. dessen Lösung mittels BNe zu unterstützen ist.

Sicherlich wäre ein BN anschaulicher, das für eine konkrete Probleminstanz aus der Praxis manuell und anhand von Expertenwissen erstellt worden ist, aber für eine empirische Studie ist ein solches BN leider eher ungeeignet.

7.2.2 Charakteristika der Reihenfolgeplanung

Die Reihenfolgeplanung ist im Allgemeinen eines der schwierigsten Probleme der PPS (siehe Abschnitte 2.5 und 2.6.2). Die Reihenfolgeplanung legt für jede Maschine fest, in welcher Reihenfolge die Maschine die Fertigungsaufträge bearbeitet, welche die technologischen Maschinenfolgen bzw. die Arbeitspläne dieser Maschine zuordnen.

Die Ziele der Reihenfolgeplanung entsprechen den technizitären Ersatzzielen der PPS, die bereits Abschnitt 2.1 beschreibt und formalisiert: optimale Termintreue, minimale Durchlaufzeiten, minimaler Lagerbestand und maximale Kapazitätsauslastung. Die Nebenbedingungen der Reihenfolgeplanung – in Hinblick auf Menge, Zeit und Kapazität – stellt ebenfalls Abschnitt 2.1 zusammen.

Die Reihenfolgeplanung bestimmt die Termine, zu denen die Arbeitsgänge der Fertigungsaufträge auf den Maschinen beginnen und enden. Entweder legt sie diese Termine explizit fest, oder die Reihenfolge der Fertigungsaufträge impliziert diese Termine. Zusätzlich zu den speziell für die Reihenfolgeplanung entwickelten Verfahren sind die meisten der in Kapitel 3 aufgeführten universellen Verfahren mit unterschiedlichem Erfolg auf die Reihenfolgeplanung angewandt worden.

7.2.3 Prinzipielle Möglichkeiten der Unterstützung der Reihenfolgeplanung durch BNe

BNe bzw. die Verfahren der Wissensverarbeitung in ihrem Kontext können die Reihenfolgeplanung auf verschiedene Arten unterstützen. Denkbar ist zum einen eine direkte Unterstützung, bei der ein BN Termine oder Reihenfolgen explizit ermittelt, welche die Fertigung direkt übernimmt und umsetzt – vorausgesetzt, dass die Planung aktuell ist und noch keine Störungen aufgetreten sind.

Zum anderen ist eine indirekte Unterstützung denkbar, bei der ein BN Ausgaben erzeugt, die – angewandt – dazu führen, dass günstige Reihenfolgen und somit Termine entstehen. Die indirekte Unterstützung kann wiederum auf mehrere Arten bzw. über mehrere Stufen hinweg erfolgen: BNe unterstützen die Reihenfolgeplanung, indem sie der Fertigung Handlungsanweisungen geben, die, durch die Mitarbeiter in der Fertigung umgesetzt, zu Reihenfolgen und Terminen führen. BNe parametrisieren aber auch andere Verfahren zur PPS oder liefern ihnen Eingaben, so dass diese Verfahren Reihenfolgen und somit auch Termine besser oder überhaupt erst ermitteln. Kapitel 3 führt solche Verfahren zur PPS auf und und stellt ausgewählte vor.

Die direkte Unterstützung der PPS im Allgemeinen und durch Wissensverarbeitung im Kontext BNe im Speziellen weist folgende Nachteile auf:

1. Wie bereits angedeutet, führt eine direkte Unterstützung dazu, dass nach kurzer Zeit die vorgegebenen Mengen, Zeiten, Termine und Zuordnungen, selbst wenn sie zum Zeitpunkt der Planung in sich konsistent gewesen sind, in der Fertigung nicht mehr eingehalten bzw. umgesetzt werden können, da die Fertigung stochastischen Einflüssen unterliegt und es ihr Zustand bereits nach kurzer Zeit objektiv nicht mehr zulässt, die exakten Vorgaben umzusetzen.

2. Die direkte Unterstützung geht davon aus, dass „Lücken“ in Plänen möglich seien, die aber in der Realität unter organisatorischen Aspekten nur schwer durchzusetzen sind. So kann man beispielsweise einen Mitarbeiter nur schwer davon überzeugen, trotz des Umstandes, dass sich Arbeitsvorrat vor seiner Maschine staut, nicht zu produzieren, weil in Kürze ein wichtiger(er) Auftrag eintreffe, der dann nicht sofort bearbeitet werden könne und dessen Warten Verspätung hervorrufe. Auch dürfte es schwerfallen, einen Mitarbeiter davon zu überzeugen, einen Transportbehälter nur zu einem gewissen Teil zu füllen, weil sonst das Material für einen zukünftigen, sich noch nicht in der Fertigung befindenden Auftrag nicht mehr zur Verfügung stehe und der Mitarbeiter so den Plan gefährde. Die Resultate der direkten Unterstützung sind also organisatorisch nur schwer umzusetzen.

3. Unterstützt man die Reihenfolgeplanung direkt und gibt Termine für die Arbeitsgänge oder explizit Reihenfolgen für die Aufträge vor, so besteht zudem die Gefahr, dass ungültige Lösungen entstehen, welche obige Nebenbedingungen verletzen oder gar Dead Locks verursachen, bei denen Aufträge gegenseitig aufeinander warten und so sich und den Fortschritt weiter Teile der Produktion blockieren. Deadlocks sind keine ungültigen Lösungen im eigentlichen Sinne, da sie keine Nebenbedingungen verletzten. Deadlocks führen nur dazu, dass der Zielerreichungsgrad der Reihenfolgeplanung unendlich schlecht wird.

4. Die Zufallsvariablen BNe überspannen im Allgemeinen nur begrenzte, diskrete Wertebereiche, die Anzahl der Zufallsvariablen BNe ist im Allgemeinen konstant, und auch DBNe erstrecken sich nur über periodische Zeitintervalle. Bei einer direkten Unterstützung der PPS müssten BNe reellwertige Termine unter fortschreitender Zeit errechnen, Zuordnungen zwischen Elementen sich dynamisch verändernder Objektmengen vornehmen sowie reellwertige Mengen ermitteln, was

keinem intuitiven (Vorgehen beim) Einsatz BNe entspräche. Natürlich besteht die Möglichkeit, aus diskreten Wahrscheinlichkeitsverteilungen reelle Werte zu ermitteln (siehe Abschnitt 4.4.3) oder relative Termine zu bestimmen bzw. allgemein lediglich Änderungen zu propagieren, was aber in erster Instanz nicht notwendig ist.

Da die indirekte Unterstützung der Reihenfolgeplanung durch BNe die dargelegten Nachteile nicht oder nur z. T. aufweist, schlägt die empirische Untersuchung den Weg der indirekten Unterstützung ein. Eines der in Kapitel 3 analysierten Verfahren zu parametrisieren, ist zu weit von der PPS entfernt, als dass es im Kontext der vorliegenden Arbeit untersucht werden sollte, zumal besagte Verfahren nicht PPS-spezifisch sind. Somit ermittelt die empirische Untersuchung das Potenzial BNe zur Unterstützung der PPS, indem sie BNe dazu verwendet, der Fertigung Handlungsanweisungen vorzugeben, die, durch Mitarbeiter in der Produktion umgesetzt, implizit zu Reihenfolgen und somit auch zu Terminen führen.

7.2.4 Prioritätsregeln zur Reihenfolgeplanung

Prioritätsregeln sind einfache, nicht-optimierende Heuristiken, welche in Entscheidungssituationen, in denen zwei oder mehr Fertigungsaufträge um eine Ressource konkurrieren, festlegen, welcher Fertigungsauftrag die Ressource zuerst belegt. Sie priorisieren also einen der Fertigungsaufträge, der dann zuerst bearbeitet wird.[1] [Cor04, S. 504 ff.] und [Kur05, S. 166 ff.] stellen die gebräuchlichsten Prioritätsregeln vor und diskutieren Prioritätsregeln und ihre Wirkung ausführlich. Die tendenzielle Wirkung vierer Prioritätsregeln auf die vier technizitären Ersatzziele der PPS gibt Tabelle 7.1 an.

7.2.5 Situationsabhängige Auswahl von Prioritätsregeln mittels DBNe

Wie die Abschnitte des Kapitels 7 bis hierher nahelegen, ist als Problem, anhand dessen das Potenzial BNe zur Unterstützung PPS aufgezeigt werden soll, die situationsabhängige Auswahl von Prioritätsregeln mittels DBNe ausgewählt worden. Die situationsabhängige Auswahl von Prioritätsregeln mittels DBNe verwendet grundsätzlich die für eine Simulationsstudie bzw. für eine Experimentenreihe erforderlichen Komponenten und Subkomponenten, auf die bereits Abschnitt 6.5.2 ausführlich eingeht. Die situationsabhängige

[1] Prioritätsregeln sind also keine Produktionsregeln wie die in Abschnitt 3.4 vorgestellten.

Tabelle 7.1: Vier ausgewählte Prioritätsregeln bzw. Kriterien zur Auswahl des Fertigungsauftrages, der als nächstes zu bearbeiten ist, und qualitative Aussagen zur Wirkung besagter Prioritätsregeln auf die vier technizitären Ersatzziele der PPS (angelehnt an [Cor04, S. 506])

Ziel	Prioritätsregel bzw. Auswahlkriterium			
	Kürzeste Operations-zeit	Kürzeste Fertigungs-restzeit	Höchster dynami-scher Wert	Geringste Schlupfzeit
Minimale Termin-abweichung	schlecht	mäßig	mäßig	sehr gut
Minimale Durchlauf-zeit	sehr gut	gut	mäßig	mäßig
Minimale Kapital-bindung	gut	mäßig	sehr gut	mäßig
Maximale Kapazitäts-auslastung	sehr gut	gut	mäßig	gut

Auswahl von Prioritätsregeln mittels DBNe läuft prinzipiell als Simulationsstudie bzw. als Experimentenreihe ab, wie sie Anhang B.5 anhand eines UML-Kollaborationsdiagrammes detailliert beschreibt.

Nichtsdestotrotz legt dieser Abschnitt noch einmal den Ablauf einer situationsabhängigen Auswahl von Prioritätsregeln mittels DBNe auf höherem Abstraktionsniveau im Rahmen einer Simulationsstudie bzw. einer Experimentreihe dar. Der Ablauf gliedert sich in drei Phasen:

1. *Abtasten der Produktion*: Im ersten Schritt werden zyklisch mehrere Zustandsvariablen der Produktion bzw. ihres Simulationsmodelles abgetastet. Aus dem Abtasten resultieren Fälle bzw. eine multivariate Zeitreihe, die den Verlauf der Produktion für den Zeitraum des Abtastens beschreibt. Die abgetasteten Zustandsvariablen umfassen zwingend die anliegenden Prioritätsregeln und die technizitären Ersatzziele der PPS.

2. *Akquisition des DBNes*: Im zweiten Schritt wird das DBN aus der im ersten Schritt ermittelten multivariaten Zeitreihe erstellt und entrollt (siehe Definition 4.5).

3. *Anwenden des DBNes*: Im dritten Schritt erfolgt das ebenfalls zyklische Anwenden des DBNes: Der derzeitige Zustand der Produktion sowie die für die Zukunft erwünschten Ausprägungen der technizitären Ersatzziele werden im DBN als Evidenz gesetzt (siehe Abschnitt 5.2.2), und mittels PI schließt das DBN auf die Prioritätsregeln, welche – sofern angewandt – die für die Zukunft erwünschten Ausprägungen der technizitären Ersatzziele beim derzeitigen Zustand der Produktion mit höchster Wahrscheinlichkeit hervorrufen. Die ermittelten Prioritätsregeln werden in der Produktion angewandt, bis ein erneutes Anwenden des DBNes nach einem festen Zeitintervall andere Prioritätsregeln vorgibt.

Im Rahmen der Experimente und zu Kontrollzwecken wird die Phase 1, das Abtasten der Produktion, auch während des Anwendens des DBNes weitergeführt, um den Einfluss zu bestimmen, den die situationsabhängige Auswahl von Prioritätsregeln mittels DBNe auf die technizitären Ersatzziele ausübt, und um die Ausprägungen der technizitären Ersatzziele vor und während des Anwendens des DBNes miteinander zu vergleichen.

Abbildung 7.1 stellt noch einmal schematisch dar, wie DBNe bzw. Methoden der Wissensverarbeitung in ihrem Kontext Prioritätsregeln situationsabhängig zur Reihenfolgeplanung auswählen.

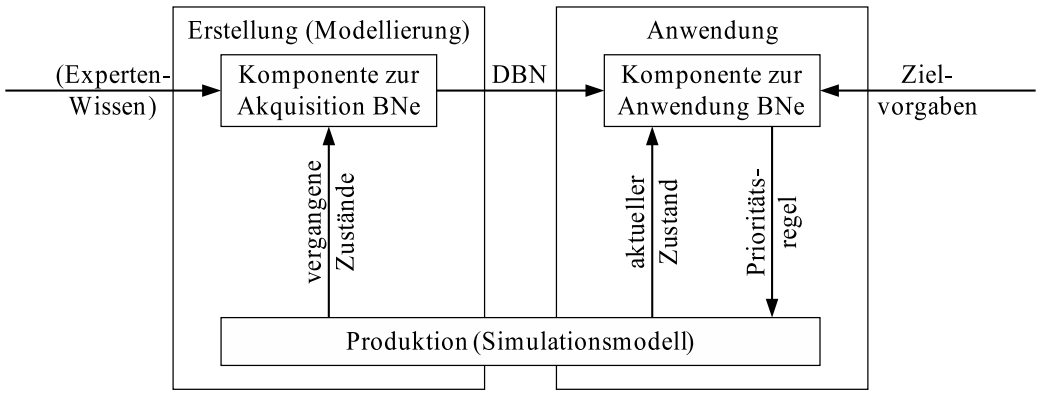

Abbildung 7.1: Situationsabhängige Auswahl von Prioritätsregeln mittels DBNe zur Reihenfolgeplanung

Das DBN wählt genau eine von sechs einfachen[2] Prioritätsregeln, die zur Reihenfolgeplanung als Handlungsalternativen zur Auswahl stehen, und gibt sie für alle Maschinen in der Fertigung vor:

1. „Frühester Fertigstellungstermin“: Derjenige Auftrag wird als erster bearbeitet, dessen gewünschter Fertigstellungstermin der früheste ist.

2. „First Come, First Served“: Derjenige Auftrag wird zuerst bearbeitet, der als erster an der Maschine eingetroffen ist.

3. „Höchster dynamischer Wert“: Derjenige Auftrag wird als erster bearbeitet, der im Verlaufe seiner bisherigen Bearbeitung den höchsten (Material)wert

$$w = w_s + \sum_{i=0}^{k-1} \Delta w_i \tag{7.1}$$

erlangt hat, wobei w den dynamischen Wert des Auftrages, w_s den Initialwert des Auftrages, k den Index des aktuellen, noch nicht bearbeiteten Arbeitsganges des Auftrags und Δw_i den Wertezuwachs bezeichnen, den der i-te Arbeitsgang zum Wert des Auftrags beigetragen hat.[3]

[2]Im Rahmen der Experimente ist ausdrücklich nicht auf komplexe Prioritätsregeln, welche z. B. einfache Prioritätsregeln additiv oder multiplikativ verknüpfen, zurückgegriffen worden, weil nicht Prioritätsregeln untersucht werden sollen, sondern in erster Linie das Potenzial BNe zur Unterstützung der PPS. Dem Einsatz anderer Prioritätsregeln, die womöglich besser sind als die hier verwendeten, steht natürlich nichts entgegen.

[3]Aus Gründen der Vereinfachung sind während der Experimente immer ein Initialwert $w_s = 0$ und $\Delta w_i \leftarrow t_{b_i}$ angenommen worden, wobei t_{b_i} die Bearbeitungszeit des i-ten Arbeitsganges des Fertigungsauftrages bezeichne.

4. „Kürzeste Operationszeit“: Derjenige Auftrag wird zuerst bearbeitet, dessen Arbeitsgang auf der Maschine die geringste Bearbeitungszeit aufweist.

5. „Kürzeste Fertigungsrestzeit“: Derjenige Auftrag wird als erster bearbeitet, dessen Fertigungsrestzeit

$$t_r = \sum_{i=k}^{n-1} t_{b_i} \tag{7.2}$$

am geringsten ist, wobei t_r die Fertigungsrestzeit des Auftrages, n die Anzahl der Arbeitsgänge des Auftrages, k den Index des aktuellen, noch nicht bearbeiteten Arbeitsganges des Auftrages und t_{b_i} die Bearbeitungszeit des i-ten Arbeitsganges des Auftrages bezeichnen.[4]

6. „Kürzeste Schlupfzeit“: Derjenige Auftrag wird zuerst bearbeitet, dessen Schlupfzeit

$$t_s = t_e - t_r - t_{\text{now}} \tag{7.3}$$

am geringsten ist, wobei t_s die Dauer des Schlupfes, t_e den gewünschten Fertigstellungstermin des Auftrages, t_r die Fertigungsrestzeit des Auftrages und t_{now} den aktuellen Zeitpunkt bezeichnen.

Wird eine Maschine frei und stehen mehrere alternative Aufträge zur Bearbeitung an, erfolgt die Auswahl des als nächsten zu bearbeitenden Auftrages aufgrund der aktuell für die Maschine bzw. für die gesamte Produktion vorgegebenen Prioritätsregel. Sofern sich die Prioritäten der Aufträge mit der Zeit ändern, werden sie vor jeder Auswahl neu berechnet.

Für welche Fertigung BNe Prioritätsregeln auswählen, beschreibt der folgende Abschnitt.

7.3 Beschreibung der PPS-Testdaten für die empirische Untersuchung

7.3.1 Charakterisierung des „11-Maschinen-Problems“ unter Organisations- und Datenaspekten

Um eine gewisse Vergleichbarkeit mit anderen empirischen Studien zu gewährleisten, soll zumindest in erster Instanz nicht die in Abschnitt 6.5.1 beschriebene Komponente zur Generierung von PPS-Daten verwendet, sondern

[4] Der erste Arbeitsgang ist implizit mit 0 indiziert worden, so dass der Index des letzten Arbeitsganges nicht n, sondern $n-1$ lautet.

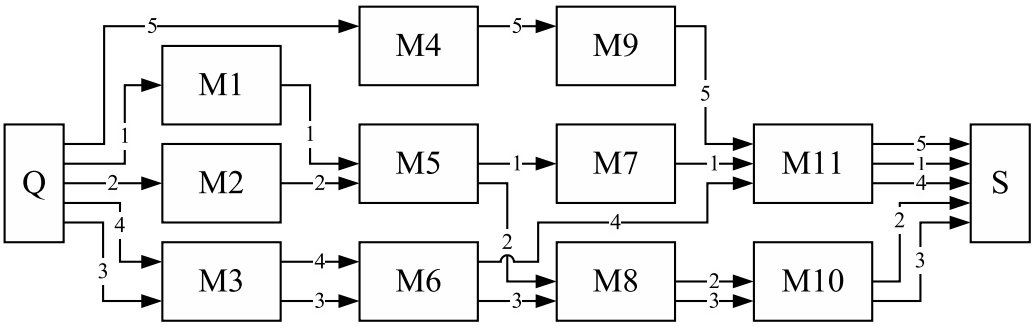

Abbildung 7.2: Das Produktionssystem und die fünf alternativen Maschinenfolgen des „11-Maschinen-Problems“. Die Buchstaben Q, M und S stehen für Quelle, Maschine respektive Senke.

Tabelle 7.2: Wahrscheinlichkeitsverteilung der Zwischenankunftszeiten t_a der Fertigungsaufträge beim „11-Maschinen-Problem“

t_a	$\Pr(t_a)$
10 min	0,25
20 min	0,60
30 min	0,15

auf eine Probleminstanz zurückgegriffen werden, die bereits in mehreren empirischen Studien beschrieben und untersucht worden ist, so unter anderem in [Sch02b, S. 245 ff.]. Die Probleminstanz geht auf [Bub93] zurück, ist unter dem Namen „11-Maschinen-Problem“ bekannt und wird im weiteren Verlaufe der vorliegenden Arbeit mit diesem Namen bezeichnet.

Das „11-Maschinen-Problem“ gestaltet sich wie folgt (siehe auch Abbildung 7.2): Ein Produktionssystem enthalte elf Maschinen. Nach zufälligen, aber festen Zwischenankunftszeiten t_a treffen Fertigungsaufträge ein. Die diskrete Wahrscheinlichkeitsverteilung der Zwischenankunftszeiten der Fertigungsaufträge stellt Tabelle 7.2 dar.

Jedem eintreffenden Fertigungsauftrag werde – ebenfalls zufällig – einer von fünf alternativen Arbeitsplänen a zugeordnet. Tabelle 7.3 stellt die Maschinenfolgen dar und gibt die durch die Arbeitsplanpositionen vorgegebenen Bearbeitungszeiten t_b der Fertigungsaufträge bzw. Arbeitsgänge auf den Maschinen m an. Die Maschinenfolgen eines jeden Fertigungsauftrages und die Bearbeitungszeiten seiner Arbeitsgänge auf den einzelnen Maschinen seien ausschließlich vom Arbeitsplan, der dem Fertigungsauftrag zugeordnet worden ist, und dessen Arbeitsplanpositionen abhängig. Abbildung 7.2 stellt das

Tabelle 7.3: Arbeitspläne mit Maschinenfolgen und Bearbeitungszeiten t_b für das „11-Maschinen-Problem“

Arbeitsplanposition	1		2		3		4	
Arbeitsplan bzw. Produkt	m	t_b	m	t_b	m	t_b	m	t_b
1	1	10	5	12	7	25	11	10
2	2	15	5	10	8	12	10	20
3	3	10	6	10	8	10	10	10
4	3	20	6	35	11	5	–	–
5	4	12	9	5	11	15	–	–

Tabelle 7.4: Wahrscheinlichkeiten der Arbeitspläne, die den ankommenden Fertigungsaufträgen zugeordnet zu werden

Arbeitsplan	Auftrittswahrscheinlichkeit
1	0,3
2	0,1
3	0,2
4	0,1
5	0,3

Produktionssystem und die fünf alternativen Maschinenfolgen des „11-Maschinen-Problems“ grafisch dar.

Tabelle 7.4 gibt die Wahrscheinlichkeiten der Arbeitspläne an, den ankommenden Fertigungsaufträgen zugeordnet zu werden.

Die Bediensysteme (Maschinen) verhalten sich streng nicht-abweisend, das heißt, dass sämtliche Fertigungsaufträge auf ihre Bedienung warten, wenn die Maschinen gerade andere Fertigungsaufträge bearbeiten, und dass ihre Wartezeit nicht nach oben beschränkt ist.

Alle über das „11-Maschinen-Problem“ getroffenen Aussagen zu einem Beispiel kombiniert, trifft mit einer Wahrscheinlichkeit von 0,25 alle 10 min ein Fertigungsauftrag ein, dem mit einer Wahrscheinlichkeit von 0,3 der Arbeitsplan 1 zugeordnet wird, so dass der Fertigungsauftrag über die Maschinen 1, 5, 7 und 11 läuft und auf besagten Maschinen 10, 12, 25 respektive 10 min bearbeitet wird.

7.3.2 Analyse der Maschinenauslastung[5] beim „11-Maschinen-Problem“

Die Zwischenankunftszeit der Fertigungsaufträge sei beliebig verteilt mit dem Mittelwert $\overline{t_a}$ und der Ankunftsrate

$$\lambda = \frac{1}{\overline{t_a}}. \tag{7.4}$$

Handelt es sich bei der Verteilung der Zwischenankunftszeit wie beim „11-Maschinen-Problem“ um eine Stufenfunktion mit i Stufen, so gilt

$$\overline{t_a} = \sum_i p_{t_{a_i}} t_{a_i}, \tag{7.5}$$

wobei t_{a_i} die der i-ten Stufe entsprechende Zwischenankunftszeit darstellt und $p_{t_{a_i}}$ die Eintrittwahrscheinlichkeit der Zwischenankunftszeit t_{a_i} verkörpert. Konkret gelten

$$\overline{t_a} = 0{,}25 \cdot 10\,\text{min} + 0{,}6 \cdot 20\,\text{min} + 0{,}15 \cdot 30\,\text{min} = 19\,\text{min} \tag{7.6}$$

und somit

$$\lambda = \frac{1}{19\,\text{min}} \approx \frac{0{,}0526}{\text{min}} \approx \frac{3{,}16}{\text{h}}, \tag{7.7}$$

was heißt, dass in einer Stunde im Mittel rund 3,16 Aufträge in der Fertigung eintreffen bzw. freigegeben werden.

Die arbeitsplanspezifischen Ankunftsraten $\vec{\lambda}$ ergeben sich als Produkt aus der Gesamtankunftsrate λ und den relativen Häufigkeiten $\vec{p}$ der Arbeitspläne. Für das „11-Maschinen-Problem“ gilt

$$\vec{\lambda} = \lambda\vec{p} \approx \frac{3{,}16}{h} \begin{pmatrix} 0{,}3 \\ 0{,}1 \\ 0{,}2 \\ 0{,}1 \\ 0{,}3 \end{pmatrix} \approx \begin{pmatrix} 0{,}947\,\text{h}^{-1} \\ 0{,}316\,\text{h}^{-1} \\ 0{,}632\,\text{h}^{-1} \\ 0{,}316\,\text{h}^{-1} \\ 0{,}947\,\text{h}^{-1} \end{pmatrix}, \tag{7.8}$$

wobei $\vec{\lambda}$ der Vektor der arbeitsplanspezifischen Ankunftsraten ist und $\vec{p}$ der Vektor der relativen Häufigkeiten der Auftragstypen. Das heißt, dass pro Stunde durchschnittlich z. B. 0,947 Aufträge mit dem Arbeitsplan 1 in der Fertigung eintreffen (siehe erstes Element des letzten Vektors in Formel 7.8).

[5] Die Analyse des besagten Problems ist in Zusammenarbeit mit Herrn Doktor rer. pol. Sven Völker entstanden, dem an dieser Stelle noch einmal herzlicher Dank gilt.

Jeder eintreffende Fertigungsauftrag ist Träger einer genau bemessenen Kapazitätsnachfrage für alle Maschinen, die in seinem Arbeitsplan aufgeführt worden sind. Es gebe $t_{b,n,m}$ die Kapazitätsnachfrage eines Auftrags mit dem Arbeitsplan n bei Maschine m an. Beispielsweise fragt ein Auftrag mit dem Arbeitsplan 1 eine Kapazität bei Maschine 1 von $t_{b,1,1} = 10\,\mathrm{min}$ nach (siehe erstes Element der Matrix in Formel 7.9).

Die Matrix $\vec{T_b}$ der Kapazitätsnachfragen[6] nimmt für das „11-Maschinen-Problem“ folgende Gestalt an:

$$\vec{T_b} = \begin{pmatrix} 10 & 0 & 0 & 0 & 12 & 0 & 25 & 0 & 0 & 0 & 10 \\ 0 & 15 & 0 & 0 & 10 & 0 & 0 & 12 & 0 & 20 & 0 \\ 0 & 0 & 10 & 0 & 0 & 10 & 0 & 10 & 0 & 10 & 0 \\ 0 & 0 & 20 & 0 & 0 & 35 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 5 & 0 & 15 \end{pmatrix}. \tag{7.9}$$

Falls – wie beim „11-Maschinen-Problem“ – eine Maschine in einem Arbeitsplan maximal einmal auftritt, entsprechen die Elemente der Matrix $\vec{T_b}$ den Bedienzeiten t_b.

Die Elemente $u_{b,m}$ des Zeilenvektors

$$\vec{u_b} = \lambda^T \vec{T_b} \tag{7.10}$$

geben die Kapazitätsnachfrage an, welche an Maschine m pro Zeiteinheit im Mittel anfällt. Für das „11-Maschinen-Problem“ gilt

$$\vec{u_b} \approx \begin{pmatrix} 0{,}947\,\mathrm{h}^{-1} \\ 0{,}316\,\mathrm{h}^{-1} \\ 0{,}632\,\mathrm{h}^{-1} \\ 0{,}316\,\mathrm{h}^{-1} \\ 0{,}947\,\mathrm{h}^{-1} \end{pmatrix}^T \cdot \vec{T_b} \approx \begin{pmatrix} 9{,}474\,\mathrm{min/h} \\ 4{,}737\,\mathrm{min/h} \\ 12{,}632\,\mathrm{min/h} \\ 11{,}368\,\mathrm{min/h} \\ 14{,}526\,\mathrm{min/h} \\ 17{,}368\,\mathrm{min/h} \\ 23{,}684\,\mathrm{min/h} \\ 10{,}105\,\mathrm{min/h} \\ 4{,}737\,\mathrm{min/h} \\ 12{,}632\,\mathrm{min/h} \\ 25{,}263\,\mathrm{min/h} \end{pmatrix}^T \approx \begin{pmatrix} 0{,}158 \\ 0{,}079 \\ 0{,}211 \\ 0{,}189 \\ 0{,}242 \\ 0{,}289 \\ 0{,}395 \\ 0{,}168 \\ 0{,}079 \\ 0{,}211 \\ 0{,}421 \end{pmatrix}^T. \tag{7.11}$$

Die Elemente $u_{b,m}$ von $\vec{u_b}$ tragen im Beispiel die Einheit min/h. Werden sie in h/h umgerechnet, so verlieren sie ihre Einheit und entsprechen — im eingeschwungenen Zustand — der Auslastung der Maschinen. Engpass

[6] Die Elemente der Matrix sind in Minuten angegeben und ihre Einheiten aus Platzgründen weggelassen worden.

ist demnach Maschine 11 mit einer Auslastung von rund 0,421 bzw. 42,1 % (siehe letztes Element im letzten Vektor in Formel 7.11).

Es sei

$$m_{max} = \text{argmax}(\vec{u_b}) = 11 \tag{7.12}$$

die Engpassmaschine. Der Zusammenhang zwischen mittlerer Zwischenankunftszeit $\overline{t_a}$ und Engpassauslastung $u_{b,m_{max}}$ lautet

$$u_{b,m_{max}} = \vec{\lambda}^T \cdot \vec{t_{b,m_{max}}} = \lambda \vec{p}^T \cdot \vec{t_{b,m_{max}}}, \tag{7.13}$$

wobei $\vec{t_{b,m_{max}}}$ den m_{max}-ten Spaltenvektor der Matrix $\vec{T_b}$ repräsentiert und $\vec{p}$ wiederum den Vektor der relativen Auftragshäufigkeiten (siehe Formel 7.8).

Umstellen der Formel 7.4 ergibt

$$\overline{t_a} = \frac{1}{\lambda}, \tag{7.14}$$

und Umstellen der Formel 7.13 führt zu

$$\lambda = \frac{u_{b,m_{max}}}{\vec{p}^T \cdot \vec{t_{b,m_{max}}}}. \tag{7.15}$$

Formel 7.15 in Formel 7.14 eingesetzt, ergibt die mittlere Zwischenankunftszeit als

$$\overline{t_a} = \frac{\vec{p}^T \cdot \vec{t_{b,m_{max}}}}{u_{b,m_{max}}}. \tag{7.16}$$

Für das „11-Maschinen-Problem“ bedeutet das

$$\begin{aligned} \overline{t_a} &\approx \frac{0{,}3 \cdot 10\,\text{min} + 0{,}1 \cdot 0\,\text{min} + 0{,}2 \cdot 0\,\text{min} + 0{,}1 \cdot 5\,\text{min} + 0{,}3 \cdot 15\,\text{min}}{0{,}421} \\ &\approx \frac{8\,\text{min}}{0{,}421} \\ &\approx 19\,\text{min}, \end{aligned} \tag{7.17}$$

was aus Formel 7.17 bereits bekannt ist. Soll die Engpassauslastung $u_{b,m_{max}}$ nun beispielsweise 85 % betragen, so muss eine mittlere Zwischenankunftszeit von

$$\overline{t_a} = \frac{8\,\text{min}}{0{,}85} \approx 9\,\text{min}\,25\,\text{s} \tag{7.18}$$

gewählt werden. Eine Engpassauslastung von

$$u_{b,m_{max}} \geq 1 \tag{7.19}$$

zu wählen, ist nicht sinnvoll, weil das Bediensystem dann vollliefe.

Tabelle 7.5: Maschinenübergangsmatrix mit den Übergangswahrscheinlichkeiten p_{ij} von Maschine i zu Maschine j für das „11-Maschinen-Problem“

Von Maschine	... zu Maschine											
	1	2	3	4	5	6	7	8	9	10	11	0
0	0,3	0,1	0,3	0,3	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0,75	0,25	0	0	0	0
6	0	0	0	0	0	0	0	$0,\overline{6}$	0	0	$0,\overline{3}$	0
7	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	1

7.3.3 Organisationsgrad des „11-Maschinen-Problems“

Der Vollständigkeit halber soll für das „11-Maschinen-Problem“ der Organisationsgrad o berechnet werden. Auch wenn der Organisationsgrad o im Rahmen der vorliegenden empirischen Studie nicht variiert wird, vermittelt er einen Eindruck davon, wo sich das „11-Maschinen-Problem“ im Bereich zwischen reiner Werkstatt- und reiner Fließfertigung bewegt. Aus den Maschinenfolgen, die Tabelle 7.3 und Abbildung 7.2 angeben, sowie aus den Wahrscheinlichkeiten dieser Maschinenfolgen bzw. der Arbeitspläne, die Tabelle 7.4 angibt, ist die Matrix der Maschinenübergangswahrscheinlichkeiten p_{ij} erstellt worden. Tabelle 7.5 beschreibt diese Matrix, aus der sich der Organisationsgrad o nach Formel 6.1 leicht berechnen lässt. Das „11-Maschinen-Problem“ weist einen Organisationsgrad von $o \approx 0{,}79$ auf, was auf eine Werkstattfertigung mit einer Tendenz des Problems zur Fließfertigung schließen lässt. Da in der Realität ein Organisationsgrad von unter 0,5 eher selten auftritt und die Maschinenfolgen der Aufträge bzw. Arbeitspläne nicht gänzlich gleich sind, kann man beim „11-Maschinen-Problem“ getrost von Werkstattfertigung sprechen.

Mittlere Zwischenankunftszeit $\overline{t_a}$	Durchschnittliche Engpassauslastung $u_{b,m_{max}}$
10	$0,80$
11	$0,73$
12	$0,67$
13	$0,62$

Tabelle 7.6: Alternative mittlere Zwischenankunftszeiten und zugehörige durchschnittliche Engpassauslastungen

7.3.4 Modifikation der Auslastung beim „11-Maschinen-Problem“

Erste Simulationsexperimente haben gezeigt, dass die Produktion beim „11-Maschinen-Problem“ so gering ausgelastet ist, dass die Produktionssteuerung grundsätzlich keine erkennbare Wirkung auf die technizitären Ersatzziele ausübt. Damit DBNe mit der situationsabhängigen Auswahl von Prioritätsregeln zur Reihenfolgeplanung überhaupt eine Möglichkeit haben, die technizitären Ersatzziele zu beeinflussen, muss die Auslastung der Produktion beim „11-Maschinen-Problem“ erhöht werden, ohne die Struktur des Problems sonst zu verändern.

Die Analyse, die Abschnitt 7.3.2 beschreibt, bestimmt die Auslastung der Produktion des „11-Maschinen-Problems“: Die Produktion ist mit einer durchschnittlichen Enpassauslastung von 42,1 % und einer durchschnittlichen Gesamtauslastung von 22,2 % so gering ausgelastet, dass sie sich unterhalb der unteren Grenze des Auslastungsbereichs bewegt, in dem die Fertigungssteuerung bzw. -regelung eine Produktion überhaupt merklich beeinflussen kann.

Damit die Fertigungssteuerung im Allgemeinen und somit auch DBNe mit der situationsabhängigen Auswahl von Prioritätsregeln zur Reihenfolgeplanung im Speziellen die technizitären Ersatzziele überhaupt beeinflussen können, ist die globale Zwischenankunftszeit beim „11-Maschinen-Problem“ für die Experimente so variiert worden ist, dass sich die Engpassauslastung zwischen 0,62 und 0,85 bewegt. Tabelle 7.6 führt die mittleren Zwischenankunftszeiten $\overline{t_a}$ sowie die dazugehörigen durchschnittlichen Engpassauslastungen $u_{b,m_{max}}$ auf.

Um das „11-Maschinen-Problem“ nicht stärker als nötig und nicht in seiner Struktur zu verändern, sind der Produktmix bzw. die relativen Häufigkeiten $\vec{p}$ der Arbeitspläne und die Arbeitspläne selbst nicht modifiziert worden.

7.4 Auswahl und Konfiguration der Methoden zur Wissensverarbeitung im Kontext BNe

7.4.1 Auswahl der Wissensrepräsentationsform

Als Wissensrepäsentationsform stehen für die empirische Studie nur BNe und DBNe zur Disposition. Da es sich bei der Fertigung, die geregelt werden soll, um einen stochatischen Prozess handelt und sich DBNe zur Abbildung und Regelung stochastischer Prozesse eignen, fällt die Wahl der Wissenrepräsentationsform auf DBNe.

Für entrollte DBNe ist noch festzulegen, über wie viele Zeitscheiben sie sich erstrecken sollen. Solange nicht die Notwendigkeit besteht, die DBNe über mehr als zwei Zeitscheiben $V[0]$ und $V[1]$ auszudehnen, sollen sie nur genau diese zwei Zeitscheiben beinhalten. Der Aufwand für die PI über den DBNen bleibt dann verhältnismäßig gering, und wenn bereits DBNe, die sich nur über zwei Zeitscheiben erstrecken, die Reihenfolgeplanung erfolgreich unterstützen, belegt das das Potenzial BNe zur Unterstützung der PPS nur umso stärker.

7.4.2 Auswahl und Konfiguration des Verfahrens zur Wissensakquisition

Zur automatischen Wissensakquisition stellt das Softwaresystem zur Unterstützung der PPS mittels BNe zwei Gütemaße zur Verfügung: das Bayes'sche Gütemaß (siehe Formel 4.14) sowie die minimale Beschreibungslänge (siehe Formel 4.15). [Bou95, S. 68 ff.] weist analytisch nach, dass beide Gütemaße zwei beliebige BNe unterschiedlicher Struktur immer insofern gleich bewerten, als dass sie jeweils dasselbe dem anderen vorziehen, was die Gütemaße äquivalent macht. Da beide Gütemaße bei ihrer Anwendung zudem einen ähnlich hohen Rechenaufwand verursachen, braucht für die Experimente im Rahmen der empirischen Studie ausschließlich eines von beiden verwendet zu werden. Die Wahl ist auf das ältere, anerkannte und weiter verbreitete Bayes'sche Gütemaß gefallen.

Zur automatischen Wissensakquisition bietet das Softwaresystem zur Unterstützung der PPS mittels BNe drei Suchalgorithmen an: den K2-Algorithmus (siehe Algorithmus A.1), den B-Algorithmus (siehe Algorithmus A.2) sowie Rejection-Free Annealing (siehe Abschnitt 6.4.2). Der K2-Algorithmus erfordert als Eingabe eine kausale Reihenfolge der Zufallsvariablen des BNes. Da diese Reihenfolge nicht eindeutig und nicht automatisch angegeben werden kann, kommt der K2-Algorithmus trotz seines geringen Rechenaufwandes

nicht für die Experimente im Rahmen der empirischen Studie in Frage.

Rejection-Free Annealing verursacht als universeller Algorithmus zur kombinatorischen Optimierung relativ hohen Rechenaufwand, brächte ein Abbruchkriterium als weiteren komplexen Parameter ein, und seine Laufzeit ist in der Praxis nur schwer abzuschätzen. Trotz seiner vermutlich guten Resultate, welche die der anderen zwei Suchalgorithmen, die sich „gierig“ verhalten, sicherlich übertreffen werden, soll er vorerst nicht und nur dann zum Einsatz gelangen, wenn die anderen Algorithmen keine akzeptablen Ergebnisse erzielen.

Der B-Algorithmus verkörpert einen guten Kompromiss zwischen Ergebnisqualität und Rechenaufwand und bedarf auch keiner kausalen Vorsortierung der Zufallsvariablen, weshalb die Wahl des Suchalgorithmus für die Experimente im Rahmen der empirischen Studie auf ihn fällt.

Zur Berechnung der bedingten Wahrscheinlichkeiten ist die Formel 4.19 verwendet worden.

Aus über 50 für die Beschreibung des Zustandes der Fertigung potenziell in Frage kommenden Variablen sind mittels einer einfachen Korrelationsanalyse bzw. mittels einer Korrelationsmatrix stark korrelierende Variablen aussortiert worden. Die verbliebenen 22 Variablen[7] liegen einem jeden BN zugrunde, das während der Experimente im Rahmen der empirischen Studie automatisch akquiriert worden ist:

1. der dynamische Wert (in Bearbeitungszeit) aller Aufträge im System,
2. die Anzahl der Arbeitsgänge, die während des letzten Intervalls eingegangen sind,
3. die Summe der Arbeit (in Bearbeitungszeit), die während des letzten Intervalls eingegangen ist,
4. die Durchlaufzeit, gemittelt über alle Aufträge, die während des letzten Intervalls die Fertigung verlassen haben,
5. die Terminabweichung, gemittelt über alle Aufträge, die während des letzten Intervalls die Fertigung verlassen haben,
6. die Anzahl der Arbeitsgänge, die sich in Warteschlangen befinden,
7. die Anzahl der Arbeitsgänge in der Fertigung,

[7] Es ist darauf verzichtet worden, die Variablen an dieser Stelle formal zu beschreiben, da sie nicht weiter verrechnet werden, sondern ihre Werte der Komponente zur automatischen Akquisition BNe aus Daten lediglich als Eingabe dienen.

8. die Anzahl der Aufträge, die sich in Warteschlangen befinden,
9. die Anzahl der Arbeitsgänge, die während des letzten Intervalls die Fertigung verlassen haben,
10. die Summe der Arbeit (in Bearbeitungszeit), die während des letzten Intervalls die Fertigung verlassen hat,
11. die Summe der Stillstandszeiten aller Maschinen während des letzten Intervalls,
12. die Summe der Zeiten, die alle Maschinen während des letzten Intervalls aktiv gewesen sind,
13. die Summe der Wartezeiten, die während des letzten Intervalls angefallen sind,
14. die Prioritätsregel, die während des letzten Intervalls angewandt worden ist,
15. die Anzahl der bereits fertiggestellten Arbeitsgänge in der Fertigung,
16. die Summe der bereits erledigten Arbeit (in Bearbeitungszeit) in der Fertigung,
17. die Summe aller Schlupfzeiten in der Fertigung,
18. die Anzahl aller Arbeitsgänge in der Fertigung, die noch bearbeitet werden müssen,
19. die Summe aller Arbeit (in Bearbeitungszeit) in der Fertigung, die noch zu erledigen ist,
20. die Auslastung der Fertigung,
21. die Summe aller Arbeit (in Bearbeitungszeit) in den Warteschlangen und
22. die Summe aller Arbeit (in Bearbeitungszeit) in der Fertigung.

Reellwertige und ganzzahlige Variablen sind in drei bis fünf Zustände diskretisiert worden, und die Anzahl der Eltern pro Variable ist beschränkt worden, so dass die Ausdehnung der hochdimensionalen Potenzialfunktionen nicht den für die PI zur Verfügung stehenden Hauptspeicher sprengt. Die für die automatische Akquisition BNe aus Daten notwendige multivariate Zeitreihe ist durch das Abtasten der 22 Variablen der simulierten Fertigung alternativ alle 50, 100, 200, 400 oder 800 Zeiteinheiten entstanden und umfasst immer 20.000 Datensätze.

7.4.3 Auswahl und Konfiguration des Verfahrens zur Wissensanwendung

Zur Wissensanwendung bzw. zur PI stellt die Komponente zur Anwendung BNe einzig das in Abschnitt 4.3 beschriebene exakte Verfahren bereit. Solange es keinen zu hohem Speicherbedarf anmeldet oder nicht zu lange Laufzeiten verursacht, soll nicht auf Verfahren der approximativen PI zurückgegriffen werden, wie sie Abschnitt 4.3.4 beschreibt.

Das exakte Verfahren wird auf Entscheidungsunterstützung eingestellt. Stellgröße ist allein die Variable Nummer 14: Prioritätsregel. Regelgrößen sind alternativ die Variablen Nummer 5, 4, 1 und 20 aus Abschnitt 7.4.2: die Terminabweichung, die Durchlaufzeit, der dynamische Wert (Kapitalbindung, Lagerbestand) und die Kapazitätsauslastung. Das Abtasten der Zustände der Fertigung und die Anwendung des BNes zur Auswahl der derzeit „besten“ Prioritätsregel findet alternativ alle 50, 100, 200, 400 oder 800 Zeiteinheiten statt. Für jedes im Rahmen der empirischen Studie durchgeführte Experiment erfolgt 5.000 mal PI zur Auswahl einer Prioritätsregel.

Die Komponente zum Management von Experimenten nimmt sämtliche der hier vorgestellten Einstellungen an den Verfahren bzw. an den Komponenten zur Wissensakquisition und zur Wissensanwendung für jedes Experiment vollständig und automatisch vor.

7.5 Ergebnisse der empirischen Untersuchung hinsichtlich der Güte der Reihenfolgeplanung

7.5.1 Definition der Güte der Reihenfolgeplanung

Die Terminabweichung, die Durchlaufzeit, die Kapitalbindung und die Kapazitätsauslastung bestimmen die Güte der Reihenfolgeplanung. Die Formeln 2.2, 2.3, 2.4 und 2.5 formalisieren diese vier Größen über der Fertigung und definieren somit die Güte der Reihenfolgeplanung zu einem Zeitpunkt bzw. über einen Zeitraum hinweg. Je geringer die Terminabweichung, die Durchlaufzeit und die Kapitalbindung und je höher die Kapazitätsauslastung, desto besser ist der Zustand der Fertigung zu einem Zeitpunkt bzw. während eines Zeitraumes gewesen und desto größeren Erfolg zeitigt die Reihenfolgeplanung. Da die vier Größen z. T. konfliktäre Ziele der PPS im Allgemeinen und der Reihenfolgeplanung im Speziellen sind, werden sie nicht parallel verfolgt, sondern alternativ. Die BNe laufen so nicht Gefahr, einen Kompromiss

zwischen den Zielen zu schließen, der insgesamt ein schlechteres Ergebnis liefert, als wenn nur ein Ziel verfolgt würde.

7.5.2 Exkurs: Einfluss des Organisationstyps auf die Güte der Prognose und Diagnose mittels DBNe

Da das im Rahmen der empirischen Studie zu untersuchende „11-Maschinen-Problem“ nur einen einzigen Organisationsgrad aufweist (siehe Abschnitt 7.3.3) kann der Einfluss des Organisationstyps der Produktion auf die Güte der Reihenfolgeplanung anhand des „11-Maschinen-Problems“ nicht ermittelt werden. Aus diesem Grunde stellt der vorliegende Abschnitt nicht den Einfluss des Organisationstyps auf die Güte der Reihenfolgeplanung mittels BNe dar, sondern den Einfluss des Organisationstyps auf die Prognose- und Diagnosefähigkeit DBNe[8] anhand eines vom „11-Maschinen-Problem“ verschiedenen Problems, was einen vagen Schluss auf den Einfluss des Organisationstyps auf die Güte der Reihenfolgeplanung zulässt. Der qualitative Organisationstyp wird durch den quantitativen Organisationsgrad o ausgedrückt (siehe Formel 6.1).

Zum Zwecke der Untersuchung sind Simulationsmodelle dreier Fertigungssysteme mit jeweils 10 Maschinen und mit Maschinenübergangsmatrizen erstellt worden, welche die Organisationsgrade 0,4, 0,7 respektive 1,0 aufweisen. Im Anschluss ist für jedes Fertigungssystem simulativ eine Zwischenankunftszeit t_a für Fertigungsaufträge ermittelt worden, die eine Auslastung des Fertigungssystems von 0,6 hervorgerufen hat. Für jedes Fertigungssystem sind nun drei Simulationen erfolgt: für $t_a(1 - 0{,}15)$ für t_a und für $t_a(1 + 0{,}15)$. Während der Simulationen sind periodisch Systemzustände abgetastet worden. Folgende Variablen charakterisierten den Systemzustand: die Anzahl l der Fertigungsaufträge im System, die mittlere Durchlaufzeit $\bar{t}_d$, Wartezeit $\bar{t}_w$ und Bearbeitungszeit $\bar{t}_b$ der Fertigungsaufträge, die in der vergangenen Periode das Fertigungssystem verlassen haben, sowie die Warteschlangenlänge $\bar{l}_w$, über den Maschinen gemittelt. Das Abtastintervall ist im voraus aus hochfrequent abgetasteten Zeitreihen visuell bestimmt und bei allen Experimenten beibehalten worden. Aus den Simulationen resultieren neun multivariate Zeitreihen, um Ein- und Ausschwingphase bereinigt, mit jeweils $|X| = 5$ Variablen und mit jeweils 13.000 Datensätzen, von denen später 10.000 zur Akquisition und 3.000 zum Testen des jeweiligen DBNes verwandt worden sind.

Im Anschluss ist die automatische Akquisition des DBNes erfolgt, die mit einer Diskretisierung der reellwertigen Variablen begonnen hat. Die Werte-

[8] Die Untersuchung geht auf [Mun00] zurück.

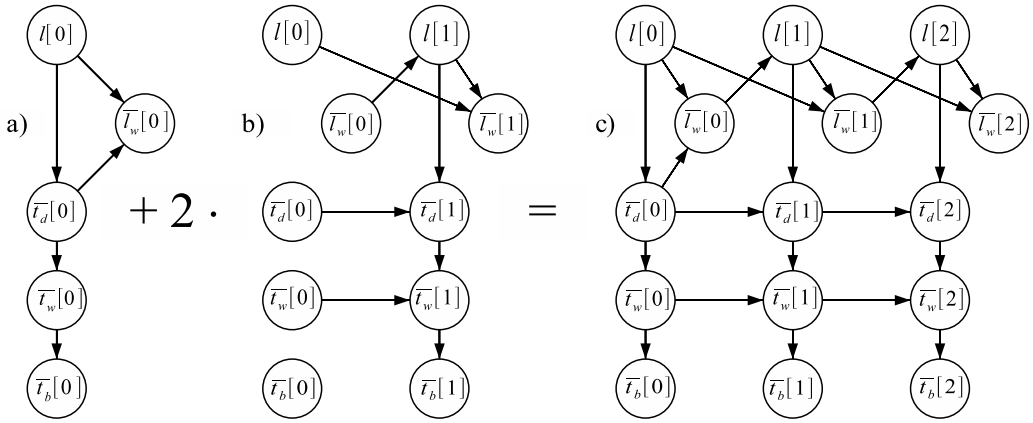

Abbildung 7.3: Graphen a) des Eingangsnetzes B_0 und b) des Übergangsnetzes $B_{\rightarrow}$ sowie c) eines über drei Zeitscheiben entrollten korrespondierenden DBNes B_U

bereiche der Variablen sind jeweils in sieben gleichgroße Intervalle unterteilt worden. Die Graphen B_0 und $B_{\rightarrow}$ des Eingangs- und des Übergangsnetzes sind nur einmal aus der multivariaten Zeitreihe über dem Fertigungssystem mit dem Organisationsgrad $o = 0{,}7$ und der Zwischenankunftszeit t_a erlernt worden. Für alle anderen multivariaten Zeitreihen sind lediglich die Variablen neu diskretisiert worden, da sich neue Wertebereiche ergeben haben, und nach Formel 4.18 sind die bedingten Wahrscheinlichkeitstabellen neu berechnet worden. Abbildung 7.3 zeigt die Graphen von B_0 des Eingangs- und $B_{\rightarrow}$ des Übergangsnetzes sowie den Graphen B_U eines korrespondierenden entrollten DBNes mit beispielsweise drei Zeitscheiben.

Für jede multivariate Zeitreihe ist aus dem jeweiligen Eingangs- und Übergangsnetz ein DBN mit zehn Zeitscheiben entrollt worden, das im Durchschnitt einen Zeitraum des zirka $2,5$-fachen der mittleren Durchlaufzeit eines Auftrages durch die Fertigung überspannt. Jedes dieser DBNe ist mit den 3.000 Datensätzen getestet worden, die mit den 10.000 Datensätzen korrespondieren, aus denen das DBN akquiriert worden ist. Jedes DBN ist zum einen hinsichtlich seiner Prognosefähigkeit (Zustände der Variablen in der 0-ten Zeitscheibe gesetzt) und zum anderen hinsichtlich seiner Diagnosefähigkeit (Zustände der Variablen in der neunten Zeitscheibe gesetzt) untersucht worden. Aus den sich für die nicht gesetzten Variablen ergebenden Randverteilungen sind die Erwartungswerte errechnet, die zugehörigen Testdaten von letzteren subtrahiert, und die Differenz ist mit dem Wertebereich der jeweiligen Variable auf das Standardintervall $[0; 1]$ normiert worden. Die mittlere, die mittlere absolute, die mittlere quadratische Abweichung und die Stan-

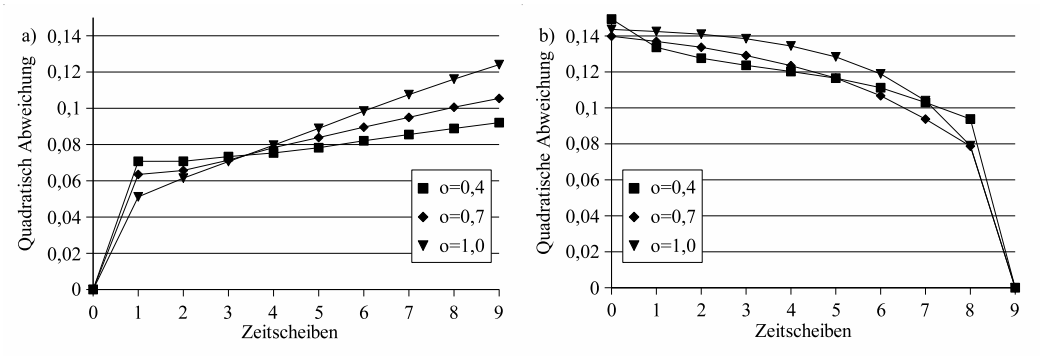

Abbildung 7.4: Auswirkung des Organisationsgrades *o* auf die Güte a) der Prognose und b) die Diagnose der mittleren Auftragsdurchlaufzeit

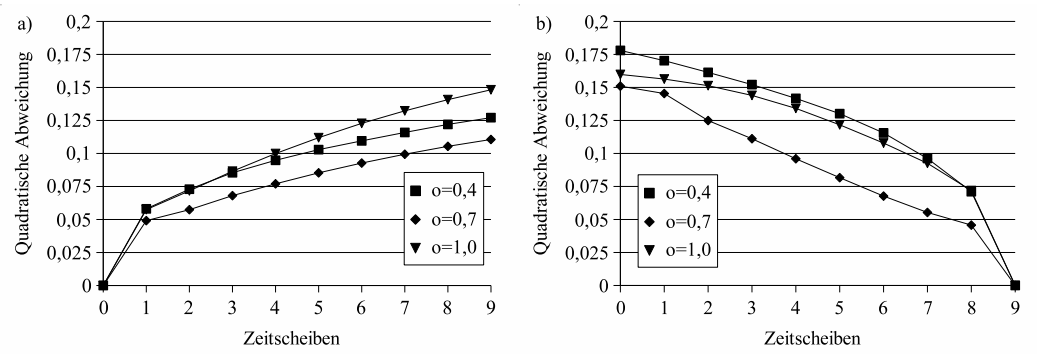

Abbildung 7.5: Auswirkung des Organisationsgrades *o* auf die Güte a) der Prognose und b) die Diagnose der mittleren Warteschlangenlänge

dardabweichung von der mittleren Abweichung sind ermittelt worden. Die mittleren Abweichungen aller Variablen haben bei allen DBNen um 0 geschwankt, so dass ein systematischer Fehler in den DBNen ausgeschlossen werden kann. Sämtliche Standardabweichungen sind gering gewesen und im Zeitverlauf konstant geblieben.

Die Diagramme in den Abbildungen 7.4 und 7.5 stellen dar, wie sich der Organisationsgrad auf a) die Prognose und b) die Diagnose der mittleren Auftragsdurchlaufzeit und der mittleren Warteschlangenlänge auswirkt. Die Fertigung ist jeweils zu durchschnittlich 60 % ausgelastet gewesen. Gemessen worden ist jeweils die quadratische Abweichung der mittels des DBNes prognostizierten bzw. diagnostizierten Wertes vom real aufgetretenen Wert, wobei die Werte vorab mit den Wertebereichen der Variablen normiert worden sind.

Folgende Schlüsse können aus den vier Diagrammen gezogen werden: Die

Prognose- und die Diagnosegüte nehmen im Zeitverlauf degressiv ab. Die Prognosegüte ist etwas besser als die Diagnosegüte, was auf die Eingangsnetze der entrollten DBNe zurückgeführt wird. Die Prognose und Diagnose der Durchlaufzeit gelingt dem DBN etwas besser als die Prognose und Diagnose der Warteschlangenlänge. Die Prognose- und Diagnosegüte der DBNe ist insgesamt eher gering.

Unabhängig von der tatsächlichen Prognose- und Diagnosegüte der DBNe ist aber zu erkennen, dass der Organisationsgrad zwar einen Einfluss auf die Prognose- und Diagnosegüte ausübt, dieser Einfluss aber eher zufällig erscheint: Die Diagramme lassen in ihrer Gesamtheit weder den Schluss zu, dass ein hoher Organisationsgrad eine bessere Prognose- und Diagnosegüte erlaubt, was zu vermuten gewesen wäre, noch, dass ein niedriger Organisationsgrad eine hohe Prognose- und Diagnosegüte zulässt. Es ist auch nicht zu erkennen, dass bei niedrigem Organisationsgrad die Prognose- und Diagnosegüte global besser oder schlechter ausfällt. Zudem unterscheiden sich die Prognose- und Diagnosegüten von Organisationsgrad zu Organisationsgrad nur leicht. Dass der Organisationsgrad keinen entscheidenden Einfluss auf die Prognose- und Diagnosegüte DBNe ausübt, ist ein Indiz dafür, dass er auch die Entscheidungsunterstützung mittels DBNe nicht entscheidend beeinflusst, was auch insofern zu vermuten wäre, als dass das Verfahren zur PI bei Prognose, Diagnose und Entscheidungsunterstützung dasselbe ist. Ob die Reihenfolgeplanung mittels BNe allerdings bei unterschiedlichen Organisationsgraden tatsächlich ähnliche Ergebnisse liefert, bleibt zu untersuchen.

Der vorliegende Exkurs hat den Einfluss des Organisationstyps auf die Güte der Prognose und der Diagnose der Produktion mittels DBNe empirisch untersucht und dargelegt. Das zugrundeliegende Problem ist ein vom „11-Maschinen-Problem“ verschiedenes mit nur zehn Maschinen, aber dafür mit drei voneinander verschiedenen Organisationsgraden gewesen. Das „11-Maschinen-Problem“ weist nur einen einzigen Organisationsgrad auf und ist somit nicht für die Untersuchung geeignet gewesen. In den folgenden Abschnitten steht nun wieder das „11-Maschinen-Problem“ im Fokus.

7.5.3 Einfluss der Auslastung des Produktionssystems auf die Güte der Reihenfolgeplanung

Die Auslastung der zu untersuchenden Fertigung aus Abschnitt 7.3.1 ist für die Experimente im Rahmen der empirischen Studie variiert worden (siehe Abschnitt 7.3.4). Tabelle 7.6 stellt dar, welche mittleren Zwischenankunftszeiten alternativ gewählt worden sind und welche durchschnittliche Engpassauslastung sie hervorgerufen haben. Zur Berechnung der durchschnittli-

chen Engpassauslastung ist Formel 7.16 nach $u_{b,m_{max}}$ umgestellt und verwendet worden.

Um die Güte der Reihenfolgeplanung mittels DBNe zu quantifizieren, ist das jeweils verfolgte technizitäre Ersatzziel während des Zeitraumes periodisch gemessen worden, während dem auch die Daten für die automatische Akquisition des DBNes nach gleicher Periode von der Fertigung abgegriffen worden sind. Aus den für das technizitäre Ersatzziel gemessenen Werten ist der „absolute Mittelwert“ $\bar{m}_{\text{abs}}$ errechnet worden:

$$\bar{m}_{\text{abs}} = \frac{\sum_{i=0}^{n-1} |w_i|}{n}, \tag{7.20}$$

wobei n die Anzahl der Werte und w_i den i-ten Messwert bezeichne.[9]

Während des Zeitraumes zwischen dem Erstellen des DBNes und dem Ende des Experimentes ist das technizitäre Ersatzziel ebenfalls periodisch gemessen worden – jeweils zeitgleich mit der Anwendung des DBNes auf die Fertigung. Über den Messwerten ist ebenfalls der absolute Mittelwert errechnet worden. Die Abweichung des zweiten absoluten Mittelwertes $\bar{m}_{\text{abs,2}}$ vom ersten $\bar{m}_{\text{abs,1}}$ in Prozent,

$$\frac{\bar{m}_{\text{abs,1}} - \bar{m}_{\text{abs,2}}}{\bar{m}_{\text{abs,1}}} 100\%,$$

entspricht der Verbesserung bzw. Verschlechterung des Zielfunktionswertes durch den Einsatz des DBNes zur Reihenfolgeplanung.

Die Zeitreihe in Abbildung 7.6 zeigt die typische Entwicklung eines technizitären Ersatzzieles über der Zeit. Bis zu dem breiten, senkrechten Strich werden die Werte zum Errechnen des ersten absoluten Mittelwertes und nach ihm die Werte zum Errechnen des zweiten gemessen. Man kann auch optisch erkennen, dass sich während des Einsatzes des DBNes (ab dem breiten, senkrechten Strich) die Termintreue verbessert, obwohl sie weiterhin negativ bleibt, und dass der Einsatz des DBNes die starken Schwankungen in der Termintreue ausgleicht.

Das Diagramm in Abbildung 7.7 zeigt den Einfluss der Auslastung der Fertigung bzw. der extern vorgegebenen mittleren Zwischenankunftszeit auf die Termintreue. Es ist zu erkennen, dass die Reihenfolgeplanung mittels DBNe eine umso bessere Termintreue erzielt, je höher die Auslastung der

[9] Die absoluten Beträge sind gebildet worden, weil die Termintreue bzw. die Terminabweichung auch negativ werden kann und ein verfälschender Ausgleich von Verspätungen durch „Verfrühungen“ bei der Summation vermieden werden sollte. Die Werte der anderen technizitären Ersatzziele sind allesamt größer oder gleich null und werden durch das Bilden des absoluten Betrages nicht verändert.

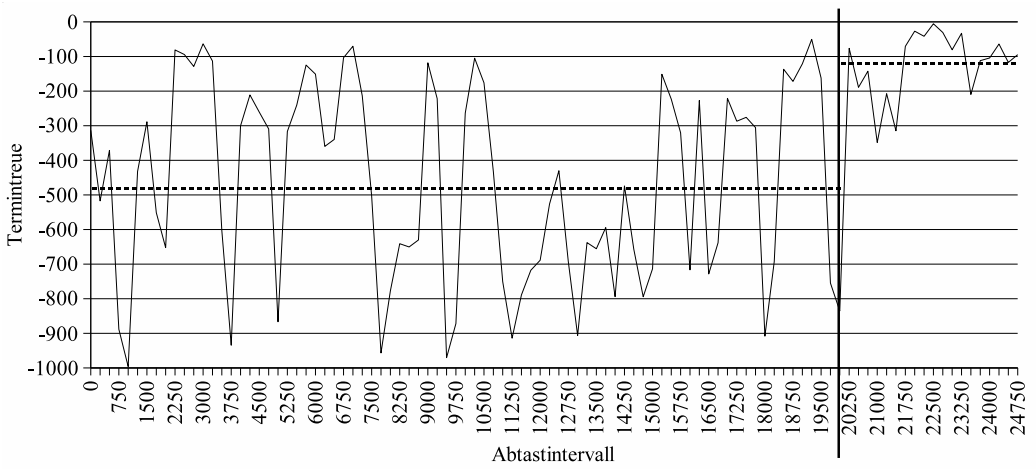

Abbildung 7.6: Entwicklung der Termintreue über der Zeit (mittlere Enpassauslastung 80 %). Die gestrichelten Linien geben die mittlere Termintreue vor und nach der Anwendung des DBNes an.

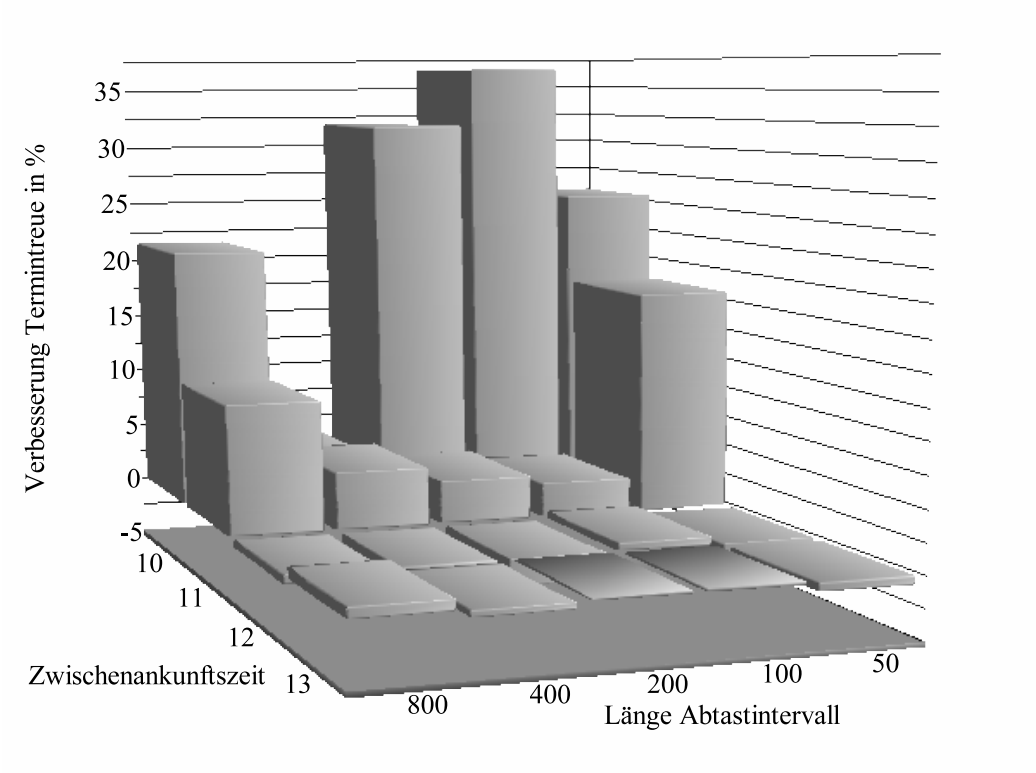

Abbildung 7.7: Einfluss der mittleren Zwischenankunftszeit und der Länge des Abtastintervalles auf die Verbesserung der Termintreue

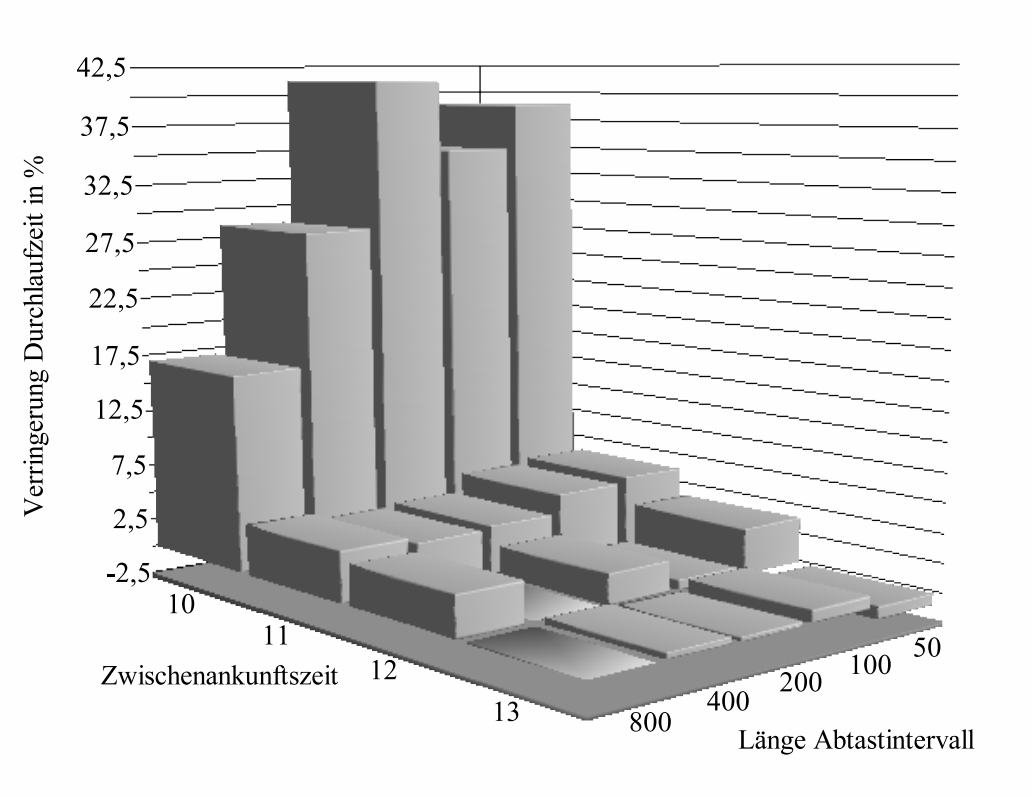

Abbildung 7.8: Einfluss der mittleren Zwischenankunftszeit und der Länge des Abtastintervalles auf die Verringerung der Durchlaufzeit

Fertigung bzw. je geringer die extern vorgegebene Zwischenankunftszeit ist. Liegt die Auslastung der Fertigung unter 70 %, ist kein Einfluss der Reihenfolgeplanung mittels DBNe auf die Termintreue mehr nachzuweisen. Die nur geringfügige Verbesserung der Termintreue um zirka 3,5 % bei einer Länge des Abtastintervalles von 400 und einer mittleren Zwischenankunftszeit von 10 Zeiteinheiten kann durch eine falsch orientierte Kante im DBN erklärt werden, das der Reihenfolgeplanung zugrundeliegt.

Die Wertebereiche der reellwertigen Variablen der 16 DBNe, die während der 16 mit den Säulen im Diagramm aus Abbildung 7.7 korrespondierenden Experimente erstellt und angewandt worden sind, sind sämtlich in vier gleichgroße Intervalle diskretisiert worden.

Das Diagramm in Abbildung 7.8 spiegelt die Wirkung der Auslastung der Fertigung bzw. der extern vorgegebenen mittleren Zwischenankunftszeit auf die mittlere Durchlaufzeit wider. Auch dieses Diagramm lässt erkennen, dass die Reihenfolgeplanung mittels DBNe umso erfolgreicher ist, je höher die Auslastung der Fertigung bzw. je geringer die extern vorgegebene Zwi-

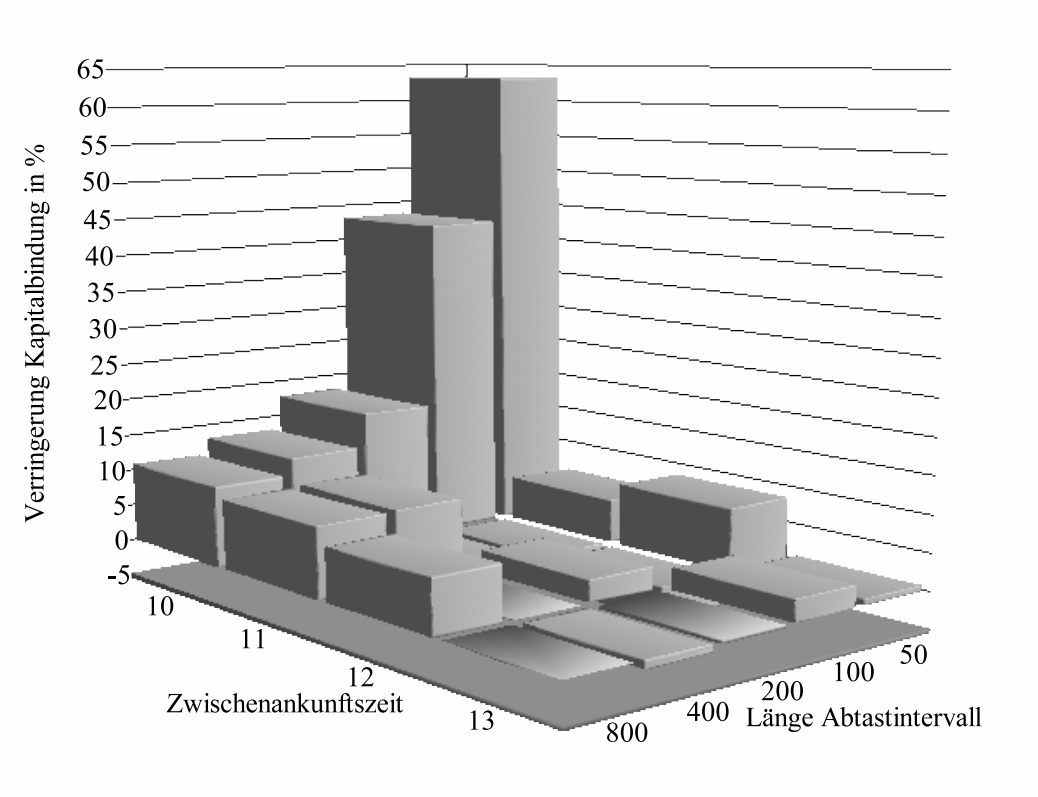

Abbildung 7.9: Einfluss der mittleren Zwischenankunftszeit und der Länge des Abtastintervalles auf die Verringerung der Kapitalbindung

schenankunftszeit ist. Die Verringerung der mittleren Durchlaufzeit steigt mit zunehmender Auslastung bzw. abnehmender Zwischenankunftszeit progressiv. Sinkt die Auslastung allerdings unter 65 %, kann kein Einfluss der Reihenfolgeplanung mittels DBNe auf die mittlere Durchlaufzeit mehr festgestellt werden.

Die Wertebereiche der reellwertigen Variablen der DBNe, die an den 16 Experimenten für das Diagramm aus Abbildung 7.8 beteiligt gewesen sind, sind jeweils in drei gleichgroße Intervalle diskretisiert worden.

Das Diagramm in Abbildung 7.9 veranschaulicht den Effekt, den die Auslastung der Fertigung bzw. die extern vorgegebenen mittleren Zwischenankunftszeit auf die mittlere Kapitalbindung ausübt. Mit steigender Auslastung der Fertigung bzw. mit sinkender Zwischenankunftszeit, nimmt auch hier der positive Einfluss der Reihenfolgeplanung mittels DBNe auf das Ziel der Reihenfolgeplanung zu: Die Verringerung der Kapitalbindung nimmt mit zunehmender Auslastung überproportional stark zu. Das heißt, dass die Kapitalbindung mit steigender Auslastung überproportional stark sinkt. Ein durch-

gängig starker und positiver Einfluss der Auslastung auf die Kapitalbindung ist allerdings erst ab zirka 75 % Auslastung der Fertigung zu erkennen.[10]

Die Wertebereiche der reellwertigen Variablen der an den Experimenten für das Diagramm aus Abbildung 7.9 beteiligten DBNe sind in jeweils drei gleichgroße Intervalle diskretisiert worden.

Die Diagramme in den Abbildungen 7.7, 7.8 und 7.9 zeigen, dass die Reihenfolgeplanung mittels DBNe unterhalb einer Engpassauslastung von 75 bis 65 % keine erkennbare Wirkung auf das Erreichen des jeweils verfolgten technizitären Ersatzzieles mehr ausübt. Das ist bei einem statischen Engpass (Maschine 11, siehe Formel 7.12) und seiner relativ geringen Auslastung auch nicht weiter verwunderlich: Welche Reihenfolgeentscheidung soll das DBN auch treffen, wenn meist kein und oft nur ein Auftrag vor der Enpassmaschine wartet? Da die anderen Maschinen noch geringer ausgelastet sind als der Engpass, baut sich vor ihnen natürlich noch seltener eine Warteschlange mit mehr als einem Fertigungsauftrag auf. Eine solche Warteschlange ist aber vonnöten, damit das DBN überhaupt erst die Chance hat, eine Reihenfolgeentscheidung zu treffen und den Erreichungsgrad des jeweiligen technizitären Ersatzzieles zu beeinflussen. Der Umstand, dass die Reihenfolgeplanung mittels DBNe unterhalb einer gewissen Auslastung des statischen Engpasses keinen Einfluss mehr auf den Wert des jeweils verfolgten technizitären Ersatzzieles ausübt, kann also erklärt werden. Der Umstand ist allerdings nicht auf ein Manko DBNe oder der (verwendeten) Prioritätsregeln zurückzuführen, sondern ergibt sich aus den Charakteristika der zugrundeliegenden Fertigung. Eine erfolgreiche Reihenfolgeplanung ist bei der vorliegenden Fertigung unterhalb einer gewissen Engpassauslastung nicht nur unmöglich, sondern auch objektiv nicht sinnvoll bzw. unnötig.

Ob die extern vorgegebene Auslastung der Fertigung bzw. die mittlere Zwischenankunftszeit das letzte technizitäre Ersatzziel, die Kapazitätsauslastung beeinflusst, stellt das Diagramm in Abbildung 7.10 dar. Es sei darauf hingewiesen, dass sich die Erhöhung der Kapazitätsauslastung zwischen $-0{,}4$ % und $+0{,}4$ % bewegt. Es kann ergo kein Einfluss der extern vorgegebenen Auslastung bzw. der mittleren Zwischenankunftszeit auf die Erhöhung der Kapazitätsauslastung durch Reihenfolgeplanung mittels DBNe nachgewiesen werden. Das ist insofern nicht verwunderlich, als dass die mittlere Zwischenankunftszeit und der Produktmix extern vorgegeben worden sind und das DBN sie nicht durch die Wahl einer Prioritätsregel beeinflussen

[10] Dass die Verringerung der Kapitalbindung bei konstantem Abtastintervall und variabler, aber hoher Zwischenankunftszeit stark variiert, ist lediglich Zufallseinflüssen auf die Fertigung zuzuschreiben, da das DBN bei hoher Zwischenankunftszeit und somit geringer Kapazitätsauslastung noch keine Reihenfolgeentscheidung treffen kann, wie nachfolgend noch erläutert wird.

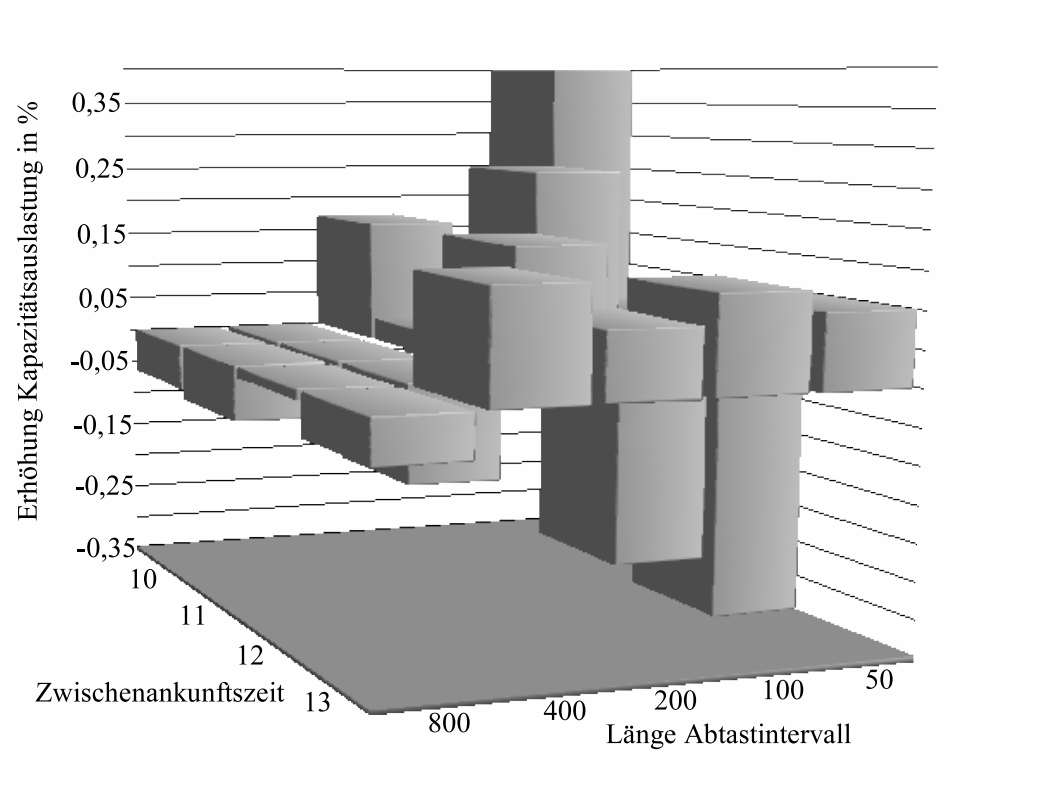

Abbildung 7.10: Einfluss der mittleren Zwischenankunftszeit und der Länge des Abtastintervalles auf die Erhöhung der Kapazitätsauslastung

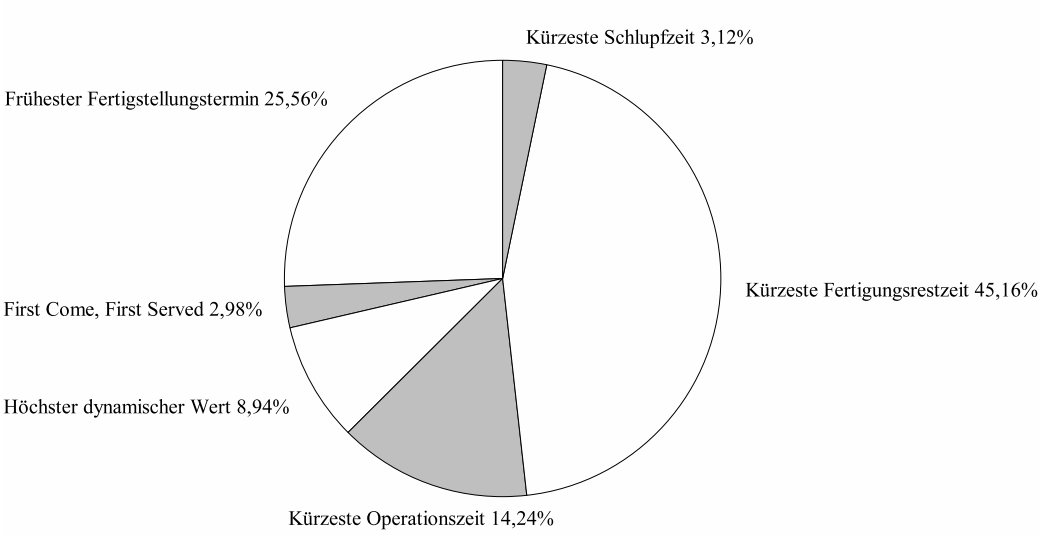

Abbildung 7.11: Relative Häufigkeiten der Anwendungen von Prioritätsregeln bei einem Experiment (Ziel: bessere Termintreue)

kann. Selbst wenn das Ändern der Bearbeitungsreihenfolge der Aufträge auf den Maschinen eine kurzfristige Erhöhung der Kapazitätsauslastung nach sich zöge, flössen weitere Aufträge nicht schneller nach, und die Kapazitätsauslastung bliebe in einem Intervall im Mittel trotzdem konstant. Wenn die Reihenfolgeplanung mittels DBNe dazu führt, dass die Kapazitätsauslastung der Fertigung steigt und die mittlere Auftragsdurchlaufzeit abnimmt, so sollten mehr Aufträge freigegeben werden.

Der Vollständigkeit halber sei noch angemerkt, dass die Wertebereiche der reellwertigen Variablen der an den 16 Experimenten für das Diagramm aus Abbildung 7.10 beteiligten DBNe in jeweils vier gleichgroße Intervalle diskretisiert worden sind.

Bisher sind noch keine Aussagen dazu getroffen worden, wie häufig die DBNe die Prioritätsregeln bei der Reihenfolgeplanung wechseln. Es ist denkbar, dass die DBNe beim Verfolgen genau eines technizitären Ersatzzieles eine Prioritätsregel als grundsätzlich beste erkennen und diese Regel ausschließlich anwenden, was den operativen Einsatz der DBNe überflüssig machte. Der pathologische Fall, dass ein DBN ausschließlich eine Prioritätsregel anwendet, tritt allerdings bei keinem Experiment auf.

Das Diagramm in Abbildung 7.11 stellt anhand eines Beispiels dar, wie häufig jede Prioritätsregel während der Testphase eines Experimentes angewendet wird. Bei dem Experiment handelt es sich um dasjenige, welches bereits für das Diagramm in Abbildung 7.6 die Daten geliefert hat. Das DBN

hat bei diesem Experiment die Termintreue als technizitäres Ersatzziel verfolgt und mit der Verringerung der mittleren Verspätung um knapp 40 % ein relativ gutes Resultat erzielt. Von 5.000 Möglichkeiten, die Prioritätsregel zu wechseln, hat das DBN 2.325 ergriffen und in 2.675 Fällen die während des letzten Intervalles geltende Prioritätsregel beibehalten.

7.5.4 Einfluss der Methoden der Wissensverarbeitung im Kontext BNe auf die Güte der Reihenfolgeplanung

Aufgrund der guten Ergebnisse aus Abschnitt 7.5.3 und somit in Übereinstimmung mit den Zielen der empirischen Untersuchung aus Abschnitt 7.1 sind die Methoden der Wissensverarbeitung im Kontext BNe während der empirischen Untersuchung des Potenzials BNe zur Unterstützung der Reihenfolgeplanung kaum variiert worden. Zudem hat das extern für die empirische Untersuchung vorgegebene „11-Maschinen-Problem“ oft nur jeweils eine Methode der Wissensverarbeitung im Kontext BNe von mehreren verfügbaren impliziert, die dann einzig angewendet worden ist. So sind z. B. nur DBNe als Wissensrepräsentationsform für die Regelung eines komplexen stochastischen Prozesses in Frage gekommen und demzufolge ausschließlich angewandt worden. Darüber hinaus unterscheiden sich alternative Methoden der Wissensverarbeitung im Kontext BNe hinsichtlich ihrer Leistungsfähigkeit oft nur so unwesentlich voneinander, dass es genügt, eine anzuwenden. So sind z. B. die verfügbaren Gütemaße äquivalent.

Zwei Parameter sind im weiteren Rahmen der Wissensakquisition und -anwendung im Kontext BNe für die Experimente dennoch variiert worden:

1. die Länge der Periode („Abtastintervall“), nach der einerseits die multivariaten Stichproben bzw. Zeitreihen abgetastet worden sind, die der automatischen Akquisition DBNe als Eingabe gedient haben, und nach der andererseits jeweils der Zustand der Fertigung gemessen und das DBN zur Reihenfolgeplanung angewendet worden ist, sowie

2. die Anzahl der gleichgroßen, sich gegenseitig ausschließenden und den Wertebereich der zugrundeliegenden reellwertigen oder ganzzahligen Variable im Spektrum der vorkommenden Werte vollständig abdeckenden Intervalle, die dem Diskretisieren der Variable entspringen und als Zustände für die Variable im BN verwendet werden.

Letztere Anzahl wird im hiesigen Abschnitt kurz als „Anzahl der Intervalle“ oder „Anzahl der Zustände“ bezeichnet werden. Sie wird bei der automati-

schen Akquisition eines BNes vorgegeben und ist für alle reellwertigen und ganzzahligen Variablen gleich groß.

Wie sich die Länge des Abtastintervalles auf die Güte der Reihenfolgeplanung mittels DBNe auswirkt, stellen bereits die Diagramme in den Abbildungen 7.7 bis 7.10 dar: Experimentübergreifend lässt sich schlussfolgern, dass die Güte der Reihenfolgeplanung mittels DBNe zunimmt, je kürzer das Abtast- und Anwendungsintervall gewählt wird, und abnimmt, je länger sich das Abtastintervall ausdehnt. Besonders deutlich zeigt sich der Zusammenhang im Diagramm in Abbildung 7.9. Es sei aber darauf hingewiesen, dass das Abtastintervall nicht linear, sondern exponentiell mit $50 \cdot 2^{(x-1)}$ zunimmt, so dass die Güte der Reihenfolgeplanung mittels DBNe mit zunehmender Abtastintervallänge wesentlich langsamer abnimmt, als es auf den ersten Blick den Anschein erweckt. Die Länge des Abtastintervalles übt also bei weitem keinen so starken Einfluss auf die Güte der Reihenfolgeplanung mittels BNe aus, wie zunächst vermutet, und die zwei präferierten Vorgehensweisen zur Bestimmung der Länge des Abtastintervalles aus Abschnitt 4.4.2 genügen vollauf, um eine akzeptable Abtastintervallänge zu bestimmen.

Wie die Anzahl der Intervalle bzw. der diskreten Zustände der reellwertigen und ganzzahligen Variablen die Ziele der Reihenfolgeplanung mittels DBNe beeinflusst, zeigen die drei Diagramme in den Abbildungen 7.12, 7.13 und 7.14. Sie spiegeln den Einfluss der Anzahl der Zustände auf Termintreue, Durchlaufzeit respektive Kapitalbindung wider, welche die Reihenfolgeplanung mittels DBNe erzielt. Bei nur zwei getesteten Zustandsanzahlen kann leider keine Tendenz festgestellt werden, ob sich eine höhere Anzahl von Intervallen/Zuständen positiv oder negativ auf die Ergebnisse der Reihenfolgeplanung mittels DBNe auswirkt. Die Anzahl der Intervalle zu erhöhen, hat nicht nur Vorteile: siehe Abschnitt 4.4.3.

Nachdem die Ergebnisse der empirischen Untersuchung hinsichtlich der Güte der Reihenfolgeplanung mittels DBNe umfassend beurteilt worden ist, sollen nun die Ergebnisse der empirischen Untersuchung hinsichtlich des Rechenaufwandes für die Reihenfolgeplanung mittels DBNe betrachtet werden.

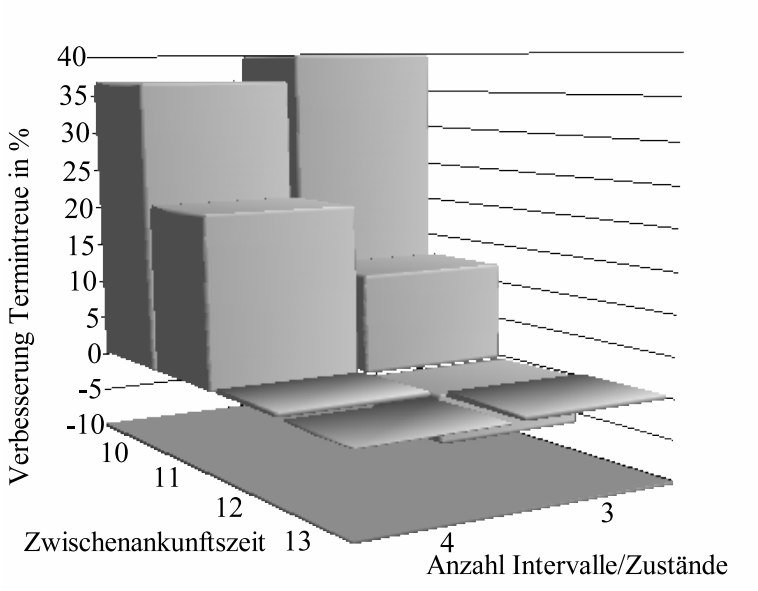

Abbildung 7.12: Einfluss der Anzahl der Intervalle/Zustände der diskretisierten Variablen und der mittleren Zwischenankunftszeit auf die Verbesserung der Termintreue

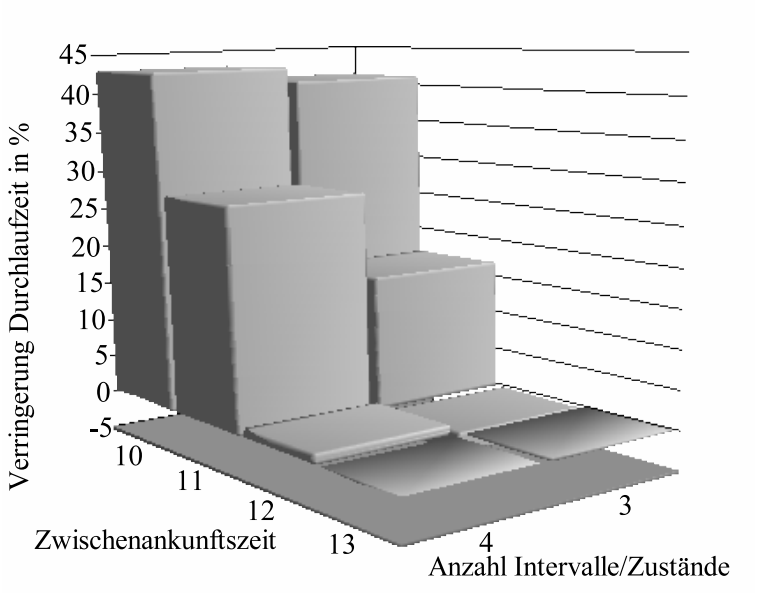

Abbildung 7.13: Einfluss der Anzahl der Intervalle/Zustände der diskretisierten Variablen und der mittleren Zwischenankunftszeit auf die Verringerung der Durchlaufzeit

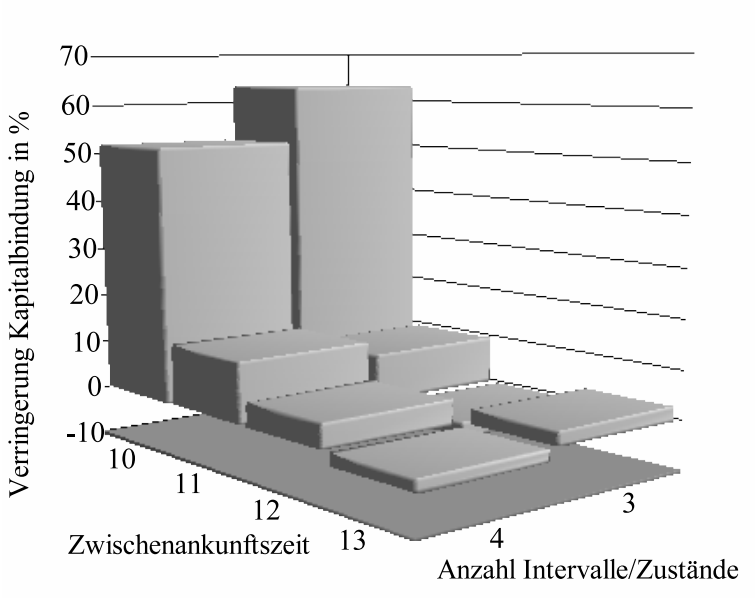

Abbildung 7.14: Einfluss der Anzahl der Intervalle/Zustände der diskretisierten Variablen und der mittleren Zwischenankunftszeit auf die Verringerung der Kapitalbindung

7.6 Ergebnisse der empirischen Untersuchung hinsichtlich des Rechenaufwandes für die Reihenfolgeplanung

7.6.1 Definition des Rechenaufwandes für die Reihenfolgeplanung

Rechenaufwand im Allgemeinen und für die Reihenfolgeplanung mittels DBNe im Speziellen gliedert sich in den Speicherbedarf der verwendeten Datenstrukturen und die Laufzeit der angewandten Algorithmen. Speicherbedarf und Laufzeit lassen sich nicht gemeinsam, sondern nur getrennt voneinander beurteilen, obwohl sich beide in gewissen Grenzen gegenseitig substituieren können. Alle folgenden Betrachtungen spalten sich ergo in diese zwei Bestandteile auf, den Speicherbedarf und die Laufzeit.

Der Rechenaufwand für die Reihenfolgeplanung mittels DBNe setzt sich aus einem fixen und einen variablen Anteil zusammen. Der fixe Anteil besteht zum einen aus der automatischen Akquisition des DBNes aus der multivariaten Stichprobe. Zum anderen besteht er aus dem Erstellen der Inferenzmaschine über dem DBN. Der Aufwand zum „Entrollen“ des DBNes ist

vernachlässigbar gering und wird demzufolge nicht als Bestandteil des fixen Anteils am Rechenaufwand berücksichtigt.

Der variable Anteil des Rechenaufwandes für die Reihenfolgeplanung mittels DBNe besteht einzig aus der PI i. e. S., wobei der Speicherbedarf für die Inferenzmaschine zum fixen Anteil des Rechenaufwandes für die Reihenfolgeplanung mittels DBNe zählt.

Zum Messen des Speicherbedarfes stellt Delphi bzw. Object-Pascal eine Funktion zur Verfügung, welche die Gesamtgröße der aktuell zugewiesenen Speicherblöcke, die von der Anwendung verwendet werden, in Byte zurückgibt. Diese Funktion stellt fest, wieviel Byte des Arbeitsspeichers die Anwendung aktuell verwendet. Die Funktion arbeitet über DLL- und Thread-Grenzen hinweg korrekt und bezieht natürlich auch ausgelagerten Speicher in die Messung ein.

Der Speicherbedarf für die Inferenzmaschine wird ermittelt, indem der Speicherbedarf direkt vor und direkt nach dem Aufbau der Inferenzmaschine gemessen wird. Die Differenz zwischen dem Speicherbedarf nach dem Aufbau der Inferenzmaschine und dem Speicherbedarf vor dem Aufbau der Inferenzmaschine ergibt den Speicherbedarf für die Inferenzmaschine. Der Speicherbedarf für die zum Aufbau der Inferenzmaschine notwendigen Hilfsgraphen ist im Verhältnis zum Aufbau der Inferenzmaschine so gering, dass er vernachlässigt wird.

Den Speicherbedarf für die automatische Akquisition des DBNes aus einer multivariaten Zeitreihe zu ermitteln, gestaltet sich schwieriger, da der Speicherbedarf während der Akquisition unter Umständen stark schwankt also auch temporär abnehmen kann. So wird der Speicherbedarf einmalig vor dem Beginn der automatischen Akquisition gemessen. Nach der ersten Iteration des Akquisitionsalgorithmus' wird ebenfalls der Speicherbedarf gemessen und als maximaler Speicherbedarf deklariert. Nach jeder weiteren Iteration wird der aktuelle zum maximalen Speicherbedarf, sofern der aktuelle Speicherbedarf höher liegt als der maximale. Nach dem Ende der automatischen Akquisition ergibt die Differenz aus dem maximalen Speicherbedarf und dem vor der Akquisition gemessenen Speicherbedarf den Speicherbedarf für die automatische Akquisition DBNe aus einer multivariaten Zeitreihe.

Misst man die Laufzeit eines Algorithmus in Realzeit, so wird sie unweigerlich durch andere Prozesse verfälscht. Der Virenscanner, Systemdienste, der Bildschimschoner und das Netzwerk laufen im Hintergrund kontinuierlich, verursachen sporadisch Rechenaufwand und bremsen so den Algorithmus temporär aus, dessen Laufzeit gemessen wird, was dazu führt, dass die Messungen nicht realistisch und nicht sinnvoll zu vergleichen sind. Aus diesem Grunde ist eine Funktion des Betriebssystems verwendet worden, welche die Summe aller Laufzeiten zurückgibt, welche sämtliche Threads eines Pro-

zesses zwischen dem Start des Prozesses und dem aktuellen Zeitpunkt im „Kernel Mode“ und im „User Mode“ verbracht haben. Die Funktion ermittelt also die reine Gesamtlaufzeit eines Prozesses seit seinem Start.

Um die Laufzeit eines Algorithmus zu ermitteln, ist die reine Gesamtlaufzeit des Prozesses direkt vor dem Beginn und direkt nach dem Ende Algorithmus gemessen worden, der natürlich in dem Prozess bzw. in seinen Threads ablaufen muss. Die Differenz zwischen der reinen Gesamtlaufzeit des Prozesses nach dem Ende des Algorithmus und der reinen Gesamtlaufzeit des Prozesses vor dem Beginn des Algorithmus ergibt die reine Laufzeit des Algorithmus. Auf die eben beschriebene Art und Weise sind die Laufzeiten für die automatische Akquisition der DBNe aus multivariaten Zeitreihen, für den Aufbau der Inferenzmaschinen über den DBNen und für die PIen i. e. S. ermittelt worden.

Natürlich verursachen die Funktionen zum Messen und Ermitteln des Speicherbedarfes und der Laufzeit ebenfalls Rechenaufwand, der allerdings ebenfalls so gering ist, dass er vernachlässigt worden ist.

Sämtlicher Rechenaufwand ist auf Rechnern mit Intel Pentium III Prozessoren, einer Taktfrequenz von 1,7 GHz und einem Arbeitsspeicher von einem GB unter dem Betriebssystem Microsoft Windows XP Professional ermittelt worden.

Die Betrachtung der Ergebnisse der empirischen Untersuchung hinsichtlich des Rechenaufwandes für die Reihenfolgeplanung separiert lediglich nach der „Anzahl der Intervalle/Zustände“, in welche die Wertebereiche reellwertiger und ganzzahliger Variablen diskretisiert worden sind. Da die Eingabedaten für die automatische Akquisition der DBNe aus multivariaten Zeitreihen immer denselben Umfang aufgewiesen haben, die DBNe immer für dieselbe Anzahl von Variablen und Zeitscheiben erstellt worden sind und der Vernetzungsgrad der DBNe nicht von der Länge des Abtastintervalles, der Zwischenankunftszeit oder dem verfolgten technizitären Ersatzziel abhängt, ist auch kein weiteres Unterteilen der Ergebnisse sinnvoll oder gar vonnöten.

7.6.2 Fixer Anteil am Rechenaufwand für die Wissensverarbeitung zur Reihenfolgeplanung

Der fixe Anteil an der Laufzeit für die Reihenfolgeplanung mittels DBNe setzt sich aus der Laufzeit für die automatische Akquisition des Eingangsnetzes, der Laufzeit für die automatische Akquisition des Übergangsnetzes sowie der Laufzeit für dem Aufbau der Inferenzmaschine zusammen. Das Diagramm in Abbildung 7.15 stellt dar, welche minimalen, mittleren und maximalen Laufzeiten alle drei Bestandteile verursachen. Die gestapelte Darstellung lässt

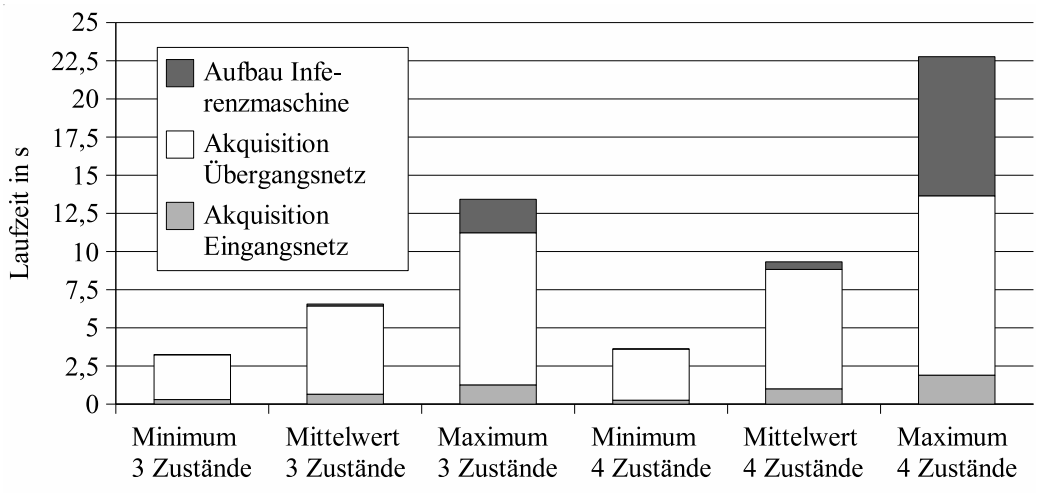

Abbildung 7.15: Gestapelte minimale, mittlere und maximale Laufzeiten für die automatische Akquisition des Eingangsnetzes und des Übergangsnetzes sowie für den Aufbau der Inferenzmaschine für drei und vier Zustände pro reellwertiger bzw. ganzzahliger Variable

auch sehr gut die minimale, mittlere und maximale Gesamtlaufzeit des fixen Anteils an der Laufzeit für die Reihenfolgeplanung mittels DBNe erkennen. Das Diagramm unterteilt in die Experimente bzw. DBNe, deren reellwertige und ganzzahlige Variablen in drei Intervalle, und die Experimente, deren Variablen in vier Intervalle bzw. Zustände diskretisiert worden sind.

Den größten Anteil am fixen Anteil an der Laufzeit zur Reihenfolgeplanung mittels DBNe nimmt die automatische Akquisition des Übergangsnetzes ein. Der Aufbau der Inferenzmaschine trägt mit zunehmender Anzahl von Zuständen pro Variable einen größeren relativen Anteil zur Laufzeit bei. Dass die Gesamtlaufzeit für den fixen Anteil an der Laufzeit selbst bei vier Zuständen pro Variable im Maximum weit unter 25 Sekunden und im Mittel sogar unter zehn Sekunden liegt, ist sehr vielversprechend. Es besteht nun entweder die Möglichkeit, die DBNe und die Inferenzmaschine wesentlich öfter zu erstellen, als zu Beginn angestrebt, oder die Möglichkeit, sowohl die DBNe als auch die Inferenzmaschine mit Hilfe von Optimierungsalgorithmen (siehe Abschnitte 3.1 und 3.2) und nicht nur mittels „gieriger“ Suchalgorithmen aufzubauen, obwohl das längere Laufzeiten verursacht. Da ein DBN und eine Inferenzmaschine für die Reihenfolgeplanung vermutlich höchstens einmal pro Tag (z. B. in einem Nachtlauf) erstellt werden müssen, sollte die zweite Möglichkeit wahrgenommen und durchaus zwei bis drei Stunden Laufzeit in sie investiert werden. Die bisher bestehende Inferenzmaschine kann während der Akquisition eines neuen DBNes und dem Aufbau einer neuen Inferenz-

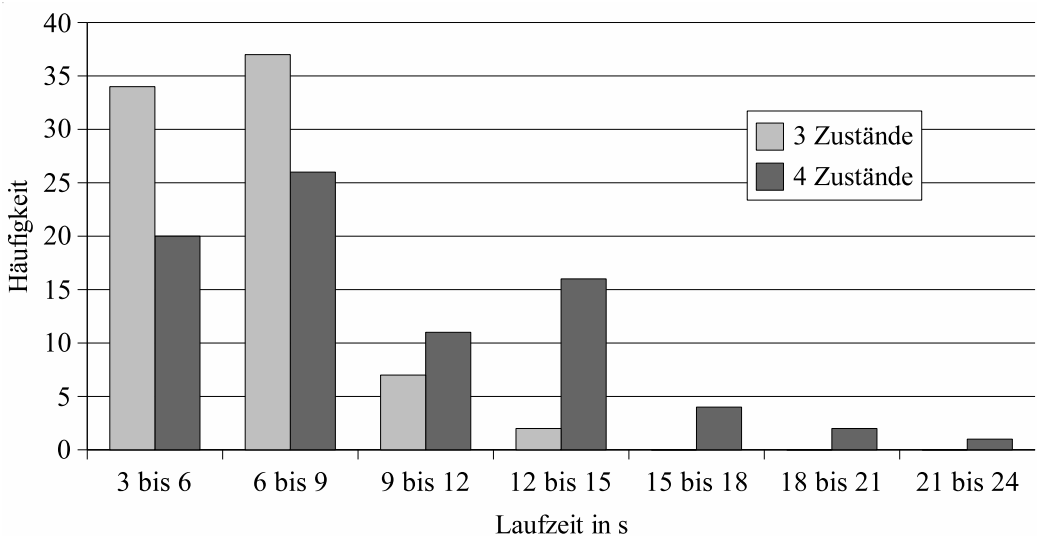

Abbildung 7.16: Histogramm der Gesamtlaufzeiten für die automatische Akquisition des DBNes und für den Aufbau der Inferenzmaschine für drei bzw. vier Zustände pro reellwertiger bzw. ganzzahliger Variable

maschine für dieses DBN ja weiter angewendet werden.

Das Diagramm in Abbildung 7.16 stellt ein Histogramm dar. Die Summen der Laufzeiten, die für die automatische Akquisition der DBNe und für den Aufbau der Inferenzmaschinen gemessen worden sind, sind in Bereiche eingeteilt worden. Die Höhe einer jeden Säule gibt die absolute Häufigkeit des Auftretens einer Laufzeit in ihrem Bereich an. Das Diagramm unterscheidet nach DBNen, deren reellwertige und ganzzahlige Variablen in drei bzw. vier Intervalle bzw. Zustände diskretisiert worden sind.

Es ist relativ deutlich zu erkennen, dass die Laufzeiten der fixen Anteile an der Reihenfolgeplanung mittels DBNe im Mittel eher gering sind und bei DBNen mit vier Zuständen pro reellwertiger/ganzzahliger Variable bis an die 24 Sekunden reichen. Im Prinzip stellt das Diagramm in Abbildung 7.16 die Inhalte des Diagramms in Abbildung 7.15 nur in einer anders aggregierten Form dar. Allerdings kann man Abbildung 7.16 erkennen, dass es sich bei der Verteilung der Laufzeiten um eine links durch null begrenzte, linkssteile bzw. rechtsschiefe und vermutlich unimodale Verteilung handelt. Die Unregelmäßigkeiten bei der Verteilung der Laufzeiten bei Variablen mit vier Zuständen im mittleren Bereich, ist darauf zurückzuführen, dass in den Stichproben, die den Häufigkeiten zugrundeliegen, Eingangs- und/oder Übergangsnetze vertreten gewesen sind, bei denen mindestens ein Knoten sehr viele Eltern aufgewiesen hat oder alle Knoten gleichmäßig wenige Eltern aufgewiesen ha-

ben, so dass die automatische Akquisition der DBNe und der Aufbau korrespondierender Inferenzmaschinen entsprechend lange oder kurze Laufzeiten verursacht haben.[11]

Das Diagramm in Abbildung 7.17 widmet sich nun dem Speicherbedarf, den die Inferenzmaschine verursacht, als fixem Anteil am Rechenaufwand für die Reihenfolgeplanung mittels DBNe. Obwohl die Inferenzmaschine zur Wissensanwendung gehört, die ja eher variablen Charakter trägt, wird sie jedoch immer genau einmal aufgebaut und existiert genau einmal, egal wie oft PI i. e. S. über ihr vollzogen wird, so dass ihr Speicherbedarf zum fixen Anteil am Rechenaufwand für die Reihenfolgeplanung mittels DBNe zählt.

Der Speicherbedarf, der für die Inferenzmaschinen gemessen worden ist, ist in Bereiche eingeteilt worden, von denen der am weitesten links liegende kleiner und der am weitesten rechts liegende größer als die sonst gleichgroßen Bereiche gewählt worden ist, um die Kompaktheit und Aussagekraft des Diagrammes zu erhöhen. Die Höhe einer jeden Säule gibt die absolute Häufigkeit des Auftretens eines Speicherbedarfes in ihrem Bereich an. Das Diagramm unterscheidet nach DBNen bzw. Inferenzmaschinen, deren reellwertige und ganzzahlige Variablen in drei bzw. vier Intervalle bzw. Zustände diskretisiert worden sind.

Das Diagramm lässt eindeutig erkennen, dass der Speicherbedarf der Inferenzmaschinen im Mittel relativ gering ist, egal, ob Variablen mit drei oder mit vier Zuständen verwandt worden sind. Der Mittelwert für den Speicherbedarf bei Inferenzmaschinen mit Variablen mit drei Zuständen beträgt zirka 5,5 MB, bei Inferenzmaschinen mit Variablen mit vier Zuständen zirka 20 MB. Es sollte allerdings nicht außer acht gelassen werden, dass die DBNe nur jeweils über zwei Zeitscheiben entrollt worden sind. Aber selbst wenn das DBN, dessen Inferenzmaschine den maximalen Speicherbedarf von knapp 285 MB anmeldet, über zehn Zeitscheiben entrollt worden wäre, hätte ein handelsüblicher PC den Speicherbedarf der Inferenzmaschine von zirka $\frac{285}{2} \cdot 10 = 1.425$ MB befriedigt, vorausgesetzt, dass der Speicherbedarf der Inferenzmaschine nur linear mit der Anzahl der Zeitscheiben ansteigt, was aufgrund der strukturell gleichen Zeitscheiben der DBNe zu vermuten ist.

Die Anzahl der Zustände pro Variable scheint auch einen großen Einfluss auf den Speicherbedarf der Inferenzmaschinen auszuüben, was aus dem starken Anstieg des Mittelwertes von 5,5 auf 20 MB und des Maximalwertes von 55 auf 285 MB geschlussfolgert wird, der aus dem Wechsel von Variablen mit

[11] Die Laufzeiten für die automatische Akquisition BNe und den Aufbau mit ihnen korrespondierender Inferenzmaschinen hängt nämlich nicht nur von den Anzahlen der Zustände der Variablen, sondern auch von den Graden der Knoten der gerichteten, azyklischen Graphen ab. Bei vielen Zuständen pro Variable wirken sich hohe Knotengrade sehr viel stärker auf den Rechenaufwand aus als bei wenigen Zuständen pro Variable.

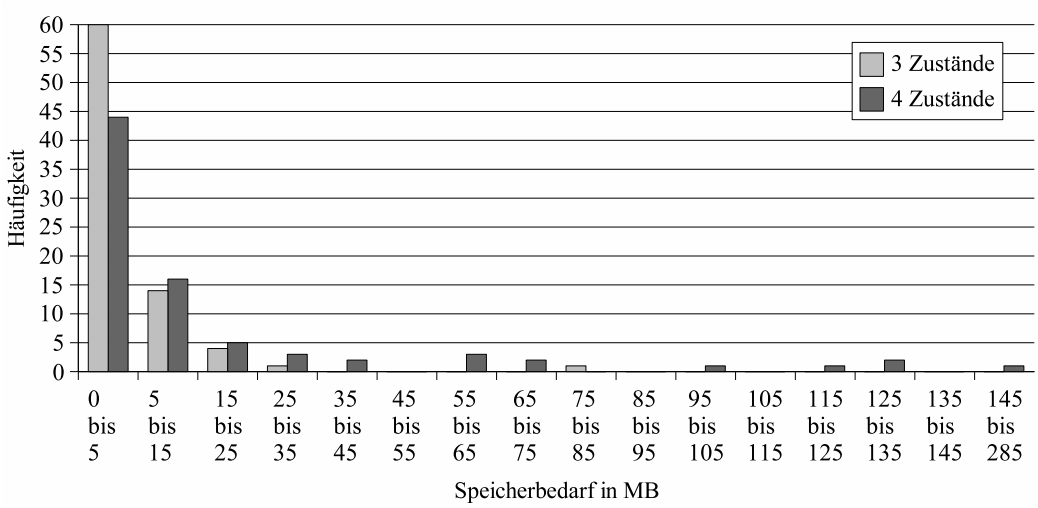

Abbildung 7.17: Histogramm des Speicherbedarfs der Inferenzmaschine für drei bzw. vier Zustände pro reellwertiger bzw. ganzzahliger Variable im DBN bzw. in der Inferenzmaschine

drei Zuständen zu Variablen mit vier Zuständen resultiert. Der Speicherbedarf für Inferenzmaschinen steigt ja im schlimmsten Falle exponentiell mit der Anzahl der Zustände pro Variable. Um ihn gering zu halten, sollten die Wertebereiche reellwertiger und ganzzahliger Variablen bewusst in möglichst wenige Intervalle untergliedert und auch rein ordinale und sogar nominale Werte gruppiert werden, sofern das inhaltlich möglich ist.

7.6.3 Variabler Anteil am Rechenaufwand für die Wissensverarbeitung zur Reihenfolgeplanung

Der variable Anteil am Rechenaufwand, den die Reihenfolgeplanung mittels DBNe verursacht, besteht einzig aus der Laufzeit für die PI i. e. S. Sie ließe sich noch unterteilen in das Absorbieren der Evidenz, das Austauschen der Nachrichten und das Herausintegrieren von Summen- oder Maximumrändern. Da aber alle Teilschritte immer gemeinsam ausgeführt werden müssen und die Laufzeit keines einzelnen gezielt beeinflusst werden kann, soll davon im weiteren abgesehen werden.

Das Histogramm in Abbildung 7.18 stellt die absoluten Häufigkeiten verschieden langer Laufzeiten der PI i. e. S. zur Reihenfolgeplanung mittels DBNe dar. Die Laufzeiten, die für verschiedene PIen gemessen worden sind, sind in Bereiche eingeteilt worden, von denen der am weitesten links liegende kleiner und der am weitesten rechts liegende größer als die sonst gleichgroßen

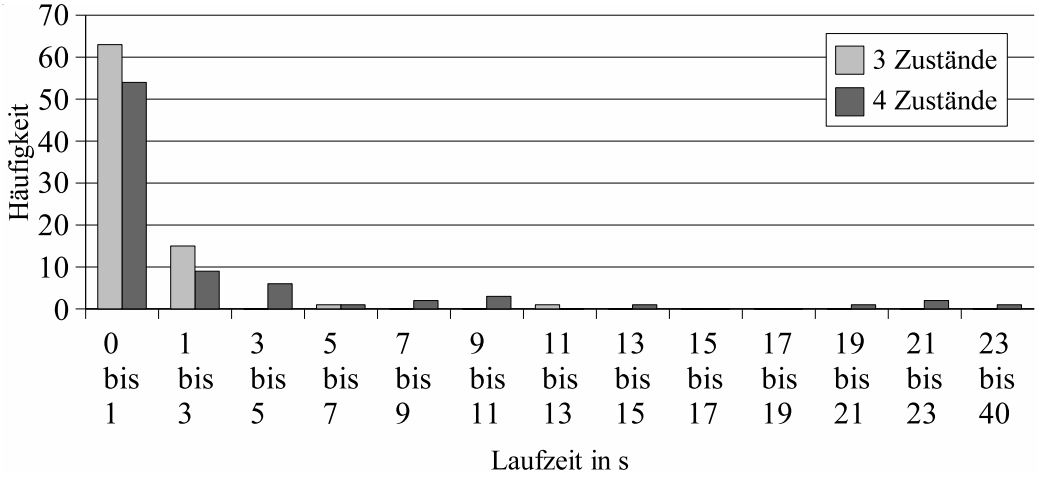

Abbildung 7.18: Histogramm der Laufzeit der PI i. e. S. für drei bzw. vier Zustände pro reellwertiger bzw. ganzzahliger Variable im DBN bzw. in der Inferenzmaschine

Bereiche gewählt worden ist, um die Kompaktheit und somit die Aussagekraft des Histogrammes zu erhöhen. Die Höhe einer jeden Säule gibt die absolute Häufigkeit des Auftretens einer Laufzeit im Bereich dieser Säule an. Das Histogramm unterscheidet nach DBNen bzw. Inferenzmaschinen, deren reellwertige und ganzzahlige Variablen in drei bzw. vier Intervalle bzw. Zustände diskretisiert worden sind.

Aus dem Histogramm in Abbildung 7.18 ist ersichtlich, dass die Laufzeit der PI i. e. S. im Mittel relativ gering ist, egal, ob Variablen mit drei oder mit vier Zuständen in der Inferenzmaschine vorkommen. Der Mittelwert für die Laufzeit der PI i. e. S. mit Variablen mit drei Zuständen beträgt rund 0,8 Sekunden, mit Variablen mit vier Zuständen rund 2,8 Sekunden. Das allübergreifende Maximum der Laufzeit für die PI i. e. S. beträgt knapp 40 Sekunden, so dass die Reihenfolgeplanung mittels DBNe jede Minute eine neue Prioritätsregel auswählen könnte, was in der vorliegenden Konstellation sicherlich nicht notwendig ist.

Es darf allerdings auch hier nicht außer acht gelassen werden, dass die DBNe nur jeweils über zwei Zeitscheiben entrollt worden sind. Aber selbst wenn das DBN, über dem die PI i. e. S. die maximale Laufzeit von knapp 40 Sekunden benötigt, über zehn Zeitscheiben entrollt worden wäre, hätte die PI nur $\frac{40}{2} \cdot 10 = 200$ Sekunden gedauert, vorausgesetzt, dass die Laufzeit der PI i. e. S. linear mit der Anzahl der Zeitscheiben ansteigt, was aufgrund der strukturell gleichen Zeitscheiben der DBNe zu vermuten ist. Eine Laufzeit von 3 Minuten und 20 Sekunden wäre in der vorliegenden Konstellation

durchaus akzeptabel gewesen, den Ansprüchen an eine Echtzeitverarbeitung genügt sie allerdings nicht.

Die Anzahl der Zustände pro Variable übt wiederum einen großen Einfluss auf die Laufzeit der PI i. e. S. aus, was aus dem starken Anstieg des Mittelwertes von 0,8 auf 2,8 Sekunden und des Maximalwertes von 11,8 auf 40 Sekunden geschlussfolgert wird, der aus dem Wechsel von Variablen mit drei Zuständen zu Variablen mit vier Zuständen resultiert. Zudem ist bei vier Zuständen pro Variable bereits mit bloßem Auge eine wesentlich größere Streuung zu erkennen. Die Laufzeit für die PI i. e. S. steigt im ungünstigsten Falle ja exponentiell mit der Anzahl der Zustände pro Variable. Um sie gering zu halten, sollten wiederum die Wertebereiche reellwertiger und ganzzahliger Variablen bewusst in möglichst wenige Intervalle untergliedert und auch rein ordinale oder nominale Werte gruppiert werden, sofern das inhaltlich möglich ist.

7.7 Beurteilung der Ergebnisse der empirischen Untersuchung bezüglich der mit ihr verfolgten Ziele

Die vorliegende empirische Untersuchung erlaubt es, das Potenzial BNe zur Unterstützung der PPS aus Sicht der Praxis einzuschätzen: BNe und insbesondere DBNe weisen nicht nur aus Sicht der Theorie (siehe Kapitel 5), sondern auch aus Sicht der Praxis ein Potenzial zur Unterstützung der PPS auf. Während der empirischen Untersuchung sind Indizien gesammelt worden, welche obige Aussage empirisch belegen:

So gelingt es durch die gezielte, periodische Auswahl von Prioritätsregeln durch DBNe, die Zielfunktionswerte dreier technizitärer Ersatzziele, der Termintreue, der Durchlaufzeit und der Kapitalbindung, z. T. drastisch zu verbessern. Einzig die Kapazitätsauslastung konnte nicht signifikant erhöht werden, weil auch bei effizienter Fertigung von extern nicht mehr Fertigungsaufträge eingesteuert worden sind.

Der fixe Anteil und der variable Anteil am Rechenaufwand für die Reihenfolgeplanung mittels DBNe ist insgesamt so gering gewesen, dass der Anwendung BNe zur PPS unter Aspekten der Laufzeit und des Speicherbedarfes nichts entgegensteht. Eher erlaubt es der geringe Rechenaufwand, rechenaufwendige Optimierungsalgorithmen zur automatischen Akquisition BNe und zum Aufbau der Inferenzmaschine einzusetzen und/oder umfangreichere BNe anzuwenden, was potenziell jeweils zu einer Verbesserung der Effizienz der BNe führt.

Die empirische Untersuchung hat zudem belegt, dass es nicht nötig ist, Methoden zur Wissensverarbeitung im Kontext BNe aufwendig zu parametrisieren: Schon durch eine erste und einzige Auswahl plausibler Methoden und das Belegen ihrer Parameter mit Standardwerten, sind gute Ergebnisse bei geringem Rechenaufwand erzielt worden. Das Feineinstellen oder der Austausch der Methoden zur Wissenverarbeitung im Kontext BNe sind nicht vonnöten gewesen und brauchten somit auch nicht vorgenommen zu werden, obwohl sie natürlich erfolgsträchtig sind. Darüber hinaus hat die automatische Akquisition DBNe nicht so sensibel auf das Ändern des Abtastintervalles reagiert, wie erwartet worden ist, sondern sehr robust, so dass man bei der Wahl des Abtastintervalles mit den in Abschnitt 4.4.2 präferierten Verfahren bereits sehr gute Ergebnisse erzielt.

Um eine hohe Aussagekraft der Ergebnisse der empirischen Untersuchung zu gewährleisten, sind 5.000 PIen i. e. S. für jedes Experiment vollzogen worden, die gemeinsam mit den fixen Anteilen am Rechenaufwand für die Reihenfolgeplanung mittels DBNe bereits knapp zweieinhalb Wochen Nettolaufzeit in Anspruch genommen haben, was einer wesentlich höheren Bruttolaufzeit entspricht. Jeder weitere untersuchte Parameter hätte die Laufzeit mindestens verdoppelt.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Zusammenfassung

Die PPS wartet mit einer Reihe von Problemen auf, die derzeit noch nicht oder noch nicht zufriedenstellend gelöst werden. Bestimmte Charakteristika, wie z. B. Strukturdefekte, kennzeichnen diese Probleme. BNe besitzen Eigenschaften und Fähigkeiten, die die begründete Vermutung zulassen, dass BNe in der Lage sind, Probleme mit solchen Charakteristika und somit auch die Probleme der PPS zu lösen oder zumindest ihre Lösung zu unterstützen.

Das Ziel der vorliegenden Arbeit hat darin bestanden, das Potenzial BNe zur Unterstützung der PPS auszuloten. Zu diesem Zwecke sind in Kapitel 2 diejenigen Probleme der PPS identifiziert worden, die bisher noch nicht oder noch nicht zufriedenstellend gelöst werden und demzufolge einer besseren Lösung bedürfen. Die übergreifenden Charakteristika dieser PPS-Probleme sind herausgearbeitet worden, um später nicht das Potenzial BNe zur Lösung jedes einzelnen PPS-Problems beurteilen zu müssen, sondern um das Potenzial BNe zur Lösung von Problemen mit diesen Charakteriatika einzuschätzen und daraus zu schließen, wie gut BNe die Lösung der Probleme der PPS unterstützen, die diese Charakteristika tragen.

Um sicherzugehen, dass BNe nicht auf Probleme der PPS bzw. auf Probleme mit deren Charakteristika angewandt werden, deren Lösung alternative Verfahren eventuell bereits besser unterstützen, hat Kapitel 3 solche alternativen Verfahren zur Unterstützung der PPS analysiert und ist zu dem Schluss gelangt, dass diese Verfahren die Probleme der PPS auch nicht allgemeingültig effizient lösen, BNe den alternativen Verfahren durchaus ebenbürtig sind und es lohnt, das Potenzial BNe zur Unterstützung der PPS weiter zu untersuchen.

Im Anschluss hat Kapitel 4 die Wissensverarbeitung im Kontext BNe so-

wohl abstrakt als auch anhand eines durchgängigen Beispiels detailliert beschrieben. Aus der Beschreibung der Wissensverarbeitung im Kontext BNe sind die bereits angesprochenen Eigenschaften und Fähigkeiten BNe abgeleitet worden, welche ein Potenzial BNe zur Unterstützung der PPS vermuten lassen, wie z. B. das „gemischte Schließen“ zur Entscheidungsunterstützung über einem kausalen Modell in einem Schritte, die Akquisition BNe automatisch aus Daten und/oder manuell aus Expertenwissen, das Verarbeiten von Datensätzen mit fehlenden Werten, das Entdecken „versteckter“ Variablen, die Akzeptanz, Interpretierbarkeit und Plausibilit der Wissensverarbeitung im Kontext BNe, um nur einige aufzuführen.

Das folgende Kapitel 5 hat das Potenzial BNe zur Unterstützung der PPS theoretisch betrachtet, indem es den Charakteristika der Probleme der PPS die Eigenschaften und Fähigkeiten der Wissensverarbeitung im Kontext BNe gegenüberstellt und darlegt, weshalb und auf welche Art und Weise BNe in der Lage sind, die Lösung von Problemen mit solchen Charakteristika und somit auch die Lösung der Probleme der PPS zu unterstützen.

Um das Potenzial BNe zur Unterstützung der PPS empirisch zu untersuchen, ist ein leistungsfähiges Softwaresystem entworfen worden, welches die Wissensverarbeitung im Kontext BNe vollzieht bzw. unterstützt und weitere PPS-spezifische Komponenten aufweist. Kapitel 6 beschreibt dieses Softwaresystem zur Unterstützung der PPS mittels BNe. Das Softwaresystem wird als ein wesentliches Ergebnis der vorliegenden Arbeit betrachtet, weil in ihm die Wissensanwendung im Kontext BNe so implementiert worden ist, dass sie möglichst wenig Rechenaufwand verursacht, was für seinen Einsatz in der PPS die entscheidende Voraussetzung ist, und weil es – im Gegensatz zu anderen Softwaresystemen zur Wissensverarbeitung im Kontext BNe – alle Eigenschaften aufweist, die nötig sind, um es zur Unterstützung der PPS einzusetzen.

Kapitel 7 hat das theoretisch fundiert belegte Potenzial BNe zur Unterstützung der PPS mittels einer umfassenden empirischen Studie untermauert. BNe sind durchgängig in der Lage, Prioritätsregeln situationsabhängig so auszuwählen, dass technizitäre Ersatzziele der PPS stark positiv beeinflusst werden. Zudem ist der Rechenaufwand für eine Wissensanwendung im Kontext BNe experimentübergreifend so gering ausgefallen, dass einem Einsatz BNe auch für größere Probleme mit mehr Variablen nichts im Wege steht. Besonders positiv hervorzuheben ist der Umstand, dass die Parameter der Wissensverarbeitung im Kontext BNe und vor allem die Wahl des Abtastintervalles keinen entscheidenden Einfluss auf Lösungsgüte und Rechenaufwand ausüben.

Wissensakquisition, -repräsentation und -anwendung im Kontext BNe weisen ergo sowohl unter theoretischem als auch unter praktischem Aspekt

ein hohes Potenzial zur Unterstützung der PPS auf, welches unbedingt erschlossen werden sollte.

8.2 Ausblick

Um das Potenzial der Wissensverarbeitung im Kontext BNe zu erschließen, sind BNe – vorerst versuchsweise – auch auf andere offene Probleme der PPS und nicht nur auf die Reihenfolgeplanung anzuwenden. Ein Problem, für das BNe besonders geeignet scheinen, ist die Primärbedarfsprognose bei kundenanonymer Fertigung. Aber BNe können und sollten auch die Lösung der anderen schlechtstrukturierten Probleme der PPS direkt oder indirekt unterstützen.

Zudem steht ein direkter Vergleich BNe mit alternativen und sinnvoll vergleichbaren Verfahren, wie z. B. KNNen, FRSen oder XPSen, aus. Sämtliche Verfahren lösen dann dieselben Probleme und werden hinsichtlich Lösungsgüte und Rechenaufwand beurteilt und miteinander verglichen. Der manuelle Aufwand für das Erstellen der Wissensrepräsentationsformen für die hier aufgeführten Verfahren sollte allerdings nicht unterschätzt werden.

Weitere Experimente zielen erstens darauf ab, den Zusammenhang zwischen den Eigenschaften der Fertigung, wie z. B. dem Organisationsgrad, auf der einen Seite und der Lösungsgüte und dem Rechenaufwand für die Wissensverarbeitung im Kontext BNe auf der anderen Seite empirisch zu bestimmen. Zweitens müssen Experimente durchgeführt werden, um zu ermitteln, ob ein Zusammenhang zwischen den Parametern der Verfahren der Wissensverarbeitung im Kontext BNe einerseits und der Lösungsgüte und dem Rechenaufwand der Verfahren andererseits besteht.

Eine einzelne Wissensanwendung im Kontext BNe benötigt nicht viel Laufzeit, aber Millionen von Wissensanwendungen, wie sie bei umfangreichen Experimenten vorgenommen werden, in ihrer Gesamtheit schon. Um die eben vorgeschlagenen Experimente trotzdem in akzeptabler Zeit durchzuführen, sind insbesondere die Algorithmen und Datenstrukturen für die Wissensanwendung im Kontext BNe zu verteilen und/oder zu parallelisieren. Ansätze dazu liegen vor.

Um die Wissensanwendung im Kontext BNe für weitere Experimente und auch für eine schnellere PPS noch stärker zu beschleunigen, müssen zum Aufbau der Inferenzmaschine nicht nur gierige Suchverfahren, sondern Optimierungsverfahren eingesetzt werden, welche zwar höheren fixen Aufwand für den Aufbau der Inferenzmaschine verursachen, aber geringeren variablen Aufwand für die Wissensanwendung i. e. S. nach sich ziehen, so dass der Gesamtaufwand für die Wissensanwendung bei den Experimenten sinkt.

Nicht zuletzt wird die grafische Benutzeroberfläche des Softwaresystems zur Unterstützung der PPS mittels BNe um eine Zoomfunktion ergänzt, so dass auch große BNe im Überblick angezeigt werden können und die Plausibilität und Interpretierbarkeit BNe und ihre Akzeptanz bei den Anwendern, die bereits per se hoch sind, weiter steigen.

Anhang A

Wissensverarbeitung im Kontext BNe

A.1 Grundlagen der Wissensverarbeitung im Kontext BNe

A.1.1 Potenzialfunktionen und Operationen über Potenzialfunktionen

Es sei $I = \{1, \ldots, n\}$ eine endliche Indexmenge mit $n = |I|$. Es sei $V = \{x_i | i \in I\}$ eine Menge diskreter Variablen. Die Menge I stellt also die Menge V dar. Es sei S_i die endliche Menge der Zustände einer Variable x_i, und es gelte $|S_i| \geq 2$.

Das Beispiel „Marie und ihre Rosen“ aus Abschnitt 4.1.1 soll hier weitergeführt werden und die voranstehenden und folgenden Ausführungen zu Potenzialfunktionen und Operationen über ihnen veranschaulichen.

Definition A.1 (Zustandsraum)
Es sei $J \subseteq I$ die $X \subseteq V$ darstellende Indexmenge. Das kartesische Produkt $S_X = \underset{i \in J}{\times} S_i$ ist der Zustandsraum von X.

Am Beispiel: Die Indexmenge $J = \{1, 3\}$ stelle die Menge $X = \{re, st\}$ dar. Der Zustandsraum von X ist demzufolge

$$S_X = \{(ne_{re}, tr_{st}), (ne_{re}, na_{st}), (ja_{re}, tr_{st}), (ja_{re}, na_{st})\}.$$

Definition A.2 (Konfiguration)
Es seien J, X und S_X wie vorab vereinbart. Ein $|X|$- bzw. $|J|$-Tupel $s \in S_X$ heißt Konfiguration (oder Zustand) von X.

Tabelle A.1: Beispiel für eine Potenzialfunktion von $X = \{re, st\}$ a) als Menge geschachtelter 2-Tupel bzw. Paare und b) als Tabelle

a)
$$\begin{array}{ll} \{((ne_{re},tr_{st}), & 0{,}72), \\ \ ((ne_{re},na_{st}), & 0{,}08), \\ \ ((ja_{re},tr_{st}), & 0{,}002), \\ \ ((ja_{re},na_{st}), & 0{,}198)\} \end{array}$$

b)

re	st	$f[(\cdot,\cdot)]$
ne_{re}	tr_{st}	0,72
ne_{re}	na_{st}	0,08
ja_{re}	tr_{st}	0,002
ja_{re}	na_{st}	0,198

Tabelle A.2: Beispiel für a) eine Einheitspotenzialfunktion von $X = \{re, st\}$ und b) eine Bewertungsfunktion von $X \cup Y$ mit $X = \{st\}$ und $Y = \{re\}$

a)

re	st	$u[(\cdot,\cdot)]$
ne_{re}	tr_{st}	1,0
ne_{re}	na_{st}	1,0
ja_{re}	tr_{st}	1,0
ja_{re}	na_{st}	1,0

b)

re	st	$p[(\cdot,\cdot)]$
ne_{re}	tr_{st}	0,9
ne_{re}	na_{st}	0,1
ja_{re}	tr_{st}	0,01
ja_{re}	na_{st}	0,99

Eine Konfiguration bzw. ein Zustand von X wäre z. B. das 2-Tupel bzw. Paar (ne_{re}, tr_{st}).

Definition A.3 (Potenzialfunktion)
Es seien X und S_X wie vorab vereinbart. Eine (diskrete) Potenzialfunktion f von X ist eine Abbildung f vom Zustandsraum S_X in die Menge $\mathbb{R}^{\geq 0}$ der nicht-negativen reellen Zahlen.

Eine (gemeinsame) Wahrscheinlichkeitsfunktion [Fis73b, S. 62] von $X \subseteq V$ ist eine spezielle Potenzialfunktion f von X, für die gilt $\sum_{s \in S_X} f(s) = 1$. Eine Potenzialfunktion von $X = \{re, st\}$, die gleichzeitig eine Wahrscheinlichkeitsfunktion ist, stellt Tabelle A.1 dar. Links steht die Potenzialfunktion als Menge geschachtelter 2-Tupel bzw. Paare, rechts als Tabelle. Von nun an werden Potenzialfunktionen immer als übersichtlichere Tabellen dargestellt.

Definition A.4 (Einheitspotenzialfunktion)
Es seien X und S_X wie vorab vereinbart. Die Einheitspotenzialfunktion u von X ist eine spezielle Potenzialfunktion von X, für die gilt $\forall s \in S_X(u(s) = 1)$.

Tabelle A.2 a) enthält ein Beispiel für eine Einheitspotenzialfunktion von $X = \{re, st\}$. Wie definiert, sind alle Funktionswerte 1,0.

Definition A.5 (Bewertungsfunktion)
Es seien $J, K \subseteq I$ Indexmengen, welche die Mengen $X, Y \subseteq V$ darstellen. Es

Tabelle A.3: Multiplikation einer Potenzialfunktion f für $X = \{re\}$ und einer Potenzialfunktion g für $Y = \{re, st\}$

re	st	$(f \cdot g)[(\cdot,\cdot)]$
ne_{re}	tr_{st}	0,72
ne_{re}	na_{st}	0,08
ja_{re}	tr_{st}	0,002
ja_{re}	na_{st}	0,198

$\leftarrow$

re	$f[(\cdot)]$
ne_{re}	0,8
ne_{re}	0,2

$\cdot$

re	st	$g[(\cdot,\cdot)]$
ne_{re}	tr_{st}	0,9
ne_{re}	na_{st}	0,1
ja_{re}	tr_{st}	0,01
ja_{re}	na_{st}	0,99

seien S_X, S_Y und $S_{X\cup Y}$ die Zustandsräume von X, Y und $X \cup Y$. Es sei π_X eine Projektion von $S_{X\cup Y}$ auf S_X, und π_Y sei eine Projektion von $S_{X\cup Y}$ auf S_Y. Eine Bewertungsfunktion p von $X \cup Y$ ist eine spezielle Potenzialfunktion von $X \cup Y$, die definiert ist als

$$p(s) = \Pr[X = \pi_X(s) | Y = \pi_Y(s)]$$

für alle Konfigurationen $s \in S_{X\cup Y}$.

Tabelle A.2 b) zeigt ein Beispiel für eine Bewertungsfunktion von $X = \{st\}$ vereinigt mit $Y = \{re\}$. Wie zu sehen, ist $\sum_{\{s \in S_{X\cup Y} | \pi_Y(s) = z\}} p(s) = 1$ für alle $z \in S_Y$. Eine Bewertungsfunktion enthält faktisch für jeden Zustand $z \in S_Y$ von Y eine durch $Y = z$ bedingte Wahrscheinlichkeitsfunktion [Fis73b, S. 62] von X.

Definition A.6 (Produkt zweier Potenzialfunktionen)
Es seien X, Y, $S_{X\cup Y}$, π_X und π_Y wie vorab vereinbart. Es seien f und g Potenzialfunktionen von X bzw. Y. Das Produkt $f \cdot g$ ist eine Potenzialfunktion von $X \cup Y$ und definiert als

$$(f \cdot g)(s) = f[\pi_X(s)] \cdot g[\pi_Y(s)]$$

für alle Konfigurationen $s \in S_{X\cup Y}$.

Tabelle A.3 veranschaulicht die Multiplikation einer Potenzialfunktion f für $X = \{re\}$ und einer Potenzialfunktion g für $Y = \{re, st\}$. Um den Wert der Potenzialfunktion $f \cdot g$ an der Stelle $s \in S_{X\cup Y}$ zu bestimmen, werden die Potenzialfunktionen f und g an den Stellen $\pi_X(s)$ und $\pi_Y(s)$ miteinander multipliziert. Für $s = (ne_{re}, tr_{st})$ rechnet man wie folgt:

$$\begin{aligned}
(f \cdot g)[(ne_{re}, tr_{st})] &= f[\pi_X((ne_{re}, tr_{st}))] \cdot g[\pi_Y((ne_{re}, tr_{st}))] \\
&= f[(ne_{re}, tr_{st})] \cdot g[(ne_{re})] \\
&= 0{,}9 \cdot 0{,}8 \\
&= 0{,}72.
\end{aligned}$$

Bei f handelt es sich im Beispiel übrigens um eine eindimensionale Wahrscheinlichkeitsfunktion von X, bei g um eine Bewertungsfunktion von Y und bei $f \cdot g$ um eine gemeinsame Wahrscheinlichkeitsfunktion von $X \cup Y = \{re, st\}$. – Die Multiplikation lässt sich auch auf mehr als zwei Potenzialfunktionen erweitern:

Definition A.7 (Produkt von Potenzialfunktionen)
Es sei $J_k \subseteq I$ mit $k = 1, \ldots, q$ die $X_k \subseteq V$ darstellende Indexmenge. Es sei S_k der Zustandsraum von X_k. Es sei $X = \bigcup_{i=1}^{q} X_i$ mit dem Zustandsraum S. Es sei π_k die Projektion von S auf S_k. Es sei f_k die Potenzialfunktion von X_k. Das Produkt $\prod_{i=1}^{q} f_i$ ist eine Potenzialfunktion von X und definiert als

$$\left(\prod_{i=1}^{q} f_i\right)(s) = \prod_{j=1}^{q} f_j[\pi_j(s)]$$

für alle Konfigurationen $s \in S$.

Definition A.8 (Produkt Potenzialfunktion und Skalar)
Es sei $J \in I$ die $X \in V$ darstellende Indexmenge, und S sei der Zustandsraum von X. Es sei f eine Potenzialfunktion von X, und r sei ein Skalar. Das Produkt $f \cdot r$ ist eine Potenzialfunktion von X und definiert als

$$(f \cdot r)(s) = f(s) \cdot r$$

für alle $s \in S$.

Um eine Potenzialfunktion f mit einem Skalar r zu multiplizieren, werden also alle Funktionswerte von f mit r multipliziert.

Definition A.9 (Quotient zweier Potenzialfunktionen)
Es seien X, Y, $S_{X\cup Y}$, π_X, π_Y, f und g wie in Definition A.6 verwandt. Der Quotient f/g ist eine Potenzialfunktion von $X \cup Y$ und definiert als

$$\left(\frac{f}{g}\right)(s) = \frac{f[\pi_X(s)]}{g[\pi_Y(s)]} \quad \textit{mit} \quad g[\pi_Y(s)] > 0$$

für alle Konfigurationen $s \in S_{X\cup Y}$. Ist sowohl $f[\pi_X(s)] = 0$ als auch $g[\pi_Y(s)] = 0$, so sei auch $(f/g)(s) = 0$.

Definition A.10 (Quotient aus Potenzialfunktion und Skalar)
Es seien X, S, f und r wie in Definition A.8 verwandt. Der Quotient f/r ist eine Potenzialfunktion von X und definiert als

$$\left(\frac{f}{r}\right)(s) = \frac{f(s)}{r} \quad \textit{mit} \quad r > 0$$

für alle $s \in S$.

Tabelle A.4: Herausintegrieren eines Summenrandes g von $X = \{st\}$ aus einer Potenzialfunktion f von $Y = \{re, st\}$

st	$g[(\cdot)]$
tr_{st}	0,722
na_{st}	0,278

$= \sum_{Y \setminus X}$

re	st	$f[(\cdot,\cdot)]$
ne_{re}	tr_{st}	0,72
ne_{re}	na_{st}	0,08
ja_{re}	tr_{st}	0,002
ja_{re}	na_{st}	0,198

Um eine Potenzialfunktion f durch einen Skalar r zu dividieren, werden also alle Funktionswerte von f durch r dividiert.

Definition A.11 (Summenrand)
Es seien $J \subseteq K \subseteq I$ Indexmengen, welche die Mengen $X \subseteq Y \subseteq V$ darstellen. Es seien S_X und S_Y die Zustandsräume von X und Y. Es sei π_X eine Projektion von S_Y auf S_X. Es sei f eine Potenzialfunktion von Y. Der Summenrand $\sum_{Y \setminus X} f$ von X in der Potenzialfunktion f ist eine Potenzialfunktion von X und definiert als

$$\left(\sum_{Y \setminus X} f\right)(s_X) = \sum_{\{s_Y \in S_Y | \pi_X(s_Y) = s_X\}} f(s_Y)$$

für alle Konfigurationen $s_X \in S_X$.

Das Bilden eines (Summen)randes wird aus als Herausintegrieren bezeichnet. [Kle80, S. 259] Tabelle A.4 zeigt das Herausintegrieren eines Summenrandes g von $X = \{st\}$ aus einer Potenzialfunktion f von $Y = \{re, st\}$. Um den Wert der Potenzialfunktion g an der Stelle $s_X \in X$ zu bestimmen, werden die Funktionswerte $s_Y \in S_Y$ der Potenzialfunktion f addiert, für die gilt $\pi_X(s_Y) = s_X$. Für $s_X = (tr_{st})$ rechnet man wie folgt:

$$\begin{aligned}
\left(\sum_{Y \setminus X} f\right)[(tr_{st})] &= \sum_{\{s_Y \in S_Y | \pi_X(s_Y) = (tr_{st})\}} f(s_Y) \\
&= f[(ne_{re}, tr_{st})] + f[(ja_{re}, tr_{st})] \\
&= 0{,}72 + 0{,}002 \\
&= 0{,}722.
\end{aligned}$$

Wenn – wie im Beispiel der Fall – die Potenzialfunktion f von Y eine Wahrscheinlichkeitsfunktion Pr ist, dann ist der Summenrand $\sum_{Y \setminus X} f$ eine Randwahrscheinlichkeitsfunktion [Fis73b, S. 67] von X in Pr.

Definition A.12 (Maximumrand)
Es seien X, Y, S_X, S_Y, π_X und f wie vorab vereinbart. Der Maximumrand $\mathrm{Max}_{Y \setminus X} f$ *von X in der Potenzialfunktion f ist eine Potenzialfunktion von X und definiert als*

$$\left(\underset{Y \setminus X}{\mathrm{Max}} f\right)(s_X) = \max_{\{s_Y \in S_Y | \pi_X(s_Y) = s_X\}} f(s_Y)$$

für alle Konfigurationen $s_X \in S_X$.

Der Maximumrand wird ähnlich herausintegriert wie der Summenrand, nur dass der Operator $\sum$ durch den Operator max ersetzt wird. Ergo werden die relevanten Funktionswerte nicht addiert, sondern ihr Maximum wird bestimmt.

A.1.2 Grundlagen der Graphentheorie

Die im folgenden dargestellten Grundlagen der Graphentheorie sind stark auf ihrer Nutzung in der vorliegenden Arbeit ausgerichtet worden. Sie weichen deshalb – mehr oder minder stark – von Darstellungen in anderen Werken zur Graphentheorie, wie z. B. [CH94] und [Wes01], ab, orientieren sich eher an [Whi95, S. 58 ff.] und erheben keinen Anspruch, vollständig oder allgemeingültig zu sein.

Definition A.13 (Graph)
Es sei V eine endliche Menge diskreter Variablen. Es sei $E \subseteq (V \times V) \setminus \{(v, v) | v \in V\}$. Ein Graph G über V ist ein geordnetes Paar (V, E). Die Elemente von E heißen Kanten und die Elemente von V auch Knoten von G.

Laut Definition A.13 ist die Menge E also eine Teilmenge des kartesischen Produktes $V \times V$ *ohne* alle Paare (v, v), $v \in V$, mit identischen Knoten v als Elemente. Ein Graph $G = (V, E)$ weist demzufolge keine Schlingen genannten Kanten $(v, v) \notin E$ auf, die von einem Knoten $v \in V$ zu demselben Knoten v führen, da sie explizit aus dem kartesischen Produkt $V \times V$ entfernt worden sind.

Eine Kante $e \in E$ des Graphen G ist laut Definition A.13 ein geordnetes Paar (u, v) der Knoten u und v mit $u, v \in V$ und $u \neq v$. Aus Definition A.13 folgt: Der Graph $G = (E, V)$ weist keine Kante (u, v) mehrfach auf, da die Kanten E des Graphen G eine Teilmenge des kartesischen Produktes $V \times V$ sind (und nicht etwa mittels einer Inzidenzfunktion $I : E \to V \times V$ nach dem kartesischen Produkt $V \times V$ abbilden) und die Elemente einer Menge paarweise voneinander verschieden sind. [BSMM99, S. 288] Die Menge E der Kanten

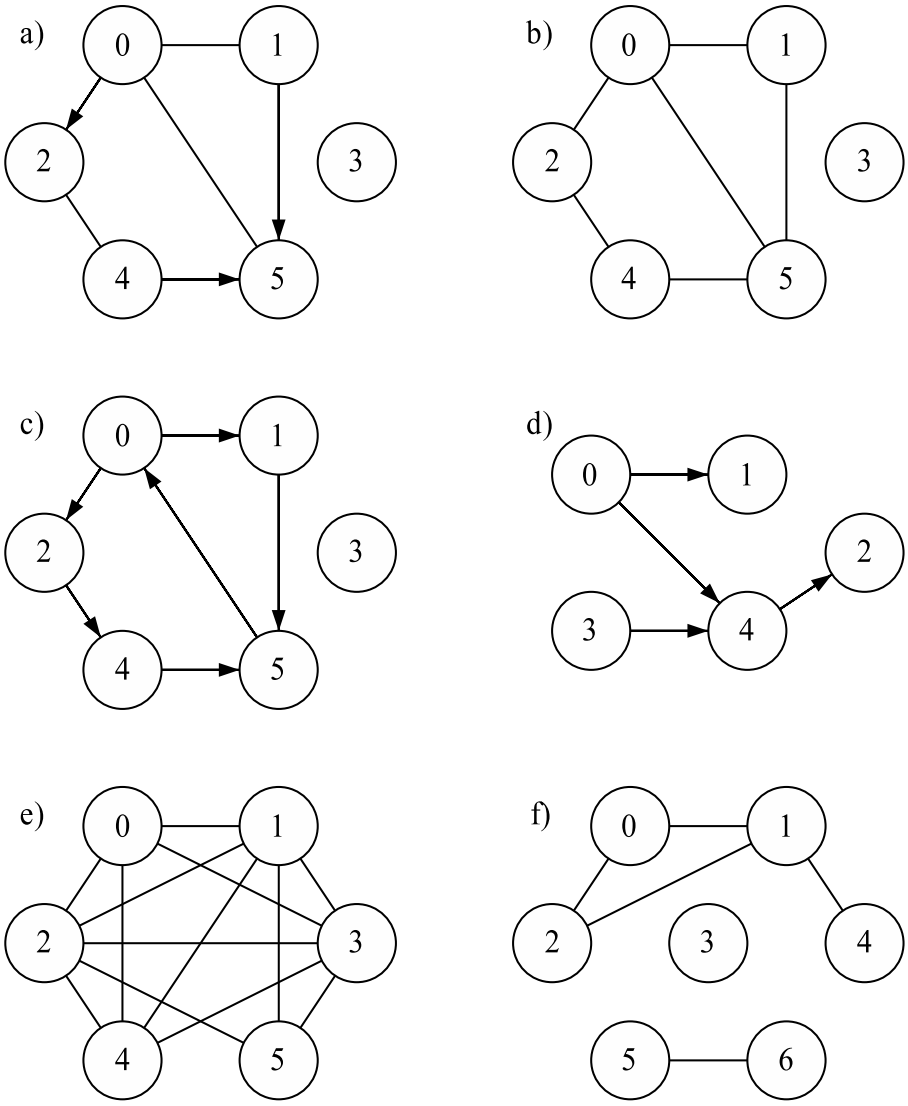

Abbildung A.1: Grafische Beispiele zu den Grundlagen der Graphentheorie

des Graphen G ist ebenfalls endlich, da die Menge der Knoten V des Graphen G endlich ist, demzufolge auch das kartesische Produkt $V \times V$ endlich ist und die Kanten $E \in V \times V$ eine Teilmenge des kartesischen Produktes $V \times V$ sind. Abbildung A.1 a) zeigt die grafische Repräsentation eines allgemeinen Graphen.[1] Seine Knotenmenge ist $V = \{0, 1, 2, 3, 4, 5, 6\}$. Seine Kantenmenge ist $E = \{(0, 1), (0, 2), (0, 5), (1, 0), (1, 5), (2, 4), (4, 2), (4, 5), (5, 0)\}$.

Im Kontext der vorliegenden Arbeit sind die Knoten des Graphen zumeist Variablen. Die Begriffe Knoten und Variable werden aus diesem Grund synonym verwandt. Sollte es sich bei den Knoten eines Graphen einmal nicht um Variablen handeln, so wird das explizit angegeben.

Definition A.14 (Ungerichteter Graph)
Der Graph $G = (V, E)$ ist ein ungerichteter Graph, wenn $\forall(u, v) \in E((v, u) \in E)$ gilt.

Laut Definition A.14 ist ein Graph $G = (V, E)$ also ungerichtet, wenn für jede Kante $(u, v) \in E$ auch ihre inverse Kante $(v, u) \in E$ vorliegt. Die Kante (u, v) und die ihr inverse Kante (v, u) gemeinsam werden auch als *ungerichtete Kante* bezeichnet und graphisch als u—v respektive als v—u repräsentiert. Abbildung A.1 b) stellt einen ungerichteten Graphen dar.

Definition A.15 (Gerichteter Graph)
Ein Graph $G = (V, E)$ ist ein gerichteter Graph, wenn $\forall(u, v) \in E((v, u) \notin E)$ gilt.

Laut Definition A.15 ist ein Graph $G = (V, E)$ also gerichtet, wenn für jede Kante $(u, v) \in E$ keine inverse Kante $(v, u) \notin E$ vorliegt. Die Kante (u, v) werde auch als *gerichtete Kante* bezeichnet und graphisch als $u \longrightarrow v$ respektive als $v \longleftarrow u$ repräsentiert. Abbildung A.1 c) stellt einen gerichteten Graphen dar.

Definition A.16 (Unterliegender Graph)
Es sei $G = (V, A)$ ein Graph, und es sei $U = (V, E)$ ein Graph. Es ist U ein dem Graphen G unterliegender Graph, wenn gilt $\forall(u, v) \in A((u, v), (v, u) \in E)$ und $\forall(u, v) \in E((u, v) \in A \vee (v, u) \in A)$.

Laut Definition A.16 liegt ein Graph $U = (N, E)$ einem Graphen $G = (V, A)$ unter, wenn erstens der Graph U dieselbe Knotenmenge wie der Graph G aufweist ($V \equiv V$), wenn zweitens für jede Kante $(u, v) \in A$ im Graphen G zwei Kanten im Graphen U vorliegen: die Kante $(u, v) \in E$ selbst und ihre inverse Kante $(v, u) \in E$, und wenn drittens für jede Kante $(u, v) \in E$ im

[1] Zwischen (Bestandteilen der) Graphen und ihren grafischen Repräsentationen wird ab jetzt vereinfachend nicht mehr unterschieden.

Graphen U die Kante $(u, v) \in A$ selbst oder ihre inverse Kante $(v, u) \in A$ im Graphen G vorliegt. Unter den genannten Bedingungen ist der Graph U zwangsläufig ein ungerichteter Graph. Abbildung A.1 b) stellt Graphen dar, der sowohl dem Graphen aus Abbildung A.1 a) als auch dem Graphen aus Abbildung A.1 c) unterliegt.

Definition A.17 (Eltern)
Es sei $G = (V, E)$ ein Graph. Die Eltern Pa_v eines Knotens $v \in V$ sind die Menge $Pa_v = \{u|(u, v) \in E\}$. Ein Element $pa_v \in Pa_v$ heißt Elter von $v \in V$.

Definition A.18 (Kinder)
Es sei $G = (V, E)$ ein Graph. Die Kinder Ch_v eines Knotens $v \in V$ sind die Menge $Ch_v = \{u|(v, u) \in E\}$. Ein Element $ch_v \in Ch_v$ heißt Kind von $v \in V$.

Definition A.19 (Nachbarn)
Es sei $G = (V, E)$ ein Graph. Die Nachbarn Ne_v eines Knotens $v \in V$ sind die Menge $Ne_v = Ch_v \cup Pa_v$. Ein Element $ne_v \in Ne_v$ heißt Nachbar von $v \in V$.

Definition A.20 (Familie)
Es sei $G = (V, E)$ ein Graph. Die Familie Fa_v eines Knotens $v \in V$ ist die Menge $Fa_v = \{v\} \cup Pa_v$.

Laut Definition A.17 sind die Eltern Pa_v eines Knotens $v \in V$ eines Graphen $G = (V, E)$ die Menge der ersten Elemente $u \in V$ aller Kanten $(u, v) \in E$, die den Knoten $v \in V$ als zweites Element aufweisen, und laut Definition A.18 sind die Kinder Ch_v eines Knotens $v \in V$ eines Graphen $G = (V, E)$ die Menge der zweiten Elemente $u \in V$ aller Kanten $(v, u) \in E$, die den Knoten $v \in V$ als erstes Element aufweisen, und laut Definition A.19 sind die Nachbarn Ne_v eines Knotens $v \in V$ eines Graphen $G = (V, E)$ die Vereinigung $Ne_v = Ch_v \cup Pa_v$ der Kinder Ch_v und der Eltern Pa_v des Knotens $v \in V$. In Abbildung A.1 d) sind $Pa_4 = \{0, 3\}$ die Eltern des Knotens 4, $Ch_0 = \{1, 4\}$ die Kinder des Knotens 0, $Ne_4 = \{0, 2, 3\}$ die Nachbarn des Knotens 4 und $Fa_4 = \{0, 2, 3, 4\}$ die Familie des Knotens 4.

Definition A.21 (Clique)
Es sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $U \subseteq V$ heißt Clique, wenn gilt $\forall u, v \in U(\{(u, v), (v, u)\} \subseteq E)$.

Definition A.22 (Maximale Clique)
Es sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $U \subseteq V$ heißt maximale Clique, wenn $\nexists v \in V \setminus U$, so dass $\{v\} \cup U$ wiederum eine Clique wäre.

Definition A.23 (Größe einer Clique)
Es sei $U = (V, E)$ ein ungerichteter Graph. Es sei C eine Clique in U mit den diskreten Variablen $c \in C$. Die Größe g der Clique C ist $g = \prod_{c \in C} |S_c|$, wobei S_c die Menge der Werte bzw. der Zustände der Variable c bezeichne.

Laut Definition A.21 ist eine Clique eine Teilmenge der Knoten eines ungerichteten Graphen, deren Knoten paarweise über ungerichtete Kanten miteinander verbunden sind. Eine Clique $U \subseteq V$ eines ungerichteten Graphen $G = (V, E)$ ist laut Definition A.22 maximal, wenn ihr kein weiterer Knoten $v \in V \setminus U$ hinzugefügt werden kann, so dass die resultierende Knotenmenge $\{v\} \cup U$ wiederum eine Clique wäre. Die im Kontext der vorliegenden Arbeit vorkommenden Cliquen sind zumeist maximal, weshalb eine maximale Clique oft lediglich als Clique bezeichnet wird. Sollte es sich bei einer Clique einmal nicht um eine maximale handeln, so wird das explizit angegeben. In Abbildung A.1 e) sind die Knoten $C = \{0, 1, 2, 3\}$ eine (nicht maximale) Clique, die Knoten $\{0, 1, 2, 3, 4\}$ eine maximale Clique und die Knoten $\{0, 1, 2, 3, 5\}$ *keine* Clique.

Die Größe einer Clique ist laut Definition A.23 das Produkt der Mächtigkeiten der Mengen der Werte bzw. der Zustände ihrer Variablen. Hätten die Wertemengen der Knoten der Clique C die Mächtigkeiten 2, 4, 3 und 2, so betrüge die Größe der Clique 48.

Definition A.24 (Moralischer Graph)
Es sei $D = (V, A)$ ein gerichteter und $U = (V, E)$ ein ungerichteter Graph. Es ist U der moralische Graph zum Graphen D, wenn gilt $E = \{(u, v), (v, u) | u, v \in V \wedge ((u, v) \in A \vee (v, u) \in A \vee (v, u \in Pa_w \wedge w \in V \wedge u \neq v))\}$[2].

Definition A.24 sagt aus, dass ein Graph $U = (V, E)$ ein moralischer Graph zu einem gerichteten Graphen $D = (V, A)$ ist, wenn für jede Kante $(u, v) \in A$ des Graphen D die Kanten $(u, v), (v, u) \in E$ im Graphen U vorkommen, wenn für alle voneinander verschiedenen Eltern u und w eines jeden Knotens $v \in V$ die Kanten (u, w) und (w, u) im Graphen U auftreten und wenn für alle Kanten $(u, v) \in E$ gilt, dass entweder die Kante (u, v) oder die Kante (v, u) im Graphen G vorliegt oder dass die Knoten u und v Eltern eines Knotens $w \in A$ sind. Abbildung 4.10 a) zeigt einen gerichteten Graphen, und Abbildung 4.10 b) den zu ihm moralischen Graphen.

Definition A.25 (Pfad)
Ein Pfad c in einem Graphen $G = (V, E)$ ist eine Folge $v_1, \ldots, v_k$ der Länge

[2] Alternative Bedingung für den moralischen Graphen: $\forall (u, v) \in A((u, v), (v, u) \in E)$ und $\forall v \in V \forall u, w \in Pa_v((u, w), (w, u) \in E | u \neq w)$ und $\forall (u, v) \in E((u, v) \in A \vee (v, u) \in A \vee (u, v \in Pa_w | w \in A))$

$k \geq 2$ von Knoten $\{v_1, \ldots, v_k\} \subseteq V$, für die gilt $(v_i, v_{i+1}) \in E$ für $i = 1, \ldots, k-1$.

Definition A.26 (Einfacher Pfad)
Es sei G ein Graph, und $c = v_1, \ldots, v_k$ sei ein Pfad der Länge k in G. Der Pfad c ist ein einfacher Pfad, wenn gilt $|\{v_1, \ldots, v_k\}| = k$.

Definition A.27 (Zyklus)
Ein Zyklus in einem Graphen G ist ein Pfad $v_1, \ldots, v_k, v_1$ in G.

Definition A.28 (Einfacher Zyklus)
Ein einfacher Zyklus in einem Graphen G ist ein Zyklus $v_1, \ldots, v_k, v_1$ in G, für den gilt $|\{v_1, \ldots, v_k\}| = k$.

Definition A.29 (Sehne (eines Zyklus'))
Es sei $G = (V, E)$ ein Graph, und $c = v_1, \ldots, v_k, v_1$ sei ein einfacher Zyklus in G. Eine Kante $(u, w) \in E$ ist eine Sehne von c, wenn gilt $u, w \in \{v_1, \ldots, v_k\}$ und $(u, w) \notin \bigcup_{i=1}^{k-1} \{(v_i, v_{i+1}), (v_{i+1}, v_i)\} \cup \{(v_k, v_1), (v_1, v_k)\}$.

Laut Definition A.25 ist ein Pfad c in einem Graphen $G = (V, E)$ also eine Folge $c = (v_1, v_1, \ldots, v_k)$ von k Knoten $v_i \in V$ für $i = 1, \ldots, k$, die mindestens zwei Knoten enthält ($k \geq 2$) und in der alle direkt aufeinanderfolgenden Paare von Knoten $(v_i, v_{i+1}) \in E$ für $i = 1, \ldots k-1$ Elemente der Menge E der Kanten des Graphen G sind. Laut Definition A.27 ist ein Zyklus in einem Graphen also ein Pfad, dessen erster Knoten gleich seinem letzten Knoten ist. In einem einfachen Zyklus unterscheiden sich alle Knoten paarweise voneinander; nur der erste und der letzte sind identisch. Eine Sehne ist laut Definition A.29 eine Kante zwischen zwei Knoten eines einfachen Zyklus', die mit keiner der Kanten zwischen zwei aufeinanderfolgenden Knoten des Zyklus' identisch ist. In Abbildung A.1 a) ist $\{4, 5, 0, 1, 5, 0, 2\}$ ein (nicht einfacher) Pfad, $\{4, 5, 0, 2\}$ ein einfacher Pfad, $\{2, 4, 5, 0, 1, 5, 0, 2\}$ ein (nicht einfacher) Zyklus, $\{4, 5, 0, 2, 4\}$ ein einfacher Zyklus und $(0, 5)$ eine Sehne des Zyklus' $\{4, 5, 1, 0, 2, 4\}$.

Definition A.30 (Vorfahren)
Es sei $G = (V, E)$ ein Graph. Die Vorfahren An_v eines Knotens $v \in V$ sind all die Knoten $u \in V$, für die gilt, dass es einen Pfad $u, \ldots, v$ von u nach v gibt.

Definition A.31 (Nachfahren)
Es sei $G = (V, E)$ ein Graph. Die Nachfahren De_u eines Knotens $u \in V$ sind all die Knoten $v \in V$, für die gilt, dass es einen Pfad $u, \ldots, v$ von u nach v gibt.

Laut Definition A.30 handelt es sich bei den Vorfahren An_v eines Knotens $v \in V$ also um diejenigen Knoten $u \in An_v \subseteq V$, von welchen aus man den Knoten v via Pfade im Graphen G erreicht. Bei den Nachfahren De_u eines Knotens $u \in V$ handelt es sich laut Definition A.31 hingegen um diejenigen Knoten $v \in De_v \subseteq V$, welche man vom Knoten u aus via Pfade im Graphen G erreicht. In Abbildung A.1 d) sind $An_2 = \{0, 3, 4\}$ die Vorfahren des Knotens 2 und $\{1, 2, 4\}$ die Nachfahren des Knotens 0.

Definition A.32 (Gerichteter, azyklischer Graph)
Ein gerichteter Graph $G = (V, E)$ ist ein gerichteter, azyklischer Graph (DAG), wenn er keine Zyklen aufweist.

Laut Definition A.32 kann man sich in einem gerichteten, azyklischen Graphen also nicht entlang und in Richtung der (gerichteten) Kanten, von einem Knoten v ausgehend, zum Knoten v zurück bewegen. Der dem gerichteten, azyklischen Graphen unterliegende Graph darf allerdings Zyklen aufweisen. Abbildung B.5 zeigt einen gerichteten, azyklischen Graphen.

Definition A.33 (Triangulärer Graph)
Ein ungerichteter Graph U heißt triangulär, wenn jeder einfache Zyklus der Länge $k \geq 4$ in U mindestens eine Sehne aufweist.

Ein ungerichteter Graph $U = (V, E)$ ist laut Definition A.33 dann triangulär, wenn in U kein einfacher Pfad existiert, der von einem Knoten $v \in V$ zurück zu demselben Knoten v führt, vier oder mehr Knoten umfasst und nicht mindestens eine Sehne aufweist. Der Graph in Abbildung A.1 b) ist nicht triangulär. Würde man jedoch die Menge $\{(2, 5)\}$, die eine Kante bzw. Sehne enthält, seiner Kantenmenge hinzufügen, wäre er es.

Definition A.34 (Zusammenhängende Knoten)
Es sei G ein Graph, und $U = (N, E)$ sei der dem Graphen G unterliegende Graph. Zwei Knoten $u, v \in N$ heißen zusammenhängend in G, wenn es mindestens einen Pfad $u, \ldots, v$ in U gibt.

Definition A.35 (Zusammenhängender Graph)
Ein Graph $G = (V, E)$ hängt zusammen, wenn alle Knoten $u, v \in V$ zusammenhängen.

Zwei Knoten $u, v \in V$ eines Graphen $G = (V, A)$ hängen laut Definition A.34 also dann zusammen, wenn es gelingt, sich entlang der Kanten $\subseteq A$ vom Knoten u zum Knoten v zu bewegen, ohne die Richtung der Kanten zu beachten. Wenn alle Knoten $u, v \in V$ in einem Graphen $G = (V, E)$ paarweise

zusammenhängen, dann ist laut Definition A.35 der Graph G zusammenhängend. In Abbildung A.1 f) hängen die Knoten 0 und 4 zusammen, die Knoten 0 und 5 aber nicht. Die Graphen in Abbildung A.1 d) und e) sind zusammenhängend, aber die in Abbildung A.1 a), b), c) und f) nicht.

Definition A.36 (Komponente eines Graphen)
Es sei $G = (V, A)$ ein Graph, und $K \subseteq V$ sei eine Teilmenge von V. Die Menge K ist eine Komponente von G, wenn gilt, dass alle Knoten $u, v \in K$ zusammenhängen, und $\forall u \in K \forall w \in V \setminus K((u, w), (w, u) \notin A)$.

Laut A.36 ist eine Knotenmenge $K \subseteq V$ eines Graphen $G = (V, A)$ dann eine Komponente von G, wenn ihre Knoten $u, v \in K$ zusammenhängen, aber nicht mit den restlichen Knoten $w \in V \setminus K$ des Graphen G über mindestens eine Kante verbunden sind. Abbildung A.1 stellt einen Graphen mit drei Komponenten dar: $\{0, 1, 2, 4\}$, $\{3\}$ und $\{5, 6\}$.

Definition A.37 (Baum)
Es sei G ein Graph, und $U = (N, E)$ sei der dem Graphen G unterliegende Graph. Der Graph G ist ein Baum, wenn es für alle Knoten $u, v \in N$ mit $u \neq v$ genau einen einfachen Pfad $u, \ldots, v$ in U gibt.

Mit anderen Worten sagt Definition A.37 aus, dass ein Graph $G = (V, A)$ dann ein Baum ist, wenn er zusammenhängend ist, aber zwei Knoten $u, v \in V$ mit $u \neq v$ im G unterliegenden Graphen nur über genau einen einfachen Pfad zusammenhängen. Abbildung A.1 d) zeigt einen Graphen, der ein Baum ist.

A.2 Algorithmen zum automatischen Erstellen BNe aus Daten

Im Folgenden werden der K2-Algorithmus [CH92, S. 321 ff.] und der B-Algorithmus [Bun91, S. 56 f.] zur Suche BNe hoher Güte vorgestellt. Beide Algorithmen starten mit einem BN bzw. mit einem gerichteten, azyklischen Graphen ohne Kanten und suchen „gierig“, das heißt, dass sie das BN sukzessive aufbauen und nur Verbesserungen zulassen bzw. Kanten hinzufügen und es nicht erlauben, einem lokalen Optimum durch das temporäre Entfernen von Kanten bzw. eine temporäre Verschlechterung des BNes zu entkommen. Der K2-Algorithmus verlangt zusätzlich eine „kausale“ Ordnung der Variablen in dem Sinne, dass sich sämtliche Ursachen vor ihren Wirkungen bzw. sämtliche Wirkungen nach ihren Ursachen befinden müssen. Das liegt darin begründet, dass K2 keine Kante (v, u) einfügen kann, wenn sich eine Variable u im Sinne der Ordnung vor einer Variable v befindet. Diese Ordnung lässt

sich oft relativ einfach manuell bestimmen, indem man das zeitliche Nacheinander des Auftretens von Ereignissen berücksichtigt. Den Pseudokode für K2 stellt Algorithmus A.1 dar. Initial entfernt K2 alle Kanten im BN, indem

Algorithmus A.1 K2-Algorithmus

Eingabe: die n Variablen V eines BNes in einer kausalen Ordnung, eine Datenbasis D über den Variablen V, ein Gütemaß q

Ausgabe: die Eltern π_v eines jeden Knotens $v \in V$

```
1:  prozedur K2(V, D)
2:      für i ← 1, ..., n führe aus
3:          π_i = ∅
4:      ende für
5:      für i ← 2, ..., n führe aus
6:          wiederhole
7:              v ← argmax_{u ∈ {v_1,...,v_{i-1}} \ π_i} q(v_i, π_i ∪ {u}, D)
8:              Δ ← q(v_i, π_i ∪ {v}, D) − q(v_i, π_i, D)
9:              wenn Δ > 0 dann
10:                 π_i ← π_i ∪ {v}
11:             ende wenn
12:         bis Δ ≤ 0 ∨ π_i = {v_1, ..., v_{i-1}}
13:     ende für
14: ende prozedur
```

er allen Variablen ihre Eltern nimmt. Dann durchläuft K2 die Knoten von v_2 bis $v_{|V|}$. Variable v_1 kann demzufolge keine Eltern aufweisen und darf es laut obiger Festlegung auch nicht. In einer inneren, fußgesteuerten Schleife wählt K2 aus den Variablen, die in der Ordnung vor v_i liegen und noch keine Eltern von v_i sind, diejenige Variable v aus, die als zusätzliches Elter die Güte der Variable v_i bezüglich der Datenbasis D maximiert.[3] Dann bestimmt er den Gütezuwachs Δ zwischen den Güten der Variable v_i mit und ohne Variable v als Elter. Ist Δ größer 0, wird die Variable v den Eltern π_i der Variablen v_i hinzugefügt. Die innere Schleife bricht ab, wenn durch das Hinzufügen eines Knotens v zu den Eltern π_i der Variablen v_i kein weiterer Gütezuwachs erzielt werden würde oder wenn alle Variablen, die sich im Sinne der Ordnung vor Variable v_i befinden, bereits Eltern der Variable v_i sind. Endet auch die äußere Schleife, sind die Eltern aller Variablen determiniert und somit die Kanten des BNes bestimmt.

[3] Der Operator $\text{argmax}_{x \in X}(f(x))$ gibt ein beliebiges Element der Menge $\{y \in X | g(y) = \max_{z \in X}(g(z))\}$ zurück.

Der B-Algorithmus bedarf keiner kausalen Ordnung der Variablen. Er sucht die Kante, die – eingefügt – dem BN den höchsten Gütezuwachs brächte, und fügt sie ein, sofern der Gütezuwachs positiv ist und die Kante keinen Zyklus verursacht. Dem B-Algorithmus liegt als Datenstruktur eine zweidimensionale $n \times n$-Matrix A zugrunde. Ein Element $A[i, j]$ dieser Matrix gibt entweder den Gütezuwachs an, den eine Kante (v_j, v_i) dem BN brächte, oder dass keine Kante erlaubt bzw. die Kante bereits enthalten ist. Im ersten Fall ist der Gütezuwachs $A[i, j] = q(v_i, \pi_i \cup \{v_j\}, D) - q(v_i, \pi_i, D)$; im zweiten Fall ist $A[i, j] = -\infty$. Algorithmus A.2 gibt B im Pseudokode wieder. Zu Beginn entfernt B alle Kanten aus dem BN, indem er die Eltern aller Variablen austrägt. Im Anschluss berechnet er den initialen Gütezuwachs für alle Elemente $A[i, j]$ mit $i \neq j$. Alle Elemente $A[i, j]$ mit $i = j$ werden auf $-\infty$ gesetzt, um Schleifen (v, v) mit $v \in V$ im Graphen zu verbieten. Solange nun der größte Gütezuwachs $A[i, j]$ mit $i, j \in \{1, \ldots, n\}$ noch positiv ist, wird das Paar (i, j) bestimmt, bei dem $A[i, j]$ maximal ist, dem Netz die Kante (v_j, v_i) bzw. der Variable v_i ein Elter v_j hinzugefügt und die Kante (v_j, v_i) durch $A[i, j] \leftarrow -\infty$ als bereits vorhanden markiert. Im Anschluss werden alle Kanten verboten, die, nachdem die Kante (v_j, v_i) eingefügt worden ist, einen Zyklus verursachen würden. Es werden faktisch zwischen jedem Vorfahren v_a von v_i und jedem Nachfahren v_b von v_i und v_i selbst Kanten durch $A[a, b] \leftarrow -\infty$ verboten. Da sich die Eltern π_i von v_i durch das Hinzufügen von v_j geändert haben, aktualisiert B nun lediglich die i-te Zeile von A, indem er den Gütezuwachs für alle $A[i, k]$ mit $k = 1, \ldots, n$ ermittelt, sofern $A[i, k] > -\infty$. Dann beginnt die kopfgesteuerte Schleife von vorn. Der B-Algorithmus terminiert, wenn der maximal mögliche Gütezuwachs nicht mehr positiv ist.

Für einen theoretischen und empirischen Vergleich von K2 und B sei auf [Bou95, S. 105 ff. bzw. S. 114 ff.] verwiesen.

A.3 Exakte probabilistische Inferenz mittels Clustering

A.3.1 Moralisieren eines gerichteten azyklischen Graphen

Zum Bau der Inferenzmaschine M über dem BN $B = (D, P)$ wird zuerst mittels Algorithmus A.4 ein zum gerichteten Graphen D moralischer Graph U erstellt. Im ersten Schritt ruft Algorithmus A.4 den Algorithmus A.3 auf, der den Graphen $U = (N, E)$ erstellt, der dem Graphen $D = (N, E)$ unterliegt.

Algorithmus A.2 B-Algorithmus

Eingabe: die n Variablen V eines BNes, eine Datenbasis D über den Variablen V, ein Gütemaß q

Ausgabe: die Eltern π_v eines jeden Knotens $v \in V$

prozedur $\mathrm{B}(V, D)$
  für $i \leftarrow 1, \ldots, n$ **führe aus**
    $\pi_i = \emptyset$
  ende für
  für $i \leftarrow 1, \ldots, n$ **führe aus**
    für $j \leftarrow 1, \ldots, n$ **führe aus**
      wenn $i = j$ **dann**
        $A[i, j] \leftarrow -\infty$
      sonst
        $A[i, j] \leftarrow q(v_i, \{v_j\}, D) - q(v_i, \emptyset, D)$
      ende wenn
    ende für
  ende für
  solange $\max\limits_{i,j \in \{1,\ldots,n\}} A[i, j] > 0$ **führe aus**
    $(i, j) \leftarrow \operatorname{argmax}\limits_{i,j \in \{1,\ldots,n\}} A[i, j]$
    $\pi_i \leftarrow \pi_i \cup \{v_j\}$
    $A[i, j] \leftarrow -\infty$
    für alle $v_a \in An_i$ **führe aus**
      für $v_b \in De_i \cup \{v_i\}$ **führe aus**
        $A[a, b] \leftarrow -\infty$
      ende für
    ende für
    für $k \leftarrow 1, \ldots, n$ **führe aus**
      wenn $A[i, k] > -\infty$ **dann**
        $A[i, k] \leftarrow q(v_i, \pi_i \cup \{v_k\}, D) - q(v_i, \pi_i, D)$
      ende wenn
    ende für
  ende solange
ende prozedur

Zum Erstellen von U wird der Knotenmenge N des Graphen U die Knotenmenge V des Graphen D zugewiesen, die Kantenmenge E des Graphen U geleert und der Kantenmenge E des Graphen U jede Kante $(u, v) \in A$ und ihre inverse Kante (v, u) hinzugefügt.

Algorithmus A.3 Erstellen eines unterliegenden Graphen

Eingabe: ein Graph $D = (V, A)$
Ausgabe: ein Graph $U = (N, E)$, der dem Graphen D unterliegt

```
1: funktion UnterliegenderGraph(D)
2:     N ← V
3:     E ← ∅
4:     für alle (u, v) ∈ A führe aus
5:         E ← E ∪ {(u, v), (v, u)}
6:     ende für
7:     gib zurück U
8: ende funktion
```

Nachdem der dem Graphen D unterliegende Graph U mittels Algorithmus A.3 erzeugt worden ist, wird der Algorithmus A.4 zum Erstellen eines Graphen U, der sich moralisch zum Graphen D verhält, fortgesetzt: Dazu werden der Kantenmenge E des Graphen U all diejenigen Kanten (pa_1, pa_2) und ihre inversen Kanten (pa_2, pa_1) hinzugefügt, für die gilt, dass pa_1 und pa_2 voneinander verschiedene Eltern desselben Knotens $v \in V$ sind.

Algorithmus A.4 Erstellen eines moralischen Graphen

Eingabe: ein gerichteter Graph $D = (V, A)$
Ausgabe: ein zum Graphen D moralischer Graph $U = (N, E)$

```
1: funktion ErstelleMoralischenGraphen(D)
2:     U ←rufe UnterliegenderGraph(D)
3:     für alle v ∈ V führe aus
4:         für alle pa_1, pa_2 ∈ Pa_v führe aus
5:             wenn pa_1 ≠ pa_2 dann
6:                 E ← E ∪ {(pa_1, pa_2), (pa_2, pa_1)}
7:             ende wenn
8:         ende für
9:     ende für
10:    gib zurück U
11: ende funktion
```

A.3.2 Triangulieren eines moralischen Graphen

Nachdem der zum Graphen G moralische Graph U erzeugt worden ist, muss der Graph U trianguliert werden. Definition A.33 beschreibt einen triangulären Graphen. Das Triangulieren wird als der entscheidende Schritt beim Erstellen der Inferenzmaschine M bezeichnet [DÓ6, S. 13], weil die Güte des Triangulierens den Rechenaufwand für die PI im engeren Sinne entscheidend beeinflusst. Beim Triangulieren wird der Graph U solange mit ungerichteten Kanten aufgefüllt, bis er triangulär ist. Auf welche Art und Weise dieses Auffüllen mit gerichteten Kanten erfolgt, ist jedoch entscheidend für den Speicherplatz, den die Inferenzmaschine M beansprucht, und für die Anzahl der Rechenschritte bei der PI im engeren Sinne. Einen nichttriangulären Graphen optimal zu triangulieren, ist ein NP-vollständiges Problem. [CDLS99, S. 58] Demzufolge wird an dieser Stelle eine Heuristik beschrieben, die gute Resultate zeitigt. [CDLS99, S. 58], [Alm95, S. 170] Die Heuristik, die in Algorithmus A.6 gefasst worden ist, läuft wie folgt ab: Zu Beginn werden sämtliche Knoten $v \in U$ der Menge X der noch nicht behandelten Knoten zugewiesen. Solange noch unbehandelte Knoten vorliegen, wird mittels Algorithmus A.5 der Knoten $x \in X$ bestimmt, welcher als nächster zu behandeln ist: Es wird ein beliebiger Knoten $x \in X$ ausgewählt und zurückgegeben, dessen Auffüllgröße $s = \frac{1}{2} \sum \ln |S_{\{ne_1, ne_2\}}|$ minimal ist, wobei die Summe über alle ne_1 und ne_2 gebildet wird, für die gilt: $(ne_1, ne_2 \in Ne_x \cap X) \wedge (ne_1 \neq ne_2) \wedge ((ne_1, ne_2) \notin E)$.

Nachdem ein beliebiger Knoten $x \in X$ mit der minimalen Auffüllgröße s ausgewählt worden ist, werden alle Nachbarn $ne_1, ne_2 \in Ne_x$ des Knotens x, die noch nicht behandelt und noch nicht durch eine Kante miteinander verbunden worden sind, durch eine ungerichtete Kante miteinander verbunden. Alsdann wird der nunmehr behandelte Knoten x aus der Menge der unbehandelten Knoten X entfernt. Das Triangulieren endet, wenn alle Knoten behandelt worden sind, das heißt, wenn $X = \emptyset$.

A.3.3 Finden und Numerieren maximaler Cliquen im triangulären Graphen

Im Anschluss an das Triangulieren des Graphen U identifiziert der Algorithmus A.8 im nächsten Schritt zum Bau der Inferenzmaschine M die Cliquen des triangulären Graphen U. Da es sich bei allen Cliquen um maximale Cliquen handelt, werden sie von nun an vereinfachend nur noch als Cliquen bezeichnet. Zuerst ruft der Algorithmus A.8 den Algorithmus A.7 auf, der die Knoten des Graphen so numeriert, dass ein beliebiger derjenigen Knoten als nächster numeriert wird, für welche die Mächtigkeit der Menge derjenigen

Algorithmus A.5 Wählen des nächsten Knotens

Eingabe: eine Teilmenge $X \subseteq N$ eines ungerichteten Graphen $U = (N, E)$
Ausgabe: ein Knoten $v \in X$

```
1: funktion WähleNächstenKnoten(X)
2:     s_min ← ∞
3:     für alle x ∈ X führe aus
4:         s ← 0
5:         für alle ne_1, ne_2 ∈ Ne_x ∩ X führe aus
6:             wenn ne_1 ≠ ne_2 ∧ (ne_1, ne_2) ∉ E dann
7:                 s ← s + ln |S_{ne_1,ne_2}|
8:             ende wenn
9:         ende für
10:        s ← s/2
11:        wenn s < s_min dann
12:            v ← x
13:            s_min ← s
14:        ende wenn
15:    ende für
16:    gib zurück v
17: ende funktion
```

Algorithmus A.6 Triangulieren eines ungerichteten Graphen

Eingabe: ein ungerichteter Graph $U = (N, E)$
Ausgabe: der trianguläre Graph U

```
1: prozedur TriangliereGraphen(U)
2:     X ← N
3:     solange X ≠ ∅ führe aus
4:         v ←rufe WähleNächstenKnoten(X)
5:         für alle ne_1, ne_2 ∈ Ne_v ∩ X führe aus
6:             wenn ne_1 ≠ ne_2 dann
7:                 E ← E ∪ {(ne_1, ne_2), (ne_2, ne_1)}
8:             ende wenn
9:         ende für
10:        X ← X ∩ {v}
11:    ende solange
12: ende prozedur
```

Nachbarn maximal ist, die bereits numeriert worden sind. [CDLS99, S. 56]

Algorithmus A.7 Numerieren der Knoten eines triangulären Graphen

Eingabe: ein triangulärer Graph $U = (N, E)$
Ausgabe: die Knoten N, perfekt numeriert

```
 1: prozedur NumeriereKnoten(U)
 2:     X ← N                                  ▷ X Menge der ungeordneten Knoten
 3:     für alle v ∈ X führe aus
 4:         crd(v) ← 0                          ▷ setze Kardinalität auf 0
 5:     ende für
 6:     für i ← 1, ..., |N| führe aus
 7:         v ← ein beliebiges Element in {x ∈ X|crd(x) = max_{y∈X}(crd(y))}
 8:         num(v) ← i
 9:         für alle x ∈ Ne_v ∩ X führe aus
10:             crd(x) ← crd(x) + 1
11:         ende für
12:         X ← X \ {v}
13:     ende für
14: ende prozedur
```

Algorithmus A.7 wird auch als Maximalzahlsuche [BKI03, S. 430] bzw. Maximum Cardinality Search [Gol80, S. 87] bezeichnet. Zu Beginn des Algorithmus A.7 werden sämtliche Knoten $v \in N$ des Graphen der Menge X der noch nicht numerierten Knoten zugewiesen. Im Anschluss daran wird die spezielle Mächtigkeit $crd(v)$ aller Knoten $v \in N$ mit 0 initialisiert. In einer Zählschleife wird im i-ten Durchlauf ein beliebiger Knoten $v \in X$ gewählt, für den die Mächtigkeit der Menge derjenigen seiner Nachbarn maximal ist, die bereits numeriert worden sind. Der Knoten erhält die Nummer $crd(v) = i$, die Kardinalität $crd(x)$ jedes seiner noch nicht numerierten Nachbarn $x \in Ne_v \cap X$ wird inkrementiert, und die Menge der noch nicht numerierten Knoten X wird um den gerade numerierten Knoten v reduziert. Der Algoritmus endet, wenn alle Knoten numeriert worden sind.

Sind die Knoten des triangulären Graphen U numeriert worden, setzt Algorithmus A.8 zum Finden der Cliquen wir folgt fort: Die Menge C der Cliquen wird mit der leeren Menge initialisiert, und die Cliquennummer j wird auf 0 gesetzt. In einer Zählschleife wird im $i-ten$ Durchlauf der Knoten $u \in N$ ausgewählt, für dessen Nummer $num(u) = i$ gilt. Handelt es sich bei u nicht um den zuletzt numerierten Knoten ($num(u) = i < |N|$), wird zusätzlich der nächste Knoten $v \in N$ ausgewählt, für den $num(v) = i + 1$ gilt. Ist $crd(v) < crd(u) + 1$, wird die j-te Clique C_j aus dem Knoten u

Algorithmus A.8 Finden der Cliquen

Eingabe: ein triangulärer Graph $U = (N, E)$ mit numerierten Knoten
Ausgabe: die geordnete Menge C der Cliquen in U

```
1: funktion FindeCliquen(U)
2:     rufe NumeriereKnoten(U)                          ▷ Algorithmus A.7
3:     C ← ∅
4:     j ← 1
5:     für i ← 1, ..., |N| führe aus
6:         u ← ein beliebiges Element in {x ∈ N|num(x) = i}
7:         wenn i < |N| dann
8:             v ← ein beliebiges Element in {x ∈ N|num(x) = i + 1}
9:             wenn crd(v) < crd(u) + 1 dann
10:                C_j ← {u} ∪ Ne_u
11:                num(C_j) ← j
12:                C ← C ∪ {C_j}
13:                j ← j + 1
14:            ende wenn
15:        sonst
16:            C_j ← {u} ∪ Ne_u
17:            num(C_j) ← j
18:            C ← C ∪ {C_j}
19:        ende wenn
20:    ende für
21:    gib zurück C
22: ende funktion
```

und seinen Nachbarn Ne_u gebildet, C_j wird mit $num(C_j) = j$ numeriert, die Clique C_j wird den Cliquen C hinzugefügt, und die Cliquennummer j wird inkrementiert. Handelt es sich bei u hingegen um den zuletzt numerierten Knoten ($num(u) = i = |N|$), wird die Clique C_j aus u und seinen Nachbarn Ne_u gebildet, mit $num(C_j) = j$ numeriert und den Cliquen C hinzugefügt. Algorithmus A.8 endet, wenn alle Knoten $u \in N$ betrachtet und alle Cliquen gefunden worden sind.

Die nach ihrer Numerierung geordneten Cliquen $(C_1, \ldots, C_{|C|})$ weisen die Eigenschaft des fortlaufenden Schnittes [BKI03, S. 434 f.] bzw. die Running Intersection Property (RIP) [CDLS99, s. 56 f.] auf.

Definition A.38 (Eigenschaft des fortlaufenden Schnittes)
Es seien C die Cliquen eines ungerichteten Graphen. Eine Ordnung der Cliquen $(C_1, \ldots, C_{|C|})$, weist die Eigenschaft des fortlaufenden Schnittes auf, wenn es für jedes $i = 2, \ldots, |C|$ ein $j < i$ gibt, so dass gilt:

$$C_i \cap \bigcup_{k=1}^{i-1} C_k \subseteq C_j. \tag{A.1}$$

Informal sagt die Eigenschaft des fortlaufenden Schnittes aus, dass es für jede Clique $c \in C$ mit $num(c) > 1$ mindestens eine Clique $d \in C$ mit $num(d) < num(c)$ gibt, so dass $|c \cap d| > 1$.

A.3.4 Erstellen des Cliquenbaumes

Die Eigenschaft des fortlaufenden Schnittes ist vonnöten, um mittels Algorithmus A.9 den Cliquenbaum zu erstellen: Zu Beginn wird die Kantenmenge S des Cliquenbaumes geleert. Alsdann wird für jede Clique $c \in C$ ein beliebiger Knoten $d \in C$ mit $num(d) < num(c)$ ausgewählt, für den gilt, dass Schnittmenge $d \cap c$ maximale Mächtigkeit aufweist, der Kantenmenge S werden die Kanten (c, d) und (d, c) hinzugefügt, und beiden Kanten wird als Separator die Schnittmenge $c \cap d$ zugewiesen.

Ist der Cliquenbaum T erstellt worden, werden seinen Cliquen C mittels Algorithmus A.10 die Knoten des V des Graphen D zugeordnet: Zu Beginn werden alle den Cliquen $c \in C$ zugeordneten Knoten entfernt. Danach wird jeder Knoten $v \in V$ einer beliebigen derjenigen Cliquen zugeordnet, welche die Familie Fa_v des Knotens enthält und minimale Mächtigkeit aufweist.

A.3.5 Aufspannen und Füllen von Potenzialfunktionen

Im letzten Schritt auf dem Weg zum Erstellen der Inferenzmaschine M werden den Cliquen $c \in C$ des Cliquenbaumes T die Potenzialfunktionen $f(c)$

Algorithmus A.9 Umwandeln eines Cliquengraphen in einen Cliquenbaum

Eingabe: ein Cliquengraph $T = (C, S)$ und eine Ordnung über C, die der Eigenschaft des fortlaufenden Schnittes genügt
Ausgabe: ein Cliquenbaum $T = (C, S)$

prozedur WandleInCliquenBaum(T)
    $S \leftarrow \emptyset$
    für alle $c \in C$ **führe aus**
        $d \leftarrow$ eine beliebiges Element in
$\{x \in C | (num(x) < num(c)) \wedge (|x \cap c| = max_{y \in C} |y \cap c|)\}$
        $S \leftarrow S \cup \{(c, d), (d, c)\}$
        $sep[(c, d)] \leftarrow sep[(d, c)] \leftarrow c \cap d$
    ende für
ende prozedur

Algorithmus A.10 Assoziieren der Knoten des BNes zu den Cliquen

Eingabe: eine Menge von Knoten V und eine Menge von Cliquen C
Ausgabe: die Cliquen C mit assoziierten Knoten $v \in V$

prozedur AssoziiereKnotenMitCliquen(V, C)
    für alle $c \in C$ **führe aus**
        $Nd(c) \leftarrow \emptyset$
    ende für
    für alle $v \in V$ **führe aus**
        $c \leftarrow$ eine beliebige Clique in
$\{d \in C | (Fa_v \subseteq d) \wedge (|d| = \min_{b \in C} |b|)\}$
        $Nd(c) \leftarrow Nd(c) \cup \{v\}$
    ende für
ende prozedur

zugewiesen: Für jede Clique $c \in C$ werden sämtliche Funktionswerte der Potenzialfunktion$f(c)$ mit 1 intialisiert. Im Anschluss wird die Potenzialfunktion mit den Bewertungsfunktionen $p(nd)$ aller Knoten $nd \in Nd(c)$ multipliziert.

Algorithmus A.11 Füllen der Potenzialfunktionen

Eingabe: eine Inferenzmaschine $M = (T, F)$ mit $T = (C, S)$
Ausgabe: die Inferenzmaschine M mit passenden Potenzialfunktionen F

```
1: prozedur FüllePotenzialfunktionen(T)
2:     für alle c ∈ C führe aus
3:         für alle s_c ∈ S_c führe aus
4:             f(c = s_c) ← 1
5:         ende für
6:         f(c) ← f(c) ∏_{nd∈Nd(c)} p(nd)
7:     ende für
8: ende prozedur
```

A.3.6 Zusammenfassung des Erstellens einer Inferenzmaschine

Algorithmus A.12 fasst die Schritte zum Erstellen der Inferenzmaschine M noch einmal zusammen.

Algorithmus A.12 Erstellen der Inferenzmaschine

Eingabe: ein BN $B = (G, P)$ mit $D = (V, A)$
Ausgabe: eine Inferenzmaschine $M = (T, F)$ mit $T = (C, S)$

```
1: funktion ErstelleInferenzMaschine(D)
2:     U ←rufe ErstelleMoralischenGraphen(D)
3:     rufe TriangliereGraphen(U)
4:     rufe NumeriereKnoten(U)
5:     C ←rufe FindeCliquen(U)
6:     rufe WandleInCliquenBaum(T)
7:     rufe AssoziiereKnotenMitCliquen(V, C)
8:     rufe FüllePotenzialfunktionen(T)
9:     gib zurück M
10: ende funktion
```

A.3.7 Absorbieren von Evidenz

Nachdem die Inferenzmaschine erstellt worden ist, beginnt die PI im engeren Sinne: Zu Beginn wird mittels Algorithmus A.13 die für die evidenten Variablen $E \subseteq V$ vorliegende Evidenz $e \in S_E$ von den Potenzialfunktionen $f \in F$ absorbiert. Zu diesem Zweck werden all diejenigen Funktionswerte der Potenzialfunktionen $f \in F$ auf 0 gesetzt, deren Argumente nicht mit e übereinstimmen. Formal lautet die Absorption von Evidenz wie folgt:

Definition A.39 (Absorption von Evidenz)
Es sei I eine V darstellende Indexmenge. Es sei $J \subseteq I$ eine $X \subseteq V$ darstellende Indexmenge. Es sei $K \subseteq I$ eine Indexmenge, welche die evidenten Variablen $E \subseteq V$ darstellt. Es seien S_X, S_E und $S_{X \cup E}$, die Zustandsräume von X, E bzw. $X \cup E$. Es sei f eine Potenzialfunktion von X, und $e \in S_E$ sei die Evidenz. Es sei π_X die Projektion von $S_{X \cup E}$ auf S_X, und π_E sei die Projektion von $S_{X \cup E}$ auf S_E. Die Absorption $f \triangleleft e$ der Evidenz e durch die Potenzialfunktion f ist eine Potenzialfunktion von X und definiert als

$$(f \triangleleft e)[\pi_X(s)] = \begin{cases} f[\pi_X(s)] | \pi_E(s) = e \\ 0 \quad\quad | \pi_E(s) \neq e \end{cases}$$

für alle Konfigurationen $s \in S_{X \cup E}$.

Algorithmus A.13 Absorbieren der Evidenz

Eingabe: die Evidenz $e \in S_E$ mit $E \subseteq V$, die Potenzialfunktionen F von M

Ausgabe: die Potenzialfunktionen F mit absorbierter Evidenz e

---

```
1: prozedur AbsorbiereEvidenz(e, F)
2:     für alle f ∈ F führe aus
3:         f ← f ◁ e
4:     ende für
5: ende prozedur
```

A.3.8 Austauschen von Nachrichten zwischen Potenzialfunktionen

Im Anschluss an das Absorbieren der Evidenz werden die Nachrichten zwischen den Potenzialfunktionen $f(c)$ und $f(d)$ mit $c, d \in C$ und $\{(c,d),(d,c)\} \subseteq S$ ausgetauscht. Zu diesem Zwecke ruft Algorithmus A.16 nacheinander die Algorithmen A.14 und A.15 mit einer beliebigen Clique $c \in C$ auf.

Algorithmus A.14 Sammeln der Evidenz

Eingabe: die Cliquen c und d

Ausgabe: die Potenzialfunktion $f(c)$, multipliziert mit den Nachrichten $msg[(nb(c), c)]$ mit $nb(c) \in Nb(c) \setminus \{d\}$,
die aus $f(c)$ herausintegrierte Nachricht $msg[(c, d)]$

```
1:  prozedur SammleEvidenz(c, d)
2:      für alle nb(c) ∈ Nb(c) \ {d} führe aus
3:          rufe SammleEvidenz(nb(c), c)
4:          f(c) ← f(c) · msg[(nb(c), c)]
5:      ende für
6:      wenn d ≠ NULL dann
7:          wenn Propagation = Summe dann
8:              msg[(c, d)] ← Σ_{c\d} f(c)
9:          sonst
10:             msg[(c, d)] ← max_{c\d} f(c)
11:         ende wenn
12:     ende wenn
13: ende prozedur
```

Zum Sammeln der Evidenz ruft Algorithmus A.14 sich für alle Nachbarn $nb(c) \in Nb(c)\setminus\{d\}$ außer dem Nachbarn d, vom dem er aufgerufen worden ist, selbst auf und multipliziert die empfangenen bzw. gesammelten Nachrichten $msg[(nb(c), c)]$ mit der Potenzialfunktion $f(c)$. Ist die Clique d vorhanden, mit welcher der Algorithmus A.14 aufgerufen worden ist, wird die Nachricht $msg[(c, d)]$ aus der Potenzialfunktion $f(x)$ herausintegriert (und steht der Clique d zur Multiplikation mit ihr zur Verfügung). Abhängig davon, ob die Summe oder das Maximum propagiert werden soll, wird die Nachricht $msg[(d, nb(d))]$ durch das Herausintegrieren des Summenrandes oder des Maximumrandes gewonnen. Der Algorithmus A.14 terminiert, wenn kein Nachbar $nb(c) \in Nb(c) \setminus \{d\} = \emptyset$ mehr vorhanden ist (außer dem Nachbarn c, mit dem der Algorithmus aufgerufen worden ist).

Algorithmus A.15 verteilt Evidenz. Sofern die Clique c vorhanden ist, wird die Potenzialfunktion $f(d)$ mit der Nachricht $msg[(c, d)]$ multipliziert. Wird nicht die Summe, sondern das Maximum propagiert, wird ein beliebiges Argument s_c der Potenzialfunktion $f(d)$ ausgewählt, dessen Funktionswert maximal ist, alle Funktionswerte der Potenzialfunktion $f(d)$ auf 0 und der Funktionswert $f(c = s_c)$ auf 1 gesetzt. Dann wird an alle Nachbarn $nb(d)$ der Clique d, außer an die Clique c, von der die Nachricht $msg[(c, d)]$ gekommen ist, eine Nachricht $msg[(d, nb(d))]$ gesendet, die vorab aus der Potenzialfunk-

Algorithmus A.15 Verteilen der Evidenz

Eingabe: die Cliquen c und d

Ausgabe: die Potenzialfunktion $f(c)$, multipliziert mit der Nachricht $msg[(c,d)]$, die aus $f(c)$ herausintegrierten Nachrichten $msg[(d, nb(d))]$ mit $nb(d) \in Nb(d) \setminus \{c\}$

1: **prozedur** VERTEILEEVIDENZ(c, d)
2: **wenn** $c \neq$ NULL **dann**
3: $f(d) \leftarrow f(d) \cdot msg[(c,d)]$
4: **ende wenn**
5: **wenn** $Propagation \neq Summe$ **dann**
6: $s_c \leftarrow$ ein beliebiges Element in $\{x \in S_c | f(x) = \max_{y \in S_c} f(y)\}$
7: $f(c) \leftarrow f(c) \cdot 0$
8: $f(c = s_c) \leftarrow 1$
9: **ende wenn**
10: **für alle** $nb(d) \in Nb(d) \setminus \{c\}$ **führe aus**
11: **wenn** $Propagation = Summe$ **dann**
12: $msg[(d, nb(d))] \leftarrow \dfrac{\sum_{d \setminus nb(d)} f(d)}{msg[(nb(d), d)]}$
13: **sonst**
14: $msg[(d, nb(d))] \leftarrow \dfrac{\max_{d \setminus nb(d)} f(d)}{msg[(nb(d), d)]}$
15: **ende wenn**
16: **rufe** VERTEILEEVIDENZ$(d, nb(d))$
17: **ende für**
18: **ende prozedur**

tion $f(d)$ herausintegriert und durch die ihr inverse Nachricht $msg[(nb(d), d)]$ dividiert worden ist. Abhängig davon, ob die Summe oder das Maximum propagiert werden soll, wird die Nachricht $msg[(d, nb(d))]$ durch das Herausintegrieren der Summe oder des Maximums gewonnen. Der Algorithmus A.15 terminiert, wenn kein Nachbar $nb(d) \in Nb(d) \setminus \{c\} = \emptyset$ mehr vorhanden ist (außer dem Nachbarn c, mit dem der Algorithmus aufgerufen worden ist).

Algorithmus A.16 Austauschen der Nachrichten

Eingabe: eine Inferenzmaschine $M = (T, F)$ mit $T = (C, S)$
Ausgabe: die Inferenzmaschine M im Gleichgewicht

1: **prozedur** TauscheNachrichtenAus(M)
2: $c \leftarrow$ ein beliebiges Element in C
3: **rufe** SammleEvidenz(c, NULL)
4: **rufe** VerteileEvidenz(NULL, c)
5: **ende prozedur**

Nachdem die Algorithmen A.14 und A.15 zum Sammeln und zum Verteilen der Evidenz aufgerufen und abgearbeitet und die Potenzialfunktionen mittels Algorithmus A.17 auf 1 normalisiert worden sind, befindet sich die Inferenzmaschine M im Gleichgewicht.

Algorithmus A.17 Normalisieren der Potenzialfunktionen

Eingabe: eine Inferenzmaschine $M = (T, F)$ mit $T = (C, S)$
Ausgabe: eine Inferenzmaschine M mit $\sum_{c \in C} f(c) = 1$

1: **prozedur** NormalisierePotenzialfunktionen(M)
2: **für alle** $c \in C$ **führe aus**
3: $$f(c) \leftarrow \frac{f(c)}{\sum_{s_c \in S_c} f(c = s_c)}$$
4: **ende für**
5: **ende prozedur**

A.3.9 Herausintegrieren von Randverteilungen, Bestimmen einer wahrscheinlichsten Konfiguration

Nun kann damit begonnen werden, die Randverteilungen $p(v)$ mittels Algorithmus A.18 aus den Potenzialfunktionen $f(c)$ herauszuintegrieren. Um den Rechenaufwand gering zu halten, wird zu diesem Zwecke für jede Variable $v \in V$ eine beliebige Clique c ausgewählt, die v enthält und minimale

Algorithmus A.18 Herausintegrieren

Eingabe: eine Inferenzmaschine $M = (T, F)$ mit $T = (C, S)$,
ein BN $B = (G, P)$ mit $G = (V, A)$
Ausgabe: die Variablen $v \in V$ des BNes Randverteilungen $p(v)$

```
1: prozedur IntegriereRandVerteilungenHeraus(M, B)
2:     für alle v ∈ V führe aus
3:         c ← ein beliebiges Element in {x ∈ C|(|x|) = min_{y∈C}(|y|)}
4:         für alle s_v ∈ S_v führe aus
5:             p(v = s_v) = Σ_{s_c ∈ S_c \ {v}} f(v = s_v, c \ {v} = s_c)
6:         ende für
7:     ende für
8: ende prozedur
```

Mächtigkeit aufweist. Aus dieser Clique c wird dann die jeweilige Randverteilungsfunktion $\Pr(v)$ herausintegriert. Ist das Maximum propagiert worden, kann anstelle des Algorithmus' A.18 der Algorithmus A.19 verwandt werden, der die wahrscheinlichste Konfiguration $cfg(v)$ aller Variablen $v \in V$ und somit die wahrscheinlichste Konfiguration $cfg(V)$ bestimmt. Eine Erweiterung des Algorithmus' A.19 ermöglicht sogar, die k wahrscheinlichsten Konfigurationen, geordnet nach ihrer jeweiligen Wahrscheinlichkeit, zu bestimmen. [CDLS99, S. 101 ff.] Die Randverteilungen $\Pr(v)$ und die wahrscheinlichste Konfiguration $cfg(V)$ verstehen sich immer als Verteilungen respektive Konfiguration unter der vorliegenden Evidenz.

Algorithmus A.19 Bestimmen der wahrscheinlichsten Konfiguration

Eingabe: eine Inferenzmaschine $M = (T, F)$ mit $T = (C, S)$,
ein BN $B = (G, P)$ mit $G = (V, A)$
Ausgabe: die Variablen $v \in V$ des BNmit Konfigurationen $cfg(v)$

```
1: prozedur BestimmeWahrscheinlichsteKonfiguration(M, B)
2:     für alle v ∈ V führe aus
3:         c ← ein beliebiges Element in {x ∈ C|(|x|) = min_{y∈C}(|y|)}
4:         für alle s_v ∈ S_v führe aus
5:             p(v = s_v) = max_{s_c ∈ S_c \ {v}} f(v = s_v, c \ {v} = s_c)
6:         ende für
7:     ende für
8: ende prozedur
```

A.3.10 Zusammenfassung der PI

Algorithmus A.20 fasst die PI im engeren Sinne noch einmal zusammen.

Algorithmus A.20 Probabilistische Inferenz im engeren Sinne

Eingabe: eine Inferenzmaschine $M = (T, F)$ mit $T = (C, S)$,
ein BN $B = (G, P)$ mit $G = (V, A)$, die Evidenz $e \in S_E$ mit $E \subseteq V$
Ausgabe: die Randverteilungen $Pr(v)$ mit $v \in V$,
Konfigurationen $cfg(v)$

```
1: prozedur PropagiereProbabilistisch(M, B)
2:     rufe AbsorbiereEvidenz(e, F)
3:     rufe TauscheNachrichtenAus(M)
4:     rufe NormalisierePotenzialfunktionen(M)
5:     wenn Propagation = Summe dann
6:         rufe IntegriereRandVerteilungenHeraus(M)
7:     sonst
8:         rufe BestimmeWahrscheinlichsteKonfiguration(M)
9:     ende wenn
10: ende prozedur
```

Algorithmus A.21 beschreibt die gesamte PI noch einmal als den Bau der Inferenzmaschine M und die PI im engeren Sinne. Wenn keine PI im engeren Sinne mehr vollzogen werden soll, bricht Algorithmus $A.21$ ab.

Der Aufbau der Inferenzmaschine M erfolgt einmalig und entspricht somit fixen Kosten. Die PI i. e. S. erfolgt mehrfach über derselben Inferenzmaschine M und verursacht demzufolge variable Kosten. Eine bessere Inferenzmaschine zu erstellen, führt zu höheren Fixkosten, die sich aber ab einer bestimmten Anzahl von PIen i. e. S. amortisieren.

Algorithmus A.21 Probabilistische Inferenz

Eingabe: ein BN $B = (D, P)$ mit $D = (V, A)$
Ausgabe: die Randverteilungen $p(v)$ mit $v \in V$
oder die wahrscheinlichste Konfiguration $s_V \in S_V$

```
1: prozedur InferiereProbabilistisch(B, R)
2:     M ←ErstelleInferenzMaschineB
3:     wiederhole
4:         rufe PropagiereProbabilistisch(M, B)
5:     bis Abbruch
6: ende prozedur
```

A.4 Stochastische Simulation: Beispiel und Algorithmus

Das folgende Beispiel zur stochastischen Simulation basiert auf dem Beispiel „Marie und ihre Rosen“ aus Abschnitt 4.1.1. Die stochastische Simulation erfolgt über dem BN. Im Prinzip werden mehrere Konfigurationen von V über dem BN generiert bzw. Stichproben gezogen. Zu Beginn werden die Bewertungsfunktionen der Variablen ausgewählt, die keine Eltern aufweisen. Im Beispiel betrifft das die Variablen *re* und *sp*. Aufgrund ihrer Bewertungsfunktionen werden Zustände für sie festgelegt. Für *re* wird eine gleichverteilte Zufallszahl im Intervall $[0;1)$ generiert. Ist sie kleiner als 0,8, erhält die Variable *re* den Zustand ne_{re}, sonst den Zustand ja_{re}. Für *sp* wird ebenfalls eine Zufallszahl im Intervall $[0;1)$ generiert. Ist sie kleiner als 0,95, erhält die Variable *sp* den Zustand au_{sp}, sonst den Zustand ei_{sp}. Nun werden die Variablen ausgewählt, deren Eltern bereits alle mit einem Zustand versehen worden sind. Im Beispiel betrifft das die Variablen *st* und *bo*. Aufgrund der für ihre Eltern festgelegten Zustände werden aus den Bewertungsfunktionen der Variablen *st* und *bo* bedingte Wahrscheinlichkeitsfunktionen ermittelt. Angenommen, für die Variablen *re* und *sp* seien die Zustände ne_{re} respektive au_{sp} ermittelt worden, dann ergeben sich die bedingten Wahrscheinlichkeitsfunktionen $\Pr(st|re = ne_{re}) = \{0{,}9; 0{,}1\}$ und $\Pr(bo|re = ne_{re}, sp = au_{sp}) = \{0{,}85; 0{,}15\}$. Nun generiert man für jede dieser Wahrscheinlichkeitsfunktionen wiederum eine Zufallszahl und bestimmt adäquat zur obigen Vorgehensweise jeweils einen Zustand für die Variablen *st* und *bo*. Angenommen, die Zustände tr_{st} und tr_{bo} seien ermittelt worden. Nun werden die nächsten Variablen ausgewählt, für die noch kein Zustand bestimmt worden ist und deren Eltern bereits alle mit einem Zustand versehen worden sind. Lediglich die Variable *ro* wird als letzte gewählt. Für sie wird die Wahrscheinlichkeitsfunktionen $\Pr(ro|bo = tr_{bo}) = \{0{,}75; 0{,}25\}$ bestimmt und mittels Zufallszahl der Zustand sc_{ro} ermittelt. Sind für sämtliche Variablen V Zustände ermittelt worden, wird die Konfiguration $S_V = (ne_{re}, au_{sp}, tr_{st}, tr_{bo}, sc_{ro})$ gespeichert, und eine weitere Konfiguration wird generiert. Der Algorithmus terminiert, sofern ein bestimmte Anzahl von Konfigurationen generiert oder eine bestimmte Genauigkeit erreicht worden ist.

Im Anschluss werden die Randwahrscheinlichkeitsfunktionen für die Variablen V ermittelt. Angenommen, 1.000 Konfigurationen sind generiert worden, und die Randwahrscheinlichkeitsfunktion für die Variable *ro* werde bestimmt. In den Konfigurationen trete 517 mal der Zustand sc_{ro} und 483 mal der Zustand gu_{ro} auf. Dann lautet die Randwahrscheinlichkeitsfunktion für die Variable *ro* $\Pr(ro) = \{517/1.000, 483/1.000\}$. Für alle anderen Variablen

$V \setminus \{ro\}$ werden die Randwahrscheinlichkeitsfunktionen auf adäquate Art und Weise bestimmt.

Algorithmus A.22 zeigt den etwas gestrafften Pseudokode für die stochastische Vorwärtssimulation. Liegt Evidenz vor bzw. sind bestimmte Variablen instanziiert worden, werden diejenigen Konfigurationen verworfen, die nicht mit der Evidenz übereinstimmen. Das führt zu Problemen, wenn wenig wahrscheinliche Evidenz vorliegt, weil dann nahezu alle Konfigurationen verworfen werden müssen. Angenommen, die Straße sei nass, und den Rosen gehe es (trotzdem) schlecht. Die gemeinsame Wahrscheinlichkeit dafür beträgt $\Pr(st = na_{st}, ro = sc_{ro}) \approx 0{,}0725$. Wenn man nun 1000 gültige Konfigurationen ermitteln wollte, müsste man $1.000/0{,}0725 \approx 14.000$ Stichproben ziehen, von denen man zirka 13.000 verwerfen würde. Diesem Problem begegnet ein Algorithmus namens Gibbs Sampling[4], auf welchen hier lediglich verwiesen wird.

[4]Der Algorithmus ist von [GG84] entworfen und nach Josiah Willard Gibbs (1839-1903), einem Vertreter der statistischen Physik, benannt worden – wegen einer Analogie zwischen dem Algorithmus und einem Algorithmus der statistischen Physik. [GG90, S. 453]

Algorithmus A.22 Stochastische Vorwärtssimulation

Eingabe: ein BN $B = (D, F)$ mit $D = (V, A)$,
eine Anzahl c von Simulationsläufen
Ausgabe: die Randwahrscheinlichkeitsfunktionen $\Pr(v)$ mit $v \in V$

prozedur SimuliereStochastischVorwärts(B)
  für alle $v \in V$ **führe aus**
    $R_v \leftarrow \emptyset$ ▷ Stichprobe Variable v
  ende für
  für $i \leftarrow 1 \dots c$ **führe aus**
    $X \leftarrow V$ ▷ noch nicht instanziiert
    $Y \leftarrow \{x \in X | (|Pa_x| = 0)\}$ ▷ als nächste zu instanziieren
    $E \leftarrow \emptyset$ ▷ instanziiert
    solange $|X| > 0$ **führe aus**
      für alle $v \in Y$ **führe aus**
        Pr ← Verteilungsfunktion von v, gewonnen aus Bewertungsfunktion p_v unter Bedingung, dass für alle $pa \in Pa_v$ gilt: $pa = e_{pa}$
        $s \leftarrow \Pr^{-1}[\text{rand}(\quad [0;1) \quad)]$ ▷ Ziehe Zustand anhand inverser Verteilungsfunktion
        $e_v \leftarrow s$ ▷ instanziiert
        $R_v \leftarrow R_v \cup \{(i, s)\}$
      ende für
      $X \leftarrow X \setminus Y$
      $E \leftarrow E \cup Y$
      $Y \leftarrow \{x \in X | Pa_x \subseteq E\}$
    ende solange
  ende für
  für alle $v \in V$ **führe aus**
    für alle $s \in S_v$ **führe aus**
      $\Pr(v = s) \leftarrow \dfrac{|\{(\cdot, r) \in R_v | r = s\}|}{c}$
    ende für
  ende für
ende prozedur

Anhang B

Details zum Softwaresystem zur Unterstützung der PPS mittels BNe

B.1 Komponente zur Repräsentation BNe

B.1.1 Graphen, Potenzial- und Bewertungsfunktionen

Abbildung B.1 zeigt das UML-Klassendiagramm der Graphen und der speziellen n-dimensionalen, diskreten, reellwertigen Funktionen[1], aus denen sich BNe und ihre Inferenzmaschinen zusammensetzen.

Drei parallele Vererbungshierarchien von Graphen, ihren Knoten und ihren Kanten stehen nebeneinander. Von allgemeinen Graphen erben ungerichtete und gerichtete Graphen, von denen erben Cliquengraphen und (gerichtete) azyklische Graphen, und von denen erben wiederum Inferenzmaschinen respektive BNe. Jeder Graph hat Knoten und Kanten, und alle Typen von Graphen weisen spezielle Knoten und Kanten auf. Die Abkürzungen GG, UG, AG, CG und IM in den Klassennamen der Knoten und Kanten stehen für gerichteter Graph, ungerichteter Graph, (gerichteter) azyklischer Graph, Cliquengraph und Inferenzmaschine. Die Abkürzungen sagen aus, dass der mit ihnen bezeichnete Knoten- oder Kantentyp zu einem entsprechenden Graphentyp gehört. Dass ein Knoten oder eine Kante immer nur einem passenden Graphen angehört, stellen eine Kombination von Klassenreferenzvariablen[2] und Fabrikmethoden [GHJV04, S. 131 ff.] des jeweiligen

[1] siehe Anhang A.1.1

[2] Klassenreferenzvariablen sind eine nützliche Besonderheit der Programmiersprache Object Pascal.

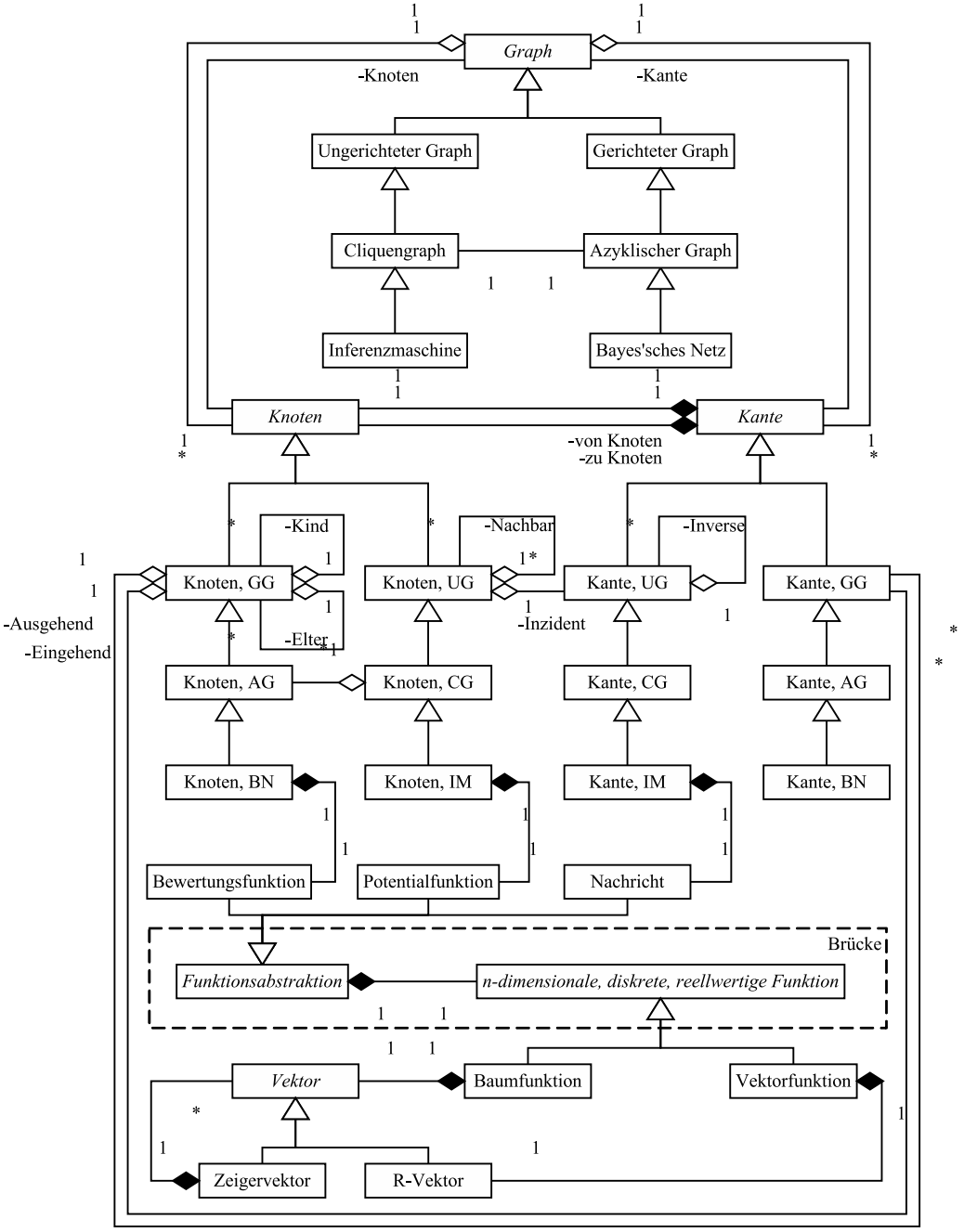

Abbildung B.1: UML-Klassendiagramm spezieller Graphen, spezieller n-dimensionaler, diskreter, reellwertiger Funktionen und des Zusammenhanges zwischen den Graphen und den Funktionen

Graphen zum Erzeugen seiner Knoten und Kanten sicher.

Eine Kante kennt immer (ihre) zwei Knoten. Jeder Knoten und jede Kante kennt seinen bzw. ihren Graphen. Knoten gerichteter Graphen weisen immer Eltern und Kinder sowie eingehende und ausgehende Kanten auf. Knoten ungerichteter Graphen verfügen über Nachbarn (Nachbarknoten) und inzidente Kanten. Eine Kante eines ungerichteten Graphen hält immer die zu ihr inverse Kante, die obligatorisch ist. Ein Cliquengraph kennt den gerichteten, azyklischen Graphen, aus dem er hervorgeht. Ein Knoten eines Cliquengraphen enthält die Knoten eines gerichteten, azyklischen Graphen, auf denen er beruht.

Ein Knoten eines BNes enthält seine Bewertungsfunktion. Ein Knoten einer Inferenzmaschine enthält seine Potenzialfunktion. Eine Kante eines Cliquengraphen enthält eine Potenzialfunktion, welche die Nachricht verkörpert, die entlang der Kante „fließt“ (siehe Algorithmus A.16).

Bewertungsfunktion, Potenzialfunktion und Nachricht erben von einer abstrakten „Funktionsabstraktion“ [GHJV04, S. 189], welche eine wiederum abstrakte n-dimensionale, diskrete, reellwertige Funktion kapselt. Diese Funktion tritt in zwei konkreten Ausprägungen auf: als „Baumfunktion“ oder als „Vektorfunktion“. [GHJV04, S. 186 ff.] bezeichnet die gesamte Konstruktion als „Brücke“. Den „Client“ der Funktionsabstraktion verkörpert der Algorithmus, welcher die Potenzialfunktionen, die Bewertungsfunktionen und die Nachrichten als Funktionsabstraktionen im Rahmen der PI miteinander verrechnet. Abbildung B.1 stellt diesen Client-Algorithmus nicht mit dar. Der Vorteil der Brücke besteht im hier vorliegenden Falle darin, dass die Implementierungen, die Vektor- und die Baumfunktion, je nach Anwendungsfall ausgetauscht werden können, ohne dass es ihre spezialisierten Abstraktionen, die Potenzialfunktion, die Bewertungsfunktion und die Nachricht, berührt.

Das Klassendiagramm in Abbildung B.1 blendet die Attribute und Methoden der Klassen zum Wohle der Übersichtlichkeit und der Kompaktheit aus. Die Attribute der Klassen entsprechen im Wesentlichen den Assoziationen, Aggregationen und Kompositionen aus dem Klassendiagramm. Allgemeine Knoten tragen zudem Namen, und Knoten BNe, die ja Variablen verkörpern, weisen zusätzlich diskrete Zustände auf. Die Graphen verfügen über Methoden zum Anlegen, Hinzufügen und Löschen von Knoten und Kanten. Die Knoten erlauben, ihnen – abhängig vom Typ – vorhandene Knoten als Eltern, Kindern und Nachbarn sowie eingehende, ausgehende und inzidente Kanten hinzuzufügen. Die Knoten erlauben zudem, letztere sechs wieder aus ihnen zu entfernen. Implizit führen die Knoten all diese Aktionen auf ein Hinzufügen/Entfernen von Kanten zu/aus ihrem Graphen zurück. Dass ein gerichteter, azyklischer Graph auch tatsächlich azyklisch ist und vor allem bleibt, stellt eine Methode sicher, die zu Beginn des expliziten oder implizi-

ten Hinzufügens einer Kante zum gerichteten, azyklischen Graphen aufgerufen wird. Die Methode prüft rekursiv, ob die Kante, hinzugefügt, mindestens einen Zyklus verursachen würde, und verhindert das Hinzufügen der Kante gegebenenfalls.

Die Methoden der im Klassendiagramm in Abbildung B.1 dargestellten speziellen Funktionen entsprechen im Wesentlichen den in Abschnitt A.1.1 definierten. Wie das Multiplizieren und Dividieren von Potenzialfunktionen sowie das Herausintegrieren von Summen- und Maximumrändern aus ihnen im Softwaresystem effizient vonstatten geht, beschreibt Abschnitt 6.4.3 en detail.

B.1.2 Modifikationen von Bewertungsfunktionen

Folgenden Operationen über den Variablen und dem Graphen eines BNes verursachen eine gleichzeitige Modifikation von Bewertungsfunktionen des BNes. Alle anderen Operationen, die eine Modifikation von Bewertungsfunktionen nach sich ziehen, – betreffen sie das Entfernen von Knoten oder das Invertieren von Kanten –, lassen sich auf die nachfolgenden fünf Operationen zurückführen. Es wird insbesondere dargelegt, wie die Operationen Bewertungsfunktionen modifizieren, die sie betreffen.

1. Das *Hinzufügen einer Kante*[3] zu einem BN erstellt für die Variable auf der Kindseite der Kante eine neue Bewertungsfunktion, die all die bedingten Wahrscheinlichkeiten enthält, die für alle verschiedenen Konfigurationen der bisherigen beeinflussenden Variablen, der neuen beeinflussenden Variable und der beeinflussten Variable vonnöten sind. Jede neue bedingte Wahrscheinlichkeit resultiert aus der bedingten Wahrscheinlichkeit der bisherigen Bewertungsfunktion, deren zugehörige Zustände mit den Zuständen der neuen bedingten Wahrscheinlichkeit übereinstimmen, wobei die Zustände der neuen bedingten Wahrscheinlichkeit einen Zustand mehr aufweisen, einen Zustand der Variable auf der Elternseite der Kante. Die neue Bewertungsfunktion ersetzt die bisherige.

2. Das *Löschen einer Kante*[4] aus einem BN erstellt für den Kindknoten bzw. für die Variable auf der Kindseite der Kante eine neue Bewertungsfunktion, die nur noch die bedingten Wahrscheinlichkeiten enthält, die für alle verschiedenen Konfigurationen der verbleibenden beeinflussenden Variablen und der beeinflussten Variable vonnöten sind. Jede neue

[3] bzw. das Hinzufügen eines Elters zu einem Knoten

[4] bzw. das Entfernen eines Elters eines Knotens

bedingte Wahrscheinlichkeit resultiert aus der Summe der bedingten Wahrscheinlichkeiten der bisherigen Bewertungsfunktion, deren zugehörige Zustände bis auf den Zustand der Variable auf der Elternseite der Kante übereinstimmen, geteilt durch die Anzahl der Zustände der Variable auf der Elternseite der Kante. Die neue Bewertungsfunktion ersetzt die bisherige.

3. Das *Einfügen eines Zustandes* in eine Variable fügt in die Bewertungsfunktion der Variable neue bedingte Wahrscheinlichkeiten ein und initialisiert sie mit 0. Die bereits vorhandenen bedingten Wahrscheinlichkeiten ändern ihren Wert nicht. Zudem werden in die Bewertungsfunktionen der Kinder der Variable neue bedingte Wahrscheinlichkeiten bzw. Verteilungen eingefügt und mit gleichverteilten Werten initialisiert. Die bereits vorhandenen bedingten Wahrscheinlichkeiten bzw. Verteilungen ändern ihre Werte nicht.

4. Das *Entfernen eines Zustandes* aus einer Variable entfernt aus der Bewertungsfunktion der Variable die betroffenen bedingten Wahrscheinlichkeiten und normalisiert die verbleibenden Werte zu bedingten Wahrscheinlichkeiten. In den Bewertungsfunktionen der Kinder der Variable werden die betroffenen bedingten Wahrscheinlichkeiten bzw. Verteilungen lediglich entfernt. Die verbleibenden bedingten Wahrscheinlichkeiten bzw. Verteilungen ändern ihre Werte nicht.

Die Tabellen B.1 und B.2 verdeutlichen noch einmal anhand zweier Beispiele, welche Modifikation die Bewertungsfunktion eines Knotens erfährt, wenn aus seinem BN eine Kante entfernt oder seinem BN eine Kante hinzugefügt wird, die ihn als Kindknoten hat. Die zwei Beispiele verdeutlichen zudem, dass es sich beim Entfernen und Hinzufügen von Kanten bzw. Elternknoten um irreversible Operationen handelt.

Die Methoden zum Modifizieren BNe sind „geradeaus“ implementiert worden, da das Softwaresystem sie selten wiederholt automatisch aufruft, sondern der Anwender sie einzeln ausführt und die Methoden einzeln weder rechenzeit- noch hauptspeicherkritisch sind.

B.2 Komponente zur Akquisition BNe

B.2.1 Datenstrukturen für bedingte Häufigkeiten

Jeder Knoten eines BNes erhält eine *Häufigkeitenstruktur* zum Akkumulieren und zum Zugriff auf bedingte Häufigkeiten. In einer solchen Häufigkei-

Tabelle B.1: Modifikation der Bewertungsfunktion des Knotens *bo* [siehe Tabelle 4.2 a)] durch das Entfernen der Kante $sp \longrightarrow bo$ aus dem BN „Marie und ihre Rosen“ bzw. durch das Entfernen seines Elters *sp*: a) die ursprüngliche und b) die neue Bewertungsfunktion des Knotens *bo*

a)

re	sp	bo	$p(bo\|re, sp)$
ne_{re}	au_{sp}	tr_{bo}	0,85
ne_{re}	au_{sp}	fe_{bo}	0,15
ne_{re}	ei_{sp}	tr_{bo}	0,05
ne_{re}	ei_{sp}	fe_{bo}	0,95
ja_{re}	au_{sp}	tr_{bo}	0,10
ja_{re}	au_{sp}	fe_{bo}	0,90
ja_{re}	ei_{sp}	tr_{bo}	0,01
ja_{re}	ei_{sp}	fe_{bo}	0,99

b)

re	bo	$p(bo\|re)$
ne_{re}	tr_{bo}	$\frac{0,85+0,05}{2} = 0,450$
ne_{re}	fe_{bo}	$\frac{0,15+0,95}{2} = 0,550$
ja_{re}	tr_{bo}	$\frac{0,10+0,01}{2} = 0,055$
ja_{re}	fe_{bo}	$\frac{0,90+0,99}{2} = 0,945$

Tabelle B.2: Modifikation der Bewertungsfunktion des Knotens *bo* [siehe Tabelle B.1 b)] durch das Wiederhinzufügen der Kante $sp \longrightarrow bo$ zu dem BN „Marie und ihre Rosen“ bzw. durch das Wiederhinzufügen des Knotens *sp* als Elter zum Knoten *bo*: a) die ursprüngliche und b) die neue Bewertungsfunktion des Knotens *bo*

a)

re	bo	$p(bo\|re)$
ne_{re}	tr_{bo}	0,450
ne_{re}	fe_{bo}	0,550
ja_{re}	tr_{bo}	0,055
ja_{re}	fe_{bo}	0,945

b)

re	sp	bo	$p(bo\|re, sp)$
ne_{re}	au_{sp}	tr_{bo}	0,450
ne_{re}	au_{sp}	fe_{bo}	0,550
ne_{re}	ei_{sp}	tr_{bo}	0,450
ne_{re}	ei_{sp}	fe_{bo}	0,550
ja_{re}	au_{sp}	tr_{bo}	0,055
ja_{re}	au_{sp}	fe_{bo}	0,945
ja_{re}	ei_{sp}	tr_{bo}	0,055
ja_{re}	ei_{sp}	fe_{bo}	0,945

tenstruktur werden die bedingten Häufigkeiten gespeichert, mit denen Kombinationen der Ausprägungen der Knoten seiner Familie in der Datenbasis vorkommen. Diese Häufigkeitenstrukturen geben unter anderem zu ihrem Typ passende Iteratoren zurück, welche es erlauben, sequentiell über die tatsächlich vorkommenden Ausprägungskombinationen und ihre bedingten Häufigkeiten zu iterieren, so dass die Gütemaße berechnet werden können. Die Häufigkeitenstrukturen unterliegen beim Ändern des Graphen des BNes durch das Hinzufügen oder Entfernen von Eltern bzw. Kanten sehr oft Struktur- und Werteänderungen, die akzeptabel schnell erfolgen müssen. Zudem iterieren die Algorithmen zur Berechnung der Gütemaße über diesen Häufigkeitenstrukturen, was sehr schnell erfolgen muss. Nicht zuletzt handelt es sich bei diesen Häufigkeitenstrukturen um logisch hochdimensionale Gebilde, welche – naiv umgesetzt – zu viel Speicherplatz beanspruchen oder zu lange Laufzeiten verursachen. Aus diesem Grunde sind drei alternative Häufigkeitenstrukturen zur Abbildung bedingter Häufigkeiten implementiert und bewertet worden:

- Ein *serieller Häufigkeitenraum* enthält ein Feld ganzer Zahlen. Seine Adressierung erfolgt analog zu der in Abschnitt 6.4.3 für Potenzialfunktionen beschriebenen. Während Struktur- und Werteänderungen sehr schnell vonstatten gehen, ist das Feld nur spärlich mit Häufigkeiten > 0 besetzt. Bei hochdimensionalen Familien ist nur ein Bruchteil seiner Elemente tatsächlich belegt. Der überproportional hohe Speicherbedarf wirkt sich dann zusätzlich negativ auf die zum Durchlaufen des seriellen Häufigkeitenraumes benötigte Rechenzeit aus.

- Eine *Häufigkeitenliste* verwaltet tatsächlich vorkommende Ausprägungskombinationen der Knoten einer Familie samt ihrer Häufigkeiten in einer sortierten Liste[5]. Der Speicherbedarf für die Häufigkeitenliste wächst maximal linear mit dem Umfang der Datenbasis, was von Vorteil ist. Das Einfügen neuer Ausprägungskombinationen verursacht aber relativ hohen Aufwand. Das Iterieren über den bedingten Häufigkeiten gestaltet sich dafür aber relativ schnell.

- Eine *ternärer Häufigkeitenbaum* verwendet intern einen ternären Baum. [BS97] empfehlen eine solche Datenstruktur für den vorliegenden Anwendungsfall. Beim ternären Baum besitzt jeder Elternknoten wie beim

[5]Intern handelt es sich bei dieser Liste um ein sortiertes Feld. Wenn ein Element nicht am Ende des Feldes, sondern weiter vorn aus dem Feld gelöscht oder in das Feld eingefügt wird, werden hinter dem Element liegende Elemente blockweise verschoben, und das Feld wird – falls nötig – vorab blockweise erweitert.

binären Baum eine Referenz auf „kleinere“ und eine Referenz auf „größere“ Kindknoten und zusätzlich eine Referenz auf Kindknoten „gleicher Größe“. Treten Ausprägungskombinationen auf, die noch nicht im Baum vorliegen, erweitert sich der Baum selbst um die notwendigen Knoten und hängt die Ausprägungskombinationen an den passenden Stellen ein. Das Erstellen des Baumes ist zwar aufwendig, aber das Finden und Aktualisieren bedingter Häufigkeiten geht sehr schnell. Da der ternäre Baum nur eine relativ geringe mittlere Tiefe aufweist und somit wenig Speicherplatz benötigt, ist er bei großen BNen, deren Knoten potenziell viele Eltern aufweisen, die Datenstruktur der Wahl.

Tests mit Daten, die mittels der Algorithmen aus den Abschnitten 6.5.1 und 6.5.2 generiert worden sind, haben ergeben, dass sich der serielle Häufigkeitenraum und der ternäre Häufigkeitenbaum am besten für das Akkumulieren und Iterieren über bedingten Häufigkeiten eignen. Eine einfache Heuristik legt fest, welche Häufigkeitenstruktur zu verwenden ist: Überschreitet die Ausdehnung des seriellen Häufigkeitenraumes eine bestimmte Grenze, wird der ternäre Häufigkeitenbaum verwendet.

B.3 Operationen über Potenzialfunktionen

Die Komponente zur Anwendung BNe fußt auf

1. den Potenzialfunktionen und Operationen, die Anhang A.1.1 definiert,
2. der speziellen Graphentheorie aus Anhang A.1.2
3. der exakten PI, die Anhang A.3 ausführt, und
4. der Komponente zur Repräsentation BNe aus Abschnitt 6.4.1.

Das Verstehen der Komponente zur Anwendung BNe setzt die Kenntnis des Inhaltes der vier Dinge voraus, welche die Punkte 1 bis 4 aufführen.

Nachdem die Komponente zur Akquisition BNe BNe mittels Gütemaßen und Algorithmen automatisch aus Daten erstellt, oder der Anwender diese BNe manuell aufbaut oder sie parallel modifiziert und dazu die GUI aus Abschnitt 6.6.1 nutzt, vollzieht die Komponente zur Anwendung BNe exakte PI über besagten BNen.

Anhang A.3 beschreibt bereits exakte PI umfassend und abstrakt, und Abschnitt 4.3.3 untermauert die Beschreibung der exakten PI ausführlich anhand des Beispiels „Marie und ihre Rosen“. Die Graphenalgorithmen A.3 bis A.10 aus diesem Abschnitt führt die Komponente zur Anwendung BNe

direkt über den Datenstrukturen aus, welche die Komponente zur Repräsentation BNe aus Abschnitt 6.4.1 enthält. Die Algorithmen A.11 und A.13 bis A.19 zum Aufbau der Potenzialfunktionen und zur exakten PI i. e. S. laufen ebenfalls direkt über den Datenstrukturen ab, welche die Komponente zur Repräsentation BNe bereitstellt. Ergo ist es überflüssig, die genannten Algorithmen noch einmal zu beschreiben.

Die letztgenannten acht Algorithmen enthalten allerdings komplexe Operationen. Der vorliegende Abschnitt stellt die effizienten prozeduralen Repräsentationen dieser Operationen en detail dar. Bei den komplexen Operationen handelt es sich um die folgenden:

1. die Multiplikation zweier n-dimensionaler, diskreter, reellwertiger Funktionen (siehe Definition A.6),

2. die Division zweier n-dimensionaler, diskreter, reellwertiger Funktionen (siehe Definition A.9),

3. die Absorbtion von Evidenz durch eine n-dimensionale, diskrete, reellwertige Funktion (siehe Definition A.39),

4. das Herausintegrieren eines Summenrandes aus einer n-dimensionalen, diskreten, reellwertigen Funktion (siehe Definition A.11) und

5. das Herausintegrieren eines Maximumrandes aus einer n-dimensionalen, diskreten, reellwertigen Funktion (siehe Definition A.12).

Potenzialfunktionen, Bewertungsfunktionen, Nachrichten, Ränder und Randverteilungen sind sämtlich n-dimensionale, diskrete, reellwertige Funktionen, so dass Operationen über ihnen auf die obigen fünf Operationen zurückgeführt werden können. Multiplikation und Division solcher Funktionen sind inverse Operationen und nahezu identisch. Sie unterscheiden sich lediglich dadurch, dass intern einmal eine Multiplikation und einmal eine Division zweier reeller Zahlen erfolgt. Darum wird hier nur auf eine der beiden Operationen eingegangen – und zwar auf die Multiplikation zweier solcher Funktionen. Das Herausintegrieren eines Summenrandes und das Herausintegrieren eines Maximumrandes aus einer solchen Funktion unterscheiden sich lediglich dadurch, dass intern einmal eine Summe und einmal ein Maximum reeller Zahlen berechnet wird. Deshalb wird hier nur auf eine der beiden Operationen eingegangen – und zwar auf das Herausintegrieren eines Summenrandes aus einer solchen Funktion.

B.3.1 Adressrechnung in Potenzialfunktionen

Es sei f eine Potenzialfunktion. Es sei $V = \{b_0, \ldots, b_{n-1}\}$ die geordnete Menge der Variablen, die f zugrundeliegen. Die Anzahl der Variablen von f ist $n = |V|$. Die geordnete Menge der Zustände der Variable $b_i \in V$ heiße S_i mit $i = 0, \ldots, (n-1)$. Die reellen Werte von f liegen als

$$l = \prod_{i=0}^{n-1} |S_i| \tag{B.1}$$

Elemente in einem Vektor bzw. Feld $\vec{v} = (v_0, \ldots, v_{l-1})$ vor. Abschnitt 6.4.1 begründet, warum es zweckmäßig ist, dass ein (eindimensionaler) Vektor $\vec{v}$ die Werte der Potenzialfunktion hält.

Argument von f ist ein Vektor $\vec{a} = (a_0, \ldots, a_{n-1})$ auf Null basierender Indizes der Zustände der Variablen V von f. Der Vektor $\vec{a}$ wird auch als *n-dimensionale Adresse* bezeichnet. Um den Funktionswert von f für das Argument $\vec{a}$ zu ermitteln bzw. um auf einen einzelnen Wert v der Werte $\vec{v}$ einer Potenzialfunktion f über $\vec{a}$ zuzugreifen, ist zuvörderst eine *Adresskonvertierung* vonnöten, die eine n-dimensionale Adresse $\vec{a}$ in eine eindimensionale Adresse, einen Index namens p, umwandelt. Zu Beginn der Adresskonvertierung ermittelt man rekursiv die *Konvertierungsfaktoren* $\vec{c} = (c_0, \ldots, c_{n-1})$:

$$c_i = \begin{cases} 1, & i = 0 \\ c_{i-1} \cdot |S_i|, & 0 < i \leq (n-1) \end{cases} . \tag{B.2}$$

Das Berechnen der Konvertierungsfaktoren erfolgt nur einmal für alle Zugriffe auf die Werte von f. Hernach wird die n-dimensionale Adresse $\vec{a}$ wie folgt in eine *eindimensionale Adresse*, einen Index namens p, konvertiert:

$$p = \sum_{i=0}^{n-1} c_i \cdot a_i. \tag{B.3}$$

Alsdann erfolgt mittels p der indizierte Zugriff auf das p-te Element v_p der Werte $\vec{v}$ von f. Der Zugriff auf einen einzelnen Wert v der Werte $\vec{v}$ einer Potenzialfunktion f über eine Adresse $\vec{a}$ erfordert ergo n Multiplikationen.

Eine eindimensionale Adresse p kann mittels Algorithmus B.1 in eine n-dimensionale Adresse $\vec{a}$ zurückkonvertiert werden. Die Adresskonvertierung ist eineindeutig.

B.3.2 Partielle Iteration über Potenzialfunktionen

Um Potenzialfunktionen miteinander zu multiplizieren (siehe Definitionen A.6 und A.7), Ränder aus ihnen herauszuintegrieren (siehe Definitionen A.11

Algorithmus B.1 Konvertieren einer ein- in eine n-dimensionale Adresse

Eingabe: eine Potenzialfunktion f mit ihren Konvertierungsfaktoren $\vec{c}$, eine eindimensionale Adresse p

Ausgabe: eine n-dimensionale Adresse $\vec{a}$

```
1: funktion KonvertiereEinInMehrdimensionaleAdresse(f, p)
2:     a⃗ ← lege Adresse mit n Elementen an
3:     für i ← (n − 1), . . . , 0 führe aus      ▷ Durchlaufe Indizes rückwärts
4:         a_i ← p ÷ c_i                          ▷ gleichbedeutend mit a_i ← ⌊p / c_i⌋
5:         p ← p mod c_i
6:     ende für
7:     gib zurück a⃗
8: ende funktion
```

und A.12) oder sie Evidenz absorbieren zu lassen (siehe A.39), ist es nötig, über diesen Potenzialfunktionen vollständig oder partiell zu iterieren. Vollständige Iteration über einer Potenzialfunktion durchläuft die Elemente des Vektors $\vec{v}$ in der Reihenfolge, in der sie in ihm auftreten, und kann über indizierten Zugriff auf ein Feld oder über das Inkrementieren eines Schreib- und Lesezeigers und das Dereferenzieren dieses Zeigers relativ einfach realisiert werden.

Partielle Iteration über einer Potenzialfunktion erfolgt dann, wenn Variablen der Potenzialfunktion auf bestimmte ihrer Zustände gesetzt, wenn sie *fixiert* worden sind. Welche Variablen einer Potenzialfunktion fixiert und auf welche Zustände die Variablen gesetzt worden sind, sage ein Vektor $\vec{z} = (z_0, \ldots, z_{n-1})$ aus, dessen Elemente Zeiger sind, die auf Indizes der Zustände zeigen, auf welche die entsprechenden Variablen fixiert worden sind. Zeigt ein Element z des Vektors $\vec{z}$ auf die Konstante -1, so ist die korrespondierende Variable nicht fixiert worden.

Partielle Iteration durchläuft im Gegensatz zur vollständigen Iteration nicht alle Elemente des Vektors v, sondern nur diejenigen, die n-dimensionale Adressen $\vec{a}$ aufweisen, deren Elemente mit den korrespondierenden Elementen des Vektors $\vec{z}$ übereinstimmen, die nicht -1 gleichen. Im Gegensatz zur vollständigen Iteration ist die partielle Iteration schwierig effizient zu implementieren. Würde man die n-dimensionale Adresse $\vec{a}$ mit Nullen initialisieren, die fixierten Indizes in $\vec{a}$ ebenfalls festsetzen, über den verbleibenden Elementen bzw. Stellen von $\vec{a}$ iterieren (wie beim Zählwerk eines Kassettenspielers) und bei jeder Iteration nach Formel B.3 mit $\vec{a}$ auf ein Element v von $\vec{v}$ zugreifen, so vollzöge man für eine partielle Iteration im pathologischen Falle, in dem kein Zustand fixiert worden ist, $l \cdot n$ Multiplikationen ganzer Zahlen,

was für PI i. e. S. untragbar wäre. Das Ziel besteht nun darin, mit möglichst wenig Rechenaufwand partiell über die Werte einer Potenzialfunktion f zu iterieren.

Der Vektor $\vec{m}$ der *maximalen Zustandsindizes* enthalte für jede Variable $b \in V$ den maximalen Zustandsindex m. Der Index m stimmt nicht mit der Anzahl der Zustände $|S|$ der Variable $b \in V$ überein, sondern beträgt $m = |S| - 1$, da die Indizierung der Zustände der Variablen bei 0 beginnt.

Der Vektor $\vec{d}$ der *Lese- und Schreibindexdekremente* enthalte für jede Variable $b \in V$ das Produkt aus dem korrespondierenden Adresskonvertierungsfaktor c und dem korrespondierenden maximalen Zustandsindex m.

Für die partielle Iteration ist lediglich die geordnete Menge $V' \subset V$ der Variablen relevant, die *nicht* auf einen bestimmten ihrer Zustände gesetzt bzw. fixiert worden sind, deren korrespondierende Zeiger z also auf -1 weisen. Demzufolge werden bei der Initialisierung der partiellen Iteration folgende Vektoren gefüllt: Die Adresse $\vec{a'}$ enthalte für jede Variable $b \in V'$ den Wert 0. Der Vektor $\vec{m'}$ enthalte für jede Variable $b \in V'$ den korrespondierenden maximalen Zustandsindex m aus dem Vektor $\vec{m}$ aller maximalen Zustandsindizes. Der Vektor $\vec{c'}$ enthalte für jede Variable $b \in V'$ den mit ihr korrespondierenden Adresskonvertierungsfaktor c aus dem Vektor $\vec{c}$ aller Adresskonvertierungsfaktoren. Der Vektor $\vec{d'}$ enthalte für jede Variable $b \in V'$ das korrespondierende Lese- und Schreibindexdekrement d aus dem Vektor $\vec{d}$ aller Lese- und Schreibindexdekremente.

Da bereits die Variablen $V' \in V$ fixiert worden sind, startet die partielle Iteration im Allgemeinen nicht beim 0-ten Element des Vektors $\vec{v}$ der Werte der Potenzialfunktion f, sondern die Lese- und Schreibposition p wird bereits wie folgt initialisiert[6]:

$$p = \sum_{i=0}^{n-1} \begin{cases} z_i \uparrow \cdot c_i, & z_i \uparrow \geq 0 \\ 0, & z_i \uparrow = -1 \end{cases} . \tag{B.4}$$

Das Initialisieren der partiellen Iteration endet mit dem Setzen der binären Variable h auf WAHR, das aussagt, dass noch mindestens ein weiterer Wert im Rahmen der partiellen Iteration vorliegt.

Der entscheidende und rechenaufwendigste Schritt bei der partiellen Iteration ist das Weitersetzen – hier auch Inkrementieren – der Schreib- und Leseposition p über den Werten $\vec{v}$ der Potenzialfunktion f. Algorithmus B.2 vollzieht diesen Schritt.

[6]Der Postfix-Operator $\uparrow$, angewandt auf einen Zeiger z, gebe den Wert zurück, auf den der Zeiger z weist, im hiesigen Falle eine ganze Zahl, den Index, auf den die mit z korrespondierende Variable fixiert worden ist.

Algorithmus B.2 Inkrementieren der Schreib- und Leseposition

Eingabe: eine Potenzialfunktion f mit der Schreib- und Leseposition p, der Schreib- und Leseadresse $\vec{a'}$, den Vektoren $\vec{m'}$, $\vec{d'}$ und $\vec{c'}$ und dem Wahrheitswert h

Ausgabe: Funktion f mit inkrementiertem p und aktualisiertem $\vec{a'}$ und h

```
 1: prozedur InkrementiereSchreibeLesePosition(f)
 2:     i ← rufe GibStellenAnzahl(a') −1
 3:     solange (i ≥ 0) ∧ (a_i = m_i), führe aus
 4:         a_i ← 0
 5:         p ← p − d_i
 6:         i ← i − 1
 7:     ende solange
 8:     wenn i ≥ 0, dann
 9:         a_i ← a_i + 1
10:         p ← p + c_i
11:     sonst
12:         h ← FALSCH
13:     ende wenn
14: ende prozedur
```

Er beginnt bei jedem Aufruf mit der letzten Stelle der Adresse $\vec{a'}$, indem er den Adressindex i auf den Index dieser Stelle setzt. Solange der Adressindex i noch nicht nach links aus der Adresse $\vec{a'}$ herausgelaufen ist und die aktuelle Stelle a_i ihren Maximalwert erreicht hat, initialisiert er die aktuelle Stelle a_i mit 0, dekrementiert die Lese- und Schreibposition p um die Anzahl d_i von Positionen, die dem Zurücksetzen der aktuellen Stelle a_i auf ihren Initialwert 0 entspricht, und dekrementiert den Adressindex i um eins bzw. wählt die nächsthöhere Stelle der Adresse $\vec{a'}$ (eine Stelle weiter links).

Wenn nun der Adressindex i noch nicht nach links aus der Adresse $\vec{a'}$ herausgelaufen ist, inkrementiert der Algorithmus die aktuelle Stelle a_i der Adresse $\vec{a'}$ um eins und die Schreib- und Leseposition p um die Anzahl c_i von Positionen, die dem Inkrementieren der aktuelle Stelle a_i um eins entsprechen. Sonst signalisiert der Algorithmus, dass die Potenzialfunktion f keine weiteren Werte aufweist, deren Adressen mit den fixierten Stellen in $\vec{z}$ harmonieren, bzw. dass die Werte $\vec{v}$ der Potenzialfunktion f vollständig partiell durchlaufen worden sind, indem er die binäre Variable h auf FALSCH setzt.

Auf abstrakter Ebene betrachtet, vollzieht Algorithmus B.2 zweierlei: Er schaltet erstens die Adresse $\vec{a'}$ weiter – ähnlich dem Zählwerk eines Kassettenspielers –, und er setzt zweitens – auf der Adresse $\vec{a}$ basierend – die

Lese- und Schreibposition p weiter. Algorithmus B.2 nimmt also keine Adresskonvertierung und somit auch keinerlei Multiplikationen mehr vor, sondern lediglich eine minimale Anzahl von Inkrement- und Dekrementoperationen, die ihn, die gesamte partielle Iteration und somit auch die Komponente zur Anwendung BNe zur PPS effizient machen.

Die Algorithmen B.3 und B.4 rufen Algorithmus B.2 auf. Sie geben den *nächsten* Wert v der Werte $\vec{v}$ einer Potenzialfunktion f im Rahmen einer partiellen Iteration zurück respektive setzen ihn. Die Algorithmen zum Zurückgeben und Setzen des *aktuellen* Wertes v von f greifen lediglich auf die Schreib- und Leseposition p zu, ohne sie zu verändern. Auf ihre Darstellung wird an dieser Stelle bewusst verzichtet.

Algorithmus B.3 Zurückgeben des nächsten Wertes

Eingabe: eine Potenzialfunktion f mit dem Vektor $\vec{v}$ ihrer Werte und der Schreib- und Lesepostion p
Ausgabe: der nächste Wert v der Potenzialfunktion f

```
1: funktion GibNächstenWert(f)
2:     v ← v_p
3:     rufe InkrementiereSchreibeLesePosition(f)
4:     gib zurück v
5: ende funktion
```

Algorithmus B.4 Setzen des nächsten Wertes

Eingabe: eine Potenzialfunktion f mit dem Vektor $\vec{v}$ ihrer Werte und der Schreib- und Lesepostion p sowie der Wert v
Ausgabe: die der Potenzialfunktion f mit dem nächsten Wert v

```
1: prozedur SetzeNächstenWert(f, v)
2:     v_p ← v
3:     rufe InkrementiereSchreibeLesePosition(f)
4: ende prozedur
```

B.3.3 Multiplikation zweier Potenzialfunktionen

Die erste komplexe Operation über Potenzialfunktionen, welche die bis hierher gelegten Grundlagen implizit und explizit nutzt, ist die *Multiplikation* von Potenzialfunktionen. Algorithmus B.5 stellt sie dar.

Bei dieser Operation wird die Multiplikation so verstanden, dass der Multiplikand, nachdem die Operation ausgeführt worden ist, das Produkt reprä-

Algorithmus B.5 Multiplikation zweier Potenzialfunktionen

Eingabe: zwei Potenzialfunktionen f und g mit den Variablen V und U, wobei $U \subset V$

Ausgabe: die Potenzialfunktion f multipliziert mit g

```
1: prozedur MultipliziereePotenzialFunktionen(f, g)
2:     rufe InitialisiereVollständigeIteration(g)
3:     rufe SetzeFixierteAdressStellenZurück(f)
4:     n ← rufe GibAnzahlVariablen(g)
5:     für i ← 0, ..., (n − 1) führe aus
6:         b ← rufe GibVariable(g, i)
7:         j ← rufe GibVariablenPosition(f, b)
8:         z ← rufe GibAdressStellenZeiger(g, i)
9:         rufe FixiereAdressStelle(f, j, z)
10:    ende für
11:    solange rufe HatWeitereWerte(g), führe aus
12:        rufe InitialisierePartielleIteration(f, FALSCH)
13:        v ← rufe GibNächstenWert(g)
14:        solange rufe HatWeitereWerte(f), führe aus
15:            u ← rufe GibAktuellenWert(f)
16:            rufe SetzeNächstenWert(f, u · v)
17:        ende solange
18:    ende solange
19: ende prozedur
```

sentiert. Es entsteht also keine neue Potenzialfunktion, sondern der Multiplikator wird in den Multiplikanden gewissermaßen „hineinmultipliziert“. Damit eine solche Multiplikation vonstatten gehen kann, muss $U \in V$ gelten, wobei V die Menge der Variablen des Multiplikanden f und U die Menge der Variablen des Multiplikators g bezeichnet.

Algorithmus B.5 zur Multiplikation zweier Potenzialfunktionen arbeitet grob wie folgt: Er durchläuft die Werte des Multiplikators g vollständig. In jeder Iteration fixiert er beim Multiplikanden f die Variablen auf die Stellen, auf denen die Schreib- und Leseadresse des Multiplikators g gerade steht. Dann durchläuft er den Multiplikanden f partiell und multipliziert jeden Wert des Multiplikanden f mit dem aktuellen Wert des Multiplikators g. Das Fixieren der Stellen der Lese- und Schreibadresse des Multiplikanden f erfolgt über Zeiger auf die Lese- und Schreibadresse des Multiplikators g.

Im Einzelnen agiert Algorithmus B.5 wie folgt: Er initialisiert eine vollständige Iteration beim Multiplikator g, da der ja vollständig zu durchlaufen ist. Er fixiert keine Stellen der Lese- und Schreibadresse des Multiplikators g, sondern setzt nur die Lese- und Schreibposition von g auf den 0-ten Wert und initialisert die Lese- und Schreibadresse von g mit 0.

Bei der Lese- und Schreibadresse des Multiplikanden f fixiert Algorithmus B.5 alsdann die Stellen, an denen sich die Variablen des Multiplikators g im Multiplikanden f wiederfinden. Dabei zeigen die fixierten Stellen auf die entsprechenden Stellen in der Lese- und Schreibadresse des Multiplikators g, so dass sich die fixierten Stellen des Multiplikanden f mit dem Weiterschalten des Multiplikators g ändern. Anders ausgedrückt, setzt Algorithmus B.5 alsdann eventuell fixierte Adressstellen des Multiplikanden f zurück, um sie im Anschluss neu zu setzen: Er durchläuft die Variablen des Multiplikators g, identifiziert die Position j jeder i-ten Variable b in den Variablen des Multiplikanden f, ermittelt den Adressstellenzeiger z für die i-te Variable b des Multiplikators und fixiert die j-te Variable des Multiplikanden f auf z. Wird der Multiplikator g im Anschluss weitergeschaltet, so sind die Elemente des Vektors $\vec{z}$ immer auf die korrekten Indizes fixiert, weil nicht Kopien der Adressstellen des Multiplikators g vorliegen, sondern die Zeiger $\vec{z}$ auf die Adressstellen des Multiplikators g.

Solange der Multiplikator g nun noch weitere Werte aufweist, die binäre Variable h also WAHR ist, iteriert Algorithmus B.5 weiterhin vollständig über ihm. Die partielle Iteration über dem Multiplikanden f initialisiert er als erstes. Der zweite Parameter, der den Wert FALSCH trägt, sagt aus, dass die fixierten Stellen erhalten bleiben. Es handelt sich bei ihnen um die Zeiger $\vec{z}$ in f, die auf die aktuelle Lese- und Schreibadresse der Multiplikators g zeigen. Den aktuellen bzw. nächsten Wert des Multiplikators g ermittelt Algorithmus B.3 und Variable v speichert ihn zwischen. Solange nun der Mul-

tiplikand f nun noch weitere Werte bereithält, über denen partiell iteriert werden kann, übergibt der Multiplikand f seinen aktuellen Wert der Hilfsvariable u, und der aktuelle bzw. nächste Wert des Multiplikanden f ergibt sich aus dem Produkt seines aktuellen Wertes u und dem aktuellen Wert v des Multiplikators g.[7] Sind die beiden ineinander geschachtelten kopfgesteuerten Wiederholschleifen durchlaufen worden, ist die Multiplikation erfolgt.

Es sei noch einmal explizit darauf hingewiesen, dass die Algorithmen B.3 und B.4 zum Holen respektive Setzen des nächsten Wertes einer Potenzialfunktion eigentlich den aktuellen Wert der Potenzialfunktion zurückgeben bzw. setzen und an ihrem jeweiligen Ende den Algorithmus B.2 zum Inkrementieren der Schreib- und Leseposition der Potenzialfunktion aufrufen.

B.3.4 Absorbieren von Evidenz

Die zweite komplexe Operation über Potenzialfunktionen, welche die zu Beginn dieses Abschnittes gelegten Grundlagen implizit und explizit nutzt, ist das *Absorbieren von Evidenz* durch eine Potenzialfunktion. Algorithmus B.6 stellt sie dar.

Algorithmus B.6 zur Absorbtion von Evidenz durch eine Potenzialfunktion agiert wie folgt: Er kopiert die betroffene Potenzialfunktion f samt ihrer Evidenz und legt die Kopie in der Variable g ab. Alsdann initialisiert er die Werte der Potenzialfunktion g allesamt mit 0. Hiernach durchläuft er alle n Variablen der Potenzialfunktionen und lässt sich jeweils den Zeiger z auf die evidente Position der i-ten Variable geben. Liegt für die i-te Variable keine Evidenz vor, so gibt Funktion GibEvidenzPositionsZeiger den Zeiger z auf die Konstante -1 zurück. Algorithmus B.6 fixiert dann in der originalen Potenzialfunktion f und in ihrer Kopie g die Adressstelle, die zur i-ten Variable gehört, auf den Zeiger z. Im Anschluss an die Zählschleife initialisiert der Algorithmus die partielle Iteration über beiden Potenzialfunktionen gleichermaßen, und zwar ohne die fixierten Adressstellen zurückzusetzen (zweiter Parameter FALSCH). Dann iteriert er über beiden Potenzialfunktionen parallel: Solange noch weitere Werte, über denen partiell iteriert werden soll, in den Potenzialfunktionen f und g vorliegen, wird der nächste Wert der originalen Potenzialfunktion f auf die Variable v und von ihr auf den nächsten Wert der Potenzialfunktionskopie g übertragen. Nach dem Verlassen der kopfgesteuerten Wiederholschleife weist die Potenzialfunktionskopie g an den Stellen, deren Lese- und Schreibadressen mit der Evidenz überein-

[7] Die Division der Potenzialfunktion f durch die Potenzialfunktion g unterscheidet sich lediglich dadurch von der Multiplikation beider Potenzialfunktionen, dass an dieser Stelle nicht das Produkt $u \cdot v$, sondern der Quotient $\frac{u}{v}$ berechnet wird(, wobei gegebenenfalls eine Division durch 0 abzufangen ist).

Algorithmus B.6 Absorbieren von Evidenz

Eingabe: eine originale Potenzialfunktion f mit evidenten Variablen
Ausgabe: eine Kopie von f namens g mit absorbierter Evidenz

```
1: funktion AbsorbiereEvidenz(f)
2:     g ← rufe KopierePotenzialFunktion(f)
3:     rufe InitialisiereWerte(g, 0)
4:     n ← rufe GibAnzahlVariablen(f)
5:     für i ← 0, ..., (n − 1) führe aus
6:         z ← rufe GibEvidenzPositionsZeiger(g, i)
7:         rufe FixiereAdressStelle(f, i, z)
8:         rufe FixiereAdressStelle(g, i, z)
9:     ende für
10:    rufe InitialisierePartielleIteration(f, FALSCH)
11:    rufe InitialisierePartielleIteration(g, FALSCH)
12:    solange rufe HatWeitereWerte(f), führe aus
13:        v ← rufe GibNächstenWert(f)
14:        rufe SetzeNächstenWert(g, v)
15:    ende solange
16:    gib zurück g
17: ende funktion
```

stimmen, die Werte der originalen Potenzialfunktion f auf, und an all den Stellen, deren Lese- und Schreibadressen *nicht* mit der Evidenz übereinstimmen, enthält die Potenzialfunktionskopie g jeweils den Wert 0. Die originale Potenzialfunktion f kann nun außerhalb des Algorithmus' B.6 durch ihre Kopie g, welche Evidenz absorbiert hat, ersetzt werden.[8] Das Ergebnis des Algorithmus' B.6 entspricht demzufolge exakt der Definition A.39. Algorithmus B.6 setzt aber nicht die Stellen der Potenzialfunktion f auf den Wert 0, die nicht der Evidenz entsprechen, sondern kopiert die Potenzialfunktion f, setzt alle Werte ihrer Kopie g auf den Wert 0 und kopiert die Werte, die der Evidenz entsprechen, aus der originalen Potenzialfunktion f in ihre Kopie g.

B.3.5 Herausintegrieren eines Summenrandes

Die dritte und letzte komplexe Operation über Potenzialfunktionen, welche die zu Beginn dieses Abschnittes gelegten Grundlagen implizit und explizit nutzt, ist das *Herausintegrieren* eines Summenrandes aus einer Potenzialfunktion. Algorithmus B.7 stellt sie dar.

Der „Initialisierungsteil" des Algorithmus' B.7, der nach der Zählschleife endet, entspricht exakt dem des Algorithmus' B.5 zur Multiplikation zweier Potenzialfunktionen, ist bereits ebenda verbal beschrieben worden und wird demzufolge hier nicht nochmals erläutert. Nach seinem Initialisierungsteil iteriert Algorithmus B.7 vollständig über den Werten des Summenrandes g: Solange der Summenrand g noch weitere Werte aufweist, initialisiert Algorithmus B.7 die partielle Iteration über der Potenzialfunktion f, ohne die fixierten Adressstellen zurückzusetzen (zweiter Parameter FALSCH). Die reellwertige Variable v, welche einen Wert des Summenrandes extern akkumulieren soll, erhält ihren Initialwert 0. Solange die Potenzialfunktion f noch weitere Werte aufweist, erhält die reelle Variable u den nächsten Wert der Potenzialfunktion f, und der Wert der Variablen v wird um den der Variablen u erhöht[9]. Nach dem Ende der inneren Wiederholschleife, setzt Algorithmus B.7 den nächsten Wert des Summenrandes auf den Wert der Summe v. Nachdem auch die äußere Wiederholschleife abgearbeitet worden ist, liegt der Summenrand g vollständig gefüllt vor.

[8] Es bietet sich an, die originale Potenzialfunktion f im Rahmen einer PI zwischenzuspeichern, da sie oft für weitere PI verwendet wird und dann jedesmal durch rechenaufwendige Multiplikationen von Bewertungsfunktionen neu erstellt werden müsste.

[9] Das Herausintegrieren eines Maximumrandes g aus der Potenzialfunktion f unterscheidet sich lediglich dadurch vom Herausintegrieren eines Summenrandes g aus der Potenzialfunktion f, dass an dieser Stelle nicht die Summe $v + u$, sondern das Maximum $\max(v, u)$ berechnet wird.

Algorithmus B.7 Herausintegrieren eines Summenrandes

Eingabe: zwei Potenzialfunktionen f und g mit den Variablen U und V, wobei $V \subset U$

Ausgabe: der aus f herausintegrierte Summenrand g

```
1: prozedur IntegriereHeraus(f, g)
2:     rufe InitialisiereVollständigeIteration(g)
3:     rufe SetzeFixierteAdressStellenZurück(f)
4:     n ← rufe GibAnzahlVariablen(g)
5:     für i ← 0, ..., (n − 1) führe aus
6:         b ← rufe GibVariable(g, i)
7:         j ← rufe GibVariablenPosition(f, b)
8:         z ← rufe GibAdressStellenZeiger(g, i)
9:         rufe FixiereAdressStelle(f, j, z)
10:    ende für
11:    solange rufe HatWeitereWerte(g), führe aus
12:        rufe InitialisierePartielleIteration(f, FALSCH)
13:        v ← 0
14:        solange rufe HatWeitereWerte(f), führe aus
15:            u ← rufe GibNächstenWert(f)
16:            v ← v + u
17:        ende solange
18:        rufe SetzeNächstenWert(g, v)
19:    ende solange
20: ende prozedur
```

B.4 Komponente zur Generierung von PPS-Daten

B.4.1 Untersuchung der $10 \cdot 10$-Fisher-Thompson-Probleminstanz

Bei der Analyse der für die 10·10-Fisher-Thompson-Probleminstanz erstellten Maschinenübergangsmatrix sind mehrere Phänomene aufgefallen:

1. Die Maschinenübergänge häufen sich auf der und im Bereich um die Hauptdiagonale. Die Anzahlen der Maschinenübergänge in den Elementen direkt rechts bzw. direkt unterhalb der Hauptdiagonale sind allerdings jeweils 0.

2. Die Anzahl der Maschinenübergänge nimmt mit zunehmendem absoluten horizontalen und vertikalen Abstand von der Hauptdiagonale überproportional stark ab.

3. In den Elementen in und oberhalb bzw. rechts der Hauptdiagonale treten insgesamt mehr als doppelt so viele Maschinenübergänge auf (76) wie unterhalb bzw. links der Hauptdiagonale (36).

Tabelle B.3 und Abbildung B.2 stellen besagte Phänome noch einmal quantitativ respektive grafisch dar. Bei der „Maschine 0“ handelt es sich um das Umfeld der Fertigung, aus dem die Aufträge kommen und in das sie, wenn sie bearbeitet worden sind, zurückkehren, bzw. um entsprechende Lager. Wie ein idealisierter, aus der Maschinenübergangsmatrix extrahierter Zeilenvektor aussähe, zeigt Abbildung B.3 in einem Diagramm. Dass die Maschinenübergangsmatrix für die $10 \cdot 10$-Fisher-Thompson-Probleminstanz Regelmäßigkeiten aufweist und die Übergänge nicht zufällig über der Matrix gleichverteilt worden sind, ist a priori nicht erwartet worden, a posteriori allerdings plausibel. Die folgenden drei Aussagen begründen die oben erforschten Phänomene:

1. Die Maschinen sind in der Fertigung so angeordnet, dass die Aufträge häufiger kurze als lange Wege zwischen zwei im Sinne des Arbeitsplanes aufeinanderfolgenden Maschinen zurücklegen. Die Aufträge bewegen sich also von einer Maschine häufiger zu einer im Sinne der Indizierung der Maschinen näherliegenden Maschine als zu einer weiter entfernten. (Die Aufträge belegen allerdings nicht zweimal hintereinander dieselbe Maschine.)

2. Weite Wege zwischen zwei im Sinne des Arbeitsplanes aufeinanderfolgenden Maschinen sind überproportional weniger häufig bzw. wahrscheinlich.

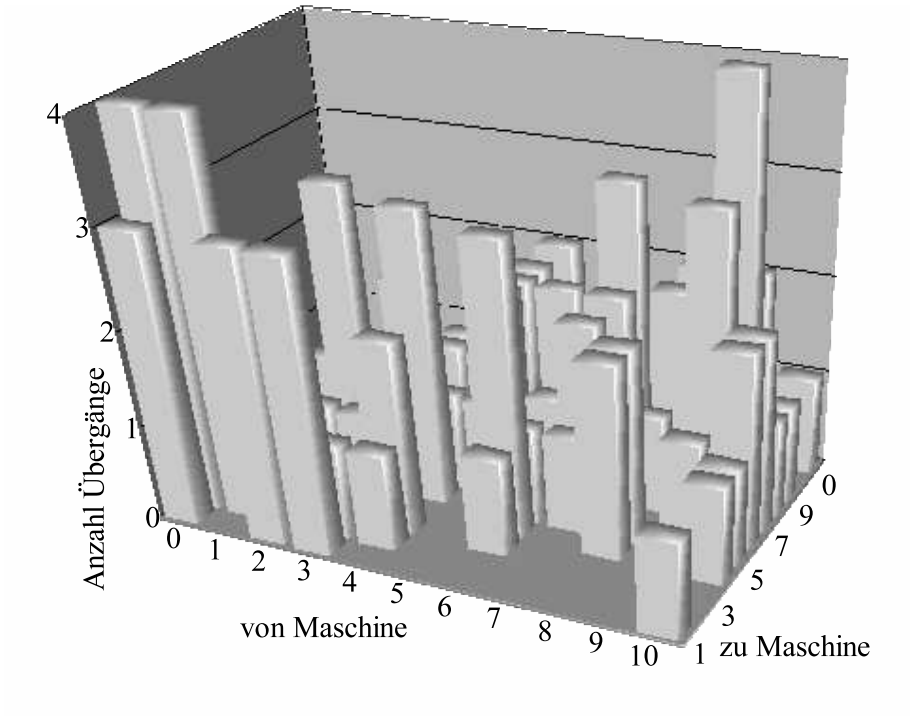

Abbildung B.2: Maschinenübergangsmatrix mit den absoluten Übergangshäufigkeiten h_{ij} von Maschine M_i zu Maschine M_j für die $10 \cdot 10$-Fisher-Thompson-Probleminstanz

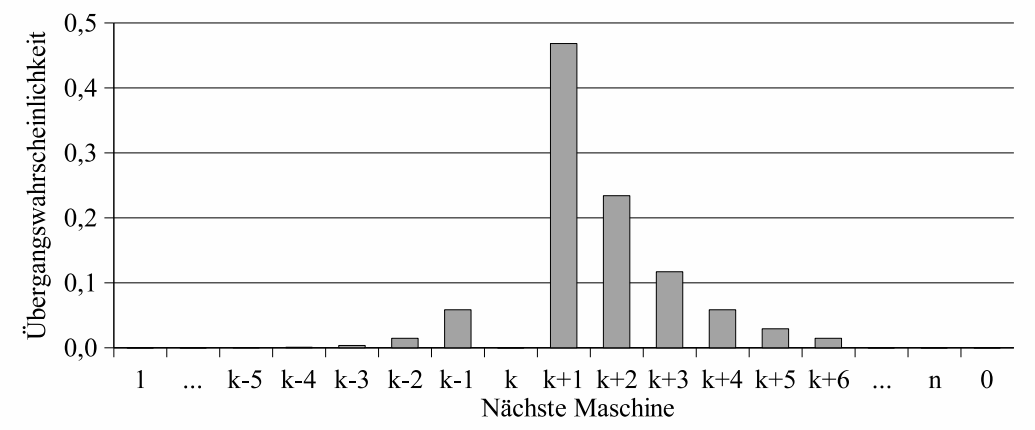

Abbildung B.3: Idealisierter Zeilenvektor aus einer Maschinenübergangsmatrix, der die Wahrscheinlichkeiten für die Übergänge *von* Maschine k *zu* den Maschinen $1, \ldots, n, 0$ beschreibt

Tabelle B.3: Maschinenübergangsmatrix mit den absoluten Übergangshäufigkeiten h_{ij} von Maschine M_i zu Maschine M_j für die $10 \cdot 10$-Fisher-Thompson-Probleminstanz

Von M_i	… zu M_j										
	1	2	3	4	5	6	7	8	9	10	0
0	3	4	3	0	0	0	0	0	0	0	0
1	0	4	2	2	1	0	1	0	0	0	0
2	3	0	2	1	0	3	1	0	0	0	0
3	3	1	0	1	1	0	2	0	1	1	0
4	0	1	2	0	3	1	0	0	1	1	1
5	0	0	0	0	0	1	2	2	2	1	2
6	0	0	1	3	1	0	1	2	0	1	1
7	0	0	0	0	1	2	0	2	3	1	1
8	0	0	0	2	2	0	1	0	2	1	2
9	0	0	0	0	0	1	0	3	0	4	2
10	1	0	0	1	1	2	2	1	1	0	1

3. Die Aufträge bewegen sich tendenziell in einer Richtung – nämlich vorwärts im Sinne der Indizierung der Maschinen – durch die Fertigung.

Es sei M die Menge der Maschinen. Ist in einer Maschinenübergangsmatrix, welche die relativen Häufigkeiten bzw. die Wahrscheinlichkeiten der Maschinenübergänge beschreibt, in jeder Zeile ein Element 1, und alle anderen Elemente der Matrix sind 0, gilt also

$$\forall i \in \{0, \ldots, |M|\} \left[\left(\max_{j=0}^{|M|} (p_{ij}) = 1 \right) \wedge \left(\sum_{j=0}^{|M|} (p_{ij}) = 1 \right) \right], \tag{B.5}$$

so handelt es sich um reine Fließfertigung: Sämtliche Aufträge weisen exakt dieselbe Maschinenfolge auf. Sind hingegen sämtliche Elemente eines jeden Zeilenvektors gleichverteilt, gilt also:

$$\forall i, j \in \{0, \ldots, |M|\} \left(p_{ij} = \frac{1}{|M| + 1} \right), \tag{B.6}$$

so handelt es sich um reine Werkstattfertigung: Jede mögliche Maschinenfolge ist gleich wahrscheinlich. Diese Form der Werkstattfertigung dürfte in der Praxis allerdings kaum vorkommen, da eine gewisse Fließrichtung immer gegeben ist.

B.4.2 Erzeugen einer Maschinenübergangsmatrix aufgrund eines Organisationsgrades

Algorithmus B.8 erzeugt eine Maschinenübergangsmatrix aufgrund eines gegebenen Organisationsgrades o und einer Menge M von Maschinen: Zu Beginn wird eine Maschinenübergangsmatrix U entsprechend der obigen Annahmen so initialisiert, dass sie einen mittleren Organisationsgrad $o(U)$ zwischen 0 und 1 aufweist. Die Zeilenvektoren der Matrix U entsprechen in etwa der Verteilung aus Abbildung B.3. Ist der Organisationsgrad $o(U)$ größer als der gewünschte Organisationsgrad o, so wird die Matrix U der Matrix R zugewiesen, und die Matrix L wird so initialisiert, dass sie einen Organisationsgrad von $o(L) = 0$ aufweist, der reiner Werkstattfertigung entspricht (siehe Formel B.6). Ist der Organisationsgrad $o(U)$ hingegen kleiner als der gewünschte Organisationsgrad o, so wird die Matrix U der Matrix L zugewiesen, und die Matrix R wird so initialisiert, dass sie einen Organisationsgrad von $o(R) = 1$ aufweist, der reiner Fließfertigung entspricht (siehe Formel B.5). Die Matrix L liegt nun mit ihrem Organisationsgrade $o(L)$ links bzw. unterhalb und die Matrix R mit ihrem Organisationsgrade $o(R)$ rechts bzw. oberhalb des gewünschten Organisationsgrades o.

Solange sich nun der Organisationsgrad $o(U)$ der Matrix U noch nicht innerhalb der Epsilon-Umgebung ε des gewünschten Organisationsgrades o befindet, wird aus der „linken“ Matrix L und der „rechten“ Matrix R eine Matrix bestimmt, deren Organisationsgrad zwischen den Organisationsgraden $o(L)$ und $o(R)$ der Matrizen L und R liegen, und diese Matrix wird der Matrix U zugewiesen. Ist der Organisationsgrad $o(U)$ größer als der gewünschte Organisationsgrad o, so wird die Matrix U der Matrix R zugewiesen. Ist der Organisationsgrad $o(U)$ hingegen kleiner als der gewünschte Organisationsgrad o, so wird die Matrix U der Matrix L zugewiesen. Die Matrizen L und R liegen nun wiederum mit ihren Organisationsgraden $o(L)$ und $o(R)$ links und rechts respektive unterhalb und oberhalb des gewünschten Organisationsgrades o. Befindet sich der Organisationsgrad $o(U)$ der Matrix U innerhalb der Epsilon-Umgebung ε des gewünschten Organisationsgrades o, so terminiert die kopfgesteuerte Wiederholschleife.[10] In einer nachgelagerten Stufe erzeugt der Algorithmus B.8 in der Maschinenübergangsmatrix U zeilenweise Verteilungs*funktionen* aus den einzelnen Übergangswahrscheinlichkeiten. Letztendlich gibt er die Maschinenübergangsmatrix U zurück.

[10] Bei der Schleife handelt es sich ergo um eine Art Halbierungsverfahren, das die Matrix U sukzessive dahingehend modifiziert, dass sich ihr Organisationsgrad $o(U)$ dem gewünschten Organisationsgrad beliebig nah annähert.

Algorithmus B.8 Erzeugen einer Maschinenübergangsmatrix

Eingabe:
Maschinen M,
Organisationsgrad o mit $0 \leq o \leq 1$,
ε-Umgebung $\varepsilon = 0{,}001$ als Standardwert für den Organisationsgrad o

Ausgabe:
eine $(|M|+1)^2$-Maschinenübergangsmatrix U
mit dem Organisationsgrad $|o(U) - o| \leq \varepsilon$

1: **funktion** ErzeugeMaschinenÜbergangsMatrix($M, o, \varepsilon \leftarrow 0{,}001$)
2: Initialisiere U entsprechend obiger Annahmen, so dass sie einen „mittleren“ Organisationsgrad $o(U)$ aufweist mit $0 < o(U) < 1$.
3: **wenn** $o < o(U)$, **dann** ▷ Wenn $0 \leq o < o(U)$, dann ...
4: $R \leftarrow U$ ▷ ... „rechte“ Matrix erhält U.
5: **für alle** $i, j \in \{0, \ldots, |M|\}$ **führe aus**
6: $L_{ij} \leftarrow \dfrac{1}{|M|+1}$ ▷ „Linke“ Matrix reine Werkstattfertigung
7: **ende für**
8: **sonst** ▷ Wenn $o(U) \leq o \leq 1$, dann ...
9: $L \leftarrow U$ ▷ ... „linke“ Matrix erhält U.
10: **für alle** $i, j \in \{0, \ldots, |M|\}$ **führe aus**
11: $R_{ij} \leftarrow \begin{cases} 1 | i = j \\ 0 | i \neq j \end{cases}$ ▷ „Rechte“ Matrix reine Fließfertigung
12: **ende für**
13: **ende wenn**
14: **solange** $|o(U) - o| \geq \varepsilon$, **führe aus** ▷ Solange o nicht erreicht, ...
15: $U \leftarrow \dfrac{L+R}{2}$ ▷ Bestimme „mittlere“ Matrix.
16: **wenn** $o < o(U)$, **dann** ▷ Wenn $o(L) \leq o < o(U)$, dann ...
17: $R \leftarrow U$ ▷ ... „rechte“ Matrix erhält U.
18: **sonst** ▷ Wenn $o(U) \leq o \leq o(R)$, dann ...
19: $L \leftarrow U$ ▷ ... „linke“ Matrix erhält U.
20: **ende wenn**
21: **ende solange**
22: **für alle** $i \in \{0, \ldots, |M|\}$ **führe aus** ▷ Erstelle Verteilungs*funktion*en
23: **für** $j \leftarrow 1, \ldots, |M|$ **führe aus** ▷ Erstelle Verteilungs*funktion* i
24: $U_{i,j} \leftarrow U_{i,j-1} + U_{i,j}$ ▷ Setze Funktionswert
25: **ende für**
26: **ende für**
27: **gib zurück** U
28: **ende funktion**

B.4.3 Generieren von Maschinenfolgen

Algorithmus B.9 Erzeugen einer Maschinenfolge

Eingabe:
eine Menge M von Maschinen,
eine zu M passende Maschinenübergangsmatrix U
Ausgabe: eine Maschinenfolge S

```
1:  funktion ErzeugeMaschinenFolge(U, M)
2:      k ← 0                          ▷ Initialisiere „von Maschine“ mit „Umfeld“.
3:      wiederhole                     ▷ Erzeuge Maschinenfolge i. e. S.
4:          r ← rand[0; 1)             ▷ Erzeuge Zufallszahl.
5:          l ← 0                      ▷ Initialisiere „zu Maschine“ mit „Umfeld“.
6:          solange U_{k,l} < r führe aus   ▷ Ermittle „zu Maschine“.
7:              l ← l + 1              ▷ Inkrementiere „zu Maschine“.
8:          ende solange
9:          rufe HängeAn(S, l)         ▷ Hänge „zu Maschine“ an Folge S.
10:         k ← l                      ▷ Setze „von Maschine“ auf „zu Maschine“.
11:     bis k = |M|                    ▷ ..., bis „Umfeld“ erreicht.
12:     rufe HängeAb(S,rufe GibLetztesElement(S))   ▷ Entferne „Umfeld“.
13:     gib zurück S
14: ende funktion
```

Algorithmus B.9 startet mit der „von Maschine“ k in Zeile 0, dem „Umfeld“, und ermittelt aufgrund einer Pseudozufallszahl aus dem Intervall $[0; 1)$ und der Verteilungsfunktion aus Zeile k zufällig die „zu Maschine“ l. Diese Maschine l hängt der Algorithmus an die Maschinenfolge S an. Die „zu Maschine“ l wird zur neuen „von Maschine“ k, und die fußgesteuerte Schleife beginnt von neuem. Die Schleife bricht ab, wenn als „zu Maschine“ l bzw. k die Spalte 0, das „Umfeld“, erreicht worden ist. Das letzte Element, das „Umfeld“, wird aus der Maschinenfolge S entfernt, und die Maschinenfolge S wird zurückgegeben.[11]

[11] Erzeugt Algorithmus B.9 Maschinenfolgen in großer Anzahl, gleicht eine über den Maschinenfolgen ermittelte Maschinenübergangsmatrix der ursprünglichen Matrix U aufgrund des Gesetzes der großen Zahlen und weist auch den gleichen gewünschten Organisationsgrad o auf.

B.4.4 Generieren von Zufallszahlen

Zum Erzeugen von im Standardintervall $[0;1)$ gleichverteilten Zufallszahlen ist ausschließlich der „Mersenne Twister“ [MN98] verwendet worden. Solche Zufallszahlen sind Voraussetzung für die Generierung von Zufallszahlen, die anderen Verteilungen entsprechen.

Um stochastisch Dauern zu generieren, die zwischen der Ankunft zweier Fertigungsaufträge liegen, ist die Exponentialverteilungsfunktion

$$F(x) = \begin{cases} 0 & |x < 0, \\ 1 - e^{-\lambda x} & |x \geq 0 \end{cases} \tag{B.7}$$

zu ihrer Umkehrfunktion

$$F^{-1}(y) = t = -\frac{\ln(1-y)}{\lambda}, \text{ für } 0 \leq y < 1, \tag{B.8}$$

invertiert worden, welche die $[0;1)$-gleichverteilte Zufallszahl y in mit dem Parameter λ exponentialverteilte Zufallszahlen transformiert. Exponentialverteilte Zufallszahlen für stochastische Zwischenankunftsdauern zu verwenden, empfehlen unter anderen [PSS95], [KG95] und [BSMM99]. Parameter für das Generieren exponentialverteilter Zufallszahlen sind – alternativ – der Erwartungswert μ, die Standardabweichung σ oder der Parameter λ der Exponentialverteilung, die mittels der Gleichungen

$$\mu = \frac{1}{\lambda} \quad \text{und} \quad \sigma^2 = \frac{1}{\lambda^2} \tag{B.9}$$

paarweise ineinander umgerechnet werden können.

Dem stochastischen Generieren von Bearbeitungsdauern für Arbeitsplanpositionen und Arbeitsgänge liegt nicht die Exponentialverteilung zugrunde, weil bei ihr beliebig kurze Bearbeitungszeiten am häufigsten auftreten, was nicht realistisch ist, sondern die linkssteile bzw. rechtsschiefe logarithmische Normalverteilung mit der Dichte

$$f(t) = \begin{cases} 0 & |t \leq 0, \\ \dfrac{1}{t\sigma_L\sqrt{2\pi}} \exp\left(-\dfrac{(\ln t - \mu_L)^2}{2\sigma_L^2}\right) & |t > 0 \end{cases} \tag{B.10}$$

und der Verteilungsfunktion

$$F(x) = \begin{cases} 0 & |x \leq 0, \\ \dfrac{1}{\sigma_L\sqrt{2\pi}} \displaystyle\int_{-\infty}^{\ln x} \exp\left(-\frac{(t-\mu_L)^2}{2\sigma_L^2}\right) dt & |x > 0 \end{cases}, \tag{B.11}$$

bei der die Wahrscheinlichkeit für kurze Bearbeitungsdauern wieder fällt, was eher der Realität entspricht (siehe auch [Völ03, S. 114]).

Um logarithmisch normalverteilte Zufallszahlen zu erzeugen, ist zuvörderst das Erzeugen normalverteilter Zufallszahlen vonnöten: Die Box-Muller-Methode transformiert zwei im Standardintervall $[0;1)$ gleichverteilte Zufallsvariablen u_1 und u_2 mittels der Berechnungsvorschriften [EHP00, S. 150]

$$n_1 = \sqrt{-2\log u_1}\sin(2\pi u_2) \tag{B.12}$$

$$n_2 = \sqrt{-2\log u_1}\cos(2\pi u_2) \tag{B.13}$$

in zwei unabhängige standardnormalverteilte Zufallsvariablen n_1 und n_2.[12] Im Anschluss transformiert die Formel [EHP00, S. 133]

$$l = e^{\mu_L + \sigma_L n} \tag{B.14}$$

jede standardnormalverteilte Zufallsvariable n in eine logarithmisch normalverteilte Zufallsvariable l mit dem Erwartungswert [BSMM99]

$$\mu = e_L^{\mu} + \frac{\sigma_L^2}{2} \tag{B.15}$$

und der Standardabweichung[13] [LK00]

$$\sigma = \sqrt{\left(e^{\sigma_L^2} - 1\right) e^{2\mu_L + \sigma_L^2}}. \tag{B.16}$$

Zur Generierung logarithmisch normalverteilter Zufallszahlen müssten nun die Parameter μ_L und σ_L der logarithmischen Normalverteilung vorgeben werden. Diese Parameter erscheinen dem Anwender vermutlich nicht sofort plausibel. Aus diesem Grunde sind die Gleichungen B.15 und B.16 in die Formeln

$$\mu_L = \ln \frac{\mu^2}{\sqrt{\mu^2 + \sigma^2}} \tag{B.17}$$

[12] [BFS87, S. 223 f.] fällen folgendes Urteil über die Box-Muller-Methode: „As an approximation to a pair of *independent* variates, $(Y_i; Z_i)$ [hier $(n_1; n_2)$] is terrible. As an approximation to *normal* variates, Y_i and Z_i [hier n_1 and n_2] are poor.“ Aufgrund dieses Urteils ist im Verlauf der Experimente auf die „Polar Method“ von Box, Muller und Marsaglia ausgewichen worden, wie sie [Knu02] in Abschnitt 3.4.1, Unterabschnitt C, Algorithmus P beschreibt. Sie generiert zwei unabhängige Werte, indem sie lediglich einen Logarithmus und eine Quadratwurzel errechnet, und ist somit zusätzlich zu ihrer Güte auch noch effizient.

[13] [BSMM99] geben die Formel zur Berechnung der Standardabweichung mit $\sigma = \sqrt{e^{\sigma_L^2 - 1} e^{2\mu_L + \sigma_L^2}}$ falsch an. [LK00] hingegen stellen die Formel korrekt dar, so, wie sie auch hier angegeben worden ist.

und

$$\sigma_L = \sqrt{\ln\left(\frac{\sigma^2}{\mu^2} + 1\right)} \tag{B.18}$$

umgeformt worden (siehe Anhang B.4.5), die es erlauben, den Erwartungswert μ und die Standardabweichung σ für die logarithmisch normalverteilten Zufallszahlen direkt vorzugeben.

B.4.5 Lognormalverteilung: Ermitteln von μ_L und σ_L

Gegeben sind die Berechnungsvorschriften B.15 für μ und B.16 für σ. Das Ziel besteht darin, die Berechnungsvorschriften so umzuformen, dass sowohl μ_L als auch σ_L jeweils explizit aus μ und σ berechnet werden können.

Zu Beginn wird Gleichung B.15 nach μ_L umgestellt:

$$\begin{aligned}
\mu &= e^{\mu_L + \frac{1}{2}\sigma_L^2} \Big| \ln && \text{(B.19)}\\
\ln \mu &= \mu_L + \frac{1}{2}\sigma_L^2 && \text{(B.20)}\\
\mu_L &= \ln \mu - \frac{1}{2}\sigma_L^2. && \text{(B.21)}
\end{aligned}$$

Im Anschluß wird Gleichung B.16 nach μ_L umgestellt:

$$\begin{aligned}
\sigma^2 &= \left(e^{\sigma_L^2} - 1\right) e^{2\mu_L + \sigma_L^2} && \text{(B.22)}\\
e^{2\mu_L + \sigma_L^2} &= \frac{\sigma^2}{e^{\sigma_L^2} - 1} \Big| \ln && \text{(B.23)}\\
2\mu_L + \sigma_L^2 &= \ln\left(\frac{\sigma^2}{e^{\sigma_L^2} - 1}\right) && \text{(B.24)}\\
2\mu_L &= \ln\left(\frac{\sigma^2}{e^{\sigma_L^2} - 1}\right) - \sigma_L^2 && \text{(B.25)}\\
\mu_L &= \frac{\ln\left(\frac{\sigma^2}{e^{\sigma_L^2} - 1}\right) - \sigma_L^2}{2} && \text{(B.26)}
\end{aligned}$$

Nun werden die Gleichungen B.21 und B.26 gleichgesetzt, und es wird nach

σ_L umgeformt:

$$\ln\mu - \frac{1}{2}\sigma_L^2 = \frac{\ln\left(\frac{\sigma^2}{e^{\sigma_L^2}-1}\right) - \sigma_L^2}{2}\Bigg| \cdot 2 \tag{B.27}$$

$$2\ln\mu - \sigma_L^2 = \ln\left(\frac{\sigma^2}{e^{\sigma_L^2}-1}\right) - \sigma_L^2\Bigg| + \sigma_L^2 \tag{B.28}$$

$$2\ln\mu = \ln\left(\frac{\sigma^2}{e^{\sigma_L^2}-1}\right) \tag{B.29}$$

$$\ln\mu^2 = \ln\left(\frac{\sigma^2}{e^{\sigma_L^2}-1}\right)\Bigg| e^{\cdot} \tag{B.30}$$

$$\mu^2 = \frac{\sigma^2}{e^{\sigma_L^2}-1}\Bigg| \left(e^{\sigma_L^2}-1\right) \tag{B.31}$$

$$e^{\sigma_L^2} - 1 = \frac{\sigma^2}{\mu^2}\Bigg| + 1 \tag{B.32}$$

$$e^{\sigma_L^2} = \frac{\sigma^2}{\mu^2} + 1\Bigg| \ln \tag{B.33}$$

$$\sigma_L^2 = \ln\left(\frac{\sigma^2}{\mu^2}+1\right)\Bigg| \sqrt{\cdot} \tag{B.34}$$

$$\sigma_L = \underline{\sqrt{\ln\left(\frac{\sigma^2}{\mu^2}+1\right)}}. \tag{B.35}$$

Gleichung B.35 ist das erste Endergebnis des Umstellens. Im letzten Schritt wird die Gleichung B.34 in die Gleichung B.15 eingesetzt, und es wird nach

μ_L umgestellt:

$$\mu = e^{\mu_L + \frac{1}{2}\ln\ \frac{\sigma^2}{\mu^2}+1} \Big| \ln \tag{B.36}$$

$$\ln\mu = \mu_L + \frac{1}{2}\ln\left(\frac{\sigma^2}{\mu^2}+1\right) \tag{B.37}$$

$$\mu_L = \ln\mu - \frac{1}{2}\ln\left(\frac{\sigma^2}{\mu^2}+1\right) \tag{B.38}$$

$$\mu_L = \ln\mu - \ln\left(\frac{\sigma^2}{\mu^2}+1\right)^{\frac{1}{2}} \tag{B.39}$$

$$\mu_L = \ln\mu - \ln\sqrt{\frac{\sigma^2}{\mu^2}+1} \tag{B.40}$$

$$\mu_L = \ln\frac{\mu}{\sqrt{\frac{\sigma^2}{\mu^2}+1}} \tag{B.41}$$

$$\mu_L = \ln\frac{\mu}{\sqrt{\frac{\mu^2+\sigma^2}{\mu^2}}} \tag{B.42}$$

$$\mu_L = \underline{\ln\frac{\mu^2}{\sqrt{\mu^2+\sigma^2}}}. \tag{B.43}$$

Gleichung B.43 ist das zweite und letzte Endergebnis des Umstellens. Zur Kontrolle wird die Gleichung B.34 in die Gleichung B.16 eingesetzt, und es

wird nach μ_L umgestellt:

$$\sigma^2 = \left(e^{\ln\ \frac{\sigma^2}{\mu^2}+1}\ -1\right) e^{2\mu_L+\ln\ \frac{\sigma^2}{\mu^2}+1} \tag{B.44}$$

$$\sigma^2 = \left(\left(\frac{\sigma^2}{\mu^2}+1\right)-1\right) e^{2\mu_L} e^{\ln\ \frac{\sigma^2}{\mu^2}+1} \tag{B.45}$$

$$\sigma^2 = \frac{\sigma^2}{\mu^2} e^{2\mu_L} \left(\frac{\sigma^2}{\mu^2}+1\right) \tag{B.46}$$

$$\mu^2 = e^{2\mu_L} \left(\frac{\sigma^2}{\mu^2}+1\right) \tag{B.47}$$

$$\frac{\mu^2}{\left(\frac{\sigma^2}{\mu^2}+1\right)} = e^{2\mu_L} \tag{B.48}$$

$$e^{2\mu_L} = \frac{\mu^2}{\frac{\sigma^2+\mu^2}{\mu^2}} \tag{B.49}$$

$$e^{2\mu_L} = \frac{\mu^4}{\sigma^2+\mu^2} \tag{B.50}$$

$$2\mu_L = \ln \frac{\mu^4}{\sigma^2+\mu^2} \tag{B.51}$$

$$\mu_L = \frac{1}{2} \ln \frac{\mu^4}{\sigma^2+\mu^2} \tag{B.52}$$

$$\mu_L = \ln \left(\frac{\mu^4}{\sigma^2+\mu^2}\right)^{\frac{1}{2}} \tag{B.53}$$

$$\mu_L = \ln \sqrt{\frac{\mu^4}{\sigma^2+\mu^2}} \tag{B.54}$$

$$\mu_L = \ln \frac{\mu^2}{\sqrt{\sigma^2+\mu^2}}. \tag{B.55}$$

B.4.6 Generieren von Fertigungsaufträgen

Algorithmus B.10 zum Erzeugen von Fertigungsaufträgen generiert lediglich die Maschinenübergangsmatrix U, bestimmt den Ankunftszeitpunkt t für den nächsten Auftrag mittels Formel B.8 und ruft mehrfach Algorithmus B.11 auf, der einen einzelnen Auftrag a_i generiert und somit Modus zwei verkörpert. Algorithmus B.10 zum Generieren der Auftragsmenge A terminiert, wenn die Anzahl z_{max} zu generierender Aufträge erreicht worden oder der Zeitraum $[t_0; t_0 + d_{max})$ vergangen ist, in dem Aufträge generiert werden sollen.

Algorithmus B.10 Aufträge Erzeugen

Eingabe:

Maschinen $M = \{M_1, \ldots, M_n\}$,
Organisationsgrad o mit $0 \leq o \leq 1$,
Zeitpunkt t_0, ab dem Aufträge generiert werden sollen,
mittlere Zwischenankunftsdauer $\overline{d}_a$ der Aufträge,
maximale Anzahl z_{max} an Aufträgen, die generiert werden sollen,
maximale Dauer d_{max}, für die Aufträge generiert werden sollen,
mittlere Bearbeitungsdauern $B = \{\overline{b}_1, \ldots, \overline{b}_n\}$ auf den Maschinen M,
Standardabweichungen $S = \{s_1, \ldots, s_n\}$ der mittleren Bearbeitungsdauern B auf den Maschinen M

Ausgabe:

Aufträge $A = \{a_1, \ldots, a_m\}$ mit $m \leq z_{max}$,
Arbeitsgänge $G_i = \{g_{i1}, \ldots, g_{ik_i}\}$ des Auftrags a_i für $i = 1, \ldots, m$,
Dauern $B_i = \{b_{i1}, \ldots, b_{ik_i}\}$ der Arbeitsgänge G_i für $i = 1, \ldots, m$,
Startzeitpunkte $t_{i1}, \ldots, t_{ik_i}$ der Arbeitsgänge G_i für $i = 1, \ldots, m$,
Folge von Maschinen $S_i = (M_{i1}, \ldots, M_{ik_i})$ mit $\bigcup_{j=1}^{k_i}\{M_{ij}\} \subset M$, auf denen die Arbeitsgänge G_i zu bearbeiten sind, für $i = 1, \ldots, m$

1: **funktion** ErzeugeAufträge($M, o, t_0, \overline{d}_a, z_{max}, d_{max}, B, S$)
2: $U \leftarrow$**rufe** ErzeugeMaschinenÜbergangsMatrix(M, o)
3: $A \leftarrow \emptyset$
4: $t \leftarrow t_0 - \ln(1 - \text{rand}[0;1))\overline{d}_a$ ▷ erster Ankunftszeitpunkt
5: $i \leftarrow 0$
6: **solange** $(|A| < z_{max}) \wedge (t < (t_0 + d_{max}))$, **führe aus**
7: $i \leftarrow i + 1$
8: $a_i \leftarrow$**rufe** ErzeugeNächstenAuftrag(i, M, U, t, B, S)
9: $A \leftarrow A \cup \{a_i\}$
10: $t \leftarrow t - \ln(1 - \text{rand}[0;1))\overline{d}_a$ ▷ nächster Ankunftszeitpunkt
11: **ende solange**
12: **gib zurück** A
13: **ende funktion**

Algorithmus B.11 zum Erzeugen eines einzelnen Auftrages a_i ruft zu Beginn Algorithmus B.9 auf, der eine Maschinenfolge S_i generiert, und durchläuft dann diese Maschinenfolge. Während eines Durchlaufes erzeugt er einen Arbeitsgang a_{ij}, setzt seinen Startzeitpunkt t_{ij}, bestimmt die mittlere Bearbeitungsdauer μ und ihre Standardabweichung σ anhand der Position der Maschine M_{ij}, auf welcher der Arbeitsgang a_{ij} bearbeitet werden soll, und errechnet maschinenindividuell eine $[\mu; \sigma]$-lognormalverteilte Zufallszahl für die Bearbeitungsdauer b_{ij} des Arbeitsganges a_{ij}. Den Startzeitpunkt t_{ij+1} für den nächsten Arbeitsgang a_{ij+1} errechnet er, indem er zum Startzeitpunkt t_{ij} des aktuellen Arbeitsganges a_{ij} die Bearbeitungsdauer b_{ij} dieses Arbeitsganges addiert. Die Termine für die Aufträge und Arbeitsgänge werden also mittels einfacher, linearer Vorwärtsterminierung ermittelt (hier zur Vereinfachung ohne maschinenspezifische Puffer- bzw. Übergangszeit).[14]

Algorithmus B.11 generiert also einen Fertigungsauftrag bzw. einen individuellen Arbeitsplan als Vorlage für einen Fertigungsauftrag.[15] Der gewünschte Fertigstellungstermin des Auftrages entspricht dem Endzeitpunkt seines letzten Arbeitsganges.

Erfolgt der Aufruf des Algorithmus B.11 zum Erzeugen eines Auftrages nicht durch Algorithmus B.10 sondern im zweiten Modus von extern, so kann besagter Auftragsstrom generiert werden, der quasi nie versiegt.

B.5 Ablauf einer Simulationsstudie

Den komponentenübergreifenden Ablauf einer Simulationsstudie stellt Abbildung B.4 vereinfacht als Kollaborationsdiagramm[16] dar:

Die Nachrichten 1 bis 10 dienen dem Erstellen der statischen Komponenten des Simulationsprogrammes: Die Simulation erstellt den Experimentator und initialisiert die Experimente (1). Der Experimentator seinerseits erstellt den Faktorbaum (2) und die Variablen[17] (3). Die Simulation erstellt nun die

[14] In Ermangelung einer Repräsentation für Zeiger und objektorientierte Datenstrukturen in mathematischem Pseudoquelltext werden die Algorithmen hier prozedural angegeben. Die vorliegende Software verwendet natürlich Objektorientierung als Programmierparadigma, so dass z. B. Arbeitsgänge Aggregate des Fertigungsauftrags sind und jeder Arbeitsgang mit der Maschine assoziiert ist, auf der er bearbeitet werden soll.

[15] Auftragsbezogene Arbeitspläne kommen beispielsweise im Werkzeugbau vor, in dem im Normalfall kein Auftrag dem anderen gleicht, da Werkzeuge immer individuell gefertigt werden.

[16] Die partielle Axialsymmetrie im unteren Teil des Kollaborationsdiagrammes dient lediglich einem ansprechenden Aussehen. Sie ist inhaltlich nicht begründet. Bis auf die Auftragsquelle und die Auftragssenke entsprechen sich einander gegenüberliegende Diagrammelemente ergo nicht.

[17] Bei den Variablen, die an mehreren Stellen in Abbildung B.4 vorkommen, handelt es

Algorithmus B.11 Nächsten Auftrag Erzeugen

Eingabe:

Index i des zu erzeugenden Auftrages a_i,
Maschinen $M = \{M_1, \ldots, M_n\}$,
Maschinenübergangsmatrix U für die Maschinen M,
Zeitpunkt t, ab dem der erste Arbeitsgang des Auftrags a_i starten soll,
mittlere Bearbeitungsdauern $B = \{\overline{b}_1, \ldots, \overline{b}_n\}$ auf den Maschinen M,
Standardabweichungen $S = \{s_1, \ldots, s_n\}$ der mittleren Bearbeitungsdauern B auf den Maschinen M

Ausgabe:

nächster Auftrag a_i,
Maschinenfolge $S_i = (M_{i1}, \ldots, M_{ik_i})$ mit $\bigcup_{j=1}^{k_i}\{M_{ij}\} \subset M$ für Auftrag a_i,
Arbeitsgänge $G_i = \{g_{i1}, \ldots, g_{ik_i}\}$ des Auftrags a_i,
Startzeitpunkte $t_{i2}, \ldots, t_{ik_i}$ der Arbeitsgänge G_i,
Dauern $B_i = \{b_{i1}, \ldots, b_{ik_i}\}$ der Arbeitsgänge G_i

```
1:  funktion ErzeugeNächstenAuftrag(i, M, U, t, B, S)
2:      a_i ← erzeuge Auftrag
3:      S_i ←rufe ErzeugeMaschinenFolge(U, M)
4:      G_i ← ∅
5:      B_i ← ∅
6:      für j ← 1, ..., |S_i| führe aus
7:          g_ij ← erzeuge Arbeitsgang
8:          t_ij ← t                                  ▷ setze Startzeitpunkt
9:          p ← bestimme Position der Maschine M_ij in M = (M_1, ..., M_n)
10:         μ ← p-te mittlere Bearbeitungsdauer in B
11:         σ ← p-te Standardabweichung der Bearbeitungsdauer in S
12:         b_ij ← erzeuge [μ; σ]-lognormalverteilte Zufallszahl
13:         t ← t + b_ij                  ▷ berechne nächsten Startzeitpunkt
14:         B_i ← B_i ∧ {b_ij}
15:         G_i ← G_i ∧ {g_ij}
16:     ende für
17:     gib zurück a_i
18: ende funktion
```

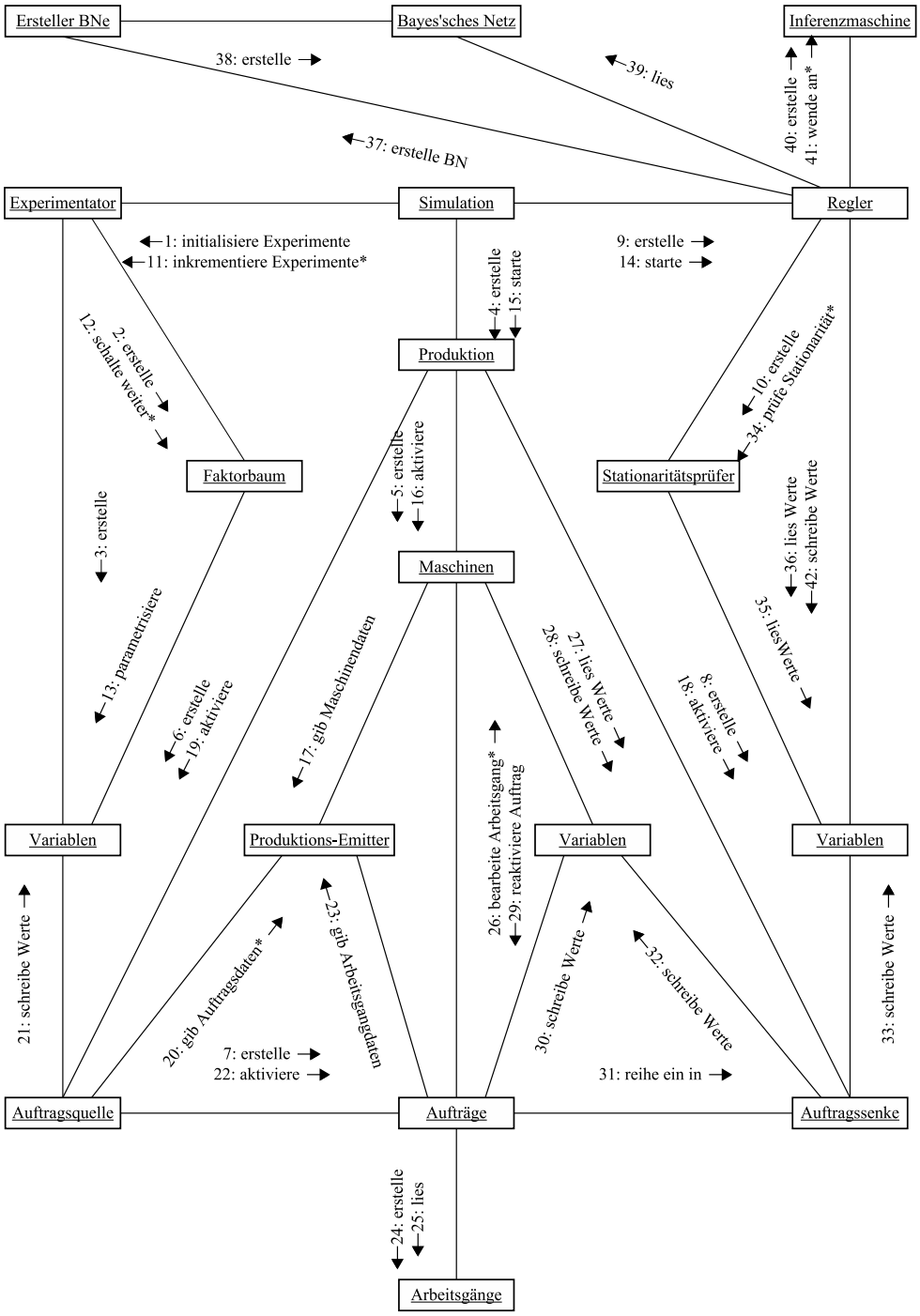

Abbildung B.4: Ablauf einer Simulationsstudie in Form eines UML-Kollaborationsdiagrammes

Produktion (4), welche wiederum eine Datenstruktur für die Maschinen (5) sowie die Auftragsquelle (6) erstellt. Die Auftragsquelle erzeugt nunmehr eine Datenstruktur für die Aufträge (7). Im Anschluss erstellt die Produktion die Auftragssenke (8). Im Aufrufkellerspeicher an der Reihe, erstellt die Simulation den Regler (9), welcher seinerseits den Stationaritätsprüfer instanziiert (10). Die Simulation verfügt nun über die für ihren Ablauf notwendigen statischen Komponenten und verfährt wie folgt weiter:

Solange noch weitere Experimente anstehen, inkrementiert die Simulation die Experimente (11), woraufhin der Exprimentator den Faktorbaum „weiterschaltet“ (12). Der Faktorbaum seinerseits parametrisiert nun die Variablen (13) bezüglich ihrer Relevanz und bezüglich der für sie angestrebten Ausprägung, sofern es sich bei ihnen um Regel- bzw. Führungsgrößen handelt. Wie das Weiterschalten des Faktorbaumes und das Parametrisieren anderer Komponenten durch ihn vonstatten gehen, beschreibt Abschnitt 6.6.2 en detail. So, wie er die Variablen parametrisiert, parametrisiert der Faktorbaum auch andere statische Komponenten, deren Parameter sich im Laufe der Simulationsstudie bzw. von Experiment zu Experiment ändern. Zu diesen Komponenten gehören der Produktions-Emitter, der Regler, der Ersteller BNe, das Bayes'sche Netz an sich sowie die Inferenzmaschine. Das Parametrisieren dieser Komponenten stellt Abbildung B.4 nicht dar, um das Kollaborationsdiagramm einfach und übersichtlich zu halten. Der Faktorbaum parametrisiert den Regler beispielsweise mit dem Abtast- bzw. Reglungsintervall und den Produktionsemitter z. B. mit der (Verteilung der) mittleren Zwischenankunftszeit, mit der Anzahl der Maschinen und mit dem Organisationsgrad. Der Faktorbaum parametrisiert den Ersteller BNe unter anderem mit der Anzahl der Intervalle, in welche reellwertige Variablen zu diskretisieren sind, und mit den Gütemaßen und Algorithmen zum Erstellen BNe aus Daten.

Nachdem alle Komponenten parametrisiert worden sind, aktiviert bzw. startet die Simulation den Regler (14) und die Produktion (15). Die Produktion aktiviert die Maschinen (16), so dass sie auf das Eintreffen von Aufträgen bzw. Arbeitsgängen warten und sie beim Eintreffen nacheinander abarbeiten. Ihre spezifischen Daten erlangen die Maschinen vom Produktions-Emitter (17). Im Anschluss aktiviert die Produktion die Auftragssenke (18) und die Auftragsquelle (19) in dieser Reihenfolge, damit die Auftragsquelle nicht Aufträge emittiert, welche die Auftragssenke noch gar nicht absorbieren kann. Die Auftragsquelle fordert nun Auftragsdaten inklusive einer

sich um *dieselben*. Ihre mehrfache grafische Repräsentation ermöglicht eine „kreuzungsfreie“ Darstellung des Graphen des Kollaborationsdiagrammes und erhöht somit seine Übersichtlich- und Lesbarkeit.

Zwischenankunftszeit beim Produktions-Emitter an (20), verzögert um die Zwischenankunftzeit, schreibt ihre aktuelle Statistik in die Variablen (21) und aktiviert bzw. startet den Auftrag (22). Der Auftrag fordert die Daten zu seinen Arbeitsgängen beim Produktions-Emitter an (23) und erstellt auf Basis dieser Daten seine Arbeitsgänge (24). Der Produktions-Emitter verwendet zum Ermitteln der Arbeitsgangdaten den Organisationsgrad und eine auf ihm basierende Übergangsmatrix (siehe Abschnitt 6.5.1 und Algorithmen B.8 bis B.10). Alsdann liest der Auftrag seinen nächsten Arbeitsgang (25) und fordert die Maschinen auf, ihn zu bearbeiten (26). Die Maschinen lesen Werte für ihre Stellgrößen aus den Variablen (27) und bearbeiten die jeweils anstehenden Arbeitsgänge entsprechend dieser Stellwerte. Im hiesigen Falle handelt es sich bei den Stellgrößen um Variablen, die als Ausprägungen verschiedene Prioritätsregeln für die Maschinen annehmen. Hat eine Maschine einen Arbeitsgang bearbeitet, aktualisiert sie die Variablen (28) und reaktiviert den Auftrag (29), dessen Arbeitsgang sie fertiggestellt hat, auf dass er seinen nächsten Arbeitsgang auf der passenden Maschine bearbeiten lasse. Die Nachrichten 25 bis 29 werden für jeden Auftrag solange ausgetauscht und verarbeitet, bis alle Arbeitsgänge des Auftrags abgearbeitet worden sind. Danach aktualisiert der Auftrag die Variablen (30) und reiht sich in die Auftragssenke ein (31). Die Auftragssenke aktualisiert nun ihrerseits die Variablen (32 bzw. 33) und schleust den Auftrag aus dem System aus. Bezüglich des Sammelns von Daten während der Simulation sei angemerkt, dass die Auftragsquelle, die Auftragssenke, die Maschinen und die Aufträge Daten aufnehmen, z. T. akkumulieren und an die Variablen abgeben. Die fertiggestellten Aufträge geben darüber hinaus einen Teil ihrer Daten an die Auftragssenke ab, welche die Daten akkumuliert und wiederum an die Variablen abgibt.

Der eben geschilderte Ablauf der Simulation und im Speziellen die Reihenfolge des Abarbeitens der vor den Maschinen wartenden Aufträge bzw. Arbeitsgänge wird durch den Regler bestimmt, der wie folgt arbeitet: Nach jedem Abtast- bzw. Regelungsintervall veranlasst der Regler den Stationaritätsprüfer, zu prüfen, ob sich die simulierte Produktion bereits in einem eingeschwungenen Zustand befindet (34). Zu diesem Zwecke greift der Stationaritätsprüfer auf die Variablen und die in ihnen gespeicherten multivariaten Zeitreihen zu (35) und bestimmt unter Zuhilfenahme der Formel 6.2 ob der Zustand der Produktion stationär ist. Sobald dem so ist, gibt der Regler den Stellgrößen unter den Variablen über längere Zeiträume hinweg mehrere Prioritätsregeln fest vor (42), so dass die Maschinen die Fertigung nach diesen Regeln simulieren. Zudem tastet der Regler nach jedem Abtastintervall die Variablen ab (36), die von den anderen Komponenten aktualisiert worden sind, und ergänzt die multivariaten Zeitreihen der Variablen um die

abgetasteten Werte (42). Die multivariaten Zeitreihen dienen dem Erstellen des eigentlichen Reglers, des BNes. Ist der Zeitraum zum Erstellen der multivariaten Zeitreihen verronnen, greift der Regler auf die Zeitreihen der Variablen zu (36) und übergibt sie dem Ersteller BNe (37), auf dass er ein DBN aus ihnen erstelle, welches der Ersteller auch er- und bereitstellt (38). Der Regler verfügt nun über ein BN (39) und weist die Inferenzmachine an, sich zu erstellen (40). Nach jedem weiteren Reglungsintervall liest nun der Regler die aktuellen Werte der Variablen (36), wendet mit diesen Werten die Inferenzmachine an (41) und setzt die Stellgrößen unter den Variablen auf die ermittelten Werte (42). Die Inferenzmaschine schließt mittels „gemischten Schließens“ vom Zustand der simulierten Fertigung und von den für die Regelgrößen vorgegebenen Führungswerten auf Werte für die Stellgrößen (siehe Abschnitt 4.5.3).

B.6 Algorithmen zum Graphenzeichnen

Algorithmen zum Graphenzeichnen verfolgen z. B. diese drei, z. T. konkurrierenden Ziele [BETT98]:

1. Die Kanten eines Graphen sollten sich nicht kreuzen. Falls sie sich jedoch objektiv kreuzen müssen – z. B. bei Graphen, die nicht planar sind –, sollten sie sich so selten wie möglich kreuzen.

2. Die Kanten eines Graphen sollten durch Geraden repräsentiert werden und keine Knicke oder Rundungen aufweisen. Falls sie jedoch Knicke oder Rundungen aufweisen, sollten sie das so selten und so „gering“ wie möglich.

3. Die Fläche, welche ein gezeichneter Graph benötigt, sollte minimal sein und in ihren Proportionen einer gegebenen Fläche, z. B. einer Buch- oder Bildschirmseite entsprechen.

Ein Heuristik, welche diese Ziele für gerichtete, azyklische Graphen verfolgt, denen ja die Graphen BNe angehören, und diese Ziele verhältnismäßig gut erreicht, ist in das Softwaresystem zur Unterstützung der PPS mittels BNe implementiert worden. Die Knoten des Graphen werden so in horizontalen Schichten angeordnet, dass die Kanten immer von oben nach unten weisen, und die Knoten werden innerhalb dieser Schichten so sortiert und verschoben, dass sich die Kanten möglichst wenig kreuzen und möglichst gerade verlaufen. Die Breite des Graphen ist mit b Knoten beschränkt, und die Höhe ist akzeptabel. Abbildung B.5 stellt die geschichtete grafische Repräsentation eines DAGen dar.

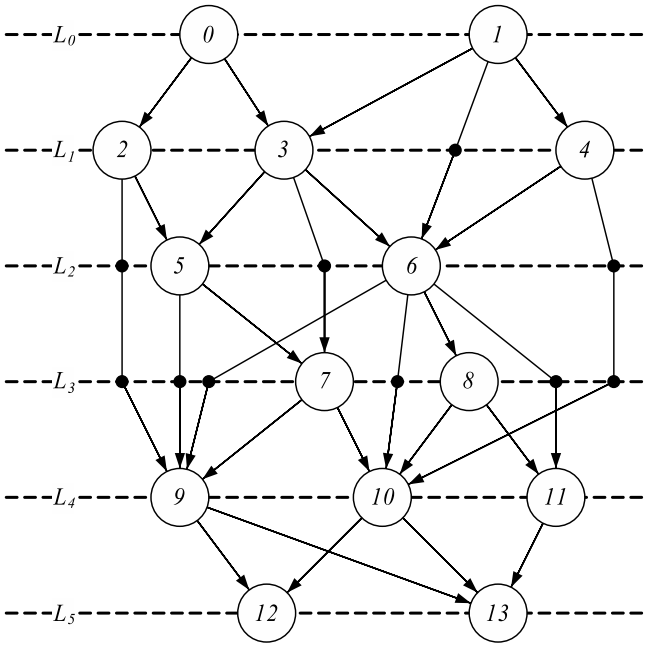

Abbildung B.5: Beispiel für die geschichtete grafische Repräsentation eines gerichteten, azyklischen Graphen eines BNes; angelehnt an [BETT98]

Zu Beginn der Heuristik werden die Knoten des Graphen des BNes topologisch sortiert, was z. B. mittels Algorithmus B.12 geschehen kann: Die Menge N der als nächstes einzusortierenden Knoten enthält zu Beginn all die Knoten, welche keine Eltern besitzen. Die Schlange S nimmt später die Knoten in topologischer Ordnung auf. Die Menge M ist leer. Sie dient als Hilfsmenge für spätere Mengenoperationen und muss immer dieselben Knoten aufweisen wie S. Algorithmus B.12 durchläuft eine Schleife für alle Knoten: Er wählt einen beliebigen Knoten v aus der Menge N der als nächstes einzusortierenden Knoten und entfernt diesen Knoten v aus der Menge N. Alsdann fügt er v sowohl an S an als auch M hinzu. Die letzte Anweisung in der Schleife ergänzt die als nächstes einzusortierenden Knoten N um die Kinder des Knotens v, deren Eltern bereits sämtlich in M bzw. S enthalten sind. Nach $|V|$ Durchläufen der Zählschleife enthält N auch keine (nicht einsortierten) Knoten mehr. Die Schlange S enthält am Ende des Algorithmus' die Knoten $v \in V$ topologisch sortiert, und der aufrufende Algorithmus erhält S zurück.

Ist eine topologische Sortierung vorgenommen worden, die bei einem gerichteten, azyklischen Graphen immer möglich ist, gibt es im DAG keine gerichtete Kante (u, v), deren Knoten v in der Sortierung S vor dem Knoten u liegt. Anders ausgedrückt, weisen sämtliche Kanten des DAG von einem

Algorithmus B.12 Topologisches Sortieren

Eingabe: Menge der Knoten V eines DAGen mit jeweiligen Eltern/Kindern
Ausgabe: die Knoten V in der Schlange S topologisch sortiert

```
 1: funktion SortiereTopologisch(V)
 2:     N ← {v ∈ V | Pa_v = ∅}                    ▷ als nächstes einzusortieren
 3:     S ←rufe ErzeugeSchlange()                  ▷ sortierte Knoten
 4:     M ← ∅                                      ▷ Hilfsmenge gleichen Inhaltes wie S
 5:     für i ← 0, ..., (|V| − 1) führe aus
 6:         v ←rufe WähleBeliebigesElementVon(N)
 7:         N ← N \ {v}
 8:         rufe HängeAn(S, v)                     ▷ Sortiere v in S ein
 9:         M ← M ∪ {v}                            ▷ Füge v auch Hilfsmenge S hinzu
10:         N ← N ∪ {u ∈ Ch_v | Pa_u ⊂ M}          ▷ Ergänze N (siehe Text)
11:     ende für
12:     gib zurück S
13: ende funktion
```

Knoten u, der weiter vorn in der Sortierung S liegt, zu einem Knoten v, der weiter hinten in der Sortierung S liegt.

Die topologische Numerierung ist der erste Schritt des Algorithmus B.13, der von Coffman und Graham [JG72] stammt. Er ist ursprünglich für die Lastverteilung auf Mehrprozessorrechnern entworfen worden. Er ordnet die Knoten so Schichten zu, dass keine Schicht breiter als b ist und die Kanten immer von Knoten in einer Schicht mit einem geringeren Index zu Knoten in einer Schicht mit einem höheren Index weisen (siehe auch Abbildung B.5). Nach der topologischen Numerierung im ersten Schritt initialisiert der Algorithmus die erste Schicht als leere Menge. Solange noch Knoten die sortierte Schlange S bevölkern, entnimmt der Algorithmus den ersten Knoten v und fügt ihn der aktuellen Schicht L_i hinzu, sofern die aktuelle Schicht noch nicht zu breit ist und die Eltern Pa_v des Knotens v sich bereits alle in vorherigen Schichten befinden. Trifft eine der letzten beiden Bedingungen nicht zu, so entsteht eine neue Schicht, die zur aktuellen wird und den Knoten v enthält. Hat Algorithmus B.13 alle Knoten Schichten zugeordnet, gibt er die in L aufgereihten Schichten zurück und terminiert.

Die maximale Breite b ist im Original des Algorithmus' die Anzahl der zur Verfügung stehenden Prozessoren gewesen. Der Einfachheit halber ist in Algorithmus B.13 die maximale Breite b einer Schicht in „Anzahl Knoten“ angegeben worden. Die Software berücksichtigt in der Realität hingegen den horizontal zur Verfügung stehenden Platz und die Breite der beschrifteten Ellipsen, welche die Knoten repräsentieren (siehe Abbildung

Algorithmus B.13 Schichten der Knoten

Eingabe: Menge der Knoten V eines DAGen mit jeweiligen Eltern/Kindern, maximale Breite $b > 0$ einer Schicht

Ausgabe: Schichten L mit den Knoten V, von denen keine breiter als b ist

```
1:  funktion SchichteKnoten(V, b)
2:      S ←rufe SortiereTopologisch(V)
3:      L ←rufe ErzeugeSchlange()                    ▷ ... für die Schichten
4:      i ← 0                                        ▷ Initialisiere Schichtindex
5:      L_i ← ∅                                      ▷ erste Schicht leer
6:      rufe HängeAn(L, L_i)                         ▷ Hänge erste Schicht an L an
7:      solange rufe GibAnzahl(S)> 0, führe aus
8:          v ←rufe EntnimmErstenAus(S)
9:          wenn (|L_i| < b) ∧ (Pa_v ⊂ ⋃_{j←0}^{k−1} L_j), dann      ▷ (siehe Text)
10:             L_i ← L_i ∪ {v}                      ▷ Füge v aktueller Schicht hinzu
11:         sonst
12:             i ← i + 1                            ▷ Inkrementiere Schichtindex i
13:             L_i ← {v}                            ▷ Initialisiere neue Schicht mit v
14:             rufe HängeAn(L, L_i)                 ▷ Hänge neue Schicht an L an
15:         ende wenn
16:     ende solange
17:     gib zurück L
18: ende funktion
```

6.5), wenn sie entscheidet, ob sie eine Schicht „umbricht“ oder einen Knoten noch der aktuellen Schicht hinzufügt. Dass der Algorithmus eine maximale Breite einhält, hat den Vorteil, dass der Benutzer nur vertikal scrollen muss, wenn der Graph mehr Platz als eine Bildschirmseite beansprucht. Der Coffman–Graham-Algorithmus erzeugt immer weniger Schichten als das Doppelte seiner minimalen Höhe [NTB05]; der Graph wird also nicht zu hoch.

Sind die Knoten den Schichten zugeordnet worden, werden Hilfsknoten in diejenigen Kanten eingefügt, welche Schichten überspringen, und zwar pro übersprungener Schicht ein Knoten, der dann der übersprungenen Schicht zugeordnet wird. In Abbildung B.5 repräsentieren die kleinen schwarzen Kreise die Hilfsknoten. Im Anschluss erfolgt das Umordnen der Knoten in den Schichten, so dass sich möglichst wenige Kanten kreuzen. Zu diesem Zwecke wird ein „Schicht-um-Schicht“-Durchlauf durchgeführt, der sukzessive jeweils zwei aufeinanderfolgende Schichten betrachtet, in einer die Knoten in ihrer Reihenfolge fixiert und in der anderen die Knoten so anordnet, dass sich zwischen den Schichten möglichst wenige Kanten kreuzen.[BJM04] Die verwendeten Verfahren heißen Barycenter- und Medianmethode, wie sie auch [LS02] und [GKNV93] verwenden. Obwohl interessant, führte ihre Darstellung an dieser Stelle zu weit. Sind die Knoten in den Schichten sortiert worden, so werden sie abschließend so horizontal positioniert, dass die jeweils zwei den Hilfsknoten inzidenten Kanten möglichst auf einer Geraden liegen[18], und die Hilfsknoten werden entfernt. – Sollte die Repräsentation des Graphen nun noch nicht den Ansprüchen des Betrachters genügen, kann er selbst die Knoten in der grafischen Schnittstelle zum Benutzer nach Gutdünken verschieben.

B.7 Komponente zum Management von Experimenten

B.7.1 Struktur und Erstellen eines Faktorbaumes

Abbildung B.6 stellt die Faktor- und Werteklassen der Komponente zum Management von Experimenten sowie den Zusammenhang zwischen diesen Klassen in Form eine UML-Diagrammes dar. Besonderes Augenmerk liegt auf dem abstrakten Faktorcontainer und seinen Subklassen, den parallelen und den alternativen Faktoren. Der Faktorcontainer beinhaltet untergeordnete Faktoren. Die parallelen Faktoren sind alle (gleichzeitig, parallel) zum

[18]Dieser Schritt ist in der Schnittstelle zum Benutzer nicht implementiert worden, da sein Nutzen den Aufwand für die für ihn empfohlene lineare Optimierung [BETT98] nicht rechtfertigt.

Parametrisieren des Softwaresystems für ein Experiment vonnöten. Von den alternativen Faktoren parametrisiert hingegen nur genau einer das Softwaresystem für ein Experiment. Untergeordnete Faktoren werden wie Werte, sogenannte Faktorwerte, behandelt, die ihrerseits einen Faktor halten. Für alle elementaren Datentypen sowie für Listen liegen Faktoren und Werte vor.

Ein Faktorbaum wird in XML definiert. Das XML-Format für Faktorbäume beschreibt eine eigens entworfene DTD. Der Aufbau eines Faktorbaumes geht wie folgt vonstatten: Der bereits in Abschnitt 6.4.1 erwähnte DOM-Parser liest das in einer Textdatei vorliegende XML in ein DOM ein. Für jedes XML-Element des DOMs existiert eine eigene Erzeugerklasse[19]. Die Erzeugerklasse erstellt aufgrund des XML-Elementes eine Instanz der passenden Faktorklasse und befüllt die so erstellte Faktorinstanz anhand des XML-Elementes mit Werten bzw. einem Wertebereich. Handelt es sich beim aktuellen Faktor um einen Faktorcontainer, so ruft die Erzeugerklasse für die untergeordneten XML-Elemente passende Erzeugerklassen, welche aufgrund der untergeordneten XML-Elemente weitere Faktorinstanzen erstellen. Die ursprüngliche Erzeugerklasse ordnet diese Faktorinstanzen dem Faktorcontainer zu. Nun setzen die Erzeugerklassen die Tiefensuche im DOM rekursiv fort, so dass der Faktorbaum entsteht. Abbildung 6.8 stellt ein Exemplar eines Faktorbaumes dar.

B.7.2 Iterieren über einem Faktorbaum

Das Durchlaufen des Faktorbaumes geht wie folgt vonstatten: Von der Wurzel des Faktorbaumes beginnend, initialisiert sich jeder Faktor selbst, indem er eine interne Referenz auf seinen ersten Wert bzw. seinen ersten Subfaktor setzt und gegebenenfalls seine Subfaktoren veranlasst, sich ebenfalls zu initialisieren. Anschließend erfolgt, wiederum von der Wurzel des Faktorbaumes beginnend, das Inkrementieren des Faktorbaumes: Jeder Faktorcontainer bittet nacheinander jeden ihm untergeordneten Faktor, sich zu inkrementieren, bis ihm einer Erfolg meldet oder ihm kein weiterer untergeordneter Faktor mehr zur Verfügung steht. Kann sich ein Faktor nicht inkrementieren, so initialisiert er sich und meldet seinem übergeordneten Faktor Misserfolg. Gelangt der Misserfolg bis zur Wurzel des Faktorbaumes zurück, so ist der Faktorbaum vollständig durchlaufen worden.

Wie Algorithmus B.2 erinnert das Durchlaufen des Faktorbaumes an das Weiterschalten des Zählwerkes eines Kassettenspielers, wohingegen der Fak-

[19]Die Erzeugerklassen bilden eine zu den Faktor- und Werteklassen parallele Klassenhierarchie. Da die Klassenhierarchie der Erzeugerklassen mit der Klassenhierarchie der Faktor- und Werteklassen deckungsgleich ist, wird auf eine grafische Darstellung der Erzeugerklassenhierarchie an dieser Stelle verzichtet.

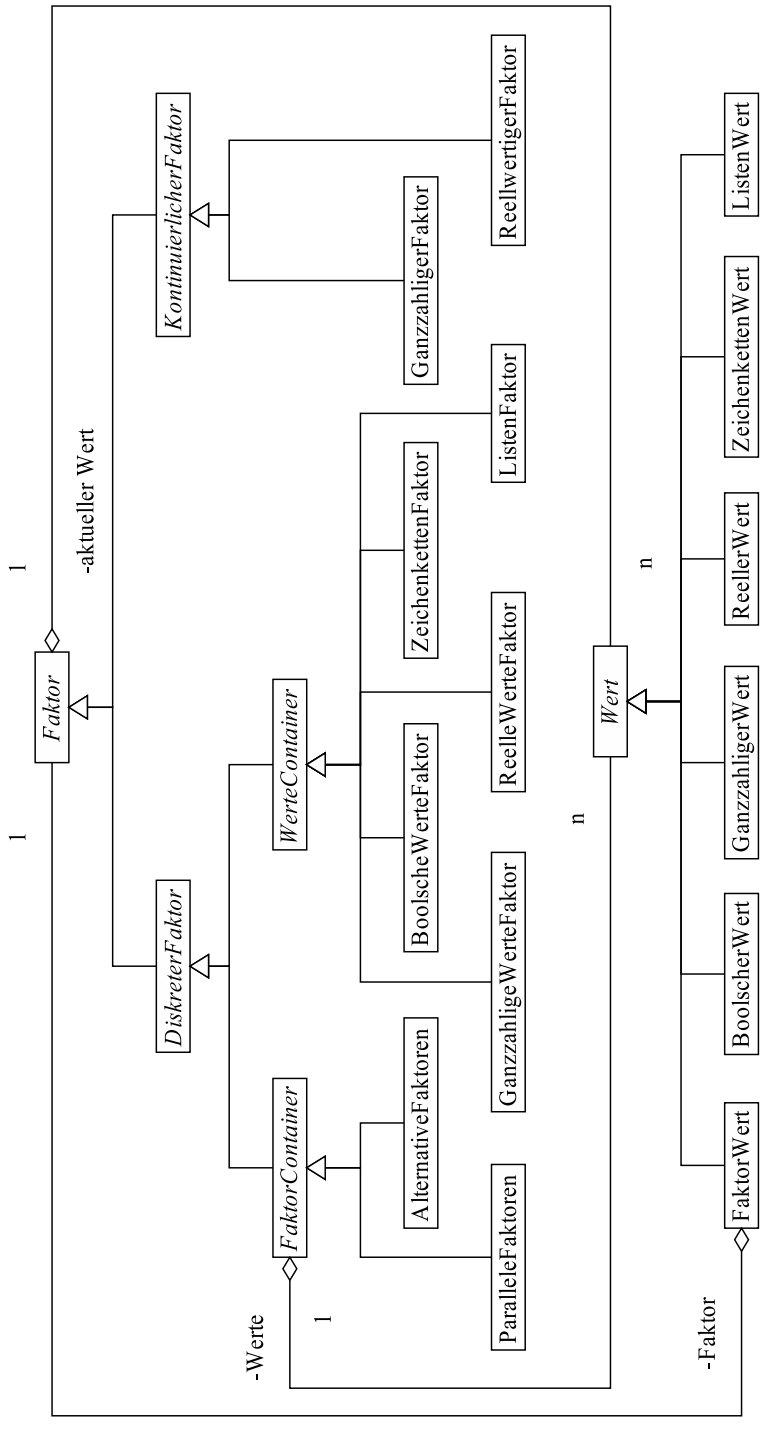

Abbildung B.6: Faktor- und Werteklassen der Komponente zum Management von Experimenten sowie der Zusammenhang zwischen ihnen

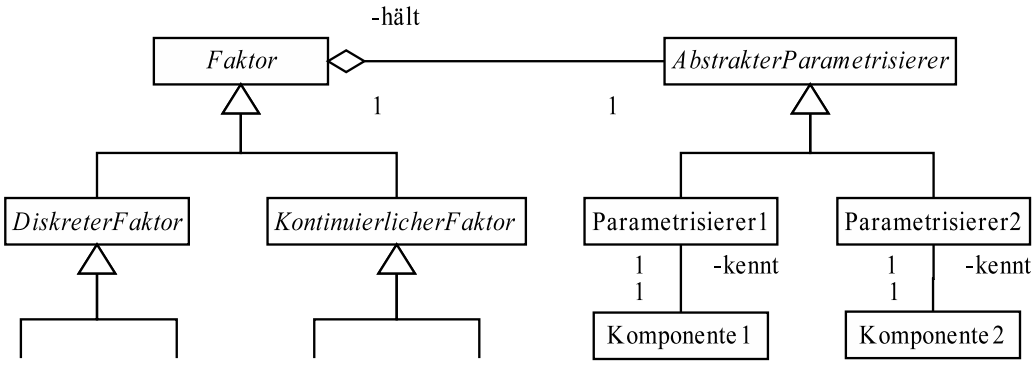

Abbildung B.7: Lose Kopplung zwischen Faktoren und zu parametrisierenden Komponenten über zwischengeschaltete Parametrisierer

torbaum hierarchisch strukturiert ist.

B.7.3 Parametrisieren von Softwarekomponenten

Das Parametrisieren eines Parameters bzw. einer gesamten Komponente mit dem Wert eines Faktors findet immer dann statt, wenn sich ein Faktor im Faktorbaum initialisiert oder erfolgreich inkrementiert. Somit ist gewährleistet, dass ein Faktor nur dann parametrisiert, wenn er zwingend parametrisieren muss.

Damit die Komponente zum Management von Experimenten allgemeingültig bleibt und der Faktorbaum nicht nur die in diesem Kapitel vorgestellten Softwarekomponenten parametrisieren kann, sondern auch beliebige andere, dürfen die Faktoren die von ihnen zu parametrisierenden Komponenten nicht kennen und nicht direkt auf sie referenzieren. Umgekehrt dürfen auch die Softwarekomponenten den Faktorbaum und seine Bestandteile nicht kennen und nicht direkt referenzieren. Um beide Verbote einzuhalten, ist die Klassenhierarchie von Parametrisierern in Abbildung B.7 entworfen worden. Für jeden Parameter bzw. für jede zu parametrisierende Komponente existiert ein spezieller Parametrisierer, welcher seinen Parameter bzw. seine Komponente direkt kennt. Jeder spezielle Parametrisierer ist direkt oder indirekt von einem abstrakten Parametrisierer abgeleitet worden. Der jeweilige spezielle Parametrisierer ist namentlich im XML des Faktorbaumes vermerkt. Die Erzeugerklasse eines Faktors identifiziert den speziellen Parametrisierer über seinen Namen und ordnet ihn dem Faktor zu, wobei der Faktor zwar den speziellen Parametrisierer hält, ihn aber nur als abstrakten Parametrisierer kennt und anspricht, wodurch die Komponente zum Management von Experimenten eine lose Kopplung zwischen sich und den anderen Komponenten

herstellt und auch das Management von Experimenten mit anderen Komponenten als den in diesem Kapitel aufgeführten erlaubt. Abbildung B.7 stellt die lose Kopplung zwischen Faktoren und zu parametrisierenden Komponenten über zwischengeschaltete Parametrisierer noch einmal in Form eines UML-Klassendiagrammes dar. Das entwickelte Entwurfsmuster kann keinem der Muster von [GHJV04] zugeordnet werden.

B.8 Komponenten zum Test des Softwaresystems

B.8.1 Aufspannen eines zufälligen, gerichteten Baumes

Algorithmus B.14 spannt einen rein zufälligen, gerichteten Baum über den Knoten des Graphen eines BNes auf. Der Baum umfasst alle Knoten des BNes, so dass der Graph nur noch aus einer Komponente besteht. Zwei beliebige, voneinander verschiedene Knoten des unterliegenden ungerichteten Graphen sind dann auch immer nur über genau einen Pfad verbunden. Zu

Algorithmus B.14 Aufspannen eines zufälligen Baumes

Eingabe: ein gerichteter Graph $G = (V, E)$
Ausgabe: ein gerichteter Baum G, der aus genau einer Komponente besteht

```
1:  prozedur SpanneZufallsBaumAuf(G)
2:      E ← ∅                                          ▷ Entferne alle Kanten
3:      N ← ∅                              ▷ Initialisiere Menge der Komponenten
4:      für i ← 0, ..., (|V| − 1) führe aus
5:          N_i ← {v_i}          ▷ Initialisiere i-te Komponente mit i-tem Knoten
6:          N ← N ∪ {N_i}                          ▷ Füge i-te Komponente N hinzu
7:      ende für
8:      für i ← 0, ..., (|V| − 2) führe aus                          ▷ |E| = |V| − 1
9:          N_u ←rufe WähleBeliebigesElementVon(N)
10:         N ← N \ {N_u}                          ▷ Entferne Komponente u aus N
11:         N_v ←rufe WähleBeliebigesElementVon(N)
12:         u ←rufe WähleBeliebigesElementVon(N_u)
13:         v ←rufe WähleBeliebigesElementVon(N_v)
14:         E ← E ∪ {(u, v)}                       ▷ Füge Kante (u, v) E hinzu
15:         N_v ← N_v ∪ N_u          ▷ Vereine N_v mit N_u, da eine Komponente
16:     ende für
17: ende prozedur
```

Beginn entfernt Algorithmus B.14 alle Kanten aus dem Graphen G, ord-

net jeden Knoten v_i einer Komponente N_i des Graphen G zu und fügt die Komponente N_i der Menge N aller Komponenten hinzu. Ein Baum, der aus genau einer Komponente besteht, weist immer genau $|E| = |V| - 1$ Kanten auf, so dass Algorithmus B.14 die zweite Zählschleife genau $|V| - 1$ mal durchläuft, um der Menge der Kanten E genau $|V| - 1$ Kanten hinzuzufügen. Er wählt eine beliebige Komponente N_u aus der Menge N der Komponenten und entfernt N_u aus N. Nun wählt er wiederum eine beliebige Komponente N_v aus der Menge N der verbleibenden Komponenten. Im Anschluss wählt der Algorithmus einen beliebigen Knoten u aus der Komponente N_u und einen beliebigen Knoten v aus der Komponente N_v und verbindet u mit v zur gerichteten Kante (u, v), die er der Menge E der Kanten des Graphen G hinzufügt. Zum Ende der Zählschleife werden die Komponenten N_u und N_v zu N_v vereint, da die Kante (u, v) sie ja jetzt verbindet, sie somit eine gemeinsame Komponente bilden und N_u bereits sowieso nicht mehr in der Menge N der Komponenten enthalten ist.

In der Realität ist Algorithmus B.14 so erweitert worden, dass er die Anzahl eingehender und/oder ausgehender Kanten für einen Knoten auf einen konstanten, aber parametrisierbaren Wert begrenzt.

Gegebenenfalls werden dem Graphen G nun noch zufällig weitere Kanten hinzugefügt, so dass er kein Baum mehr ist und für die Tests einen höheren „Vernetzungsgrad“ aufweist. Auch hier kann die Anzahl eingehender und/oder ausgehender Kanten für einen Knoten begrenzt werden, so dass man ähnlich große Familien im BN erhält, was für einen Vergleich der PI über mehreren BNen von Vorteil ist. Damit von vornherein keine Zyklen im BN entstehen, können die Knoten des Baumes mittels Algorithmus B.12 topologisch sortiert[20] und die zusätzlichen Kanten konform zur Sortierung eingefügt werden.

Die Komponente zum Test des Softwaresystems versieht alle Variablen eines generierten BNes mit der gleichen, aber parametrisierbaren Anzahl von Zuständen, was eine bessere Test- und Vergleichbarkeit der zu testenden Komponenten gewährleistet. Die Bewertungsfunktionen spannt das BN selbständig auf.

[20] Eine topologische Sortierung erhielte man auch, wenn man in Algorithmus B.14 die Kanten immer von einem Knoten mit kleinerem Index zu einem Knoten mit größerem Index einfügen würde.

B.8.2 Erstellen verschiedener Typen von Bewertungsfunktionen

Um besagte Typen von Bewertungsfunktionen zu erstellen, wird davon ausgegangen, dass die $|S|-1$ Zustände $S = \{s_0, \ldots, s_{|S|-1}\}$ eines jeden Knotens des BNes entsprechend ihren Indizes geordnet vorkommen, reellen Zahlen im Intervall $[0;1]$ entsprechen und somit „normiert“ vorliegen.

Die Funktion

$$f_{\mathbb{N}}(s_i) = i \tag{B.56}$$

bestimmt den Index i des i-ten Zustandes $s_i \in S$ und gilt für alle $0 \leq i < |S|$. Die Funktion

$$f_{[0;1]}(s) = \frac{f_{\mathbb{N}}(s)}{|S|-1} \tag{B.57}$$

gibt für jeden Zustand $s \in S$ die ihm zugeordnete reelle Zahl aus dem Intervall $[0;1]$ zurück. Der erste Zustand s_0 entspricht somit der 0, der letzte s_{n-1} der 1, und für die Zustände $s_1, \ldots, s_{n-2}$, die zwischen dem ersten Zustand s_0 und dem letzten s_{n-1} liegen, gilt

$$f_{[0;1]}(s_i) - f_{[0;1]}(s_{i-1}) = f_{[0;1]}(s_{i+1}) - f_{[0;1]}(s_i) \tag{B.58}$$

für $0 < i < |S-1|$. Die Zustände $s_0, \ldots, s_{n-1}$ bzw. die ihnen zugeordneten reellen Zahlen $f_{[0;1]}(s_0), \ldots, f_{[0;1]}(s_{n-1})$ sind ergo in der Reihenfolge ihrer Indizierung gleichmäßig im Standardintervall $[0;1]$ verteilt.

Es soll nun die Bewertungsfunktion einer Variable bzw. eines Knotens eines BNes erstellt werden. Die Variablen $u_0, \ldots, u_{n-1}$ beeinflussen die Variable v direkt und kausal, und sie befinden sich in den Zuständen $s_0, \ldots, s_{n-1}$. Um die bedingte Wahrscheinlichkeit $\Pr(v = s | u_0 = s_0, \ldots, u_{n-1} = s_{n-1})$ für alle $s \in S_v$ zu ermitteln, muss zuerst ein Mittel- bzw. Erwartungswert m aus den Zuständen $s_0, \ldots, s_{n-1}$ bzw. aus den ihnen zugeordneten reellen Werten $f_{[0;1]}(s_0), \ldots, f_{[0;1]}(s_{n-1})$ errechnet und auf das Standardintervall $[0;1]$ projiziert werden. Das geschieht für jeden Typ einer Bewertungsfunktion auf verschiedene Art und Weise.

Aus m ist dann eine bedingte Wahrscheinlichkeitsverteilung abzuleiten. Die grundlegende Idee besteht darin, dass die bedingten Wahrscheinlichkeiten für die Zustände s hoch sind, deren reelle Werte $f_{[0;1]}(s)$ dem Mittelwert m gleichen, und dass die sonstigen bedingten Wahrscheinlichkeiten niedrig sind. Da m im Allgemeinen nicht auf den reellen Wert $f_{[0;1]}(s)$ eines Zustandes s der Variable v fällt, wird die Wahrscheinlichkeit 1,0 auf die Zustände direkt rechts und links von m in Abhängigkeit von ihrem Abstand zu m verteilt. Der

Abstand zwischen dem reellen Wert $f_{[0;1]}(s)$ eines Zustandes s der Variable v und dem Wert m beträgt

$$b = |f_{[0;1]}(s) - m|. \tag{B.59}$$

Ist b kleiner als der Abstand

$$a = \frac{1}{|S_v| - 1} \tag{B.60}$$

zwischen den reellen Werten zweier aufeinanderfolgender Zustände der Variable v, so liegt $f_{[0;1]}(s)$ direkt neben m, ohne dass ein Wert eines von s verschiedenen Zustandes der Variable v zwischen $f_{[0;1]}(s)$ und m liegt. Für die Zustände $s \in S_v$, für die $b < a$ gilt, beträgt die bedingte Wahrscheinlichkeit $1 - b(|S_v| - 1)$. Für alle von s verschiedenen Zustände beträgt sie 0. Die folgende Formel B.61 fasst die Berechnung einer bedingten Wahrscheinlichkeit für eine Bewertungsfunktion aus m noch einmal zusammen:

$$\Pr(v = s | u_0 = s_0, \ldots, u_{n-1} = s_{n-1}) = \begin{cases} 1 - b(|S_v| - 1), \text{ wenn } b < a \\ 0, \text{ sonst.} \end{cases} \tag{B.61}$$

In der Komponente zum Test des Softwaresystems kann alternativ zu Formel B.61 aus m unter bestimmten Annahmen auch eine theoretische unimodale Verteilung für die bedingten Wahrscheinlichkeiten abgeleitet werden, welche den Wert m als Erwartungswert aufweist.

Es sind vorerst vier Typen von Bewertungsfunktionen entworfen worden, wobei zusätzliche Typen ohne weiteres denkbar wären. Für jeden Typ existiert eine spezielle Formel zum Berechnen des Wertes m.

Der Bewertungsfunktionstyp „Summe“ drückt aus, dass die bedingte Wahrscheinlichkeit für einen hohen/niedrigen Zustand der Variable v tendenziell hoch/niedrig ist, wenn die (gewichtete) Summe der Zustände der Variablen $u_0, \ldots, u_{n-1}$ ebenfalls hoch/niedrig ist. Der Wert m berechnet sich, wie folgt:

$$m = \frac{\sum_{i=0}^{n-1} f_{[0;1]}(s_i)}{n}. \tag{B.62}$$

Tabelle B.4 gibt ein Beispiel für eine Bewertungsfunktion vom Typ „Summe“. Man sieht, dass die bedingten Wahrscheinlichkeiten nahe einer imaginären Diagonale von links oben nach rechts unten jeweils hoch und sonst niedrig sind.

Der nächste Typ einer Bewertungsfunktion ist die „inverse Summe“. Sie drückt aus, dass die bedingte Wahrscheinlichkeit für einen hohen/niedrigen

Tabelle B.4: Beispiel für eine Bewertungsfunktion vom Typ „Summe“. Die Variablen v, u_0 und u_1 besitzen 4, 3 respektive 5 Zustände. Die Kopfspalten und -zeilen enthalten die den Zuständen der Variablen zugeordneten reellen Zahlen.

		v			
u_0	u_1	0,00	0,333	0,667	1,00
0	0	1	0	0	0
0	0,25	0,625	0,375	0	0
0	0,5	0,25	0,75	0	0
0	0,75	0	0,875	0,125	0
0	1	0	0,5	0,5	0
0,5	0	0,25	0,75	0	0
0,5	0,25	0	0,875	0,125	0
0,5	0,5	0	0,5	0,5	0
0,5	0,75	0	0,125	0,875	0
0,5	1	0	0	0,75	0,25
1	0	0	0,5	0,5	0
1	0,25	0	0,125	0,875	0
1	0,5	0	0	0,75	0,25
1	0,75	0	0	0,375	0,625
1	1	0	0	0	1

Tabelle B.5: Beispiel für eine Bewertungsfunktion vom Typ „inverse Summe“. Die Variablen v, u_0 und u_1 besitzen 4, 3 respektive 5 Zustände. Die Kopfspalten und -zeilen enthalten die den Zuständen der Variablen zugeordneten reellen Zahlen.

		v			
u_0	u_1	0	0,333	0,667	1
0	0	0	0	0	1
0	0,25	0	0	0,375	0,625
0	0,5	0	0	0,75	0,25
0	0,75	0	0,125	0,875	0
0	1	0	0,5	0,5	0
0,5	0	0	0	0,75	0,25
0,5	0,25	0	0,125	0,875	0
0,5	0,5	0	0,5	0,5	0
0,5	0,75	0	0,875	0,125	0
0,5	1	0,25	0,75	0	0
1	0	0	0,5	0,5	0
1	0,25	0	0,875	0,125	0
1	0,5	0,25	0,75	0	0
1	0,75	0,625	0,375	0	0
1	1	1	0	0	0

Zustand der Variable v tendenziell hoch/niedrig ist, wenn die (gewichtete) Summe der Zustände der Variablen $u_0, \ldots, u_{n-1}$ niedrig/hoch ist. Sie unterscheidet sich vom Typ „Summe“ lediglich dadurch, dass der Wert m „invertiert“ wird:

$$m = 1 - \frac{\sum_{i=0}^{n-1} f_{[0;1]}(s_i)}{n}. \tag{B.63}$$

Tabelle B.5 gibt ein Beispiel für eine Bewertungsfunktion vom Typ „inverse Summe“. Man sieht, dass die bedingten Wahrscheinlichkeiten nahe einer imaginären Diagonale von links unten nach rechts oben jeweils hoch und sonst niedrig sind.

Der dritte Typ einer Bewertungsfunktion ist das „Maximum“. Es drückt aus, dass die bedingte Wahrscheinlichkeit für einen hohen/niedrigen Zustand der Variable v tendenziell hoch/niedrig ist, wenn das Maximum der Zustände der Variablen $u_0, \ldots, u_{n-1}$ hoch/niedrig ist. Der Wert m wird für Bewertungsfunktionen vom Typ „Maximum“ wie folgt berechnet:

$$m = \max_{i=0}^{n-1} f_{[0;1]}(s_i). \tag{B.64}$$

Tabelle B.6: Beispiel für eine Bewertungsfunktion vom Typ „Maximum“. Die Variablen v, u_0 und u_1 besitzen 4, 3 respektive 5 Zustände. Die Kopfspalten und -zeilen enthalten die den Zuständen der Variablen zugeordneten reellen Zahlen.

		v			
u_0	u_1	0	0,333	0,667	1
0	0	1	0	0	0
0	0,25	0,25	0,75	0	0
0	0,5	0	0,5	0,5	0
0	0,75	0	0	0,75	0,25
0	1	0	0	0	1
0,5	0	0	0,5	0,5	0
0,5	0,25	0	0,5	0,5	0
0,5	0,5	0	0,5	0,5	0
0,5	0,75	0	0	0,75	0,25
0,5	1	0	0	0	1
1	0	0	0	0	1
1	0,25	0	0	0	1
1	0,5	0	0	0	1
1	0,75	0	0	0	1
1	1	0	0	0	1

Tabelle B.6 gibt ein Beispiel für eine Bewertungsfunktion vom Typ „Maximum“.

Der vierte und bislang letzte Bewertungsfunktionstyp ist das „Minimum“. Es drückt aus, dass die bedingte Wahrscheinlichkeit für einen hohen/niedrigen Zustand der Variable v tendenziell hoch/niedrig ist, wenn das Minimum der Zustände der Variablen $u_0, \ldots, u_{n-1}$ hoch/niedrig ist. Es unterscheidet sich vom „Maximum“ lediglich dadurch, dass der Wert m mittels des Minimum- und nicht mittels des Maximum-Operators berechnet wird:

$$m = \min_{i=0}^{n-1} f_{[0;1]}(s_i). \tag{B.65}$$

Tabelle B.7 gibt ein Beispiel für eine Bewertungsfunktion vom Typ „Minimum“.

Bisher ist lediglich von den bedingten Wahrscheinlichkeiten in Bewertungsfunktionen die Rede gewesen. Die unbedingten Wahrscheinlichkeiten der Variablen bzw. Knoten des BNes, die keine Eltern besitzen, liegen bei der Generierung BNe durch die Komponente zum Test des Softwaresystems

Tabelle B.7: Beispiel für eine Bewertungsfunktion vom Typ „Minimum“. Die Variablen v, u_0 und u_1 besitzen 4, 3 respektive 5 Zustände. Die Kopfspalten und -zeilen enthalten die den Zuständen der Variablen zugeordneten reellen Zahlen.

		v			
u_0	u_1	0	0,333	0,667	1
0	0	1	0	0	0
0	0,25	1	0	0	0
0	0,5	1	0	0	0
0	0,75	1	0	0	0
0	1	1	0	0	0
0,5	0	1	0	0	0
0,5	0,25	0,25	0,75	0	0
0,5	0,5	0	0,5	0,5	0
0,5	0,75	0	0,5	0,5	0
0,5	1	0	0,5	0,5	0
1	0	1	0	0	0
1	0,25	0,25	0,75	0	0
1	0,5	0	0,5	0,5	0
1	0,75	0	0	0,75	0,25
1	1	0	0	0	1

per Definition gleichverteilt vor.

Literaturverzeichnis

[AB99] Anthony, Martin ; Bartlett, Peter L.: *Neural Network Learning: Theoretical Foundations.* Cambridge : Cambridge University Press, 1999

[ABE+96] Abramson, Bruce ; Brown, John ; Edwards, Ward ; Murphy, Allan ; Winkler, Robert L.: Hailfinder: A Bayesian System for Forecasting Severe Weather. In: *International Journal of Forecasting* 12 (1996), März, Nr. 1, S. 57–71. – available at http://ideas.repec.org/a/eee/intfor/v12y1996i1p57-71.html

[Ada96] Adam, Dietrich: *Planung und Entscheidung: Modelle - Ziele - Methoden; mit Fallstudien und Lösungen.* 4., vollständig überarbeitete und wesentlich erweiterte Auflage. Wiesbaden : Gabler Verlag, 1996

[Ada98] Adam, Dietrich: *Produktions-Management.* 9., überarbeitete. Wiesbaden : Gabler Verlag, 1998

[AF91] Abramson, Bruce ; Finizza, Anthony: Using Belief Networks to Forecast Oil Prices. In: *International Journal of Forecasting* 7 (1991), November, Nr. 3, S. 299–315. – available at http://ideas.repec.org/a/eee/intfor/v7y1991i3p299-315.html

[AG04] Agosta, John M. ; Gardos, Thomas: Bayes Network „Smart“ Diagnostics. In: *Intel Technology Journal* 08 (2004), November, Nr. 04, S. 361–372. – ISSN 1535–864X

[AKL98] Aarts, Emile H. L. ; Korst, Jan H. M. ; van Laarhoven, Peter J. M.: Simulated Annealing. In: Aarts, Emile H. L. (Hrsg.) ; Lenstra, Jan K. (Hrsg.): *Local Search in Combinatorial Optimization.* Chichester : John Wiley & Sons, 1998, S. 91–120

[AL98] Aarts, Emile H. L. (Hrsg.) ; Lenstra, Jan K. (Hrsg.): *Local Search in Combinatorial Optimization.* Chichester : John Wiley & Sons, 1998

[Ala95] Alander, Jarmo T.: An Indexed Bibliography of Genetic Algorithms in Manufacturing / University of Vaasa, Department of Information Technology and Production Economics. 1995 (94-1MANU). – Forschungsbericht. (ftp://ftp.uwasa.fi/cs/report94-1/gaMANUbib.ps.Z)

[Alm95] Almond, Russel G.: *Graphical Belief Modeling.* London : Chapman & Hall, 1995

[AOJJ89] Andersen, Stig K. ; Olesen, Kristian G. ; Jensen, Finn V. ; Jensen, Frank: HUGIN - A Shell for Building Belief Universes for Expert Systems. In: Sridharan, N. S. (Hrsg.): *Proceedings of the 11th International Joint Conference on Artificial Intelligence.* San Mateo, Kalifornien : Morgan Kaufmann Publishers, 1989, S. 1080–1085

[AOJJ90] Andersen, Stig K. ; Olesen, Kristian G. ; Jensen, Finn V. ; Jensen, Frank: HUGIN - A Shell for Building Belief Universes for Expert Systems. In: Shafer, G. (Hrsg.) ; Pearl, Judea (Hrsg.): *Readings in Uncertain Reasoning.* San Mateo, Kalifornien : Morgan Kaufmann Publishers, 1990, S. 332–337

[AWFA87] Andreassen, Steen ; Woldbye, Marianne ; Falck, Bjørn ; Andersen, Stig K.: MUNIN - A Causal Probabilistic Network for Interpretation of Electromyographic Findings. In: *Proceedings of the 10th International Joint Conference on Artificial Intelligence.* Milano, 1987, S. 366–372

[Bar94] Barfels, Lutz: Einsatz der Simulationstechnik zur Planung einer zentralisierten Teilefertigung. In: Biethahn, Jörg (Hrsg.) ; Hummeltenberg, Wilhelm (Hrsg.) ; Schmidt, Bernd (Hrsg.) ; Witte, Thomas (Hrsg.): *Simulation als betriebliche Entscheidungshilfe – Fortschritte in der Simulationstechnik Band 8.* Braunschweig : Vieweg Verlag, 1994, S. 354–369

[Bar99a] Bartak, Roman: Constraint Programming: In Pursuit of the Holy Grail. In: *Proceedings of the Week of Doctoral Students (WDS99).* Prague : MatFyzPress, June 1999, S. 555–564

[Bar99b] Barton, Russell R.: *Graphical Methods for the Design of Experiments.* New York : Springer Verlag, 1999

[Bar02] Barton, Russell R.: Designing Simulation Experiments. In: Yücesan, E. (Hrsg.) ; Chen, C.-H. (Hrsg.) ; Snowdon, J. L. (Hrsg.) ; Charnes, J. M. (Hrsg.): *Proceedings of the 2002 Winter Simulation Conference.* San Diego, CA, December 8–11 2002, S. 45–51

[Bau91] Bauer, Heinz: *Wahrscheinlichkeitstheorie.* 4., völlig überarbeitete und neugestaltete. Berlin : Walter de Gruyter & Co., 1991 (De-Gruyter-Lehrbuch). – ISBN 3–11–012191–3

[BB97] Bradwell, Richard ; Brown, Ken: Application Of Multi-Agent Cooperative Search To The JobShop Scheduling Problem. In: *PlanSIG 16.* Durham, December 1997

[BCG07] Bellifemine, Fabio L. ; Caire, Giovanni ; Greenwood, Dominic: *Developing Multi-Agent Systems with JADE.* John Wiley & Sons, February 2007

[BCNN01] Banks, Jerry ; Carson, II, John S. ; Nelson, Barry L. ; Nicol, David M.: *Discrete-Event System Simulation.* 3. Upper Saddle River : Prentice-Hall, 2001

[BD03] Brockwell, Peter J. ; Davis, Richard A.: *Time series: theory and methods.* New York : Springer Verlag, 2003 (Springer series in statistics). – ISBN 0–387–97429–6

[Bea96] Beasley, John E.: Obtaining test problems via Internet. In: *Journal of Global Optimization* 8 (1996), Nr. 4, S. 429–433

[BETT98] Battista, Giuseppe D. ; Eades, Peter ; Tamassia, Roberto ; Tollis, Ioannis G.: *Graph Drawing: Algorithms for the Visualization of Graphs.* Upper Saddle River, NJ : Prentice-Hall, 1998. – ISBN 0133016153

[BFS87] Bratley, Paul ; Fox, Bennett L. ; Schrage, Linus E.: *A guide to simulation.* 2. New York : Springer Verlag, 1987. – ISBN 3–540–96467–3 0–387–96467–3

[BJM04] Barth, Wilhelm ; Jünger, Michael ; Mutzel, Petra: Simple and Efficient Bilayer Cross Counting. In: *Journal of Graph Algorithms and Applications* 8 (2004), Nr. 2, S. 179—-194

[BKI03] Beierle, Christoph ; Kern-Isberner, Gabriele: *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. 2., überarbeitete und erweiterte. Braunschweig : Vieweg Verlag, 2003

[BL93] Brunak, Søren ; Lautrup, Benny: *Neuronale Netze: die nächste Computer-Revolution*. München : Hanser Verlag, 1993. – ISBN 3–446–17440–0

[Bös99] Böselt, Martin: *Statistik*. 2. München : Oldenbourg Verlag, 1999

[Bou94] Bouckaert, Remco R.: Probabilistic Network Construction Using the Minimum Description Length Prinziple / Department of Computer Science, Utrecht University, Netherlands. 1994 (RUU-CS-94-29). – Forschungsbericht

[Bou95] Bouckaert, Remco R.: *Bayesian Belief Networks: From Construction to Inference*, Department of Computer Science, Utrecht University, Netherlands, Diss., 1995

[Brü95] Brüggemann, Wolfgang: *Physica-Schriften zur Betriebswirtschaft*. Bd. 52: *Ausgewählte Probleme der Produktionsplanung: Modellierung, Komplexität und neuere Lösungsmöglichkeiten*. Heidelberg : Phyisca-Verlag, 1995. – ISBN 3–7908–0882–2

[BS97] Bentley, Jon L. ; Sedgewick, Robert: Fast Algorithms for Sorting and Searching Strings. In: *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1997

[BSMM99] Bronstein, Ilja N. ; Semendjajew, Konstantin A. ; Musiol, Gerhard ; Mühlig, Heiner: *Taschenbuch der Mathematik*. 4. überarbeitete und erweiterte Auflage der Neubearbeitung. Frankfurt am Main : Verlag Harri Deutsch, 1999

[BSS92] Bader, J. ; Stahel, W. A. ; Schmid, W.: Further Results of Grossversuch IV: The Effect of the First Rocket Launched into a Potential Hail Cell. In: *Applied Meteorology* 31 (1992), S. 700–707

[Bub93] Bube, Oskar: *Architektur und Implementierung von Prozeßidentifikatoren: ein Beitrag zur Modellierung und Steuerung komplizierter Fertigungssysteme*. Ilmenau, Technische Hochschule Ilmenau, Dissertation, 1993

[Bun91] Buntine, Wray: Theory Refinement on Bayesian Networks. In: D'Ambrosio, B. D. (Hrsg.) ; Smets, P. (Hrsg.) ; Bonissone, P. P. (Hrsg.): *Uncertainty in Artificial Intelligence: Proceedings of the 7th Conference*. San Mateo, CA : Morgan Kaufmann Publishers, 1991, S. 52–60

[Cal03] Callan, Robert: *Neuronale Netze im Klartext*. München : Pearson Studium, 2003 (Pearson Studium: Im Klartext). – ISBN 3–8273–7071–X

[CC04] Caridi, Maria ; Cavalieri, Sergio: Multi-agent systems in production planning and control: an overview. In: *Production Planning and Control* 15 (2004), März, Nr. 2, S. 106–118

[CDLS99] Cowell, Robert G. ; Dawid, A. P. ; Lauritzen, Steffen L. ; Spiegehalter, David J.: *Probabilistic Networks and Expert Systems*. New York : Springer Verlag, 1999

[CF99] Corsten, Hans (Hrsg.) ; Friedl, Birgit (Hrsg.): *Einführung in das Produktionscontrolling*. München : Verlag Vahlen, 1999 (Controlling). – ISBN 3–8006–2322–6

[CFLH04] de Campos, Luis M. ; Fernández-Luna, Juan M. ; Huete, Juan F.: Fast Propagation Algorithms for Singly Connected Networks and their Application to Information Retrieval. In: Gámez, José A. (Hrsg.) ; Moral, Serafín (Hrsg.) ; Salmerón, Antonio (Hrsg.): *Advances in Bayesian Networks* Bd. 146. Berlin : Springer Verlag, 2004, S. 271–288

[CGH97] Castillo, Enrique ; Guti(́e)rrez, Jos(́e) M. ; Hadi, Ali S.: *Expert Systems and Probabilistic Network Models*. New York : Springer Verlag, 1997 (Monographs in Computer Science)

[CH92] Cooper, Gregory F. ; Herskovits, Edward: A Bayesian Method for the Induction of Probabilistic Networks from Data. In: *Machine Learning* 9 (1992), Nr. 9, S. 309–347. – ISSN 0885–6125

[CH94] Clark, John ; Holton, Derek A.: *Graphentheorie: Grundlagen und Anwendungen*. Heidelberg : Spektrum Akademischer Verlag, 1994

[Chu78] Chung, Kai L.: *Elementare Wahrscheinlichkeitstheorie und stochastische Prozesse*. Berlin : Springer Verlag, 1978. – ISBN 3–540–08971–3 – 0–387–08971–3

[CM96] Corsten, Hans ; May, Constantin: Unterstützungspotential Künstlicher Neuronaler Netze für die Produktionsplanung und -steuerung. In: Corsten, Hans (Hrsg.) ; May, Constantin (Hrsg.): *Neuronale Netze in der Betriebswirtschaft – Anwendung in Prognose, Klassifikation und Optimierung*. Wiesbaden : Gabler Verlag, 1996, S. 235–257

[Cor04] Corsten, Hans: *Produktionswirtschaft: Einführung in das industrielle Produktionsmanagement*. 10., vollständig überarbeitete. München : Oldenbourg Verlag, 2004

[Coz98] Cozman, Fabio G.: *The Interchange Format for Bayesian Networks, Version 0.3*. August 1998. – URI: `http://www.cs.cmu.edu/~fgcozman/Research/InterchangeFormat/`, abgerufen am 29.08.2007 13.20 Uhr

[Cre91] Cremers, Armin B.: *Expertensysteme für die Planung der Produktion*. Köln : Verlag TÜV Rheinland, 1991 (Schriftenreihe technische Betriebsführung)

[CS95] Chan, Philip K. ; Stolfo, Salvatore J.: A Comparative Evaluation of Voting and Meta-Learning on Partitioned Data. In: *International Conference on Machine Learning*, 1995, S. 90–98

[CSG04] Cano, Rafale ; Sodo, Carmen ; Gutieérrez, José M.: Applications of Bayesian Networks in Meteorology. In: Gámez, José A. (Hrsg.) ; Moral, Serafín (Hrsg.) ; Salmerón, Antonio (Hrsg.): *Advances in Bayesian Networks* Bd. 146. Berlin : Springer Verlag, 2004, S. 309–328

[Czy95] Czys, Robert R.: Progress in planned weather modification research: 1991-1994 / Atmospheric Sciences Division, Illinois State Water Survey. Champaign, Illinois, 1995 (Rev. Geophys. Vol. 33 Suppl.). – U.S. National Report to IUGG, 1991-1994. 1995 American Geophysical Union

[Dı́93] Díez, Francisco J.: Parameter Adjustment in Bayes Networks: The Generalized Noisy OR-Gate. In: *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*. Washington, D.C., July 1993, S. 99–105

[Dı́96] Díez, Francisco J.: Local conditioning in Bayesian networks. In: *Artificial Intelligence* 87 (1996), November, Nr. 1-2, S. 1–20

[D'A95] D'Ambrosio, Bruce: Local Expression Languages for Probabilistic Dependence. In: *Approximate Reasoning* 13 (1995), July, Nr. 1, S. 61–81

[Dak65] Dakin, Robert J.: A tree-search algorithm for mixed integer programming problems. In: *The Computer Journal* 8 (1965), Nr. 3, S. 250–255

[DD91] Domschke, Wolfgang ; Drexl, Andreas: *Einführung in Operations Research*. 2., verbesserte und erweiterte Auflage. Berlin : Springer Verlag, 1991 (Springer-Lehrbuch). – ISBN 3–540–54386–4

[DD04] Deviren, Murat ; Daoudi, Khalid: Continuous Speech Recognition Using Dynamic Bayesian Networks: A Fast Decoding Algorithm. In: Gámez, José A. (Hrsg.) ; Moral, Serafín (Hrsg.) ; Salmerón, Antonio (Hrsg.): *Advances in Bayesian Networks* Bd. 146. Berlin : Springer Verlag, 2004, S. 289–308

[Dec07] Decision Theory & Adaptive Systems Group: *XML Belief Network File Format*. 2007. – URI: `http://research.microsoft.com/dtas/bnformat/default.htm`, abgerufen am 31.08.2007 11.15 Uhr

[Den80] Dennis, Arnett S.: *International Geophysics*. Bd. 24: *Weather Modification by Cloud Seeding*. New York : Academic Press, 1980

[DH93a] Druzdzel, Marek J. ; Henrion, Max: Belief Propagation in Qualitative Probabilistic Networks. In: Carrete, N. P. (Hrsg.) ; Singh, M. G. (Hrsg.): *Qualitative Reasoning and Decision Technologies*. CIMNE, Barcelona, 1993, S. 451–460

[DH93b] Druzdzel, Marek J. ; Henrion, Max: Efficient Reasoning in Qualitative Probabilistic Networks. In: *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI–93)*. Menlo Park, CA : The MIT Press, 1993, S. 548–553

[DIN98] DIN (Deutsches Institut für Normung e. V.) (Hrsg.): *Leittechnik – Prozeßautomatisierung – Automatisierung mit Prozeßrechensystemen, Begriffe*. Beuth Verlag, Juli 1998 (DIN V 19233)

[DMV99] Döring, Thomas ; Munkelt, Torsten ; Völker, Sven: Generierung komplexer Testdaten zur statistischen Analyse von Verfahren der Produktionsplanung und -steuerung. In: Böselt, Martin (Hrsg.): *Amtliche und Nichtamtliche Statistiken. 12. Ilmenauer Wirtschaftsforum* Technische Universität Ilmenau, Fachgebiet Wirtschaftsstatistik und Operations Research, 1999, S. 34–46

[DMV00] Döring, Thomas ; Munkelt, Torsten ; Völker, Sven: The Stochastic Generation of Test Data for the Statistical Analysis of Heuristic Procedures in the Context of Production Planning and Control. In: *German Open Conference On Probability and Statistics — Abstracts.* Hamburg, March 2000, S. 82–83

[Dör04] Döring, Thomas: *Wissensbasierte Parametrisierung von Planungsverfahren am Beispiel der simultanen Termin- und Kapazitätsplanung.* Ilmenau, Technische Universität Ilmenau, Diss., 2004

[Dru92] Druzdzel, Marek J.: *Probabilistic Reasoning in Decision Support Systems: From Computation to Common Sense.* Pittsburgh, PA, Department of Engineering and Public Policy, Carnegie Mellon University, Diss., Dezember 1992

[DS90] Dueck, Gunter ; Scheuer, Tobias: Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. In: *Journal of Computational Physics* 90 (1990), S. 161–175

[DS04] Dorigo, Marco ; Stützle, Thomas: *Ant Colony Optimization.* The MIT Press, 2004

[DSV97] Domschke, Wolfgang ; Scholl, Armin ; Voss, Stefan: *Produktionsplanung: Ablauforganisatorische Aspekte.* 2. Berlin : Springer Verlag, 1997

[DSW93] Dueck, Gunter ; Scheuer, Tobias ; Wallmeier, Hans-Martin: Toleranzschwelle und Sintflut: neue Ideen zur Optimierung. In: *Spektrum der Wissenschaft* (1993), März, S. 42–51

[Dud89] Duden, Konrad: *Der Große Duden: Wörterbuch und Leitfaden der deutschen Rechtschreibung.* 5. Leipzig : Bibliographisches Institut, 1989. – Auflage bezieht sich auf die 18. Neubearbeitung

[Dud05] Duda, Jerzy: Lot-Sizing in a Foundry Using Genetic Algorithm and Repair Functions. In: Raidl, Günther R. (Hrsg.) ; Gottlieb, Jens (Hrsg.): *Evolutionary Computation in Combinatorial Optimization, 5th European Conference, EvoCOP 2005, Proceedings* Bd. 3448. Lausanne, Switzerland : Springer Verlag, March/April 2005. – ISBN 3–540–25337–8, S. 101–111

[Due93] Dueck, Gunter: New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. In: *Journal of Computational Physics* 104 (1993), S. 86–92

[DV99] Dean, Angela ; Voss, Daniel: *Design and Analysis of Experiments*. New York : Springer Verlag, 1999 (Springer Texts in Statistics)

[Eck02] Eckardt, Frank: *Ein Beitrag zu Theorie und Praxis datengetriebener Modellgeneratoren zur Simulation von Produktionssystemen*, Technische Universität Ilmenau, Diss., 2002

[EHP00] Evans, Merran ; Hastings, Nicholas A. J. ; Peacock, Brian: *Statistical Distributions*. 3. New York, NY : John Wiley & Sons, 2000 (Wiley Series in Probability and Statistics: Texts and References Section). – ISBN 0–471–37124–6

[ES95] Ezawa, Kazuo J. ; Schuermann, Til: Fraud/Uncollectible Debt Detection Using a Bayesian Network Based Learning System: A Rare Binary Outcome with Mixed Data Structures. In: Besnard, Philippe (Hrsg.) ; Hanks, Steve (Hrsg.): *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*. Montreal, Quebec, Canada, August 1995, S. 157–166

[ES00] Ester, Martin ; Sander, Jörg: *Knowledge Discovery in Databases: Techniken und Anwendungen*. Berlin : Springer Verlag, 2000

[ES03] Eiben, A. E. ; Smith, J. E.: *Introduction to evolutionary computing*. Berlin : Springer Verlag, 2003 (Natural computing series). – ISBN 3–540–40184–9

[Exp04a] Expert, Hugin: 2004. – URI: `http://www.hugin.com/cases/Industry/Advocate/Controle_Systems.article` abgerufen am 21.11.2004

[Exp04b] Expert, Hugin: *Order Production – A Framework for Prediction of Parts Demand*. 2004. – URI: `http://www.hugin.com/cases/Industry/PartDemand/PartDemand.article` abgerufen am 21.11.2004

[FA97a] Frühwirth, Thom W. ; Abdennadher, Slim: Anwendungen Constraintbasierter Programmierung. In: *Jahrestagung der Gesellschaft für Informatik 1997*. Aachen, September 1997 (Lecture Notes in Computer Science), S. 317–326

[FA97b] Frühwirth, Thom W. ; Abdennadher, Slim: *Constraint Programmierung: Grundlagen und Anwendungen*. Berlin : Springer Verlag, 1997

[Fei92] Feil, Peter: *Schriften zur Produktion*. Bd. 4: *Die wissensbasierte Lagerhaltungssimulation zur Unterstützung einer verbrauchsgesteuerten Disposition*. Frankfurt am Main, Bern, New York, Paris : Peter Lang Verlag, 1992. – ISBN 3–631–44699–3

[Fel99] Feldmann, Martin: *Naturanaloge Verfahren: Metaheuristiken zur Reihenfolgeplanung*. Wiesbaden : Deutscher Universitäts-Verlag, 1999

[FHKR95] Forbes, Jeff ; Huang, Timothy ; Kanazawa, Keiji ; Russell, Stuart J.: The BATmobile: Towards a Bayesian Automated Taxi. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Montreal, 1995, S. 1878–1885

[Fis73a] Fishman, George S.: *Concepts and Methods in Discrete Event Digital Simulation*. New York : John Wiley & Sons, September 1973. – ISBN 0471261556

[Fis73b] Fisz, Marek: *Wahrscheinlichkeitsrechnung und mathematische Statistik*. 7. Berlin : Deutscher Verlag der Wissenschaften, 1973 (Hochschulbücher für Mathematik)

[Fis01] Fishman, George S.: *Discrete Event Simulation: Modeling, Programming, and Analysis*. New York : Springer Verlag, 2001 (Springer Series in Operations Research)

[FLNP00] Friedman, Nir ; Linial, Michal ; Nachman, Iftach ; Pe'er, Dana: Using Bayesian Networks to Analyze Expression Data. In: *Computational Biology* 7 (2000), S. 601–620

[FMR98] Friedman, Nir ; Murphy, Kevin ; Russell, Stuart: Learning the Structure of Dynamic Probabilistic Networks. In: Cooper, Gregory F. (Hrsg.) ; Moral, Serafín (Hrsg.): *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. Madison, WI : Morgan Kaufmann Publishers, July 1998, S. 139–147

[For61] Forrester, Jay W.: *Industrial Dynamics*. Cambridge, Massachusetts : The MIT Press, 1961

[For71] Forrester, Jay W.: *Principles of Systems: Text and Workbook; Chapters 1 through 10*. 2nd preliminary, 5th printing. Cambridge, Massachusetts : Wright-Allen Press, 1971 (MIT Press/Wright-Allen series in system dynamics)

[FPA+99] Floudas, Christodoulos A. ; Pardalos, Panos M. ; Adjiman, Claire S. ; Esposito, William R. ; Gümüş, Zeynep H. ; Harding, Stephen T. ; Klepeis, John L. ; Meyer, Clifford A. ; Schweiger, Carl A.: *Nonconvex Optimization and Its Application*. Bd. 33: *Handbook of Test Problems in Local and Global Optimization*. Dordrecht : Kluwer Academic Publishers, 1999

[FT63] Fisher, H. ; Thompson, G. L.: Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. In: Muth, J. F. (Hrsg.) ; Thompson, G. L. (Hrsg.): *Industrial Scheduling*. Englewood Cliffs, New Jersey : Prentice-Hall, 1963, S. 225–251

[GG84] Geman, Stuart ; Geman, Donald: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1984, S. 721–741

[GG90] Geman, Stuart ; Geman, Donald: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In: Shafer, Glenn (Hrsg.) ; Pearl, Judea (Hrsg.): *Readings in Uncertain Reasoning*. Morgan Kaufmann Publishers, 1990. – ISBN 1–55860–125–2, S. 452–472

[GH98] Gross, Donald ; Harris, Carl M.: *Fundamentals of Queueing Theory*. 3. New York : John Wiley & Sons, 1998 (Wiley Series in Probability and Statistics, A Wiley-Interscience Publication). – ISBN 0–471–17083–6

[GHJV04] Gamma, Erich ; Helm, Richard ; Johnson, Ralph E. ; Vlissides, John: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. München : Addison Wesley, 2004 (Professionelle Softwareentwicklung, Programmer's choice). – ISBN 3–8273–2199–9

[GKK04] Gerdes, Ingrid ; Klawonn, Frank ; Kruse, Rudolf: *Evolutionäre Algorithmen: Genetische Algorithmen - Strategien und Optimierungsverfahren - Beispielanwendungen*. 1. Auflage. Wiesbaden : Vieweg Verlag, 2004 (Computational intelligence). – ISBN 3–528–05570–7

[GKNV93] Gansner, E. R. ; Koutsofios, E. ; North, S. C. ; Vo, K.-P.: A Technique for Drawing Directed Graphs. In: *IEEE Transactions on Software Engineering* 19 (1993), Nr. 3, S. 214–230. – ISSN 0098–5589

[GL01] Glover, Fred ; Laguna, Manuel: *Tabu Search*. 4. London : Kluwer Academic Publishers, 2001. – ISBN 0–7923–9965–X

[Gol80] Golumbic, Martin C.: *Algorithmic Graph Theory and Perfect Graphs*. New York : Academic Press, 1980

[Gol89] Goldberg, David E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Massachusetts : Addison Wesley, 1989. – ISBN 0–201–15767–5

[GR99] Gmilkowsky, Peter ; Roth, Frank: Simulation und Wissensverarbeitung als integrale Bestandteile eines Systems zur Fertigungsdisposition. In: Biethahn, Jörg (Hrsg.) ; Hummeltenberg, Wilhelm (Hrsg.) ; Schmidt, Bernd (Hrsg.) ; Stähly, Paul (Hrsg.) ; Witte, Thomas (Hrsg.): *Simulation als betriebliche Entscheidungshilfe: State of the Art und neuere Entwicklungen*. Heidelberg : Phyisca-Verlag, 1999, S. 219–254

[GS86] Greene, J. W. ; Supowit, K. J.: Simulated Annealing Without Rejected Moves. In: *IEEE Transactions on Computeraided Design of Integrated Circuits and Systems* 5 (1986), Nr. 1, S. 221–228

[Gün93] Günzel, Uwe: *Entwicklung und Einsatz eines Simulationsverfahrens für operative und strategische Probleme der Produktionsplanung und -steuerung*. München : Hanser Verlag, 1993

[Gün97] Günther, Manfred: *Kontinuierliche und zeitdiskrete Regelungen.* Stuttgart : Teubner Verlag, 1997

[Har00] Harary, Frank: *Graph Theory.* Nachdruck. Cambridge, Mass : Perseus Books, 2000

[Hay99] Haykin, Simon: *Neural Networks: A Comprehensive Foundation.* 2., internationale Ausgabe. Upper Saddle River, NJ : Prentice-Hall, 1999. – ISBN 0–13–273350–1

[HB95] Horvitz, Eric J. ; Barry, Matthew: Display of Information for Time-Critical Decision Making. In: *Proceedings of 11th Conference on Uncertainty in Artificial Intelligence.* Montreal : Morgan Kaufmann Publishers, August 1995, S. 296–305

[HBH+98] Horvitz, Eric J. ; Breese, Jack ; Heckerman, David E. ; Hovel, David ; Rommelse, Koos: The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence.* Madison, WI : Morgan Kaufmann Publishers, 1998, S. 256–265

[HE99] Hartung, Joachim ; Elpelt, Bärbel: *Multivariate Statistik: Lehr- und Handbuch der angewandten Statistik.* 6. München : Oldenbourg Verlag, 1999

[Hec88] Heckerman, David E.: An Empirical Comparison of Three Inference Methods. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence.* San Francisco, CA : Morgan Kaufmann Publishers, 1988, S. 158–169

[Hec96] Heckerman, David E.: A Tutorial on Learning with Bayesian Networks, Revised Version / Microsoft Research, Advanced Technology Division, Microsoft Corporation. Redmond, WA, 1996 (MSR-TR-95-06). – Forschungsbericht

[Hec99] Heckerman, David E.: A Tutorial on Learning with Bayesian Networks. In: Jordan, Michael I. (Hrsg.): *Learning in Graphical Models.* The MIT Press, 1999, S. 301–354

[Hed98] Hedberg, Sara R.: Is AI Going Mainstream at last? A Look inside Microsoft Research. In: *IEEE Intelligent Systems* (1998), March/April, S. 21–25

[Hep04] Hepp, Thomas: *Potential von Ruin-and-Recreate zur Lösung von Optimierungsproblemen der Produktionsplanung und -steuerung.* Ilmenau, Technische Universität Ilmenau, Diplomarbeit, Mai 2004

[HH97] Hansmann, Karl-Werner ; Hoeck, Michael: Production Control of a Flexible Manufacturing System in a Job Shop Environment. In: *International Transactions in Operations Research* 4 (1997), Nr. 5/6, S. 341–351

[HH98] Heckerman, David E. ; Horvitz, Eric J.: Inferring Informational Goals from FreeText Queries: A Bayesian Approach. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence.* Madison, WI : Morgan Kaufmann Publishers, July 1998, S. 230–237

[HHN89] Heckerman, David E. ; Horvitz, Eric J. ; Nathwani, Bharat N.: Update on the Pathfinder Project. In: *Proceedings of the Thirteenth Symposium on Computer Applications in Medical Care.* Washington, DC : IEEE Computer Society Press, November 1989, S. 203–207

[Hil98] Hill, Raymond R.: An Analytical Comparision of Optimization Problem Generation Methodologies. In: Medeiros, Deborah J. (Hrsg.) ; Watson, Edward F. (Hrsg.) ; Carson, John S. (Hrsg.) ; Manivannan, Mani S. (Hrsg.): *Proceedings of the 1998 Winter Simulation Conference.* Washington D. C., December 11–16 1998, S. 609–615

[HMS66] Hunt, Earl B. ; Marin, Janet ; Stone, Philip J.: *Experiments in Induction.* New York, London : Academic Press, 1966

[Hol75] Holland, John H.: *Adaption in Natural and Artificial Systems.* Ann Arbor, Michigan : University of Michigan Press, 1975

[Hol99] Hollmén, Jaakko: *Probabilistic Approaches to Fraud Detection*, Helsinki University of Technology, Department of Computer Science and Engineering, Diss., Dezember 1999

[Hor02] Horváth, Péter: *Controlling.* 8., vollständig überarbeitete Auflage. München : Verlag Vahlen, 2002 (Vahlens Handbücher der Wirtschafts- und Sozialwissenschaften). – ISBN 3–8006–2731–0

[HSA99] Heinsohn, Jochen ; Socher-Ambrosius, Rolf: *Wissensverarbeitung: Eine Einführung*. Heidelberg : Spektrum Akademischer Verlag, 1999

[HTd98] Hertz, Alain ; Taillard, Eric ; de Werra, Dominique: Tabu Search. In: Aarts, Emile H. L. (Hrsg.) ; Lenstra, Jan K. (Hrsg.): *Local Search in Combinatorial Optimization*. Chichester : John Wiley & Sons, 1998, S. 121–136

[Int97] Integral Solutions Limited: *Clementine Data Mining System, Reference Manual, Version 4.0*. 1994-1997

[JA90] Jensen, Frank ; Andersen, Stig K.: Approximations in Bayesian Belief Universes for Knowledge-Based Systems. In: *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*. Cambridge, Massachusetts : Morgan Kaufmann Publishers, July 1990, S. 162–169

[JAKA87] Jensen, Finn V. ; Andersen, Stig K. ; Kjærulff, Uffe ; Andreassen, Steen: MUNIN - On the Case for Probabilities in Medical Expert Systems - A Practical Exercise. In: *European Conference on Artificial Intelligence in Medicine*. Marseilles, August 1987, S. 149–160

[Jen96] Jensen, Finn V.: *An introduction to Bayesian networks*. London : University College London Press, 1996

[Jen01] Jensen, Finn V.: *Bayesian networks and decision graphs*. New York : Springer Verlag, 2001

[JG72] Jr., Edward G. C. ; Graham, Ronald L.: Optimal Scheduling for Two-Processor Systems. In: *Acta Informatica* 1 (1972), S. 200–213

[JLO90] Jensen, Finn V. ; Lauritzen, Steffen L. ; Olesen, Kristian G.: Bayesian Updating in Causal Probabilistic Networks by Local Computations. In: *Computational Statistics Quarterly* 4 (1990), S. 269–282

[Kan03] Kantardzic, Mehmed: *Data Mining: Concepts, Models, Methods and Algorithms*. Hoboken, NJ : John Wiley & Sons, 2003

[Keu99] KEUPER, Frank: *Betriebswirtschaftliche Forschung zur Unternehmensführung*. Bd. 37: *Fuzzy-PPS-Systeme: Einsatzmöglichkeiten und Erfolgspotentiale der Theorie unscharfer Mengen*. Wiesbaden : Deutscher Universitäts-Verlag, 1999. – ISBN 3–8244–9023–4

[KG95] KOŠTURIAK, Jan ; GREGOR, Milan: *Simulation von Produktionssystemen*. Wien : Springer Verlag, 1995

[KGW00] KOBYLKA, Andrea ; GÄSE, Thomas ; WIRTH, Siegfried: Datenbankbasierte Generierung von Simulationsmodellen zur Planung von Produktionssystemen. In: SCHULZE, Thomas (Hrsg.) ; LORENZ, Peter (Hrsg.) ; HINZ, Volkmar (Hrsg.): *Simulation und Visualisierung 2000*. Magdeburg : Gruner Druck, März 2000, S. 157–164

[Kjæ94] KJÆRULFF, Uffe: Reduction of Computational Complexity in Bayesian Networks through Removal of Weak Dependences. In: *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence*. San Francisco, California : Morgan Kaufmann Publishers, 1994, S. 374–382

[KKW96] KOCH, Mario ; KUHN, Thomas ; WERNSTEDT, Jürgen: *Fuzzy Control: optimale Nachbildung und Entwurf optimaler Entscheidungen*. München : Oldenbourg Verlag, 1996. – ISBN 3–486–23355–6

[Kle80] KLEITER, Gernot D.: *Bayes-Statistik: Grundlagen und Anwendungen*. Berlin : Walter de Gruyter & Co., 1980

[KN04] KORB, Kevin B. ; NICHOLSON, Ann E.: *Bayesian Artificial Intelligence*. Boca Raton, Florida : vlChapman, 2004

[Knu02] KNUTH, Donald E.: *The art of computer programming: Seminumerical algorithms*. Bd. 2. 3, 10. Nachdruck. Boston : Addison Wesley, 2002. – ISBN 0–201–89684–2

[Koh01] KOHONEN, Teuvo: *Springer series in information sciences*. Bd. 30: *Self-Organizing Maps*. 3. Berlin : Springer Verlag, 2001. – ISBN 3–540–67921–9

[Kom93] KOMMANA, Sreenivasa R.: *Produktionstechnik – Berlin*. Bd. 107: *Wissensbasierte Planung von flexiblen Fertigungssystemen*. München; Wien : Hanser Verlag, 1993. – ISBN 3–446–17476–1

[KR84] Kluge, Paul-Dieter ; Runge, Walter: *Zufallsabhängige Fertigungsprozesse. Methoden zu ihrer Beherrschung.* Berlin : Verlag Die Wirtschaft, 1984

[Kra93] Kratzer, Klaus P.: *Neuronale Netze: Grundlagen und Anwendungen.* 2., durchgesehene Auflage. München : Hanser Verlag, 1993. – ISBN 3–446–17315–3

[Kre02] Krengel, Ulrich: *Einführung in die Wahrscheinlichkeitstheorie und Statistik.* 6., verbesserte. Braunschweig : Vieweg Verlag, 2002. – ISBN 3–528–57259–0

[Kru97] Krug, Wilfried: *Intelligentes Simulations- und Optimierungssystem für Prozesse der Fertigung, Organisation und Logistik – ARENA/ISSOP.* Erlangen : SCS-Europe, 1997

[Kur99] Kursawe, Frank: *Grundlegende empirische Untersuchungen der Parameter von Evolutionsstrategien - Metastrategien.* Dortmund, Universität Dortmund, Diss., 1999

[Kur03] Kurbel, Karl: *Produktionsplanung und -steuerung: Methodische Grundlagen von PPS-Systemen und Erweiterungen.* 5., durchgesehene und aktualisierte. München : Oldenbourg Verlag, 2003 (Handbuch der Informatik)

[Kur05] Kurbel, Karl: *Produktionsplanung und -steuerung im Enterprise Resource Planning und Supply Chain Management.* 6., völlig überarbeitete. München : Oldenbourg Verlag, 2005

[KW00] Kuhn, C. ; Wernstedt, J.: Ein Beitrag zur automatisierten Merkmalsextraktion für Klassifikationssysteme. In: Hafner, S. (Hrsg.) ; Kiendl, H. (Hrsg.) ; Kruse, R. (Hrsg.) ; Schwefel, H.-P. (Hrsg.): *Computational Intelligence im industriellen Einsatz.* Düsseldorf : VDI Verlag, March 11–12 2000 (VDI Berichte 1526), S. 371–386

[Lan04] Lange, Carsten: *Wirtschaftswissenschaftliche Beiträge.* Bd. 192: *Neuronale Netze in der wirtschaftswissenschaftlichen Prognose und Modellgenerierung: Eine theoretische und empirische Betrachtung mit Programmier- Beispielen*. Heidelberg : Phyisca-Verlag, 2004. – ISBN 3–7908–0059–7

[LD60] Land, A. H. ; Doig, A. G.: An automatic method of solving discrete programming problems. In: *Econometrica* 28 (1960), Nr. 3, S. 497–520

[LD94] Li, Zhaoyu ; D'Ambrosio, Bruce: Efficient Inference in Bayes Networks As A Combinatorial Optimization Problem. In: *Approximate Reasoning* 11 (1994), July, Nr. 1, S. 55–81

[LD02] Lacave, Carmen ; Díez, Francisco J.: A Review of Explanation Methods for Bayesian Networks. In: *Knowl. Eng. Rev.* 17 (2002), Nr. 2, S. 107–127. – ISSN 0269–8889

[Len91] Lenz, Andreas: *Knowledge engineering für betriebliche Expertensysteme: Erhebung, Analyse und Modellierung von Wissen.* Wiesbaden : Deutscher Universitäts-Verlag, 1991 (Informatik Hochschulschrift). – ISBN 3–8244–2023–6

[Lid07] Lidokhover, Andrej: *Multi-Agenten-Systeme: Grundlagen, Konzepte, Methoden.* Saarbrücken : vlVDM, 2007

[Lie95] Liebl, Franz: *Simulation. Problemorientierte Einführung.* 2. Auflage. München : Oldenbourg Verlag, 1995

[Lin87] Lindley, Dennis V.: The Probability Approach to the Treatment of Uncertainty in AI and Expert Systems. In: *Statistical Science* 2 (1987), Nr. 1, S. 17–24

[LK00] Law, Averill M. ; Kelton, W. D.: *Simulation modeling and analysis.* 3., international. Boston : McGraw-Hill, 2000 (McGraw-Hill series in industrial engineering and management science). – ISBN 0–07–116537–1

[LMS03] Lorenço, Helena R. ; Martin, Olivier ; Stützle, Thomas: Iterated Local Search. In: Glover, Fred W. (Hrsg.) ; Kochenberger, Gary A. (Hrsg.): *Handbook of Metaheuristics* Bd. 57. Springer Verlag, 2003. – ISBN 1–4020–7263–5

[Loh94] Lohrbach, Thomas: *Göttinger Wirtschaftsinformatik.* Bd. 10: *Einsatz von Künstlichen Neuronalen Netzen für ausgewählte betriebswirtschaftliche Aufgabenstellungen und Vergleich mit konventionellen Lösungsverfahren.* Göttingen : Unitext-Verlag, 1994

[LS02] Li, Xiao Y. ; Stallmann, Matthias F. M.: New Bounds on the Barycenter Heuristic for Bipartite Graph Drawing. In: *Information Processing Letters* 82 (2002), Nr. 6, S. 293–298

[Lud95] Ludyk, Günter: *Theoretische Regelungstechnik 1: Grundlagen, Synthese linearer Regelungssysteme.* Berlin : Springer Verlag, 1995

[MAP02] Metaxiotis, K. S. ; Askounis, Dimitris ; Psarras, John: Expert systems in production planning and scheduling: A state-of-the-art survey. In: *Journal of Intelligent Manufacturing* 13 (2002), August, Nr. 4, S. 253–260

[Meh03] Mehrwald, Christian: *SAP Business Information Warehouse 3: Architektur, Konzeption, Implementierung.* 1.; korrigierter Nachdruck. Heidelberg : dpunkt.verlag, 2003. – ISBN 3–89864–179–1

[Mer90] Mertens, Peter: *Expertensysteme in der Produktion: Praxisbeispiele aus Diagnose und Planung; Entscheidungshilfen für den wirtschaftlichen Einsatz.* München : Oldenbourg Verlag, 1990. – ISBN 3–486–21549–3

[Mer04] Mertens, Peter: *Integrierte Informationsverarbeitung 1: Operative Systeme in der Industrie.* 14., überarbeitete. Wiesbaden : Gabler Verlag, 2004

[MF00] Michalewicz, Zbigniew ; Fogel, David B.: *How to Solve It: Modern Heuristics.* 2. Berlin : Springer Verlag, 2000

[Mic96] Michalewicz, Zbigniew: *Genetic Algorithms + Data Structures = Evolution Programs.* 3. Berlin : Springer Verlag, 1996

[Mil56] Miller, George A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. In: *The Psychological Review* 63 (1956), S. 81–97

[MK97] McLachlan, Geoffrey J. ; Krishnan, Thriyambakam: *The EM Algorithm and its Extensions.* New York : John Wiley & Sons, 1997. – ISBN 0471123587

[MM92] Müller-Merbach, Heiner: *Operations Research: Methoden und Modelle der Optimalplanung.* 10. Nachdruck der 3., durchgesehenen Auflage. München : Verlag Vahlen, 1992 (Vahlens

Handbücher der Wirtschafts- und Sozialwissenschaften). – ISBN 3–8006–0388–8

[MMHT05] Marriott, Kim ; Moulder, Peter ; Hope, Lucas ; Twardy, Charles: Layout of Bayesian Networks. In: *ACSC 05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*. Darlinghurst, Australia : Australian Computer Society, Inc., 2005. – ISBN 1–920–68220–1, S. 97–106

[MN98] Matsumoto, Makoto ; Nishimura, Takuji: Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. In: *ACM Transactions on Modeling and Computer Simulation: Special Issue on Uniform Random Number Generation* Bd. 8, 1998, S. 3–30

[Möl95] Möller, Manfred: *Berichte aus dem Institut für Eisenhüttenkunde*. Bd. 95: *Expertensystem zur Auftragseinplanung bei der Sortenfertigung von Warmband*. Aachen : Shaker Verlag, 1995

[Mon94] Monsler, Bodo: *Aachener Beiträge zu Humanisierung und Rationalisierung*. Bd. 14: *Methode zur Aquisition und Repräsentation von Wissen beim Aufbau eines wissensbasierten Systems zur aktuellen Kapazitätsbelegung in Gießereien*. Aachen : Verlag der Augustinus-Buchhandlung, 1994

[Mön06] Mönch, Lars: *Agentenbasierte Produktionssteuerung komplexer Produktionssysteme*. Wiesbaden : Deutscher Universitäts-Verlag, 2006

[MS98] Munkelt, Torsten ; Schmidt, Lutz: Einsatz von Soft-Computing-Methoden zur Früherkennung von Materialverlusten. In: Kuhl, Jochen (Hrsg.) ; Nissen, Volker (Hrsg.) ; Tietze, Martin (Hrsg.): *Soft Computing in Produktion und Materialwirtschaft. Tagungsband zum 4. Göttinger Symposium Soft Computing*. Göttingen : Cuvillier Verlag, 12. März 1998, S. 63–70

[MTVM93] Maes, Sam ; Tuyls, Karl ; Vanschoenwinkel, Bram ; Manderick, Bernard: Credit Card Fraud Detection Using Bayesian and Neural Networks. In: Maciunas, R. J. (Hrsg.): *Interactive image-guided neurosurgery* American Association Neurological Surgeons, 1993, S. 261–270

[MTVM02] Maes, Sam ; Tuyls, Karl ; Vanschoenwinkel, Bram ; Manderick, Bernard: Credit Card Fraud Detection Using Bayesian and Neural Networks. In: *Proceedings of the 1st International ICSC Conference on Neuro-Fuzzy Technologies (NF 2002)*. Havana, 2002. – ISBN 3–906454–29–0, S. [nur auf CD]

[Muc00] Mucksch, Harry (Hrsg.): *Das Data Warehouse-Konzept: Architektur, Datenmodelle, Anwendungen; mit Erfahrungsberichten*. 4., vollständig überarbeitete und erweiterte Auflage. Wiesbaden : Gabler Verlag, 2000. – ISBN 3–409–42216–1

[Mun95] Mundigl, Robert: *Europäische Hochschulschriften: Reihe 5, Volks- und Betriebswirtschaft*. Bd. 1651: *Ansätze künstlicher neuronaler Netze zur Lösung von Tourenplanungsproblemen*. Frankfurt am Main : Peter Lang Verlag, 1995. – ISBN 3–631–48067–9

[Mun00] Munkelt, Torsten: Einsatz dynamischer Bayes'scher Netze zur Fertigungssteuerung. In: *Computational Intelligence im industriellen Einsatz, Fachtagung des VDI/VDE*. Düsseldorf : VDI Verlag, Mai 2000, S. 295–300

[Mur02] Murphy, Kevin P.: *Dynamic Bayesian Networks: Representation, Inference and Learning*, University of California, Berkley, Diss., 2002

[MVD01] Munkelt, Torsten ; Völker, Sven ; Döring, Thomas: Prognosis, Diagnosis and Control of Production by means of Dynamic Bayesian Networks. In: Fleischmann, Bernhard (Hrsg.) ; Lasch, Rainer (Hrsg.) ; Derigs, Ulrich (Hrsg.) ; Domschke, Wolfgang (Hrsg.) ; Rieder, Ulrich (Hrsg.): *Operations Research Proceedings 2000: Selected Papers of the Symposium on Operations Research (OR 2000)*. Dresden : Springer Verlag, 9.-12. September 2001, S. 97–102

[NB99] Nissen, Volker ; Biethahn, Jörg: Ein Beispiel zur stochastischen Optimierung mittels Simulation und einem Genetischen Algorithmus. In: Biethahn, Jörg (Hrsg.) ; Hummeltenberg, Wilhelm (Hrsg.) ; Schmidt, Bernd (Hrsg.) ; Stähly, Paul (Hrsg.) ; Witte, Thomas (Hrsg.): *Simulation als betriebliche Entscheidungshilfe: State of the Art und neuere Entwicklungen*. Heidelberg : Phyisca-Verlag, 1999, S. 108–125

[Nea04] Neapolitan, Richard E.: *Learning Bayesian Networks*. Upper Saddle River, NJ : Prentice-Hall, 2004

[Nis94] Nissen, Volker: *Evolutionäre Algorithmen: Darstellung, Beispiele, betriebswirtschaftliche Anwendungsmöglichkeiten*. Wiesbaden : Deutscher Universitäts-Verlag, 1994

[NTB05] Nikolov, Nikola S. ; Tarassov, Alexandre ; Branke, Jürgen: In Search for Efficient Heuristics for Minimum-Width Graph Layering with Consideration of Dummy Nodes. In: *Journal of Experimental Algorithmics (JEA)* 10 (2005), S. 2–7. – ISSN 1084–6654

[Ort02] Ortmann, Matthias: *Die Anwendung von Evolutionsstrategien zur adaptiven, echtzeitfähigen Trajektorienplanung von Manipulatorarmsystemen*. Bochum, Ruhr-Universität Bochum, Diss., 2002

[OV04] Ombuki, Beatrice M. ; Ventresca, Mario: Local Search Genetic Algorithms for the Job Shop Scheduling Problem. In: *Applied Intelligence* 21 (2004), July – August, Nr. 1, S. 99–109

[Pag91] Page, Bernd: *Diskrete Simulation – Eine Einführung mit Modula-2*. Berlin : Springer Verlag, 1991

[Pla96] Platz, Jan: *Europäische Hochschulschriften: Reihe 5, Volks- und Betriebswirtschaft*. Bd. 1847: *Computergestützte Planungsverfahren für das Produktionsmanagement auf der Grundlage des Treshold Accepting*. Frankfurt am Main : Peter Lang Verlag, 1996. – ISBN 3–631–49765–2

[Pöp00] Pöppe, Christoph: Optimieren mit Bomben. In: *Spektrum der Wissenschaft* (2000), Mai, S. 8–10

[PPMH94] Pradhan, Malcolm ; Provan, Gregory ; Middleton, Blackford ; Henrion, Max: Knowledge Engineering for Large Belief Networks. In: *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*. San Francisco, CA : Morgan Kaufmann Publishers, 1994, S. 484–490

[Pra97] Prabhu, Narahari U.: *Foundations of Queueing Theory*. Boston : Kluwer Academic Publishers, 1997 (International Series in Operations Research & Management Science: ISOR 7). – ISBN 0–7923–9962–5

[PSS95] Pegden, Claude D. ; Shannon, Robert E. ; Sadowski, Randall P.: *Introduction to Simulation using SIMAN*. 2. New York : McGraw-Hill, 1995

[Pup88] Puppe, Frank: *Einführung in Expertensysteme*. Heidelberg, Berlin : Springer Verlag, 1988 (Studienreihe Informatik). – ISBN 3–540–19481–9, 0–387–19481–9

[Pup91] Puppe, Frank: *Einführung in Expertensysteme*. 2. Heidelberg, Berlin : Springer Verlag, 1991 (Studienreihe Informatik). – ISBN 3–540–54023–7

[Qui93] Quinlan, J. R.: *C4.5: Programs for Machine Learning*. San Mateo, CA : Morgan Kaufmann Publishers, 1993. – ISBN 1–55860–238–0

[Ram98] Ramachandran, Sowmya: *Theory Refinement of Bayesian Networks with Hidden Variables*. Austin, TX, University of Texas, Diss., May 1998

[Rec65] Rechenberg, Ingo: Cybernetic Solution Path of an Experimental Problem / Ministry of Aviation, Royal Aircraft Establishment. Farnborough, United Kingdom, 1965 (1122). – Library Translation

[Rig94] Rigoll, Gerhard: *Kontakt & Studium, EDV-Grundlagen*. Bd. 446: *Neuronale Netze: eine Einführung für Ingenieure, Informatiker und Naturwissenschaftler*. Renningen-Malmsheim : expert-Verlag, 1994. – ISBN 3–8169–0975–2

[Rob73] Robinson, Robert W.: Counting Labeled Acyclic Digraphs. In: Harary, Frank (Hrsg.): *New Directions in the Theory of Graphs*. New York : Academic Press, 1973, S. 239–273

[Rot02] Rothlauf, Franz: *Studies in fuzziness and soft computing*. Bd. 104: *Representations for genetic and evolutionary algorithms, with 53 tables*. Heidelberg : Phyisca-Verlag, 2002. – ISBN 3–7908–1496–2

[RR05] Reichelt, Dirk ; Rothlauf, Franz: Reliable Communication Network Design with Evolutionary Algorithms, paper accepted (to appear). In: *International Journal of Computational Intelligence and Applications (IJCIA), Special Issue on Nature-Inspired Algorithms to Networks and Telecommunications* 0

(2005), October, Nr. 0. – Guest Editors: Yong Xu and Sancho Salcedo-Sanz and Xin Yao

[RS99] Ramoni, Marco ; Sebastiani, Paolo: Learning Conditional Probabilities from Incomplete Data: an Experimental Comparison. In: Heckerman, David (Hrsg.) ; Whittaker, Joe (Hrsg.): *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics (Uncertainty-99)*. San Mateo, CA : Morgan Kaufmann Publishers, 1999

[RS01] Ramoni, Marco ; Sebastiani, Paolo: Robust Learning with Missing Data. In: *Machine Learning* 45 (2001), November, Nr. 2, S. 147—-170

[Sac99] Sachs, Lothar: *Angewandte Statistik. Anwendung statistischer Methoden*. 9. Berlin : Springer Verlag, 1999

[SAS94] Shachter, Ross D. ; Andersen, Stig K. ; Szolovits, Peter: Global Conditioning for Probabilistic Inference in Belief Networks. In: *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA : Morgan Kaufmann Publishers, 1994, S. 514–522

[Sch75] Schwefel, Hans-Paul: *Evolutionsstrategie und numerische Optimierung*. Berlin, Technische Universität Berlin, Fachbereich Verfahrenstechnik, Diss., 1975

[Sch95] Schwarzmaier, Ulrich: *Verfahren zur Produktionsplanung und Produktionssteuerung in industriellen Unternehmen*. Berlin : Duncker und Humblot, 1995 (Abhandlungen aus dem Industrieseminar der Universität Mannheim)

[Sch00] Schneider, Herfried (Hrsg.): *Produktionsmanagement in kleinen und mittleren Unternehmen*. Stuttgart : Schäffer-Poeschel Verlag, 2000

[Sch01] Schöning, Uwe: *Theoretische Informatik – kurzgefasst*. 4. Heidelberg : Spektrum Akademischer Verlag, 2001. – ISBN 3–8274–1099–1

[Sch02a] Scharnbacher, Kurt: *Statistik im Betrieb: Lehrbuch mit praktischen Beispielen*. 13., aktualisierte Auflage. Wiesbaden : Gabler Verlag, 2002 (Gabler-Lehrbuch). – ISBN 3–409–37027–7

[Sch02b] Schulz, Roland: *Parallele und Verteilte Simulation bei der Steuerung komplexer Produktionssysteme.* Ilmenau, Technische Universität Ilmenau, Diss., Oktober 2002

[Sch03] Schuster, Christoph J.: *No-wait Job-Shop-Scheduling: Komplexität und Local Search.* Duisburg, Universität Duisburg-Essen, Fakultät für Naturwissenschaften, Diss., Januar 2003

[SGH97] Spieckermann, Sven ; Griffel, Nils ; Hoffmann, Hartmut: Neues Simulationsmodell bildet Materialfluß im Rohbau eines Automobilherstellers ab. In: *Logistik im Unternehmen* 11 (1997), Nr. 10, S. 80–83

[SGS93] Spirtes, Peter ; Glymour, Clark ; Scheines, Richard: *Causation, Prediction, and Search.* New York : Springer Verlag, 1993

[SGS00] Spirtes, Peter ; Glymour, Clark ; Scheines, Richard: *Causation, Prediction, and Search.* 2nd. Cambridge, Massachusetts : The MIT Press, 2000

[SHF94] Schöneburg, Eberhard ; Heinzmanna, Frank ; Feddersen, Sven: *Genetische Algorithmen und Evolutionsstrategien: eine Einführung in Theorie und Praxis der simulierten Evolution.* Bonn : Addison Wesley, 1994

[SHG90] Schöneburg, Eberhard ; Hansen, Nikolaus ; Gawelczyk, Andreas: *Neuronale Netzwerke: Einführung, Überblick und Anwendungsmöglichkeiten.* Haar bei München : Markt-&-Technik-Verlag, 1990. – ISBN 3–89090–329–0

[Sib98] Sibbel, Rainer: *Fuzzy-Logik in der Fertigungssteuerung am Beispiel der retrograden Terminierung.* Münster : LIT Verlag, 1998 (Ökonomie und Komplexität 1). – ISBN 3–8258–3651–7

[Sie95] Siedentopf, Jukka: The Threshold Waving Algorithm for Job Shop Scheduling / University of Leipzig, Institute of Production Management and Industrial Information Management. Leipzig, Germany, August 1995 (17). – Report

[SR98] Sharda, Ramesh ; Rampal, Rohit: Neural Networks and Management Science/Operations Research: A Bibliographic Essay. In: *Encyclopedia of Library and Information Science* Bd. 61. New York : Marcel Dekker, Inc., 1998, S. 247–259

[SRM96] Spirtes, Peter ; Richardson, T. ; Meek, Christopher: Heuristic Greedy Search Algorithms for Latent Variable Models. In: *Proceedings of the 6th International Workshop on Artificial Intelligence and Statistics*, 1996

[SSGM94] Scheines, Richard ; Spirtes, Peter ; Glymour, Clark ; Meek, Christopher: *TETRAD II: Tools for Causal Modeling, User's Manual and Software*. Hillsdale, NJ : Lawrence Erlbaum Associates, 1994

[SSSWD00] Schrimpf, Gerhard ; Schneider, Johannes ; Stamm-Wilbrandt, Hermann ; Dueck, Gunter: Record Breaking Optimization Results Using the Ruin and Recreate Principle. In: *Journal of Computational Physics* 159 (2000), Nr. 2, S. 139–171. – ISSN 0021–9991

[Sta00] Stahel, Werner A.: *Statistische Datenanalyse: eine Einführung für Naturwissenschaftler*. 3. Braunschweig : Vieweg Verlag, 2000

[Ste93] Steinmann, Dieter: *Beiträge zur Wirtschaftsinformatik*. Bd. 6: *Einsatzmöglichkeiten von Expertensystemen in integrierten Systemen der Produktionsplanung und -steuerung (PPS)*. Heidelberg : Phyisca-Verlag, 1993

[Str05] Stroock, Daniel W.: *An introduction to Markov processes*. Berlin : Springer Verlag, 2005 (Graduate texts in mathematics 230). – ISBN 3–540–23451–9

[Stu26] Sturges, H. A.: The Choice of a Class Interval. In: *Journal of the American Statistical Association* 21 (1926), March, S. 65–66

[Stü99] Stützle, Thomas G.: *Dissertationen zur künstlichen Intelligenz*. Bd. 220: *Local Search Algorithms for Combinatorial Problems – Analysis, Algorithms, and New Applications*. Sankt Augustin : infix, 1999

[Sue92] Suermondt, Henri J.: *Explanation in Bayesian Belief Networks*. Stanford, CA, Stanford University, Departments of Computer Science and Medicine, Report STAN-CS-92-1417, 1992

[Thi01] Thiel, Matthias: *Simulationsbasierte Reihenfolgeplanung in der Halbleiterfertigung*, Technische Universität Ilmenau, Diss., 2001

[Tim94] Timm, Heiko: *Europäische Hochhschulschriften*. Bd. 1571: *Gestaltungsprinzipien zur Erhöhung des Einsatznutzens von Expertensystemen in der betrieblichen Praxis*. Frankfurt am Main : Peter Lang Verlag, 1994. – ISBN 3–631–47445–8

[Tol98] Tolksdorf, Markus: *Entwurf und Implementierung einer Shell für Bayessche Netzwerke*. Würzburg, Bayerische Julius-MaximiliansUniversität Würzburg, Diplomarbeit, Februar 1998

[TTL99] Taudes, Alfred ; Trcka, Michael ; Lukanowicz, Martin: Organizational Learning in Production Networks / Adaptive Information Systems and Modelling in Economics and Management Science, Vienna University of Economics and Business Administration in cooperation with Vienna University of Technology. Vienna, Austria, June 1999 (38). – Forschungsbericht

[VDM01] Völker, Sven ; Döring, Thomas ; Munkelt, Torsten: The generation of large test data for the empirical analysis of heuristic procedures for production planning and control. In: Fleischmann, Bernhard (Hrsg.) ; Lasch, Rainer (Hrsg.) ; Derigs, Ulrich (Hrsg.) ; Domschke, Wolfgang (Hrsg.) ; Rieder, Ulrich (Hrsg.): *Operations Research Proceedings 2000. Selected Papers of the Symposium on Operations Research (OR 2000), Dresden, September 9-12, 2000*. Berlin : Springer Verlag, 2001, S. 266–270

[Vid03] Vidyasagar, Mathukumalli: *Learning and Generalisation: With Applications to Neural Networks*. 2. London : Springer Verlag, 2003 (Communications and Control Engineering). – ISBN 1–85233–373–1

[Völ03] Völker, Sven: *Reduktion von Simulationsmodellen zur simulationsbasierten Optimierung in der Termin- und Kapazitätsplanung*. Frankfurt am Main : Peter Lang Verlag, 2003 (Europäische Hochschulschriften)

[WAS04] Witkowski, Tadeusz ; Antczak, Paweł ; Strojny, Grzegorz: Use Constraint Satisfaction Adaptive Neural Network for Job-Shop Scheduling. In: *Proceedings of the IIIrd International Conference on Advances in Production Engineering (APE 2004)*. Warschau, Juni 2004

[WD80] Wegman, Edward J. (Hrsg.) ; DePriest, Douglas J. (Hrsg.): *Statistical Analysis of Weather Modification Experiments*. New York : Marcel Dekker Inc., 1980

[Web92] Weber, Richard: *Entwicklung und Anwendung von Verfahren zur automatischen Akquisition unsicheren Wissens*. Düsseldorf : VDI Verlag, 1992 (10 211). – ISBN 3–18–141110–8

[Wei02] Weicker, Karsten: *Evolutionäre Algorithmen*. 1. Auflage. Stuttgart : Teubner Verlag, 2002 (Leitfäden der Informatik). – ISBN 3–519–00362–7

[Wek07] Weka: The Waikato Environment for Knowledge Analysis: *Attribute-Relation File Format (ARFF), Version 3.5.1*. May 2007. – URI: `http://weka.sourceforge.net/wekadoc/index.php/en:ARFF_%283.5.1%29`, abgerufen am 29.08.2007 13.15 Uhr

[Wen95] Wendt, Oliver: *Tourenplanung durch Einsatz naturanaloger Verfahren: Integration von Genetischen Algorithmen und Simulated Annealing*. Wiesbaden : Deutscher Universitäts-Verlag, 1995 (Gabler Edition Wissenschaft: Logistik und Verkehr). – ISBN 3–8244–6181–1

[Wer89] Wernstedt, Jürgen: *Experimentelle Prozeßanalyse*. Berlin : Prentice-Hall, 1989

[Wes01] West, Douglas B.: *Introduction to Graph Theory*. 2. Upper Saddle River, NJ : Prentice-Hall, 2001. – ISBN 0–13–014400–2

[WG90] Waschulzik, Thomas ; Geiger, Hans: Theorie und Anwendung strukturierter Konnektionistischer Systeme. In: Dorffner, Georg (Hrsg.): *Konnektionismus in Artificial Intelligence und Kognitionsforschung, 6. Österreichische Artificial-Intelligence-Tagung (KONNAI)* Bd. 252. Salzburg : Springer Verlag, September 1990. – ISBN 3–540–53131–9, S. 143–152

[Whi95] Whittaker, Joe: *Graphical Models in Applied Multivariate Statistics*. Chichester : John Wiley & Sons, 1995 (Wiley Series in Probability and Mathematical Statistics). – ISBN 0–471–91750–8

[Win00] Winkler, Gerhard: Stochastische Prozesse in der statistischen Modellierung / GSF - Forschungszentrum für Umwelt und Gesundheit GmbH. Oberschleißheim, Oktober 2000. – Forschungsbericht

[YW01] Yang, Shengxiang ; Wang, Dingwei: A New Adaptive Neural Network and Heuristics Hybrid Approach for Job-Shop Scheduling. In: *Computers and Operations Research* 28 (2001), Januar, Nr. 10, S. 955–971

[Zäp82] Zäpfel, Günther: *Produktionswirtschaft: operatives Produktionsmanagement.* Berlin : Walter de Gruyter & Co., 1982

[Zel00] Zell, Andreas: *Simulation neuronaler Netze.* 3., unveränderter Nachdruck. München : Oldenbourg Verlag, 2000. – ISBN 3–486–24350–0

[Zie98] Ziegenbein, Klaus: *Controlling.* 6., überarbeitete und erweiterte. Ludwigshafen (Rhein) : Friedrich Kiel Verlag, 1998 (Kompendium der praktischen Betriebswirtschaft). – ISBN 3–470–70596–8

[Zim93] Zimmermann, Hans-Jürgen (Hrsg.): *Fuzzy Technologien: Prinzipien, Werkzeuge, Potentiale.* Düsseldorf : VDI Verlag, 1993

[Zim95] Zimmermann, Hans-Jürgen (Hrsg.): *Datenanalyse: Anwendung von DataEngine mit Fuzzy Technologien und Neuronalen Netzen.* Düsseldorf : VDI Verlag, 1995