Situativ trainierte Regeln zur Ablaufsteuerung in Fertigungssystemen und ihre Integration in Simulationssysteme

Dissertation
zur Erlangung der Würde eines
Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)
der Universität Paderborn

vorgelegt von
Dipl.-Wirt.-Inf. Mark Aufenanger

Paderborn, August 2009

Referent: Prof. Dr.-Ing. habil. Wilhelm Dangelmaier
Korreferentin: Prof. Dr. Leena Suhl

Vorwort

Geht es um das Schreiben, so gehört nicht zuletzt das Vorwort zu den schöneren Abschnitten. Zum einen für den Autor, denn dann ist der Rest des Buches bereits fertig. Zum anderen für die Leser, denn hier erfahren sie nicht selten wie aufwändig und schwierig das Unterfangen des Autors war und wer ihm unterstützend zu Seite gestanden hat. Es ist auch die Zeit danke zu sagen.

Meinem Doktorvater Herrn Prof. Dr.-Ing. habil. Wilhelm Dangelmaier gilt besonderer Dank. Er hat mir die Möglichkeit zur Promotion gegeben. Mit seiner Unterstützung und den Hilfestellungen sowie den stets hilfreichen Anregungen hat er zum Gelingen dieser Arbeit beigetragen. Herzlich bedanken möchte ich mich auch bei Frau Prof. Dr. Leena Suhl für die Übernahme des Zweitgutachtens. Zudem gebührt mein Dank Herrn Prof. Dr. Stefan Betz und Herrn Prof. Dr. Dennis Kundisch, die mir als Mitglieder der Promotionskommission wertvolles Feedback gegeben haben.

Den Studenten Herrn Matthias Giese, Herrn Nedim Lipka M.Sc., Herrn Dipl.-Inf. Patrick van Lück, Herrn Yi Tan M.Sc. und Herrn Dipl.-Wirt.-Inf. Hendrik Varnholt, die viel Arbeit und Zeit in die Implementierung investiert haben.

Ebenfalls sei ganz herzlich Herrn Dr. rer. pol. Jörg Habich gedankt. Er hat mir, in mir unzählig vorkommenden Situationen, stets mit Rat und Tat zur Seite gestanden. Dabei hat er so einige Wochenenden für Diskussionen mit mir geopfert. Danke Jörg.

Lieber Dank gebührt Herrn Prof. Dr. theol. habil. Hans Jürgen Brandt, der mich seit dem Beginn meines Studium ständig unterstützte und stets mit „Herr Doktor“ begrüßte seitdem ich promovierte. Was ich bis jetzt immer mit „Noch nicht Herr Professor“ verneinen musste. Die Zeit ist nun vorbei. Dankeschön Onkel Jürgen.

Dank auch Herrn Prof. Dr. theol. Karl Hengst, der mich während meiner Zeit als Doktorand immer unterstützt hat.

Auch sei Herrn Dieter Sander gedankt, der mich auf die Idee brachte, das Abitur anzugehen. Onkel Dieter, das war ein guter Rat, denn hiernach ergab sich das Studium.

Ein ganz besonders lieber Dank gilt meinen Eltern Monika und Ferdinand – meiner Familie insgesamt.

Katrin, Liebe meines Lebens! Für den Dank, den ich dir entgegenbringen möchte gibt es keine Worte.

Paderborn, im November 2009 Mark Aufenanger

Abstrakt

Die Arbeit „Situativ trainierte Regeln zur Ablaufsteuerung in Fertigungssystemen und ihre Integration in Simulationssysteme“ widmet sich dem sensiblen unternehmerischen Problembereich der Optimierung der Fertigung. Schwerpunkt hierbei ist die zeitkritische Auftragsreihenfolgeplanung.

Das entwickelte Verfahren kann dem Planungsproblem der Maschinenbelegung schnell eine gute eine Lösung zuführen. Es werden aktive Ablaufpläne erzeugt, welche den zu durchsuchenden Lösungsraum erheblich einschränken, da der im Sinne der verfolgten Zielfunktion optimale Ablaufplan immer auch ein aktiver ist. Hierzu bildet das Verfahren in Situationen mit Entscheidungsnotwendigkeit eine Konfliktmenge der in Frage kommenden Auftragskandidaten. Um diese Menge passend aufzulösen, wurde ein maschinelles Lernverfahren als situativer wissensbasierter Entscheider integriert. Dieser trifft über die Beschreibung von Steuerungssituationen die Entscheidung für die in der jeweiligen Situation passendste Steuerungsregel. Das Verfahren wird damit durch den Lösungsraum gesteuert und nur ein Pfad, der zu einer guten Lösung führt, durchlaufen. Für das Training des wissensbasierten Entscheiders wird der Zeitraum vor Beginn des Fertigungsprozesses genutzt. Das Verfahren ist parallel zur laufenden Fertigung adaptierbar.

Im Rahmen der Evaluierung des Verfahrens mit alternativen Methoden der Ablaufplanung und -steuerung anhand von Benchmark-Problemen konnten sehr gute Ergebnisse erzielt werden.

Leidenschaft braucht man nicht, Willen braucht man.
(Helmut Schmidt)

Für Katrin & Mich

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

α	Art und Anordnung der zur Verfügung stehenden Maschinen eines Fertigungssystems
β	Charakteristika der Arbeitsaufträge (Jobs)
γ	Optimalitätskriterium (Zielfunktion)
σ	Lösungskandidat einer Probleminstanz I
$c_{j,m}^{st}$	Fertigstellungszeitpunkt des Jobs j auf Maschine m auf Fertigungsstufe st
$C_j(I, \sigma)$	Fertigstellungszeitpunkt des Lösungskandidaten σ für die Probleminstanz I
C_{max}	Zielfunktion: Minimierung der maximalen Durchlaufzeit (Makespan)
CL	Menge der verwendeten Klassen cl_i
cl_i	Klasse der Menge CL
CM	Anzahl der Maschinen in der Konfliktmenge
CS	Konfliktmenge bestehend aus um eine Ressource konkurrierenden Jobs bzw. Operationen
$d(o)$	Fertigstellungzeit einer Operation o
d_j	Liefertermin eines Jobs j
D_Π	Menge der Probleminstanzen
d_{min}	Minimum der Fertigstellungzeitpunkte
dl_j	Verspätung eines Jobs j zum aktuellen Zeitpunkt
$dmlf_{st}$	Mittlere Auslastung aller Maschinen der Fertigungsstufe st
E/T_{max}	Zielfunktion: Minimierung der Summe aller Terminabweichungen
$F(c_i)$	Geometrisches Mittel aus Präzision $prec(c_i)$ und Recall $rec(c_i)$
$f(o)$	Fertigstellungzeit einer Operation o
f_j	Fertigstellungzeit des Jobs j
f_{min}	Minimale Fertigstellungzeit
$Fort(st)$	Position der Fertigungsstufe st im Verhältnis zu allen Fertigungsstufen FS
$Fortschritt(j)$	Bearbeitungsfortschritt des Jobs j
FQk	Flexible-Flow-Shop mit bis zu k uniformen Maschinen auf den Fertigungsstufen

FRk	Flexible-Flow-Shop mit bis zu k unverwandten Maschinen auf den Fertigungsstufen
FS	Anzahl der Fertigungsstufen in einem Flexible-Flow-Shop
I	Eine Probleminstanz, $I \in D_\Pi$
J	Job-Shop
j	Index der Aufträge einer Stufe st
jst_j	Frühester Starttermin von Job j
K	Durchschnittliche Kapazitätsauslastung eines Fertigungssystems
K_{min}	Zielfunktion: Maximierung der durchschnittlichen Kapazitätsauslastung
L_m	Leerzeit einer Maschine m
L_{max}	Zielfunktion: Minimierung der Summe aller Leerzeiten
LB	Untere Schranke
M	Anzahl der zur Verfügung stehenden Maschinen
m	Index der Maschinen
$M(o)$	Maschine, die Operation o bearbeitet
$M_{i,st}$	Maschine i auf Fertigungsstufe st
M_{st}	Anzahl Maschinen auf Stufe st
$mb(m)$	Anzahl der belegten Zeiteinheiten einer Maschine m bis zum aktuellen Zeitpunkt
mr_m^{st}	Freigabetermin von Maschine m auf Stufe st
mst_m	Früheste Freigabe von Maschine m
N	Anzahl einzuplanender Jobs
$N(o)$	Menge der Operationen o technologisch als nächstes nachfolgenden Operation
O	Menge aller noch einplanbarer Operationen
o'	Aus der Konfliktmenge CS ausgewählte Operation
$o_{j,m}^{st}$	Operationszeit von Job j auf Maschine m der Stufe st
o_j^{st}	Standard-Bearbeitungszeit des Jobs j auf Stufe st
$o_{j,m}$	Operationszeit von Job j auf Maschine m
o_{min}	Minimale Fertigstellungszeit von Operation o
o_{prio}	Operation o, die von Prioritätsregel $prio$ ausgewählt wird
$of(j)$	Anzahl der bereits verplanten Operationen des Jobs j
$oj(j)$	Anzahl aller Operationen des Jobs j
or	Index der technologischen Reihenfolge einen Jobs
$P(\sigma)$	Lösungspfad des Lösungskandidaten σ durch den Zustandsraumgraphen
$p_{j,m}^{st}$	Bearbeitungszeit des Jobs j auf Maschine m auf Fertigungsstufe st

$p_{j,m}$	Bearbeitungszeit von Job j auf Maschine m
$pf(j)$	Bereits verplante Bearbeitungszeit des Jobs j
$pj(j)$	Gesamte Bearbeitungszeit des Jobs j
$pjc(j)$	Bearbeitungszeit eines Jobs j
$prec(c_i)$	Präzision als Maß für die relevanten Klassen c_i im Klassifikationsergebnis (Trefferquote eines Klassifizierers)
$prio$	Eine Prioritätsregel
ps_j^{st}	Standard-Bearbeitungszeit des Jobs j auf Stufe st
pt	Bearbeitungszeit einer Operation
pt_j	Gesamte Restbearbeitungszeit eines Jobs j
$pw(o)$	Prioritätswert von Operation o
pw_{max}	Höchster Prioritätswert unter den vergebenen Werten
$r(o)$	Frühestmöglicher Startzeitpunkt einer Operation o
r_i	Bereitstellungszeitpunkt der Operationen
r_j	Freigabetermin von Job j
$rec(c_i)$	Recall als Maß der gefundenen Klassen c_i im Verhältnis zur gesamten Anzahl der Klassen
$Rest(j)$	Restliche Bearbeitungszeit des Jobs j ab der aktuellen Fertigungsstufe
S	Menge der aktuell einplanbaren Operationen
s	Zustand eines Fertigungssystems bzw. Knoten im Zustandsraumgraphen
S_v	Partieller Schedule im Knoten v
$S_{\Pi}(I)$	Lösungskandidaten eines Problems Π
sd_j	Summe der Bearbeitungszeiten der bereits eingeplanten Operationen o eines Jobs j
sf_j	Stufenbasierter Fortschritt von Job j
SPI	Menge der Teilprobleminstanzen einer Shop-Probleminstanz
st	Eine Stufe des Fertigungssystems, $st \in FS$
$t(o)$	Frühestmöglicher Startzeitpunkt einer Operation o
T_{max}	Zielfunktion: Minimierung der Summe aller Verspätungen
UB	Obere Schranke
$v_{j,m}^{st}$	Relative Maschinengeschwindigkeit der Maschine m auf Fertigungsstufe st in Abhängigkeit von Job j
v_i	Ein Zustand im Zustandsraumgraph
wl_m	Auslastung einer Maschine m

1. Einleitung

Obwohl Job-Shops (Werkstattfertigungen) und Flexible-Flow-Shops (flexible Fließfertigungen) seit langem untersucht werden, fehlen bislang immer noch praxistaugliche Steuerungsverfahren.[1] Steuerungsnotwendigkeit besteht immer dann, wenn mehrere Arbeitsaufträge um die Zugehörigkeit zu einem Arbeitsmittel konkurrieren oder ein Auftrag von mehreren Arbeitsmitteln bearbeitet werden kann.[2] Daher ist im Rahmen der Ablaufplanung und -steuerung die Bestimmung der Auftragsreihenfolge (Scheduling) von besonderer Relevanz. Scheduling ist die Erzeugung und / oder Umsetzung von Abarbeitungsreihenfolgen für Aufträge. Ergebnis ist der Ablaufplan (Schedule). Ein Ablaufplan ist die Vorschrift der zeitlichen Abläufe von Produktionsprozessen, insbesondere die zeitliche Zuordnung der Aufträge zu den Maschinen (Maschinenbelegung). Damit gibt der Ablaufplan die Bearbeitungsreihenfolge der Aufträge an.

Aufgrund der Unsicherheiten in einem Fertigungssystem können Ablaufpläne jedoch in den seltensten Fällen wie geplant ausgeführt werden; unvorhergesehene Ereignisse können einen einmal erzeugten Plan unzulässig machen.[3] Rescheduling ist der Prozess, einen existierenden Schedule zu aktualisieren bzw. zu erneuern, wenn sich ein oder mehrere Parameter des Problems ändern. *Vieira et al.* beschreiben diesen Rescheduling-Sachverhalt als Aufgabe der Produktionsplaner, die in dynamischen und stochastischen Fertigungssystemen Lösungen erzeugen müssen. Jedoch ist das Ermitteln einer guten Lösung unter den genannten Bedingungen schwierig: „[...] determining the best rescheduling solution still remains an open research issue and, consequently, is the most difficult part of the rescheduling process.“[4] So werden heute in der Produktionsplanung vor dem Start des Produktionsprozesses intensive Überlegungen über den Ablauf,[5] im Betrieb jedoch nur noch die Durchsetzung und Steuerung nach fest definierten Regeln unternommen.[6] Das Problem der Produktionswirtschaft besteht demzufolge darin,

[1] Vgl. (Vieira et al., 2003).
[2] Vgl. (Schekelmann, 1999, Seite 4) und (Vitiello, 1997, Seite 12 f.).
[3] Vgl. (Georgi, 1995, Seite 92).
[4] (Vieira et al., 2003)
[5] Der Ablauf wird beispielsweise mittels eines mathematischen Verfahrens oder einer Heuristik simuliert.
[6] Die Regeln werden ohne Bezug zur aktuellen Ausprägung des Fertigungssystems angewendet.

dass ein geeignetes Verfahren zum Rescheduling in Job-Shops und Flexible-Flow-Shops fehlt.

In dieser Arbeit soll eine Umgebung geschaffen werden, mit der vor und parallel zum laufenden Produktionsbetrieb in Job-Shops und Flexible-Flow-Shops Handlungsalternativen erarbeitet und bewertet werden können, um bei Steuerungsnotwendigkeit angemessen entscheiden zu können. Zunächst erfolgt hierzu in Kapitel 2 die Konkretisierung der betrachteten Organisationsformen. Job-Shops zeichnen sich u. a. durch eine hohe Flexibilität und weite Verbreitung im praktischen Einsatz als auch in der Literatur aus. Die hier verwendeten Methoden liefern somit gute Vergleichsmöglichkeiten mit dem entwickelten Verfahren. Flexible-Flow-Shops werden gerade in kapitalintensiven Fertigungen immer häufiger angewendet und sind auf einer Fertigungsstufe den Job-Shops sehr ähnlich. Daher ist in Flexible-Flow-Shops der dezentrale Einsatz des entwickelten Verfahrens evaluierbar. Anschließend werden die Bestandteile bei der Ablaufplanung und -steuerung in Job-Shops und Flexible-Flow-Shops angeführt, der Rahmen und die Anforderungen an ein mögliches Lösungsverfahren dargelegt. Das Kapitel schließt mit der Ableitung der Problemstellung. Kapitel 3 beleuchtet den für den Untersuchungskontext dieser Arbeit relevanten Stand der Technik. Hierzu werden existierende Verfahren und Konzepte dargestellt und auf Erfüllung der Anforderungen an ein Lösungsverfahren hin untersucht.[7] Das vierte Kapitel verbindet den relevanten Stand der Technik mit der dargelegten Problemstellung zu den sich hieraus ergebenden Arbeitspaketen, um das aufgeworfene Problem sukzessiv zu lösen. Die Aufgliederung der Arbeitspakete orientiert sich an dem im zweiten Kapitel dargestellten Rahmen eines Lösungsverfahrens sowie den erarbeiteten Anforderungen. Kapitel 5 widmet sich der Konzeptionsaufgabe zur Lösung der Problemstellung mittels situativ trainierter Regeln zur Ablaufplanung und -steuerung in Fertigungssystemen. Zunächst werden die Grundlagen des Verfahrens dargelegt, der Ablauf des Verfahrens in verschiedene Phasen untergliedert und die erforderlichen Schnittstellen vorgestellt. Damit ist das Verfahren für die speziellen Ausprägungen in Job-Shops und Flexible-Flow-Shops einfach zu konfigurieren und leicht erweiterbar. Anschließend werden die Hauptbestandteile entwickelt. Neben der Konzeption der allgemeinen Heuristik werden die wissensbasierte Methode zur Auswahl der Steuerungsregeln sowie situationsbeschreibende Merkmale und Steuerungsregeln selbst beschrieben. Das Kapitel schließt mit dem Konzept zur Rückkopplung. Innerhalb von Kapitel 6 wird das entwickelte Lösungsverfahren evaluiert, um dessen Zweckmäßigkeit darzulegen. Dazu werden die in den einzelnen Experimenten zu Job-Shops und Flexible-Flow-Shops erreichten Ergebnisse aufgezeigt und die generelle Funktionsfähigkeit des

[7] Dabei kann es einen Erkenntniszuwachs nur geben, wenn bestehende Verfahren falsifiziert und deren Grenzen aufzeigt werden, um auf dieser Basis ein neues Verfahren zu entwickeln (vgl. Popper, 2002).

Verfahrens vorgestellt. Anschließend wird die Leistung des entwickelten Verfahrens in Job-Shops dargestellt. Die Ergebnisse beim dezentralen Verfahrenseinsatz in Flexible-Flow-Shops beenden das Kapitel. Ein Fazit schließt die Untersuchung in Kapitel 7 ab.

2. Problemstellung

2.1. Ablaufplanung und -steuerung in Job-Shops und Flexible-Flow-Shops

2.1.1. Betrachtete Organisationsformen

Fertigung ist die Herstellung physischer Erzeugnisse mittels fortschreitender, diskontinuierlicher und zielgerichteter Transformationen in einem abgegrenzten System, dem Fertigungssystem.

„Ein Fertigungssystem ist eine technisch, organisatorisch (und kostenrechnerisch) selbständige Allokation von Potentialfaktoren zu Fertigungszwecken. Es besteht aus (elementaren) Arbeitssystemen, die die kleinste Einheit einer Kombination der Potentialfaktoren Betriebsmittel und Arbeitskräfte darstellen und eine oder mehrere Klassen von Transformationen durchführen können.“[1]

Job-Shops und Flexible-Flow-Shops sind verschiedene Problemklassen für Organisationsformen der Fertigung.

Job-Shops

Job-Shops (Werkstattfertigungen) sind nach dem Verrichtungsmerkmal zentralisiert.[2] Das hier betrachtete Job-Shop-Problem ist ein Problem der Art $J||C_{max}$.[3] Im Fall der

[1] (Dangelmaier, 2001, Seite 5)

[2] Vgl. (Dangelmaier, 2001, Seite 79).

[3] Die Drei-Felder-Notation $\alpha|\beta|\gamma$ wird zur Beschreibung von Problemklassen der Ablaufplanung und -steuerung verwendet. Der α-Wert kennzeichnet Art und Anordnung der zur Verfügung stehenden Maschinen. Charakteristika der Aufträge werden durch den β-Wert dargestellt und der γ-Wert beschreibt das Optimalitätskriterium (Zielfunktion). Eine Beschreibung der Drei-Felder-Notation ist in Anhang A.9 zu finden. Für weiterführende Informationen zu der auf (Graham et al., 1979) zurückgehenden $\alpha|\beta|\gamma$-Notation sei auf (Bräsel et al., 2003; Leung, 2004; Brucker, 2007; Blazewicz et al., 2007) verwiesen.

Plananpassung während des laufenden Produktionsprozesses wird das Problem zu einem $J|r_i|C_{max}$, bei dem die Bereitstellungszeiten r_i der Aufträge nicht alle Null sind. Nachfolgend werden die Eigenschaften der untersuchten Job-Shop-Umgebung vorgestellt:

- Job-Shop mit mehreren Arbeitsmitteln (Maschinen) $m = \{1, \ldots, M\}$ $(M \geq 2)$, die unterschiedliche Aufträge (Jobs) $j = \{1, \ldots, N\}$ bearbeiten können.
- Jede Maschine m kann zur gleichen Zeit höchstens einen Auftrag j bearbeiten.
- Aufträge bestehen aus Operationen o. Dabei ist $o_{j,m}$ die Bearbeitungszeit des Auftrags j auf Maschine m.
- Transportzeiten zwischen den Maschinen sind zu vernachlässigen.
- Jeder Auftrag $j \in N$ ist zur gleichen Zeit höchstens von einer Maschine $m \in M$ bearbeitbar.
- Flexible Fertigungsumgebung mit vernachlässigbaren Rüstzeiten und -kosten.
- Ausreichend große Lagerkapazitäten.

Dem System ist ein hohes Maß an Unsicherheit inhärent, so dass ein vor Beginn der Fertigung erstellter Ablaufplan während der Fertigung in der Regel mehrfach anzupassen ist.

Flexible-Flow-Shops

Flexible-Flow-Shops (flexible Fließfertigungen) werden nach dem Objektmerkmal zentralisiert.[4] Das hier betrachtete Flexible-Flow-Shop-Problem ist ein Problem der Art $FQk||C_{max}$ bzw. $FRk||C_{max}$. Im Fall von erforderlichen Plananpassungen während des laufenden Produktionsprozesses wird das Problem zu einem $FQk|r_i|C_{max}$ bzw. $FRk|r_i|C_{max}$, bei dem die Bereitstellungszeiten r_i der Jobs nicht alle Null sein müssen. Die Eigenschaften der untersuchten Flexible-Flow-Shop-Umgebung werden nachfolgend vorgestellt:

- Flexible-Flow-Shop mit in der Regel mehreren Fertigungsstufen $st = \{1, \ldots, FS\}$ $(FS \geq 2)$.
- In der Regel mehr als eine Maschine $m = \{1, \ldots, M_{st}\}$ $(M_{st} \geq 1)$ pro Fertigungsstufe st. Maschinen können nur einen Auftragstyp bearbeiten.

[4] Vgl. (Dangelmaier, 2001, Seite 79).

- Aufträge bestehen aus Operationen o. Dabei ist $o_{j,m}^{st}$ die Bearbeitungszeit des Auftrags j auf Maschine m auf Stufe st.
- Ausprägung 1: $FQk||C_{max}$ bzw. $FQk|r_i|C_{max}$ mit uniformen Maschinen auf den Fertigungsstufen. Uniforme Maschinen haben unterschiedliche Bearbeitungsgeschwindigkeiten. Dabei ist das Verhältnis der Bearbeitungsgeschwindigkeiten unabhängig von den Jobs immer gleich.
- Ausprägung 2: $FRk||C_{max}$ bzw. $FRk|r_i|C_{max}$ mit unverwandten Maschinen pro Stufe. Die Bearbeitungsgeschwindigkeit der Maschinen ist unterschiedlich und wird von den Jobs determiniert.
- Jede Maschine m kann zur gleichen Zeit höchstens einen Job $j = \{1, \ldots, N\}$ bearbeiten.
- Jeder Job j ist zur gleichen Zeit höchstens von einer Maschine m bearbeitbar.
- Transportzeiten zwischen den Maschinen sind zu vernachlässigen.
- Flexible Fertigungsumgebung mit vernachlässigbaren Rüstzeiten und -kosten.
- Ausreichend große Lagerkapazitäten zu Beginn und am Ende der Fertigung sowie zwischen den Fertigungsstufen.
- Kapitalintensive Fertigung komplexer Produkte.

Vor Beginn der Fertigung erstellte Ablaufpläne sind während der Fertigung in der Regel mehrfach anzupassen, da dem System ein hohes Maß an Unsicherheit inhärent ist.

2.1.2. Zielfunktionen der Ablaufplanung und -steuerung

Im Rahmen der Ablaufplanung und -steuerung verfolgte Ziele werden über Zielfunktionen abgebildet.[5]

[5]Die Optimierungsziele im Bereich der Ablaufplanung und -steuerung gehören zur Klasse der kombinatorischen Optimierungsprobleme. Ein **kombinatorisches Optimierungsproblem** Π ist entweder ein Minimierungs- oder ein Maximierungsproblem und besteht aus den folgenden Teilen: Einer Menge D_Π von Probleminstanzen I, jeweils einer endlichen Menge von **Lösungskandidaten** $S_\Pi(I)$ für jede der Probleminstanzen $I \in D_\Pi$ und einer Zielfunktion f_Π, die jeder Probleminstanz $I \in D_\Pi$ und jedem Lösungskandidaten $\sigma \in S_\Pi(I)$ eine **Lösungsgüte** $f_\Pi(I, \sigma) \in \mathbb{Q}^+$ zuweist. Ist Π ein Minimierungsproblem, dann ist die optimale Lösung einer Instanz $I \in D_\Pi$ ein Lösungskandidat $\sigma^* \in S_\Pi(I)$, so dass für alle $\sigma \in S_\Pi(I)$ gilt: $f_\Pi(I, \sigma^*) \leq f_\Pi(I, \sigma)$. Ist Π ein Maximierungsproblem, dann ist die optimale Lösung einer Instanz $I \in D_\Pi$ ein Lösungskandidat $\sigma^* \in S_\Pi(I)$, so dass für alle $\sigma \in S_\Pi(I)$ gilt: $f_\Pi(I, \sigma^*) \geq f_\Pi(I, \sigma)$.

Eine Zielfunktion (Zielkriterium) bewertet die Entscheidungen. Die Zielfunktion bringt Entscheidungen in eine (besser-schlechter) Reihenfolge. So gestatten z. B. Kostenfunktionen als Zielfunktionen die Kostenbewertung von Fertigungsentscheidungen.[6]

Um die nachfolgend dargestellten Zielfunktionen zu skizzieren, sind folgende Aspekte zu berücksichtigen. Sei $I \in D_\Pi$ als eine Probleminstanz I für ein Problem Π mit den Lösungskandidaten $S_\Pi(I)$ gegeben. Die Probleminstanz I besteht aus N Aufträgen, wobei sich für jeden Auftrag j durch einen Lösungskandidaten σ ein Fertigstellungszeitpunkt $C_j(I, \sigma)$ ergibt. Um eine optimale Lösung eines Scheduling-Problems mit einem Optimierungskriterium zu finden, das einer regulären Zielfunktion unterliegt, reicht es nach *French* aus, die Menge der Lösungskandidaten auf aktive Schedules[7] zu beschränken, da die Menge der semiaktiven Schedules[8], die der aktiven enthält und der optimale Schedule immer auch ein aktiver ist.[9] Eine reguläre Zielfunktion $f_\Pi(I, \sigma)$ verhält sich in allen Variablen $C_j(I, \sigma)$ mit $j \in \{1, \ldots, N\}$ monoton wachsend. Es muss gelten:

$$C_j(I, \sigma) \leq C_j(I, \sigma') \text{ mit } j \in \{1, \ldots, N\} \Rightarrow f_\Pi(I, \sigma) \leq f_\Pi(I, \sigma')$$

Wie Abbildung 2.1 illustriert, ist der optimale Schedule immer ein aktiver Schedule.

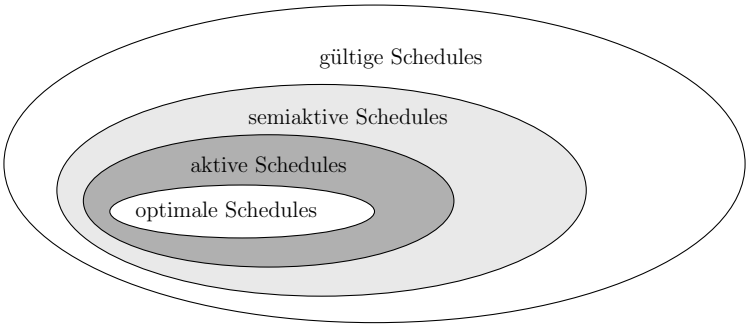


Abbildung 2.1.: Taxonomie von Ablaufplänen[10]

Zwischen den verschiedenen Zielen lassen sich konkurrierende, komplementäre und indifferente Beziehungen zueinander unterscheiden.[11] Zwei Ziele sind *konkurrierend*, wenn

[6] In Anlehnung an (Schneeweiß, 2002, Seite 108). Vgl. auch (Domschke et al., 1997, Seite 291 ff.) und (Zäpfel und Braune, 2005, Seite 18 f.).

[7] Ein Schedule ist immer dann aktiv, wenn es nicht möglich ist, den Beginn irgendeiner Operation, eventuell durch Vertauschung der Auftragsreihenfolge vorzuverlegen, ohne dabei den Beginn mindestens einer anderen Operation zu verzögern (vgl. Zäpfel und Braune, 2005, Seite 29).

[8] Ein Schedule ist immer dann semiaktiv, wenn keine Operation vorher als im Plan vorgesehen durchführbar ist, ohne dabei eine der Nebenbedingungen zu verletzen oder die Reihenfolge der Bearbeitung der Operationen auf den Maschinen zu verändern (vgl. Zäpfel und Braune, 2005, Seite 9).

[9] Vgl. (French, 1982).

[10] Vgl. (Stosik, 2005, Seite 49).

[11] Hierbei existiert zwischen konkurrierenden Zielen einen Zielkonflikt.

mit der Verschlechterung (Verbesserung) der einen Zielgröße die andere verbessert (verschlechtert) wird. Ziele sind *komplementär* zueinander, wenn mit der Verbesserung (Verschlechterung) der einen Zielgröße die andere ebenfalls verbessert (verschlechtert) wird. Komplementäre Zielgrößen sind substituierbar. Dabei kann eine Zielgröße durch die andere ersetzt werden.[12] Beeinflussen sich Ziele weder konkurrierend noch komplementär, so sind diese als *indifferente* Ziele zu bezeichnen.

Die Optimierung konkurrierender Produktionsziele, auch als Dilemma der Produktionssteuerung bezeichnet, wurde bereits 1951 von *Gutenberg* aufgezeigt:[13] Maximale Kapazitätsauslastung kann beispielsweise nur durch hohe Bestände erreicht werden, was wiederum zu hohen Durchlaufzeiten führt. Jedoch benötigt die Minimierung der mittleren Durchlaufzeit geringe Bestände. Es werden folgende Zielsetzungsgruppen unterschieden:[14]

Kapazitätsorientierte Zielsetzungen

Kapazitätsorientierte Ziele sollen eine möglichst hohe produktive Ausnutzung der vorhandenen Ressourcen (Maschinen) gewährleisten. Nachfolgend werden exemplarisch einige kapazitätsorientierte Ziele aufgeführt.

L_{max} *Minimierung der Summe aller Leerzeiten*

Sei $p_{j,m}$ die Bearbeitungszeit von Job j auf Maschine m. Die Leerzeit L_m einer Maschine m ist die Summe der Zeitspannen, in denen m keinen Auftrag innerhalb der Belegungszeit des ganzen Fertigungssystems durch den gegebenen Auftragsbestand bearbeitet:

$$L_m = \text{Makespan}^{15} - \sum_{j=1}^{N} p_{j,m}$$

Es handelt sich um ein Minimierungsproblem, der Lösungskandidat mit dem geringsten Wert der gegebenen Zielfunktion wird gesucht. Für die Minimierung der Summe der Leerzeiten ist die Zielfunktion:

$$f_{\Pi}(I, \sigma) : \sum_{m=1}^{M} L_m$$

[12] „Minimierung der Zykluszeit“ und „Minimierung der maximalen Durchlaufzeit“ bilden beispielsweise zwei komplementäre Zielgrößen (vgl. Rixen, 1997, Seite 18).
[13] Vgl. (Gutenberg, 1951). Vgl. auch (Kurbel und Rohmann, 1995, Seite 20) und (Wiendahl, 2008, Seite 252 f.).
[14] Vgl. (Domschke et al., 1997).
[15] Zur Definition des Makespans siehe durchlaufzeitbezogene Zielsetzungen.

K_{min} *Maximierung der durchschnittlichen Kapazitätsauslastung*

Die durchschnittliche Kapazitätsauslastung K eines Fertigungssystems ist:

$$K = \frac{\sum_{j=1}^{N} \sum_{m=1}^{M} p_{j,m}}{M}$$

Für die Maximierung der durchschnittlichen Kapazitätsauslastung ist die Zielfunktion:

$$f_{\Pi}(I, \sigma) : \frac{K}{\text{Makespan}^{15}}$$

Terminorientierte Zielsetzungen

Bei terminorientierten Zielen wird die Zielsetzung verfolgt, mit Kunden vereinbarte Termine möglichst gut einzuhalten. Dazu bewerten terminorientierte Ziele den Auftragsbestand bezüglich dessen Einhaltung der Zieltermine. Hierzu zählen z. B. die folgenden Zielsetzungen.

T_{max} *Minimierung der Summe aller Verspätungen*

Für einen Auftrag j ist der Liefertermin d_j gegeben. Für die Minimierung der Summe aller Verspätungen ist die Zielfunktion:

$$f_{\Pi}(I, \sigma) : \sum_{j=1}^{N} max\{C_j(I, \sigma) - d_j; 0\}$$

E/T_{max} *Minimierung der Summe aller Terminabweichungen*

Die Minimierung der Summe aller Terminabweichungen berücksichtigt verfrühte und verspätete Fertigstellungen und hat die Zielfunktion:

$$f_{\Pi}(I, \sigma) : \sum_{j=1}^{N} |C_j(I, \sigma) - d_j|$$

Die Zielfunktion T_{max} ist regulär, E/T_{max} ist nicht regulär, da die verfrühte und die verspätete Fertigstellung negativ gewichtet werden.

Durchlaufzeitbezogene Zielsetzungen

Bei durchlaufzeitbezogenen Zielen wird die Zielsetzung verfolgt, durch einen schnellen Produktionsfluss die auftragsbedingten Kapitalbindungskosten zu minimieren.

Durchlaufzeit ist die Zeitdauer, die zwischen dem Zugang und dem Abgang eines Arbeitsauftrags in einem bestimmten Arbeitssystem vergeht.[16]

Eng mit der Durchlaufzeit sind die Termintreue, die Kapazitätsauslastung sowie die Produktionskosten und -leistungen verknüpft. Bei funktionaler Betrachtung steht die Planung der Durchlaufzeit im Zentrum produktionswirtschaftlicher Überlegungen. Mit frei werdenden Ressourcen ist daher insgesamt mehr Wertschöpfung zu erzielen.[17] Das Streben, die Produktion möglichst schnell durchzuführen, zeigt sich in der Forderung nach kurzen Durchlaufzeiten.[18]

Ausgehend von der Durchlaufzeit wird bei der „Minimierung der maximalen Durchlaufzeit“ ein Schedule gesucht, bei dem die Zeit für die Durchführung aller notwendigen Operationen minimal ist.

C_{max} *Minimierung der maximalen Durchlaufzeit (Makespan[19])*
: Ziel ist es, die maximale Durchlaufzeit (Makespan) – vom Beginn der Bearbeitung des erstens Auftrags – der N Aufträge in I zu minimieren. Der Makespan sei $C_{max}(I,\sigma) = max\{C_1(I,\sigma), C_2(I,\sigma), \ldots, C_N(I,\sigma)\}$ für einen Lösungskandidaten σ. Die Zielfunktion für die Minimierung der maximalen Durchlaufzeit ist:

$$f_{\Pi}(I,\sigma) : C_{max}(I,\sigma)$$

Für das Minimierungsproblem der mittleren Durchlaufzeit ist die Zielfunktion:

$$f_{\Pi}(I,\sigma) : \frac{1}{N}\sum_{j=1}^{N} C_j(I,\sigma)$$

Ein Großteil von Veröffentlichungen im Bereich der Ablaufplanung und -steuerung in Fertigungssystemen verwendet die reguläre Zielfunktion C_{max}, die zur Sicherstellung der Vergleichbarkeit auch in dieser Arbeit verwendet wird.[20] Das heißt, im weiteren Verlauf dieser Arbeit wird das durchlaufzeitbezogene Optimierungsziel der „Minimierung der maximalen Durchlaufzeit“ (Makespan) als primäres Ziel betrachtet.

[16] Vgl. (Kern et al., 1996, Seite 362).
[17] Vgl. (Schneeweiß, 2002, Seite 265).
[18] Vgl. (Günther und Tempelmeier, 2005, Seite 3).
[19] Der Makespan wird auch als schedule length, Zykluszeit oder Gesamtbearbeitungszeit bezeichnet (vgl. Domschke et al., 1997, Seite 292).
[20] Vgl. u. a. (Adams et al., 1988; Bierwirth, 1995; Jain und Meeran, 1999; Jansen et al., 1999).

2.1.3. Komplexität des Reihenfolgeplanungsproblems

Da für die Bestimmung einen Schedules die Komplexität ein entscheidender Faktor ist, sind bei der Entwicklung von Verfahren zur Ablaufsteuerung komplexitätstheoretische Aspekte zwingend zu beachten. Beispielsweise sind Job-Shop-Scheduling-Probleme der Art $J||C_{max}$ mit 2 Maschinen ($M = 2$) stark $\mathcal{NP}$-vollständig und schwach $\mathcal{NP}$-vollständig für $M = 2$ und $n_j \leq 3$ oder für $M = 3$ und $n_j \leq 2$ für alle $j \in \{1, \ldots, N\}$, mit n_j als Anzahl der Operationen des Auftrags j, N Aufträgen und M Maschinen. Probleme mit $M = 2$ und $n_j \leq 2$ für alle $j \in \{1, \ldots, N\}$ sind in polynomialer Zeit lösbar.[21] Gleiches gilt z. B. auch für Flexible-Flow-Shop Scheduling-Probleme, für welche die Zugehörigkeit zur Klasse der $\mathcal{NP}$-vollständigen Probleme als sicher gilt.[22] Einerseits existieren also Klassen von Probleminstanzen von Scheduling-Problemen, die effizient lösbar sind. Andererseits gibt es Klassen von Probleminstanzen, für die kein effizienter Algorithmus existiert. Nach *Jain und Meeran* hat ein effizienter Algorithmus die Eigenschaft, eine optimale Lösung zu konstruieren, wobei die Anforderungen des Algorithmus an Laufzeit und Speicherplatz mit der Eingabegröße des Problems polynomial ansteigen.[23]

Bei Scheduling-Problemen handelt es sich im Regelfall um ganzzahlige bzw. gemischtganzzahlige Modelle. Diese führen im Gegensatz zu nichtganzzahligen Optimierungsmodellen zu einer kombinatorischen Explosion. Für ein Job-Shop-Scheduling-Problem ist die Anzahl gültiger aktiver Schedules mittels der Formel $(N!)^M$ berechenbar.[24] Tabelle 2.1 zeigt das Wachstum exemplarisch. Schon für kleine, aus wenigen Maschinen (M) und Aufträgen (N) bestehende Scheduling-Probleme existiert eine sehr große Anzahl von möglichen Wertekombinationen, die selbst mit aktueller IT-Unterstützung nicht in vertretbaren Zeiten zu enumerieren sind.[25]

Hopp und Spearman geben ein weiteres Beispiel zur Veranschaulichung der Problemkomplexität. Sie untersuchen die erforderliche Rechenzeit zur Problemlösung in Abhängigkeit der Problemgröße. Hierbei wird die Rechenzeit zwischen zwei Computersystemen verglichen. Ein langsames System, das 1.000.000 Sequenzen pro Sekunde vergleichen kann und ein zweites, 1.000-mal schnelleres System. Bereits bei 20 Aufträgen brauchen

[21] Vgl. u. a. (Brucker, 2007; Blazewicz et al., 2007; Pinedo, 2005; Brucker und Knust).
[22] Vgl. (Huang et al., 2004; Chang und Liao, 1992).
[23] Vgl. (Jain und Meeran, 1998). Vorausgesetzte Grundlagen der Komplexitätstheorie sind beispielsweise in (Garey und Johnson, 1990) und (Cormen et al., 2001) nachzulesen.
[24] Jede Maschine $m \in M$ muss jeden Auftrag $j \in N$ genau einmal bearbeiten. Für jede Maschine ergeben sich $N!$ verschiedene Reihenfolgen, in welcher die Aufträge bearbeitbar sind. Diese Möglichkeit ist für jede der Maschinen vorhanden, weswegen $N!$ mit der Anzahl der Maschinen M zu potenzieren ist (vgl. Zäpfel und Braune, 2005, Seite 21).
[25] Vgl. (Suhl und Mellouli, 2006, Seite 9).

(N, M)	Anzahl möglicher Schedules
$(2, 2)$	4
$(3, 3)$	216
$(5, 5)$	24.883.200.000
$(10, 10)$	$3,95940866122425193243875570782 67e + 65$
$(20, 10)$	$7,26517648789896838337820155001 72e + 183$
$(100, 50)$	$3,16360851496287831932031224687 51e + 7898$
$(100, 100)$	$1,00084188359456282948226356925 22e + 15797$

Tabelle 2.1.: Komplexität von Scheduling-Problemen

beide Systeme ca. 77 Jahre, um die optimale Lösung des Problems zu finden.[26] Das Problem potenziert sich konsequenterweise bei Flexible-Flow-Shops.[27]

2.2. Rahmen eines Ablaufplanungs- und Steuerungsverfahrens

Auf dieser Grundlage der bisherigen Ausführungen kann jetzt der Aufbau eines Lösungsverfahrens dargelegt werden. Zur Entwicklung eines geeigneten Ablaufplanungs- und Steuerungsverfahrens wird dieses in Komponenten unterteilt. Anschließend kann die Entwicklung der Anforderungen an die Komponenten vorgenommen werden. Mit den aufgezeigten Anforderungen ist im Kapitel 3 der Stand der Technik auf geeignete Verfahren zu bewerten bzw. Defizite bestehender Methoden aufzuzeigen.

2.2.1. Aufbau eines Lösungsverfahrens

Abbildung 2.2 illustriert den Aufbau eines Lösungsverfahrens.

Die *Steuerungskomponente* stellt den Rahmen des Verfahrens dar und generiert den grundlegenden Schedule. Im Allgemeinen werden Algorithmen zur Lösung komplexer Probleme häufig mit Strategien verbessert, die auf Hypothesen und Vermutungen aufbauen und mit höherer Wahrscheinlichkeit die Lösungsfindung beschleunigen. Derartige Strategien werden als Heuristiken bezeichnet. Dies sind typischerweise Faustregeln früher beobachteter Eigenschaften von Lösungen oder das Nachbilden des menschlichen

[26] Vgl. (Hopp und Spearman, 2007, Seite 522 ff.). Bei dem Experiment wird vorausgesetzt, dass jede der möglichen Lösungen zu betrachten ist, um das Optimum zu finden.
[27] Zur Komplexität von Flexible-Flow-Shop-Problemen vgl. u. a. (Brucker, 2007, Seite 150 ff.).

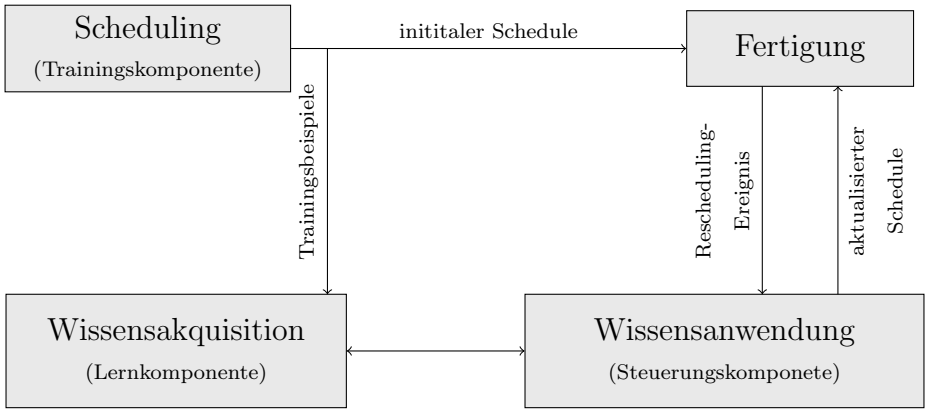


Abbildung 2.2.: Aufbau eines Lösungsverfahrens

Problemlösungsprozesses.[28] Demzufolge wird eine Art Wissen zur Erstellung von Lösungen angewendet. Um zu guten Lösungen zu kommen, muss die Steuerungskomponente in der Lage sein, den Lösungsraum grundsätzlich einzuschränken, sowie Situationen mit der Notwendigkeit einer Steuerungsentscheidung darzustellen. Beim Durchlaufen des Lösungsraums sind also gute Entscheidungen zu treffen.

Dazu sind nach *Aytug et al.* geeignete Wissensrepräsentationsformen und dynamische Lernmethoden erforderlich.[29] Ein derartiger Ansatz ist auf die Ablaufplanung und -steuerung in Fertigungssystemen übertragbar.[30] Dem folgend soll eine *Lernkomponente* (wissensbasierter Klassifikationsmechanismus) in die Steuerungskomponente eingesetzt werden, um bei Steuerungsnotwendigkeit passende Entscheidungen zu treffen. Dazu trainiert die Lernkomponente die Entscheidungen vor Beginn des Produktionsprozesses und kann anschließend auf dieser Basis gute Steuerungsentscheidungen treffen. Eingabe ist die Situation des Fertigungssystems, die Ausgabe eine passende Steuerungsentscheidung. Die Lernkomponente soll eine dynamische Methode sein, damit das Wissen aus speziellen Lösungen für ähnliche Probleminstanzen verwendet werden kann und permanent durch neue Trainingsbeispiele anpassbar ist. Im Kontext der Ablaufplanung und -steuerung entstehen ähnliche Probleme immer dann, wenn sich Parameter wie Bearbeitungszeiten, Anzahl der Maschinen und neue Aufträge der ursprünglichen Instanz nicht übermäßig stark verändern. Dieses ist beim Rescheduling der Fall. Für die Lernkomponente muss daher die grundsätzlich Annahme gelten, dass ähnliche Probleminstanzen auch ähnlich zu lösen sind. Auch muss es möglich sein, die wissensbasierte Lernkomponente einfach in die Steuerungskomponente zu integrieren, nicht zuletzt um die Lernkomponente ohne großen Aufwand austauschen zu können. Allerdings muss

[28] Vgl. (Engesser, 1988, Seite 264).
[29] Vgl. (Aytug et al., 1994).
[30] Vgl. (Wang, 2007; Tavakkoli-Moghaddam und Daneshmand-Mehr, 2005).

die Steuerungskomponente zur Ablaufplanung und -steuerung auch ohne eine Lernkomponente eingeschränkt funktionsfähig sein. Obwohl eine Lernkomponente also nicht zwingend erforderlich ist, ist diese zu integrieren, da die Güte der erzeugten Lösungen zur Ablaufplanung und -steuerung ohne Lernkomponente gering ist.

Damit die Lernkomponente in Steuerungssituationen wissensbasiert entscheiden kann, ist eine Wissensbasis zu erstellen, es sind Trainingsbeispiele zu erzeugen. Die *Trainingskomponente* löst beispielhaft Scheduling-Probleminstanzen und übergibt diese als Trainingslösungen. Das bedeutet, dass das zum Training verwendete Fertigungssystem mit dem für den Wissenseinsatz verwandt ist.[31] Hierbei sollte ein Verfahren verwendet werden, das möglichst gute Lösungen erzeugt. Da das Training vor dem Start des Fertigungsprozesses durchgeführt wird, steht „ausreichend“ Zeit zur Generierung einer Vielzahl von guten Trainingsbeispielen zur Verfügung. Das beispielhafte Lösen von Scheduling-Probleminstanzen zum Training ist von verschiedenen Verfahren durchführbar (Ablaufsimulatoren, Heuristiken, optimierende Verfahren).

Die Unterscheidung in eine Steuerungs-, Lern- und Trainingskomponente hat mehrere Vorteile. Zum einen können die Komponenten autark entwickelt werden und sind in einem gewissen Rahmen selbstständig lauffähig. Da die Trainings- und die Lernkomponente direkt agieren, erscheint es sinnvoll die Trainings- in die Lernkomponente zu integrieren, um so den indirekten fehleranfälliger Datenaustausch durch wenige Schnittstellen zu vermeiden. Zum anderen sind ganze Komponenten oder einzelne Teile leicht austauschbar. So kann z. B. das integrierte Trainingsverfahren durch ein anderes ersetzt werden, das bessere Beispiele liefert.

2.2.2. Anforderungen an eine Steuerungskomponente

2.2.2.1. Reduktion der Komplexität

Zur Bewältigung der in Abschnitt 2.1.3 problematisierten Komplexität des zu lösenden Planungs- und Steuerungsproblems in Job-Shops und Flexible-Flow-Shops ist die

[31] Als verwandte Systeme können beispielsweise zwei Job-Shops bezeichnet werden, da sie die grundlegend gleiche Organisationsform besitzen. Ein Job-Shop zum Training und ein Open-Shop als Anwendungssystem sind z. B. zwei unverwandte Systeme. Hieraus wird jedoch ersichtlich, dass das Trainieren auf beispielsweise einem nach dem Job-Shop-Prinzip angeordneten Fertigungssystem mit zehn Maschinen für die Wissensanwendung auf einem Job-Shop mit 500 Maschinen zu wahrscheinlich weniger guten Ergebnissen führt als auf einem Fertigungssystem mit elf Maschinen.

Steuerungspositionierung von entscheidender Bedeutung. Damit sollen komplexe Entscheidungsprobleme derart reduziert werden, dass die Entscheidungsfindung in akzeptabler Zeit erfolgen kann. Dies gilt insbesondere für Flexible-Flow-Shops, da hier vor jeder Fertigungsstufe Entscheidungen der Ablaufplanung und -steuerung zu treffen sind. Im Normalfall ist der Anteil der Planung in zentralen Steuerungssystemen höher als in dezentralen. Grundsatzidee der dezentralen Ansätze ist die Verteilung von Entscheidungen. Hierbei ist jedoch anzumerken, dass aufgrund der dezentralen Steuerungseingriffe die Systemdynamik verändert wird.[32]

Zentral

Die Steuerung stellt in der Logistik ein elementares Instrument dar. Im Rahmen dieser Steuerung sind im Vorfeld der Ausführung Entscheidungen über unterschiedliche Systemparameter und -variablen des gesamten Fertigungssystems möglich. Auf dieser Basis kann das System anschließend im mathematischen Sinne als Optimierung konfiguriert werden. Bestandteile der zentralen Planung und Steuerung sind: Eine Zielfunktion, die vom Verfahren bestmöglich zu erreichen ist, Variablen, von denen die jeweilige Zielfunktion abhängig ist, und die bei der Lösungsbestimmung einzuhaltenden Restriktionen.[33]

Bei einer zentralen Planung und Steuerung werden sämtliche Entscheidungen des Systems zentral an einer einzigen Stelle getroffen. In komplexen und dynamischen Fertigungssystemen ist dieses jedoch aus mehreren Gründen oft nicht möglich. Zunächst ist das Entscheidungsproblem zu komplex,[34] woraus eine hohe Rechenzeit resultiert. Gleichzeitig steht im Fertigungsprozess nicht genügend Zeit zur Berechnung einer optimalen Lösung zur Verfügung. Auch sind die benötigten Informationen nicht immer zum Zeitpunkt der Entscheidung vorhanden. Daher wird in zentral gesteuerten Systemen zur Erreichung eines stabilen Systemverhaltens eine suboptimale Leistung akzeptiert. Aufgrund der oft unrealistischen Annahmen in den Modellen einer zentralen Steuerung sind derartige Verfahren nicht im Untersuchungskontext einsetzbar.[35] Ebenso können sie keine situativen Entscheidungen treffen.

Dezentral

Zur Verbesserung der Stabilität und Robustheit, somit der Effizienz, empfiehlt sich

[32] Vgl. (Scholz-Reiter et al., 2008, Seite 122).
[33] Vgl. (Scholz-Reiter et al., 2008, Seite 123).
[34] Siehe Abschnitt 2.1.3.
[35] Vgl. (Hopp und Spearman, 2007, Seite 524 f.).

der Rückgriff auf dezentrale Ansätze.[36] *Scholz-Reiter et al.* erklären die Einführung von dezentralen Steuerungen in Logistiksystemen damit, dass in komplexen und dynamischen Systemen eine augenblickliche optimale Entscheidungsfindung häufig nicht möglich ist.[37]

Im Falle von dezentralen Steuerungsstrategien wird von Selbststeuerung der Produktion gesprochen. „Selbststeuerung setzt voraus, dass interagierende Elemente in nichtdeterministischen Systemen die Fähigkeit und Möglichkeit zum autonomen Treffen von Entscheidungen besitzen. Ziel des Einsatzes von Selbststeuerung ist eine höhere Robustheit und positive Emergenz des Gesamtsystems durch eine verteilte, flexible Bewältigung von Dynamik und Komplexität.“[38] Die Idee der Dezentralisierung besteht darin, Entscheidungen zu verteilen, um einzelne Entscheidungsfunktionen von einer zentralen Instanz auf einzelne Steuerungspunkte in Job-Shops und Flexible-Flow-Shops zu verlagern. Damit können z. B. Maschinen oder Jobs aufgrund von lokalen Informationen Entscheidungen über die Reihenfolge bei der Bearbeitung oder die Route durch das Maschinennetz treffen.[39] Durch die Dezentralisierung ist das komplexe Entscheidungsproblem soweit zu reduzieren, dass augenblickliche Steuerungsentscheidungen möglich werden. Damit lässt sich unter Inkaufnahme von global ungünstigen Ergebnissen schnell und flexibel auf kurzfristige Änderungen oder nicht vorhersagbare Ereignisse reagieren.

Aufgrund der Überlegenheit der dezentralen Ansätze sollte die Steuerungskomponente ebenfalls einen dezentralen Einsatz ermöglichen.

2.2.2.2. Bewältigung von Unsicherheiten

Im Zusammenhang mit Job-Shops und Flexible-Flow-Shops ist der Planungs- und Steuerungszeitpunkt, d. h. die Berücksichtigung von Unsicherheiten von Relevanz.[40] Eine Anpassung des initialen Plans ist immer dann erforderlich, wenn etwas zum ursprünglichen Planungszeitpunkt Unbekanntes wie Maschinenausfälle, fehlende Montageteile oder Materialmängel in der Fertigung auftritt.[41] In derartigen Fällen ist die verfügbare Zeit zur Überarbeitung bzw. Neuerstellung eines Planes oder zum Treffen einer Steuerungsentscheidung gering.

[36] Vgl. (Zäpfel, 1998, Seite 17 ff.).
[37] Vgl. (Scholz-Reiter et al., 2008, Seite 123).
[38] (Scholz-Reiter et al., 2007)
[39] Derartige Entscheidungen werden häufig durch relativ simple Regeln getroffen (vgl. u. a. (Günther und Tempelmeier, 2005, Seite 224) und (Reese, 1996, Seite 868 f.)).
[40] Zur grundsätzlichen Bewältigung von Unsicherheiten in der Produktionsplanung vgl. (Schneeweiß, 2002, Seite 104 ff.).
[41] Zu weiteren Rescheduling-Ereignissen vgl. u. a. (Vieira et al., 2003).

Proaktiv

Eine proaktive Ablaufplanung und -steuerung berücksichtigt mögliche Unsicherheiten schon bei der initialen Planerstellung vor dem Beginn des Fertigungsprozesses. Ein Schedule ist proaktiv, wenn dieser für eine bestimmte Zeitspanne vorausschauend erstellt ist. Unter Berücksichtigung der Nebenbedingungen wird versucht, die Zielfunktion zu optimieren. Aufgrund der unvollkommenen Information zum Planungszeitpunkt wird zunächst von einer statischen Planungsumgebung gesprochen.[42] Durch die bestehenden Unsicherheiten sind viele Systeme allerdings sehr dynamisch. Daher sind derartige Probleme bei der proaktiven Ablaufplanung und -steuerung zunächst in statische Probleme zu transformieren. Liegen beispielsweise stochastische Bearbeitungszeiten vor, kann der ungünstigste anzunehmende Fall (Worst-Case-Fall) einberechnet werden, um einen unter allen Umständen ausführbaren Plan zu erhalten.[43] Dadurch lässt sich ein Rescheduling vermeiden.[44] Doch treten unter besser als zum Planungszeitpunkt angenommenen Umständen unnötige Kosten auf, da zur Verfügung gehaltene Ressourcen nicht optimal genutzt werden. Diese Kosten sind durch eine genauere Planung zwar vermeidbar, jedoch nur auf Kosten von Systemausfällen.

Reaktiv

Bei der reaktiven Planung und -steuerung wird ein partieller bzw. temporär gültiger Schedule erzeugt, in Situationen mit Steuerungsnotwendigkeit zum Entscheidungszeitpunkt ad hoc entschieden. Dabei kann durchaus ein Ablaufplan erstellt werden, der abgearbeitet wird, solange sich keine entscheidungsrelevanten Parameter verändern. Im Falle von Änderungsereignissen wird der Plan dann durch einen neuen ersetzt. *Sauer* merkt hierzu an, dass die Reaktion auf Änderungen möglichst schnell erfolgen muss.[45] Durch die reaktive Steuerungsmethode entstehen in vielen Fällen jedoch schlechte Entscheidungen bzgl. des langfristigen Fertigungshorizonts. *Aytug et al.* betonen: „In a strictly reactive situation [...], the future is taken as it occurs – everything is a surprise and there is no active mitigation of possible side-effects and problems.“[46]

Proaktiv-reaktiv

Aufgrund der Nachteile der dargestellten proaktiven und reaktiven Ansätze bietet sich oft eine Kombination aus beiden an. Diese proaktiv-reaktiven Verfahren lassen sich in

[42] Vgl. (Sauer, 2002, Seite 12).
[43] Vgl. (Aytug et al., 2005).
[44] Gravierende Defekte des Fertigungssystems ausgenommen. Es wird also von normalen Umständen ausgegangen.
[45] Vgl. (Sauer, 2002, Seite 17).
[46] (Aytug et al., 2005)

zwei Schritte unterteilen. Zunächst wird nach proaktiver Vorgehensweise ein initialer Ablaufplan für einen bestimmten Zeithorizont erstellt. Dieser ist statisch und braucht Unsicherheiten gar nicht oder nur in geringem Maß zu berücksichtigen. Während der Abarbeitung des Plans findet ständig ein Soll-Ist-Vergleich statt. Bei Abweichungen wird eine reaktive Plananpassung bzw. Planerneuerung durchgeführt.[47] Abbildung 2.3 zeigt das Schema eines solchen Ablaufs. Grundsätzlich ist bei derartigen Verfahren zu klären: Wann und wie ist die Plananpassung durchzuführen? *Vieira et al.* bezeichnen diese Fragestellung als „rescheduling policies".[48]

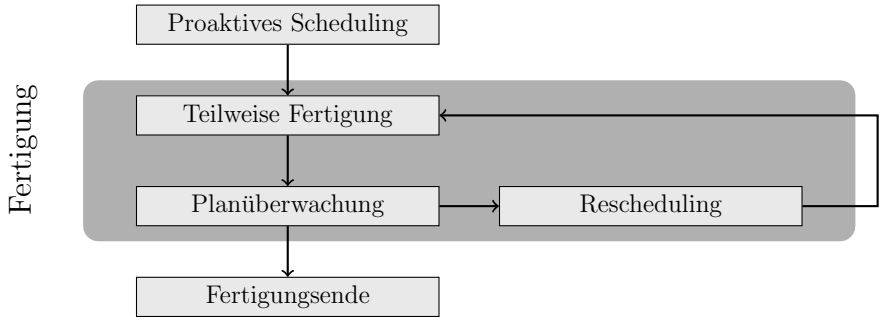


Abbildung 2.3.: Schema eines proaktiv-reaktiven Verfahrens zur Ablaufsteuerung

Proaktiv-reaktive Verfahren bieten ausreichende Flexibilität. Im Idealfall finden fast alle Planungsschritte vor dem Beginn des Produktionsprozesses statt, so dass die Planung das Fertigungssystem nur in Ausnahmefällen verlangsamt.

Lawrence und Sewell zeigen, dass in unsicheren und schnelle Entscheidungen fordernden Fertigungssystemen reaktive Planungs- und Steuerungsverfahren den Offline-Techniken wie proaktiven Ansätzen überlegen sind.[49] Als Ablaufplanungs- und Steuerungsverfahren der in dieser Arbeit zu betrachteten Job-Shops und Flexible-Flow-Shops kommen daher (proaktiv-)[50] reaktive Verfahren zum Einsatz.[51] Nachteil derartiger Verfahren ist aber, dass sie nur auf lokalen Betrachtungen der aktuellen Fertigungssituation beruhen. Durch die Kombination einer Steuerungs- und Lernkomponente kann dieser Nachteil

[47] Vgl. (Aytug et al., 2005; Sabuncuoglu und Bayiz, 2000; Church und Uzsoy, 1992).
[48] Vgl. (Vieira et al., 2003).
[49] Vgl. (Lawrence und Sewell, 1997). Vgl. auch (Wan, 1995), der zeigt, dass bei zufälligen Bearbeitungszeiten ein globaler offline Algorithmus zu schlechteren Ergebnissen im Vergleich zu prioritätsregelbasierten Lösungen führt. Eine weitere Übersicht zu reaktiven und proaktiv-reaktiven Ansätzen geben u. a. (Aytug et al., 2005).
[50] Hierbei wird die proaktive Phase (insbesondere der dabei erzeugte Ablaufplan) als gegeben vorausgesetzt. Mittels des initial erstellten Ablaufplanes wird das Verfahren für den reaktiven Einsatz, das vollständige Erneuern des Planes, trainiert.
[51] Ein Ablaufplan wird generiert. Dieser wird, sobald sich signifikante Änderungen ergeben, verworfen und ein neuer Plan generiert. Umrüstkosten, Transportwege usw. werden hierbei nicht mit in Betracht gezogen. Dabei werden der proaktive Verfahrensteil sowie der Soll-Ist-Vergleich und die Auslösung der Umplanung als gegeben vorausgesetzt.

jedoch mit einem globalen Plan im Vorfeld umgangen werden. Zusätzlich wird somit das Wissen über die proaktive Planerstellung gesichert; der Aufwand zur initialen Planerstellung ist bei Änderungen nicht umsonst.

2.2.2.3. Minimierung der Rechenzeit

Während der Anpassung des Plans werden die Ressourcen des Fertigungssystems blockiert.[52] Es steht wenig Zeit zum Generieren eines neuen Ablaufplans zur Verfügung. Aus diesem Grund muss eine Plananpassung derart schnell durchgeführt werden, dass der neue Plan besser ist als der initial erstellte.[53] „Fast computation methods are needed for the heuristics of [...] scheduling problems in practical manufacturing environments.“[54] Beispiel 2.1 zeigt den Zusammenhang zwischen der erforderlichen Rechenzeit zur Planerstellung und dem Makespan, der durch den neu berechneten Plan erreichbar ist.

Beispiel 2.1 Rechenzeit und Makespan
Das Job-Shop-Beispiel verdeutlicht den Einfluss der Rechenzeit eines Ablaufplanungs- und Steuerungsverfahrens auf den Makespan (C_{max}). Ausgangspunkt ist der in Abbildung 2.4 dargestellte Gantt-Chart[55] des initialen Schedules. Hier sind die Operationen ($o_{j,m}$) von drei Aufträgen ($j = 1, \ldots, 3$) eingeplant, die von drei Maschinen ($m = 1, \ldots, 3$) bearbeitet werden.

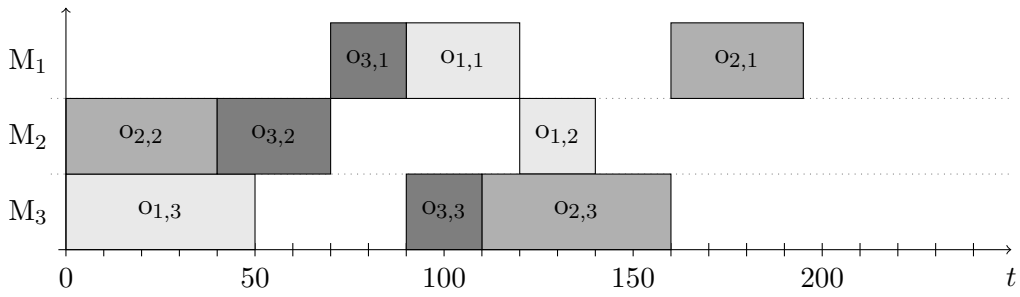


Abbildung 2.4.: Beispielhafter initial gegebener Schedule

[52] Werden während der Umplanung Ressourcen neu belegt, kann der erstellte Plan ggf. nicht mehr angewendet werden. Ist die Plananpassung also nicht schnell genug, gehen im laufenden Fertigungssystem Handlungsoptionen verloren.
[53] Vgl. (Mahajan, 2007).
[54] (Li et al., 2006)
[55] Der Ökonom Henry L. Gantt gilt als Namensgeber für Gantt-Diagramme. Diese sind Balkendiagramme und stellen die Belegungsdauer von Maschinen durch Aufträge in Form von Balken auf der Zeitachse (Abszisse) dar. Es kann zwischen zwei Diagrammtypen unterschieden werden. Auftragsorientierte Gantt-Diagramme bilden die Aufträge auf der Ordinate ab, maschinenorientierte Gantt-Diagramme die Maschinen. An Ersten lassen sich Arbeitsfortschritte der Aufträge gut erkennen, wohingegen Produktions- und Stillstandzeiten der Maschinen an Letzteren gut zu erkennen sind.

Zum Zeitpunkt $t = 120$ tritt ein Ereignis mit Steuerungsnotwendigkeit auf. Es wird ein neuer Auftrag 4 ($j = 4$) mit den Operationen $o_{4,1} = 20$, $o_{4,2} = 20$ und $o_{4,3} = 25$ hinzugefügt. Ein schnelles Ablaufplanungs- und Steuerungsverfahren, das für die Neuberechnung eines Schedules zehn Zeiteinheiten benötigt, könnte den in Abbildung 2.5 illustrierten Schedule erstellen.

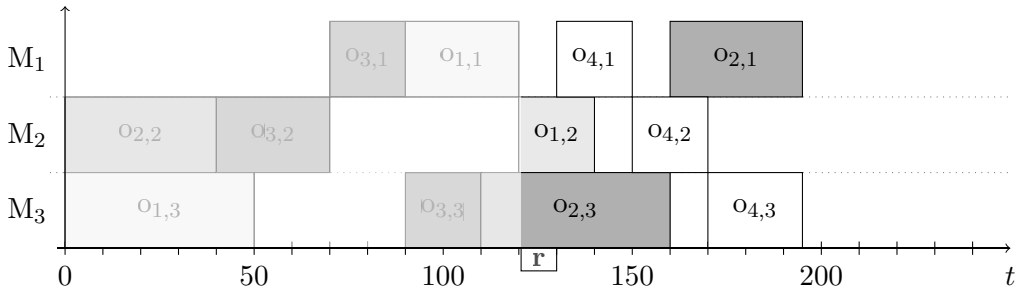


Abbildung 2.5.: Effizienter Schedule nach Umplanung

Wenn das Ablaufplanungs- und Steuerungsverfahren einen vollständig neuen Schedule für diejenigen Operationen berechnet, die sich nicht in Bearbeitung finden, dürfen während der Berechnung keine weiteren Operationen ihre Bearbeitung starten. Beginnt eine Operation während des Umplanungsprozesses, ist diese mittels des Ablaufplanungs- und Steuerungsverfahrens nicht mehr umplanbar. Vor der Berechnung des neuen Schedules ist noch unklar, welche Startzeit passend (optimal) ist; auch sind Unterbrechungen nicht erlaubt. Darüber hinaus muss es ein zeitliches Intervall geben, in dem das Verfahren terminiert. Dem Gantt-Diagramm ist zu entnehmen, dass der neue Schedule den Makespan im Vergleich zu dem initialen Schedule nicht verlängert. Die Anwendung des Verfahrens erfordert wenig Zeit für die Berechnung, was sich positiv auf das Rescheduling-Ergebnis auswirkt. Beispielsweise könnte ein Verfahren, das eine Rechenzeit von 45 Zeiteinheiten besitzt, den Schedule in Abbildung 2.6 generieren.

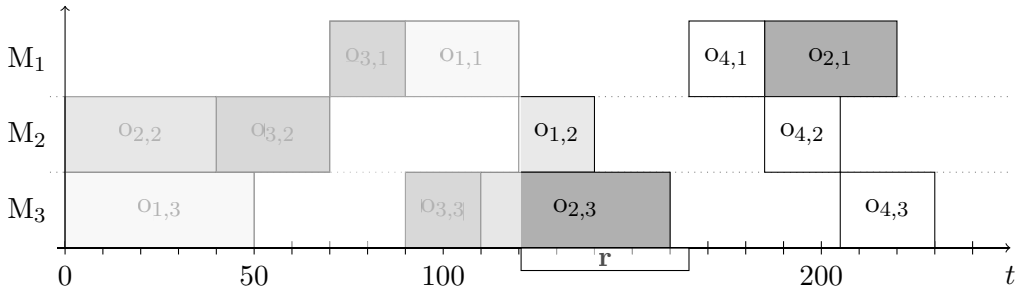


Abbildung 2.6.: Ineffizienter Schedule nach Umplanung

Dieser Schedule hat einen um 35 Zeiteinheiten längeren Makespan als der vom ersten Verfahren generierte ($C^1_{max} = 195$ zu $C^2_{max} = 230$), was durch die längere Zeit zur

Berechnung der Lösung verursacht wird. Er ist jedoch schneller (effizienter) als die einfache Ausführung des alten (initialen) Schedules und die anschließende Abarbeitung des neu eingetroffenen Auftrags, was zu einem Makespan von 260 Zeiteinheiten führt. So würde selbst durch das schlechtere Ablaufplanungs- und Steuerungsverfahren ein um 30 Zeiteinheiten besserer Makespan im Vergleich zur Lösung ohne Rescheduling erreicht. Das schnellere Verfahren erreicht hier allerdings eine Verbesserung um 65 Zeiteinheiten. □

Es wird deutlich, dass das zu entwickelnde Verfahren zur Ablaufplanung und -steuerung wenig Rechenzeit verbrauchen darf. Andernfalls würde der Fertigungsprozess unnötig verlangsamt, was zu zusätzlichen Kosten führt.

2.2.2.4. Maximierung der Lösungsqualität

Neben der Rechenzeit stellt die Qualität der Lösung einen weiteren Faktor des zu entwickelnden Verfahrens zur Ablaufplanung und -steuerung dar. Die Qualität[56] einer Lösung ist deren Abweichung vom optimalen Zielfunktionswert.[57] *Piramuthu et al.* schreiben: „The need for generating efficient manufacturing schedules has further increased in the face of increasing global competition that requires a reduction in manufacturing cycle times, as well as greater adaptability and flexibility.“[58]

Weil im Kontext der betrachteten Job-Shops und Flexible-Flow-Shops eine optimale Lösung nur schwer – in sinnvoller Rechenzeit – ermittelt werden kann, ist das Ziel, unter den gegeben Umständen eine möglichst hohe Lösungsqualität zu erreichen.[59]

[56] Qualität ist mit einer Wertung – wie ausgezeichnete oder schlechte Qualität – verbunden, welche die Zweckmäßigkeit zum Ausdruck bringt. Die Qualität ist vom jeweiligen Bezugssystem – hier determiniert über die Zielfunktion – abhängig (vgl. Klaus und Buhr, 1987, Seite 996 ff.).

[57] Vgl. (Nissen, 1997, Seite 19).

[58] (Piramuthu et al., 1994)

[59] Vgl. (Georgi, 1995, Seite 91). Insgesamt gilt es, bei der Entwicklung eines Lösungsverfahrens folgendes zu beachten: Komplizierten Verfahren ist eine größere Fehleranfälligkeit inhärent. Gleichfalls verringert der höhere Erklärungsbedarf die Akzeptanz. Einfache Verfahren sind demnach leichter in der betrieblichen Praxis umzusetzen. Werden durch das kompliziertere Verfahren theoretisch bessere Lösungen erzielt, ist abzuwägen, ob dieser Mehrnutzen bei der praktischen Umsetzung tatsächlich erreichbar ist und der Mehraufwand für die Einführung und den Betrieb gerechtfertigt sind (vgl. Lödding, 2008, Seite 80). Daher ist bei gleicher oder besserer Lösungsqualität das einfachere Verfahren vorzuziehen. *Blumer et al.* nennen als eine der verschiedenen Begründungen für dieses Vorgehen (Sparsamkeitsprinzip): Es existieren viele komplexe Modelle, die einen Sachverhalt erklären, jedoch nur wenige einfache Modelle, die ebenfalls den Sachverhalt erklären. Wenn durch ein einfaches Modell ein Sachverhalt erklärt werden kann, dann ist die Wahrscheinlichkeit, dass dieses zukünftige Ereignisse ebenfalls erklärt, höher als bei einem komplexen Modell (vgl. Blumer et al., 1987). Zusätzlich sind Modelle mit wenigen Annahmen einfacher zu falsifizieren als komplexere.

2.2.3. Anforderungen an eine Lernkomponente

2.2.3.1. Approximation von Trainingsverfahren

Das Lernverfahren muss in der Lage sein zu erlernen, wie der Trainer sich verhält. Es muss die Entscheidungsmethodik des Trainers mit einer hohen Genauigkeit approximieren. Da ein Lernverfahren als Auswahlmethode in Steuerungssituationen verwendet werden soll, muss folgende Annahme gelten: Es besteht ein Zusammenhang zwischen der Genauigkeit der Auswahlmethode und dem erreichten Planungs- und Steuerungsergebnis, gemessen an der Zielfunktion. Trifft die Lernkomponente gute Entscheidungen, ist die Lösungsgüte des entwickelten Verfahrens hoch. Im Folgenden wird also davon ausgegangen, dass die Abweichung des Ablaufplanungs- und Steuerungsverfahrens zur optimalen Lösung mit steigender Auswahlgenauigkeit der Lernkomponente abnimmt.

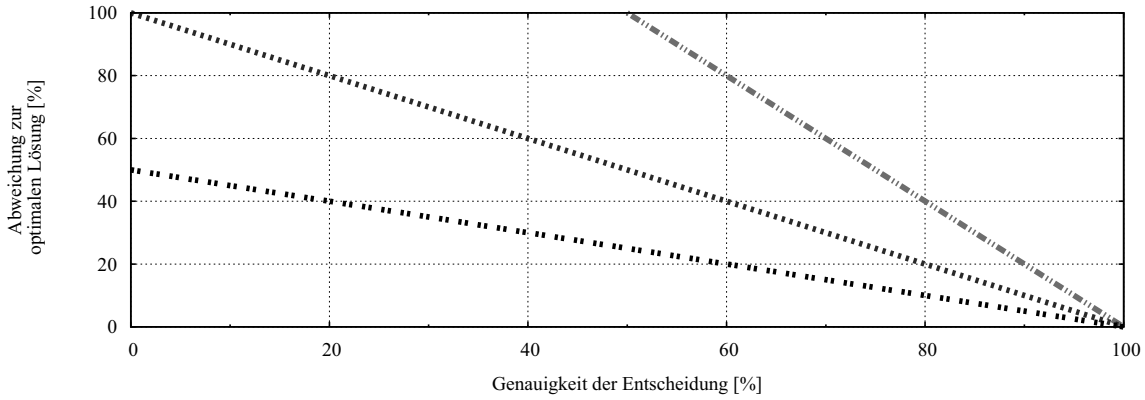


Abbildung 2.7.: Hypothese zur Entscheidungsgenauigkeit und Lösungsgüte

Abbildung 2.7 illustriert die Hypothese des Zusammenhangs zwischen Genauigkeit der Entscheidung des Lernverfahrens und dem Planungsergebnis. Es sind beispielhaft drei Geraden für die Beziehung dargestellt, jedoch ist nicht zwingend von einem linearen Zusammenhang auszugehen. Ebenso wird das Ausmaß einer Verbesserung der Güte der Entscheidungen mit der damit einhergehenden Verbesserung des Planungsergebnisses von den, dem Verfahren zur Verfügung stehenden Attributen (Merkmalen) und Klassen (Steuerungsregeln), beeinflusst. Grundsätzlich kann – unabhängig von der Genauigkeit – die optimale Lösung nur dann erreicht werden, wenn mit den vorhandenen Steuerungsregeln auch optimal zu entscheiden ist. Im Falle einer null-prozentigen Abweichung des Ablaufplanungs- und Steuerungsverfahrens von der optimalen Lösung, mit der trainiert wurde, approximiert das Ablaufplanungs- und Steuerungsverfahrens das Trainingsverfahren zu 100 Prozent.

Demzufolge sind die Approximationsfähigkeit von anderen Verfahren, insbesondere des zum Training eingesetzten, sowie eine hohe Genauigkeit bei den Entscheidungen von der Lernkomponente zu erfüllende Anforderungen.

2.2.3.2. Adaptierbarkeit

Die Charakteristik des Produktionssystems und der Aufträge kann sich im Laufe des Einsatzes verändern. Wird beispielsweise ein neuer Typ von Aufträgen eingeführt oder eine Maschine ersetzt, mit dem die Lernkomponente bisher nicht trainiert wurde, so sind keine guten Entscheidungen für die neuen Situationen möglich. Ebenso können beim Einsatz des Ablaufplanungs- und Steuerungsverfahren im laufenden Produktionsprozess weitere Informationen, z. B. Trainingsbeispiele über das System, gewonnen werden. Damit diese zusätzlichen Daten nutzbar werden, muss das Lernverfahren – die Beispielmenge der Wissensbasis, auf deren Grundlage durch die Lernkomponente Entscheidungen getroffen werden – aktualisierbar sein. Die Aktualisierung der Wissensbasis muss zusätzlich ohne großen Aufwand und automatisiert durchzuführen sein, um die neuen Informationen schnell in den Entscheidungsprozess einzubinden.

Damit sich das Verhalten des Lösungsverfahrens dem Fertigungssystem im laufenden Fertigungsprozess anpassen kann, ist die Anforderung nach einer adaptierbaren Lernkomponente zu erfüllen.

2.2.3.3. Ausgabe der Entscheidungssicherheit

Im Allgemeinen ist davon auszugehen, dass das Ziel bezüglich Entscheidungen der Lernkomponente darin besteht, eine möglichst hohe Erfolgsrate, im Sinne einer hohen Anzahl korrekter Entscheidungen für die Vorhersagen zu erzielen. Daraus folgt, dass das Ergebnis für eine zuzuordnende Situation immer dann korrekt ist, wenn die Vorhersage mit dem tatsächlichen Wert identisch ist. Inkorrekt ist ein Ergebnis, wenn Vorhersage und tatsächlicher Wert nicht übereinstimmen. Die Zuordnung ist eine Null- oder Eins-Entscheidung und es sind keine Randbereiche vorhanden. Immer dann, wenn vom Lernverfahren bei dessen Einsatz entweder eine korrekte oder eine inkorrekte Vorhersage zu treffen ist, ist der Erfolg das zu wählende Maß.[60] Jedoch ist in den hier betrachteten Organisationsformen in vielen Fällen nicht eindeutig zwischen diesen Ausprägungen

[60] Dieses Erfolgsmaß ist auch als 0-1-Verlustfunktion bekannt. Der Verlust ist Null bei korrekter Vorhersage und Eins bei inkorrekter. Die Anwendung des Verlustbegriffs ist also konventionell, doch könnte eine optimistische Terminologie das Ergebnis ebenso im Hinblick auf den Erfolg ausdrücken (vgl. Witten und Frank, 2005, Seite 142).

differenzierbar und die Zuordnung einer Wahrscheinlichkeit zur Vorhersage wünschenswert. Ebenso sind in Fertigungssystemen selten alle Informationen vorhanden, so dass keine 100-prozentig korrekten Aussagen erzielbar sind. Dieses Mapping auf eine Null-Eins-Entscheidung würde hier zu Potentialverlusten des Verfahrens führen. Ein mit einer Wahrscheinlichkeit von 90 Prozent korrekt vorhergesagtes Ergebnis kann mehr Gewicht haben als ein mit 34 Prozent vorhergesagtes. Bei beispielsweise drei Alternativen ist die korrekte Vorhersage mit einer Wahrscheinlichkeit von 34 Prozent nur geringfügig besser als eine inkorrekte Vorhersage mit einer Wahrscheinlichkeit von 30 Prozent.[61] Diese Unschärfe ist ausnutzbar. Es können mehrere alternative Lösungen erstellt und miteinander verglichen werden. Würde es nur zu Null-Eins-Entscheidungen kommen, wäre dieses nicht möglich, die Lösungen wären alle gleich. Erst durch die alternativen Lösungen mit Wahrscheinlichkeiten wird eine qualitative und gezielte Berechnung von Alternativen möglich.

Aus diesem Grund muss die Lernkomponente in der Lage sein, die Wahrscheinlichkeiten bei der Zuordnung in Steuerungsentscheidungen mit auszugeben.

2.2.3.4. Minimierung der Lern- und Vorhersagezeit

Sollen maschinelle Lernverfahren in Fertigungssystemen eingesetzt werden, ist es aufgrund der Datenmengen erforderlich, dass die Lernalgorithmen auf den Datenmengen effizient arbeiten. Um eine komplexe Situation beschreiben zu können, muss das Lernverfahren mit einer großen Anzahl an beschreibenden Merkmalen arbeiten können, es muss skalieren. Dabei sind die zwei voneinander unabhängigen Größen Speicherplatz und Rechenzeit kritisch zu betrachten. Die relevante der beiden Größen in Fertigungssystemen ist die Rechenzeit.[62] Diese wird zum einen durch die Lernzeit, die erforderliche Zeit zur Verarbeitung der gegeben Trainingsbeispiele für das Training, und zum anderen durch die Vorhersagezeit, die benötigte Zeit zum Treffen einer Entscheidung, determiniert.

Verhält sich die *Lernzeit* nicht linear oder zumindest annähernd linear zur Anzahl der Trainingsdaten, ist die Verarbeitung von großen Datenmengen irgendwann nicht mehr möglich. Ähnlich verhält es sich bei einer kleineren Datenmenge. Ist das Lernverfahren langsam, so ist die Anwendung in der zeitkritischen Ablaufplanung und -steuerung von Fertigungssystemen bereits hier nicht möglich. Dieses gilt besonders bei der Adaptierbarkeit des Verfahrens zur Laufzeit. Demzufolge sind bei Anwendungen, in denen die

[61] Vgl. (Witten und Frank, 2001, Seite 142).
[62] Speicherplatz wird als in ausreichender Menge gegeben vorausgesetzt.

Anzahl der Attribute ein kritischer Faktor ist, nur Lernverfahren akzeptabel, die sich linear zur Zahl der Attribute verhalten.[63]

Neben der erforderlichen Zeit für das Training des Lernverfahrens ist insbesondere die *Vorhersagezeit* bei der Anwendung zur Ablaufplanung und -steuerung in den betrachteten Job-Shops und Flexible-Flow-Shops ein kritischer Faktor.[64] Da im Produktionsprozess unter strengen Zeitbeschränkungen zu agieren ist,[65] ist es wichtig, dass das Lernverfahren das Wissen schnell auswerten kann, um benötigte Entscheidungen bereitzustellen. Die Vorhersagezeit ist im Produktionsprozess also kritischer zu bewerten als die Trainingszeit. Verzögerungen der Produktion durch zu späte Entscheidungen des Lernverfahrens führen zu Kosten.

Insgesamt ist also zu berücksichtigen, dass die Lernkomponente mit großen Datenmengen umgehen können muss und dabei eine minimale Lern- und Vorhersagezeit gewährleistet.

2.2.3.5. Bewältigung von kardinalen Merkmalen

Die Werte von situationsbeschreibenden Merkmalen für ein maschinelles Lernverfahren können aus verschiedenen Formaten bestehen.[66] In Fertigungssystemen sind die Werte eines Merkmals in aller Regel numerischer Natur, d. h. kardinal skaliert. Der Vorteil von kardinalen Merkmalen ist, dass mit deren Hilfe Situationen präziser beschreibbar und vergleichbar sind als beispielsweise bei der Verwendung von Werten mit nominaler Skalierung. Gleichzeitig stellen kardinale Merkmale eine Herausforderung an Lernverfahren dar, weil theoretisch unzählige Ausprägungen eines kardinal skalierten Merkmals existieren.

Die sich hieraus ergebende Anforderung ist, dass das eingesetzte Lernverfahren in der Lage sein muss, kardinal skalierte Merkmale angemessen zu verarbeiten.

[63] Vgl. (Witten und Frank, 2001, Seite 352).

[64] Einfache Lernverfahren wie nach der Methode der nächsten Nachbarschaft müssen für die Ableitung einer einzigen Vorhersage beispielsweise die gesamte Datenbank durchsuchen, was zu sehr langen Vorhersagezeiten führt (vgl. Witten und Frank, 2001, Seite 352).

[65] Siehe Abschnitt 2.2.2.3.

[66] Vgl. u. a. (Tan et al., 2006, Seite 26), der die Maßebenen von Merkmalen als Nominal, Ordinal, Intervall und Ratio ordnet. Dabei sind die Maßebenen Nominal und Ordinal kategorischer Natur, Intervall und Ratio numerischer.

2.3. Ableitung der Problemstellung

Ziel ist die Entwicklung eines Verfahrens zur Ablaufplanung und -steuerung für den Einsatz im laufenden Fertigungsprozess in zeitrestriktiven Fertigungssystemen – hier Job-Shops und Flexible-Flow-Shops. Es ist ein (proaktiv-) reaktives Verfahren unter Berücksichtigung maschineller Lernverfahren zu konzipieren. Dabei ist sicherzustellen, dass dieses vor dem operativen Einsatz trainiert wird, um zu zweckmäßigen Entscheidungen hinsichtlich der Zuordnung von Jobs zu Maschinen in Job-Shops und Flexible-Flow-Shops zu gelangen. Gleichzeitig soll das Verfahren in der Lage sein, Echtzeit nahe Entscheidungen zu ermöglichen, um dem dynamischen Charakter des Fertigungssystems gerecht zu werden. Aufgrund des stufenweisen (dezentralen) Einsatzes in Flexible-Flow-Shops sowie der vom Fertigungssystem geforderten Rechenzeit minimalen Steuerungsentscheidungen kann es aus zentralem Blickwinkel betrachtet zu ungünstigen Entscheidungen kommen. Weil das Verfahren für Rechenzeit minimale Steuerungsentscheidungen situativ trainiert wird, kann es zu schlechteren Ergebnissen kommen als bei offline Planungsverfahren mit ausreichender Berechnungszeit und sämtlichen Situationsinformationen. Diese potentiellen Einbußen von Lösungsqualität sind den Anforderungen des Fertigungssystems an das Ablaufplanungs- und Steuerungsverfahren geschuldet. Über eine Simulation des Fertigungsprozesses sind Methoden zur Erzeugung von Trainingsdaten zu entwickeln. Das Verfahren ist zu adaptieren.

3. Stand der Technik

Kapitel 3 beleuchtet den für den Untersuchungskontext dieser Arbeit relevanten Stand der Technik, um vorhandene Methoden zur Problemlösung zu identifizieren. Hierzu werden existierende Verfahren und Konzepte dargestellt und auf Erfüllung der Anforderungen an ein Lösungsverfahen hin untersucht. Dazu werden die in Abschnitt 2.2 skizzierten Anforderungen herangezogen.

3.1. Analyse von Verfahren der Ablaufplanung und -steuerung

Verfahren der Ablaufplanung und -steuerung lassen sich in optimierende und heuristische Verfahren unterteilen. *Optimierende Verfahren* finden mit ihrer Ausführung eine bzw. alle optimalen Lösungen. *Heuristische Verfahren* werden zur Lösung von praktischen Problemen eingesetzt, für die in der Regel eine exakte Lösung nur schwer oder zeitaufwändig zu bestimmen ist. Dabei haben Heuristiken zwar eine geringe Laufzeit, garantieren jedoch keine optimale Lösung.[1] Eine mögliche Aufteilung heuristischer Verfahren im Bereich des Shop-Schedulings bieten *Jain und Meeran*.[2] Die Autoren unterscheiden konstruierende und iterative Heuristiken. Als *konstruierend* lassen sich auf Prioritätsregeln basierende Verfahren bezeichnen, weil sie Lösungen aufbauen, ohne bereits erstellte Teillösungen zu ändern. Die Teillösungen werden zu einer Gesamtlösung zusammengesetzt. *Iterative* Algorithmen (Suchverfahren) nähern sich einer Lösung an bzw. verbessern schrittweise eine mögliche Lösung oder Teillösung. Dazu gehören hauptsächlich lokale Suchverfahren sowie solche mit künstlicher Intelligenz.

[1] Vgl. (Zäpfel und Braune, 2005, Seite 22 f.).
[2] Vgl. (Jain und Meeran, 1999) und Anhang A.1.

3.1.1. Optimierende Verfahren

Mit optimierenden Verfahren sind, bis auf wenige Spezialfälle, hohe Zeitanforderungen an die Lösungserstellung verbunden. Es gibt sogenannte effiziente Methoden, die eine kleine Klasse von Shop-Scheduling-Problemen mit polynomial wachsender Laufzeit in der Eingabegröße optimal lösen. Dazu gehören Job-Shop-Scheduling-Probleme mit N Aufträgen und M Maschinen, die darauf beschränkt sind, dass sie maximal aus zwei Aufträgen ($2 \times M$-Problem) oder aus höchstens zwei Operationen pro Auftrag ($N_j \times 2$-Problem) bestehen. Ebenso können $N_j \times 2$-Probleme mit beliebig vielen Operationen für jeden Auftrag j effizient gelöst werden, wenn die Bearbeitungszeiten einheitlich sind. Solange $\mathcal{P} \neq \mathcal{NP}$ gilt, existiert zur Lösung von Scheduling-Problemen realistischer Größe kein effizientes Verfahren. Vollständig enumerierende Verfahren können aufgrund der Größe des Lösungsraumes nur begrenzt eingesetzt werden. Da es bis zu $(N!)^M$ Lösungen geben kann, ist ein Aufzählen aller Lösungen schon bei einer kleinen Maschinen- und Auftragsanzahl M und N nicht möglich.[3] Optimierende Verfahren stoßen also sehr schnell an Grenzen.

Branch-and-Bound-Verfahren

Zu den schnellsten optimierenden Verfahren der Ablaufplanung und -steuerung in Job-Shops und in Flexible-Flow-Shops gehören die auf der Grundlage der Graphensuche aufbauenden Branch-and-Bound-Verfahren.

Ein Graph $G = (V, E)$ sei ein Tupel, welches aus einer nicht-leeren Knotenmenge $V = \{v_1, \ldots v_n\}$ sowie einer Kantenmenge E (E ist eine Menge von Paaren aus V) besteht. Ein Weg der Länge l zwischen zwei Knoten $u, v \in V$ sei eine Folge von Kanten (u, v_1), (v_1, v_2), $\ldots$, (v_{l-1}, v). Ein Kreis sei ein Weg mit $l \geq 2$, wobei alle Kanten verschieden sind. Ein Graph sei zusammenhängend, wenn es zwischen zwei beliebigen Knoten u, v einen Weg gibt. Ein Baum sei ein zusammenhängender, zyklenfreier Graph $G = (V, E)$ mit $|V| - 1$ Kanten.

Branch-and-Bound-Verfahren können für eine größere Klasse von Problemen optimale Lösungen finden. Dazu durchsuchen sie den Suchraum mit potentiellen Lösungen möglichst geschickt. Eine Baumstruktur, die dynamisch erzeugt wird, repräsentiert die möglichen Lösungen. Jeder innere Knoten steht für einen partiellen Schedule und jedes Blatt für einen vollständigen. Ein partieller Schedule unterscheidet sich von einem vollständigen dadurch, dass noch nicht alle Operationen eines Problems eingeplant sind. Aus einem partiellen Schedule können vollständige Schedules aufgebaut werden. Ein

[3] Siehe Abschnitt 2.1.3.

vollständiger Schedule enthält alle partiellen Schedules seiner Vorgänger in der Baumstruktur. Branch-and-Bound-Verfahren führen eine Tiefensuche durch, die mit der Wurzel in Tiefe Null des Baumes beginnt und mit einem Blatt endet. Zusätzlich gibt es eine obere Schranke (UB) und eine geschätzte untere (LB), die den noch zu untersuchenden Bereich einschränken. Die Qualität der besten bisher gefundenen Lösung bestimmt die obere Schranke. Wird diese Schranke überschritten, bedeutet dies, dass bereits eine bessere Lösung bekannt ist. Die Lösungsqualität eines partiellen Schedules wird geschätzt, wenn der vollständige Schedule für eine exakte Bestimmung noch nicht aufgebaut ist. Sie bildet die untere Schranke. Die Schätzung darf dabei die Lösungsqualität nicht unterschätzen und bessere Werte annehmen als die vollständigen Schedules, die den partiellen Schedule enthalten, da ansonsten die Terminierung des Branch-and-Bound-Verfahrens nicht garantiert werden kann. Sobald $LB(v)$ für den durch den Knoten v repräsentierten partiellen Schedule S_v größer als UB ist, können alle Schedules, die auf S_v aufbauen, nicht mehr besser werden als UB. Dies hat zur Folge, dass alle folgenden Schedules, die den partiellen Schedule enthalten, nicht weiter untersucht werden müssen. Dadurch ist der Suchraum schneller explorierbar. Das Branch-and-Bound-Prinzip wird beispielsweise von *Brucker* sowie *Blazewicz et al.* beschrieben.[4]

Verfahren nach dem Branch-and-Bound-Prinzip führen eine systematische Suche aus, das heißt, die Suche ist beendet, wenn alle Knoten genau einmal untersucht sind. Dies kann explizit oder implizit erfolgen, indem ein Teil des Suchraums ausgeschlossen wird, dessen Lösungsqualitäten schlechter sind als die aktuelle obere Schranke. Im ungünstigsten Fall ist der Aufwand eines Branch-and-Bound-Verfahrens für Job-Shop-Scheduling-Probleme $\mathcal{O}((N!)^M)$ mit N Aufträgen und M Maschinen. In der sich aus dem Aufwand benötigten Berechnungszeit liegt die größte Schwäche von Branch-and-Bound-Verfahren. Realitätsnahe Probleme sind von diesen auch nach Jahren der Forschung und immer schnelleren Computern nicht in akzeptabler Zeit lösbar.[5] Gleiche Aussagen treffen auch auf Branch-and-Bound-Verfahren für Flexible-Flow-Shops zu.[6]

Analyse optimierender Verfahren

Hackstein beschreibt die Problematik des Reihenfolgeproblems in Ablaufplänen mit exakten Verfahren zur Optimierung, die mit vertretbarer Rechenzeit auskommen, als praktisch nicht lösbar.[7] *Aytug et al.* fassen die Unzulänglichkeiten optimierender Verfahren wie folgt zusammen: „The implication is obvious-realistically sized problems cannot

[4] Vgl. (Brucker, 2007, Seite 56 ff. und 202 ff.) und (Blazewicz et al., 2007, Seite 33 ff.).
[5] Siehe Abschnitt 2.1.3.
[6] Vgl. u. a. (Lo et al., 2008; Brockmann und Dangelmaier, 1998).
[7] Vgl. (Hackstein, 1989, Seite 18).

be solved optimally [...]. Furthermore, the static nature of most OR models ignores the dynamic nature of most real-time scheduling environments [...]. The OR scheduling problem is a classic. Yet, although textbooks promote these analytical models, few real-life applications are known."[8] Diese Meinung teilen *Hopp und Spearman*: „Unfortunately, most real-world problems violate the assumptions made in the classic scheduling theory literature [...]. We cannot hope to find optimal solutions of many realistic-size scheduling problems."[9] *Wang* ergänzt: „The branch and bound solution method is an exact method, which provides optimal solutions. One major limitation is that the exact method is computationally expensive, and has been proved impractical for even modestly sized problems."[10]

	Reduktion der Komplexität	Bewältigung von Unsicherheiten	Minimierung der Rechenzeit	Maximierung der Lösungsqualität
Optimierende Verfahren	⊘	⊘	⊖⊖	⊕⊕

Tabelle 3.1.: Bewertung: Optimierende Verfahren[11]

Die Bewertung von optimierenden Verfahren zur Ablaufplanung und -steuerung anhand der in Abschnitt 2.2.2 dargelegten Anforderungen für eine Steuerungskomponente zeigt Tabelle 3.1. Optimierende Verfahren können also nicht zur Ablaufplanung und -steuerung im Kontext der hier betrachteten Organisationsformen eingesetzt werden. Aufgrund der mit optimierenden Verfahren erreichbaren Lösungsqualität können diese jedoch trotz ihrer langen Rechenzeit vor dem Beginn des Fertigungsprozesses zum Trainieren verwendet werden.

3.1.2. Heuristische Verfahren

Nicht zuletzt aufgrund der langen Rechenzeiten der optimierenden Verfahren wurden und werden für den Einsatz in Fertigungssystemen Heuristiken entwickelt und verwendet.[12]

[8] (Aytug et al., 1994).
[9] (Hopp und Spearman, 2007, Seite 525).
[10] (Wang, 2005).
[11] Legende zur Erfüllung der Anforderungen bei der Bewertung: ⊖⊖ = sehr schlecht, ⊖ = schlecht, ⊘ = einigermaßen, ⊕ = gut und ⊕⊕ = sehr gut.
[12] Vgl. (Hackstein, 1989, Seite 18).

Prioritätsregeln

Die Erstellung von Schedules mit Hilfe von Prioritätsregeln (Dispatching-Rules) ist eine der ersten Methoden, die wegen ihrer Einfachheit noch heute angewendet werden.[13]

Eine Reihenfolgepolitik, die eine bestimmte Auftragsreihenfolge-Entscheidung bei jedem Freiwerden einer Maschine oder Ankunft eines neuen Auftrages festlegt, wird Prioritätsregel genannt. Eine solche determiniert den als nächsten aus dem Wartevorrat für die Wiederbelegung der Maschine auszuwählenden Arbeitsauftrag oder die Maschine aus den zur Bearbeitung des Auftrages bereitstehenden.[14]

„Machine scheduling in most production systems is done by allocating priorities to jobs waiting at various machines through dispatching heuristics.“[15] Als Hauptgrund kann angeführt werden, dass der Berechnungsaufwand sehr gering ist. Prioritätsregeln sind als einfache Steuerungsmethoden einsetzbar. Dabei haben Prioritätsregeln den entscheidenden Vorteil, lokal und in Echtzeit angewendet werden zu können.

Panwalkar und Iskander stellen etwa 113 verschiedene Prioritätsregeln vor, so dass an dieser Stelle von einer umfassenden Darlegung abgesehen werden muss.[16] *Dabbas und Fowler* ordnen Prioritätsregeln als lokal oder global. *Lokale Regeln* ziehen nur lokale Kriterien zur Ablaufplanung und -steuerung heran, wohingegen *globale Regeln* auch vergangene und / oder erwartete Situationen berücksichtigen. *Dabbas und Fowler* geben in ihrer Klassifikation, wie in Abbildung 3.1 dargestellt, Nuancen der Unterscheidungen an. Ebenfalls wird von ihnen eine Möglichkeit der Erweiterung des Dispatching-Ansatzes vorgestellt. Demzufolge sind verschiedene Regeln derart in einer Linearkombination verknüpfbar, dass das Dispatching gleichzeitig mehreren Zielfunktionen[17] gerecht wird.

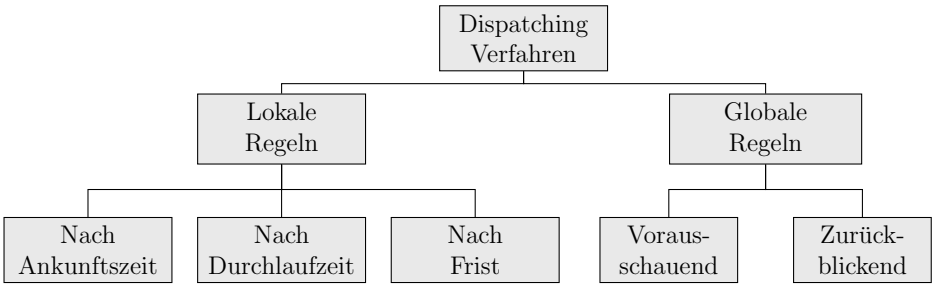


Abbildung 3.1.: Taxonomie der Echtzeit-Prioritätsregeln[18]

[13] Vgl. (Günther und Tempelmeier, 2005, Seite 224), (Nyhuis et al., 2009) und (Aytug et al., 1994).
[14] Vgl. (Haupt, 1996, Seite 1419).
[15] (Piramuthu et al., 2000)
[16] Vgl. (Panwalkar und Iskander, 1977).
[17] Siehe Abschnitt 2.1.2.
[18] Vgl. (Dabbas und Fowler, 2003).

Wie *Lee et al.* sowie *Dabbas und Fowler* zeigen, besteht der Vorteil von Prioritätsregeln trotz der immer leistungsstärker werdenden Computersysteme in vielen Einsatzgebieten[19] fort.[20] Insbesondere dann, wenn große Unsicherheiten vorherrschen und zugleich schnelle Entscheidungen gefordert werden, kommen Prioritätsregeln vielfach zum Einsatz. Jedoch existiert den Feststellungen von *Blackstone Jr. et al.* zufolge keine Regel, die in Bezug auf eine Zielsetzung stets bessere Ergebnisse liefert als alle anderen Regeln.[21] So stellen verschiedene Studien fest, dass es keine optimale Prioritätsregel gibt.[22] Die dynamische Verwendung von Regeln führt also zu besseren Ergebnissen als die Verwendung einer einzelnen Regel. Nach *Piramuthu et al.* hängt die Effizienz von Prioritätsregeln vom Status des Systems ab, der über die Systemattribute wiedergegeben wird.[23] Dazu stellen *Piramuthu et al.* heraus: „[...] it may be possible to improve system performance by implementing a scheduling policy rather than a single dispatching rule. Since the values of these attributes change continually in a dynamic system, it appears natural to use an approach that adaptively employs different scheduling heuristics at various points in time.“[24] Hierzu sind Mechanismen erforderlich, die den aktuellen Systemstatus identifizieren und die passende Prioritätsregel auswählen. *Jeong* schlägt zur Ermittlung der besten Prioritätsregel die Nutzung von Simulationstechniken vor.[25]

Bei dem Einsatz von Prioritätsregeln kann in vielen Fällen keine eindeutige Priorisierung eines Auftrages für ein Ziel vorgenommen werden. Regeln bieten hinsichtlich eines bestimmten Ziels gute Lösungen, unterstützen andere Ziele jedoch nicht gut.[26] Um Konflikte während der Zuteilung zu vermeiden und mehrere Ziele zu vereinbaren, ist eine Kombination von Prioritätsregeln vorzunehmen. Hierbei sind die Kombinationsmöglichkeiten additiv, multiplikativ und alternativ umsetzbar. Bei Anwendung der *additiven* Kombination werden die Prioritätskennzahlen addiert. *Multiplikative* Regelverknüpfungen setzen die Regeln ins Verhältnis. Diese erlauben eine Gewichtung der Regeln über den Exponenten. Mittels *alternativ* Verknüpfungen werden Bedingungen formuliert, über die immer nur eine Prioritätsregel ausgewählt wird.[27]

Daher können Prioritätsregeln dezentral eingesetzt werden und erlauben ebenso eine

[19] Voraussetzung für den Einsatz von Prioritätsregeln ist eine hinreichend große Flexibilität des Fertigungssystems. So dürfen beispielsweise keine allzu langen Vorlaufzeiten vor der Nutzung eines Arbeitssystems erforderlich sein (vgl. Aytug et al., 2005).
[20] Vgl. (Lee et al., 2009; Dabbas und Fowler, 2003).
[21] Vgl. (Blackstone Jr. et al., 1982). Vgl. auch (Priore et al., 2006).
[22] Vgl. (Jain und Meeran, 1998).
[23] Vgl. (Piramuthu et al., 2000).
[24] (Piramuthu et al., 2000)
[25] Vgl. (Jeong, 1998).
[26] Vgl. (Canbolat und Gundogar, 2004, Seite 528).
[27] Vgl. (Zäpfel und Braune, 2005, Seite 36).

schnelle situative Lösungsfindung.[28] Allerdings ist die Qualität, von mit Prioritätsregeln erstellten Lösungen in den seltensten Fällen gut im Sinne der verfolgten Zielfunktion. Aus diesem Grund sind Prioritätsregeln nicht direkt zur Ablaufplanung und -steuerung in dem in Kapitel 2 dargelegten Problem geeignet.

Verfahren nach Giffler/Thompson

Das Verfahren nach Giffler-Thompson ist Grundlage vieler auf Prioritätsregeln basierender Algorithmen, da sich mit dessen Hilfe aktive Schedules[29] erstellen lassen.[30] Mit der Giffler-Thompson-Heuristik wird aus der Menge der aktuell einplanbaren Operationen diejenige ausgewählt, für die sich der geringste Fertigstellungszeitpunkt berechnet. Andere Operationen, die auf derselben Maschine der ausgewählten Operation bearbeitet werden müssen und einen früheren Startzeitpunkt als den Fertigstellungszeitpunkt der Auswahl haben, bilden einen Konflikt. Eine Auswahlregel legt fest, welche Alternative (Operation) bei einem Konflikt bevorzugt und eingeplant wird. Eine formalere Beschreibung des Giffler-Thompson-Verfahrens in der Umsetzung von *Zäpfel und Braune*[31] gibt Algorithmus 3.1. Die zugehörige Legende zeigt Tabelle 3.2.

Algorithmus 3.1 : Giffler-Thompson-Heuristik nach (Zäpfel und Braune, 2005)

1 $SO := \{o|\text{erste Operation } o \text{ eines Auftrags}\}$;
2 $t(o) := 0, \forall o \in SO$;
3 **while** $SO \neq \emptyset$ **do**
    `// Bestimmen der Operation mit der kleinsten Fertigstellungszeit`
4 $d(o) := t(o) + p(o), \forall o \in SO$;
5 $d_{min} := min\{d(o)|o \in SO\}$;
    `// Bilden der Konfliktmenge`
6 $CS := \{o|o \in SO \wedge M(o) = M(o_{min}) \wedge t(o) < d_{min}\}$;
7 Wähle mit Prioritätsregel eine Operation $o' \in CS$ und plane diese ein;
    `// Einplanen der ausgewählten Operation`
8 $t(o) := d(o'), \forall o \in CS \setminus \{o'\}$;
9 $SO := SO \cup N(o') \setminus \{o'\}$;
10 $t(o) := d(o'), \forall o \in N(o')$;
11 **end**

Die vorgestellte Heuristik generiert bei einigen Scheduling-Problemen jedoch ungültige Schedules, so dass es zu Doppelbelegungen von Maschinen kommen kann. Dieser Sachverhalt wird in Beispiel A.2 anschaulich dargestellt. Ebenso kann das Verfahren von

[28] Vgl. (Kern et al., 1996, Seite 23 f.).
[29] Siehe Abschnitt 2.1.2.
[30] Vgl. (Giffler und Thompson, 1960).
[31] Vgl. (Zäpfel und Braune, 2005, Seite 32).

SO	Menge der aktuell einplanbaren Operationen
CS	Konfliktmenge gleichzeitig auf einer Maschine einplanbarer Operationen
$N(o)$	Menge der Operationen o technologisch direkt nachfolgenenden Operation
$M(o)$	Maschine die Operation o bearbeitet
$t(o)$	Frühestmöglicher Startzeitpunkt einer Operation o
$p(o)$	Bearbeitungszeit einer Operation o
$d(o)$	Fertigstellungzeit einer Operation o
d_{min}	Minimum der Fertigstellungzeitpunkte
o_{min}	Operation o mit minimaler Fertigstellungszeit

Tabelle 3.2.: Legende zur Giffler-Thompson-Heuristik nach *Zäpfel und Braune*

Giffler und Thompson in dieser Form nur zur Ablaufplanung und -steuerung in Fertigungsumgebungen wie Job-Shops mit einmal vorhandenen Maschinentypen eingesetzt werden. Stehen mehrere gleichartige Maschinen, wie beispielsweise in einem Flexible-Flow-Shop, zur Bearbeitung von Aufträgen parallel zur Verfügung, ist das Verfahren nicht einsetzbar. Einen rudimentären Lösungsvorschlag liefern *Giffler und Thompson* selbst. Von ihnen wird vorgeschlagen, in Produktionssystemen, in denen parallele Maschinen mit gleichen Fähigkeiten vorhanden sind, die Aufträge beliebig zuzuteilen. „Suppose that there are k machines available [...]. Then [...] we can permit conflict sets to be as large as k without harm.“[32] Dies ist jedoch nur in Fällen mit identischen, nicht aber in dem hier betrachteten Fall von uniformen und unverwandten Maschinen in Flexible-Flow-Shops auf den einzelnen Stufen zutreffend.

Eine Adaption des Verfahrens von *Giffler und Thompson* für dynamische Job-Shops wird von *Chang und Sullivan* vorgestellt und von *Nascimento*, der Güteanalysen und Vergleiche durchführt, detaillierter untersucht.[33] Das Verfahren von *Chang und Sullivan* ist in Algorithmus 3.2 aufgeführt, die Legende in Tabelle 3.3.

Das Verfahren von *Giffler und Thompson* wäre grundsätzlich zur Ablaufplanung und -steuerung in den hier betrachten Organisationsformen geeignet. Die Anforderungen an die Verfahrenspositionierung, nach dem Planungs- und Steuerungszeitpunkt sowie nach geringer Rechenzeit werden erfüllt. Jedoch sind Korrekturen und Erweiterungen für den Einsatz in den untersuchten Job-Shops und Flexible-Flow-Shops vorzunehmen. Ebenso ist das Verfahren dahingehend zu erweitern, dass die Anforderung nach hohen Lösungsqualitäten erfüllt wird.

Verfahren der Künstlichen Intelligenz

Im Vergleich zu konventionellen Heuristiken gehören Verfahren der Künstlichen Intel-

[32](Giffler und Thompson, 1960)
[33]Vgl. (Chang und Sullivan, 1990; Nascimento, 1993).
[34]Vgl. (Chang und Sullivan, 1990).

Algorithmus 3.2 : Giffler-Thompson-Heuristik nach (Chang und Sullivan, 1990)

```
1  t = 0;
2  PS_0 = ∅;
3  Q_0 = {(j,i)| = 1,2,...,n};
4  W = {(PS_0, Q_0)};
5  while t < TO do
6      W' = ∅  while W ≠ ∅ do
7          Remove one (PS_t, Q_t) from W;
8          Find f* = min{f_ijk} for all i where (i,j) ∈ Q_t;
9          foreach (j,k) ∈ Q_t do
10             Find s*_ijk for any i such that P_ijk ≠ 0 and if s*_ijk < f* then
11                 PS_{t+1} = PS_t + {(i,j,k,s*,P)};
12                 Q_{t+1} = Q_t − {(j,k)} + {direct successor of (j,k)};
13                 Put (PS_{t+1}, Q_{t+1}) into W';
14             end
15         end
16     end
17     W = W';
18     t = t + 1;
19 end
```

ligenz, insbesondere das maschinelle Lernen, zu den aktuelleren Lösungsansätzen von Scheduling-Problemen. Derartige Verfahren unterscheiden sich durch ihre wissensbasierte Arbeitsweise von den Methoden, die in den vorangegangenen Abschnitten dargestellten wurden.

Chen und Yih sowie *Piramuthu et al.* zeigen, dass die Nutzung des maschinellen Lernens zur Auswahl der anzuwendenden Prioritätsregel bei der Ablaufplanung und -steuerung häufig vorteilhaft ist.[35] *Halevi und Wang* stellen ein Knowledge-Based-Manufacturing-System vor. Innerhalb dieses Ansatzes ist es nicht mehr die Aufgabe der Planer, Entscheidungen zu treffen bzw. Entscheidungsregeln vorzugeben; stattdessen fertigen die Produktionsplaner eine wissensbasierte Road-Map an. Jeder spätere Nutzer generiert durch die Anwendung des Systems einen Satz von Routinen, der den individuellen Bedürfnissen gerecht wird.[36] *Mahl und Krikler* beschreiben eine Softwareanwendung zur Erfassung und Wiederverwendung von regelbasiertem Wissen für die Ablaufsteuerung in Produktionssystemen.[37] *O'Kane* stellt eine wissensbasierte Anwendung vor, die genutzt wird, um reaktive Planungsszenarien in einer bestimmten Konfiguration eines Produktionssystems zu untersuchen. Konzepte des „History Logging" und des „Experten-System-learning" werden von *O'Kane* für die Bereitstellung von Entschei-

[35] Vgl. (Chen und Yih, 1996; Piramuthu et al., 1993).
[36] Vgl. (Halevi und Wang, 2007).
[37] Vgl. (Mahl und Krikler, 2007).

i	Subscript of workstations, $i = 1, 2, 3, ..., m$
j	Subscript of jobs, $j = 1, 2, 3, ..., n$
k	Subscript of operations, $k = 1, 2, 3, ..., w$
TO	Total number of operations to be scheduled
P_{ijk}	Process time of operation k of job j on workstation i
$d_{i_1 i_2}$	Transfer time form workstation i_1 to workstation i_2
s_{ijk}	Start time of operation k of job j on workstation i, $P_{ijk} \neq 0$
f_{ijk}	Finish time of operation k of job j on workstation i, i. e. $f_{ijk} = s_{ijk} + P_{ijk}$
s^*_{ijk}	Earliest possible start time of s_{ijk}
f^*_{ijk}	Earliest possible finish time of f_{ijk}
PS_i	A partial schedule containing t scheduled operations from TO operations, i. e. $PS_t = \{u \mid u = (i, j, k, s, P)\}$, where u is one of the scheduled operations represented by (i, j, k, s, P); s is the abbreviated notation of s_{ijk}, and P for P_{ijk}
Q_t	The set of schedulable operations corresponding to PS_t, i. e. $Q_t = \{h \mid h = (j, k)\}$, where h is one of the schedulable operations represented by (j, k), and where all the predecessors of operation k of job j have been scheduled
W	A set that contains all partial schedules and corresponding schedulable operation, i. e. $W = \{g \mid g = (PS_t, Q_t)\}$

Tabelle 3.3.: Legende zur Heuristik nach *Chang und Sullivan*[34]

dungen und die Kontrolle des Systems während der Planlebenszeit aufbereitet.[38] *Wang* beschreibt die Möglichkeiten der Anwendung von Data-Mining in Kombination mit Entscheidungsbäumen in der Produktionssteuerung.[39] *Canbolat und Gundogar* zeigen Fuzzy Prioritätsregeln für das Job-Shop-Scheduling.[40] *Kwak und Yih* beschreiben ein, auf einem Entscheidungsbaum basierendes Verfahren zur Steuerung einer flexiblen Kontroll- und Nachbearbeitungsmaschine. Das Competitive-Decision-Selector-Verfahren beinhaltet zwei Algorithmen, einerseits die Langzeituntersuchung der Regelanwendung, andererseits die kurzfristige Regelauswirkung mittels Data-Mining-Methoden auf Basis simulativ erstellter Trainingsdaten.[41] *Priore et al.* entwickeln ein Verfahren zur situationsabhängigen Auswahl von Prioritätsregeln in einem flexiblen Fertigungssystem. Das Verfahren setzt Methoden des maschinellen Lernens um. Sie analysieren die Performance des Systems in der Vergangenheit und gewinnen hiermit „Scheduling Wissen“, worüber sie die geeignete Prioritätsregel in jeder Situation mit Steuerungsnotwendigkeit auswählen wollen.[42] *Metan und Sabuncuoglu* entwickeln einen simulationsbasierten Lernmechanismus. Die Leistung des vorgestellten Systems wird überwacht und der Entscheidungsbaum zur Auswahl, wenn erforderlich, angepasst.[43] Weitere Verfahren aus dem Bereich der Künstlichen Intelligenz werden u. a. von *Piramuthu et al.; Park et al.; Shaw et al.* vorgestellt.[44]

[38] Vgl. (O'Kane, 2000).
[39] Vgl. (Wang, 2007).
[40] Vgl. (Canbolat und Gundogar, 2004).
[41] Vgl. (Kwak und Yih, 2004).
[42] Vgl. (Priore et al., 2006, 2001).
[43] Vgl. (Metan und Sabuncuoglu, 2005).
[44] Vgl. (Piramuthu et al., 2000, 1994, 1993; Park et al., 1997; Shaw et al., 1992).

Die dargestellten Verfahren zur Ablaufplanung und -steuerung mit Unterstützungsfunktionalitäten der Künstlichen Intelligenz bei Entscheidungen erfüllen die in Abschnitt 2.2 erarbeiteten Anforderungen an ein Lösungsverfahren für die betrachteten Organisationsformen nicht. Einige Anforderungen werden von einzelnen Verfahren zwar erfüllt, jedoch nicht in Gänze und nicht für beide hier untersuchten Organisationsformen. Dennoch wird durch die Verfahren deutlich, dass die Verwendung des maschinellen Lernens bei der Ablaufplanung und -steuerung ein guter Ansatz ist, es aber Erweiterungen und Verbesserungen bedarf.

Lokale Suchverfahren

Bei Lokalen Suchverfahren handelt es sich um kombinatorische Verfahren, die versuchen, durch Reihenfolgeveränderungen die optimale Bearbeitungsreihenfolge zu finden.[45] Zu den bekanntesten Verfahren gehören genetische Algorithmen,[46] Ant Colony Optimization,[47] Tabu-Suche[48]. Jede der genannten Methoden eignet sich für kombinatorische Optimierungsprobleme bei Scheduling-Problemen, da sie die Möglichkeit bieten, ein lokales Optimum zu verlassen. Die Wahrscheinlichkeit, ein besseres lokales oder das globale Optimum zu verpassen, sinkt damit. *Genetische Algorithmen* bilden Konzepte der Evolutionstheorie wie Selektion, Mutation und Rekombination nach. Mit der Anwendung dieser Konzepte verändern sie iterativ einzelne Lösungen aus einer Lösungsmenge, bis sie eine vorher definierte Abbruchbedingung erreichen. *Ant-Colony-Optimization-Verfahren* verwenden zur Lösung von Optimierungsproblemen Mechanismen der Schwarmintelligenz.[49] Eine *Tabu-Suche* durchsucht iterativ den Lösungsraum, indem sie eine Lösung modifiziert. Die leichte Modifikation einer Lösung bildet eine neue Lösung, die in der Nachbarschaft der ursprünglichen Lösung liegt. In einer Iteration werden gleich mehrere Lösungen in der Nachbarschaft untersucht und die beste ausgewählt.[50]

[45] Vgl. (Schneeweiß, 2002, Seite 275 f.) und (Domschke et al., 1997, Seite 46 ff.).

[46] Vgl. u. a. (Fang et al., 1993; Liu et al., 2006).

[47] Vgl. u. a. (Dorigo und Stützle, 2004; Dorigo und Blum, 2005) und aus dem verwandten Bereich der Bienenalgorithmen (Scholz-Reiter et al., 2007).

[48] Vgl. u. a. (Dell'Amico und Trubian, 1993).

[49] In einer künstlichen Ameisenkolonie, deren Ameisen aus Agenten bestehen, die Lösungen für ein Problem erzeugen, kann das Zusammenwirken der Ameisen zu einer besseren Lösung führen. Ameisen legen bei der Nahrungssuche eine Pheromonspur, die sich nach einer Weile auflöst; dabei wählen sie eher Wege mit einer stärkeren Pheromonspur als mit einer schwächeren. Kürzere und häufig benutzte Wege zur Nahrung entwickeln schnell eine starke Pheromonspur und werden bevorzugt.

[50] Es ist zulässig, dass die neue Lösung auch schlechter als die vorherige sein kann. Um Zyklen zu vermeiden, speichert die Tabu-Suche bereits bekannte Lösungen in einer Tabu-Liste. Lösungen, die in der Tabu-Liste stehen, dürfen nicht mehr ausgewählt werden. Die Länge der Liste ist für das Verhalten der Tabu-Suche ausschlaggebend, denn eine zu kurze Liste kann dazu führen, dass sich die Suche in einem Zyklus verliert. Andernfalls besteht die Gefahr, die Suche mit einer zu langen Liste zu stark einzuschränken.

Die Lösungsqualität von Lokalen Suchverfahren ist relativ gut, weil sie versuchen eine gegebene Lösung zu verbessern.[51] Jedoch stellt die Rechenzeit zur Verbesserung der Lösungen ein Problem dar, was Lokale Suchverfahren für den Einsatz in den hier betrachteten Job-Shops und Flexible-Flow-Shops ungeeignet erscheinen lässt. Ebenfalls sind einzelne situative Steuerungsentscheidungen nicht abbildbar. Lokale Suchverfahren eignen sich dennoch zur Generierung von Trainingsdaten vor dem Beginn des Fertigungsprozesses.

Shifting-Bottleneck-Verfahren

Shifting-Bottleneck-Verfahren stellen den Versuch eines Kompromisses der Lösungsqualität und der zur Generierung der Lösung erforderlichen Rechenzeit dar.[52] Da diese grundsätzlich immer gleich aufgebaut sind, wird nachfolgend exemplarisch das Job-Shop-Verfahren nach *Adams et al.* beschrieben.[53] Es ist auch für andere Problemklassen wie Flexible-Flow-Shops anpassbar.[54] Das Shifting-Bottleneck-Verfahren zerlegt die Job-Shop-Probleminstanz, um sie für die einzelnen Maschinen optimal zu lösen. Dabei gilt die Annahme, dass viele Abhängigkeiten zwischen der optimalen Lösung eines Ein-Maschinenproblems und der optimalen Gesamtlösung bestehen.[55] Die dem Verfahren zugrunde liegende Idee ist, in Iteration i_t alle in den vorhergehenden Iterationen $(i_0, \ldots, i_{t-1})$ festgelegten Auftragsreihenfolgen zu berücksichtigen.[56] Für das Shifting-Bottleneck-Verfahren sind folgende Teilaufgaben zu lösen: Auswahl einer Maschine $(m, \forall m \in M)$, Erstellung einer Ein-Maschinenprobleminstanz I und Berechnung einer optimalen Lösung für I. Der Shifting-Bottleneck-Algorithmus referenziert die beiden Mengen M' und M. Menge M' beinhaltet alle schon untersuchten Maschinen und ist initial leer. Menge M besteht aus den Maschinen der Probleminstanz. Solange beide Mengen ungleich sind $(M \neq M')$, wird das Verfahren fortgesetzt.

Shifting-Bottleneck-Verfahren eignen sich zwar grundsätzlich als Lösungsverfahren für die betrachteten Job-Shops und Flexible-Flow-Shops, allerdings hat die Lösung aufgrund des angestrebten Kompromisses oft große Abweichungen zum Optimum. Auch ist die erforderliche Rechenzeit für den (proaktiv-) reaktiven Einsatz zu hoch.[57] Ebenso ist das Generieren von einzelnen situativen Entscheidungen kaum möglich. Der Einsatz

[51] Vgl. u. a. (Domschke et al., 1997).
[52] Vgl. (Fleischmann, 1998, Seite 1367).
[53] Vgl. (Adams et al., 1988).
[54] Vgl. u. a. (Paternina-Arboleda et al., 2007; Acero-Domínguez und Paternina-Arboleda, 2004; Xu et al., 2003).
[55] Vgl. (Blazewicz et al., 2007, Seite 362 ff.).
[56] Vgl. (Domschke et al., 1997, Seite 408 ff.).
[57] Vgl. (Mönch, 2006, Seite 26); (Henning, 2002, Seite 104); (Blazewicz et al., 2007, Seite 362 ff.) und (Domschke et al., 1997, Seite 408 ff.).

von Shifting-Bottleneck-Verfahren zur Generierung von Trainingsbeispielen ist dennoch denkbar.

Simulationsverfahren

Die Simulation kann ebenfalls als Hilfsmittel zur Ablaufplanung und -steuerung eingesetzt werden[58] und ist als Methode zur Auswahlunterstützung zu bezeichnen.[59] Hierbei können die zuvor diskutierten Verfahren innerhalb der Simulation angewendet werden. *Semini et al.* geben eine gute Übersicht bzgl. des Einsatzes von diskreten ereignisgesteuerten Simulationsverfahren in Fertigungssystemen.[60] Auch bei Verfahren der Ablaufsimulationen sind – ebenso wie bei optimierenden Lösungsverfahren – nicht alle möglichen Lösungsvarianten in der zur Verfügung stehenden Zeit simulierbar, da dies einer vollständigen Aufzählung sämtlicher Lösungen des $\mathcal{NP}$-schweren Problems entsprechen würde. Stattdessen wird das Ziel auf eine höhere Ebene projiziert, d. h. durch den Einsatz der Simulation sollen Tendenzen erkannt und zweckmäßige Entscheidungen im Sinne der Zielfunktion identifiziert werden.[61]

Die meisten der in Simulationen getroffenen Entscheidungen beruhen auf Prioritätsregeln und deren passender Auswahl.[62] Wie beschrieben existiert keine dominante Regel. Aus diesem Grund wird das Problem in aktuellen Simulationsansätzen in Zeitintervalle zerlegt, in denen unterschiedliche Regeln verwendet werden. Ansätze, welche eine Regel für das beste Ergebnis über das gesamte Problem suchen, werden als Single-Pass Methoden bezeichnet. Die Unterteilung in Teilintervalle ist unter dem Begriff Multi-Pass bekannt. Eines der Probleme bei der Umsetzung derartiger Simulationsverfahren ist die Bestimmung einer geeigneten Größe der einzelnen Teilintervalle. Hierzu wurden u. a. Ansätze von *Wu und Wysk; Ishii und Talavage; Kim und Kim* vorgestellt.[63] Diese in den neunziger Jahren entwickelten Grundlagen haben bis heute Gültigkeit. Moderne Simulationsalgorithmen sind in der Regel Multi-Pass Verfahren, die den Vorteil von aufgeteilten Simulationszeiträumen nutzen. Teilweise werden die Simulationszeiträume dynamisch bestimmt oder andere Erweiterungen untersucht, um die Performance der Systeme weiter zu steigern. Ansätze hierzu werden beispielsweise von *Shiue und Guh; Metan und Sabuncuoglu; Yoo et al.* gegeben.[64]

[58] Vgl. (Teich, 1998).
[59] Vgl. (Schultz et al., 1995).
[60] Vgl. (Semini et al., 2006).
[61] Vgl. (Tavakkoli-Moghaddam und Daneshmand-Mehr, 2005).
[62] Vgl. (Semini et al., 2006) und (Manivannan und Banks, 1992, Seite 155 ff.).
[63] Vgl. (Wu und Wysk, 1989; Ishii und Talavage, 1991; Kim und Kim, 1994).
[64] Vgl. (Shiue und Guh, 2006; Metan und Sabuncuoglu, 2005; Yoo et al., 2004).

Wie beschrieben führt der Wechsel der Steuerungsregel bei Störungen im Fertigungssystem zu einer Verbesserung der Performance. Das Initiieren eines neuen Durchlaufs der Simulation bei einer Störung ist insbesondere deshalb wichtig, weil sich das Fertigungssystem vereinfacht haben könnte. So arbeitet die gewählte Regel zwar weiterhin innerhalb der zulässigen Performancegrenzen, jedoch wären mit einer anderen Prioritätsregel aufgrund des vereinfachten Systems noch bessere Ergebnisse erreichbar.

Simulationsbasierte Verfahren erfüllen die Positions- und Zeitpunktanforderung in gutem Maße und sind mit den aktuellen Multi-Pass-Ansätzen in der Lage, Prioritätsregeln auf ihre grundsätzliche Eignung zur Lösungserstellung zu untersuchen. Die situative Auswahl von Steuerungsregeln zur Erreichung einer hohen Lösungsqualität ist jedoch auch mit den Multi-Pass-Ansätzen nicht möglich. Für das Generieren von situativ guten Entscheidungen im Kontext der betrachteten Organisationsformen ist die benötigte Rechenzeit zu hoch.[65] Dennoch sind Simulationsverfahren grundsätzlich geeignet, um Trainingsbeispiele für die Lernkomponente zu generieren. Hierzu bedarf es vor der Verwendung der Beispiele zum Training einer Verarbeitung.

Analyse heuristischer Verfahren

Die Ergebnisse der vorangegangenen Analyse fasst Tabelle 3.4 zusammen. Mit heuristischen Verfahren zur Ablaufplanung und -steuerung sind zwar schnell Lösungen zur Verfügung zu stellen, die Qualität der Lösung ist aber selten optimal.[66] Heuristische Verfahren werden vielmehr dazu eingesetzt, um in kurzer Zeit annehmbare Lösungen zu erstellen. Für den hier betrachteten Untersuchungskontext im Fertigungsprozess ist keines der vorgestellten Verfahren direkt geeignet. Weiterentwicklungen sind erforderlich. Hierzu zeichnen sich das Verfahren von *Giffler und Thompson* sowie Prioritätsregeln in Kombination mit Verfahren der Künstlichen Intelligenz (maschinelle Lernverfahren) als vielversprechend ab.

	Reduktion der Komplexität	Bewältigung von Unsicherheiten	Minimierung der Rechenzeit	Maximierung der Lösungsqualität
Heuristische Verfahren	$\oslash \cdots \oplus$	$\oslash \cdots \oplus\oplus$	$\oslash \cdots \oplus\oplus$	$\ominus \cdots \oplus$

Tabelle 3.4.: Bewertung: Heuristische Verfahren[67]

[65] Vgl. (Suhl und Mellouli, 2006, Seite 293 ff.).
[66] Vgl. (Knauer, 2002, Seite 60).
[67] Legende zur Erfüllung der Anforderungen bei der Bewertung: $\ominus\ominus$ = sehr schlecht, $\ominus$ = schlecht, $\oslash$ = einigermaßen, $\oplus$ = gut und $\oplus\oplus$ = sehr gut.

3.2. Analyse von Lernverfahren

Funktionslernen aus Beispielen ist für die Lernkomponente eine geeignete Art der maschinellen Lernverfahren, da diese die Lösungsmethode des Trainingsverfahrens erlernen können. Das Funktionslernen aus Beispielen gehört zu den populärsten, am häufigsten genutzten Lernaufgaben des maschinellen Lernens.[68]

Sei O eine Menge möglicher Instanzbeschreibungen, D eine Wahrscheinlichkeitsverteilung auf O, C eine Menge möglicher Zielwerte (Klassen) und F eine Menge zulässiger Funktionen. Funktionslernen aus Beispielen ist, bei einer gegebenen Menge E von Beispielen der Form $(o, c) \in O \times C$, für die gilt $y = f(o)$, für eine unbekannte Funktion f eine Funktion $f' \in F$ zu finden, so dass der wahre Fehler $err(f', f)$ von f' im Vergleich zu f, bei gemäß Verteilung D gezogenen Instanzen aus O, möglichst gering wird.[69]

Die Lernaufgabe A des Funktionslernens aus Beispielen ist also die Approximation einer unbekannten Funktion f, welche die Instanzen einer definierten Form einer gegebenen Menge von Klassen klassifiziert.[70] Hierbei ist die Erfahrung E die Anzahl der bereits von f klassifizierten Instanzen und das Performancemaß P durch den Anteil korrekt zugeordneter Instanzen gegeben. Daraus ergibt sich die vereinfachte Definition eines durch das Funktionslernen aus Beispielen erstellten Klassifizierers:

Ein Klassifizierer sei eine Funktion f annähernde Funktion f', so dass f und f' sich möglichst ähnlich sind. Wenn f und f' ungleich sind, macht der Klassifizierer Fehler und kann nicht alle Eingaben korrekt klassifizieren. Ein Klassifizierer hat eine hohe Klassifikationsgenauigkeit, wenn er wenig Fehler macht.

Damit bestmögliche Ergebnisse mittels Methoden des maschinellen Lernens erreichbar sind, müssen die bereits klassifizierten Beispielinstanzen in E eine repräsentative Stichprobe aller in O vorkommenden Elemente abbilden. Die schon klassifizierten Instanzbeschreibungen müssen mit derselben Wahrscheinlichkeitsverteilung aus der Menge der möglichen Instanzbeschreibungen O gezogen werden, wie dieses im späteren Einsatzgebiet des erzeugten Klassifizierers sein wird.

Vor der Diskussion von Klassifikationsverfahren, wird im Folgenden auf deren Evaluierung im Einsatz eingegangen. Zur experimentellen Untersuchung von Klassifizierern eigenen sich verschiedene Vorgehensweisen bzw. Gütemaße.[71] Vordergründigstes Evaluierungsmaß ist die Fehlerquote bzw. die hieraus resultierende Klassifikationsgenauigkeit,

[68] Vgl. (Görz et al., 2003, Seite 521).
[69] Vgl. (Görz et al., 2003, Seite 523).
[70] Vgl. (Beierle und Kern-Isberner, 2008, Seite 102 f.).
[71] Vgl. (Tan et al., 2006; Witten und Frank, 2005).

da der Nutzen eines Klassifizierers mit steigender Klassifikationsgenauigkeit zunimmt.[72] Um die Fehlerrate zu berechnen, werden vom trainierten Klassifizierer Testinstanzen klassifiziert. Anschließend ist die Fehlerrate aus dem Verhältnis von falsch klassifizierten Testinstanzen zur Anzahl aller klassifizierten Testinstanzen zu berechnen. Die Klassifikationsgenauigkeit $(100 - Fehlerrate)$ spiegelt die Erfolgsrate des Klassifizierers wider. Von besonderem Interesse ist, wie sich ein Klassifizierer zur Laufzeit einer Anwendung verhält. Aufgrund der häufig unterschiedlich ausfallenden Klassifikationsgenauigkeiten bei der Klassifikation von unbekannten oder bekannten Testinstanzen ist zu berücksichtigen, welche Instanzen vom Klassifizierer während der Laufzeit zu klassifizieren sind.[73] Instanzen gelten als bekannt, falls sie für das Training des Klassifizierers Verwendung finden, anderenfalls als unbekannt. Muss der Klassifizierer im Einsatz unbekannte Instanzen zuordnen, wie im Anwendungsszenario dieser Arbeit, sollten die zur Evaluierung eingesetzten Testinstanzen ebenfalls unbekannt sein.

Die Messung von Fehlerrate und Klassifikationsgenauigkeit als alleiniges Gütemaß für Klassifizierer ist nicht ausreichend, da verschiedene Klassifizierer damit nicht immer aussagekräftig miteinander zu vergleichen sind. Beispielsweise werden als Resultat von *Internetsuchmaschine A* auf eine Suchanfrage 100 Ergebnisse zurückgeliefert, von denen 40 dem gewünschten Inhalt entsprechen. Bei *Internetsuchmaschine B* werden 400 Ergebnisse zurückgeliefert. Hiervon entsprechen 160 dem gewünschten Inhalt. Weil beide eine Genauigkeit von 40 Prozent ausweisen, stellt sich die Frage, welche der beiden Suchmaschinen (Klassifizierer) das bessere Ergebnis liefert.[74] Für eine genauere Bewertung des Klassifizierers bezüglich der verwendeten Klassen $C = \{c_1, \ldots, c_n\}$ der Instanzen gibt es die Maße Recall[75] $rec(c_i)$, Precision[76] $prec(c_i)$ und F-Maß $F(c_i)$, wobei letztere eine Kombination aus Recall und Precision ist.[77] Gegeben sei die Menge der Testinstanzen $T^*_{c_i}$, die zu der Klasse c_i gehören, und die Menge T_{c_i} der Testinstanzen, die ein Klassifizierer zu c_i zuordnet. Dann sind Precision und Recall:

$$prec(c_i) = \frac{|T^*_{c_i} \cap T_{c_i}|}{|T_{c_i}|},\ rec(c_i) = \frac{|T^*_{c_i} \cap T_{c_i}|}{|T^*_{c_i}|} \tag{3.1}$$

[72] Siehe Abschnitt 2.2.3.1.
[73] Vgl. (Witten und Frank, 2005, Seite 127 ff.).
[74] Vgl. (Witten und Frank, 2005, Seite 155 ff.).
[75] Recall ist das Maß, das den Anteil der vom System erkannten relevanten Elemente der Klasse x angibt.
[76] Precision ist das Maß, das den Anteil der von einen System korrekt erkannten Elemente der Klasse x im Verhältnis zu allen zur Klasse x zugeordneten Elementen angibt.
[77] Vgl. (van Rijsbergen, 1979) und (Witten und Frank, 2005, Seite 155 ff.). Weitere Maße zur Auswertung wie beispielsweise die falschen positiven (FP) gegenüber falschen negativen Klassifikationen (FN) finden sich u. a. in (Witten und Frank, 2005).

Das F-Maß als Kombination aus Precision und Recall ergibt sich mit:

$$F(c_i) = \frac{2}{1/prec(c_i) + 1/rec(c_i)} \tag{3.2}$$

Ein Beispiel zu Berechnung von Precision, Recall und F-Maß ist in Abschnitt A.5 aufgeführt.[78]

3.2.1. Entscheidungsbaumverfahren

Görz et al. beschreiben Entscheidungsbäume als zu den häufig verwendeten Klassifikationsverfahren gehörend, die insbesondere für die Approximation von nominalen Funktionen geeignet sind. Der Vorteil von Entscheidungsbäumen liegt u. a. in deren einfacher Lesbarkeit. Entscheidungsbäume sind als Wenn-Dann-Regeln darstellbar und damit für den Anwender verständlich.[79] Ein Entscheidungsbaum besteht aus mindestens einem Wurzelknoten und ist ein endlicher Baum.[80] Der Wurzelknoten unterteilt die Menge X der möglichen Instanzbeschreibungen in paarweise disjunkte Teilmengen. An jedem inneren Knoten werden die Teilmengen rekursiv in weitere paarweise disjunkte Teilmengen aufgeteilt. Diese Zerlegung wird mit Hilfe eines Split-Kriteriums durchgeführt, das die Instanzbeschreibung testet. Ausgehende Äste eines inneren Knotens entsprechen einem möglichen Attributwert. Jedem Blattknoten ist ein Klasse $c \in C$ der möglichen Zielwerte zugeordnet. Entscheidungsbäume klassifizieren die Instanzen, indem sie, vom Wurzelknoten ausgehende, abwärts bis zu einem Blatt einsortieren. Durch die Konstruktion der Split-Kriterien folgt jedes Element genau einem Pfad von der Wurzel bis zu einem Blatt des Baumes. Dabei entspricht die Klasse des Blattes der Klassifikation des Elementes.

Die Zerlegung der Menge möglicher Instanzbeschreibungen mit dem Split-Kriterium wird durch den Test von Attributen vorgenommen. In welcher Reihenfolge die Attribute getestet werden, ist nicht festgelegt, hat jedoch großen Einfluss auf die Struktur des Entscheidungsbaumes. Nachfolgend werden beispielhaft der ID3-Algorithmus und dessen Strategie zur Auswahl des Split-Kriteriums beschrieben.[81] Die Split-Kriterien sollen möglichst so ausgewählt werden, dass diese die Trainingsmenge nach ihrer Klassenzugehörigkeit aufteilen. Demzufolge ist ein Attribut besonders für die Klassifikation mit

[78] Für weitere Beispiele sei u. a. auf (Witten und Frank, 2005) verwiesen.
[79] Vgl. (Görz et al., 2003).
[80] Siehe Abschnitt 3.2.
[81] Für weitere zum Entscheidungsbaumlernen eingesetzte Algorithmen vgl. u. a. (Witten und Frank, 2005), (Russel und Norvig, 2003, Seite 653 ff.), (Klösgen und Żytkow, 2002, Seite 267 ff.) und (Mitchell, 1997, Seite 52 ff.).

hohem Informationsgewinn geeignet, wenn es die Trainingsmenge so aufteilt, dass die neu entstehenden Kindknoten einer Klasse genau zugeordnet werden können. Der vom Split-Kriterium erzeugte Informationsgewinn wird beim ID3-Algorithmus über die Abnahme der Entropie innerhalb der neu entstehenden Kindknoten determiniert. Entropie beschreibt hierbei die Reinheit (Uniformität) einer Menge bezüglich der Klasse, der in der Menge enthaltenen Elemente. Entropie ist wie folgt berechenbar:[82]

$$Entropie(T) = \sum_{i=1}^{C} -p_i \log_2 p_i \tag{3.3}$$

Wobei C die Anzahl der möglichen Klassen und p_i den Anteil der Klasse i an der Menge der Beispiele in T angibt. Mittels Gleichung 3.3 kann jetzt der erreichte Informationsgewinn durch die Auswahl des Attributes A für eine Trainingsmenge T definiert werden:[83]

$$Informationsgewinn(A, T) = Entropie(T) - \sum_{k \in Klassen(A)} \frac{|T_k|}{|T|} \cdot Entropie(T_k) \tag{3.4}$$

$Klassen(A)$ steht für die Menge aller möglichen Wertebelegungen von Attribut A. Die Anzahl der Beispiele mit der Ausprägung k im Attribut A kennzeichnet $|T_k|$, $|T|$ die Anzahl der Beispiele in der Menge T insgesamt.

Der ID3-Algorithmus überprüft sämtliche Attribute auf ihre Tauglichkeit als Split-Attribut. Anschließend wird das Attribut mit dem höchsten Informationsgewinn ausgewählt. Dieses Verfahren wird rekursiv bis zur Erreichung einer Abbruchbedingung auf alle neuen Kindknoten angewandt: Die Entropie eines Knotens ist null, da alle Trainingsbeispiele innerhalb des Knotens derselben Klasse angehören, dann wird dem Blattknoten die entsprechende Klasse zugeordnet. Sämtliche Attribute auf dem Pfad des Split-Elements wurden eingesetzt, folglich wird dem Blattknoten die Klasse derjenigen Instanz mit der größten Ausprägung innerhalb des Knotens zugeordnet.

Es werden also die Attribute mit dem höchsten Informationsgewinn bevorzugt. Die rekursive Zerlegung in Teilmengen wird abgebrochen, sobald die Elemente einheitliche Klassen besitzen. Hierdurch weisen mittels des ID3-Algorithmus erzeugte Entscheidungsbäume zwei Merkmale auf: Attribute, die einen hohen Informationsgewinn erzielen, sind nahe der Wurzel angeordnet. Die generierten Entscheidungsbäume haben eine geringe Größe, sie sind jedoch mit der Trainingsmenge konsistent. Aufgrund der Tatsache, dass kleine Entscheidungsbäume erzeugt werden, kann tendenziell schneller

[82] Vgl. (Beierle und Kern-Isberner, 2008, Seite 116).
[83] Vgl. (Mitchell, 1997, Seite 55 ff.).

klassifiziert werden. Auch entsprechen kleinstmögliche Entscheidungsbäume dem Prinzip der Sparsamkeit in der Wissenschaft (Occam's Razor).[84] Dieses besagt, dass bei mehreren Theorien, die einen Sachverhalt erklären, die einfachste zu bevorzugen ist. Für eine spätere Beschneidung (pruning) des Entscheidungsbaumes ist der Umstand, dass wichtige Attribute in der Nähe des Wurzelknotens sind, ebenfalls von Vorteil. Die Beschneidung des Baumes wird zur Vermeidung von Überanpassung durchgeführt.

Überanpassung (Overfitting) bedeutet, dass sich der Entscheidungsbaum (Klassifizierer) zu stark an die Trainingsbeispiele angepasst hat.

Eine Funktion $h \in H$[85] heißt genau dann überangepasst (overfitted) an die Trainingsmenge, wenn eine Funktion $h' \in H$ existiert, so dass h einen kleinen Trainingsfehler aber einen größeren wahren Fehler als h' besitzt.

Durch das vorgestellte Verfahren werden die Äste des Entscheidungsbaumes derart aufgeteilt, dass die Trainingsmenge fehlerfrei klassifiziert wird. Auf den ersten Blick erscheint der Einsatz dieser Strategie sinnvoll, kann in der Praxis allerdings zu Schwierigkeiten führen. Immer dann, wenn die Trainingsmenge keine repräsentative Stichprobe der Menge von möglichen Instanzbeschreibungen ist oder die Trainingsmenge (beispielsweise durch verrauschte Messdaten) unkorrekt ist, kommt es zu Fehlern. Dann wird vom Algorithmus ein Entscheidungsbaum erzeugt, der überangepasst an die Trainingsmenge ist.

Die Überanpassungsgefahr besteht immer dann, wenn Daten verrauscht sind oder einzelne Daten Fehler bei den Werten der Attribute oder Klassen enthalten. Enthält die Trainingsmenge zufällige Regelmäßigkeiten, die keinerlei Bezug zur gesuchten Funktion f aufweisen, kann es auch bei fehlerfreien Daten zu einer Überanpassung kommen. In einer Studie kommt *Mingers* zu dem Ergebnis, das sich die Performance des ID3-Algorithmus durch die Überanpassung bei verrauschten Daten um 10-25 Prozent verschlechtert.[86] Zur Vermeidung von Überanpassung ist ein erzeugter Klassifizierer hierauf zu testen. Ist der Trainingsfehler des Entscheidungsbaumes kleiner als der Fehler auf einer unabhängigen Testmenge, kann von einer Überanpassung ausgegangen werden. Da sich ein Entscheidungsbaum mit steigender Komplexität immer exakter der Trainingsmenge anpasst, ist die Komplexität des Baumes zu verringern, um der Überanpassung entgegen zu wirken. Hierzu ist der Entscheidungsbaum mit dem sogenannten Pruning-Verfahren von unten, d. h. von den Blättern stutzen. Dabei werden die inneren

[84] Vgl. (Blumer et al., 1987).
[85] Sei H die Menge zulässiger Funktionen.
[86] Vgl. (Mingers, 1989b).

Knoten oberhalb der Blattknoten testweise in Blattknoten umgewandelt. Als Klassifizierung wird den neuen Blattknoten, die in dem von diesen aufgespannten Unterbaum am häufigsten vorkommenden Klassenmenge zugewiesen. Anschließend wird der neue Entscheidungsbaum erneut mit der Testmenge überprüft und entsprechend der Veränderung angepasst, welche die größte Verbesserung erzielt. Dieser Vorgang wird so lange für die restlichen inneren Knoten durchgeführt, bis der Entscheidungsbaum auf eine Größe schrumpft, so dass weitere Vereinfachungen keine Verbesserung ergeben.

Für den Einsatz in der Praxis existieren unterschiedliche Modifikationen. Diese ändern die grundsätzliche Methodik des Entscheidungsbaumlernens nicht, tragen aber verschiedenen Herausforderungen der Praxis Rechnung. Derartige Ansätze setzen sich beispielsweise mit der Nutzung von kontinuierlichen nummerischen Attributwerten sowie fehlenden Attributwerten auseinander.[87] Der ID3-Algorithmus ist nur in der Lage, diskrete Attributwerte zu verarbeiten, weil andernfalls die Anzahl der Äste aus einem Knoten zu groß ist. Im praktischen Einsatz der Ablaufplanung und -steuerung in Job-Shops und Flexible-Flow-Shops sind jedoch oft kontinuierliche nummerische Attribute zu finden.[88] Damit diese vom Entscheidungsbaum verarbeitet werden können, ist der Wertebereich in wenige disjunkte Intervalle zu unterteilen. *Fayyad* und *Fayyad* untersuchten hierzu die Berechnung der optimalen Grenzwerte für die Intervalle.[89] Zum Umgang mit fehlenden Attributwerten wurde von *Mingers* eine Strategie untersucht, in der fehlende Attributwerte durch den am häufigsten vorkommenden Attributwert unter einer Menge von ähnlichen Beispielen ergänzt werden. Alle Beispiele, die einen gemeinsamen Pfad zu dem Knoten wie das zu Testende besitzen, werden in Betracht gezogen.[90] *Quinlan* beschreibt den C4.5-Algorithmus, eine Weiterentwicklung des ID3-Algorithmus, zum Umgang mit fehlenden Attributwerten.[91]

Analyse von Entscheidungsbäumen

Entscheidungsbäume könnten als Lernkomponente eingesetzt werden. Sie sind in der Lage, mit einer großen Anzahl von beschreibenden Merkmalen und großen Trainingsmengen zu arbeiten. Die Klassifizierung einer unbekannten Instanz erfolgt sehr schnell, da maximal so viele einfache Tests erforderlich sind, wie Merkmale existieren. Mit den dargelegten Erweiterungen können Entscheidungsbäume auch mit fehlenden und numerischen Attributen umzugehen. Jedoch kann dadurch die benötigte Zeit für eine Klas-

[87] Vgl. (Witten und Frank, 2005; Mitchell, 1997; Quinlan, 1993).
[88] Siehe Abschnitt 2.2.3.5.
[89] Vgl. (Fayyad, 1992; Fayyad und Irani, 1992).
[90] Vgl. (Mingers, 1989a).
[91] Vgl. (Quinlan, 1993). Vgl. auch (Klösgen und Żytkow, 2002, Seite 267 ff.).

sifizierung leicht ansteigen. Ebenso ist eine Erweiterung der Trainingsmenge problematisch, da dazu der gesamte Baum neu aufzubauen ist. Tabelle 3.5 zeigt die Bewertung von Entscheidungsbäumen anhand der in Abschnitt 2.2.3 dargelegten Anforderungen an eine Lernkomponente.

	Approximation	Adaptierbarkeit	Entscheidungssicherheit	Lern- und Vorhersagezeit	Kardinale Merkmale
Entscheidungsbäume	⊕⊕	⊘	⊘	⊕⊕	⊕

Tabelle 3.5.: Bewertung: Entscheidungsbäume[92]

3.2.2. Künstliche Neuronale Netze

Künstliche Neuronale Netze stellen eine weitere Methode zum Erlernen von Funktionen aus Beispielen dar. Sie basieren auf einer Analogie menschlicher Gehirnstrukturen, versuchen deren Lern- und Gedächtnisfähigkeit nachzubilden und bestehen aus künstlichen Neuronen, den sogenannten Perzeptrons[93]. Ein Perzeptron besitzt x Eingänge sowie x den Eingängen zugehörige Gewichte und einen Ausgang. Durch die Anwendung des Skalarprodukts aus Eingangsvektor und Gewichtsvektor auf die Übertragungsfunktion[94] wird der Wert des Ausgangs eines Perzeptrons berechnet. Perzeptrons sind oft in die drei Schichten: Eingabeschicht, verdeckte Schicht und Ausgabeschicht unterteilt.

Aus der Verbindung vieler Perzeptrons entsteht ein Künstliches Neuronales Netz. Dieses lernt aus der Trainingsmenge, durch Modifizierung der Gewichte an den Eingängen der Perzeptrons, so dass die Trainingsinstanzen korrekt klassifiziert werden. Kantengewichte geben die Verbindungsstärke zwischen den jeweiligen Knoten unterschiedlicher Schichten an. Zusätzlich existiert für jede Kante ein sogenannter Node-bias-Wert. Hierbei handelt es sich um eine Konstante, die den Einflussgrad des Knotens zur Ermittlung des neuen Ergebniswertes angibt.[95] Beide Werte werden vor Trainingsbeginn mit klei-

[92] Legende zur Erfüllung der Anforderungen bei der Bewertung: ⊖⊖ = sehr schlecht, ⊖ = schlecht, ⊘ = einigermaßen, ⊕ = gut und ⊕⊕ = sehr gut.

[93] Vgl. (Russel und Norvig, 2003, Seite 740) und (Tan et al., 2006, Kapitel 5.4.1).

[94] Die Übertragungsfunktion, auch Aktivierungsfunktion, beschreibt den zukünftigen Erregungszustand einer Zelle i lokal durch die Zustände der verbundenen Zellen e_j. Die bei weitem am häufigsten eingesetzte Übertragungsfunktion ist die Sigmoid-Funktion $f(x) = 1/1 + e^{-\alpha x}$ (vgl. Görz et al., 2003, Seite 84).

[95] Vgl. (Shmueli et al., 2007, Seite 170).

nen Zufallswerten initialisiert. Diese repräsentieren den Anfangszustand ohne Wissen über das Künstliche Neuronale Netzwerk.

Die Konstruktion eines Künstlichen Neuronalen Netzes ist eine schwierige und zeitintensive Aufgabe, die sehr viel Wissen über die zugrundeliegende Problemstellung erfordert. Hier gilt es, sowohl die Anzahl an Zwischenschichten als auch die Anzahl an Knoten der Zwischenschichten und Ausgabeschicht zu bestimmen.[96] Häufig werden deshalb mehrere Architekturen ausprobiert oder Erfahrungswerte der Vergangenheit genutzt.[97]

Beim Training werden Fehler, die das erstellte Netz auf den Trainingsinstanzen macht, gemessen und die Gewichte der einzelnen Perzeptrons mittels des Gradienten-Abstiegs-Verfahrens fehlerminimierend modifiziert. Dieser Vorgang ist so lange iterativ zu wiederholen, bis das erstellte Netz das gewünschte Verhalten zeigt. Aus dem Vergleich zwischen bekannter Klasse der Trainingsinstanz und vorhergesagter Klasse des Lernverfahrens kann eine mögliche Abweichung abgeleitet werden. Anschließend wird dieser Fehler, ausgehend von der Ausgabeschicht, zu den betroffenen Zwischenschichten zurückgegeben. Dieser Vorgang wird als Rückwärtspropagierung – Backpropagation – bezeichnet. Auf jeder beteiligten Schicht sind neue Kantengewichte sowie Node-bias-Werte zu berechnen.

Künstliche Neuronale Netze verfügen über die Fähigkeit, sehr komplexe Zusammenhänge darzustellen. Folglich besitzen sie eine gute Klassifikationsleistung bei schwierigen Problemstellungen. Hierzu müssen die Kantengewichte jedoch optimal eingestellt werden, was bereits bei einfachen Problemstellungen eine hohe Anzahl verfügbarer Trainingsinstanzen sowie ausreichend Trainingszeit benötigt. Ein Problem Künstlich Neuronaler Netze stellt die Bestimmung der besten Netzgröße dar. Analog zu Entscheidungsbäumen ist ein Künstliches Neuronales Netz mit steigender Größe in der Lage, sich komplexen Funktionen anzunähern. Allerdings wächst mit steigender Größe des Netzes die Gefahr der Überanpassung. Hier ist eine Abstimmung zwischen Trainingsdauer und Genauigkeit zu finden.

Analyse von Neuronalen Netzen

Problematisch sind qualitativ schlechte Trainingsdaten. Diese führen während des Rückpropagierungs-Schrittes zu abweichenden Ergebniswerten und einer starken Anpassung der Kantengewichte sowie Bias-Werte. Aus diesem Grund ist eine gezielte Auswahl von

[96] Weitere Informationen zur Konstruktion von Neuronalen Netzen sind u. a. in (Tan et al., 2006, Seite 255) zu finden.
[97] Vgl. (Shmueli et al., 2007, Seite 181).

Daten und Merkmalen Voraussetzung zur Erzielung guter Leistungen. Der Lernfaktor[98] beeinflusst die Berechnung neuer Kantengewichte und Bias-Werte und hilft diesem Problem vorzubeugen. Die Laufzeit von Künstlichen Neuronalen Netzen steigt mit der Anzahl verwendeter Merkmale, da mehr Gewichte berechnet werden müssen. Weil die Kantengewichte während des Trainingsprozesses fortlaufend angepasst werden, ist der Umgang mit redundanten Merkmalen unproblematisch. Gewichte derartiger Merkmale sind in der Regel sehr klein und beeinflussen den Klassifizierungsvorgang nicht. Tabelle 3.6 zeigt die Bewertung Künstlicher Neuronaler Netze anhand der dargelegten Anforderungen an eine Lernkomponente.

	Approximation	Adaptierbarkeit	Entscheidungssicherheit	Lern- und Vorhersagezeit	Kardinale Merkmale
Künstliche Neuronale Netze	⊕⊕	⊖	⊕	⊖⊖	⊕⊕

Tabelle 3.6.: Bewertung: Künstliche Neuronale Netze[99]

Die größte Schwäche von Künstlichen Neuronalen Netzen liegt in der benötigten langen Trainingszeit. Diese ist ungleich länger als die erforderliche Zeit für den Aufbau eines Entscheidungsbaums. Ist das Künstliche Neuronale Netz zu adaptieren, wird hierzu viel Zeit benötigt. Auch sieht der Anwender das Künstliche Neuronale Netz immer nur als eine „Blackbox“. Die modellierten Strukturen bleiben zum Nachteil des Systemverständnisses im Inneren des Netzes als versteckte Schichten verborgen, womit Entscheidungen nicht nachvollziehbar sind.

3.2.3. Naive-Bayes-Klassifizierer

Ein weiteres Verfahren zum Erlernen von Funktionen aus Beispielen ist das Naive-Bayes-Klassifikationsverfahren. Dieses Verfahren nutzt einen anderen Ansatz als den der bisher vorgestellten Entscheidungsbäume und Künstliche Neuronale Netze. Es ist ein probabilistisches Klassifikationsverfahren, da es auf statistischen Wahrscheinlichkeiten basiert. Der Naive-Bayes-Klassifizierer erzeugt auf Grundlage des Bayes'schen-Theorems eine mathematische Funktion f', die jedem Punkt X des Merkmalsraums eine Klasse Y

[98] Vgl. (Shmueli et al., 2007, Seite 173).

[99] Legende zur Erfüllung der Anforderungen bei der Bewertung: ⊖⊖ = sehr schlecht, ⊖ = schlecht, ⊘ = einigermaßen, ⊕ = gut und ⊕⊕ = sehr gut.

zuordnet. Dabei wird die Trainingsmenge nicht aufgeteilt, so dass jedes der Trainingsbeispiele die Wahrscheinlichkeit für eine Klasse an einem Punkt im Merkmalsraum beeinflussen kann. Der Lernvorgang eines Naive-Bayes-Klassifizierers besteht in der Bestimmung bedingter Wahrscheinlichkeiten. Unter der naiven Annahme, die Merkmale in X seien bedingt unabhängig,[100] ist die Formel des Naive-Bayes-Klassifizierer wie in Gleichung 3.5 gegeben. Dabei stellen x_j die Merkmale und y_i die Klassen dar.[101]

$$y = \underset{(y_i \in Y)}{\operatorname{argmax}} \left(P(y_i) \cdot \prod_{j=1}^{n} P(x_j|y_i) \right) \tag{3.5}$$

Die Anzahl von erforderlichen Beispielen steigt beim Naive-Bayes-Klassifizierer nur linear mit der Anzahl der Attribute.[102] Aus diesem Grund ist das Naive-Bayes-Verfahren ein beliebter Ansatz für praxisrelevante Probleme. Allerdings ist die Annahme, dass die beobachteten Attribute bedingt unabhängig voneinander sind, stark vereinfachend und trifft im Allgemeinen nicht zu. Dennoch zeigt die Erfahrung, dass mit dem Naive-Bayes-Verfahren trotzdem sehr gute Ergebnisse erzielt werden.[103] Aus dem Aufbau des Naive-Bayes-Klassifizierers ergeben sich zwei zu beachtende Punkte, wenn der Algorithmus erfolgreich eingesetzt werden soll.[104] Attribute werden als bedingt unabhängig betrachtet, deshalb ist eine sorgfältige und gezielte Attributauswahl vorzunehmen. Da die errechneten Wahrscheinlichkeiten Produkte aus Einzelwahrscheinlichkeiten sind, dürfen keine Null-Wahrscheinlichkeiten vorkommen. Diese würden die Terme dominieren. Bei der *Attributwahl* können abhängige Attribute die Leistung eines Naive-Bayes-Klassifizierers vermindern. Falls im Extremfall dem Merkmalsraum ein neues Attribut x_{n+1} hinzugefügt wird, das mit einem vorhandenen Attribut x_n vollkommen identisch ist, gehen die Einzelwahrscheinlichkeiten des Attributs x_n, im Vergleich mit dem Merkmalsraum ohne das Attribut x_{n+1}, quadratisch in die Gleichung ein. Dieser Effekt verschlimmert sich mit der Anzahl von identischen Attributen. Demzufolge bekommt das Merkmal eine wesentlich höhere Gewichtung. Um Trainingsdaten mit korrelierenden Attributen so zu modifizieren, dass sie von einem Naive-Bayes-Klassifizierer besser nutzbar sind, existieren verschiedene Techniken. Sie basieren auf der Wahl einer möglichst geeigneten Untermenge der Attribute.[105] Diese sind allerdings extrem rechenintensiv und ersetzen

[100] Zufallsvariable A heißt bedingt unabhängig von B bei gegebenem C wenn eine der drei äquivalenten Bedingungen erfüllt ist: $P(A|B,C) = P(A|B)$ oder $P(B|A,C) = P(B|C)$ oder $P(A,B|C) = P(A|C) \cdot P(B|C)$

[101] Vgl. (Mitchell, 1997, Seite 182 f.).

[102] Vgl. (Witten und Frank, 2001, Seite 352).

[103] Vgl. (Mitchell, 1997, Seite 128 f.).

[104] Vgl. (Witten und Frank, 2005, Seite 84 f.).

[105] Eingesetzte Techniken sind z. B. die verfahrensspezifische Attributauswahl mit Rückwärtselimination oder Vorwärtsauswahl sowie die Zusammenfassung korrelierender Attribute zu einem (multivariaten)

nicht die gute Vorauswahl der Attribute. Die Laplace-Erweiterung wird eingesetzt, um das Problem von *Nulltermen* zu beheben.[106] Das Naive-Bayes-Klassifikationsverfahren bildet für die Berechnung der Wahrscheinlichkeit der Zugehörigkeit einer Instanz zu einer Klasse y_i das Produkt über alle Terme $P(x_j|y_i)$. Existiert in der begrenzt großen Trainingsmenge kein Beispiel für einen Wert, nimmt dieser Term den Wert null an und dominiert unerwünschter Weise das Produkt. Da die Ausprägung eines einzelnen Attributes in Kombination mit der Klasse y_i nicht in der Trainingsmenge vorhanden ist, wird die Wahrscheinlichkeit für die Klasse y_i, unabhängig wie große der Wert der restlichen Terme ist, mit null berechnet. Mit der Laplace-Erweiterung werden virtuelle Instanzen in die Trainingsmenge eingefügt. Auf diese Art und Weise kann vermieden werden, dass einzelne Nullterme die Wahrscheinlichkeitsberechnungen für eine Klasse dominieren und keine Null-Wahrscheinlichkeiten mehr auftreten. Die technische Umsetzung lautet wie folgt: Die Zähler aller Terme $P(x_j|y_i) = y_{x_j}/y$[107] werden für jeweils ein virtuell beobachtetes Beispiel um eins erhöht. Gleichzeitig sind die Nenner für die mit den virtuellen Beispielen um n vergrößerte Trainingsinstanz um n zu erhöhen. Durch das Einfügen von virtuellen Beispielen werden die beobachteten relativen Häufigkeiten verändert. Ist eine a priori Wahrscheinlichkeitsverteilung $(P_1, \ldots, P_n)|P_i > 0, \sum_{i=1}^{n} P_i = 1$ über die Attribute bekannt, kann diese zur detaillierteren Modellierung verwendet werden. Dann wird auf den Zähler des Attributs x_j nicht 1, sondern $1 \cdot P_j$ addiert. In der Praxis sind diese a priori Wahrscheinlichkeitsverteilungen häufig nicht bekannt. Jedoch haben diese einen so geringen Einfluss,[108] dass in der Regel eine einfach eingesetzte Laplace-Erweiterung ausreicht.[109]

Immer dann, wenn ein Attribut einem nominalen Wertebereich unterliegt, ist die Anwendung des Naive-Bayes-Klassifizierers einfach. Die Wahrscheinlichkeiten sind wie beschrieben ermittelbar. Numerische Attribute sind allerdings nicht so einfach wie nominale Attribute auszuzählen.[110] Beispielsweise ist ein Attributwert x_j von 10 ungleich einem Wert x_j von 11. Dennoch ist auszudrücken, dass eine Instanz mit dem nummerischen Attributwert $x_j = 10$ einer Instanz mit $x_j = 11$ ähnlicher ist als einer Instanz mit einem Wert $x_j = 83$. Daher wird angenommen, dass die beobachteten Attributwerte einer Verteilungsfunktion unterliegen.[111] Für jede Klasse y_i werden Mittelwert und Standardabweichung der ermittelten Werte eines numerischen Attributs berechnet, um

Attribut (Clustering) (vgl. Witten und Frank, 2005, Seite 210 ff.).

[106] Vgl. (Mitchell, 1997, Seite 175 f.) und (Witten und Frank, 2005, Seite 84 f.).

[107] Mit y als Anzahl der Trainingsbeispiele mit Klasse y_i und y_{x_j} als Anzahl der Trainingsbeispiele mit $x_j = wahr$ unter den Beispielen von y.

[108] Eine ausreichend große Trainingsmenge sei vorausgesetzt.

[109] Vgl. (Witten und Frank, 2005, Seite 85).

[110] Siehe Abschnitt 3.2.1.

[111] Üblicherweise einer Normalverteilung, die im Bedarfsfall jedoch angepasst bzw. ersetzt werden kann.

anschließend die Wahrscheinlichkeit mit Hilfe der Wahrscheinlichkeitsdichtefunktion zu ermitteln. Diese gibt die Wahrscheinlichkeit dafür an, dass der Wert eines numerischen Attributs im Bereich ϵ um den Punkt y liegt. Dieser Wert eignet sich für den Einsatz als bedingte Wahrscheinlichkeit im Naive-Bayes-Klassifikationsverfahren; das Ergebnis der Wahrscheinlichkeitsdichtefunktion für einen gegebenen Wert eines numerischen Attributs[112] ist als bedingte Wahrscheinlichkeit $P(Attribut = Wert|Klasse)$ nutzbar.[113]

Analyse von Naive-Bayes-Klassifizierern

Der Naive-Bayes-Klassifizierer erfüllt alle gestellten Anforderungen an eine Lernkomponente. Der Klassifizierer kann mit vielen Merkmalen und einer großen Trainingsmenge umgehen. Die Klassifikation erfolgt schnell und fehlende Attributwerte beeinträchtigen den Klassifizierer nicht. Tabelle 3.7 zeigt die Bewertung des Naive-Bayes-Klassifizierers anhand der in Abschnitt 2.2.3 dargelegten Anforderungen. Durch die Annahme, dass alle Attribute bedingt unabhängig sind[114] und unter Berücksichtigung aller Trainingsdaten für jede Klassifikationsentscheidung,[115] ignoriert der Naive-Bayes-Klassifizierer Zufallsattribute zuverlässig. Zu den Stärken des Naive-Bayes-Klassifizierers gehört, dass jedes Trainingsbeispiel in die Berechnung der Wahrscheinlichkeit zur Zugehörigkeit einer Instanz zu einer Klasse mit einfließt. Überdies kann eine Hypothese auch aufrecht gehalten werden, wenn dieser ein einzelnes Trainingsbeispiel widerspricht. Ebenso ist der Trainingsdatensatz einfach erweiterbar. Zusätzlich zu der Klassifikation wird beim Naive-Bayes-Klassifizierer eine Wahrscheinlichkeitsverteilung über alle Klassen berechnet. Damit ist die Ausgabe der Zuverlässigkeit des Ergebnisses der Klassifikation sowie wahrscheinlicher Alternativen möglich.

	Approximation	Adaptierbarkeit	Entscheidungssicherheit	Lern- und Vorhersagezeit	Kardinale Merkmale
Naïve-Bayes-Klassifizierer	$\oplus\oplus$	$\oplus\oplus$	$\oplus\oplus$	$\oplus\oplus$	$\oplus\oplus$

Tabelle 3.7.: Bewertung: Naive-Bayes-Klassifizierer[116]

[112] Sowie Mittelwert und Standardabweichung für das numerische Attribut.
[113] Vgl. (Witten und Frank, 2005, Seite 85 ff.).
[114] Diese Annahme ist insbesondere für ein komplett unabhängiges Zufallsattribut korrekt.
[115] Andere Verfahren zerlegen die Menge der Trainingsbeispiele, wodurch enthaltene „Zufallsattribut“ großen Einfluss erhalten können.
[116] Legende zur Erfüllung der Anforderungen bei der Bewertung: $\ominus\ominus$ = sehr schlecht, $\ominus$ = schlecht, $\oslash$ = einigermaßen, $\oplus$ = gut und $\oplus\oplus$ = sehr gut.

4. Zu leistende Arbeit

Im Folgenden wird der dargelegte Stand der Technik zur Ablaufplanung und -steuerung in Job-Shops und Flexible-Flow-Shops sowie zu Lernverfahren mit der dargelegten Problemstellung zu den sich hieraus ergebenden Arbeitspaketen verbunden. Eine Aufteilung in Arbeitspakete ist zweckmäßig, um das aufgeworfene Problem sukzessive zu lösen. Die Aufgliederung der Arbeitspakete orientiert sich an den in Kapitel 2 erarbeiteten Anforderungen an ein Lösungsverfahren.

4.1. Entwicklung einer Steuerungskomponente

Im ersten Schritt soll eine Heuristik für Job-Shop-Fertigungssysteme entwickelt werden, um eine effiziente Steuerungskomponente sicherzustellen. Diese ist in einem zweiten Schritt für den Einsatz in Flexible-Flow-Shops-Systemen zu erweitern. Dabei ist das hier vorherrschende Auswahlproblem zu berücksichtigen. Grundsätzlich ist auf die Beschränkung des Lösungsraumes zu achten.

Die Ablaufsteuerungsheuristik muss einen Entscheidungssituation fordernden Bestandteil enthalten. Hier ist ein Lernverfahren (Lernkomponente) nach Art trainierter Regeln zu integrieren. In Konfliktsituationen sind geeignete Steuerungsregeln auszuwählen, wobei in Flexible-Flow-Shop-Umgebungen zwei Entscheidungen zu treffen sind.

4.2. Entwicklung einer Lernkomponente

Wissensbasierte Auswahlmethode

Es ist ein wissensbasiertes Verfahren als Klassifizierer zur situationsabhängigen Auswahl von Steuerungsregeln zu entwickeln. Hierbei muss ein für den vorliegenden Untersuchungsgegenstand passendes Lernverfahren in der Lage sein, den in Abschnitt 2.2.3

gestellten Anforderungen gerecht zu werden. Ergebnis ist eine wissensbasierte Auswahlmethode (Klassifikationsmethode), die einfach in die Steuerungskomponente zu integrieren ist.

Situationsmerkmale

Aufgabe dieses Teilarbeitspakets ist die Entwicklung einer initialen Menge von Attributen (Situationsmerkmalen) zur quantitativen Differenzierung von Systemsituationen. Dazu dienen Vorarbeiten wie die von *Chen und Yih*[1] oder *Priore et al.*[2] Es ist zwischen den Merkmalen für die betrachteten Organisationstypen zu unterscheiden. In Flexible-Flow-Shops ist zusätzlich zwischen lokalen und globalen Merkmalen zu differenzieren, da hier mehrere Fertigungsstufen betrachtet werden.

Steuerungsklassen

Schwerpunkt dieses Teilarbeitspaketes ist die Auswahl und Konzeption von geeigneten Steuerungsregeln als Klassen für die wissensbasierte Klassifikationsmethode. Hierbei sind Fragen zu beantworten wie: Welche Entscheidung kann mit welcher Regel abgebildet werden? Wie groß muss der Regelkatalog sein – so groß wie nötig, so klein wie möglich? Wie beeinflussen sich verschiedene Regeln gegenseitig? Eine initiale Menge von Steuerungsregeln ist zu definieren. Es ist zwischen Regeln für Job-Shops und Flexible-Flow-Shops zu unterscheiden, da in Flexible-Flow-Shops zur Auftragsauswahl noch eine zusätzliche Auswahl der Maschine erfolgen muss.

4.3. Entwicklung eines Trainingsverfahrens

Die Entwicklung geeigneter Methoden zum Training der Lernkomponente ist Aufgabe in Arbeitspaket 4.3. Für das Training in Job-Shops soll ein optimierendes Verfahren eingesetzt werden, für das in Flexible-Flow-Shop-Umgebungen ein Ablaufsimulator. Zur effizienten Generierung einer Vielzahl von Trainingsbeispielen sowie der Verfahrensevaluierung sind erforderliche Erweiterungen vorzunehmen.

[1] Vgl. (Chen und Yih, 1996).
[2] Vgl. (Priore et al., 2006).

4.4. Entwicklung einer Rückkopplung in das Trainingsverfahren

Aufgabe dieses Arbeitspakets ist die Veränderung des bestehenden statischen Systems in ein dynamisches System, welches die Adaption – simulative Informationserweiterung in Steuerungssituationen sowie permanente Adaption der wissensbasierten Auswahlmethode – ermöglicht. Gleichfalls wird der Simulator durch die Rückkopplung des entwickelten Verfahrens um dessen Steuerungsfunktionalität erweitert.

4.5. Evaluierung des Ablaufplanungs- und Steuerungsverfahrens

Aufgabe dieses Arbeitspakets ist die Evaluierung des entwickelten Verfahrens zur Ablaufplanung und -steuerung in den untersuchten Job-Shops und Flexible-Flow-Shops. Hierbei bedingen unterschiedliche Ziele unterschiedliche Kennzahlen zur Evaluierung. Da der Schwerpunkt dieses Verfahrens in der guten Zielerfüllung (minimaler Makespan) und geringen Lösungsfindungszeit liegt, sollen hierfür zunächst Kennzahlen wie Lösungsgeschwindigkeit, Lösungsapproximation und Trainingsmenge eingesetzt werden. Zur Evaluierung der Ablaufplanung und -steuerung sind Verfahren wie Vergleich zur zufälligen sowie fixen Auswahl von Steuerungsregeln anzuwenden. Ebenfalls werden die Ergebnisse des Verfahrens mit denen eines Offline-Verfahrens verglichen. Dazu wird die optimale Lösung mit den zum Entscheidungszeitpunkt unbekannten Ereignissen und ausreichender Berechnungszeit generiert.

5. Konzeption

Zur Lösung der vorgestellten Problemstellung ist im Folgenden ein Verfahren zur Ablaufplanung und -steuerung zu entwickeln. Abschnitt 5.1 gibt eine Übersicht über die Phasen und Schnittstellen des Verfahrens. Die grundsätzliche Ablaufplanungs- und Steuerungsheuristik für die beiden betrachteten Umgebungen wird in Abschnitt 5.2 vorgestellt. Darauf aufbauend wird die wissensbasierte Komponente zur Entscheidungsfindung in Abschnitt 5.3 aufgezeigt. Hierbei wird neben den Merkmalen zur Situationserkennung und den Steuerungsregeln als Klassen das Training vorgestellt. Der abschließende Abschnitt stellt das Konzept für die Rückkopplung in die simulative Trainingskomponente zur Vorschau in Entscheidungssituationen sowie als Steuerungsfunktionalität vor.

5.1. Ablauf des Verfahrens zur Ablaufplanung und -steuerung

Beim Ablauf des Verfahrens ist zu unterscheiden, dass es eine erste Phase im Vorfeld der Fertigung und eine zweite Phase während der Fertigung gibt. In Abschnitt 2.2.2 wurde dargelegt, dass ein zweckmäßiges Ablaufplanungs- und Steuerungsverfahren eine (proaktiv-) reaktive Steuerung umsetzt, um für (stochastische und dynamische)[1] Job-Shops und Flexible-Flow-Shops geeignet zu sein. So wird zunächst proaktiv ein initialer Schedule erzeugt und in einem zweiten Schritt – wenn erforderlich – reaktiv erneuert. Der initiale Schedule sei gegeben, da dieser a priori vor dem Start des Fertigungsprozesses und dem möglichen Einsatz des Ablaufplanungs- und Steuerungsverfahrens

[1] Bei statischen Ablaufplanungs- und Steuerungsproblemen sind alle Aufträge des Problems bekannt und stehen vor Beginn der Fertigung zur Bearbeitung bereit, d. h. die Auftragsmenge ist endlich. Dynamische Probleme haben die Eigenschaft, dass immer neue Aufträge, auch nach dem Start des Fertigungsprozesses, zur Verarbeitung eintreffen, d. h. die Auftragsmenge ist unendlich. Probleme können zudem deterministisch oder stochastisch sein. Bei deterministischen Problemen sind alle Informationen der Aufträge wie Bearbeitungszeiten, Arbeitsgangfolgen und Ankunftszeiten gegeben. Andererseits kann das Problem stochastisch sein. Dann sind einige Informationen unsicher oder unbekannt. Für weiterführende Literatur wird auf (Domschke et al., 1997, Seite 280 f.) verwiesen.

berechnet werden kann. Dieser dient als initiale Lösung und ist gleichzeitig geeignet, Wissen über die Erstellung dieser Lösung zu gewinnen. Es werden, wie in Tabelle 5.1 dargestellt, die zwei Phasen proaktiv-reaktiver Strategien um die Wissensakquisition und die Adaption erweitert.

Die *Phase I* umfasst die Lern- beziehungsweise Trainingskomponente, mit deren Hilfe eine Wissensbasis vor dem operativen Einsatz aufgebaut wird. Um hieraus zu lernen, werden die Informationen über eine Situation mit Steuerungsnotwendigkeit sowie ausgewähltem Auftrag festgehalten. Diese guten Entscheidungen werden in der Wissensbasis abgelegt. Auf Grundlage dieser Informationen wird mit der Lernkomponente ein Klassifizierer erzeugt.

Die *Phase II* läuft während des operativen Einsatzes, d. h. während des Fertigungsprozesses ab. Diese Phase lässt sich weiter in die beiden Teilphasen Wissensanwendung der zuvor akquirierten Informationen und Adaption der Lernkomponente untergliedern. Bei der *Wissensanwendung* (Phase IIA) wird ausschließlich mit den vor dem operativen Einsatz gewonnenen Informationen gearbeitet. Der in Phase I erzeugte Klassifizierer wird genutzt, um Konfliktsituationen des zu lösenden Shop-Problems aufzulösen. Hierbei muss nicht zwingend ein kompletter Schedule generiert werden. Ebenso kann der Klassifizierer auch nur eine einzige Entscheidung treffen, die direkt umgesetzt wird. Dieser Vorgang wiederholt sich dann für jede zu treffende Steuerungsentscheidung. Insgesamt ist es dadurch möglich, nicht nur sehr schnell, sondern auch gute Lösungen zur Ablaufplanung und -steuerung für ein Shop-Problem zu finden. In der *Adaptionsphase* (Phase IIB) wird der Klassifizierer während des Ablaufes permanent angepasst, z. B. Steuerungsklassen oder neue Trainingsbeispiele hinzugefügt oder entfernt. So ist der Klassifizierer an ein spezielles Fertigungssystem anpassbar. Informationen über den bisherigen Lösungsprozess und Charakteristika des zu lösenden Problems werden aufgezeichnet und verarbeitet. Dadurch ist es möglich, das System kontinuierlich auf das zu lösende Problem abzustimmen und die Qualität der erzeugten Lösung zu erhöhen. Die beiden Teilphasen Wissensanwendung und Adaption können parallel ablaufen. Das Verfahren ist ohne die Adaptionsmechanismen funktionsfähig.

Bei der Betrachtung der Schnittstellen sind die Trainings- und Klassifikationsschnittstelle von besonderer Relevanz, da hier zwischen den beteiligten Komponenten die entscheidungsrelevanten Informationen transferiert werden. Die *Trainingsschnittstelle* für den Aufbau der Trainingsdaten ist zwischen der Trainingskomponente und der Lernkomponente angesiedelt. Ausgehend von dem in der Lernkomponente eingesetzten Klassifizierer, ist der Aufbau der Trainingsbeispiele festzulegen. Die *Klassifikationsschnittstelle* dient dem Aufruf der Lern- aus der Steuerungskomponente. In Situationen mit Kon-

Phase I Im Vorfeld der Fertigung	Phase II Während der Fertigung	
	Phase IIa Wissensanwendung	Phase IIb Adaption
Erstellen des initialen Schedules	Falls erforderlich, Erstellen eines neuen Schedules	Generieren von neuen bzw. zusätzlichen Trainigsbeispielen
Lernen aus den initialen Schedulinglösungen	Generieren von situationserweiternden Informationen	

Tabelle 5.1.: Phasen des entwickelten Verfahrens

flikten innerhalb der Steuerungskomponente wählt die Lernkomponente, wie beschrieben, die situativ passendste(n) Regel(n) aus. Wie beschrieben soll die wissensbasierte Entscheidungskomponente zur Auflösung der Steuerungssituation auch ohne die Steuerungskomponente einsetzbar sein. Dabei sind Merkmalausprägungen zu beachten, d. h. spezifische, nur in der Steuerungskomponente vorhandene Merkmale müssen herausgefiltert werden.

5.2. Entwicklung der Steuerungskomponente

Als Heuristik für die Steuerungskomponente bietet sich das von *Giffler und Thompson* entwickelte Verfahren an, da dieses die in Abschnitt 2.2.2 erarbeiteten Anforderungen an eine Steuerungskomponente am Besten erfüllt.[2]

5.2.1. Notwendige Vorüberlegungen

Das Verfahren nach *Giffler und Thompson* besitzt eine lineare Laufzeit und kann mit $M \cdot N$ als der Anzahl der einzuplanenden Operationen, in $\mathcal{O}(M \cdot N)$ einen aktiven Schedule erzeugen. Wie beschrieben ist hierbei die Beschränkung der Lösungssuche auf aktive Ablaufpläne bei regulären Zielfunktionen ausreichend, da die optimale Lösung ein aktiver Ablaufplan ist.[3] Mit der Heuristik sind alle aktiven Schedules einer Scheduling-Probleminstanz erzeugbar. Stünden der Heuristik hinreichend viele Auswahlregeln zur

[2] Siehe Abschnitt 3.1.2.
[3] Siehe Abschnitt 2.1.2.

Anwendung bereit, wären durch die unterschiedliche Anwendung der Regeln alle aktiven Lösungen für das Scheduling-Problem generierbar. Demzufolge muss eine (oder mehrere) Reihenfolge(n) der Regelanwendung existieren, die zu der gleichen Lösung führt (führen), welche von jedem anderen, aktive Lösungen erzeugenden Scheduling-Verfahren erstellt werden. Das Verfahren eignet sich für alle regulären Zielfunktionen.[4] Da das hier konzipierte Ablaufplanungs- und Steuerungsverfahren auf der Giffler-Thompson-Heuristik aufbaut und wissensbasiert ist, soll das Verfahren im Folgenden wissensbasierter Giffler-Thompson (WBSGT) heißen.

An der Giffler-Thompson-Heuristik sind Korrekturen und Erweiterungen zur Konfliktauflösung vorzunehmen.[5] Von *Zhao und Deng* wurde beispielsweise eine Funktion entwickelt, die für jeden Schedule eines Job-Shops einen ähnlichen aktiven Schedule konstruieren kann.[6] Damit werden die Lösungen für ein Job-Shop-Problem in den Bereich der aktiven Schedules verschoben. Angewandt auf lokale Suchmethoden wie Genetische Algorithmen oder Simulated Annealing, hat sich eine deutliche Verbesserung der Algorithmen gezeigt. Allerdings steht dem praktischen Einsatz dieser Funktion in der jetzigen Form der noch deutlich zu hohe Zeitaufwand für das Verschieben einer Lösung in den Bereich der aktiven Ablaufpläne entgegen. Jedoch unterstützt die Untersuchung die theoretische Überlegung, dass ein hoher Performancegewinn im Sinne der besseren Erfüllung der Zielfunktion durch die Einschränkung auf aktive Schedules erreichbar ist.

Mit der Verwendung der Konstruktionsidee von *Giffler und Thompson* und der damit verbundenen Einschränkung auf die Konstruktion von aktiven Ablaufplänen bei der Suche nach einer guten Lösung für ein Shop-Problem ergeben sich folgende weitere Vorteile:

- Der zu durchsuchende Raum wird deutlich reduziert, enthält aber weiterhin alle optimalen Lösungen.[7]
- Die durchschnittliche Lösungsgüte innerhalb der Menge aktiver Schedules ist höher als die durchschnittliche Lösungsgüte aller zulässigen Lösungen. Demzufolge wird bereits in einem Unterraum von besseren Lösungen gesucht.[8]

[4] Aus Vergleichbarkeitsgründen mit anderen Verfahren wird die Evaluierung in Kapitel 6 über die Minimierung des Makespans durchgeführt (siehe Abschnitt 2.1.2).
[5] Für die Anwendung in Job-Shops erforderliche Korrekturen werden in Abschnitt 5.2.2 entwickelt, die Erweiterungen für Flexible-Flow-Shops in Abschnitt 5.2.3.
[6] Vgl. (Zhao und Deng, 2006).
[7] Siehe Abschnitt 2.1.2.
[8] Vgl. (Zhao und Deng, 2006).

Bei der dargestellten Konstruktionsmethode wird der zu betrachtende Zustandsraum also bereits durch die aktiven Schedules eingeschränkt.

Die Menge aller Teilprobleminstanzen einer Shop-Probleminstanz $SPI = (P, T)$ mit den Bearbeitungszeiten P und den technologischen Reihenfolgen T, die durch die Einplanung der Operationen $o_{j,m} \in SPI$ entstehen, definieren den Zustandsraum. Die Verbindung der Elemente des Zustandsraumes mit Kanten, die mit den eingeplanten Operationen beschriftet sind, beschreibt einen Zustandsraumgraphen.

Beispiel 5.1 Ersparnis bei der Einschränkung auf aktive Ablaufpläne
Sei ein Scheduling-Problem bestehend aus drei Jobs ($N = 3$) und zwei Maschinen ($M = 2$) gegeben. Die Bearbeitungszeiten der einzelnen Operationen und die Reihenfolge mit der die einzelnen Jobs die Maschinen durchlaufen müssen, sind in Tabelle 5.2 gegeben. Die *or*-Werte bilden die organisatorische Reihenfolge eines Jobs ab, die *pt*-Werte die Bearbeitungszeiten auf den Maschinen.

	Maschine 1		Maschine 2	
	or	*pt*	*or*	*pt*
Job 1	I	2	II	4
Job 2	II	4	I	3
Job 3	I	1	II	3

Tabelle 5.2.: Beispielhaftes Job-Shop-Problem

Die erzeugbaren Ablaufpläne sind als Baum darstellbar.[9] Jede Auswahl aus der Konfliktmenge stellt einen inneren Knoten des Baumes dar, mit einer Anzahl von Ästen, die der Anzahl der Operationen in der Konfliktmenge entspricht. Der Baum ist in Abbildung 5.1 illustriert. Im Baum sind nur die Konflikte mit einer Mindestgröße von zwei abgetragen. Eingeplante Operationen ohne Konflikt werden nicht dargestellt.

Die Wurzel des Baums steht für den leeren Startzustand, in dem noch keine Operation eingeplant ist. Die Beschriftungen an den Ästen geben den zugehörigen Job der gewählten Operationen an. Jeder Ast aus einem Knoten entspricht einer möglichen Auflösung einer Konfliktmenge. In einem Blatt sind alle Konfliktmengen aufgelöst und alle Operationen eingeplant. Damit entspricht ein Blatt einem vollständigen und aktiven Schedule. Dem Verfahren entsprechend, ist jeder der acht möglichen Schedules $S_1, \ldots, S_8$ ein aktiver; die Schedules $S_1, \ldots, S_8$ bilden die Menge aller zu diesem Problem existierenden aktiven Ablaufpläne. Insgesamt gibt es $3!^2 = 36$ Möglichkeiten, um die drei Aufträge auf

[9] Siehe Abschnitt 5.2.2 und 5.2.3 zur Generierung des jeweiligen Baums für Job-Shops und Flexible-Flow-Shops.

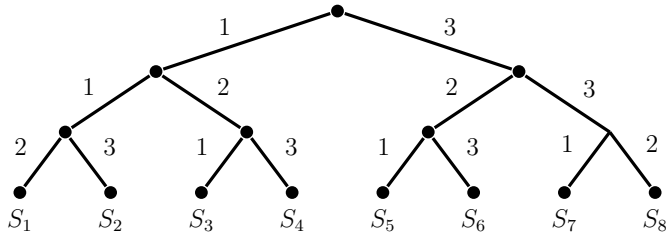


Abbildung 5.1.: Baum der mit dem Steuerungsalgorithmus generierbaren Ablaufpläne

den zwei Maschinen anzuordnen.[10] Die Einschränkung auf die acht aktiven Schedules $S_1, \ldots, S_8$ reduziert in diesem Beispiel die Menge der zu durchsuchenden Ablaufpläne somit auf weniger als 25 Prozent der ursprünglichen Alternativen. □

5.2.2. Steuerungskomponente für Job-Shops

In der in Abschnitt 3.1.2 beschriebenen Giffler-Thompson-Heuristik kann es zur gleichzeitigen Belegung einer Maschine von mehr als einer Operation kommen, insbesondere wenn die Bearbeitungszeiten der Operationen sehr inhomogen sind.[11] Um dieses Problem zu verhindern, wurde die Menge O eingeführt. Diese enthält die Informationen zu sämtlichen noch einplanbaren Operationen. Die Menge umfasst die aktuell frühesten, maschinenbedingten Einplanungszeiten. Die Heuristik wurde um die Funktionen zur Verwaltung der Menge O erweitert. Algorithmus 5.1 zeigt die hier entwickelte und korrigierte Ablaufplanungs- und Steuerungsheuristik für Job-Shops. Diese baut auf dem von *Zäpfel und Braune* vorgestellten und in Algorithmus 3.1 dargelegten Verfahren auf.

Mit dem hier entwickelten Lösungsansatz werden weiterhin die Vorteile eines verbesserten Lösungsraums ausgenutzt, indem die Menge der betrachteten Lösungen von Anfang an auf aktive Ablaufpläne eingegrenzt wird. Beim Start des Algorithmus 5.1 werden die Start- und die Fertigstellungszeiten aller Operationen auf null gesetzt. Diese Zeiten werden durch die Anwendung des Algorithmus derart angepasst, dass nach dem Verfahrensdurchlauf ein korrekter aktiver Schedule beschrieben wird. Die Hauptschleife des Algorithmus wird solange ausgeführt, bis sämtliche noch einzuplanenden Operationen in den Schedule aufgenommen sind. Hierzu wird zunächst eine Konfliktmenge CS aufgebaut. In dieser sind diejenigen Jobs enthalten, deren nächste Operation so in

[10] Siehe Abschnitt 2.1.3.
[11] Siehe Abschnitt 3.1.2 und Anhang A.4.

Algorithmus 5.1 : Steuerungsalgorithmus für Job-Shops

```
1  O := {o| ∀ Operationen o ∀ Jobs N};
2  SO := {o| erste einplanbare Operation o eines Jobs j};
3  r(o) := 0, ∀o ∈ SO;
4  while SO ≠ ∅ do
       // Berechne die Operation mit der kleinsten Fertigstellungszeit
5      f(o) := r(o) + p(o), ∀o ∈ SO;
6      f_min := min{f(o)|o ∈ SO};
       // Bilde die Konfliktmenge
7      CS := {o|o ∈ SO ∧ M(o) = M(o_min) ∧ r(o) < f_min};
       // Wähle mittels einer Prioritätsregel Operation o' ∈ CS aus
       // Plane die ausgewählte Operation ein
8      if N(o') ≠ ∅ then
9          r(o) := f(o'), ∀o ∈ N(o') ∧ r(o) < f(o');
10         SO := SO ∪ N(o') \ {o'};
11     end
       // Passe die nicht gewählten Operationen an
12     O := O \ {o'};
13     r(o) := f(o'), ∀o ∈ O ∧ r(o) < f(o') ∧ M(o) = M(o');
14 end
```

den Schedule integriert werden kann, dass sich daraus ein aktiver Schedule ergibt. Zur Bildung der Konfliktmenge wird der Job mit der kleinsten Fertigstellungszeit für die als nächstes einzuplanende Operation gesucht. Die minimale Fertigstellungszeit f_{min} ergibt sich durch Berechnung der minimalen Fertigstellungszeit für jeden Job o_{min}, die durch Hinzufügen der nächsten Operation des Jobs zum Schedule erreicht wird. Dieser Job, dessen nächste Operation den minimalen Fertigstellungszeitpunkt f_{min} besitzt, wird zum Aufbau der Konfliktmenge genutzt. In die Konfliktmenge werden zusätzlich alle Aufträge mit aufgenommen, deren nächste zu bearbeitende Operation mit der Operation des zum Konfliktmengenaufbau verwendeten Jobs in Konflikt steht.[12] Diese Konfliktmenge enthält jetzt alle Jobs, deren nächste Operation so in den Schedule eingeplant werden kann, dass ein aktiver Schedule entsteht. Aus dieser Menge wird dann eine Operation o' ausgewählt.

Nach der Auswahl sind die Startzeiten $r(o)$ aller Operationen anzupassen, dabei ist die Einplanung der gewählten Operation zu berücksichtigen. Die Nachfolgeoperation des gewählten Jobs kann nicht vor Beendigung der Vorgängeroperation starten. Daher wird ihre Startzeit, sofern diese einen geringeren Wert hatte, auf den Fertigstellungzeitpunkt

[12] Hierbei steht eine bearbeitungsbereite Operation B mit einer Operation A in Konflikt, wenn durch die Einplanung der Operation A in den Schedule die Ausführung von Operation B verzögert wird. Dies ist immer genau dann der Fall, wenn der Fertigstellungszeitpunkt der Operation A nach dem frühestmöglichen Startzeitpunkt der Operation B liegt (siehe Anhang A.4).

O	Menge aller noch einplanbaren Operationen
SO	Menge aller aktuell einplanbaren Operationen
N	Menge aller zu bearbeitenden Jobs
M	Menge aller Maschinen
CS	Konfliktmenge von Operationen auf einer Maschine M
$N(o)$	Menge der Operationen o in einem Job j direkt nachfolgenender Operationen
$M(o)$	Maschine die Operation o bearbeitet
$r(o)$	Frühestmöglicher Startzeitpunkt einer Operation o
$p(o)$	Bearbeitungszeit einer Operation o
$f(o)$	Fertigstellungzeit einer Operation o
f_{min}	Minimale Fertigstellungzeit
o_{min}	Minimaler Fertigstellungszeit von Operation o
o'	Aus der Konfliktmenge CS ausgewählte Operation

Tabelle 5.3.: Legende zum Steuerungsalgorithmus für Job-Shops

$f(o')$ der eingeplanten Operation angehoben. Gleichfalls kann die von der ausgewählten Operation belegte Maschine in dieser Zeit keine anderen Operationen bearbeiten. Es sind die Startzeitpunkte aller Operationen, die noch auf dieser Maschine zu bearbeiten sind, auf jenen Wert zu setzen, der dem Fertigstellungszeitpunkt der gewählten Operation entspricht. Besitzen diese Operationen durch andere Einschränkungen wie beispielsweise Fertigstellungszeiten ihrer Vorgängeroperationen bereits spätere Startzeitpunkte, dürfen diese nicht verändert werden, damit die bisherigen Bedingungen nicht unterlaufen werden.

Die Steuerungskomponente unterliegt der simplen Konstruktionsvorschrift: So lange nicht-eingeplante Operationen vorhanden sind und kein Konflikt entsteht, plane die Operation mit frühestem Fertigstellungszeitpunkt ein. Anders formuliert: Ein Konflikt ist eine Menge einplanbarer Alternativen. Die Steuerungskomponente kann den Zustandsraumgraphen für eine Probleminstanz als Baum abbilden, in dem die Wurzel ein leerer Schedule ist, die Blätter aktive, vollständige und die inneren Knoten partielle Schedules sind.[13]

Abbildung 5.2 zeigt einen beispielhaften Zustandsraumgraphen eines Job-Shops. Die Verzweigungen in diesem Baum entsprechen einem Konflikt. Einen Konflikt mit n alternativen Operationen bilden n ausgehende Kanten ab. Da die Operationen eines Konflikts auf derselben Maschine m ausgeführt werden, haben die ausgehenden Kanten die Beschriftungen $o_{j_0,m}$ bis $o_{j_n,m}$. Die Steuerungskomponente baut den Zustandsraumgraphen iterativ auf. Jede Iteration des Verfahrens nach Algorithmus 5.1 bildet durch die Einplanung einer Operation $o_{j,m}$ eine Kante, die zu einem partiellen bzw. vollständigen Schedule führt. Der partielle Schedule des direkten Vorgängerknotens wird um die

[13] Sämtliche möglichen Lösungen, die durch Auswahl aller Operationen aus der Konfliktmenge entstehen, sind im Zustandsraumgraph visualisierbar. Die Lösungen sind die aktiven Schedules, die zu dieser Job-Shop-Probleminstanz existieren (vgl. (Giffler und Thompson, 1960, Seite 495) und siehe Beispiel 5.1).

Operation $o_{j,m}$ ergänzt. Die Blätter repräsentieren alle existierenden aktiven Lösungen für eine Scheduling-Probleminstanz.

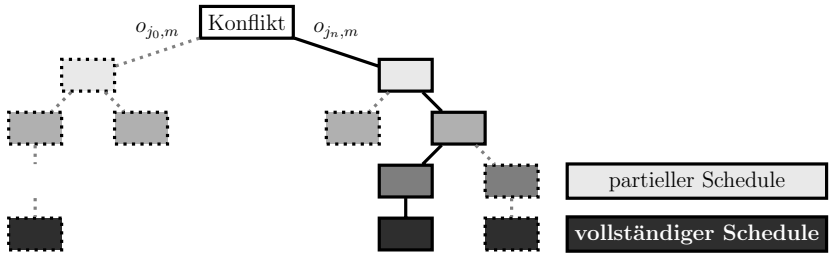


Abbildung 5.2.: Zustandsraumgraph durch Anwendung der Steuerungskomponente

Abbildung 5.3 zeigt ein Beispiel für zwei Knoten $v_S, v_{S'}$ des Baumes, welche die Schedules S und S' repräsentieren.

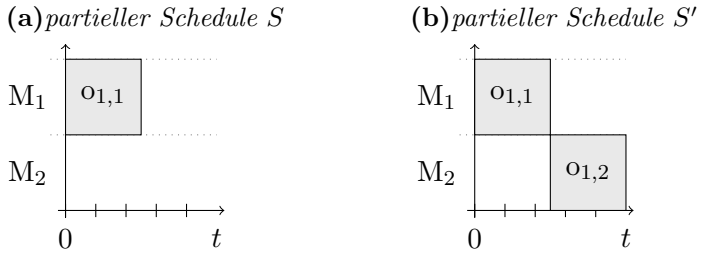


Abbildung 5.3.: Inklusion von zwei Schedules

Im Zustandsraumgraphen ist v_S der direkte Vorgänger von $v_{S'}$. Diese Eigenschaft bedingt, dass alle Operationen in S mit den gleichen Startzeiten in S' enthalten sind. Der erzeugte Baum eines Job-Shops hat die Tiefe $N \cdot M$, mit N Aufträgen und M Maschinen, weil $N \cdot M$ Operationen eingeplant werden. Er enthält nur gültige Schedules, da seine Konstruktion die technologischen Reihenfolgen der Aufträge berücksichtigt. Die maximalen Knotengrade sind abhängig von der Tiefe des Baumes. In Job-Shops gibt es zwei Schranken für den maximalen Grad, wobei die jeweils niedrigere Schranke gilt: die Anzahl verbleibender Operationen und die Anzahl der Aufträge. Es können maximal N Aufträge um eine Maschine konkurrieren; damit kann ein Knoten maximal N verschiedene Nachfolger haben. Des Weiteren kann die Anzahl der Nachfolger die Anzahl der noch nicht eingeplanten Operationen nicht übersteigen. In Tiefe $(N \cdot M) - 2$ kann es z. B. höchstens zwei Nachfolger geben und in Tiefe $(N \cdot M) - 1$ genau einen. Die maximalen Knotengerade in Flexible-Flow-Shops werden in Abschnitt 5.2.3 vorgestellt.

Bisher ist keine Aussage darüber getroffen worden, welche Operation aus der Konfliktmenge zu wählen ist. Um einen möglichst guten Pfad durch diesen Baum (Lösungsraum)

zu finden, werden die Entscheidungen für eine Operation in den Knoten von einem maschinellen Lernverfahren, d. h. der Lernkomponente getroffen werden. Dieses Verfahren wird in Abschnitt 5.3 entwickelt. Zuvor ist im nachfolgenden Abschnitt die bisherige Steuerungskomponente für den Einsatz in Flexible-Flow-Shop-Umgebungen zu erweitern, um damit zu den Aufträgen zusätzlich noch Maschinen bei der Entscheidung zu berücksichtigen.

5.2.3. Steuerungskomponente für Flexible-Flow-Shops

Es ist das Wesen der Ablaufplanung und -steuerung, eine Menge von gleichzeitig fertigungsbereiten Aufträgen den zur Verfügung stehenden Maschinen zuzuteilen und dabei eine zeitliche Abfolge festzulegen. Im Fall der bisher betrachteten Job-Shops ist die Art aller hierzu notwendigen Entscheidungen gleich. Aus einer Auftragsmenge ist zu einem bestimmten Zeitpunkt genau ein Auftrag zur Bearbeitung auf einer bestimmten, frei werdenden Maschine auszuwählen. Es muss aus mehreren Aufträgen, nicht aber aus mehreren Maschinen gewählt werden. Im Fall von Flexible-Flow-Shops bekommt genau diese Entscheidung Relevanz. Da auf den einzelnen Stufen der Fertigungsumgebung mehrere parallele Maschinen vorhanden sind, ist zusätzlich zum Auftrag auch noch eine Maschine zu wählen. Aus diesem Grund kommt der Entscheidung für eine der parallelen Maschinen pro Stufe in dem hier betrachteten Flexible-Flow-Shops zusätzliche Bedeutung zu, insbesondere dann, wenn es sich um unterschiedlich schnelle – uniforme oder unverwandte – Maschinen handelt. Zur Ablaufplanung und -steuerung im Fall der hier betrachteten Flexible-Flow-Shops sind also immer wieder zwei verschiedenartige Entscheidungen zu treffen. Dazu ist zunächst die Frage zu beantworten, in welcher Reihenfolge dieses geschehen soll:

- Maschinenentscheidung → Auftragsentscheidung
- Auftragsentscheidung → Maschinenentscheidung

Hierbei ist zwischen Entscheidungen bei uniformen oder unverwandten Maschinen zu differenzieren.

Im Falle von *uniformen* Maschinen ist die Reihenfolge der Entscheidungen irrelevant, weil sich die Geschwindigkeit der Maschinen einer Fertigungsstufe nicht erst nach Auswahl eines Auftrags ergibt. Die Uniformität der Maschinen bedeutet, dass diese zwar im Vergleich untereinander unterschiedlich schnell sind, die Geschwindigkeit aber nicht zusätzlich von dem auf diesen bearbeitetem Auftrag determiniert wird. Auch wenn unberücksichtigt bleibt, dass die Aufträge – deren Operationen – unterschiedlich lange

Durchlaufzeiten aufweisen, ergibt sich aus der Uniformität der Arbeitssysteme, dass Entscheidungen, die auf der Geschwindigkeit der Maschinen beruhen, unabhängig von der Auswahl eines Auftrags gleich lauten. Im Ergebnis kann bei uniformen Maschinen die Entscheidung für eine Maschine also beliebig vor oder auch nach der Auswahl eines Auftrags erfolgen. In dem hier entwickelten Verfahren soll die Wahl eines Auftrags der Wahl einer Maschine vorangehen. Als ein Grund hierfür ist anzuführen, dass die Auswahl aus einer potenziell großen Menge von Aufträgen oft länger dauert als die Auswahl aus einer in der Regel kleineren Menge an Maschinen. Im Fall, dass sich die Fertigungssituation während des ersten Entscheidungsprozesses leicht verändert, wäre der zweite, schnellere Entscheidungsprozess in der Lage, diese Änderung noch zu berücksichtigen.[14]

Bei *unverwandten* Maschinen ist die Reihenfolge der Entscheidungen (Auftrag und Maschine) nicht einfach trennbar. Hier bedingt die Wahl eines Auftrags die Geschwindigkeit der Maschine und umgekehrt. Die Bearbeitungszeiten werden durch die Variable ps_j^{st} als Standard-Bearbeitungszeit des Auftrags j auf Stufe st abgebildet. Die auftragsabhängige Maschinenleistung der Maschine m auf Stufe st bei Auftrag j gibt $v_{j,m}^{st}$ an.

Beispiel 5.2 Auftragsauswahl bei unverwandten Maschinen
Seien die beiden Aufträge 1 und 2 mit den Standard-Bearbeitungszeiten $ps_1^{st} = 4$ und $ps_2^{st} = 16$ sowie die beiden Maschinen m_1 und m_2 gegeben. Weiterhin seien die auftragsabhängigen Maschinenleistungen mit $v_{1,1}^{st} = 0,2$; $v_{1,2}^{st} = 1,0$; $v_{2,1}^{st} = 1,0$ und $v_{2,2}^{st} = 0,8$ gegeben. Die Bearbeitungszeit ergibt sich jetzt mit $ps_j^{st}/v_{j,m}^{st}$. Werden zunächst die Aufträge ausgewählt, dann ist für Auftrag 1 Maschine 2 die schnellere und für Auftrag 2 umgekehrt Maschine 1. Findet hingegen erst die Auswahl der Maschine statt, dann ist bei Maschine 1 der zweite Auftrag und bei Maschine 2 der erste Auftrag der schnellere. Hierdurch wird deutlich, dass Auswahl von Maschine und Auftrag sich gegenseitig bedingen – die Entscheidungen sind abhängig voneinander. □

Infolgedessen ist das in Abschnitt 5.2.2 entwickelte Verfahren für den Einsatz in Flexible-Flow-Shops zu erweitern.[15] Einen Ansatz geben *Giffler und Thompson* für den Fall paralleler Arbeitssysteme mit gleichen Fähigkeiten selbst.[16] Dieser ist nur für identische, nicht für die hier betrachteten uniformen und unverwandten Maschinen auf den einzelnen Stufen zutreffend. Ein weiterer Vorschlag zur Erweiterung für Systeme mit unifor-

[14] Der umgekehrte Entscheidungsprozess würde ebenso funktionieren. Jedoch sind hier gegebenenfalls erneute Entscheidungen erforderlich. Fällt bei der Auftragsauswahl, nachdem die vorherige Entscheidung für eine Maschine abgeschlossen ist, diese Maschine aus, muss der Prozess neu gestartet werden und eine neue Maschine ist zu wählen.

[15] Vorteilhafte Eigenschaften der Steuerungskomponente für Job-Shops wie das ausschließliche Generieren von aktiven Schedules bleiben erhalten.

[16] Vgl. (Giffler und Thompson, 1960) und siehe Abschnitt 3.1.2.

men Arbeitssystemen liefern *Nascimento*[17] sowie *Chang und Sullivan*[18], deren Verfahren in Abschnitt 3.1.2 vorgestellt wurde. *Nascimento* identifiziert das Verfahren von *Giffler und Thompson* aufgrund von durchgeführten Experimenten als gut geeignet für flexible Fertigungssysteme.[19] Demzufolge liefert das Verfahren von *Giffler und Thompson* in vergleichsweise kurzer Zeit gute Ergebnisse, wenn es in einer von *Chang und Sullivan* vorgestellten Abwandlung zur Ermittlung aller aktiven Schedules in Umgebungen mit mehreren flexiblen Fertigungssystemen eingesetzt wird. Die von *Chang und Sullivan* vorgeschlagenen Modifikationen bestehen hauptsächlich in der Anweisung, nach der Auftragswahl zusätzlich die bearbeitende Maschine auszuwählen. Die vorgeschlagenen Adaptionen beziehen sich allerdings auf flexible Fertigungssysteme und nicht auf die hier betrachteten Flexible-Flow-Shops.

Die Überlegungen von *Chang und Sullivan* sind auf Flexible-Flow-Shops übertragbar, da Umgebungen aus flexiblen Fertigungssystemen, welche teilweise die gleichen Arbeiten verrichten können, im Vergleich zu Flexible-Flow-Shops aber eine Verallgemeinerung darstellen. Umgekehrt gilt für Flexible-Flow-Shops eine Einschränkung. Verglichen mit Umgebungen, die aus mehreren flexiblen Fertigungssystemen bestehen, kann in Flexible-Flow-Shops auf einer Stufe eingesetzten Maschinen nur ein Auftragstyp bearbeitet werden. Hieraus folgt, dass das Verfahren nach *Giffler und Thompson* in der Abwandlung von *Chang und Sullivan* eine geeignete Grundlage für die Ablaufsteuerung in Flexible-Flow-Shops darstellt. Damit die in Abschnitt 5.2.2 konzipierten Heuristik in Flexible-Flow-Shop-Umgebungen eingesetzt werden kann, sind folgende Erweiterungen durchzuführen:

- Methoden zur Behandlung von mehreren Maschinen sowie zur Berechnung der kleinsten Fertigstellungszeiten bei uniformen und unverwandten Maschinen sind zu entwickeln, um hiermit die Konfliktmenge mit konkurrierenden Operationen aufzubauen.

- In der Konfliktmenge mit den als nächstes zu bearbeitenden Operationen werden nur Operationen betrachtet, die aktuell zur Bearbeitung anstehen oder unmittelbar vor der betrachteten Fertigungsstufe bearbeitet werden. Bei letzteren muss die Bereitstellungszeit, d. h. der neue Startzeitpunkt bekannt sein. Dies verringert sowohl den Berechnungsaufwand als auch den Einfluss von Unsicherheiten. Würden

[17] Vgl. (Nascimento, 1993).
[18] Vgl. (Chang und Sullivan, 1990).
[19] Ein flexibles Fertigungssystem umfasst eine Reihe von Fertigungseinrichtungen, die über ein gemeinsames Steuerungs- und Fördersystem so miteinander verknüpft sind, dass die Fertigung von fertigungsähnlichen Objekten (Produktarten und Werkstückarten) mit relativ kleinen Losgrößen stattfinden kann (vgl. Tempelmeier, 1998, Seite 502).

Stufen, die technologisch weiter zurück liegen ebenfalls mit betrachtet, könnten die Startzeiten nur noch mit großen Unsicherheiten berechnet werden. Es steht noch nicht fest, von welcher Maschine die Operationen auf der zurückliegenden Stufe bearbeitet werden – damit ist die Bearbeitungszeiten extrem unsicher.

- Wie beschrieben, ist nach der Auswahl der auf der betrachteten Fertigungsstufe als nächste zu bearbeitenden Operation in der Regel eine der auf dieser Stufe zur Verfügung stehenden Maschinen, zu wählen. Dieses entspricht den von *Chang und Sullivan* getroffenen Annahmen.[20] Jedoch sind bei der Entscheidung für Operation und Maschine alternative Lösungsmöglichkeiten zu berücksichtigen: Ein System mit eben diesen zwei aufeinander folgenden Entscheidungen (Zwei-Klassifizierer-System) oder ein System, das die Auswahl für Maschine und Operation in einer gemeinsamen Entscheidung vereint (Ein-Klassifizierer-System).

Die Auswahl von Operation und Maschine werden mit der in Abschnitt 5.3 entwickelten Lernkomponente für Flexible-Flow-Shops getroffen. Das Zwei-Klassifizierer-System auf Basis von zwei Entscheidungen erfordert einen zweiten Klassifizierer. Bei diesem System wird mit dem ersten Klassifizierer die zu bearbeitende Operation aus der Konfliktmenge gewählt. Ausgehend von dieser Entscheidung ist mittels des zweiten Klassifizierers eine Maschine auszuwählen.[21] Beim Ein-Klassifizierer-System wird eine Entscheidung für Operation und Maschine getroffen. Hierzu ist die Bildung der Konfliktmenge anzupassen. Beim Aufbau der Konfliktmenge sind alle Maschinen zu berücksichtigen. Das Verfahren zur Ablaufsteuerung in Flexible-Flow-Shops wird in Algorithmus 5.2 dargestellt.[22] Tabelle 5.4 zeigt die verwendenden Mengen und Variablen. Diese orientieren sich an den bereits aus Algorithmus 5.1 bekannten Notationen und erweitern diese um die, für Flexible-Flow-Shops erforderlichen.

[20] Vgl. (Chang und Sullivan, 1990).
[21] Dabei werden nur die zum Entscheidungszeitpunkt zur Bearbeitung bereiten Maschinen berücksichtigt.
[22] Hier für das Ein-Klassifizierer-System.

Algorithmus 5.2 : Steuerungsalgorithmus für Flexible-Flow-Shops

```
   // Initialisieren der Auftragsmenge
1  SO = {j|j ∈ {1,2,...,n} ∧ sf_j = st};
   // Startzeiten der Aufträge setzen
2  jst_j = r_j, ∀j ∈ SO;
   // Startzeiten der Maschinen setzen
3  mst_m = mr^st_m, ∀m = {1,2,...,M_st};
   // Lösen des Problems für die aktuelle Stufe
4  while SO ≠ ∅ do
       // Bestimme die Operation mit der kleinsten Fertigstellungszeit
5      f_min = ∞; m_temp = 0; j_temp = 0;
       // Die Maschinen der aktuellen Stufe untersuchen
6      foreach j ∈ SO do
7          for m = 1; m ≤ M_st; m++ do
8              if mst_m > jst_j then
9                  f_j = mst_m + ps^st_j / v^st_{j,m};
10             else
11                 f_j = jst_j + ps^st_j / v^st_{j,m};
12             end
13             if f_j < f_min then
14                 f_min = f_j; m_temp = m; j_temp = j;
15             end
16         end
17     end
       // Bilden der Konfliktmenge
18     CS := {(j,m)|j ∈ SO ∧ jst_j < f_min ∧ mst_m < f_min};
       // Auflösen der Konfliktmenge
19     if alle (j,m) in CS mit gleichem j then
           // Wähle Operation o^st_{j_temp,m_temp}
20         m = m_temp; j = j_temp;
21     else
           // Wähle mit Prioritätsregeln aus der Konfliktmenge
22     end
       // Einplanen der Operation
23     if mst_m > jst_j then
24         mst_m = mst_m + ps^st_j / v^st_{j,m};
25     else
26         mst_m = jst_j + ps^st_j / v^st_{j,m};
27     end
       // Startzeit anpassen
28     jst_j = mst_m;
       // Operation entfernen
29     SO = SO\{j};
30 end
```

FS	Menge aller Fertigungsstufen
SO	Menge aller auf der aktuellen Stufe einplanbaren Aufträge
CS	Konfliktmenge der Operationen
st	Index der aktuell beplanten Stufe
j	Index der Aufträge einer Stufe st
m	Index der Maschinen einer Stufe st
f_j	Fertigstellungzeit des Auftrags j
M_{st}	Anzahl Maschinen auf Stufe st
r_j	Freigabetermin von Auftrag j
mr_m^{st}	Freigabetermin von Maschine m auf Stufe st
sf_j	Stufenbasierter Fortschritt von Auftrag j
f_{min}	Minimale Fertigstellungzeit
ps_j^{st}	Standard-Bearbeitungszeit des Auftrags j auf Stufe st
mst_m	Früheste Freigabe von Maschine m
jst_j	Frühester Starttermin von Auftrag j
$v_{j,m}^{st}$	Relative Geschwindigkeit der Maschine m auf Stufe st bei Auftrag j

Tabelle 5.4.: Legende zum Steuerungsalgorithmus für Flexible-Flow-Shops

Algorithmus 5.2 wird sukzessive von Fertigungsstufe 1 bis zu Stufe FS eingesetzt, um einen Ablaufplan für Flexible-Flow-Shops zu generieren. Auf diese Art und Weise wird ein Problem mit FS Stufen in FS Teilprobleme unterteilt.[23] Bei der Generierung eines vollständigen Plans für einen Flexible-Flow-Shop wird das Teilproblem auf den Stufen $st+1$ bis FS noch nicht berücksichtigt, wenn das Teilproblem auf der Stufe st noch nicht gelöst ist.[24] Innerhalb einer Stufe berechnet der Algorithmus iterativ einen minimalen Fertigstellungstermin f_{min} und baut daraus eine Konfliktmenge CS von Operationen auf. Die Konfliktmenge in Flexible-Flow-Shops mit unverwandten ($FRk||C_{max}$) oder uniformen Maschinen ($FQk||C_{max}$) wird anders aufgebaut als die Konfliktmenge in Job-Shops. Ausganspunkt in beiden Fällen ist die Operation mit minimalem Fertigstellungszeitpunkt f_{min}, die aus der Menge noch nicht eingeplanter Operationen der aktuellen Stufe ausgewählt wird. Bei uniformen Maschinen wird hierzu die Standard-Bearbeitungszeit des Auftrags verwendet. Alle Operationen, die auf der aktuellen Stufe bereit sind oder die sich auf der vorgelagerten Stufen bereits in Bearbeitung befinden und deren frühester Startzeitpunkt auf der aktuellen Stufe oder Fertigstellungszeitpunkt auf der vorgelagerten Stufe $st-1$ vor f_{min} liegt, werden in die Konfliktmenge aufgenommen. Für unverwandte Maschinen wird die minimale Fertigstellungszeit aus der Standard-Bearbeitungszeit der Auftäge und relativer Maschinengeschwindigkeit berechnet. Auch hier werden sämtliche Operationen, die auf der aktuellen Stufe bereit sind oder die sich auf der vorgelagerten Stufen bereits in Bearbeitung befinden und deren

[23] Dabei werden die Teilprobleme auf den einzelnen Stufen aufgrund der global trainierten Lernkomponente nicht lokal, sondern mit globalen Zielen gelöst.

[24] Grundsätzlich ist das Treffen von einzelnen Entscheidungen in Steuerungssituationen auf den verschiedenen Stufen gewünscht. Die beschriebene ganzheitliche Vorgehensweise im Sinne der Generierung eines Plans für den gesamten Flexible-Flow-Shop wird aus Gründen der Evaluierung durchgeführt. Einzelne Entscheidungen werden in Flexible-Flow-Shops mit nur einer Fertigungsstufe evaluiert. Zur Evaluierung in Flexible-Flow-Shops siehe Abschnitt 6.3.

frühester Startzeitpunkt sowie der Freigabezeitpunkt der entsprechenden Maschine vor f_{min} liegen, in die Konfliktmenge aufgenommen. Hierbei können Aufträge mehrmals in der Konfliktmenge vertreten sein, da die einzuplanende Operation eines Auftrags auf den unverwandten Maschinen jeweils anders ausgeprägt sein kann.[25]

Beispiel 5.3 Konfliktmenge bei unverwandten Maschinen
Sei eine Situation mit drei Jobs ($N = 3$) und drei Maschinen ($M = 3$) auf einer Stufen eines Flexible-Flow-Shops gegeben. Abbildung 5.4 zeigt eine mögliche Ausprägung. Die einzelnen Operationen werden von Variable $o^{st}_{j,m}$ dargestellt. Dabei gilt Stufenindex st, Auftragsindex j und Maschinenindex m. Bei den Operationen $o^{st}_{j,m}$ eines Jobs j und den Maschinen M_{st} der Stufe st mit $m = \{1, \ldots, M_{st}\}$ gilt, dass diese nur von einer Maschine bearbeitet werden müssen; es wird nur ein m gewählt.

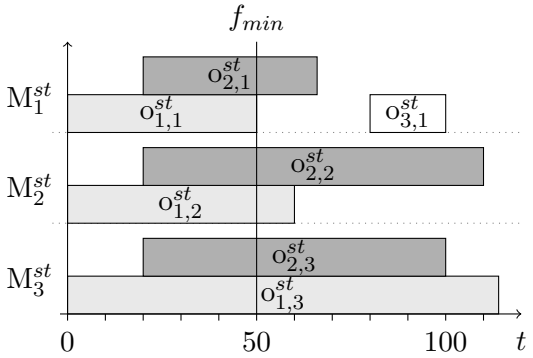


Abbildung 5.4.: Start- und Bearbeitungszeiten eines Flexible-Flow-Shops

Der früheste Fertigstellungszeitpunkt $f_{min} = 50$ wird durch den Wert $o^{st}_{1,1}$ determiniert. Die Operationen $o^{st}_{1,2}, o^{st}_{1,3}, o^{st}_{2,1}, o^{st}_{2,2}$ und $o^{st}_{2,3}$ haben einen früheren Startzeitpunkt als $o^{st}_{1,1}$ ($o^{st}_{j,m} < f_{min}$) und werden in die Konfliktmenge aufgenommen. Operation $o^{st}_{3,1}$ kann nicht vor f_{min} starten und steht daher nicht im Konflikt zu $o^{st}_{1,1}$. Aus der Konfliktmenge CS kann jetzt eine Operation $o^{st}_{j,m}$ ausgewählt und in den Schedule eingeplant werden. □

Solange noch nicht alle Aufträge auf der Stufe eingeplant werden, wird die Iteration fortgeführt. Wenn alle Aufträge eingeplant werden, ist das Teilproblem der aktuellen Stufe st gelöst. Anschließend kann die Belegungsplanung für die nächste Stufe $st + 1$ durchgeführt werden, um eine Gesamtlösung zu generieren. Dabei stellt die Lösung der Stufe st die Ausgangsituation für die nachfolgende Stufe $st + 1$ dar. Algorithmus 5.2 ist zur Berechnung einer Lösung für eine beliebige Stufe einsetzbar, unabhängig davon, ob Lösungen für die anderen Stufen existieren. Um die Eigenschaften des Algorithmus zu

[25] Das Verhältnis der Bearbeitungszeiten der Aufträge untereinander schwankt bei unverwandten Maschinen auf einer Stufe von Maschine zu Maschine und ist nicht wie bei uniformen Maschinen für eine Stufe konstant.

untersuchen, wird das Problem als ein Suchproblem betrachtet.[26] Jede Probleminstanz hat dabei einen Zustandsraum, der deren aktive vollständige und partielle Schedules als Zustände beinhaltet.

Der Zustandsraum bei Flexible-Flow-Shops ist wie bei Job-Shops durch einen Zustandsraumgraphen in der Form eines Baums darstellbar; mehrere Knoten im Zustandsraumgraphen dürfen einen gleichen Zustand repräsentieren. Abbildung 5.5 zeigt einen beispielhaften Zustandsraumgraphen einer Flexible-Flow-Shop-Probleminstanz. Die Wurzel repräsentiert einen leeren Schedule, die inneren Knoten stellen jeweils einen aktiven partiellen Schedule dar. Die Blätter repräsentieren alle vollständigen aktiven Schedules als Lösungen der Probleminstanz. Dieser Baum enthält nur die aktiven Teillösungen und Lösungen der Flexible-Flow-Shop-Probleminstanz.

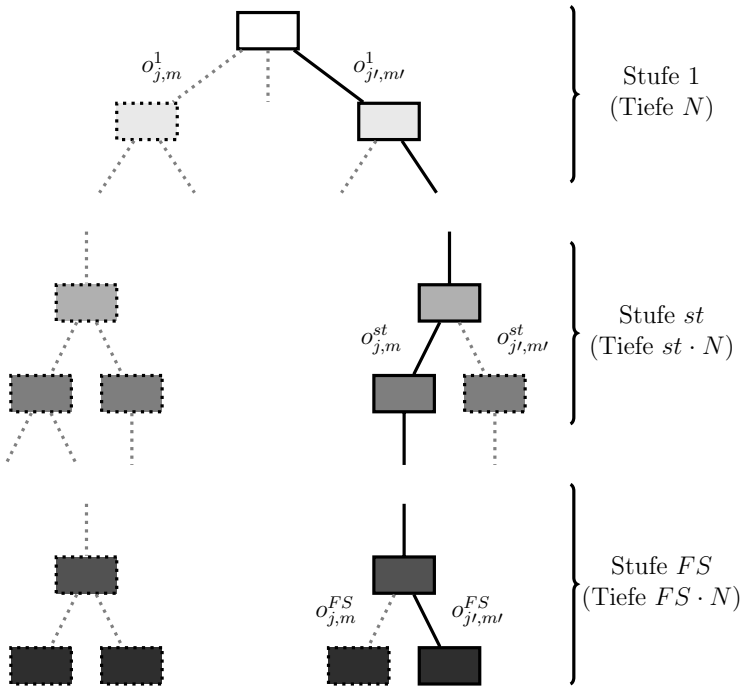


Abbildung 5.5.: Zustandsraumgraph einer Flexible-Flow-Shop-Probleminstanz

Mit Algorithmus 5.2 wird im Zustandsraumgraph eine Stufe *st* durchsucht. Für jeden Teilbaum (Stufe) und insgesamt für den gesamten Baum (Fertigungssystem) wird mittels des Verfahrens nur ein Pfad bis zu einer Lösung durchlaufen; in Abbildung 5.5 als schwarzer, nicht gestrichelter Pfad dargestellt. Jede Kante entspricht genau einer Operation. Da die Operationen eines Konflikts auf derselben Stufe *st* zu bearbeiten sind, haben die Beschriftungen $o^{st}_{j,m}$ aller Kanten einer Stufe den gleichen *st*-Wert. Die stufenweise Anwendung von Algorithmus 5.2 baut den Baum iterativ auf. Jede Iteration des

[26] Gleiches gilt für Algorithmus 5.1.

Algorithmus bildet für jede Operation $o_{j,m}^{st}$ im Konflikt eine Kante, die zu einem partiellen bzw. vollständigen Schedule führt. Es wird also, eine Teilprobleminstanz durch die Einplanung einer Operation $o_{j,m}^{st}$ zu einer anderen Teilprobleminstanz mit einem partiellen Schedule oder zu einer Lösung mit einem vollständigen Schedule verändert. Nach jeder Iteration des Algorithmus wird genau eine Operation ausgewählt und in den Schedule eingeplant. Ab dem Startknoten sind nach N Iterationen, wobei N die Anzahl der Jobs ist, die Operationen aller Aufträge auf der ersten Stufe fertig eingeplant. Wenn der Startknoten mit der Tiefe null im Zustandsbaum bezeichnet wird, wird das Teilproblem auf der ersten Stufe in der Tiefe N gelöst. Dieser Prozess wird für alle Stufen $st = \{1, 2, \ldots, FS\}$ durchgeführt. Die Knoten des Zustandsbaums in der Tiefe N repräsentieren die Teilprobleminstanzen mit jeweils einem partiellen Schedule als Teillösung. In dieser ist der Schedule für die erste Stufe vollständig erstellt. Die Schedules auf den Stufen 2 bis FS wurden noch nicht berechnet. Analog ist in Tiefe $2 \cdot N$ des Zustandsbaums der Schedule auf der Stufe 1 und 2 vollständig, auf den Stufen 3 bis FS jedoch noch offen. Auf der Tiefe $FS \cdot N$ sind alle Jobs auf allen Stufen eingeplant, d. h. die Knoten in der Tiefe $FS \cdot N$ des Baums sind die Lösungsknoten mit jeweils einem vollständigen aktiven Schedule für die Probleminstanz. Resümierend sind drei Eigenschaften des Zustandsraumgraphen von Flexible-Flow-Shop-Probleminstanzen ableitbar:

1. Der Zustandsraumgraph einer Flexible-Flow-Shops-Probleminstanz besitzt eine maximale Tiefe von $FS \cdot N$, wobei FS die Anzahl der Fertigungsstufen und N die Anzahl der Jobs der Probleminstanz ist.

2. Ein Knoten in der Tiefe i im Zustandsraumgraphen repräsentiert einen partiellen Schedule als Teillösung, in dem i Operationen fertig eingeplant sind.

3. Sei $st \in \{1, 2, \ldots, FS\}$ eine Stufe des Fertigungssystems. Ein Knoten in der Tiefe $st \cdot N$ im Zustandsraumgraphen repräsentiert eine Teilprobleminstanz mit einem partiellen Schedule als Teillösung. In diesem sind die Operationen auf den Stufen 1 bis st vollständig eingeplant, die Einplanung der Operationen auf den Stufen $st + 1$ bis FS ist noch offen.

Wie in Job-Shops wird eine erstellte Lösung für Flexible-Flow-Shops durch einen Pfad vom Wurzelknoten bis zu einem Blattknoten im Zustandsbaum dargestellt. In jedem Konflikt wird eine Operation ausgewählt und dem Pfad hinzugefügt. Auch in Flexible-Flow-Shops werden Entscheidungen zur Auswahl einer Operation aus der Konfliktmenge von der Lernkomponente getroffen. Die Auswahlstrategie in Konflikten ist also ein wichtiger Teil des Verfahrens. Diese leitet die Suchrichtung im Zustandsbaum (Lösungsraum) und bestimmt wesentlich die Qualität der generierten Lösung. Hierzu wird im

Folgenden die Lernkomponente entwickelt.

5.3. Entwicklung der Lernkomponente

Im Abschnitt 5.3.1 wird die wissensbasierte Auswahlmethode erarbeitet, die zur Entscheidungsoptimierung integriert wird. Wissensbasierte Systeme unterscheiden sich von konventionellen Programmen dahingehend, dass sie anwendungsspezifisches und -unabhängiges Wissen trennen und das anwendungsspezifische Wissen in einer Wissensbasis ablegen.[27] Herkömmliche Programme implementieren in der Regel beide Arten von Wissen in Form von Algorithmen oder Datenstrukturen. Die Trennung von Wissen in wissensbasierten Systemen ermöglicht sowohl die Transformation als auch die Sicherung des Wissens in einer Datenbank. Mit wissensbasierten Systemen lassen sich aufgrund der Verarbeitungsmöglichkeiten adaptive Systeme entwickeln, die durch Änderung der Wissensbasis dynamisch konfigurierbar sind. Zielsetzung bei dem Einsatz von wissensbasierten Systemen in der Produktionsplanung ist die möglichst performante Planung in dynamischen Job-Shop- und Flexible-Flow-Shop-Umgebungen. WBSGT ist wissensbasiert, da es anwendungsspezifisches Wissen mit einer, im Folgenden genauer konzipierten Klassifikationstechnik akquiriert und zur Problemlösungen einsetzt. Der Vorgang der Wissensakquisition erstellt einen Klassifizierer, der das Wissen repräsentiert und anwendbar macht. Sowohl die hierfür erforderlichen Merkmale als auch deren global und lokal orientierter Charakter werden in Abschnitt 5.3.2 dargelegt. Abschnitt 5.3.3 zeigt die verwendeten Steuerungsregeln als Klassen für den Klassifizierer. Abschließend wird das Training des Verfahrens vorgestellt. Hierbei wird das Training für Job-Shops und für Flexible-Flow-Shops unterschieden, da für beide Organisationstypen nicht die gleichen Trainingsverfahren zur Verfügung stehen.

5.3.1. Situative Regelauswahl mit Naive-Bayes-Klassifikation

Wie beschrieben hat die Auswahl in Konfliktsituationen entscheidende Auswirkungen auf die Qualität der Lösung des erstellten Schedules. Bei zufälliger Operationsauswahl ist die Qualität der Lösung ebenfalls immer zufällig und somit nicht kontrollierbar. Ähnliches gilt für die Auswahl mit immer der gleichen Entscheidungsregel.[28] Ist die

[27] Vgl. (Beierle und Kern-Isberner, 2008, Seite 8 ff.).
[28] Siehe Abschnitt 3.1.2.

Auswahlstrategie komplex, um damit gute Lösungen zu generieren, wird die Anforderung nach schnellen Entscheidungen des Verfahrens nicht erfüllt.[29] Aus diesem Grund ist eine Strategie zu entwickeln, die Konflikte sowohl schnell als auch gut im Sinne der Zielsetzung auflöst. Daher wird die vorgestellte Steuerungskomponente[30] um eine wissensbasierte Regelauswahlmethode – dem Klassifizierer – im Sinne trainierter Regeln erweitert. Das Wissen ist zunächst zu trainieren, um es anschließend situationsabhängig anwenden zu können.

WBSGT verwendet als maschinelles Lernverfahren den Naive-Bayes-Klassifizierer, da sich dieser bei der Analyse der Lernverfahren in Abschnitt 3.2 als am Geeignetsten herausgestellt hat. Der Naive-Bayes-Klassifizierer ist u. a. in der Lage, schnell Entscheidungen zu generieren, mit vielen Merkmalen zu arbeiten sowie große Trainingsmengen zu verarbeiten und die Wahrscheinlichkeit bei einer Entscheidung mit auszugeben.[31] Der Klassifizierer lernt aus einer Menge von Objekten (OB), die in Klassen (CL) aufgeteilt sind. Dabei ist für jedes Tupel (o_i, cl_j) bestehend aus einem Objekt o_i und einer Klasse cl_j bekannt, ob o_i zu cl_j gehört. Der Klassifizierer kann nach dem Lernvorgang bei der Wissensanwendung idealerweise entscheiden, ob o'_i zu cl'_j gehört oder nicht. Dies gilt unabhängig davon, ob der Klassifizierer mit (o'_i, cl'_j) trainiert wird oder nicht. Dieser Idealzustand wird in der Praxis jedoch nicht erreicht, da Klassifizierer Fehler machen. Für die hier vorliegende Anwendung ist dieses unproblematisch, wenn die Klassifikationsgenauigkeit hinreichend groß ist.[32]

Die Wissensanwendung ist die Entscheidung für eine Operation und / oder Maschine innerhalb der Konfliktmenge der in Abschnitt 5.2 entwickelten Steuerungskomponente für Job-Shop- oder Flexible-Flow-Shop-Probleminstanzen. WBSGT kann mittels des lenkenden Einsatzes des trainierten Klassifizierers bei Steuerungsereignissen einen Schedule für die resultierende Probleminstanz berechnen. Der Klassifizierer wählt in jedem Konfliktfall die zu verwendende Regel aus. Die Eingabe des Klassifizierers hierzu ist der Merkmalsvektor, der den aktuellen Zustand des Fertigungssystems beschreibt.

Beispiel 5.4 WBSGT-Klassifikation
Abbildung 5.6 zeigt ein Beispiel eines möglichen Ablaufs des Verfahrens für die ersten beiden Iterationen zur Auswahl einer Operation in Job-Shops.

Da es ausgehend vom leeren Schedule zu keinem Konflikt kommt, plant WBSGT in diesem Beispiel die Operation $o_{1,2}$ ein. In der zweiten Iteration entsteht ein Konflikt, da

[29] Siehe Abschnitt 2.2.2.3.
[30] Siehe Abschnitt 5.2.
[31] Siehe Abschnitt 3.2.3.
[32] Siehe Abschnitt 2.2.3.1.

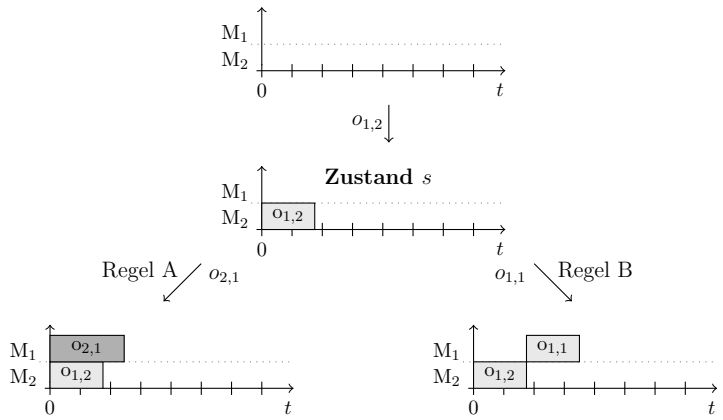


Abbildung 5.6.: Klassifikation eines Zustands

die Operationen $o_{1,1}$ und $o_{2,1}$ alternativ auf Maschine M_1 einplanbar sind. Eingabe des Klassifizierers ist der Merkmalsvektor, der den Zustand s repräsentiert. Der Klassifizierer könnte beispielsweise die Regel A klassifizieren, welche die Operation $o_{2,1}$ auswählt. WBSGT plant in diesem Fall die zugehörige Operation $o_{2,1}$ ein. Auf diese Weise wird das Verfahren fortgesetzt, bis alle Operationen eingeplant sind. □

Für den Einsatz von Techniken des maschinellen Lernens im Scheduling ist ein Modell für Scheduling-Probleminstanzen notwendig. Eine Probleminstanz im Shop-Scheduling besteht aus Parametern wie Anzahl der Maschinen und der Aufträge, technologische Reihenfolgen, Bearbeitungszeiten etc. Jede Probleminstanz hat dabei einen Zustandsraum, der alle ihre zulässigen aktiven Schedules beinhaltet. Wie gezeigt ermöglicht die Steuerungskomponente das Erstellen des Zustandsraumgraphen einer Probleminstanz.[33]

Ein probabilistisches Klassifikationsverfahren wie der Naive-Bayes-Klassifizierer basiert auf statistischen Wahrscheinlichkeiten und gibt die Wahrscheinlichkeit aus, mit der die Instanz o'_i zur Klasse cl'_j gehört. Probabilistische Verfahren sind insbesondere für Anwendungen geeignet, in denen viele Daten vorhanden sind, aber nur wenig Wissen über deren Entstehungsprinzipien. Dieses ist bei der Ablaufplanung und -steuerung in Job-Shops und Flexible-Flow-Shops der Fall: Wird die Aufgabe der Ablaufplanung und -steuerung als das Lösen von Konfliktsituationen gesehen, so wie dieses in der Steuerungskomponente der Fall ist, dann ist unklar nach welchem Prinzip die Konflikte aufzulösen sind, um zu guten Lösungen zu gelangen.

Wie gezeigt wird immer dann, wenn es innerhalb der Steuerungskomponente zu Kon-

[33] Siehe Abschnitt 5.2.

fliktsituationen kommt, der trainierte Klassifizier der Lernkomponente zur Konfliktauflösung eingesetzt. Die Konflikte werden also nicht durch die zufällige Auswahl einer Operation oder fest eingestellte Regeln aufgelöst. Insgesamt bestimmt der Klassifizierer also den Pfad durch den Zustandsraum. Es ist nicht erforderlich zeitintensiv verschiedene Alternativen zu probieren, sondern es wird nur ein Pfad von der Wurzel zur Lösung durchlaufen. Dabei wird durch die trainierte Lernkomponente eine gute Lösung sichergestellt. Damit durch die Lernkomponente eine Regel auszuwählen ist, wird die aktuelle Steuerungssituation (Zustand) wie dargestellt über Merkmale abgebildet. Darauf aufbauend errechnet der Naive-Bayes-Klassifizierer, welche der zur Verfügung stehenden Steuerungsregeln die Konfliktsituation am besten auflösen kann. Wie dargelegt ist das in Job-Shop jeweils eine einzige Entscheidung.

Kommt es in Flexible-Flow-Shops zu Steuerungssituationen mit Konflikten[34] wird auch hier vom Klassifizierer die geeignetste Regel zur Auflösung der Konfliktmenge ausgewählt. Im Unterschied zu Job-Shops ist hier zusätzlich zur Auswahl der Operationsregel auch eine Regel zur Auswahl aus den alternativen Maschinen zu wählen. Die Entscheidung ist über die zwei folgenden alternativen Auswahlmethoden abbildbar:

Auswahlmethode 1: Zwei-Klassifizierer-System[35]
Bei dem Zwei-Klassifizierer-System sind die Entscheidungen für eine Operation und eine Maschine getrennt. Nach der Auswahl einer Regel für die Operation durch den ersten Klassifizierer wird eine Regel zur Maschinenauswahl vom zweiten Klassifizierer gewählt.

Auswahlmethode 2: Ein-Klassifizierer-System[36]
Beim Ein-Klassifizierer-System wird sowohl die Operation als auch die Maschine in einer Entscheidung ausgewählt. Hierbei werden alle passenden Operationen der Konfliktaufträge mit in die Konfliktmenge aufgenommen. Wird jetzt über den Klassifizierer eine Operation ausgewählt, ergibt sich daraus automatisch die Maschine, welche die Operation bearbeitet.

Je nachdem, welche Auswahlmethode gewählt wird, muss die Steuerungskomponente geringfügig anders ausgestaltet sein. In Algorithmus 5.2 wird die Ausprägung für ein Ein-Klassifizierer-System dargestellt. Beim Zwei-Klassifizierer-System werden bei der Bestimmung der minimalen Fertigstellungszeit (Zeilen 5-16) nur die Jobs betrachtet. Maschinen werden nicht mit berücksichtigt. Die Jobs werden in die Konfliktmenge CS

[34] Siehe Algorithmus 5.2.
[35] Für dein Einsatz in Flexible-Flow-Shops mit uniformen Maschinen.
[36] Für dein Einsatz in Flexible-Flow-Shops mit unverwandten Maschinen.

nur einmal, mit deren Standard-Bearbeitungszeit ps_j^{st} aufgenommen. Die Konfliktmenge der alternativen Maschinen wird im Anschluss an die Auswahl eines Jobs aufgebaut.

Klassifizierer gehören zur Klasse der überwachten Lernmethoden.[37] Durch das Training wird jedoch nur eine endliche Menge von Trainingsbeispielen erzeugt, die in den wenigsten Fällen alle Ausprägungen einer Klasse abdeckt. Wenn alle Ausprägungen einer Klasse abgedeckt würden, wäre das eine vollständige Aufzählung und der Klassifizierer überflüssig. Wie in Abschnitt 3.1 dargestellt, sind vollständige Aufzählungen bei dem hier vorliegenden Problem nicht durchführbar. Beim Klassifizieren kommt es folglich zu Situationen, in denen Objekte zu klassifizieren sind, obwohl für die Ausprägungen deren Merkmale keine oder wenige Trainingsinstanz vorhanden ist. Im Idealfall sollten unbekannte Objekte von Klassifizierer korrekt zugeordnet werden. Wenn der Klassifizierer über eine hohe Generalisierbarkeit verfügt, kann er Objekte richtig zuordnen, deren Merkmalsausprägungen sehr unähnlich zu denen der Trainingsinstanzen sind. Bei der Erstellung eines Klassifizierers wird eine hohe Generalisierbarkeit angestrebt, die allerdings von vielen Faktoren beeinflusst wird. Die Zustände sind durch aussagekräftige (passende) Merkmale zu beschreiben. Sind gute Merkmale vorhanden, aber nur wenige Trainingsinstanzen, welche nicht die gesamte Klassifikation repräsentieren können, spezialisiert sich der Klassifizierer auf diese Beispiele und ist damit übertrainiert (overfitted).[38]

Entscheidungen sind also keinesfalls immer sicher. Der Naive-Bayes-Klassifizierer kann zusätzlich zur Entscheidung, die Wahrscheinlichkeit der Entscheidung mit ausgeben. So ist ermittelbar, wie sicher die einzelnen Entscheidungen sind, um damit bei zu unsicheren Entscheidungen weitere Maßnahmen zu ergreifen.[39] Da wie gezeigt die Merkmale zur Beschreibung der Situation im Fertigungssystem wichtig sind, werden diese im Folgenden entwickelt.

5.3.2. Situationsbeschreibende Merkmale

Damit durch den Klassifizierer in Konfliktsituationen Regeln ausgewählt werden können, müssen derartige Zustände beschrieben werden. Dazu werden situationsbeschreibenden Merkmale eingesetzt. Als Eingabe wird dem Klassifizierer ein Vektor von Merk-

[37] Zu überwachtem Lernen vgl. u. a. (Klösgen und Żytkow, 2002, Seite 69 f.).
[38] Siehe Abschnitt 3.2.
[39] Derartige Maßnahmen könnten beispielsweise die simulative Vorschau zur Informationserweiterung (siehe Abschnitt 5.4.) oder das zufällige Auswählen aus den sichersten Optionen bei gleichzeitigem Wegfall der unsichersten sein.

malen übergeben.[40] Die Merkmale repräsentieren die Steuerungszustände in Konfliktsituationen, hier partielle Schedules des Zustandsraumgraphen, d. h. die aktuelle Situation des Fertigungssystems. Merkmale beschreiben also das vom Klassifizierer geforderte Modell der Zustände und sind für die Qualität des entwickelten Verfahrens von großer Bedeutung. Nur wenn die Situation richtig erkannt wird, kann durch den Klassifizierer auch eine gute Entscheidung getroffen werden. Das Modell muss die für das Anwendungsgebiet relevanten Merkmale beinhalten, da von diesen abhängt, welche Informationen verfügbar sind und was daraus abzuleiten ist. Damit hat der Modellierer großen Einfluss auf die Güte des Klassifikationsverfahrens. Die Bedeutung der Merkmale soll zusätzlich durch folgendes Beispiel aus dem täglichen Umfeld dargelegt werden.

Beispiel 5.5 Bedeutung der eingesetzten Merkmale
Besitzt ein Navigationssystem in einem Auto keine externen Informationen wie das TMC-Signal[41], sondern nur lokale Informationen des Fahrzeugs wie Geschwindigkeit, Position usw., wird ein voraus liegender Stau nicht erkannt und das Navigationssystem reagiert nicht bzw. nicht korrekt. Das System würde zwar für sich richtig reagieren, aber eben auf die falsch erkannte Situation des nicht vorhandenen Staus. □

Es besteht also ein starker Zusammenhang zwischen den Merkmalen und der Qualität der Entscheidung des Naive-Bayes-Klassifizierers. Die Auswahl und Entwicklung der Merkmale ist schwierig, da die Zusammenhänge von Merkmalen und deren tatsächliche Relevanz bei einer Entscheidungsfindung nicht immer einsehbar sind. Nachfolgend werden die hier prototypisch eingesetzten Merkmale entwickelt. Hierbei ist zwischen den Merkmalen in Abhängigkeit der betrachteten Organisationsform zu unterscheiden. Insgesamt ist darauf zur achten, dass die Merkmale generischer Natur sind. Würde beispielsweise die Auslastung der Maschine 3 einer Fertigungsstufe als Merkmal verwendet, dann müsste im Falle eines Maschinenausfalls dieser Maschine, ein anderer Klassifizierer eingesetzt werden. Dazu wäre eine Vielzahl von unterschiedlichen Klassifizierern einsatzbereit zu halten.

5.3.2.1. Merkmale in Job-Shops

Nachfolgend werden die situationsbeschreibenden Merkmale für den Verfahrenseinsatz in Job-Shop-Umgebungen entwickelt.

[40] Siehe Abschnitt 3.2.
[41] Über den *Traffic Message Channel* (TMC) werden Informationen bzgl. Beeinträchtigungen des Straßenverkehrs im nicht hörbaren Bereich des Radio-Signals übertragen.

Merkmal 1: Verhältnis Konfliktoperationen

Das Verhältnis der Konfliktoperationen ist ein Maß für die relative Komplexität der zu treffenden Entscheidung. Je mehr Konflikte existieren, desto schwieriger wird es für den Klassifizierer die optimale Entscheidung zu treffen. Sei CS die Menge der aktuell konfliktären Jobs und N die Menge aller Jobs.

$$\text{Verhältnis Konfliktoperationen} = \frac{CS}{N} \tag{5.1}$$

Sei beispielhaft ein Job-Shop-Problem mit fünf Jobs $j = \{1, 2, \ldots 5\}$ und drei Maschinen $m = \{1, 2, 3\}$ gegeben. Aktuell wird Maschine 2 betrachtet. Angenommen Auftrag 1, 3 und 4 müssen zum jetzigen Zeitpunkt auf Maschine 2 bearbeitet werden. Damit hat das Situationsmerkmal den Wert $3/5 = 0,6$.

Merkmal 2: Durchschnittlich verplante Bearbeitungszeit

Mit diesem Merkmal wird die durchschnittlich bereits verplante Bearbeitungszeit aller Jobs N. Der Klassifizierer erkennt also beim Vergleich von Werten dieses Merkmals, wie weit die Jobs $j = \{1, 2, \ldots N\}$ bereits fortgeschritten ist. Sei $pj(j)$ die gesamte Bearbeitungszeit und $pf(j)$ die bereits verplante Bearbeitungszeit eines Jobs j.

$$\text{Durchschnittlich verplante Bearbeitungszeit} = \frac{\sum_{j=1}^{N} \frac{pf(j)}{pj(j)}}{N} \tag{5.2}$$

Sei das in Tabelle 5.5 aufgeführte Problem mit drei Aufträgen und zwei Maschinen gegeben. Werte in den Zellen geben die Bearbeitungszeiten wieder, fett darstellte Bearbeitungszeiten markieren bereits eingeplante Operationen.

	Maschine 1	Maschine 2
Job 1	**6**	7
Job 2	**12**	**13**
Job 3	**5**	14

Tabelle 5.5.: Beispielhaft verplante Operationen

Die durchschnittliche verplante Bearbeitungszeit ergibt sich wie folgt: $(6/13 + 25/25 + 5/19)/3 = 0,5749$. Durchschnittlich sind also $57,49\%$ der Bearbeitungszeiten der Jobs bereits eingeplant.

Merkmal 3: Anteil bereits verplanter Operationen

Dieses Merkmal bestimmt das prozentuale Verhältnis der bereits verplanten Operationen im Vergleich zu sämtlichen Operationen der Jobs $j = \{1, 2, \ldots N\}$. Der Klassifizierer erkennt beim Vergleich von Werten dieses Merkmals, wie weit das Scheduling bereits fortgeschritten ist. Sei $oj(j)$ die Anzahl aller Operationen und $of(j)$ die bereits verplanten Operationen eines Jobs j.

$$\text{Anteil bereits verplanter Operationen} = \frac{\sum_{j=1}^{N} of(j)}{\sum_{j=1}^{N} oj(j)} \tag{5.3}$$

Sei das in Tabelle 5.5 aufgeführte Problem mit drei Aufträgen und zwei Maschinen gegeben. Vier der sechs Operationen sind bereits verplant $4/6 = 0,66$. Der Anteil der bereits verplanten Operationen liegt also bei 66%.

Merkmal 4: Vergleich der Merkmale 2 und 3

Der Vergleich bildet ab, ob im bisherigen Scheduling-Verfahren im Durchschnitt eher kurze oder eher lange Operationen ausgewählt wurden. Dabei gibt es drei mögliche Ergebnisse:

- $0 \rightarrow$ das Verhältnis der Fortschritte ist gleich.
- $1 \rightarrow$ Merkmal 2 ist weiter fortgeschritten. Dies bedeutet, dass im Durchschnitt bisher relativ längere Operationen ausgewählt wurden.
- $-1 \rightarrow$ Merkmal 3 ist weiter fortgeschritten. Dies bedeutet, dass im Durchschnitt bisher relativ kürzere Operationen ausgewählt wurden.

Gegeben sei das gleiche Beispiel wie bei Merkmal 2 und 3. Der Fortschritt nach Bearbeitungszeiten ist $57,49\%$ und der Fortschritt nach Operationen 66%. Hieraus ergibt sich -1 als Rückgabewert für Merkmal 4. Die durchschnittliche Bearbeitungszeit der verplanten Operationen beträgt 9 Zeiteinheiten und die durchschnittliche Bearbeitungszeit aller Operationen 9,5. Damit sind bis jetzt verplante Operationen relativ kürzer.

Merkmal 5: Durchschnittliche Maschinenauslastung

Eine Maschine ist ausgelastet, wenn ein Auftrag auf ihr bearbeitet wird. Festgehalten wird die durchschnittliche prozentuale Auslastung aller Maschinen M über die gesamte bisher verplante Zeit zum Zeitpunkt t. Seien

$mb(m)$ die belegten Zeiteinheiten von Maschine m bis zum aktuellen Zeitpunkt t.

$$\text{Durchschnittliche Maschinenauslastung} = \frac{\sum_{m=1}^{M} \frac{mb(m)}{t}}{M} \tag{5.4}$$

Gegeben sei das in Abbildung 5.7 dargestellte Gantt-Diagramm. Der aktuelle Zeitpunkt t sei 25. Zum Zeitpunkt $t = 25$ ist Maschine 1 zu 92% ausgelastet (23/25) und Maschine 2 zu 56% (14/25). Daraus ergibt sich eine durchschnittliche Maschinenauslastung von 74%

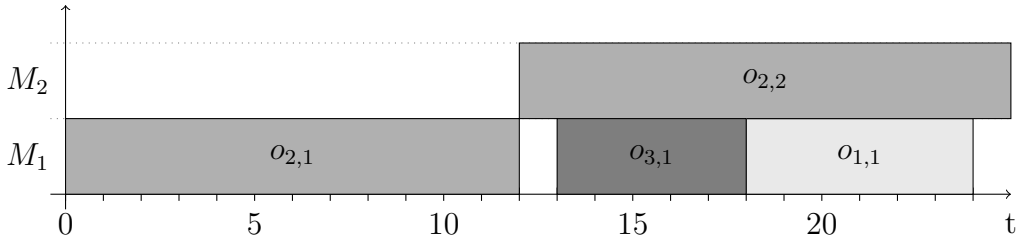


Abbildung 5.7.: Beispielhaftes Gantt-Chart zur Maschinenauslastung

Merkmal 6: Maschinenauslastungsdifferenz
Als Kennzahl für das Merkmal dient die Differenz zwischen der Maschine mit der höchsten Maschinenauslastung und der mit der niedrigsten. Sei wl_m die Auslastung der Maschine m und M die Menge der Maschinen.

$$\text{Differenz der Auslastungen} = max(wl_m) - min(wl_m), \forall m \in M \tag{5.5}$$

Gegeben sei das in Abbildung 5.7 dargestellte Gantt-Diagramm zum Zeitpunkt $t = 25$. Maschine 1 ist zu 92% ausgelastet und Maschine 2 zu 36%, womit sich die Maschinenauslastungsdifferenz von $92\% - 56\% = 36\%$ ergibt.

Merkmal 7: Minimal verbleibende Bearbeitungszeit der Jobs
Zur Berechnung der minimal verbleibenden Bearbeitungszeit der Jobs wird die verbleibende Bearbeitungszeit aller Jobs $j = \{1, 2, \ldots CS\}$ im Konflikt CS untersucht. Sei pt_j die verbleibende gesamte Bearbeitungszeit eines Jobs j.

$$\text{Minimale Bearbeitungszeit} = min(pt_j), \forall j \in CS \tag{5.6}$$

Sei das in Tabelle 5.5 dargestellte Scheduling-Problem gegeben. Die verbleibenden Zeiten der sich im Konflikt befindlichen Jobs 1 und 3 sind 7

und 14. Damit ergibt sich ein Merkmalswert von 7.

Merkmal 8: Maximal verbleibende Bearbeitungszeit der Jobs
Zur Berechnung dieses Merkmals wird die verbleibende Bearbeitungszeit aller Jobs $j = \{1, 2, \ldots CS\}$ im Konflikt CS untersucht. Zurückgegeben wird der größte dieser Werte.

$$\text{Maximale Bearbeitungszeit} = max(pt_j), \forall j \in CS \tag{5.7}$$

Sei das in Tabelle 5.5 dargestellte Scheduling-Problem gegeben. Die verbleibenden Zeiten der sich im Konflikt befindlichen Jobs 1 und 3 sind 7 und 14. Damit ergibt sich ein Merkmalswert von 14.

Merkmal 9: Durchschnittlich verbleibende Bearbeitungszeit
Zur Berechnung dieses Merkmals wird der Durchschnitt der verbleibenden Bearbeitungszeiten aller Jobs $j = \{1, 2, \ldots CS\}$ der Konfliktmenge CS untersucht. Sei $pjc(j)$ die Bearbeitungszeit eines aktuell einzuplanenden Jobs j.

$$\text{Durschnittlich verblante Bearbeitungszeit} = \frac{\sum_{j=1}^{C} Spjc(j)}{CS} \tag{5.8}$$

Gegeben sei das in Tabelle 5.5 dargestellte Scheduling-Problem. Die verbleibenden Zeiten der konfliktären Jobs 1 und 3 sind 7 und 14. Damit ergibt sich ein Merkmalswert von 10,5.

Merkmal 10: Minimale Verspätung
Dieses Merkmal beschreibt die Dauer der minimalen Verspätung unter allen Jobs $j = \{1, 2, \ldots CS\}$ in der Konfliktmenge CS. Sei t der aktuelle Zeitpunkt und ds_j die Summe der Bearbeitungszeiten der bereits eingeplanten Operationen eines Jobs j. Dann wird die Verspätung dl_j des Jobs j wie folgt definiert:

$$dl_j = t - ds_j$$

Die minimale Verspätung wird definiert als:

$$\text{Minimale Verspätung} = min(dl_j), \forall j \in CS \tag{5.9}$$

Gegeben sei das in Abbildung 5.7 dargestellte Gantt-Diagramm. Als Wert der minimalen Verspätung der sich im Konflikt befindlichen Jobs ergibt

sich zum Zeitpunkt $t = 25$ damit der Merkmalswert von 19 des Auftrags 1.

Merkmal 11: Maximale Verspätung

Dieses Merkmal beschreibt die Dauer der maximalen Verspätung unter allen Jobs $j = \{1, 2, \ldots CS\}$ in der Konfliktmenge CS. Die maximale Verspätung wird definiert als:

$$\text{Maximale Verspätung} = max(dl_j), \forall j \in CS \tag{5.10}$$

Sei das in Abbildung 5.7 dargestellte Gantt-Diagramm gegeben. Als Wert der maximalen Verspätung der sich im Konflikt befindlichen Jobs ergibt sich zum Zeitpunkt $t = 25$ damit der Merkmalswert von 20.

Merkmal 12: Durchschnittliche Verspätung

Als Rückgabewert des Merkmals wird die durchschnittliche Verspätung aller Jobs $j = \{1, 2, \ldots CS\}$ in der Konfliktmenge CS berechnet. Die durchschnittliche Verspätung wird definiert als:

$$\text{Durchschnittliche Verspätung} = \frac{\sum_{j=1}^{CS} dl_j}{CS} \tag{5.11}$$

Sei das in Abbildung 5.7 dargestellte Gantt-Diagramm gegeben. Die Verspätungen der konfliktären Aufträge 1 und 3 zum Zeitpunkt $t = 25$ ist: Auftrag 1 = 19 und Auftrag 3 = 20. Daraus ergibt sich eine durchschnittliche Verspätung von 19,75.

Merkmal 13: Ranking Steuerungsregeln

Das Merkmal Ranking der Steuerungsregeln gibt für jede verwendete Steuerungsregel eine Kennzahl zurück. Sei die Menge CS der Jobs in einem Konflikt sowie eine Steuerungsregel *prio* gegeben. Sei o_{prio} die Operation, die von *prio* aus CS auswählt wird. Die Operation o_{prio} ist eine Teiloperation eines Jobs $j_{o_{prio}}$. Die Funktion $Rank(j)$ erstellt für diesen Job ein Ranking über dessen Fortschritt im Vergleich zu anderen Jobs in CS.

$$\text{Prioritätsregelranking} = Rank(j_{o_{prio}}) \tag{5.12}$$

Bei der Funktion $Rank(j_{o_{prio}})$ werden die Merkmale 13-1 bis 13-4 unterschieden.

Sei die in Abbildung 5.8 aufgezeigte Konfliktmenge gegeben. Weiterhin sei

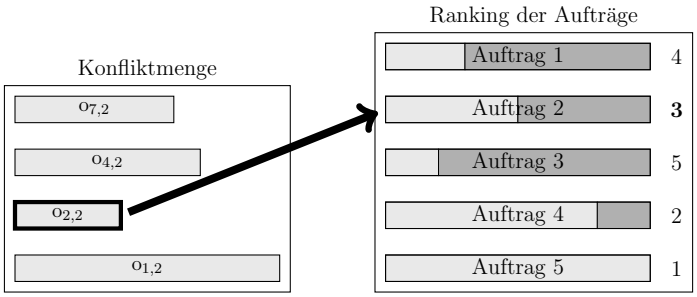


Abbildung 5.8.: Beispielhaftes Steuerungsregelranking

die Steuerungsregel SPT zur Auswahl zu verwenden. Von dieser wird aus den vier möglichen Operationen die Operation $o_{2,2}$ ausgewählt. Das aktuelle Fortschrittsranking zeigt: Auftrag 2 ist zu ungefähr 50% fertiggestellt und belegt derzeit Position drei. Für das Merkmal Steuerungsregelranking würde der Wert 3 zurückgegeben.

Merkmal 13-1: Steuerungsregelranking – Bearbeitungszeiten aller Jobs
Das Fortschrittsranking $Rank(j)$ wird anhand der Bearbeitungszeiten erstellt. Hierzu fließen alle aktuell einplanbaren Jobs in das Ranking mit ein.

Merkmal 13-2: Steuerungsregelranking – Bearbeitungszeiten der Konfliktjobs
Das Fortschrittsranking $Rank(j)$ wird anhand der Bearbeitungszeiten von Jobs, die in Konflikt miteinander stehen erstellt.

Merkmal 13-3: Steuerungsregelranking – Alle durchgeführten Operationen
Das Fortschrittsranking $Rank(j)$ wird anhand der Anzahl der bis jetzt durchgeführten Operationen erstellt. Alle einplanbaren Jobs fließen in das Ranking mit ein.

Merkmal 13-4: Steuerungsregelranking – Durchgeführte Operationen im Konflikt
Das Fortschrittsranking $Rank(j)$ wird anhand der Anzahl der bis jetzt durchgeführten Operationen erstellt. Nur in Konflikt miteinander stehende Jobs werden berücksichtigt.

5.3.2.2. Merkmale in Flexible-Flow-Shops

Nachfolgend werden die bei dem Verfahrenseinsatz in Flexible-Flow-Shops, verwendeten Merkmale entwickelt. Hierbei sind Merkmale für das Ein-Klassifizierer- und das Zwei-Klassifizierer-System zu unterscheiden. Grundsätzlich lassen sich die Merkmale in

Flexible-Flow-Shops in lokale und globale Merkmale unterteilen. Abbildung 5.9 illustriert diesen Sachverhalt.

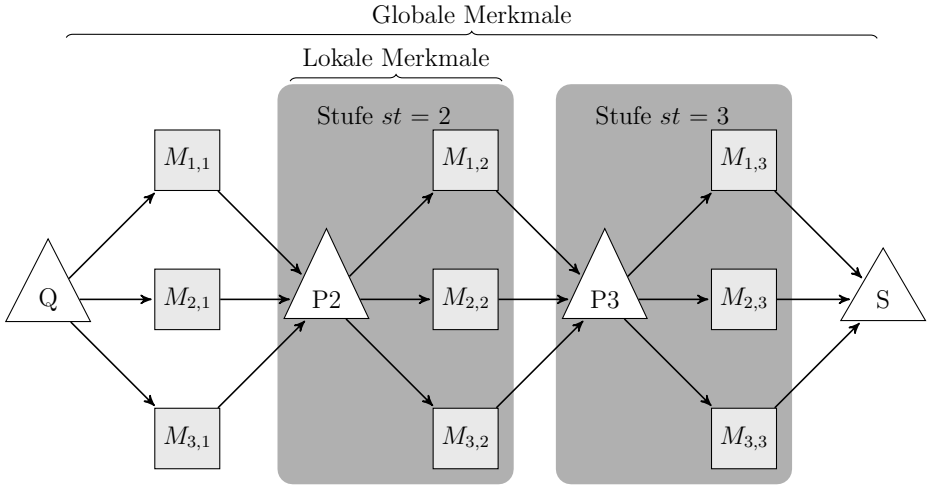


Abbildung 5.9.: Lokale und globale Situationsmerkmale

Lokale Merkmale sind nur auf der aktuellen Fertigungsstufe gültig, deren Ausprägungen unterscheiden sich im Steuerungszeitpunkt t_0 von Stufe zu Stufe. Die globalen Merkmale dagegen sind im Zeitpunkt t_0 auf allen Stufen gleich ausgeprägt, die Information eines globalen Merkmals ist für jede Fertigungsstufe identisch. Über die globalen Merkmale wird die aktuelle Situation des gesamten Flexible-Flow-Shops bei der Entscheidung des Klassifizierers berücksichtigt.

Zwei-Klassifizierer-System: Auftrag und Maschine getrennt

Nachfolgend sind die Merkmale für das Entscheidungssystem mit zwei Klassifizierern – Auftrags- und Maschinenauswahl – beschrieben. Die Merkmale werden alle in beiden Klassifizierern eingesetzt. Dieses gilt nicht für die lokalen Merkmale 5 und 6. Das Merkmal 5 wird ausschließlich im Klassifizierer für die Aufträge verwendet, Merkmal 6 nur im Klassifizierer für die Maschinen.

Lokale Merkmale: Lokale Merkmale sind auf der aktuellen Stufe messbar. Sie beschreiben die Situation dieser Stufe, lassen das restliche Fertigungssystem, die vor- und nachgelagerten Stufen, jedoch außer Acht. Über die lokalen Merkmale kann der Klassifizierer die Situation der Fertigungsstufe erkennen und lokal gute Entscheidungen treffen.

Merkmal 1: Konfliktmengengröße
Die Anzahl der in der Konfliktmenge CS vorhandenen Jobs.

Merkmal 2: Standardabweichung der Bearbeitungszeiten
Die Standardabweichung der Bearbeitungszeiten der in der Konfliktmenge CS vorhandenen Jobs auf der betrachteten Fertigungsstufe st.

Merkmal 3: Verhältnis der bearbeitbaren zu nicht bearbeitbaren Aufträgen
Das Verhältnis der Anzahl von auf der betrachteten Stufe st fertigungsbereiten Jobs N_r, zur Anzahl der noch nicht fertigungsbereiten Jobs N_{nr} in der Konfliktmenge CS.

Merkmal 4: Auslastung der Stufe
Die kumulierte Auslastung der Maschinen M_{st} auf der betrachteten Fertigungsstufe st.

Merkmal 5: Letzte Klassifizierung – Auftrag
Das Ergebnis der zuletzt vorgenommenen Klassifizierung eines Auftrags.

Merkmal 6: Letzte Klassifizierung – Maschine
Das Ergebnis der in derselben Situation bereits vorgenommenen Klassifizierung zur Auftragsauswahl.[42]

Globale Merkmale: Um bei einer Steuerungsentscheidung global gute Ergebnisse zu erzielen, muss die Entscheidung immer auch bezüglich der Situation des gesamten Fertigungssystems getroffen werden. Hierzu werden globale Merkmale eingesetzt. Diese versetzen den Klassifizierer in die Lage, globale Informationen zu berücksichtigen.

Merkmal 1: Auslastung der nachfolgenden Fertigungsstufe
Dieses Merkmal gibt die kumulierte Auslastung der Maschinen auf der nachfolgenden Stufe an. Sei $dmlf_{st}$ die gemittelte Auslastung aller Maschinen 1 bis M_{st} einer Stufe st.

$$\text{Auslastung nachfolgende Stufe} = dmlf_{st+1} \tag{5.13}$$

Merkmal 2: Position der Fertigungsstufe
Die Position der betrachteten Fertigungsstufe st in der Reihe aller Fertigungsstufen FS, gemessen als Fertigungsfortschritt $Fort(st)$.

$$Fort(st) = \frac{st}{FS} \tag{5.14}$$

[42] Es wird vorausgesetzt, dass die Auswahl der Maschine nach der Auswahl eines Auftrags durchgeführt wird. Werden die Entscheidungen in einer anderen Reihenfolge durchgeführt, sind die Merkmale anzupassen bzw. auszutauschen.

Ein-Klassifizierer-System: Auftrag und Maschine gemeinsam

Für den Fall von nur einem eingesetzten Klassifizierer wird eine andere Merkmalsmenge entwickelt. Hier wird in einer Entscheidung sowohl der als nächstes zu bearbeitende Job als auch die Maschine ausgewählt, die diesen Job bearbeitet. Auch bei der Ausprägung des Verfahrens mit einem Klassifizierer wird eine Unterteilung in lokale und globale Merkmale vorgenommen.

Lokale Merkmale:

Merkmal 1: Verhältnis Konfliktaufträge
Verhältnis der Anzahl der Jobs im Konflikt CS und der Anzahl der noch auf der aktuellen Stufe st zu planenden Jobs N.

$$\text{Verhältnis Konfliktoperationen} = \frac{CS}{N} \tag{5.15}$$

Merkmal 2: Verhältnis Maschinenkonflikte
Das Verhältnis zwischen der Anzahl der Maschinen bei denen sich Operationen im Konflikt befinden und der Anzahl aller Maschinen auf der aktuellen Stufe st. Sei CM die Anzahl der möglichen Maschinen in der Konfliktmenge CS, d. h. welche Maschinen für die Operationen $o^{st}_{j,m}$ im Konflikt zur Bearbeitung bereit stehen. Dieses wird über den Index m der Operationen $o^{st}_{j,m}$ im Konflikt berechnet. Hierbei ist M_{st} die Anzahl der Maschinen auf Stufe st.

$$\text{Verhältnis Maschinenkonflikte} = \frac{CM}{M_{st}} \tag{5.16}$$

Merkmal 3: Differenz der Maschinengeschwindigkeiten
Differenz der maximalen und minimalen Maschinengeschwindigkeiten aller Operationen im Konflikt CS. Sei $v^{st}_{j,m}$ die relative Maschinengeschwindigkeit von Operation $o^{st}_{j,m}$.

$$\text{Differenz Maschinengeschwindigkeiten} = max(v^{st}_{j,m}) - min(v^{st}_{j,m}), \forall j \in CS \tag{5.17}$$

Merkmal 4: Differenz der Bearbeitungszeiten
Differenz der maximalen und minimalen Bearbeitungszeiten aller Operationen im Konflikt CS. Sei ps^{st}_j die Standard-Bearbeitungszeit eines Jobs j

auf der Stufe st, $v_{j,m}^{st}$ die relative Maschinengeschwindigkeit der Maschine m auf Stufe st bei Job j und $p_{j,m}^{st} = ps_j^{st}/v_{j,m}^{st}$ die Bearbeitungszeit der aktuellen Operation eines Jobs j auf Maschine m der Stufe st.

$$\text{Differenz Bearbeitungszeiten} = max(p_{j,m}^{st}) - min(p_{j,m}^{st}), \forall j \in CS \quad (5.18)$$

Globale Merkmale:

Merkmal 1: Durchschnittlicher Fortschritt der Konfliktaufträge
Dieses Merkmal gibt den durchschnittlichen Fortschritt aller sich im Konflikt befindenden Jobs zurück. Sei st die aktuelle Stufe und b_j die gesamte geplante Bearbeitungszeit von Job j auf den Stufen 1 bis $st-1$, dann ist die restliche Bearbeitungszeit von j ab Stufe st gegeben als

$$Rest(j) = \sum_{st}^{FS} ps_j^{st} \cdot \frac{\sum_{m=1}^{M_{st}} v_{j,m}^{st}}{M_{st}} \quad (5.19)$$

Der Fortschritt eines Auftrags j wird berechnet mit:

$$Fortschritt(j) = \frac{b_j}{b_j + Rest(j)} \quad (5.20)$$

Über den Fortschritt der Jobs im Konflikt wird der durchschnittliche Fortschrittswert ermittelt:

$$\text{Durchschnittlicher Fortschritt} = \frac{\sum_{j=1}^{CS} Fortschritt(j)}{CS} \quad (5.21)$$

Merkmal 2: Differenz des Fortschritts aller Konfliktaufträge
Die Differenz zwischen maximalen und minimalen Fortschritt der Jobs in der Konfliktmenge CS.

$$\text{Fortschrittsdifferenz} = max(Fortschritt(j)) - min(Fortschritt(j)), \forall j \in CS \quad (5.22)$$

5.3.3. Prioritätsregeln als Klassen

Damit die vom Klassifizierer ausgewählten Klassen (Steuerungsregeln) möglichst effizient ausgeführt werden können, d. h. wenig Rechenzeit beanspruchen, ist bei der Modellierung und Auswahl von Steuerungsregeln für die Lernkomponente auf die einfache

Form der Regeln zu achten. Eine derartige Beschränkung auf einfache Regeln vergrößert zwar die erforderliche Anzahl der Regeln – jedoch nur in einem geringen Maße – und die Regeln lassen sich aufgrund deren einfachen Struktur sehr schnell ausführen. Hierzu eignen sich insbesondere Prioritätsregeln.[43] Diese verbrauchen minimale Rechenzeit und sind gut in der Steuerungskomponente einsetzbar.[44] Ebenso sind mit Prioritätsregeln generische Aussagen möglich. Würde eine Steuerungsregel beispielsweise lauten: Aus drei Aufträgen ist der mit der kürzesten Bearbeitungszeit zu wählen, so könnte diese Klasse auch nur Mengen mit drei Aufträgen bearbeiten. Prioritätsregeln dahingegen wählen aus der Auftragsmenge beispielsweise den Auftrag mit der kürzesten Bearbeitungszeit, unabhängig davon, wie groß die Auftragsmenge ist.

Beispiel 5.6 Auswahl im Zustandsraumgraph mit Prioritätsregeln
Abbildung 5.10 zeigt den Zustandsraumgraphen der Job-Shop-Probleminstanz aus dem Beispiel in Anhang A.1. Dieses Beispiel kann ebenfalls für eine Fertigungsstufe einer Flexible-Flow-Shop-Probleminstanz verstanden werden.

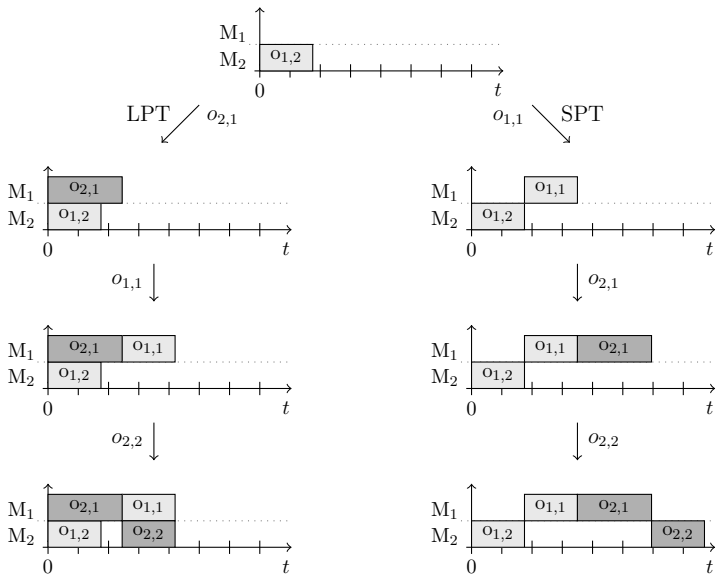


Abbildung 5.10.: Zustandsraumgraph der Job-Shop-Probleminstanz[45]

Die Probleminstanz besteht aus zwei Maschinen und zwei Aufträgen und hat demnach vier Operationen. Der Zustandsraumgraph entspricht einem Baum und hat die Tiefe

[43] Siehe Abschnitt 3.1.2.

[44] Ständen der Steuerungskomponente hinreichend viele Prioritätsregeln zur Anwendung bereit, könnte diese durch die unterschiedliche Anwendung der Regeln alle Lösungen für das Scheduling-Problem erzeugen. Demzufolge ist über die unterschiedliche Anwendung von Prioritätsregeln auch die optimale Lösung abbildbar.

[45] Für die Daten der Probleminstanz siehe Beispiel A.1.

vier, wobei die Wurzel mit schon einer eingeplanten Operation in Tiefe 1 liegt. Insgesamt gibt es zwei verschiedene vollständige Schedules, die gültig und aktiv sind. Jeder Knoten ist durch seinen zugehörigen Schedule dargestellt, die Kanten sind mit den hinzukommenden Operationen beschriftet. Die Verzweigung des Baumes ist ein Konflikt, der einerseits mit der Regel SPT und andererseits mit der Regel LPT aufgelöst wird. Die Lösung mit der kürzesten Produktionsdauer C_{max} kann mit der Regel LPT erreicht werden. □

Wie dargestellt sind Prioritätsregeln $prio_0, \ldots, prio_p$[46] als Klassen $cl_{prio_0}, \ldots, cl_{prio_p}$ für die Zustände des Zustandsraumgraphen einer Scheduling-Probleminstanz I geeignet. Ein Zustand s gehört in die Klasse cl_{prio}, wenn in ihm die Regel $prio$ angewendet wird. Dabei werden die Zustände einer initialen Lösung für I verwendet, die als Merkmalsvektoren repräsentiert werden. Jeder Knoten v_i des Zustandsraumgraphen beschreibt einen Zustand s_i. Ein Zustand mit k Nachfolgern entspricht einem Konflikt in der Steuerungskomponente mit k Alternativen. Die mit Operationen beschrifteten Kanten können auch mit den Prioritätsregeln beschriftet werden, welche die Operation einer Kante auswählen. Der Knoten v_i habe z. B. eine ausgehende Kante, die mit der Operation o beschriftet sei. Die Regel $prio$ wähle in der zu v_i gehörenden Konfliktmenge o aus. Die ausgehende Kante wird mit $prio$ beschriftet.[47]

Jeder Lösungskandidat σ einer Scheduling-Probleminstanz hat genau einen Pfad $P(\sigma)$ durch den Zustandsraumgraphen. Dieser besteht aus einer Folge von Zuständen ($v_0 = v_r, \ldots, v_{l-1}, v_l = v_\sigma$) von der Wurzel v_r ausgehend bis zum Knoten v_σ. Der Knoten v_σ steht dabei für den Lösungskandidaten σ. Eine Klasse cl_{prio} mit dem Namen einer Prioritätsregel $prio$ fasst alle Zustände $v_i \in P(\sigma)$ zusammen, wenn seine ausgehende Kante (v_i, v_{i+1}) mit der Operation beschriftet ist, welche die Regel $prio$ in Zustand v_i auswählt. Abbildung 5.11 illustriert dieses. Hier sind eine optimale Lösung und ihr Pfad – nicht gepunktete Knotenverbindungen – durch den Zustandsraumgraphen eingetragen.

Durch den Pfad $P(\sigma)$ ist für jeden Konflikt erkennbar, welche Alternative zur optimalen Lösung führt. Im Idealfall sind diese Alternativen durch jeweils eine Prioritätsregel auswählbar. Die Klasse cl_{prio} beinhaltet alle Zustände in Form von Merkmalsvektoren; diese haben eine ausgehende Kante, die

1. auf dem Pfad der optimalen Lösung liegt und
2. mit $prio$ ausgewählt wird.

[46] Seien p die verwendeten Prioritätsregeln.

[47] Die ausgehende Kante kann mit allen weiteren Prioritätsregeln, die o auswählen, beschriftet werden (siehe Abschnitt 5.3.3.3).

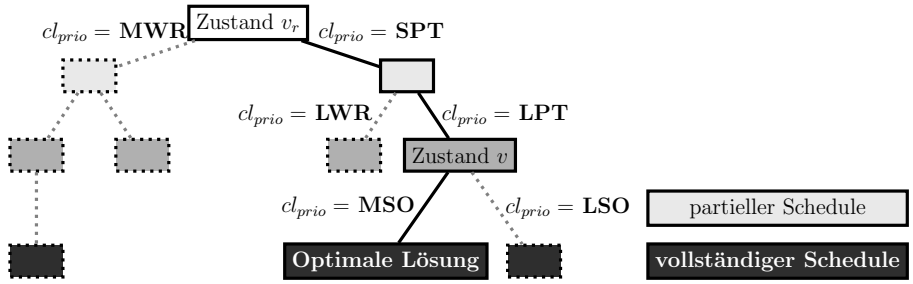


Abbildung 5.11.: Prioritätsregeln als Klassen

Die Instanzen einer Klasse cl_{prio} müssen sich nicht auf Zustände der Lösung einer Shop-Scheduling-Probleminstanz I beschränken. Es eignen sich auch mehrere Lösungen von I oder auch von ähnlichen Probleminstanzen I'. Um gute Klassifikationsergebnisse zu erzielen, sind durch die Trainingskomponente mehrere Probleminstanzen I' zu lösen.

5.3.3.1. Steuerungsregeln in Job-Shops

Um möglichst breit aufgestellt zu sein, d. h. so viele Entscheidungen wie möglich mit so wenigen Regeln wie möglich zu treffen, werden für Job-Shops prototypisch die in Tabelle 5.6 dargestellten Prioritätsregeln verwendet.

Prioritätsregel	Beschreibung
SPT (Shortest Processing Time)	Die Operation mit der kürzesten Bearbeitungszeit wird bevorzugt.
LPT (Longest Processing Time)	Die Operation mit der längsten Bearbeitungszeit wird bevorzugt.
LSO (Least Scheduled Operations)	Der Auftrag mit den am wenigsten bearbeiteten Operationen wird bevorzugt.
MSO (Most Scheduled Operations)	Der Auftrag mit den am meisten bearbeiteten Operationen wird bevorzugt.
LWR (Least Work Remaining)	Der Auftrag mit den wenigsten noch zu bearbeitenden Operationen wird bevorzugt.
MWR (Most Work Remaining)	Der Auftrag mit den meisten noch zu bearbeitenden Operationen wird bevorzugt.

Tabelle 5.6.: Prioritätsregeln in Job-Shops

Hierbei werden gegensätzliche Regeln wie SPT und LPT oder LWR und MWR einbezogen, um die mit den Prioritätsregeln abgedeckte Menge von Operationen bei einer kleinen Regelmenge möglichst groß zu halten. Diese Regelmenge für den Klassifizierer ist einfach zu verändern und damit den Gegebenheiten der realen Einsatzumgebung

anzupassen. Kommt es bei der Auswahl mit einer Regel zu einer Situation, in welcher der Auftrag nicht eindeutige auswählbar ist, sind weiter Maßnahmen zu ergreifen. Diese werden im Abschnitt 5.3.3.3 beschrieben.

5.3.3.2. Steuerungsregeln in Flexible-Flow-Shops

Bei dem Verfahren in Flexible-Flow-Shops kann, wie beschrieben, ein Ein-Klassifizierer- oder ein Zwei-Klassifizierer-System eingesetzt werden. Aus diesem Grund ist zwischen den Steuerungsregeln der beiden verwendeten Klassifikationsmethoden zu differenzieren. Hierzu werden nachfolgend die prototypischen Steuerungsregeln für die jeweilige Klassifikationsmethode vorgestellt.

Steuerungsregeln für das Zwei-Klassifizierer-System

Bei dem Zwei-Klassifizierer-System ist zwischen den Regeln für die Auftragsauswahl und die Maschinenauswahl zu unterscheiden. Für die Steuerungsregeln zur *Auftragsauswahl* gilt: Naturgemäß weisen die Aufträge in einem Flexible-Flow-Shop mit uniformen Maschinen große Ähnlichkeiten untereinander auf. Aufträge sind in der gleichen Reihenfolge und auf den gleichen Fertigungsstufen zu bearbeiten, jedoch sind die Bearbeitungszeiten der Aufträge unterschiedlich. Aus diesem Grund können die Bearbeitungszeiten der Aufträge als Unterscheidungsmerkmal, d. h. Klasse verwendet werden. Zwei Aufträge innerhalb einer Konfliktmenge sind als herausragend einzustufen; derjenige mit der kürzesten und derjenige mit der längsten Bearbeitungszeit auf der jeweils betrachteten Fertigungsstufe. Es ist zu erwarten, dass eine Entscheidung für einen dieser exponierten Aufträge im Vergleich zur Entscheidung für den anderen exponierten Auftrag einen großen – möglicherweise den größtmöglichen – Unterschied bezüglich der Lösungsgüte bedeutet. Um unabhängig von den in der Konfliktmenge vorhandenen Aufträgen eine Entscheidung zu ermöglichen, sollen aufgrund dieser Schlussfolgerungen für die Wahl eines zu bearbeitenden Auftrags die folgenden Prioritätsregeln verwendet werden:

- SPT – Shortest Processing Time
- LPT – Longest Processing Time

Damit ist es möglich, mit einer geringen Anzahl von möglichen Klassifikationsklassen, den entscheidenden Auftrag auszuwählen. Für die Steuerungsregeln zur *Maschinenauswahl* gilt: Die uniformen Maschinen einer Stufe sind untereinander sehr ähnlich, weil die Maschinen die gleichen Fähigkeiten besitzen. Zu unterscheiden sind sie im Wesentlichen durch ihre Bearbeitungsgeschwindigkeit. Diese ist jedoch für sämtliche Aufträge

identisch und wird nicht erst durch den Auftrag determiniert. Analog zu den vorgestellten Überlegungen zur Auswahl eines Auftrags, werden deshalb zur Maschinenauswahl folgende initiale Regeln verwendet:

- Fastest machine – Die schnellste Maschine auf der betrachteten Fertigungsstufe.
- Slowest machine – Die langsamste Maschine auf der betrachteten Fertigungsstufe.

Damit ist es bei den Regeln zur Maschinenauswahl, analog zu den Regeln für die Auftragsauswahl, möglich, mit einer geringen Anzahl möglicher Klassifikationsklassen, die entscheidende Maschine auszuwählen. Hierbei ist die Nutzung zahlreicher weiterer Prioritätsregeln, die beispielsweise die Auslastung der Maschinen als Entscheidungskriterium verwenden, denkbar. Gleiches gilt für die Auftragswahl.

Ähnlich wie bei den Steuerungsregeln für Job-Shops kann es bei der Auswahl mit einer Regel in Flexible-Flow-Shops mit uniformen Maschinen zu einer Situation kommen, in welcher der Auftrag bzw. die Maschinen nicht eindeutig ausgewählt werden können. Darum sollen die in Abschnitt 5.3.3.3 beschriebenen Maßnahmen zur Auswahl angewendet werden.

Steuerungsklassen für das Ein-Klassifizierer-System

Bei dem System mit einem Klassifizierer wird in einer einzigen Entscheidung sowohl die Operation, als auch die bearbeitende Maschine ausgewählt. Für diesen einstufigen Entscheidungsprozess sind mehr Auswahlregeln als beim Zwei-Klassifizierer-System erforderlich. Die hierzu prototypisch verwendeten Regeln werden nachfolgend skizziert.

Regel 1: $\mathrm{Min}\, p^{st}_{j,m}$
Die Operation mit der kürzesten Bearbeitungszeit $p^{st}_{j,m}$ erhält die höchste Priorität.

Regel 2: $\mathrm{Max}\, p^{st}_{j,m}$
Die Operation mit der längsten Bearbeitungszeit $p^{st}_{j,m}$ erhält die höchste Priorität.

Regel 3: $\mathrm{Min}\, v^{st}_{j,m}$
Die Operation mit der niedrigsten relativen Maschinengeschwindigkeit $v^{st}_{j,m}$ wird bevorzugt.

Regel 4: $\mathrm{Max}\, v^{st}_{j,m}$
Die Operation mit der höchsten relativen Maschinengeschwindigkeit $v^{st}_{j,m}$ wird bevorzugt.

Regel 5: $\mathrm{Min}\, c^{st}_{j,m}$
Die Operation mit der frühesten Fertigstellungszeit wird bevorzugt.

Regel 6: $\mathrm{Max}\, c^{st}_{j,m}$
Die Operation mit der spätesten Fertigstellungszeit wird bevorzugt.

Regel 7: $\mathrm{FCFS} - \mathrm{Min}\, p^{st}_{j,m}$
Nur FCFS bildet hier keine Prioritätsregel, sondern trifft eine Vorauswahl und wird mit anderen Prioritätsregel kombiniert, um die Prioritätsmengengröße zu reduzieren. Die Operationen $o^{st}_{j,m}$ mit der frühesten Startzeit werden bevorzugt. Das heißt, die Operationen haben den kleinsten auftrags- (jst_j) oder maschinenbedingten ($mst(i)$) Startzeitwert in der Konfliktmenge. Anschließend wird die Regel $\mathrm{Min}\, p^{st}_{j,m}$ zur Zuweisung der Prioritäten für die Operationen angewendet.

Regel 8: $\mathrm{FCFS} - \mathrm{Max}\, p^{st}_{j,m}$
Die Vorauswahl mittels FCFS reduziert die Konfliktmenge und $\mathrm{Max}\, p^{st}_{j,m}$ weißt die Prioritäten zu.

Regel 9: $\mathrm{FCFS} - \mathrm{Min}\, v^{st}_{j,m}$
Die Vorauswahl mittels FCFS reduziert die Konfliktmenge und $\mathrm{Min}\, v^{st}_{j,m}$ weißt die Prioritäten zu.

Regel 10: $\mathrm{FCFS} - \mathrm{Max}\, v^{st}_{j,m}$
Die Vorauswahl mittels FCFS reduziert die Konfliktmenge und $\mathrm{Max}\, v^{st}_{j,m}$ weißt die Prioritäten zu.

Nachfolgend wird die Operationsauswahl unter Verwendung der vorgestellten Prioritätstregeln an einem Beispiel illustriert.

Beispiel 5.7 Operationsauswahl in Flexible-Flow-Shops mit unverwandten Maschinen
Abbildung 5.12 zeigt eine beispielhafte Konfliktmenge eines Flexible-Flow-Shops mit unverwandten Maschinen, bestehend aus drei Maschinen und zwei Aufträgen mit insgesamt sechs Operationen. Die Standard-Bearbeitungszeit von Auftrag 1 ist 100, die von Auftrag 2 ist 50.

Die vorgestellten Prioritätsregeln wählen wie folgt aus:[48]
$\mathrm{Min}\, p^{st}_{j,m} = o^{st}_{2,1}$; $\mathrm{Max}\, p^{st}_{j,m} = o^{st}_{2,2}$; $\mathrm{Min}\, v^{st}_{j,m} = o^{st}_{1,2}$; $\mathrm{Max}\, v^{st}_{j,m} = o^{st}_{2,2}$; $\mathrm{Min}\, c^{st}_{j,m} = o^{st}_{2,1}$; $\mathrm{Max}\, c^{st}_{j,m} = o^{st}_{2,2}$; $\mathrm{FCFS} - \mathrm{Min}\, p^{st}_{j,m} = o^{st}_{1,2}$; $\mathrm{FCFS} - \mathrm{Max}\, p^{st}_{j,m} = o^{st}_{1,2}$; $\mathrm{FCFS} - \mathrm{Min}\, v^{st}_{j,m} = o^{st}_{1,2}$; $\mathrm{FCFS} - \mathrm{Max}\, v^{st}_{j,m} = o^{st}_{1,3}$. □

[48] Die relativen Maschinengeschwindigkeiten sind für Auftrag 1 ($v^{st}_{1,1} = 4/5; v^{st}_{1,2} = 2/5; v^{st}_{1,3} = 1$) und für Auftrag 2 ($v^{st}_{2,1} = 3/5; v^{st}_{2,2} = 11/5; v^{st}_{2,3} = 8/5$).

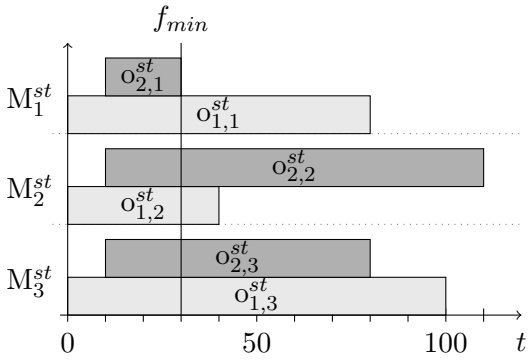


Abbildung 5.12.: Beispielhaftes Flexible-Flow-Shop Konfliktset

Wie in Job-Shops und beim Zwei-Klassifizierer-System kann es auch beim Ein-Klassifizierer-System vorkommen, dass mit der klassifizierten Steuerungsregel keine eindeutige Priorisierung eines Auftrags möglich ist. Aus diesem Grund wird bei Steuerungsentscheidungen die in 5.3.3.3 dargelegte Methode verwendet.

5.3.3.3. Potenzmengenklassen

Während des Trainings kann es passieren, dass der einzuplanende Auftrag von mehr als einer Prioritätsegel ausgewählt wird.[49] Diese Situation lässt sich auf verschiedene Art und Weise behandeln.

1. Die erste zutreffende Prioritätsregel wird mit dem Zustand in der Wissensbasis abgelegt und das Training für die aktuelle Operation anschließend beendet:
   Zustand $s = cl_{prio_A}$
2. Für jede geeignete Prioritätsregel wird ein Beispiel in der Wissensbasis abgelegt:
   Zustand $s = cl_{prio_A}$, Zustand $s = cl_{prio_B}$
3. In der Wissensbasis wird ein Beispiel abgelegt, welches die Menge aller geeigneten Prioritätsregeln (Potenzmenge) enthält:
   Zustand $s = \{cl_{prio_A}, cl_{prio_B}\}$

Die *Variante 1* ist unvollständig, da diese nicht sämtliche in der Situation vorhandenen Informationen enthält. So ist beispielsweise bei einer gesamten Regelmenge aus SPT, LPT und MWR die Aussage (SPT, LPT) informativer als lediglich (LPT). Durch (SPT, LPT) wird deutlich, dass MWR für die aktuelle Situation nicht geeignet ist. Diese Information geht beim Klassifikationsbeispiel (LPT) verloren. Bei Umsetzung der *Variante*

[49] Zum Training siehe Abschnitt 5.3.4.

2 werden für ein und dieselbe Instanz (Zustand s), mit exakt den gleichen Werten in allen beschreibenden Attributen, verschiedene Klassifizierungen abgelegt. Dies ist bei Verwendung von Klassifikationstechniken wenn möglich zu verhindern, da eine derartig aufgebaute Wissensbasis für ein maschinelles Lernverfahren sehr ungünstig ist.[50] Für diese mehrfach vorkommenden Belegungen der Attribute mit verschiedenen Klassen ist die korrekte Klasse nicht mehr eindeutig bestimmbar. *Variante 3* nutzt keine einzelnen Prioritätsregeln, sondern eine Untermenge der Potenzmenge aller möglichen Prioritätsregeln zur Klassifizierung. Hiermit wird die Information abgespeichert, welche Regelmenge im Zustand s in der Lage gewesen ist, die richtige Operation aus der Konfliktmenge auszuwählen.

Wird bei der Klassifikation eine Potenzmenge bzw. Untermenge der Potenzmenge klassifiziert, ist zu entscheiden, welche der, von den Regeln ausgewählten Operationen einzuplanen ist. Die auf den ersten Blick einfachste Lösung ist die zufällige Auswahl einer der klassifizierten Regeln. Hiermit ist die Entscheidung jedoch ebenfalls zufällig und insbesondere immer noch nicht zwingend eindeutig, da eine Prioritätsregel mehrere Aufträge gleich bewerten kann. Auch gehen hierbei, wie beschrieben, die Informationen der anderen Regeln verloren. Aus diesem Grund wird ein Ranking-Verfahren zur Auswahl eingeführt.

Beispiel 5.8 Operationsauswahl mit Potenzmengenklassen
Die aktuelle Konfliktmenge besteht aus den in Tabelle 5.7 aufgeführten Operationen. Als Potenzmengenklasse wurde (SPT, LPT, MSO) klassifiziert.

Operation	Bearbeitungszeit	Ausstehende Operationen
$o_{1,2}$	8	7
$o_{2,2}$	2	7
$o_{4,2}$	8	3
$o_{7,2}$	4	3
$o_{9,2}$	2	2

Tabelle 5.7.: Beispielhafte Job-Shop-Konfliktmenge

Mit der Regel SPT werden $o_{2,2}$ und $o_{9,2}$ ausgewählt, mit LPT $o_{1,2}$ und $o_{4,2}$ sowie $o_{9,2}$ mit MSO. Bis auf die Prioritätsregeln MSO sind die Entscheidungen nicht eindeutig. Darum sollen im Folgenden alle Regeln angewendet werden. Diese setzten den Zähler, aller von ihnen ausgewählten Operation, jeweils um Eins hoch. Für das aktuelle Beispiel führt dies zu folgendem Ergebnis:

[50] Vgl. (Witten und Frank, 2005) und siehe Abschnitt 3.2.3.

- Zähler $o_{1,2} = 1$, da von LPT ausgewählt.
- Zähler $o_{2,2} = 1$, da von SPT ausgewählt.
- Zähler $o_{4,2} = 1$, da von LPT ausgewählt.
- Zähler $o_{9,2} = 2$, da von SPT und MSO ausgewählt.

Damit wird Operation $o_{9,2}$ eindeutig ausgewählt. Für den Fall, dass immer noch mehrere Operationen gleich berechtigt sind, wird per Zufall entschieden. Die Menge der Operationen mit dem höchsten Zähler ist in der Regel kleiner als die initiale Menge, was bei dieser Zufallsauswahl zu genaueren Entscheidungen führt. Das formale Verfahren zur Operationsauswahl mittels Potenzmengenklassen zeigt Algorithmus 5.3. □

Algorithmus 5.3 : Operationsauswahl bei Potenzmengen

```
1  pw(o) := {0|o ∈ CS};
2  pw_max := 0;
3  foreach prio ∈ P do
4      foreach o ∈ CS do
           // Bewerte Operationen mit Prioritätsregel
5          if prio selects o then
               // Falls Operation von Regel gewählt wird, Zähler erhöhen
6              pw(o) := pw(o) + 1;
7          end
           // Aktualisiere den höchsten Prioritätswert
8          if pw(o) > pw_max then
9              pw_max := pw(o);
10         end
11     end
12 end
13 bestOperaitions := {o|pw(o) = pw_max};
14 Wähle so zufällig aus den Operationen mit maximalem pw_max;
```

CS	Aktuelle Konfliktmenge
P	Für aktuelle Situation ausgewählte Regelmenge
$pw(o)$	Prioritätswert von Operation o
pw_{max}	Höchster Prioritätswert
so	Eingeplante Operation

Tabelle 5.8.: Legende zur Operationsauswahl mit Potenzmengen

Hinsichtlich der Bewertung der Klassifikationsgenauigkeit von Potenzmengenklassen gilt: Weil unterschiedliche Prioritätsregeln dieselbe Operation aus dem Konfliktset wählen können, erschwert dieses die Evaluierung des Klassifizierers. In der Wissensbasis wurden Beispiele für korrekte Klassifikationen während des Trainings abgespeichert. Ist

der wahre Fehler eines Klassifikationsverfahrens zu berechnen, wird untersucht, in wie viel Prozent der Entscheidungsfälle das Verfahren die korrekte Klassifikation ausgeben kann.[51] Dieses ist bei der vorgestellten Art von Potenzmengenklassen in der Wissensbasis problematisch.

Möglichkeit 1: Originale Klassifikationsgenauigkeit (Variante 3)
Die Klassifikationsgenauigkeit wird wie in Abschnitt 3.2 dargestellt berechnet. Danach wäre die klassifizierte Klasse a bei echter Klasse (a, b) ein Fehler. Dieses ist auch ein Klassifikationsfehler per definitionem, allerdings waren während des Lernens beide Regeln (a, b) gleichwertig. In der trainierten Situation führten (a, b) zum gleichen Ergebnis, egal für welche der beiden Regeln sich entschieden wurde. Unter dieser Voraussetzung ist die Klassifikation von (a, b) kein Klassifikationsfehler.

Möglichkeit 2: Klassifikationsgenauigkeit angepasst an Potenzmengen
Die Berechnung der Klassifikationsgenauigkeit ist aufgrund der Potenzmengen anzupassen. Sämtliche Potenzmengen der Klasse werden nicht als Klassifikationsfehler gewertet. Allerdings fallen hierbei Fehler des Klassifizierers nicht auf, weshalb dieser Messwert nicht realitätsnah genug ist.

Möglichkeit 3: Keine Potenzmengenklassen (Variante 2)
Hierbei entstehen zwei Probleme. Zum einen gibt es für jede Situation, in der zwei Klassen gleichberechtigt sind, mehrere Trainingsbeispiele, was unerwünscht ist. Zum anderen gehen Informationen verloren. Stehen insgesamt beispielsweise die Klassen a, b und c zur Verfügung und es wird die Potenzklasse (a, c) klassifiziert, enthält diese Potenzklasse die zusätzliche Information, dass in der aktuellen Situation zwischen a und c Unentschlossenheit herrscht, aber die Klasse b nicht zu wählen ist. Diese Information geht ohne die Verwendung von Potenzklassen verloren.

Im Weiteren soll Möglichkeit 1 zur Bestimmung des Fehlers des Naive-Bayes-Klassifizierers eingesetzt werden, da diese Klassifikationsgenauigkeit des Klassifizierers per definitionem widerspiegelt.

Beispiel 5.9 Klassifikationsgenauigkeit
Sei ein Konfliktset aus den Operationen x, y und z gegeben. Weiterhin stehen zur Auswahl aus diesen Operationen die Prioritätsregeln a, b und c zur Verfügung. Hierbei

[51] Siehe Abschnitt 3.2.

wählen die Regeln a und b beide die korrekte Operation x im Zustand s. Bei Potenzmengenklassen (Variante 3) wurde ein Beispiel in die Wissensbasis aufgenommen, das die Regelmenge (a, b) als korrekte Klassifikation enthält. Falls das Verfahren als Klassifikation für diese Instanz a oder b separat angibt, wird diese Klassifikation als Fehler gewertet. Obwohl die Klassen a und b dieselbe Operation wählen wie die Klasse (a, b), ist nach der Definition des wahren Fehlers nur die Klassifikation der Potenzmengenklasse (a, b) korrekt. Des Weiteren werden in der Praxis die Klassifikationen (a, b, d) und (a, b, e) meist dieselbe Operation wählen.[52] Eine solche Fehlklassifikation hat weniger Auswirkungen als eine Fehlklassifikation, in der eine gegensätzliche Prioritätsregel gewählt wird. Da die Klassen jedoch nominal skaliert sind, sind derartige Fehlklassifikationen nicht zu gewichten. □

Zusammenfassend ist festzuhalten, dass die erreichte Klassifikationsgenauigkeit, wie beschrieben, in einem starkem Zusammenhang mit dem Prozentsatz der richtig gewählten Operationen steht.[53] Jedoch entspricht er diesem nicht direkt. Im Allgemeinen wird der prozentuale Anteil von korrekten Auswahlentscheidungen im Konfliktset also höher sein als die Klassifikationsgenauigkeit.

5.3.4. Entwicklung einer Trainingskomponente

Damit der Klassifizierer gute Entscheidungen treffen kann, sind nicht nur aussagekräftige Merkmale und Steuerungsregeln (Klassen), sondern auch gute Trainingsbeispiele zwingend erforderlich. Da WBSGT die Lösungsgüte des Trainingsverfahrens approximiert, kann es bei korrekter Funktionsweise – der Klassifizierer macht kaum Fehler – maximal so gute Ergebnisse erzielen wie das Trainingsverfahren.[54] Nachfolgend werden zwei Methoden zur Generierung von Trainingsdaten entwickelt. Hierbei wird zwischen dem Training für Job-Shops und Flexible-Flow-Shops unterschieden, da jeweils unterschiedliche Trainingsverfahren eingesetzt werden. Zunächst wird das Training in Job-Shops im Detail dargelegt, anschließend das Training in Flexible-Flow-Shops. Hierbei wird nur auf die Unterschiede beim Trainingsverfahren eingegangen, da das grundsätzliche Vorgehen dem in Job-Shops entspricht.

[52] Im Allgemeinen wählen Klassifikationen, deren Prioritätsregelmengen sehr ähnlich sind, häufig die gleiche Operation.

[53] Siehe Abschnitt 2.2.3.1.

[54] Siehe Abschnitt 2.2.3.1. Dieses gilt für optimierende Trainingsverfahren. Werden die Trainingsbeispiele mit nicht-optimierenden Verfahren erzeugt und findet eine Vorauswahl der Trainingsdaten statt, dann kann das Lernverfahren besser werden als der nicht-optimierende Trainier.

5.3.4.1. Training in Job-Shops

Das Erstellen von Trainingsbeispielen für Job-Shops soll mittels eines Branch-and-Bound-Algorithmus, der optimale Lösungen generiert, durchgeführt werden.[55] Die für den Algorithmus zur Verfügung stehende Berechnungszeit muss begrenzbar sein, so dass der Algorithmus beispielsweise nach x Minuten terminierbar ist und das bis dahin beste gefundene Ergebnis zurückgibt. Zum Training wird das initiale Scheduling-Problem automatisch über eine Verteilung abgeändert und die so erzeugten Probleminstanzen vom Trainingsalgorithmus gelöst. Für die Änderung wird eine Normalverteilung[56] eingesetzt, damit die abgeänderten Probleme in der Grundstruktur gleich bleiben. Die Verteilungsfunktion verändert die einzelnen Operationszeiten der Probleminstanz. Dabei wird ein Mechanismus umgesetzt, der sicherstellt, dass zum einen Operationszeiten nicht ins Negative veränderbar sind und zum anderen *kleine* Operationszeiten gering geändert werden und *große* Operationszeiten viel. Damit bleibt der grundlegende „Charakter" des Problems erhalten.

Beispiel 5.10 Veränderung der Operationszeiten zum Training
Eine Operation mit einem Bearbeitungsdauer von 7 Zeiteinheiten wird nicht um 300 Zeiteinheiten verändert, sondern in einem Rahmen von 5-15 Zeiteinheiten. Wohingegen Operationen mit beispielsweise 732 Zeiteinheiten auch mit großen Werten abgeändert werden. Kleine Abwandlungen würden hier zu einer nicht ausreichenden Änderung der Probleminstanz führen. □

Die Initialisierung von WbsGT zum Training vollzieht sich folgendermaßen: Zu Beginn wird ein beispielhaftes Scheduling-Problem (Scheduling-Probleminstanz) geladen, von welchem anschließend mehrere Varianten erzeugt werden. Diese unterscheiden sich vom initialen Problem durch die veränderten Bearbeitungszeiten der Operationen.[57] Die Veränderungen des Problems werden wie dargestellt grundsätzlich durch prozentuale Abänderungen der ursprünglichen Werte herbeigeführt.[58] Anschließend werden mit dem Branch-and-Bound-Trainingsverfahren initiale Schedules berechnet.[59] So kann der initiale Schedule eine möglichst hohe Lösungsgüte erreichen. Er bietet Trainingsin-

[55] Wie beschrieben sind hier auch anderen Trainingsverfahren denkbar.
[56] Anderen Verteilungsfunktionen wie die Exponentialverteilung oder die Weibullverteilung sind ebenfalls denkbar.
[57] Ebenso sind z. B. durch Veränderungen der Jobanzahl auch Variationen der Problemgröße möglich.
[58] Mit den Parametern *mean* (μ), *variance* (σ) und *delete* (κ) sind die Problemvariationen kontrollierbar. Die Parameter μ und σ bilden Mittelwert und Varianz einer Normalverteilung, mittels welcher die Operationszeiten des Problems prozentual angepasst werden. Der *delete*-Wert legt fest, welcher prozentuale Anteil von Jobs in den variierten Problemen gelöscht wird.
[59] Da diese Berechnung vor dem Einsatz des WbsGT-Verfahrens, d. h. vor dem Beginn des Fertigungsprozesses erfolgt, ist es möglich, dieses aufwendige Scheduling-Verfahren zu verwenden.

stanzen für die Klassen der WBSGT-Konfiguration, mit denen dann der Naive-Bayes-Klassifizierer in WBSGT trainiert wird. Die Klassifikationsgenauigkeit lässt sich durch zusätzliche Trainingsinstanzen erhöhen. Diese werden mittels der Lösung der Varianten der initialen Probleminstanz erzeugt.

Weil die Trainingsphase (Phase I) von der Betriebsphase (Phase II) getrennt[60] ist, gelten für den Aufbau der Wissensbasis nicht die für die Betriebsphase charakteristischen strengen Zeitbeschränkungen. Dennoch können aufgrund des exponentiell ansteigenden Zeitbedarfs für optimale Lösungen auch in der Trainingsphase ab einer bestimmten Größe Benchmark-Probleme nicht mehr optimal gelöst werden. In derartigen Fällen wird die beste in einer vorgegebenen Zeit gefundene Lösung des optimierenden Trainingsverfahrens zurückgegeben oder die Trainingsdaten von einem heuristischen Verfahren erzeugt.[61]

Nachdem die Referenzlösungen erstellt sind, werden diese mit der Steuerungs- und der Lernkomponente im Lernmodus durchlaufen. WBSGT bildet die Referenzlösungen nach und akquiriert dabei Trainingsbeispiele. Bei der WBSGT Ausführung ergeben sich Konfliktmengen, aus denen eine Operation zu wählen ist.[62] Durch die Referenzlösung ist die optimale bzw. beste Auswahl aus der Konfliktmenge bekannt. Für die Wissensbasis werden der über die Merkmale abstrahierter Systemzustand und die Operationsauswahl der Referenzlösung betrachtet. Die Prioritätsregeln, welche in diesem Systemzustand dieselbe Operation auswählen wie in der Referenzlösung, und der Zustand – dieser wird berechnet – werden in der Lernkomponente (deren Wissensbasis) als Trainingsbeispiele abgelegt. Anschließend wird diese Operation fest in den Schedule eingeplant und die nächste Konfliktmenge aufgebaut. Abbildung 5.13 zeigt das Durchlaufen einer Beispiellösung. Da es hier im letzten Teilschritt zu keinem Konflikt kommt, wird auch kein Trainingsbeispiel in der Wissensbasis abgelegt. Kann keine der Lernkomponente zur Verfügung stehenden Prioritätsregeln die Operation der Referenzlösung auswählen, wird ebenfalls kein Trainingsbeispiel abgelegt, die Operation eingeplant und das Verfahren fortgeführt.

5.3.4.2. Training in Flexible-Flow-Shops

Da das Scheduling in Flexible-Flow-Shops noch komplexer ist als in Job-Shops, erfordert die Generierung von Trainingslösungen mit optimierenden Verfahren mehr Zeit.

[60] Siehe Abschnitt 5.1.

[61] Eine zusätzliche, von einem lokalen Suchverfahren optimierte Lösung kann ebenfalls in Betracht gezogen werden. WBSGT bietet hierfür die Möglichkeit einer Verbesserung mit einer Nachbarschaftssuche.

[62] Siehe Abschnitt 5.2.

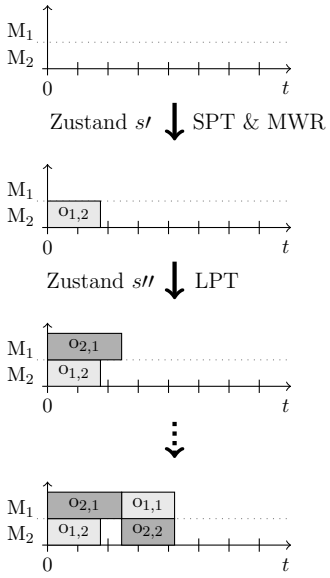


Abbildung 5.13.: Training der Lernkomponente

Hierbei kann selbst die zur Verfügung stehende Zeit vor dem Beginn des Fertigungsprozesses nicht ausreichend sein, um genügend Trainingsbeispiele zu generieren. Zusätzlich könnte beispielsweise durch das optimierende Lösen der einzelnen Stufen der Gesamtzusammenhang des Fertigungssystems verloren gehen bzw. würde nicht genau genug abgebildet. Aus diesem Grund soll für das Training in Flexible-Flow-Shops auf ein nicht optimierendes Trainingsverfahren (Ablaufsimulation) zurück gegriffen werden. Hiermit kann zusätzlich die Qualität des dezentralen Einsatzes von WBSGT unter weitgehend realistischen Voraussetzungen durch Simulation evaluiert werden. Für beide Aufgaben soll ein einheitliches Simulationssystem verwendet werden. Sehr gut geeignet für diese Zwecke ist der Ablaufsimulator d³FACT insight[63], da dieser in der Lage ist, in kurzer Zeit, eine Vielzahl von alternativen Zufallslösungen zu generieren und ohne großen Aufwand für das vorliegende Problem angepasst werden kann.

[63]Beim Ablaufsimulator d³FACT insight handelt es sich um eine in der Programmiersprache Java realisierte, ereignisbasierte Modellierungs- und Simulationsumgebung, die sich durch eine modulare Struktur auszeichnet. d³FACT insight besteht aus einer Komponente zur zweidimensionalen Modellierung, einem Simulatorkern und einer zweidimensionalen sowie einer dreidimensionalen Visualisierungskomponente. Um die im Rahmen dieser Arbeit an das Simulationssystem gestellten Anforderungen zu erfüllen, braucht lediglich das Kernelmodul von d³FACT insight eingesetzt zu werden. Es dient in diesem Zusammenhang als diskreter, ereignisgesteuerter Simulator der Materialflüsse. Zur Beschreibung der Fertigungsumgebung wird in der Kernelkomponente eine Datei im Format der Auszeichnungssprache XML (E**x**tensible **M**arkup **L**anguage) verwendet. Darin sind die in der Umgebung vorhandenen Arbeitssysteme mit ihrer Gruppierung zu Fertigungsstufen und ihre Beziehungen zueinander zu beschreiben. Zudem sind mit Hilfe der XML-Datei und darin enthaltenen Java-Quellcode-Fragmenten die in der Umgebung zu fertigenden Aufträge zu definieren. Konkret wird in der Datei häufig festgelegt, dass die Art der Aufträge aus einer weiteren Datei einzulesen ist (vgl. Laroque, 2007).

Das Training in Flexible-Flow-Shop-Umgebungen funktioniert bis auf zwei Abwandlungen genauso wie in Job-Shop-Umgebungen. Zum einen wird bei dem Zwei-Klassifizierer-System in Konfliktsituationen neben den Trainingsbeispielen für die Auswahl eines Auftrags zusätzlich ein Trainingsbeispiel für die Auswahl der Maschinen abgelegt. Im Fall des Ein-Klassifizierer-Systems entfällt das Trainingsbeispiel für die Maschinenauswahl. Zum anderen ist eine Auswahl aus den Beispiellösungen zu treffen, bevor diese WBSGT für den Aufbau der Wissensbasis übergeben werden. Da bei der zufälligen Lösung einer Probleminstanz durch einen Ablaufsimulator zufällige Lösungen generiert werden, ist die Trainingsmenge auf die guten Ergebnisse zu beschränken. Hierbei wird über einen Parameter gesteuert, welcher prozentuale Anteil der gesamten Beispiellösungen für das Training verwendet werden soll.[64]

Im Gegensatz zu Job-Shops werden in Flexible-Flow-Shops mehrere WBSGT-Instanzen benötigt, d. h.WBSGT wird auf jeder Fertigungsstufe eingesetzt. Dabei sind zwei Möglichkeiten zu unterscheiden. Zum einen kann ein globaler Klassifizierer trainiert werden und von diesem werden Instanziierungen auf den jeweiligen Stufen eingesetzt. Zum anderen kann für jede Stufe ein eigener Klassifizierer trainiert werden. Bei der ersten Möglichkeit werden zwar die Eigenschaften des gesamten Systems berücksichtigt, jedoch nicht die Eigenheiten der einzelnen Stufen. Dies führt insbesondere zu Problemen, wenn die Stufen unterschiedlich sind, d. h. unterschiedliche Maschinenzahl auf den Stufen etc. Der Klassifizierer müsste sehr generalisiert sein, damit er auf jeder Stufe eingesetzt werden kann. Aus diesem Grund wird die Variante mit einem eigenen Klassifizierer für jede Stufe angewendet. Hierbei werden die inhärenten Eigenschaften der jeweiligen Stufe berücksichtigt und globale Systemeigenschaften lassen sich über Merkmale abbilden.[65]

5.3.4.3. Adaption des Klassifizierers

Weil Fertigungssystemen eine große Dynamik inhärent ist, muss WBSGT während des Einsatzes im Produktionsprozess fortwährend adaptiert werden. Während Aufträge auf Basis einer vorhergehenden Planung abgearbeitet werden, sind bereits neue Aufträge anzunehmen und einzuplanen. Diese Dynamik wird von WBSGT bislang nicht vollständig abgebildet. Der Trainingsprozess ist nach dem initialen Training abgeschlossen und wird während der Anwendung nicht fortgeführt. Das Verfahren wird nicht adaptiert. Nachfolgend wird ein Konzept zur Adaption des Verfahrens entwickelt.

[64] Anstelle der $x\%$ besten Lösungen als Auswahlkriterium für die Trainingsmenge, ist beispielsweise auch ein Parameter denkbar, der die Auswahl der Trainingsbeispiele als prozentualen Bereich um die beste Lösung aller Beispiele beschreibt.

[65] Siehe Abschnitt 5.3.2.

Die grundsätzliche Annahme ist, dass aus dem schon bekannten Teil des Problems, Schlüsse auf das Gesamtproblem gezogen werden können. Daher kann mittels der Adaption eine Verbesserung des Verfahrens erreicht werden. Wenn ein Teil des Problems bekannt ist, kann WBSGT stärker auf dieses spezielle Problem angepasst werden. Zur Adaption müssen Eigenschaften des Schedules gefunden werden, die von dem bereits gelösten Teil bedingt werden bzw. die über die gesamte Lösung stabil bleiben. Werden Eigenschaften zu Beginn der Lösung beobachtet, sind diese für den Rest des Problems vorhersagbar und WBSGT kann entsprechend anpasst werden. Es werden zwei Arten der Adaption, die auf unterschiedlichen Eigenschaften beruhen, behandelt:

- Die Anpassung der Wissensbasis während der Laufzeit mit neuen Trainingsbeispielen aus dem schon gelösten Teil des Problems.
- Die Auswahl eines guten Prioritätsregelsatzes auf Grundlage der Lösungsgüte der Regelsätze bei dem schon gelösten Teil des Problems.

Adaption der Wissensbasis

Bei der Adaptionsmethode zur Anpassung der vorhandenen Wissensbasis wird das Scheduling-Problem mit WBSGT gelöst. Parallel zur Abarbeitung des entwickelten Schedules wird in einer Rückschau auf bereits abgearbeitete Teile des Problems eine zeitoptimale Trainingslösung erstellt.[66] Die Konfliktauflösungen dieser Trainingslösung werden als zusätzliche Beispiele in die Wissensbasis eingefügt und können damit zukünftige Entscheidungen des Klassifizierers beeinflussen. Die hierbei zugrunde liegende Hypothese lautet: Gute Konfliktauflösungen einer Probleminstanz korrelieren stärker mit denen von anderen Teilen der Probleminstanz als gute Konfliktauflösungen von verschiedenen Problemen (wie beim Training). Es ist also davon auszugehen, dass die Ähnlichkeit von verschiedenen Teilen eines einzelnen Problems – wie beim praktischen Verfahrenseinsatz üblich – zueinander höher ist als die Ähnlichkeit von Scheduling-Problemen generell. Die speziell auf das zu lösende Problem abgestimmte Wissensbasis kommt automatisch zum Einsatz, wenn durch Veränderungen im Fertigungssystem eine Aktualisierung der bestehenden Planung mit WBSGT notwendig wird. Zusätzlich kann die Wissensbasis bei großen Problemstellungen genutzt werden, um die Planung für den noch nicht bearbeiteten Teil des Schedules zu optimieren. Demzufolge können Beispiele für optimale Konfliktauflösungen des bisher bekannten oder noch ausstehenden Teilproblems die Wissensbasis positiv beeinflussen. Durch die Abstimmung der Wissensbasis auf das zu bearbeitende Problem wird die Entfernung der Wissensbasis von

[66] Als zeitoptimal werden die nach dem Abbruch des Trainingsverfahrens bisher besten Lösungen bezeichnet.

den zu lösenden Problemen effektiv verhindert. Verändern sich das Fertigungssystem oder die Charakteristik der Aufträge im Laufe der Zeit, wird die Wissensbasis durch die ständigen Aktualisierungen mit neuen Beispielen immer an die veränderten exogenen Bedingungen angepasst.[67]

Adaption der Potenzmenge

Bei der Auswahl der geeigneten Potenzregelmenge lautet die zugrunde liegende Hypothese: Die Struktur eines Scheduling-Problems ist insoweit konsistent, als dass ein Prioritätsregelsatz welcher für einen (früheren) Teil des Problems gut geeignet ist, mit hoher Wahrscheinlichkeit auch für das gesamte Problem gut geeignet ist. Dies entspricht den in Abschnitt 3.1.2 dargelegten Aussagen über die grundsätzlich gute Eignung von einzelnen Prioritätsregeln. In diesem Fall kann in einer Rückschau für den bereits gelösten Teil des Problems der optimale Prioritätsregelsatz bestimmt werden, um diesen für die weitere Bearbeitung des Problems zu priorisieren.

5.4. Rückkopplung in die simulative Trainingskomponente

Nachfolgend wird das Konzept zur Integration des entwickelten WBSGT-Verfahrens in einen Ablaufsimulator (hier d^{3}FACT insight) dargelegt, um diesen mit WBSGT zu erweitern. Dabei stellt 5.4.1 die eigentliche Integration von WBSGT in den Simulator vor. Das Konzept zur simulativen Informationserweiterung in Konfliktsituationen wird abschließend erläutert.

5.4.1. Integration als Steuerung in die Ablaufsimulation

WBSGT wird als Steuerungsmodul in d^{3}FACT insight[68] verwendet. Durch die Funktion als Steuerungsmodul im Ablaufsimulator wird dieser um eine bisher nicht vorhandene Funktionalität erweitert. WBSGT ermöglicht der Ablaufsimulation, qualifizierte Ablaufplanungs- und Steuerungsentscheidungen zu treffen und nicht nur generelle Szenarien zu entdecken. Damit wird die bisher zufällige Anwendung von Auswahlregeln zur

[67] Das Löschen von schlechten bzw. veralteten Trainingsbeispielen aus der Wissensbasis ist ebenfalls denkbar.

[68] Hierbei dient der Ablaufsimulator d^{3}FACT insight als Beispielumgebung, da dieser bereits für das Training in Flexible-Flow-Shops eingesetzt wird. Das WBSGT-Verfahren ist grundsätzlich auch in andere Ablaufsimulatoren integrierbar.

Steuerung des Fertigungssystems überwunden. Hierzu wird in jedem Steuerungspunkt des Fertigungssystems eine Instanz des WBSGT-Verfahrens integriert. Hierbei sind mit der Simulation Fertigungssysteme realitätsnah nachbildbar, womit beispielsweise Maschinenausfälle vor dem realen Einsatz untersucht werden können, um etwaige Lösungen zu trainieren. Kommt es jetzt zu entscheidungsrelevanten Situationen, wird die Regel zur Steuerung des Materialflusses, d. h. der Bearbeitungsreihenfolge direkt von WBSGT ausgewählt. Dabei können sämtliche Entscheidungen direkt abgearbeitet werden, wodurch ein Ablaufplan entsteht, der von der Simulation durchgeführt wird. Kommt es zu neuen Ereignissen, ist der alte Plan zu verwerfen und neue Entscheidungen werden getroffen.

Da WBSGT als lokale Steuerungsmethode in den Ablaufsimulator integriert wird, beispielsweise vor jeder Fertigungsstufe eines Flexible-Flow-Shops zur Verteilung der Arbeitsaufträge, kommt es zu lokal betrachtet guten Entscheidungen. Vom globalen Standpunkt aus gesehen, können dieses jedoch schlechte Entscheidungen sein. Dieser Aspekt ist der Komplexität der Fertigungssysteme sowie der zeitkritischen Entscheidungen zuzuschreiben. Allerdings kann den Entscheidungen über die situationsbeschreibenden Merkmale ein globaler Charakter gegeben werden.[69] Ist beispielsweise auf der Stufe st eine Entscheidung zu treffen und existieren Merkmale, welche die Materialverfügbarkeit auf der Stufe $st + 1$ beschreiben, können schlechte Entscheidungen verhindert werden. Genau dann, wenn die eigentlich beste Entscheidung auf Stufe st Auftrag A wäre, jedoch für diesen Auftrag auf der nächsten Bearbeitungsstufe Materialen fehlen. Diese Berücksichtigung des technologischen Vor- und Nachbereichs einer Entscheidungssituation über Merkmale kann in der Simulation detailliert (realitätsnah) untersucht werden. Dabei werden in der aktuellen Ausprägung der WBSGT-Steuerungskomponente die bekannten Aufträge auf einer, der aktuellen Fertigungsstufe vorgelagerten Stufe mit in die Steuerungsentscheidung einbezogen. Eine derartige Vorschau kann der technologischen Blickrichtung zugeordnet werden. Das bedeutet zum Entscheidungszeitpunkt t_0 können alle Informationen berücksichtigt werden, die sicher sind. Hierunter fallen genau die Aufträge, die auf den vorgelagerten Stufen bereits in Bearbeitung sind und für welche der Fertigstellungstermin, der Bereitstellungstermin auf der aktuellen Stufe, bekannt ist. Kommt es hierbei nach der getroffenen Entscheidung zu Störungen auf der vorgelagerten Stufe, wird der aktuelle von der Simulation abgearbeitete Plan verworfen und ein neuer generiert. Alternativ ist es möglich, dass kein Plan generiert wird, sondern mit der WBSGT-Steuerungskomponete jeweils nur einzelne Entscheidungen zur Steuerung getroffen werden.

[69] Siehe Abschnitt 5.3.2.

5.4.2. Informationserweiterung mittels Ablaufsimulation

Um in Konfliktsituationen weitere Informationen für WBSGT zur Entscheidung zu generieren, ist mittels der Ablaufsimulation in einem gewissen zeitlichen Rahmen eine Vorschau durchführbar. Dabei beschreibt der zeitliche Rahmen, d. h. die Länge der zeitlichen Vorschau, eine wichtige Größe. Wie beschrieben, können zum Entscheidungszeitpunkt t_0 beispielsweise die Informationen über schon in Bearbeitung befindliche Aufträge auf den direkten Vor- bzw. Nachfolgestufen mit berücksichtigt werden. Aufträge, für die noch nicht entschieden ist, auf welcher Maschine sie bearbeitet werden, sind bei dieser Betrachtung nicht zu berücksichtigen. Hierbei ist jetzt der Ablaufsimulator als Werkzeug zur Informationserweiterung zur zeitlichen Vorschau einsetzbar.

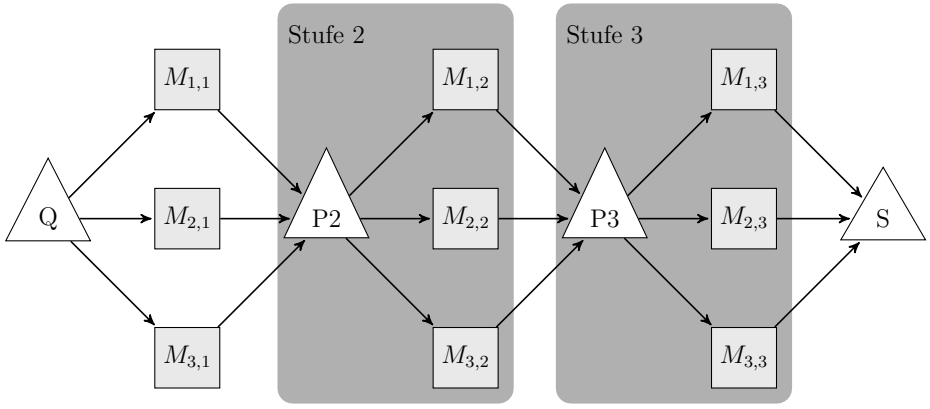


Abbildung 5.14.: Schematische Darstellung einer Flexible-Flow-Shop-Umgebung

Abbildung 5.14 zeigt eine beispielhafte Flexible-Flow-Shop-Umgebung bestehend aus drei Stufen mit jeweils drei parallelen unverwandten Maschinen. Ist auf der dritten Stufe im Puffer $P3$ zum Zeitpunkt t_0 eine Entscheidung zu treffen, kann mittels Simulation eine Vorschau durchgeführt werden. Damit ist zu eruieren, von welcher der Maschinen ($M_{1,2}$; $M_{2,2}$ oder $M_{3,2}$) die in Stufe 2 noch im Puffer $P2$ befindlichen Aufträge nach dem Zeitpunkt t_0 bearbeitet werden. Da die Maschinen unverwandt sind, ergibt sich erst aus der konkreten Zuordnung eines Auftrags zu einer Maschine die Fertigstellungszeit des Auftrags auf Stufe 2 und damit die Bereitstellungszeit auf Stufe 3. Gleiches gilt für uniforme Maschinen. Diese Informationen können dann in die Entscheidung der WBSGT-Instanz in $P3$ einfließen. Somit sind beispielsweise für einen Eilauftrag, der zwar auf Stufe 2 vorhanden, aber noch nicht in Bearbeitung ist, bereits Kapazitäten auf der aktuellen Stufe 3 freizuhalten. Es werden passende Entscheidungen bei der Zuordnung von Aufträgen und Maschinen auf der dritten Stufe getroffen.

Es sind bei einer derartigen Vorschau gewisse Aspekte zu beachten. So entstehen mit zunehmender zeitlicher und technologischer Entfernung von der aktuellen Entschei-

dungsposition (hier z. B. Stufe 3) immer größere Unsicherheiten. Informationen über den direkten Vorgängen und Nachfolger sind noch als einigermaßen sicher einzustufen, da hier die Anzahl von möglichen Alternativen und Folgeentscheidungen gering ist. Die Informationen von weiter entfernten Stufen hingegen, werden mit zunehmender Entfernung immer unsicherer. Aufträge von Stufe 1 müssen noch Stufe 2 durchlaufen bevor sie von Stufe 3 zu bearbeiten sind. Entscheidungen führen demgemäß zu Folgeentscheidungen. Damit werden die Unsicherheiten der Stufen 1 und 2 verknüpft, was zu sehr unsicheren Information über die vorgelagerte Stufe 1 bei Entscheidungen auf der dritten Stufe führt. Von der Ausgangssituation einer Entscheidung kann ein Baum der Folgeentscheidungen aufgebaut werden. Dieser wird aufgrund der möglichen alternativen Entscheidungen sehr schnell tief und breit. Aus diesem Grund müssen bei der Informationserweiterung von Entscheidungen mittels Ablaufsimulation Grenzen festgelegt werden, mit denen ein Rahmen noch signifikanter Information erstellt wird. Der Baum von Alternativen ist zu beschneiden. Nur damit kann die Simulation eingesetzt werden, ohne den Entscheidungsprozess zu verlangsamen oder das Fertigungssystem zu blockieren. Insgesamt kann bei einer Entscheidung im laufenden Fertigungsprozess also nur ein kleiner Vorschauhorizont simulativ betrachtet werden.[70]

[70](Mahajan, 2007) definiert diesen Horizont beispielsweise über die mittlere Rechenzeit des von ihm entwickelten Verfahrens.

6. Evaluierung

Innerhalb des folgenden Kapitels wird das entwickelte Lösungsverfahren WBSGT evaluiert, um dessen Zweckmäßigkeit darzulegen. Hierzu werden die in den einzelnen Experimenten zu Job-Shops und Flexible-Flow-Shops erreichten Ergebnisse aufgezeigt und die generelle Funktionsfähigkeit der Lernkomponente dargelegt. Anschließend wird die Leistung des entwickelten Verfahrens mit optimierender Trainingsmethode in Job-Shop erläutert. Die Ergebnisse der Simulationsintegration und der simulativen Trainingsmethode bei dezentralem Verfahrenseinsatz in Flexible-Flow-Shops beenden das Kapitel.

6.1. Evaluierungsaufbau

Zur Evaluierung wurde WBSGT mit verschiedenen Verfahren verglichen. Dazu wurden als externe Methoden zur Generierung von Vergleichslösungen in Job-Shops die in der Library of Scheduling Algorithms (LiSA) umgesetzten Branch-and-Bound- und Shifting-Bottleneck-Verfahren verwendet.[1] Hier ist die zur Verfügung stehende Rechenzeit für die Verfahren begrenzbar. Die simulativen Trainingsbeispiele für Flexible-Flow-Shops wurden mit dem Ablaufsimulator d^{3}FACT insight[2] generiert, der ebenfalls als Vergleichsverfahren eingesetzt wurde. Als Bibliothek für den verwendeten Naive-Bayes-Klassifizierer wurde Weka verwendet.[3] Im Einzelnen wird das entwickelte WBSGT-Verfahren anhand der nachfolgend aufgeführten Punkte evaluiert.

[1] Vgl. (Bräsel et al., 2003). Insgesamt stehen in der Scheduling-Umgebung LiSA zwei Branch-and-Bound-Implementierungen zur Verfügung. Für das Training in Job-Shops kam das Verfahren nach *Brucker et al.* zum Einsatz (vgl. Brucker et al., 1994). Für das Shifting-Bottleneck-Verfahren ist in LiSA die Version von (Adams et al., 1988) umgesetzt.

[2] Vgl. (Laroque, 2007).

[3] Vgl. (Witten und Frank, 2005).

6.1.1. Lösungsgeschwindigkeit

Die Ablaufplanung und -steuerung dient dazu, die Kosten einer Produktion zu senken.[4] Je schneller die Fertigung durchgeführt werden kann, desto eher steht das Fertigungssystem wieder zur Verfügung. Von Bedeutung für den Einsatz des Verfahrens in der zeitkritischen Ablaufplanung und -steuerung der Produktionsprozesse ist die Lösungsgeschwindigkeit, die erforderliche Rechenzeit des Verfahrens bis zur Generierung einer Lösung.[5]

6.1.2. Lösungsapproximation

Die Annahmen, WBSGT ist in der Lage, andere Lösungsverfahren – Trainingsverfahren – zu approximieren, ist zu evaluieren. Aus diesem Grund sind hierzu die folgenden Experimente durchzuführen:

Experiment 1: Allwissender Klassifizierer
Bei dem Experiment mit dem allwissenden Klassifizierer wird untersucht, was für Lösungen mit WBSGT möglich sind, wenn der Klassifizierer keine Fehler macht.

Experiment 2: Klassifikationsgenauigkeit
Bei dem Experiment zur Klassifikationsgenauigkeit wird untersucht, wie sich die Klassifikationsgenauigkeit des in WBSGT eingesetzten Klassifizierers bei Benchmark-Problemen[6] verhält.

Experiment 3: Makespan
Das Experiment zur Untersuchung des Makespans widmet sich dem Vergleich mit anderen Lösungsverfahren. Ziel ist die Evaluierung, wie die WBSGT-Lösungsgüte mit seinen unterschiedlichen Konfigurationen, d. h. mit verschiedenen Prioritätsregelsätzen bei der Lösung von Benchmark-Problemen, gegenüber den Vergleichsverfahren ist.

[4] Siehe Abschnitt 2.2.2.3.

[5] Allerdings verursacht die Ablaufplanung und -steuerung zunächst selbst Kosten, die es zu beurteilen gilt, um dessen Einsatz abzuwägen. Hierbei zu berücksichtigende Faktoren sind beispielsweise die Installation von Sensorik, um den Zustand des Systems zu messen, als auch Mechanismen, welche die Produktion für die Ausführung eines neuen Schedules umstellen. Derartige Aspekte werden hier jedoch nicht mit berücksichtigt, da sie im gewissen Maße unabhängig vom eigentlichen Steuerungsverfahren sind.

[6] Benchmark- und Scheduling-Problem werden synonym verwendet.

6.1.3. Simulatives Training

Bei der Evaluierung des simulativen Trainings wird nicht mit einem optimierenden Verfahren trainiert, sondern mit einem Ablaufsimulator. Hierbei sind die Trainingsbeispiele in der Regel nicht optimal, sondern basieren auf zufälligen Entscheidungen. Die zugrundeliegende Hypothese hierbei lautet: WBSGT führt zu besseren Lösungen als die einfache Anwendung der immer gleichen Auswahlregeln oder des simulativen Trainingsverfahrens bei zufälliger Regelauswahl. Letztere Aussage steht auf den ersten Blick der allgemeinen Hypothese, dass das WBSGT-Verfahren nicht zu besseren Ergebnissen führen kann als das Trainingsverfahren, entgegen. Jedoch bezieht sich diese Aussage auf Methoden, mit denen immer gleiche Lösungsgüten erzielt werden. Vom Ablaufsimulator werden hingegen zufällige Lösungen bei der mehrmaligen Lösung derselben Probleminstanz generiert. Aus diesem Grund sind bei der Anwendung der Simulation mit WBSGT als Steuerungsmodul bessere Lösungen im Sinne der Zielfunktion zu erwarten als vom simulativen Trainer.

6.1.4. Trainingsmenge

Der im WBSGT-Verfahren eingesetzte Klassifizierer verwendet Wissen, welches das Verfahren aus Beispiellösungen für eine Probleminstanz gewinnt. Ziel dieses Experiment ist die Untersuchung der eingesetzten Trainingsbeispiele.[7] Dabei ist zum einen zu untersuchen, welche Lösungsgüten in Abhängigkeit der Größe der Trainingsmenge erreicht werden. Zum anderen soll untersucht werden, ob bei steigender Anzahl der Simulationsdurchläufe und gleichzeitig steigendem Faktor zur Generierung der Trainingsmenge (damit bessere Beispiele in der Trainingsmenge) bessere Ergebnisse erzielbar sind. Der erste Punkt betrachtet die grundsätzliche Größe der Trainingsmenge, der zweite die Qualität der Trainingsbeispiele.

[7] Hierzu wird das Zwei-Klassifizierer-System mit simulativem Training verwendet.

6.2. Evaluierung der zentralen Verfahrensgüte in Job-Shops

6.2.1. Benchmark: Lösungsgeschwindigkeit

Wie beschrieben, ist die erforderliche Rechenzeit (Lösungsgeschwindigkeit) zur Lösungsfindung ein wichtiges Kriterium für den Einsatz eines Verfahrens zur Ablaufplanung und -steuerung in zeitkritischen Fertigungsumgebungen.[8] Eine beispielhafte Übersicht der durchschnittlichen Rechenzeiten einzelner Lösungsverfahren gibt Tabelle 6.1.

Initiales Problem	Branch-and-Bound	Shifting-Bottleneck	WbsGT
FT06 (6×6)	$<$ 1 Sek.	$<$ 1 Sek.	$<$ 1 Sek.
FT10 (10×10)	$<$ 3 Sek.	$<$ 1 Sek.	$<$ 1 Sek.
LA21 (15×10)	ca. 20 Min.	ca. 70 Sek.	$<$ 1 Sek.
LA26 (20×10)	$>$ 60 Min.	ca. 10 Min.	$<$ 1 Sek.
LA27 (20×10)	$>$ 60 Min.	ca. 15 Min.	$<$ 1 Sek.
SWV01 (20×10)	$>$ 60 Min.	ca. 21 Min.	$<$ 1 Sek.

Tabelle 6.1.: Durchschnittliche Rechenzeiten der Lösungsverfahren[9]

Für die Probleminstanzen FT6 und FT10 sind alle aufgelisteten Scheduling-Verfahren geeignet. Bei dieser Größe der Probleminstanzen sollte ein Branch-and-Bound-Algorithmus eingesetzt werden, da dieser die optimale Lösung liefert. Das WbsGT-Verfahren ist hier im Nachteil, da es zusätzlichen Aufwand für das Training benötigt. Die Instanz LA21 hat eine weit größere Komplexität und der Rechenaufwand eines Branch-and-Bound-Verfahrens ist zu hoch für die knappen Zeitbeschränkungen. Hier ist WbsGT mit seiner geringen Rechenzeit geeigneter. Der zusätzliche Aufwand des Trainings wird vor die Produktion verlagert. Gleiches gilt für die Instanzen LA26, LA27 und SWV01, wo auch das Shifting-Bottleneck-Verfahren lange Berechnungszeiten erfordert und das Branch-and-Bound-Verfahren innerhalb einer Stunde keine optimale Lösung findet.

6.2.2. Benchmark: Lösungsapproximation

Nachfolgend werden die drei Experimente *allwissender Klassifizierer*, *Klassifikationsgenauigkeit* und *Makespan* zur Lösungsapproximation dargelegt. Dabei wird das Expe-

[8] Siehe Abschnitt 2.2.2.

[9] Zur ausführlichen Darstellung der Benchmark-Probleme FT10, LA21, LA26, LA27 und SWV01 siehe Anhang A.3. Für das FT06-Problem siehe Tabelle 6.2.

riment zum *Makespan* exemplarisch an verschiedenen Benchmark-Problemen durchgeführt.

Benchmark: Allwissender Klassifizierer

Einen Test im Hinblick auf die grundsätzliche WbsGT-Lösungsqualität liefert das Experiment mit dem so genannten allwissenden Klassifizierer.[10] Es gilt die Hypothese zu evaluieren, dass eine hohe Klassifikationsgenauigkeit bei der Auswahl der Alternativen aus der Konfliktmenge zu einem guten Ergebnis führt.[11] Das bedeutet: Gäbe es einen genauen Klassifizierer für WbsGT, wäre seine Lösungsgüte hoch. Hier stellt sich jetzt die Frage, ob ein Klassifizierer wirklich die Güte des erstellten Schedules anhebt, wenn er in den meisten Konflikten die korrekte Prioritätsregel klassifiziert, welche auch die Operation wählen kann, die ein optimales Verfahren als nächstes einplanen würde. Der Zusammenhang zwischen Klassifikationsgenauigkeit und Güte eines Schedules ist auf Korrelationen zu untersuchen. Diese Untersuchung soll im Folgenden exemplarisch an einem Job-Shop-Problem durchgeführt werden. Dabei kommen keine Situationsmerkmale zum Einsatz, da diese das Ergebnis verfälschen würden. Ziel dieses Experiments ist es weiterhin, die erreichbare Lösungsgüte einer Regelmenge in Abhängigkeit der Klassifikationsgenauigkeit zu untersuchen. Da es schwierig ist, einen Klassifizierer mit einer konkreten Klassifikationsgenauigkeit zu erstellen, wird in jedem Konflikt die optimale Alternative berechnet. Damit kennt der Klassifizierer zu jedem Zeitpunkt die korrekte Entscheidung; die als nächstes auszuwählende Operation. Diese wird von einer der zur Verfügung stehenden Prioritätsregeln mit einer Wahrscheinlichkeit ausgewählt, die der gewünschten Klassifikationsgenauigkeit entspricht. Kann keine der vorhandenen Prioritätsregeln die optimale Operation auswählen, wird per Zufall mit einer der verwendeten Prioritätsregeln ausgewählt. Die Berechnung einer optimalen Alternative in einem Konflikt erfolgt durch die Anwendung eines optimalen Scheduling-Verfahrens, hier dem Trainingsverfahren.[12] Aus einem optimalen Schedule für eine Scheduling-Probleminstanz ergibt sich die optimale Auflösung aller Konflikte. Diese besteht darin, die Operationen in

[10] Bei dem allwissenden Klassifizierer kann die Klassifikationsgenauigkeit vorgegeben werden. Für das Experiment wurde diese Genauigkeit sukzessive in 5 Prozent Schritten – die Schrittgröße ist frei wählbar – von 0 auf 100 Prozent erhöht. Von gänzlichen Zufallsentscheidungen des Klassifizierer, der Klassifizierer wählt zufällig eine der zur Verfügung stehenden Prioritätsregeln, bis hin zur korrekten Klassifikation. Wird eine korrekte Klassifikation der Prioritätsregeln gewünscht, kann es dennoch vorkommen, dass mit den zur Verfügung stehenden Regeln die Auswahl der korrekten Operation aus der Konfliktmenge unmöglich ist. In derartigen Fällen wählt der allwissende Klassifizierer ebenfalls zufällig aus der Menge der zur Verfügung stehenden Regeln.

[11] Siehe Abschnitt 2.2.3.1.

[12] Hierzu wird die allgemeine Implementierung des in LiSA umgesetzten Branch-and-Bound-Algorithmus verwendet, da dieser Bereitstellungszeiten (r_i) unterstützt (vgl. Bräsel et al., 2003).

der Reihenfolge wie in dem optimalen Schedule einzuplanen.[13] Die nicht-optimale Auswahl einer Konfliktalternative ist zufällig. Durch eine nicht-optimale Auswahl weicht der berechnete Schedule vom optimalen ab. Eine Abweichung vom optimalen Pfad,[14] wie in Abbildung 6.1 dargestellt, macht die Berechnung des neuen Schedules erforderlich, der unter allen noch möglichen Schedules, welche die nicht-optimale Auswahl enthalten, der Beste ist.

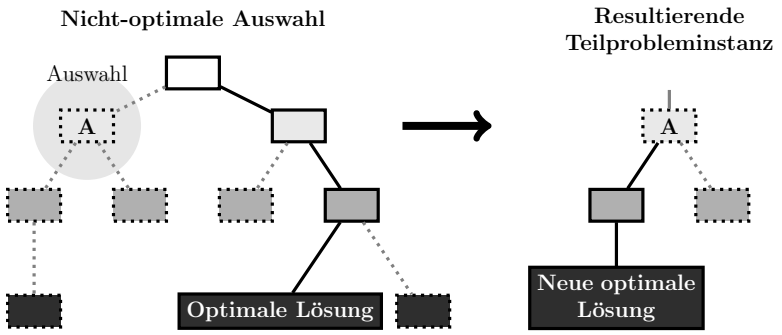


Abbildung 6.1.: Probleminstanz bei nicht-optimaler Einplanung einer Operation

Anders formuliert resultiert aus jeder nicht-optimalen Einplanung durch den Klassifizierer eine neue Teilprobleminstanz, die eine eigene neue optimale Lösungsgüte besitzt. Abbildung 6.1 illustriert dieses. Sind bereits Operationen eingeplant, welche die Einplanung ihrer Nachfolger verzögern, ist dieses zu berücksichtigen, indem für die Aufträge der neuen Instanz Bereitstellungszeiten berechnet werden. Der Fertigstellungszeitpunkt der zuletzt eingeplanten Operation i des Auftrags j wird seine neue Bereitstellungszeit $r_i = d(o_{j,m})$. Alle vorangehenden Operationen von j werden verworfen. Damit wird das ursprüngliche $J||C_{max}$-Problem zu einem $J|r_i|C_{max}$. Die Berechnung einer neuen optimalen Lösung für die resultierende Teilprobleminstanz, bei einer Abweichung von der initialen optimalen Lösung, ist aufwendig und benötigt unter Umständen lange Rechenzeiten. Darum wurde für das Experiment mit dem allwissenden Klassifizierer das noch gut lösbare *Fisher-Thompson* (FT06) Benchmark-Problem verwendet. Das FT06-Problem ist in Abbildung 6.2 dargestellt. Die *or*-Spalten beschreiben die einzuhaltende Bearbeitungsreihenfolge eines Auftrags (j) und die *pt*-Spalten die Bearbeitungszeiten auf der jeweiligen Maschine (m).

WBSGT verwendet in den jeweiligen Zuständen aus einer Potenzmenge von Prioritätsregeln die geeignete. Von der Regelmenge hängt ab, welche einzuplanenden Alternativen

[13] Siehe Abschnitt 5.3.4.
[14] Der optimale Pfad wird hier schwarz und nicht gestrichelt dargestellt.
[15] Vgl. (Fisher und Thompson, 1963).

	M_1		M_2		M_3		M_4		M_5		M_6	
	or	*pt*	*or*	*pt*	*or*	*pt*	*or*	*pt*	*or*	*pt*	*or*	*pt*
J_1	3	1	1	3	2	6	4	7	6	3	5	6
J_2	2	8	3	5	5	10	6	10	1	10	4	4
J_3	3	5	4	4	6	8	1	9	2	1	5	7
J_4	2	5	1	5	3	5	4	3	5	8	6	9
J_5	3	9	2	3	5	5	6	4	1	3	4	1
J_6	2	3	4	3	6	9	1	10	5	4	3	1

Tabelle 6.2.: Benchmark-Problem: Fisher and Thompson (FT06) 6×6[15]

in einem Konflikt theoretisch auswählbar sind.[16] Beispielsweise sind mit den Regeln SPT und LPT Konflikte mit mehr als zwei Alternativen problematisch, da möglicherweise keine der Regeln die optimale Alternative auswählen kann. Neben der Annahme, alle Alternativen wären auswählbar (hierzu dient die Konfiguration WbsGT (*alle Alternativen*)), zeigt das Experiment das Verhalten von konkreten Regelmengen, wie etwa WbsGT (*SPT LPT*). Das Giffler-Thompson-Verfahren, welches Alternativen aus der Konfliktmenge zufällig und ohne Regeln auswählt, wird in den Abbildungen 6.2 und 6.3 als *GT* und *GTiter* dargestellt.

Der optimale Wert der Zielfunktion C_{max} ist für dieses Experiment mit dem FT06-Problem 54.[17] Für das Experiment wird das Problem 100 von den Verfahren gelöst. Die Ergebnisse werden gemittelt. Die Abweichungen alternativer Verfahren, die zu genau einer Lösung führen und somit keine Mittelung benötigen, sind in Tabelle 6.3 festgehalten.

Scheduling-Verfahren	Prozentuale Abweichung zur optimalen Lösung
Shifting-Bottleneck	12%
Giffler-Thompson mit SPT	50%
Giffler-Thompson mit LPT	16%
Giffler-Thompson mit LSO	18%
Giffler-Thompson mit MSO	44%
Giffler-Thompson mit LWR	68%
Giffler-Thompson mit MWR	9%

Tabelle 6.3.: Allwissender Klassifizierer: Alternative Verfahren und Regeln

Abbildung 6.2 zeigt Beispielkonfigurationen für WbsGT, in denen ausschließlich Po-

[16] Siehe Abschnitt 5.3.3.3.
[17] Vgl. (Jain und Meeran, 1999).

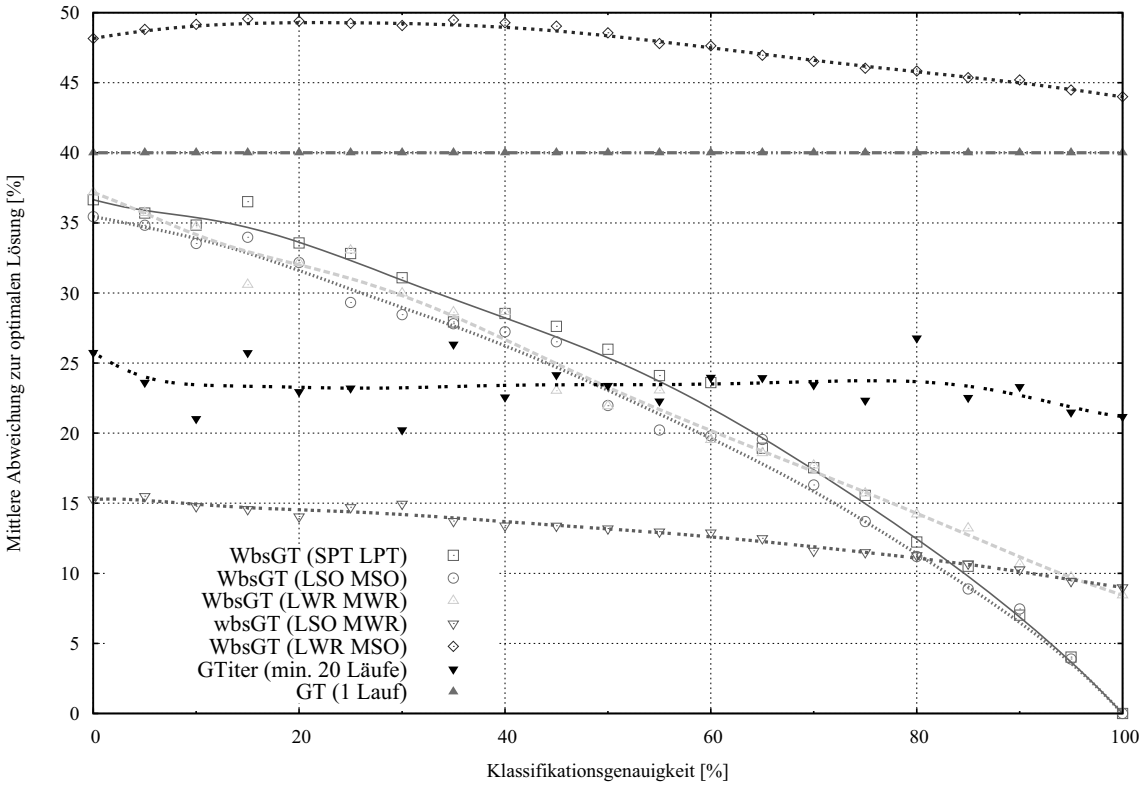


Abbildung 6.2.: Experiment allwissender Klassifizierer: Lösungsgüte des allwissender Klassifizierers mit kleinen Prioritätsregelmengen

tenzmengen bestehend aus zwei verschiedene Regeln zum Einsatz kommen. Es ist zu erkennen, dass die Abweichung des WbsGT-Verfahrens von der optimalen Lösungsgüte bei vielen Potenzmengenklassen bereits bei einer Klassifikationsgenauigkeit von 0 Prozent unterhalb der Abweichung von 40 Prozent des von der Klassifikationsgenauigkeit unabhängigen Giffler-Thompson-Verfahrens (GT) mit zufälliger Auswahl und einem Durchlauf liegt. GT verwendet keine Prioritätsregeln, die Operationen werden zufällig der Konfliktmenge entnommen. Als zweite Variante der intialen Giffler-Thompson-Heuristik wird eine iterationsbasierte Version ($GTiter$) verwendet. Diese löst das Benchmark-Problem so lange, bis der Iterationszähler größer 20 ist. Wird eine bessere Lösung als die bisher beste erreicht, wird der Zähler zurück gesetzt und erneute 20 Durchläufe sind erlaubt.[18] Die *GTiter*-Konfiguration erreicht eine durchschnittliche Lösungsgüte von ca. 23 Prozent.

Es treten zwei WbsGT-Ausprägungen durch Extrema hervor. Mittels WbsGT (*LWR MSO*) werden mit 48 Prozent deutlich schlechtere Ergebnisse erzielt als mit der GT-

[18]Dieser Zähler kann frei variiert werden, in Experimente hat sich jedoch gezeigt, das auch größere Zähler kaum besser Ergebnisse liefern.

Konfiguration.[19] Das andere Extrem mit einer Abweichung von 15 Prozent bildet WbsGT (*LSO MWR*). Hierbei ist zu beachten, dass die beiden Verfahren mit komplementären Prioritätsregeln ausgestattet sind und dass WbsGT (*LSO MWR*) grundsätzlich – unabhängig der Klassifikationsgenauigkeit – dazu geeignet ist, beim FT06-Problem gute Ergebnisse zu erzielen.[20]

Wie gezeigt, liegen bis auf WbsGT (*LSO MWR*) alle WbsGT-Ausprägungen bei einer Klassifikationsgenauigkeit von 0 Prozent über den Ergebnissen des *GTiter*. Ab einer Klassifikationsgenauigkeit von ca. 57 Prozent liegen die Abweichungen der WbsGT-Varianten unterhalb des iterativen Giffler-Thompson. WbsGT (*LWR MSO*) bildet auch hier eine Ausnahme und ist selbst bei steigender Klassifikationsgenauigkeit bis hin zu 100 Prozent nicht in der Lage, die Lösungsgüte des *GTiter* zu erreichen insbesondere diese nicht zu übertreffen. Sogar die Lösungsgüte des *GT* ist nicht erreichbar. Der Grund für das schlechte Abschneiden der Konfiguration WbsGT (*LWR MSO*), im Vergleich zu den anderen Lösungen, ist in der Regelmenge selbst zu sehen. Durch die Auswahl der Prioritätsregeln sind nicht alle Lösungen erreichbar. So liegt die initiale Lösungsgüte der Konfiguration WbsGT (*LWR MSO*) bei 0 Prozent Klassifikationsgenauigkeit sogar unterhalb der einfachen Regelanwendung von MSO (siehe Tabelle 6.3). Erst mit steigender Genauigkeit lernt das Verfahren von den beiden schlecht geeigneten Regeln die bessere MSO Regeln anzuwenden.

Insgesamt verbessern alle WbsGT-Konfigurationen mit zwei Prioritätsregeln ihre Lösungsgüte, wenn sich die Klassifikationsgenauigkeit erhöht. Doch erreichen die einzelnen Varianten unterschiedlich gute Lösungen bei einer Klassifikationsgenauigkeit von 100 Prozent. Die bei geringer Klassifikationsgenauigkeit dominierende Konfiguration WbsGT (*LSO MWR*) konnte sich nur um 7 Prozent auf eine Abweichung von 9 Prozent verbessern. Hier hat der Klassifizierer wie bei der Konfiguration WbsGT (*LWR MSO*) gelernt, dass die einfache Anwendung einer Regel (MSO) zu den besten Ergebnissen führt. Die beste Lösungsgüte der WbsGT Verfahren liefern WbsGT (*SPT LPT*) und WbsGT (*LSO MSO*) mit der optimalen Lösung. Dieses Ergebnis liegt deutlich unter der Lösungsgüte bei einfacher Anwendung der besten der beiden Prioritätsregeln LPT mit 16 und LSO mit 18 Prozent. Die Ergebnisse zeigen den Effekt, der durch Auswahl aus der Regelmenge auftritt. Funktioniert der Klassifizierer schlecht, wird mindestens das Ergebnis der besten einfachen Regel erreicht. Arbeitet der Klassifizierer hingegen gut und ist ein besseres Ergebniss mit der Regelkombination im Vergleich zur besten einfachen Regel möglich, übertrifft das Verfahren mit Klassifizierer die Leistung der

[19] Die beiden Prioritätsregeln sind also grundsätzlich schlecht zur Lösung des FT06-Problems geeignet.
[20] Die beiden Prioritätsregeln sind also grundsätzlich gut zur Lösung des FT06-Problems geeignet.

besten einzelnen Regel deutlich. WBSGT-Konfigurationen mit mehr als zwei Regeln sind in Abbildung 6.3 dargestellt.

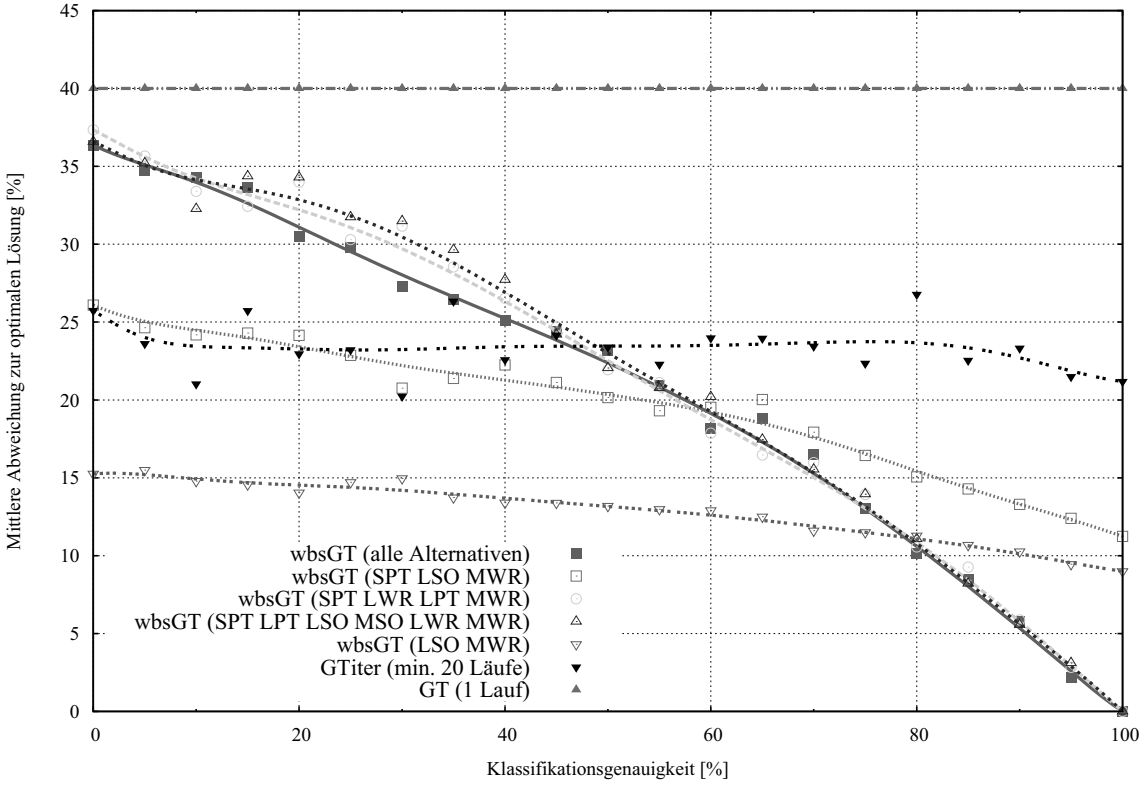


Abbildung 6.3.: Allwissender Klassifizierer: Lösungsgüte große Regelmengen

Die in Abbildung 6.3 dargestellten Ergebnisse, die mit dem Giffler-Thompson-Verfahren und der möglichen Auswahl von allen Operationen (WBSGT (*alle Alternativen*)) erreicht werden, stützen die Hypothese[21] zusätzlich. Eine höhere Klassifikationsgenauigkeit führt grundsätzlich zu besseren und insbesondere bei einer Genauigkeit von 100 Prozent zu optimalen Ergebnissen.[22] Bei einer Klassifikationsgenauigkeit von 0 Prozent wird eine 37 prozentige Abweichung zum Optimum erreicht und eine Klassifikationsgenauigkeit von 100 Prozent führt zu einer Abweichung von 0 Prozent – das optimale Ergebnis. Bei dieser Ausprägung des Verfahrens wird operationsweise aus der Konfliktmenge und nicht mittels Prioritätsregeln ausgewählt, weshalb jede Konfliktoperation wählbar ist. Im realen Einsatz können die Operationen nicht direkt aus der Konfliktmenge gewählt werden, da der Klassifizierer sonst für jede mögliche Art von Operation

[21] Siehe Abschnitt 2.2.3.1.

[22] Hierbei muss vorausgesetzt werden, dass die richtige Klassifikationsentscheidung mit der korrekten Auswahl aus der Konfliktmenge gleichzusetzten ist. Grundsätzlich sind korrekte Klassifikation und korrekte Entscheidung nicht zwingend identisch, da trotz korrekter Klassifizierung die Regeln nicht zwingend die optimale Operation wählen kann.

eine Klasse haben müsste.[23] Gleichzeitig würde das Verfahren hierdurch an Flexibilität verlieren, weshalb WBSGT (*alle Alternativen*) nur für das Experiment zur Untersuchung der Hypothese eingesetzt wird.[24]

Bei WBSGT-Konfigurationen mit mehr als zwei Regeln wird die Lösungsgüte der *GTiter*-Konfiguration bereits bei einer Klassifikationsgenauigkeit von 50 Prozent unterboten und die Ergebnisse sind nie schlechter als die des einfachen *GT* ohne Iterationen. Die Konfiguration WBSGT (*SPT LWR LPT MWR*) hat eine hinreichend große Regelmenge um die optimale Lösung zu erreichen. Dahingehend ist WBSGT (*SPT LSO MWR*) mit der kleineren Regelmenge bei geringen Klassifikationsgenauigkeiten deutlich besser, doch aufgrund der nicht ausreichenden Anzahl von Regeln zur Operationsauswahl nicht in der Lage, das optimale Ergebnis – selbst bei einer 100 prozentigen Klassifikationsgenauigkeit – zu erreichen.

Grundsätzlich zeigt dieses Experiment, dass bei steigender Klassifikationsgenauigkeit die Lösungsgüte des WBSGT-Verfahrens ansteigt. Eine hinreichend große Regelmenge vorausgesetzt, so ist mit hoher Klassifikationsgenauigkeit das optimale Ergebnis erreichbar. Damit kann die Hypothese des Zusammenhangs zwischen Lösungsgüte und Klassifikationsgenauigkeit gestützt werden.[25]

Benchmark: Klassifikationsgenauigkeit

Wie gezeigt, steht die Klassifikationsgenauigkeit in einem starken Zusammenhang mit der Lösungsgüte der WBSGT-Verfahren. Tabelle 6.4 listet die Klassifikationsgenauigkeiten einiger WBSGT-Konfigurationen auf, die mit den hier verwendeten Merkmalen und Trainingsinstanzen beim FT06-Problem erreicht werden.

In der Konfiguration WBSGT (*LWR MWR LSO MSO*) stehen dem Klassifizierer die Klassen c_{LWR}, c_{MWR}, c_{LSO}, c_{MSO}, c_{LWRMWR}, c_{LSOMSO} etc. zur Verfügung. Ob die gesamte Potenzmenge verwendet wird, hängt vom Training ab.[27] Die Klassifikationsgenauigkeit werden ermittelt und mit einer Strategie verglichen, die mit der Wahrscheinlichkeit beim Raten – hier ca. 6 Prozent – eine der genannten Regeln auswählt. Die Klassifikationsgenauigkeit von 49 Prozent ist somit 635 Prozent genauer als das zufällige Auswählen einer Klasse. Bei der Konfigurationen WBSGT (*SPT LSO MWR*)

[23] Siehe Abschnitt 5.3.3.

[24] Zusätzlich kann mit diesem Experiment anschaulich gezeigt werden, dass die Giffler-Thompson-Heuristik die optimale Lösung generieren kann.

[25] Siehe Abschnitt 2.2.3.1.

[26] Unter den F-Maß Werten sind nur die vom Klassifizierer ausgewählten Werte angegeben. Sämtliche nicht aufgeführten F-Maße haben einen Wert von Null.

[27] Gab es keine Situation in der SPT und MSO geeignet waren, ist diese Klasse nicht vorhanden.

WbsGT	Klassifikations-genauigkeit	F-Mass
LWR MWR LSO MSO	49%	0,148 MWR 0,174 LSO 0,500 MSO 0,653 MWR LSO 0,324 MWR MSO 0,472 LWR LSO 0,382 LWR MSO
SPT LSO MWR	48%	0,317 SPT 0,367 LSO 0,144 MWR 0,604 MWR LSO 0,452 SPT LSO 0,208 SPT MWR 0,845 SPT LSO MWR
LWR MWR	59%	0,467 LWR 0,671 MWR

Tabelle 6.4.: Klassifikationsgenauigkeit FT06: WbsGT-Genauigkeit[26]

(WbsGT (*LWR MWR*))ist die Klassifikationsgenauigkeiten von 48 Prozent (59 Prozent) etwa 236 Prozent (77 Prozent) genauer als die zufällige Auswahl. Damit zeigt sich zum einen, dass Klassifizieren zu besseren Ergebnissen führt als Raten. Zum anderen ziegt sich, dass durch weitere (andere) Merkmale das Ergebnis wahrscheinlich noch verbessert werden kann.

Benchmark: Makespan

Nachfolgend werden die Ergebnisse der Evaluierung des Makespans vorgestellt. Hierzu sind exemplarisch vier Benchmark-Probleme (FT06, ABZ7, SWV01 und CAR03) aufgeführt. Ergebnisse zum Adaptionsmechanismus finden sich im Anhang A.8.

Makespan FT06: Im Folgenden werden die Ergebnisse des WbsGT-Verfahrens bei der Anwendung im FT06-Benchmark-Problem[28] dargestellt. Für das Experiment sind 100 Iterationen mit abgewandelten Probleminstanzen gelöst und die Ergebnisse gemittelt worden. Die Ausprägungen der Parameter zur Änderung der Probleminstanzen waren:[29] $\mu = 0$, $\sigma = 50$ und $\kappa = 0$. Zum Training wurde das Problem mittels des Branch-and-Bound-Algorithmus in 28 abgewandelten Probleminstanzen gelöst. Die bei dem Experiment eingesetzten WbsGT-Konfigurationen – verwendete Regelmengen – sind:

[28]Siehe Tabelle 6.2.
[29]Siehe Abschnitt 5.3.4.

- WbsGT (*LWR MWR LSO MSO*)
- WbsGT (*SPT LSO MWR*)
- WbsGT (*SPT LPT LSO MSO LWR MWR*)
- WbsGT (*LWR MWR*)
- WbsGT (*LSO MWR*)
- WbsGT (*SPT LPT LWR MWR*)
- WbsGT (*SPT LPT*)
- WbsGT (*LSO MSO*)
- WbsGT (*LWR MSO*)

Die aufgeführten Regelmengen wurden ebenfalls in den weiteren Experimenten zur Evaluierung des Makespans eingesetzt. Als Lösungsverfahren bei den folgenden Experimenten zum Makespan wurden eingesetzt:[30]

- Das WbsGT-Verfahren
- Das WbsGT-Verfahren mit gleicher Regelmenge jedoch ohne Klassifizierer
- Die Giffler-Thompson-Heuristik mit einer fest eingestellten Prioritätsregel jedoch ohne Klassifizierer
- Die normale Giffler-Thompson-Heuristik mit zufälliger Auswahl aus der Konfliktmenge
- Eine Shifting-Bottleneck-Heuristik

In Abbildung 6.4 sind die Lösungsergebnisse der einzelnen Verfahren abgetragen. Die detaillierten Ausprägungen der Einzelwerte sind in Tabelle 6.5 aufgeführt. Den Referenzwert bildet jeweils die Branch-and-Bound-Lösung. Für jede Verfahrensausprägung werden der Mittelwert sowie die Varianz über die 100 gelösten Instanzen im Diagramm dargestellt.

Das Shifting-Bottleneck-Verfahren findet Lösungen, die im Durchschnitt etwa um 8,8 Prozent von den optimalen Lösungen abweichen. Die Varianz fällt dabei mit 6,63 gering aus. Das Scheduling mit Prioritätsregeln hat größere Abweichungen. Es nähert sich mit den hier besten einzelnen Regeln LSO und MWR bis ca. zu 14 Prozent und 15 Prozent

[30] Bei den Ausprägungen mit Giffler-Thompson-Heuristik wurde der Zähler für die Iterationsläufe auf 20 gesetzt.

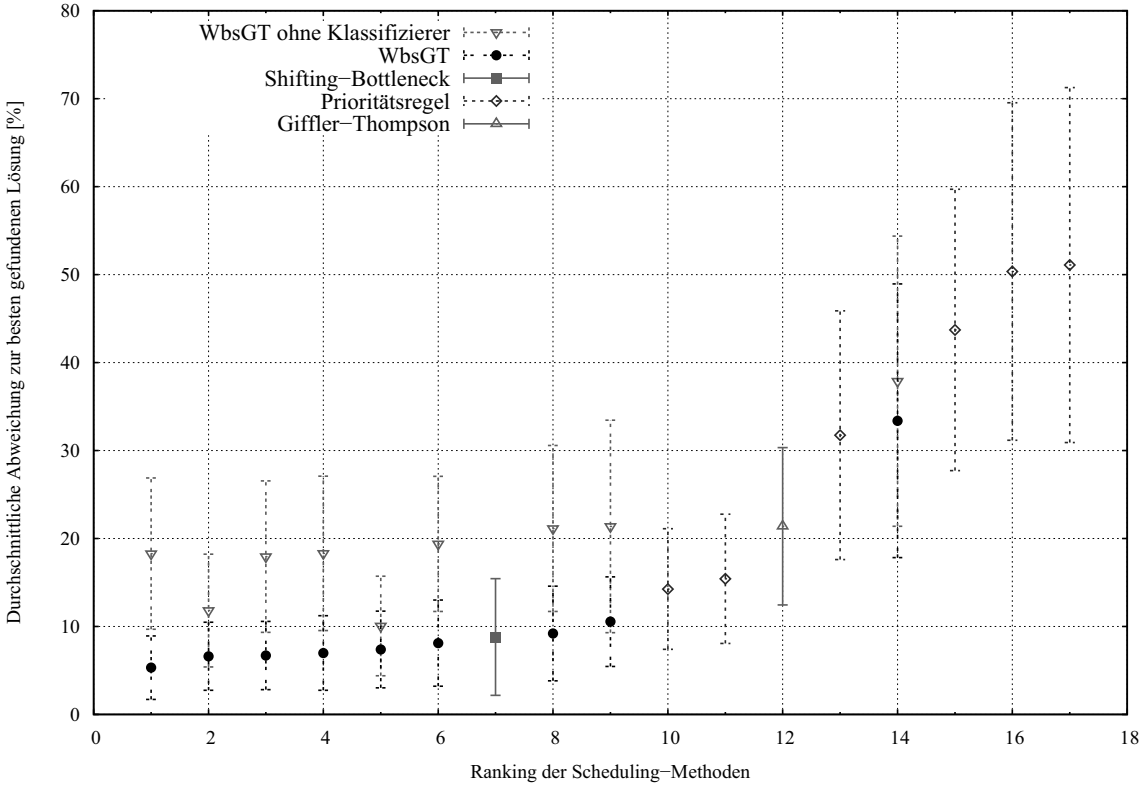


Abbildung 6.4.: Makespan FT06: Benchmark WbsGT-Verfahren

den optimalen Werten an. Die Varianzen liegen hierbei mit 6,85 und 7,43 in der Nähe der Varianz der Shifting-Bottleneck-Heuristik. Das Giffler-Thompson-Verfahren mit zufälliger Operationsauswahl erreicht eine Abweichung von etwa 21 Prozent, wobei die Varianz hier mit 8,94 relativ groß ist. Die hier beste Lösung wird von der WbsGT-Konfiguration WbsGT (*LWR MWR LSO MSO*) mit einer durchschnittlichen Abweichung von 5,32 Prozent generiert. Ebenso ist die Varianz mit einem Wert von 3,60 am geringsten. WbsGT ohne Klassifizierer und mit der Regelmenge (LWR MWR LSO MSO) dahingegen erreicht nur einen Mittelwert von 18,28 Prozent bei der relativ großen Varianz von 8,60. Es zeigt sich, dass die meisten WbsGT-Konfigurationen (6 von 9) näher an der optimalen Lösung liegen als das Shifting-Bottleneck-Verfahren. Ebenso generieren diese Konfigurationen immer bessere Lösungen als die gleichen Konfigurationen ohne Klassifizierer und mit zufälliger Prioritätsregelauswahl, bei meistens geringerer Varianz. Es ist gut zu erkennen, dass die einzelnen Regeln LSO und MWR die besten Ergebnisse unter den Regeln liefern und in Kombination das beste Ergebnis ohne Klassifizier. Offensichtlich ist die Regelmenge jedoch nicht ausreichend groß, um zu besseren Ergebnisse zu

Rang	Verfahren	Abweichung	& Varianz	Regelmenge	Abw. & Var. ohne Klassifizierer	
1	WbsGT	5,32	3,60	LWR MWR LSO MSO	18,28	8,60
2	WbsGT	6,61	3,87	SPT LSO MWR	11,81	6,40
3	WbsGT	6,7	3,87	SPT LPT LSO MSO LWR MWR	17,94	8,60
4	WbsGT	6,98	4,24	LWR MWR	18,32	8,77
5	WbsGT	7,39	4,35	LSO MWR	10,06	5,65
6	WbsGT	8,1	4,89	SPT LPT LWR MWR	19,38	7,68
7	Shifting-Bottleneck	8,8	6,63	–	–	–
8	WbsGT	9,2	5,38	SPT LPT	21,14	9,43
9	WbsGT	10,55	5,09	LSO MSO	21,37	12,08
10	LSO	14,25	6,85	–	–	–
11	MWR	15,42	7,34	–	–	–
12	Giffler-Thompson	21,39	8,94	–	–	–
13	LPT	31,74	14,14	–	–	–
14	WbsGT	33,38	15,55	LWR MSO	37,88	16,49
15	SPT	43,71	16,00	–	–	–
16	MSO	50,35	19,18	–	–	–
17	LWR	51,09	20,17	–	–	–

Tabelle 6.5.: Makespan FT06: Übersicht

gelangen.[31] So liegen die Lösungen von WbsGT (*LSO MWR*) mit einer durchschnittlichen Abweichung von 7,39 Prozent nur auf einem mittleren Rang. Erst mit größeren Regelmengen können mehr Operationen aus der Konfliktmenge ausgewählt werden, womit bessere Lösungen zu erzielen wären.

Abbildung 6.5 illustriert die Verbesserung der wissensbasierten Regelauswahl (WbsGT) beim FT06-Benchmark-Problem im Vergleich zur zufälligen Auswahl der Prioritätsregeln (WbsGT ohne Klassifizierer). Wie bereits beschrieben, wird bei letzter Variante zufällig eine Regel zur Auswahl einer Operation aus der Konfliktmenge gewählt.[32] Die Regelmengen sind für beiden Verfahren identisch.

In Abbildung 6.5 wird besonders deutlich, dass die Ergebnisse der wissensbasierten Auswahl der Prioritätsregeln bei allen neun Regelmengen besser sind als bei der zufälligen Auswahl. Sowohl bei der hier am besten geeigneten Regelmenge (LWR MWR LSO MSO), als auch bei der am schlechtesten geeignetsten Regelmenge (LWR MSO) ist eine Verbesserung durch das WbsGT-Verfahren erzielbar. Bei den geeignetsten Regeln kommt es zu einer Verbesserung von ca. 13 Prozent und bei den ungeeignetsten Regeln immer noch zu einer Verbesserung von 4 Prozent gegenüber dem Verfahren ohne Klassifizierer. Zusätzlich ist ein Unterschied in der Varianz zu erkennen. Bei den ungeeignetsten Regeln ist der Wert des Unterschieds ca. 1 und bei den geeignetsten sogar etwa 5. Insgesamt wird damit deutlich, dass dasWbsGT-Verfahren gut funktio-

[31] Siehe Abschnitt 6.1.2. Dabei sind die Werte jedoch nicht 100 prozentig zu vergleichen, da das initiale Problem beim allwissenden Klassifizierer ohne Abwandlungen gelöst wurde.

[32] Werden mehrere Operation der Konfliktmenge von der gewählten Regeln gleich bewertet, so wird aus diesen per Zufall eine Operation gewählt (siehe Abschnitt 5.3.3.3).

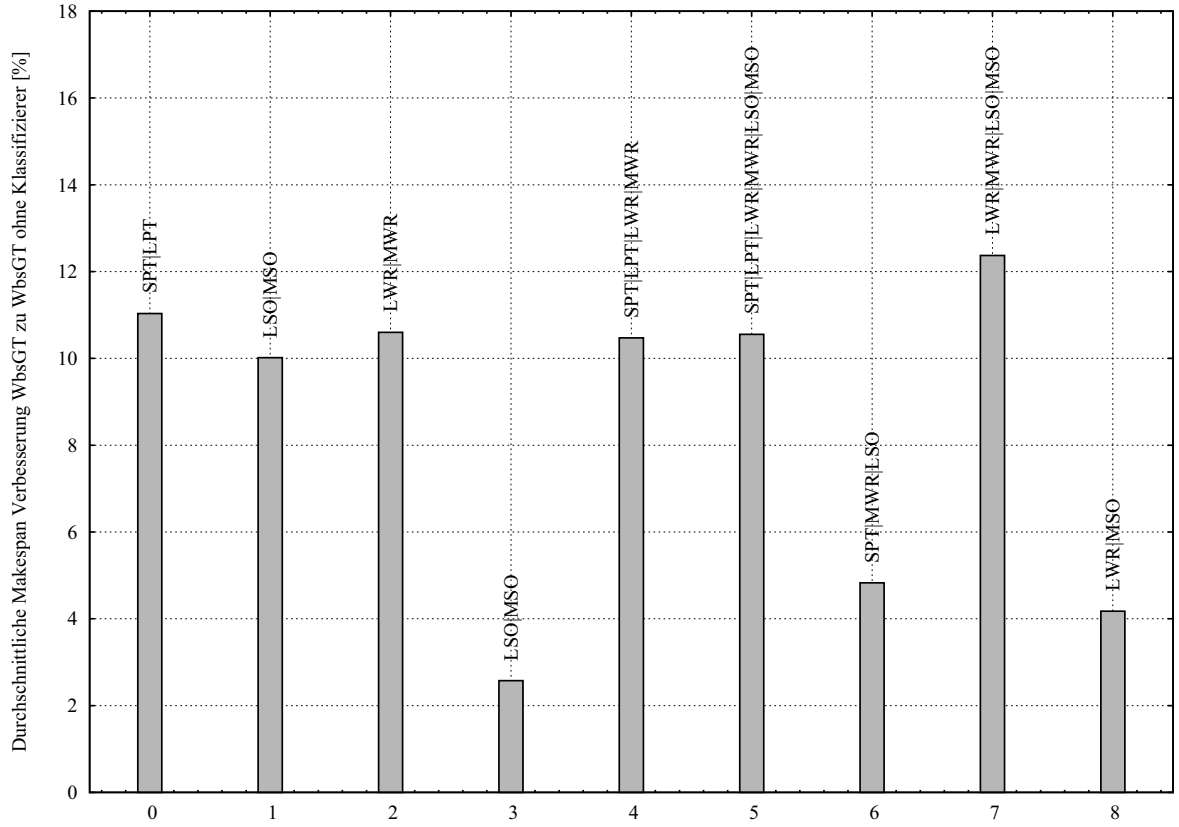


Abbildung 6.5.: Makespan FT06: Verbesserung der Lösungsgüte

niert und eine deutliche Verbesserung der Lösungen im Vergleich zu den Verfahren ohne Klassifizierer erreicht wird.

Makespan ABZ7: Nachfolgend werden die Ergebnisse des WbsGT-Verfahrens bei der Anwendung im ABZ7-Benchmark-Problem[33] dargestellt. Für das Experiment sind zehn Iterationen mit abgewandelten Probleminstanzen gelöst worden. Die Ausprägungen der Parameter zur Änderung der Probleminstanzen waren: $\mu = 0$, $\sigma = 50$ und $\kappa = 0$. Zum Training wurde das Problem mittels des Branch-and-Bound-Algorithmus in 40 abgewandelten Probleminstanzen gelöst. Im Vergleich zu dem bereits untersuchten FT06-Benchmark-Problem ist das ABZ7 deutlich komplexer. Bei diesem konnte der Branch-and-Bound-Algorithmus keine optimalen Lösungen bilden. Aus diesem Grund findet der Vergleich der Ergebnisse hier nicht mit den optimalen Lösungen statt, sondern mit der besten erreichten Lösung nach dem Beenden des Branch-and-Bound-Algorithmus.[34]

[33] Siehe Tabelle A.1.

[34] Im Experiment wurde der Branch-and-Bound-Algorithmus nach 30 Minuten beendet und das bis zu diesem Zeitpunkt erreichte Ergebnis als Trainings- bzw. Referenzlösung verwendet.

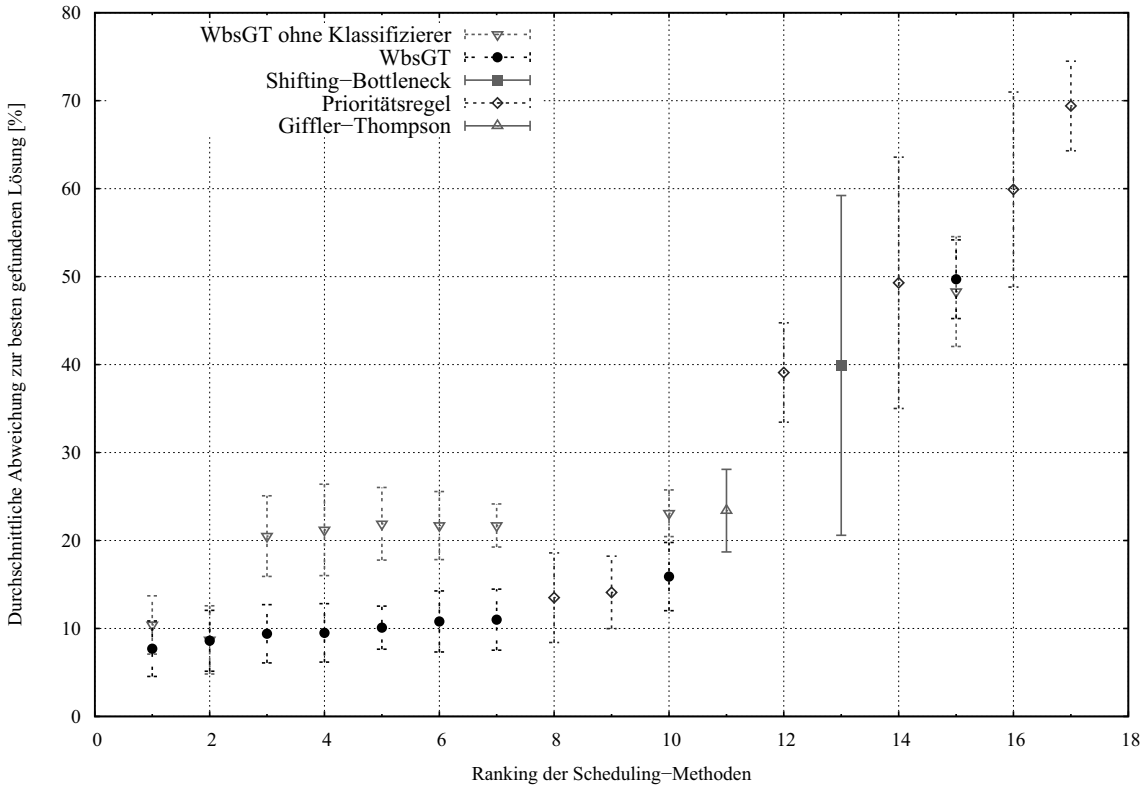


Abbildung 6.6.: Makespan ABZ7: Benchmark WbsGT-Verfahren

Abbildung 6.6 illustriert die Resultate, Tabelle 6.6 zeigt die Lösungsergebnisse der einzelnen Verfahren im Detail. Für jede Ausprägung werden Mittelwert und Varianz im Diagramm abgetragen. Im Gegensatz zum eher kleinen FT06-Benchmark-Problem findet das Shifting-Bottleneck-Verfahren beim komplexeren ABZ7 lediglich Lösungen, die im Durchschnitt um etwa 39,9 Prozent von den Branch-and-Bound-Lösungen abweichen. Auch fällt die Varianz mit 19,31 groß aus. Das Scheduling mit Prioritätsregeln hat geringere Abweichungen und nähert sich mit den hier besten einzelnen Regeln MWR und LSO bis auf ca. 13 Prozent und 14 Prozent den besten Werten an. Die Varianzen liegen hierbei mit 5,09 und 4,12 deutlich unterhalb der Varianz der Shifting-Bottleneck-Heuristik. Das Giffler-Thompson-Verfahren mit zufälliger Operationsauswahl erreicht eine 23 prozentige Abweichung vom besten Ergebnis bei einer relativ kleinen Varianz von 4,69. Damit ist die Lösungsgüte des Giffler-Thompson-Verfahrens besser als die des Shifting-Bottleneck-Verfahrens. Die beim ABZ7-Benchmark-Problem beste Lösung (nach dem Branch-and-Bound-Verfahren) wird von der WbsGT-Konfiguration WbsGT (*SPT LSO MWR*) mit einer durchschnittlichen Abweichung von 7,7 Prozent und einer Varianz von 3,16 generiert. Dahingegen erreicht WbsGT ohne Klassifizierer mit der Regelmenge (SPT LSO MWR) nur einen Mittelwert der Abweichung von 10,4 Pro-

Rang	Verfahren	Abweichung & Varianz		Regelmenge	Abw. & Var. ohne Klassifizierer	
1	WbsGT	7,7	3.16	SPT LSO MWR	10,4	3.31
2	WbsGT	8,6	3,46	LSO MWR	8,7	3,87
3	WbsGT	9,4	3,31	LWR MWR	20,5	4,58
4	WbsGT	9,5	3,31	LWR MWR LSO MSO	21,2	5,19
5	WbsGT	10,1	2,44	LSO MSO	21,9	4,12
6	WbsGT	10,8	3,46	SPT LWR LPT MWR	21,7	3,87
7	WbsGT	11,0	3,46	SPT LPT LSO MSO LWR MWR	21,7	2,44
8	MWR	13,5	5,09	–	–	–
9	LSO	14,1	4,12	–	–	–
10	WbsGT	15,9	3,87	SPT LPT	23,1	2,64
11	Giffler-Thompson	23,4	4,69	–	–	–
12	LPT	39,1	5,65	–	–	–
13	Shifting-Bottleneck	39,9	19,31	–	–	–
14	SPT	49,3	14,28	–	–	–
15	WbsGT	49,7	4,47	LWR MSO	48,3	6,24
16	MSO	59,9	11,09	–	–	–
17	LWR	69,4	5,09	–	–	–

Tabelle 6.6.: Makespan ABZ7: Übersicht

zent bei einer ähnlich großen Varianz von 3,31. Es ist zu erkennen, dass fast alle Konfigurationen (8 von 9) des WbsGT-Verfahrens näher an der Branch-and-Bound-Lösung liegen als das Shifting-Bottleneck-Verfahren. Auch generiert WbsGT immer bessere Lösungen als die gleichen Konfigurationen ohne Klassifizierer mit zufälliger Regelauswahl (WbsGT (*LWR MSO*) ausgenommen). Ferner zeigen die Ergebnisse, dass die einzelnen Regeln LSO und MWR wie beim dem FT06-Benchmark-Problem die besten Lösungen unter den einzelnen Regeln generieren. Auch bei dem komplexeren ABZ7-Benchmark-Problem ist die aus zwei Elementen bestehende Regelmenge nicht hinreichend groß. Die Lösungen von WbsGT (*LSO MWR*) liegen mit einer durchschnittlichen Abweichung von 8,6 Prozent zwar auf dem zweiten Rang, jedoch werden diese noch von der WbsGT-Konfiguration mit der um die Regel SPT erweiterten Menge übertroffen. Gleichzeitig sinkt die Varianz von 3,46 auf 3,16. Damit wird auch hier deutlich, dass erst mit ausreichend großen Regelmengen genug Operationen aus der Konfliktmenge ausgewählt werden können, um gute Lösungen zu erzielen.

Makespan SWV01: Die Ergebnisse des WbsGT-Verfahrens bei Lösung des SWV01-Scheduling-Problems[35] werden im Folgenden dargestellt. Für das Experiment wurden 50 Iterationen mit abgewandelten Probleminstanzen gelöst. Die Ausprägungen der Parameter zur Änderung der Probleminstanzen waren: $\mu = 0$, $\sigma = 50$ und $\kappa = 0$. Zum Training wurde das Problem mittels des Branch-and-Bound-Algorithmus in fünf abgewandelten Probleminstanzen gelöst. Beim komplexen SWV01-Problem konnte der Branch-

[35] Siehe Tabelle A.3.

and-Bound-Algorithmus keine optimale Lösungen bilden. Aus diesem Grund findet der Vergleich der Ergebnisse hier nicht mit den optimalen Lösungen statt, sondern mit der besten erreichten Lösung nach dem Beenden des Branch-and-Bound-Algorithmus.[36]

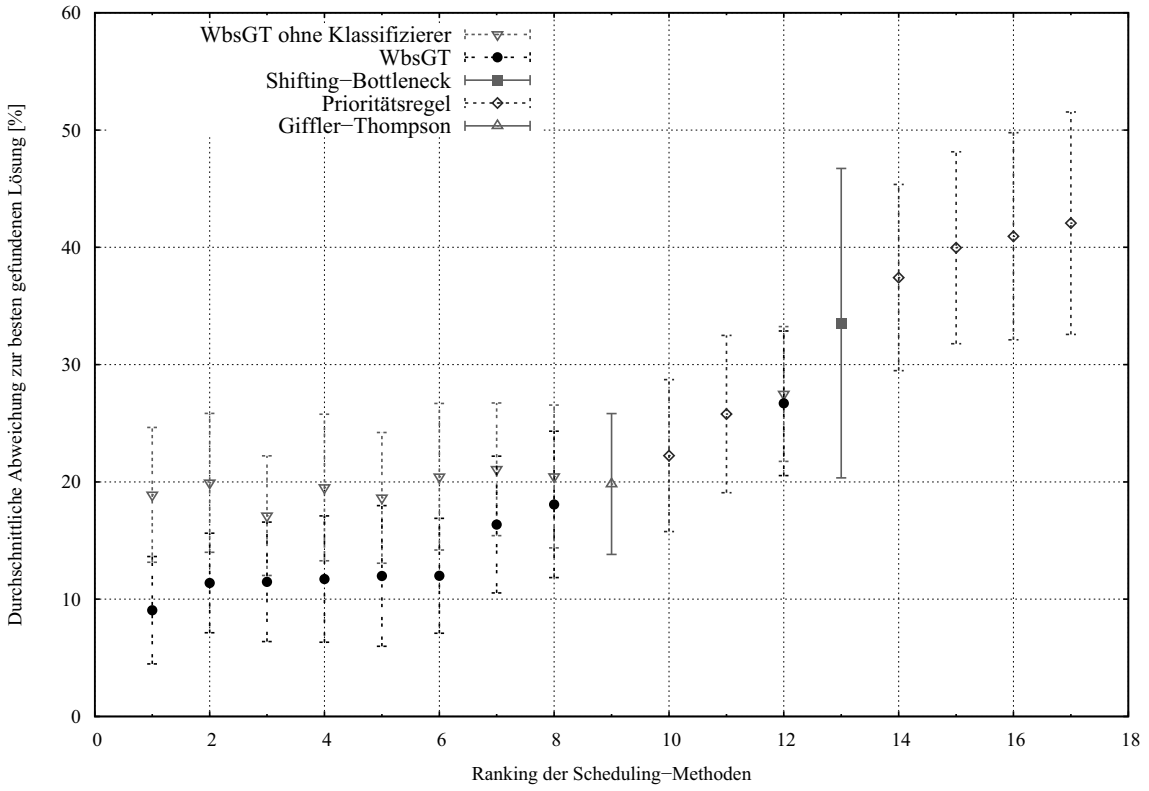


Abbildung 6.7.: Makespan SWV01: Benchmark WbsGT-Verfahren

Abbildung 6.7 zeigt die erreichten Ergebnisse. Die detaillierten Ergebnisse der einzelnen Verfahren beim SWV01-Benchmark-Problem zeigt Tabelle 6.7. Für jede Lösung werden Mittelwert und Varianz im Diagramm abgetragen. Verglichen mit dem komplexeren ABZ7-Benchmark-Problem, findet das Shifting-Bottleneck-Verfahren beim SWV01-Problem, mit einer im Durchschnitt 33,54 prozentigen Abweichung von den Lösungen des Branch-and-Bound-Verfahrens, etwas bessere Lösungen. Auch die Varianz fällt mit 13,19 Prozent geringer aus. Doch ist beim SWV01-Benchmark-Problem die Lösungsqualität des Shifting-Bottleneck-Verfahren insgesamt ebenfalls schlecht. Das Scheduling mit Prioritätsregeln erreicht bessere Lösungen als das Shifting-Bottleneck-Verfahren, bei gleichzeitig geringeren Varianzen. Die besten einzelnen Regeln MWR und LSO weichen im Mittel um 22,24 Prozent und 25,78 Prozent bei Varianzen von 6,48 und 6,70 von den Lösungen des Branch-and-Bound-Verfahrens ab. Das Giffler-Thompson-Verfahren mit

[36] Im Experiment wurde der Branch-and-Bound-Algorithmus nach 30 Minuten beendet und das bis zu diesem Zeitpunkt erreichte Ergebnis als Trainings- bzw. Referenzlösung verwendet.

Rang	Verfahren	Abweichung & Varianz		Regelmenge	Abw. & Var. ohne Klassifizierer	
1	WbsGT	9,06	4,58	LWR MWR LSO MSO	18,9	5,74
2	WbsGT	11,38	4,24	SPT LPT LSO MSO LWR MWR	19,92	5,91
3	WbsGT	11,48	5,09	SPT LSO MWR	17,12	5,09
4	WbsGT	11,72	5,38	LSO MSO	19,52	6,24
5	WbsGT	11,98	6,00	LWR MWR	18,64	5,56
6	WbsGT	12,0	4,89	SPT LPT LWR MWR	20,44	6,24
7	WbsGT	16,36	5,83	SPT LPT	21,08	5,65
8	WbsGT	18,08	6,24	LSO MWR	20,46	6,08
9	Giffler-Thompson	19,82	6,00	–	–	–
10	MWR	22,24	6,48	–	–	–
11	LSO	25,78	6,70	–	–	–
12	WbsGT	26,7	6,16	LWR MSO	27,5	5,74
13	Shifting-Bottleneck	33,54	13,19	–	–	–
14	MSO	37,42	7,93	–	–	–
15	SPT	39,96	8,18	–	–	–
16	LWR	40,94	8,83	–	–	–
17	LPT	42,06	9,48	–	–	–

Tabelle 6.7.: Makespan SWV01: Übersicht

zufälliger Operationsauswahl erreicht mit einer Abweichung von 19,82 Prozent und einer Varianz von 6,00 ein etwas besseres Ergebnis als beim ABZ7-Benchmark-Problem, jedoch ist die Varianz beim SWV01 größer. Die Lösungsgüte des Giffler-Thompson-Verfahrens ist um mehr als 13 Prozentpunkte besser als die des Shifting-Bottleneck-Verfahrens. Die beim SWV01-Benchmark-Problem besten Lösungen (nach den Lösungen des Branch-and-Bound-Verfahrens) werden von der WbsGT-Konfiguration WbsGT (*LWR MWR LSO MSO*) mit einer durchschnittlichen Abweichung von 9,06 Prozent und einer Varianz von 4,58 generiert. Dahingegen erreicht WbsGT ohne Klassifizierer mit der Regelmenge (LWR MWR LSO MSO) nur eine mittlere Abweichung von 18,9 Prozent bei einer größeren Varianz von 5,74. Die wissensbasierten Lösungen sind um mehr als 13 Prozentpunkte besser. Beim SWV01-Benchmark-Problem sind alle WbsGT-Konfigurationen näher an der Lösung des Branch-and-Bound-Verfahrens als das Shifting-Bottleneck-Verfahren. Ebenso sind alle Lösungen von WbsGT besser als die gleichen Konfigurationen ohne Klassifizierer und mit zufälliger Regelauswahl. Weiterhin zeigen die Ergebnisse, dass die einzelnen Regeln LSO und MWR wie bei dem FT06- und ABZ7-Benchmark-Problem, die besten Einzellösungen liefern. Allerdings ist auch beim SWV01-Benchmark-Problem die aus zwei Elementen bestehende Regelmenge nicht groß genug, um gute Lösungen zu ermöglichen. Die Lösungen von WbsGT (*LSO MWR*) mit einer durchschnittlichen Abweichung von 18,08 Prozent liegen nur auf dem achten Rang und damit nur um etwa 2 Prozentpunkte unter den Lösungen des Giffler-Thompson-Verfahrens mit zufälliger Operationsauswahl. Die beste Konfiguration von WbsGT mit zwei Regeln liegt erst auf dem vierten Rang. Damit zeigt sich auch beim SWV01-Experiment, dass erst mit ausreichend großen Regelmengen genü-

gend Operationen aus der Konfliktmenge auswählbar sind, um so gute Lösungen zu erzielen.

Makespan CAR03: Im Folgenden sind die Ergebnisse des WBSGT-Verfahrens bei der Anwendung innerhalb des Flow-Shop-Benchmark-Problems CAR3[37] dargestellt. Flow-Shop-Probleme sind eine Spezialisierung von Job-Shop-Problemen. Ziel dieses Experiments ist zu untersuchen, wie gut die Lösungen des WBSGT-Verfahrens beim Einsatz in verwandten Organisationsformen sind.

Für das Experiment wurden 100 Iterationen mit abgewandelten Probleminstanzen gelöst. Die Parameter zur Änderung der Probleminstanzen waren wie folgt ausgeprägt:[38] $\mu = 0$, $\sigma = 50$ und $\kappa = 0$. Zum Training wurde das Problem mittels des Branch-and-Bound-Algorithmus in 10 abgewandelten Probleminstanzen gelöst. Das CAR3-Benchmark-Problem ist etwas komplexer als das FT06, jedoch konnte der Branch-and-Bound-Algorithmus hier die optimalen Lösungen berechnen. Abbildung 6.8 ziegt die Ergebnisse dieses Experiments. Es wird deutlich, dass WBSGT auch in Flow-Shop-Umgebungen gute Lösungen erzeugen kann.

Die Lösungen der einzelnen Verfahren beim CAR3-Benchmark-Problem sind in Tabelle 6.8 dargestellt. Für jede Variante sind Mittelwert und Varianz im Diagramm abgetragen.

Rang	Verfahren	Abweichung & Varianz		Regelmenge	Abw. & Var. ohne Klassifizierer	
1	WBSGT	10,58	5,74	SPT LSO MWR	34,63	10,72
2	WBSGT	14,07	6,70	SPT LPT LSO MSO LWR MWR	40,24	11,61
3	WBSGT	15,13	7,93	SPT LWR LPT MWR	38,16	12,96
4	WBSGT	17,26	8,66	SPT LPT	39,31	12,68
5	WBSGT	22,72	7,54	LSO MSO	42,77	10,81
6	WBSGT	24,98	11,83	LWR MWR	37,90	11,83
7	WBSGT	25,21	10,53	LWR MWR LSO MSO	42,62	11,00
8	Shifting-Bottleneck	30,44	10,53	–	–	–
9	WBSGT	33,65	9,74	LSO MWR	42,15	11,26
10	Giffler-Thompson	35,73	10,36	–	–	–
11	WBSGT	36,35	9,64	LWR MSO	43,39	11,26
12	MWR	40,93	12,60	–	–	–
13	LSO	42,77	10,81	–	–	–
14	MSO	42,77	10,81	–	–	–
15	LWR	50,28	12,00	–	–	–
16	SPT	55,07	15,03	–	–	–
17	LPT	72,37	16,88	–	–	–

Tabelle 6.8.: Makespan CAR3: Übersicht

[37] Siehe Tabelle A.2.
[38] Siehe Abschnitt 5.3.4.

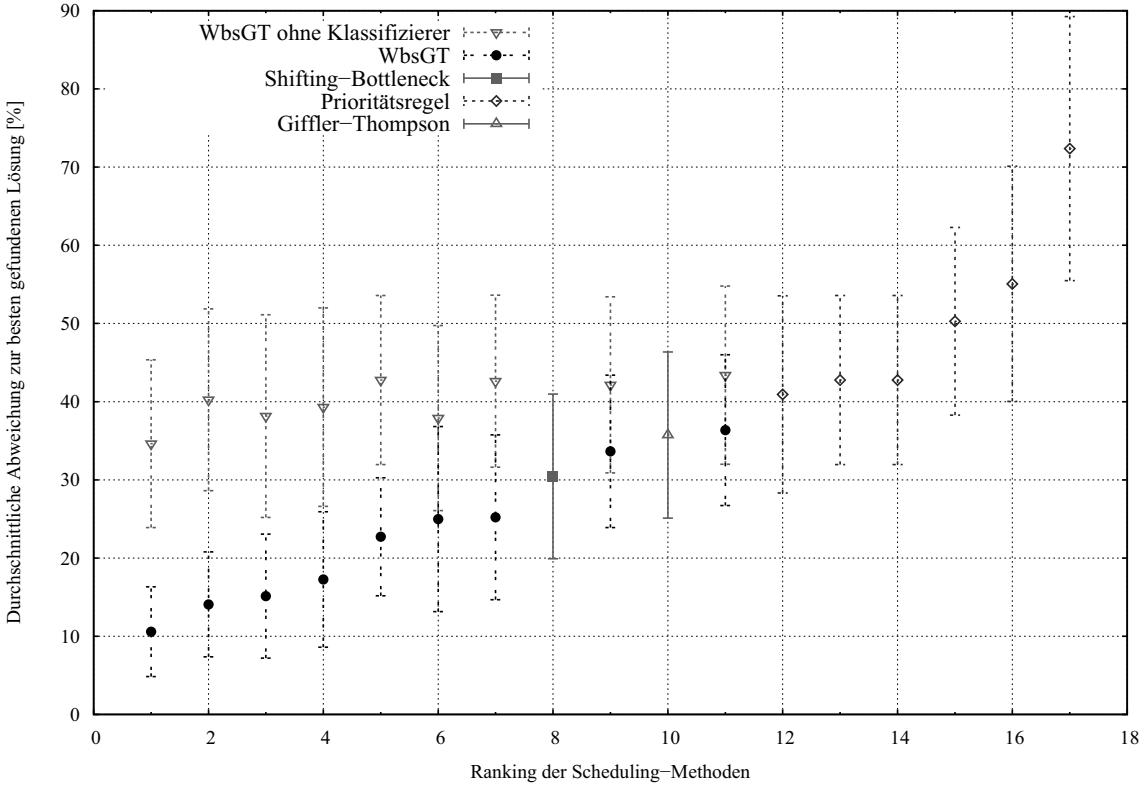


Abbildung 6.8.: Makespan CAR03: Benchmark WbsGT-Verfahren

Die Lösungen des Shifting-Bottleneck-Verfahren liegen mit einer durchschnittlichen Abweichung von 30,44 Prozent und einer Varianz von 10,53 auf dem achten Rang. Keine der einzelnen Prioritätsregeln erreicht bessere Lösungen als das Shifting-Bottleneck-Verfahren. Die Lösungen der Konfigurationen mit jeweils nur einer Regel belegen die letzten Plätze (12-17). Die erreichten Ergebnisse mit der Regel LPT weichen sogar um 72,37 Prozent bei einer Varianz von 16,88 von der optimalen Lösung ab. MWR und LSO bilden hier mit ca. 40 und 42 Prozent Abweichung noch die besten Ergebnisse. Das Giffler-Thompson-Verfahren mit zufälliger Operationsauswahl erreicht mit einer Abweichung von 35,73 Prozent und einer Varianz von 10,36 zwar eine ähnliche Güte wie das Shifting-Bottleneck-Verfahren, jedoch fällt die mittlere Abweichung um etwa 4 Prozentpunkte schlechter aus. Die beste Lösung des CAR3-Benchmark-Problems wird von WbsGT (*SPT LSO MWR*) mit einer durchschnittlichen Abweichung von 10,58 Prozent und einer Varianz von 5,74 berechnet. Auffällig ist, dass die Konfiguration ohne Klassifizierer eine mittlere Abweichung von 34,64 Prozent bei einer fast doppelt so großen Varianz erreicht. Dieser Unterschied der Abweichungen tritt bei sämtlichen Konfigurationen des WbsGT-Verfahrens beim CAR03-Problem auf. Damit wird besonders deutlich, dass der Einsatz des Klassifizierers zu großen Verbesserungen der Ergebnisse

führen kann. Die Ergebnisse zeigen, dass die einzelnen Regeln MWR und LSO, wie bei den zuvor untersuchten Benchmark-Problemen, die besten Einzellösungen liefern. Ebenfalls ist beim CAR3-Benchmark-Problem die aus zwei Elementen bestehende Regelmenge nicht groß genug, um gute Lösungen zu ermöglichen. Die Lösungen von WbsGT (*LSO MWR*) mit der durchschnittlichen Abweichung von 33,65 Prozent sind sogar die zweit schlechtesten in der Gruppe der wissensbasierten Lösungen und liegen unterhalb der Lösungsgüte des Shifting-Bottleneck-Verfahrens. Die beste WbsGT-Konfiguration mit zwei Regeln (SPT LPT) liegt auf Rang 4.

6.3. Evaluierung der dezentralen Verfahrensgüte in Flexible-Flow-Shops

Dieser Abschnitt evaluiert den dezentralen Einsatz des WbsGT-Verfahrens innerhalb eines Flexible-Flow-Shops. Nachfolgend wird das Generieren von Trainingsdaten mittels des Ablaufsimulators d^{3}FACT insight untersucht. Abschnitt 6.3.2 zeigt die Verfahrensgüte in Abhängigkeit der Größe der Trainingsmenge auf.

6.3.1. Benchmark: Simulatives Training

Zur Untersuchung der Lösungsqualität des dezentralen Einsatzes des WbsGT-Verfahrens in Flexible-Flow-Shop-Umgebungen wurden unterschiedliche Benchmark-Probleme behandelt. Die betrachteten Problemtypen mit uniformen Maschinen sind in Tabelle 6.9 dargestellt.

Bezeichnung	Fertigungsstufen	Maschinen pro Stufe	Mittlere Bearbeitungszeit[39]
1×3	1	3	35
2×4	2	4	40
3×2	3	2	32
3×3	3	3	20
4×4	4	4	25

Tabelle 6.9.: Untersuchte Flexible-Flow-Shop-Problemtypen

[39]Die Bearbeitungszeiten wurden mittels einer Normalverteilung mit der gegebenen mittleren Bearbeitungszeit um bis zu 40 Prozent verändert. Die Geschwindigkeiten der Maschinen wurden über eine Normalverteilung (*random*), die Werte zwischen 0 und 1 zurückliefert, erzeugt. Die Maximale Maschinengeschwindigkeit (*maxSpeed*) liegt bei 100 Prozent, d. h. Bearbeitungszeit des Jobs, die Minimale

Für das Experiment wurden jeweils sechs Varianten der Benchmark-Probleme erstellt. Hiervon ist je eine Variante 100-Mal zur Generierung von Beispiellösungen durch den Ablaufsimulator d³FACT insight per Zufall gelöst worden. Zum Training wurden die besten 10 Prozent der Ergebnisse verwendet.[40] Anschließend wurden die restlichen Benchmark-Probleme von den einzelnen Verfahren, d. h. WBSGT, *konstante Regeln*[41] und *zufällige Regelauswahl*[42] gelöst. Als Regeln wurde die in Abschnitt 5.3.3 entwickelten eingesetzt. Abbildung 6.9 illustriert die prozentualen Abweichungen des Makespans bei *konstanten Regeln SPT / fastest*, *SPT / slowest*, *LPT / fastest* und *LPT / slowest* sowie bei *zufälliger Regelauswahl*; jeweils im Vergleich zum durch WBSGT erzielten Referenzwert.[43] Wie aus den Werten hervorgeht, weichen die Lösungsgüten der Verfahren ohne Lernkomponente in fast allen Fällen deutlich nach oben von den wissensbasierten Lösungen ab. Eine Ausnahme stellt lediglich das, mit den Prioritätsregeln *LPT* für die Auswahl des Auftrags und *slowest* für die Auswahl der Maschine, erzielte Ergebnis im Fall des Problemtyps (1×3) dar. Es liegt 0,07 Prozent unter dem WBSGT-Wert.[44] Damit kann zunächst festgestellt werden, dass das entwickelte WBSGT-Verfahren in der Regel bessere Lösungen generiert als die Vergleichsverfahren mit fest eingestellten Regeln oder zufälliger Auswahl. Der Unterschied zu WBSGT und dem in diesem Experiment besten anderen Verfahren beträgt im Durchschnitt über alle Problemtypen und -varianten 4,62 Prozent.

Ferner zeigen die Evaluationsergebnisse, dass die anhand ihrer Lösungsgüte zu ermittelnde Rangfolge der verschiedenen nicht wissensbasierten Verfahren von Problemtyp zu Problemtyp sehr unterschiedlich ist. Zwar stellt die Ablaufplanung und -steuerung durch zufällige Auswahl der Prioritätsregeln nach der wissensbasierten Auswahl im Mittel das Zweitbeste der getesteten Verfahren dar, doch gilt dies bei einer einzelnen Betrachtung der Problemtypen nur in zwei von fünf Benchmark-Problemen. Bei dem Vergleich der verschiedenen, mit konstanten Regeln erzielten, Ergebnisse untereinander ist festzustellen, dass die Wahl der Prioritätsregel zur Auftragsauswahl einen deutlich

minSpeed bei 40 Prozent. Als Berechnungsformel für die Maschinengeschwindigkeiten wurde verwendet: $random \cdot (maxSpeed - minSpeed) + minSpeed$

[40] Der Parameter zur Bestimmung des Trainingsanteils ist frei wählbar. Ein Anteil 10 Prozent beschränkt die Trainingsmenge zwar stark, stellt jedoch sicher, dass eher gute Beispiele verwendet werden.

[41] Die Verfahrensausprägungen mit statischen Regeln (Jobauswahlregel / Maschinenauswahlregel) sind: (SPT / slowest), (SPT / fastest), (LPT / slowest) und (LPT / fastest).

[42] Jede Problemvariante wurde 100-mal mit zufälliger Regelauswahl gelöst und das Ergebnis gemittelt.

[43] Die detaillierten Ergebnisse sind in Tabelle A.9 aufgeführt.

[44] Insgesamt ist festzustellen, dass die Vorteile des entwickelten WBSGT-Verfahrens bei den untersuchten einstufigen Problemen kaum zum Tragen kommen. Dieses erscheint logisch, da das entwickelte Verfahren seine Vorteilhaftigkeit vor allem aus der globalen Orientierung zieht, welche im Fall einstufiger Probleme kaum zu einem positiven Effekt führen kann. Das einstufige Problem ist zu klein, als das WBSGT zu deutlich besseren Lösungen kommen könnte als die anderen Verfahren.

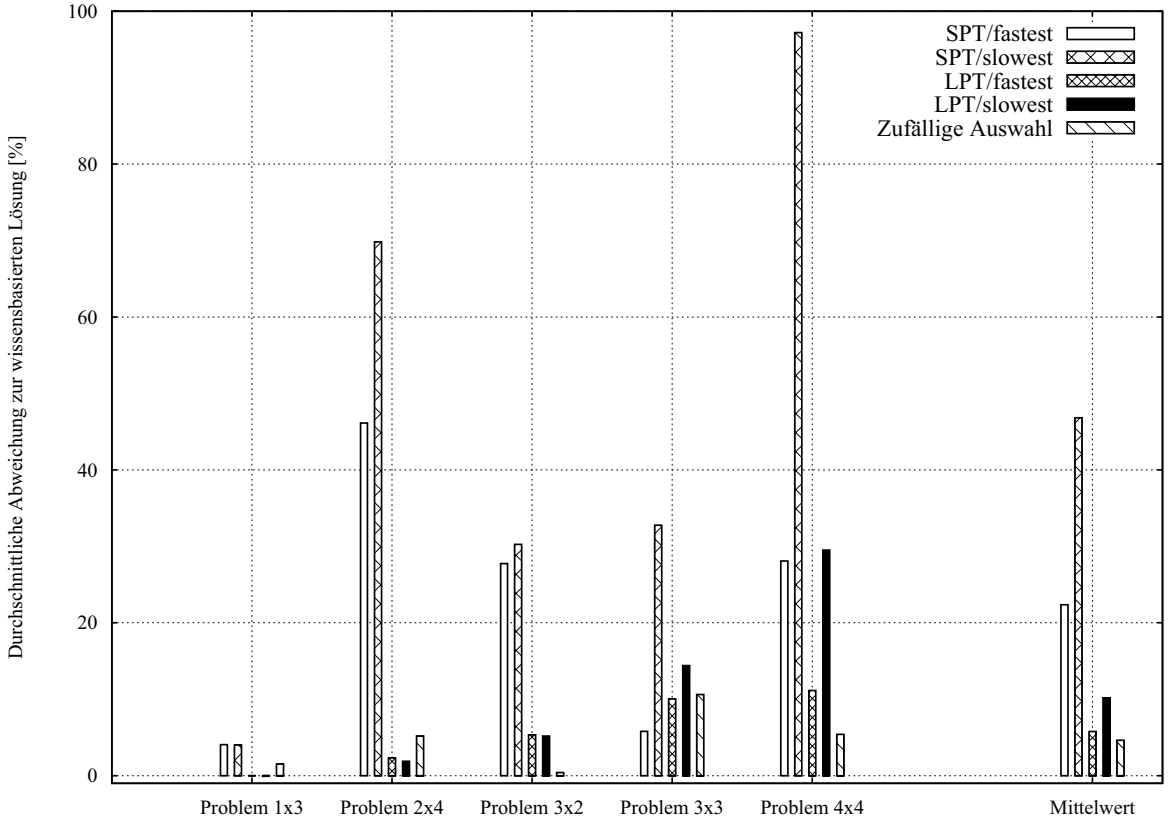


Abbildung 6.9.: Makespan simulatives Training: Benchmark WBSGT-Verfahren

größeren Einfluss auf die Lösungsgüte hat als die Wahl der Prioritätsregel für die Maschinenauswahl. Ist die zur Auftragsauswahl genutzte Regel LPT, so ist der Makespan in der Regel deutlich geringer als bei Nutzung der Regel SPT. Doch gilt dieses nicht für alle Problemtypen. Im Vergleich zu den anderen betrachteten Verfahren liefert lediglich das im Rahmen dieser Arbeit entwickelte WBSGT-Verfahren beim dezentralen Einsatz in Flexible-Flow-Shops gute und gleich bleibende Ergebnisse.

Daher kann mit Blick auf die Lösungsgüte festgehalten werden, dass das vorgestellte WBSGT-Verfahren zum adaptiven Scheduling in Flexible-Flow-Shop-Umgebungen mit uniformen Maschinen erhebliche Vorteile gegenüber den anderen getesteten Vorgehensweisen bietet. Ähnlich gute Ergebnisse wurden bei Flexible-Flow-Shops mit unverwandten Maschinen erreicht.[45]

[45] Siehe Anhang A.7.

6.3.2. Benchmark: Trainingsmenge

Als Trainingsverfahren für die Experimente in Flexible-Flow-Shop-Umgebungen wurde der Ablaufsimulator d^{3}FACT insight verwendet. Dieser garantiert keine optimalen Ergebnisse im Sinne der Zielfunktion und durch die zufällige Auswahl schwanken die Ergebnisse bei ein und derselben Scheduling-Probleminstanz von Simulationslauf zu Simulationslauf. Aus diesem Grund beeinflusst insbesondere hier die Anzahl der Simulationsdurchläufe die Güte des erstellten Klassifizierers, da mit der Anzahl der zur Verfügung stehenden Simulationsergebnisse potentiell bessere Beispiele für das Training zur Verfügung stehen. Es wird nicht die gesamte Menge der Ergebnisse aus den Simulationen zum Training verwendet, sondern die besten zehn Prozent. Dabei gilt die Annahme, dass die nach sehr vielen Simulationsdurchläufen generierten Trainingsmengen besser sind als solche, die nach weniger Simulationsdurchläufen entstehen. Nach der Annahme steigt mit der Anzahl der Simulationsläufe, wenn der Anteil der als gut und damit zum Training verwendeten Simulationsdurchläufe konstant bleibt, die Güte der in der Trainingsmenge abgebildeten Entscheidungssituationen und damit die Genauigkeit des Klassifizierers; damit die Lösungsgüte.

Der Einfluss der Anzahl der Simulationsdurchläufe wurde beispielhaft für das Flexible-Flow-Shop-Problem $FQ3||C_{max}$[46] mit drei Fertigungsstufen und je drei Maschinen untersucht.[47] Hierbei wurde der Lernprozess nach 25, 50, 75 und 100 Simulationsdurchläufen unterbrochen. Anschließend ist auf der Grundlage des bis dahin gesammelten Wissens ein Scheduling für die Problemvarianten durchgeführt worden. Die sich dabei ergebenden Ergebnisse in Form des Makespans sind in Abbildung 6.10 dargestellt.

Das Experiment zeigt, dass im Durchschnitt – fett und nicht gepunktet dargestellte Linie in Abbildung 6.10 – eine Verbesserung des erreichten Zielfunktionswerts mit der Erhöhung der Anzahl von Simulationsdurchläufen einhergeht. Diese Verbesserung flacht jedoch tendenziell ab, da nach einer gewissen Anzahl der Mittelwert der Simulationsergebnisse konstant bleibt. Somit werden die Trainingsbeispiele ab diesem Punkt ebenfalls nicht mehr verbessert. Sowohl positive als auch negative Ausreißer fallen damit kaum ins Gewicht. Bei zwei der fünf getesteten Problemvarianten ist, ab einer über ein bestimmtes Maß hinausgehenden Erhöhung der Zahl der Simulationsdurchläufe, eine Verschlechterung des Ergebnisses zu beobachten. Dieser Effekt kann auf die Einflüsse der zufälligen Entscheidungen innerhalb der Simulation bzw. Overfitting-Effekte zurückgeführt wer-

[46] Siehe Tabelle 6.9.
[47] Siehe Benchmark-Problem 3×3 in Tabelle 6.9.

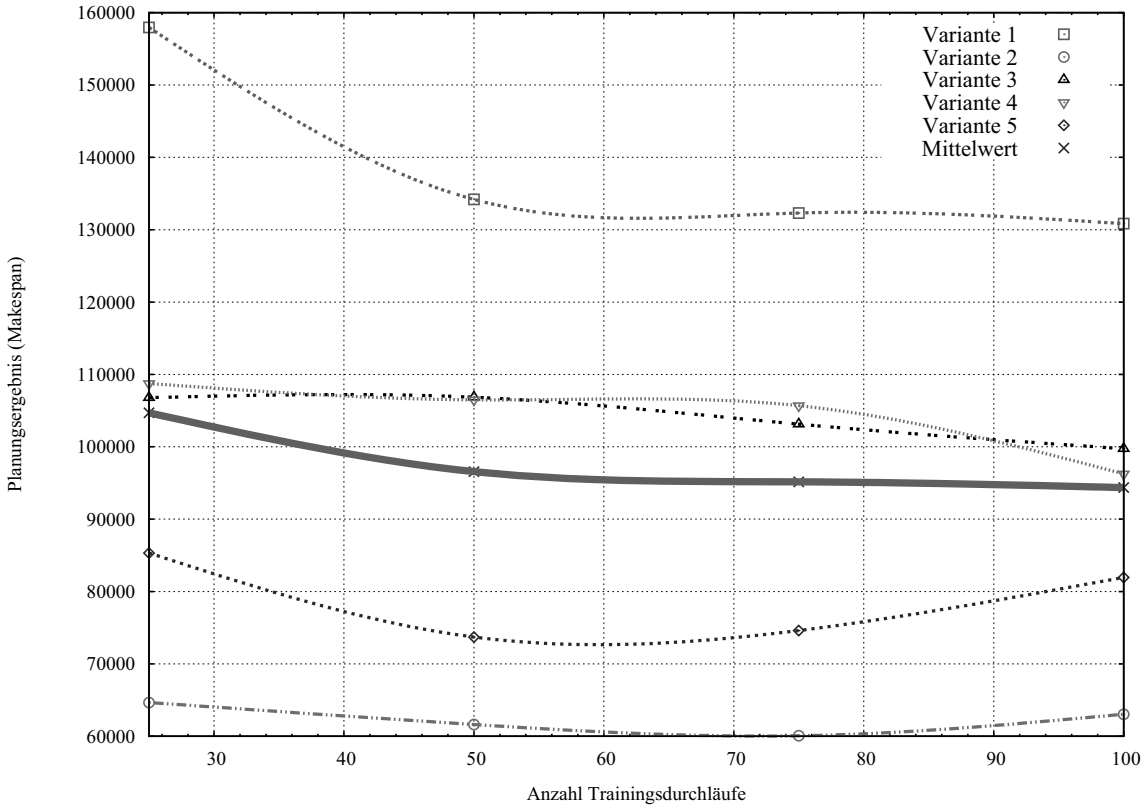


Abbildung 6.10.: Trainingsmenge: Simulationsläufe und Scheduling-Ergebnis

den.[48] Die Gesamtbetrachtung zeigt, dass mit steigender Anzahl der Trainingsdaten eine Verbesserung der Lösungsgüte einhergeht. Ab einem bestimmten Punkt ist die Durchführung weiterer Simulationen zum Training – je nach Anwendungsumgebung – möglicherweise nicht mehr durch die Verbesserung des erreichbaren Zielfunktionswerts gerechtfertigt.

Aus den in Abbildung 6.10 dargestellten Ergebnissen lässt sich folgern, dass eine Verkleinerung des Anteils der als gut eingestuften Ablaufpläne zum Training bei einer gleichzeitigen Erhöhung der Anzahl der Simulationsdurchläufe sinnvoll sein kann. Da der Klassifizierer offenbar ab einer bestimmten Anzahl zur Verfügung stehender Trainingsschedules – etwa durch Overfitting-Effekte – kaum noch von zusätzlichen Schedules profitieren kann, erscheint es demzufolge sinnvoll, die Simulationsergebnisse vor dem Lernprozess stärker zu filtern. Eine Erhöhung der Anzahl der Simulationsdurchläufe würde wahrscheinlich dazu führen, dass sich der durchschnittliche Zielfunktionswert der zum Lernen genutzten Schedules verbessert. Es ist zu erwarten, dass sich dadurch

[48] Bei dem aktuellen Trainingsanteil von 10 Prozent kann es vorkommen, dass die Trainingsbeispiele sich sehr ähnlich sind.

tendenziell eine Verbesserung der Ergebnisse mit trainiertem Klassifizierer erzielen ließe. Zur Untersuchung dieses Zusammenhangs wurde beispielhaft ein Experiment mit einem Flexible-Flow-Shop-Problem $FQ4||C_{max}$ mit vier Fertigungsstufen und je drei Maschinen durchgeführt. Ausgehend von den in Abbildung 6.10 dargestellten guten Ergebnissen bei einer Trainingsmengengröße von zehn Beispiellösungen, bestehen die Trainingsmengen bei diesem Experiment auch immer aus zehn Beispiellösungen. Die Anzahl der Trainingsläufe waren 800, 1.000 und 2.000, die der mit trainierter Lernkomponente gelösten Testprobleme jeweils 96.[49] Tabelle 6.10 illustriert die erreichten Ergebnisse.

Trainingsläufe	Trainingsfaktor	Makespan
800	80	436.816
1.000	100	421.987
2.000	200	410.800

Tabelle 6.10.: Qualität der Trainingsmenge

Die erzielten Ergebnisse stützen die aufgestellte Annahme. Die Ergebnisse können bei der Erhöhung von 800 auf 1.000 Simulationsdurchläufen um 3,4 Prozent gesteigert werden. Bei der Verdopplung von 1.000 auf 2.000 Durchläufe wurde die Güte des Ergebnisses nochmals um 2,7 Prozent erhöht.

Zusammenfassend kann festgestellt werden, dass die Annahme zutreffend ist. Mit der Erhöhung der Qualität der Trainingsbeispiele geht beim Einsatz der Lernkomponente eine Verbesserung des erreichbaren Ergebnisses einher.

[49] Zum Training wurden die x absolut besten Trainingsschedules mit x = Trainingsläufe/Trainingsfaktor verwendet.

7. Fazit

Die vorliegende Arbeit zum Thema *Situativ trainierte Regeln zur Ablaufsteuerung in Fertigungssystemen und ihre Integration in Simulationssysteme* behandelt den sensiblen unternehmerischen Problembereich der Optimierung der Fertigung. Die Überlegungen in der Arbeit setzen an der alltäglichen menschlichen Erfahrung an. Dem Menschen werden dauernd die unterschiedlichsten Entscheidungen abverlangt. Hierbei reicht es nicht aus, dass einfach nur entschieden wird, die Entscheidungen müssen auch gut sein. In Fertigungssystemen steht zur Entscheidung: welches Produkt soll hergestellt werden, wo ist die Fabrikationsanlage zu errichten, welcher Lieferant ist vorzuziehen? Einen eng mit dieser Frage verbundenen Faktor der Fertigung stellt das Problem der Geschwindigkeit der Entscheidung dar. Dabei stehen Geschwindigkeit und Qualitätsfrage der Lösung in einem Zielkonflikt. Das gilt insbesondere bei operativen Entscheidungen. Weil das zur Verfügung stehende Zeitintervall vom Auftreten eines Problems bis zu dessen Lösung immer kleiner wird, ist der Entscheidungsprozess immer schneller durchzuführen, wenn keine Handlungsoptionen verloren gehen sollen. Das Zeitproblem stellt sich vermehrt bei der Bildung von Auftragsreihenfolgen: Soll zuerst der Auftrag des Stammkunden bearbeitet werden oder erst der Auftrag des neuen Kunden, der viele neue Aufträge in Aussicht gestellt hat? Bei nur zwei Alternativen ist das Problem noch unkompliziert und damit kann eine gute Lösung schnell berechnet werden. Mit steigender Zahl der Variablen, die etwa durch zahlreiche Aufträge und unterschiedlich schnelle Maschinen gegeben ist, wird das Problem komplexer. Dieser Zuwachs an Komplexität vollzieht sich derart schnell, dass schon kleine Probleme nicht mehr schnell und optimal lösbar sind. Weil das Entscheidungsproblem immer größer wird, benötigen die Fertigungsplaner effektivere Unterstützungsmethoden zur Steuerung von Fertigungssystemen.

Der Stand der Technik von Methoden zur Ablaufplanung und -steuerung ist vielseitig, allerdings von unterschiedlicher Qualität. Optimierende Methoden sind selbst in verbesserter Form als Branch-and-Bound-Verfahren nicht einsetzbar, da sie die strengen Zeitrestriktionen zur Generierung einer Lösung nur bei kleinen, unrealistischen Problemen einhalten. Einfache Heuristiken wie Prioritätsregeln sind zwar in der Lage, schnell mögliche Reihenfolgelösungen zu finden, doch sind diese nur selten gut. Regeln sind je

nach Entscheidungssituation unterschiedlich gut zur Problemlösung geeignet. Die immer beste Regel existiert bis zum heutigen Tage nicht. Mit dem Einsatz von Ablaufsimulatoren können zwar für Zeitintervalle gut geeignete Regeln gefunden werden, ihr Nachteil besteht jedoch darin, dass die Entscheidungsgüte der Regeln für das Zeitintervall nur gemittelt wird. Treten im Intervall ungeplante Vorgänge auf, so ist die Simulation zu wiederholen. Ein zusätzliches Problem stellt die Bestimmung der korrekten Intervallgröße dar. Die für die jeweilige Entscheidungssituation passende Regelauswahl ist nicht möglich. Andere Heuristiken wie das Shifting-Bottleneck-Verfahren erzeugen zwar mögliche Lösungen und das auch relativ schnell, allerdings ist auch hier die Lösungsgüte aufgrund des eingegangenen Kompromisses zwischen Lösungsgüte und -geschwindigkeit im Allgemeinen nicht zufriedenstellend. Metaheuristiken, wie Verfahren der Künstlichen Intelligenz versuchen das Problem mit Hilfe von naturanalogen Vorgehensweisen zu lösen. Zur Optimierung der Ergebnisse wird mit künstlichen Mutationen die generierte Lösung verändert. In Expertensystemen soll das Wissen von Planungsexperten verallgemeinert werden. Allerdings ist das Erfahrungswissen der Experten schwer zu formalisieren. Resümierend ist festzustellen: Alle genannten Verfahren weisen Schwächen auf. Die vorliegende Arbeit weist nach, dass es bessere Lösungsmethoden als die bisher geläufigen gibt.

In der Arbeit ist ein Verfahren (WBSGT) zur Ablaufplanung und -steurung in zeitkritischen Situationen entwickelt worden. Grundlegede Steuerungsheuristik bildet hier das von *Giffler und Thompson* entwickelte Verfahren, welches für die untersuchten Organisationsformen Job-Shops und Flexible-Flow-Shops einerseits zu korrigieren und andererseits zu erweitern war. Die Heuristik eignet sich insbesondere deshalb, weil sie dem Planungsproblem der Maschinenbelegung in linearer Zeit eine Lösung zuführen kann. Zusätzlich werden von der Heuristik nur aktive Ablaufpläne erzeugt, die den zu durchsuchenden Lösungsraum erheblich einschränken. Der im Sinne der verfolgten Zielfunktion optimale Ablaufplan ist immer auch ein aktiver. Zusätzlich bildet die Heuristik in Situationen mit Entscheidungsnotwendigkeit eine Konfliktmenge der in Frage kommenden Auftragskandidaten. Um diese Menge gut aufzulösen, wurde ein maschinelles Lernverfahren integriert. Dieses trifft mittels der Beschreibung von Steuerungssituationen durch Merkmale die Entscheidung für die in der jeweiligen Situation passendste Prioritätsregel. Es steuert WBSGT damit durch den Lösungsraum, womit nur ein Pfad, der zu einer guten Lösung führt, zu durchlaufen ist. Hierzu wurden für den wissensbasierten Naive-Bayes-Klassifizierer prototypisch situationsbeschreibende Merkmale entwickelt. Geeignete Prioritätsregeln sind ausgewählt bzw. konzipiert worden. Für das Training des wissensbasierten Entscheiders wird der Zeitraum vor Beginn des Fertigungsprozesses genutzt, indem beispielhafte Trainingslösungen berechnet werden. Ebenso ist das

entwickelte System parallel zur laufenden Fertigung ständig adaptierbar. Als Trainingsverfahren sind u. a. optimierende Verfahren sowie der Ablaufsimulator d^{3}FACT insight eingesetzt worden. Letzterer erforderte aufgrund der nicht optimalen Lösungen vor der Aufnahme der erzeugten Trainingsbeispiele eine Vorbetrachtung.

Mittels Rückkopplung des entwickelten Verfahrens in die Ablaufsimulation ist diese um eine bisher nicht vorhandene Steuerungskomponente erweitert worden. Zusätzlich ermöglicht die Simulation eine Erweiterung der Informationen zum Entscheidungszeitpunkt. Dazu erzeugt die simulative Vorausschau zum Entscheidungszeitpunkt noch nicht bekannte Informationen in zeitlicher und maschineller Dimension.

Das Verfahren wurde prototypisch umgesetzt und anschließend evaluiert. Eine Untersuchung der theoretischen Lösungsqualität ohne situationsbeschreibende Merkmale hat aufgezeigt, dass bei hinreichend hoher Klassifikationsgenauigkeit optimale Lösungen erreicht werden können. Die Güte der Verfahrensergebnisse wurde anhand von Benchmarks untersucht und führte zu durchweg guten Ergebnissen. Im Vergleich mit alternativen Lösungsverfahren wie einer Shifting-Bottleneck-Implementierung, einem Ablaufsimulator (d^{3}FACT insight), der naiven Giffler-Thompson-Heuristik oder der einfachen / zufälligen Anwendung der Prioritätsregeln sind substanziell bessere Ergebnisse im Sinne der verfolgten Zielfunktion (hier Makespan) erreicht worden. Dabei liegt die erforderliche Rechenzeit des Verfahrens im Sekundenbereich und ist damit deutlich geringer als die der Referenz- oder Trainingsverfahren. Durch die Evaluierung konnte die Tauglichkeit des Verfahrens nachgewiesen werden.

Offene Forschungsfragen sind beispielsweise Merkmale, die Situationen mit Steuerungsnotwendigkeit beschreiben. Die Verfahrensintegration in ein PPS-System für den praktischen Einsatz ist zu untersuchen. Ebenso sind weitere Forschungsarbeiten im Bereich der Prioritätsregeln zur Auswahl in Konfliktsituationen durchzuführen. Weiterhin ist zu untersuchen, welche Methoden zur Bestimmung und zur Behandlung von zu unsicher klassifizierten Entscheidungen eingesetzt werden können. Für die Merkmalsgenerierung und -evaluierung notwendige Methoden wurden bereits umgesetzt. Die Erweiterung des Verfahrens um eine Komponente zur Betriebsdatenerfassung in Echtzeit wurde in Grundlagen vorbereitet und durch erste Experimente evaluiert. Dieses Konzept ist weiter auszuarbeiten und vollends umzusetzen.

Literaturverzeichnis

Acero-Domínguez, M. J. und Paternina-Arboleda, C. D.: Scheduling jobs on a k-stage flexible flow shop using a toc-based (bottleneck) procedure. In: Jones, M. H.; Patek, S. D. und Tawney, B. E. (Hrsg.) *Proceedings of the 2004 Systems and Information Engineering Design Symposium*. 2004, Seiten 295–298.

Adams, J.; Balas, E. und Zawack, D.: The shifting bottleneck procedure for job shop scheduling. In: *Management Science*, Band 34(3), Seiten 391–401, 1988.

Aytug, H.; Bhattacharyya, S.; Koehler, G. J. und Snowdon, J. L.: A review of machine learning in scheduling. In: *IEEE Transactions on Engineering Management*, Band 41, Seiten 165–171, 1994.

Aytug, H.; Lawley, M. A.; McKay, K.; Mohan, S. und Uzsoy, R.: Executing production schedules in the face of uncertainties: A review and some future directions. In: *European Journal of Operational Research*, Band 161, Seiten 86–110, 2005.

Beasley, J. E.: Job shop scheduling. `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html`, 1990a. Stand 14. November 2008.

Beasley, J. E.: OR-library: Distributing test problems by electronic mail. In: *The Journal of the Operational Research Society*, Band 41(11), Seiten 1069–1072, 1990b.

Beierle, C. und Kern-Isberner, G.: *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. Springer. Vieweg+Teubner / GWV Fachverlage GmbH Wiesbaden, Wiesbaden, 4., verbesserte Auflage, 2008.

Bierwirth, C.: A generalized permutation approach to job shop scheduling with genetic algorithms. In: *OR Spectrum*, Band 17, Seiten 87–92, 1995.

Blackstone Jr., J. H.; Phillips, D. T. und Hogg, G. L.: State-of-the-art survey of dispatching rules for manufacturing job shop operations. In: *International Journal of Production Research*, Band 20(1), Seiten 27–45, 1982.

Blazewicz, J.; Ecker, K. H.; Pesch, E.; Schmidt, G. und Weglarz, J.: *Handbook of Scheduling. From Theory to Applications*. Springer, Berlin, Heidelberg, 2007.

Blazewicz, J.; Lenstra, J. K. und Rinnooy Kan, A. H. G.: Scheduling subject to reresource constraints: classification and complexity. In: *Discrete Appl. Math. 5*, Springer, Seiten 11–24. 1983.

Blumer, A.; Ehrenfeucht, A.; Haussler, D. und Warmuth, M. K.: Occam's razor. In: *Information processing letters*, Band 24(6), 1987.

Bräsel, H.; Dornheim, L.; Kutz, S.; Mörig, M. und Rössling, I.: LiSA - A Library of Scheduling Algorithms, Einführung zu LiSA. Technischer Bericht, Otto-von-Guericke Universität Magdeburg, Fakultät der Mathematik, 2003.

Brockmann, K. und Dangelmaier, W.: A parallel branch & bound algorithm for make-span optimal sequencing in flow shops with parallel machines. In: *In Proc. 2nd IMACS International Multiconference on Computational Engineering in Systems Applications (CESA)*. 1998, Seiten 431–436.

Brucker, P.: *Scheduling Algorithms*. Springer, Berlin, u. a., 5. Auflage, 2007.

Brucker, P.; Jurisch, B. und Sievers, B.: A branch and bound algorithm for the job-shop scheduling problem. In: *Discrete Appl. Math.*, Band 49(1–3), Seiten 107–127, 1994.

Brucker, P. und Knust, S.: Complexity results for scheduling problems. Online.

Canbolat, Y. B. und Gundogar, E.: Fuzzy priority rule for job shop scheduling. In: *Journal of Intelligent Manufacturing*, Band 15(4), Seiten 527–533, 2004.

Carlier, J.: Ordonnancements à contraintes disjonctives. In: *R.A.I.R.O. Recherche operationelle/Operations Research*, Band 12, Seiten 333–351, 1978.

Chang, S.-C. und Liao, D.-Y.: Scheduling flexible flow shops with no setup effects. In: *The 1992 IEEE International Conference on Robotics and Automation*. 1992, Band 2, Seiten 1179–1184.

Chang, Y.-L. und Sullivan, R. S.: Schedule generation in a dynamic job shop. In: *International Journal of Production Research*, Band 28(1), Seiten 65–74, 1990.

Chen, C. C. und Yih, Y.: Identifying attributes for knowledge-based development in dynamic scheduling environments. In: *International Journal of Production Research*, Band 34(6), Seiten 1739–1755, 1996.

Church, L. K. und Uzsoy, R.: Analysis of periodic and event-driven rescheduling policies in dynamic shops. In: *International Journal of Computer Integrated Manufacturing*, Band 5(3), Seiten 153–163, 1992.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. und Stein, C.: *Introduction to Algorithms*. McGraw-Hill, Boston Burr Ridge , IL Dubuque , IA Madison , WI New York San Francisco St. Louis, 2. Auflage, 2001.

Dabbas, R. M. und Fowler, J. W.: A new scheduling approach using combined dispatching criteria in wafer fabs. In: *Semiconductor Manufacturing, IEEE Transactions on Semiconductor Manufacturing*, Band 16(3), Seiten 501–510, 2003.

Dangelmaier, W.: *Fertigungsplanung*. Springer, Berlin, u. a., 2001.

Dell'Amico, M. und Trubian, M.: Applying tabu search to the job-shop scheduling problem. In: *Annals of Operations Research*, Band 41, Seiten 231–252, 1993.

Domschke, W.; Scholl, A. und Voss, S.: *Produktionsplanung: Ablauforganisatorische Aspekte*. Springer-Lehrbuch. Springer, Berlin, 2., überarb. und erw. Auflage, 1997.

Dorigo, M. und Blum, C.: Ant colony optimization theory: a survey. In: *Theoretical Computer Science*, Band 344(2–3), Seiten 243–278, 2005.

Dorigo, M. und Stützle, T.: *Ant Colony Optimization*. MIT Press, 2004.

Dréo, J.; Pétrowski, A.; Siarry, P. und Taillard, E.: *Metaheuristics for hard optimization: Methods and Case Studies*. Springer, 2006.

Engesser, H. (Hrsg.): *Duden Informatik: ein Sachlexikon für Studium und Praxis*. Dudenverlag, Mannheim, u.; 1988.

Evers, K.: *Simulationsgestützte Belegungsplanung in der Multiressourcen-Montage*. Universität Hannover, Dissertation, 2002.

Fang, H.-L.; Ross, P. und Corne, D.: A promising genetic algorithm approach to job-shop scheduling, re-scheduling, and open-shop scheduling problems. In: Forrest, S. (Hrsg.) *Proc. of the Fifth Int. Conf. on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993, Seiten 375–382.

Fayyad, U. M. und Irani, K. B.: On the handling of continuous-valued attributes in decision tree generation. In: *Machine Learning*, Band 8(1), 1992.

Fayyad, U. M.: *On the induction of decision trees for multiple concept learning*. Dissertation, University of Michigan, Ann Arbor, MI, USA, 1992.

Fisher, M. L. und Thompson, G. L.: Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J. und Thompson, G. (Hrsg.) *Industrial Scheduling*, Prentice-Hall, NY, Seiten 225–251. 1963.

Fleischmann, B.: Operations Research für die Produktion. In: Kern, W.; Schröder, H.-H. und Weber, J. (Hrsg.) *Handwörterbuch der Produktionswirtschaft*, Schäffer-Poeschel, Stuttgart, Seiten 1357–1370. 2., völlig neu gestaltete Auflage, 1998.

French, S.: *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Ellis Horwood series in mathematics and its applications. Ellis Horwood Limited, Chichester, 1982.

Garey, M. R. und Johnson, D. S.: *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, USA, 1990.

Georgi, G.: *Job Shop scheduling in der Produktion: Einsatzorientierte Lösungen für ein Kernproblem der Produktionsplanung und -steuerung bei mittleren Auftragszahlen und variierenden Einsatzbedingungen*, Band 11 *Wirtschaftswissenschaftliche Beiträge*. Physica-Verlag, Heidelberg, 1995.

Giffler, B. und Thompson, G. L.: Algorithms for solving production scheduling problems. In: *Operations Research*, Band 8(4), Seiten 487–503, 1960.

Görz, G.; Rollinger, C.-R. und Schneeberger, J. (Hrsg.): *Handbuch der künstlichen Intelligenz*. Oldenbourg, München, u. a., 4., korrigierte Auflage, 2003.

Graham, R. L.; Lawler, E. L.; Lenstra, J. K. und Rinnooy Kan, A. H. G.: Optimization and approximation in deterministic sequencing and scheduling: A survey. In: *Annals Discrete Mathematics*. 1979, Band 5, Seiten 287–326.

Günther, H.-O. und Tempelmeier, H.: *Produktion und Logistik*. Springer, Berlin, u. a., 6. Auflage, 2005.

Gutenberg, E.: *Die Produktion*. Nummer Band 1 in Grundlagen der Betriebswirtschaftslehre. Springer, Berlin, u. a., 1. Auflage, 1951.

Hackstein, R.: *Produktionsplanung und -steuerung*. VDI-Verlag, Düsseldorf, 2. Auflage, 1989.

Halevi, G. und Wang, K.: Knowledge based manufacturing system (kbms). In: *Journal of Intelligent Manufacturing*, Band 18(4), Seiten 467–474, 2007.

Haupt, R.: Prioritätsregeln für die Reihenfolgeplanung. In: Kern, W.; Schröder, H.-H. und Weber, J. (Hrsg.) *Handwörterbuch der Produktionswirtschaft*, Schäffer-Poeschel, Stuttgart, Seiten 1418–1426. 2., völlig neu gestaltete Auflage, 1996.

Henning, A.: *Praktische Job-Shop Scheduling-Probleme*. Thüringer Universitäts- und Landesbibliothek, Jena, 2002.

Hopp, W. J. und Spearman, M. L.: *Factory physics: foundations of manufacturing management*. Irwin/McGraw-Hill, Boston, u. a., 3. Auflage, 2007.

Huang, P.-Y.; Hong, T.-P.; Horng, G. und Kao, C.-Y.: Extending the palmer algorithm to solve group flexible flow-shop problems. In: *Proc. 30th Annual Conference of IEEE Industrial Electronics Society IECON 2004*. 2004, Band 2, Seiten 1902–1907.

Ishii, N. und Talavage, J. J.: A transient-based real-time scheduling algorithm in fms. In: *International Journal of Production Research*, Band 29/12, Seiten 2501–2510, 1991.

Jain, A. S. und Meeran, S.: A state-of-the-art review of job-shop scheduling techniques. Technischer Bericht, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, 1998.

Jain, A. S. und Meeran, S.: Deterministic job-shop scheduling: Past, present, and future. In: *European Journal of Operational Research*, Band 113(2), Seiten 390–434, 1999.

Jansen, K.; Solis-Oba, R. und Sviridenko, M.: Makespan minimization in job shops: a polynomial time approximation scheme. In: *Annual ACM Symposium on Theory of Computing: Proceedings of the thirty-first annual ACM symposium on Theory of computing*. ACM Press, New York, NY, USA, 1999, Seiten 394–399.

Jeong, K. C.: A real-time scheduling mechanism for a flexible manufacturing system using simulation and dispatching rules. In: *International journal of production research*, Band 36(9), Seiten 2609–2626, 1998.

Jungwattanakit, J.; Reodecha, M.; Chaovalitwongse, P. und Werner, F.: Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. In: *The International Journal of Advanced Manufacturing Technology*, Band 37(3), Seiten 354–370, 2008.

Kern, W.; Schröder, H.-H. und Weber, J. (Hrsg.): *Handwörterbuch der Produktionswirtschaft*. Schäffer-Poeschel, Stuttgart, 2., völlig neu gestaltete Auflage, 1996.

Kim, M. H. und Kim, Y.-D.: Simulation-based real-time scheduling in a flexible manufacturing system. In: *Journal of Manufacturing Systems*, Band 13/2, Seiten 85–93, 1994.

Klaus, G. und Buhr, M. (Hrsg.): *Philosophisches Wörterbuch*, Band 2. Lamaismus bis Zweckmäßigkeit. Verlag Das Europ. Buch, Westberlin, 14. Auflage, 1987.

Klösgen, W. und Żytkow, J. M.: *Handbook of data mining and knowledge discovery*. Oxford Univ. Press, Oxford, u. a., 2002.

Knauer, S.: *Neue heuristische Methoden zur Optimierung des Fertigungsablaufs*. Logos-Verl., Berlin, 1. Auflage, 2002.

Kurbel, K. und Rohmann, T.: Ein Vergleich von Verfahren zur Maschinenbelegungsplanung: Simulated Annealing, Genetische Algorithmen und mathematische Optimierung. In: *Wirtschaftsinformatik*, Band 37(6), Seiten 581–593, 1995.

Kwak, C. und Yih, Y.: Data-mining approach to production control in the computer-integrated testing cell. In: *IEEE Transactions on Robotics and Automation*, Band 20(1), Seiten 107–116, 2004.

Laroque, C.: *Ein mehrbenutzerfähiges Werkzeug zur Modellierung und richtungsoffenen Simulation von wahlweise objekt- und funktionsorientiert gegliederten Fertigungssystemen*. Universität Paderborn, Dissertation, Paderborn, 2007.

Lawrence, S. R.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). 1984.

Lawrence, S. R. und Sewell, E. C.: Heuristic, optimal, static, and dynamic schedules when processing times are uncertain. In: *Journal of Operations Management*, Band 15(1), Seiten 71–82, 1997.

Lee, Y. F.; Jiang, Z. B. und Liu, H. R.: Multiple-objective scheduling and real-time dispatching for the semiconductor manufacturing system. In: *Computers & Operations Research*, Band 36(3), Seiten 866–884, 2009.

Leung, J. Y.-T.: Introduction and Notation. In: Leung, J. Y.-T. (Hrsg.) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapman & Hall/CRC, Boca Raton, u. a., Kapitel 1-1. 2004.

Li, X.; Liu, L. und Wu, C.: A fast method for heuristics in large-scale flow shop scheduling. In: *Tsinghua Science & Technology*, Band 11(1), Seiten 12–18, 2006.

Liu, T.-K.; Tsai, J.-T. und Chou, J.-H.: Improved genetic algorithm for the job-shop scheduling problem. In: *The International Journal of Advanced Manufacturing Technology*, Band 27, Seiten 1021–1029, 2006.

Lo, M.-C.; Chen, J.-S. und Chang, Y.-F.: Two-machine flexible flow-shop scheduling with setup times. In: *Journal of Applied Sciences*, Band 8(12), Seiten 2217–2225, 2008.

Lödding, H.: *Verfahren der Fertigungssteuerung: Grundlagen, Beschreibung, Konfiguration*. Springer, Berlin, 2008.

Mahajan, K. R.: *A combined simulation and optimization based method for predictive - reactive scheduling of flexible production systems subject to execution exceptions*. Dissertation, Paderborn, Univ., Diss., 2007.

Mahl, A. und Krikler, R.: Approach for a rule based system for capturing and usage of knowledge in the manufacturing industry. In: *Journal of Intelligent Manufacturing*, Band 18(4), Seiten 519–526, 2007.

Manivannan, S. und Banks, J.: Design of a knowledge-based on-line simulation system to control a manufacturing shop floor. In: *IIE Transactions*, Band 24(3), Seiten 72–83, 1992.

Mathirajan, M. und Sivakumar, A. I.: A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. In: *The International Journal of Advanced Manufacturing Technology*, Band 29(9), Seiten 990–1001, 2006.

Metan, G. und Sabuncuoglu, I.: A simulation based learning meachanism for scheduling systems with continuous control and update structure. In: Kuhl, M. E.; Steiger, M.; Armstrong, F. B. und Joines, J. A. (Hrsg.) *WSC '05: Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference, Orlando, Florida, 2005, Seiten 2148–2156.

Mingers, J.: An empirical comparison of pruning methods for decision tree induction. In: *Machine Learning*, Band 4(2), Seiten 227–243, 1989a.

Mingers, J.: An empirical comparison of selection measures for decision-tree induction. In: *Machine Learning*, Band 3(4), Seiten 319–342, 1989b.

Mitchell, T. M.: *Machine learning*. McGraw-Hill, New York, u. a., 1997.

Mönch, L.: *Agentenbasierte Produktionssteuerung komplexer Produktionssysteme*. Deutscher Universitäts-Verlag, Ilmenau, 2006.

Nascimento, M. A.: Giffler and Thompson's algorithm for job shop scheduling is still good for flexible manufacturing systems. In: *The journal of the Operational Research Society*, Band 44(5), Seiten 521–524, 1993.

Nissen, V.: *Einführung in Evolutionäre Algorithmen. Optimierung nach dem Vorbild der Evolution*. Vieweg, 1997.

Nyhuis, P.; Hartmann, W. und Fischer, A.: Der Einfluss von Prioritätsregeln auf logistische Zielgrößen. In: *Productivity management*, Band 14(3), Seiten 17–20, 2009.

O'Kane, J. F.: A knowledge-based system for reactive scheduling decision-making in fms. In: *Journal of Intelligent Manufacturing*, Band 11(5), Seiten 461–474, 2000.

Panwalkar, S. S. und Iskander, W.: A survey of scheduling rules. In: *Operations Research*, Band 25(1), Seiten 45–61, 1977.

Park, S. C.; Raman, N. und Shaw, M. J.: Adaptive scheduling in dynamic flexible manufacturing systems: a dynamic rule selection approach. In: *IEEE Transactions on Robotics and Automation*, Band 13(4), Seiten 486–502, 1997.

Paternina-Arboleda, C.; Montoya-Torres, J.; Acero-Dominguez, M. und Herrera-Hernandez, M.: Scheduling jobs on a k -stage flexible flow-shop. In: *Annals of Operations Research*, 2007.

Pinedo, M. L.: *Planning and scheduling in manufacturing and services*. Springer series in operations research. Springer, New York, NY, 2005.

Piramuthu, S.; Raman, N. und Shaw, M. J.: Learning-based scheduling in a flexible manufacturing flow line. In: *IEEE Transactions on Engineering Management*, Band 41(2), Seiten 172–182, 1994.

Piramuthu, S.; Raman, N.; Shaw, M. J. und Park, S. C.: Integration of simulation modeling and inductive learning in an adaptive decision support system. In: *Decision Support Systems*, Band 9(1), Seiten 127–142, 1993.

Piramuthu, S.; Shaw, M. und Fulkerson, B.: Information-based dynamic manufacturing system scheduling. In: *International Journal of Flexible Manufacturing Systems*, Band 12(2), Seiten 219–234, 2000.

Popper, K. R.: *Logik der Forschung*. Mohr, Tübingen, nachdr. der 10. verbesserten u. vermehrten Auflage, 2002.

Priore, P.; de la Fuente, D.; Gomez, A. und Puente, J.: A review of machine learning in dynamic scheduling of flexible manufacturing systems. In: *(AI EDAM) Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Band 15(3), Seiten 251–263, 2001.

Priore, P.; de la Fuente, D.; Puente, J. und Parreno, J.: A comparison of machine-learning algorithms for dynamic scheduling of flexible manufacturing systems. In: *Engineering Applications of Artificial Intelligence*, Band 19(3), Seiten 247–255, 2006.

Quinlan, J. R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Francisco, 1. Auflage, 1993.

Reese, J.: Kapazitätsbelegungsplanung. In: Kern, W.; Schröder, H.-H. und Weber, J. (Hrsg.) *Handwörterbuch der Produktionswirtschaft*, Schäffer-Poeschel, Stuttgart, Seiten 862–873. 2., völlig neu gestaltete Auflage, 1996.

Rixen, I.: *Maschinenbelegungsplanung mit evolutionären Algorithmen.* Deutscher Universitäts-Verlag, Wiesbaden, 1997.

Ruiz, R. und Maroto, C.: A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. In: *European Journal of Operational Research*, Band 169(3), Seiten 781–800, 2006.

Russel, S. und Norvig, P.: *Artificial Intelligence: A Modern Appproch.* 2003.

Sabuncuoglu, I. und Bayiz, M.: Analysis of reactive scheduling problems in a job shop environment. In: *European Journal of Operational Research*, Band 126(3), Seiten 567–586, 2000.

Sauer, J.: *Multi-Site Scheduling.* Habiliationsschrift, Universität Oldenburg, 2002.

Schekelmann, A.: *Materialflußsteuerung auf der Basis des Wissens mehrerer Experten*, Band 47. Heinz-Nixdorf-Inst., Univ.-GH Paderborn, Paderborn, 1. Auflage, 1999.

Schneeweiß, C.: *Einführung in die Produktionswirtschaft.* Springer, Berlin, u. a., 8., verb. und erw. Auflage, 2002.

Scholz-Reiter, B.; de Beer, C.; Freitag, M.; Hamann, T.; Rekersbrink, H. und Topi Tervo, J.: Dynamik logistischer Systeme. In: Nyhuis, P. (Hrsg.) *Beiträge zu einer Theorie der Logistik*, Springer, Berlin, Seiten 109–138. 2008.

Scholz-Reiter, B.; Jagalski, T. und Bendul, J.: Bienenalgorithmen zur Selbststeuerung logistischer Prozesse. In: *Industrie Management*, Band 5(Oktober 2007), Seiten 7–10, 2007.

Schoner, P.: *Operative Produktionsplanung in der verfahrenstechnischen Industrie.* kassel university press GmbH, Kassel, 2008.

Schultz, J.; Weigelt, M. und Mertens, P.: Verfahren für die rechnerunterstützte produktionsfeinplanung. ein Überblick. In: *Wirtschaftsinformatik*, Band 37(6), Seiten 594–608, 1995.

Semini, M.; Fauske, H. und Strandhagen, J. O.: Applications of discrete-event simulation to support manufacturing logistics decision-making: A survey. In: Perrone, L. F.; Wieland, F. P.; Liu, J.; Lawson, B. G.; Nicol, D. M. und Fujimoto, R. M. (Hrsg.) *WSC '06: Proceedings of the 38th conference on Winter simulation.* IEEE Computer Society, Los Alamitos, CA, USA, 2006, Seiten 1946–1953.

Shaw, M. J.; Park, S. C. und Raman, N.: Intelligent scheduling with machine learning capabilities: The induction of scheduling knowledge. In: *IIE Transactions*, Band 24(2), Seiten 156–168, 1992.

Shiue, Y.-R. und Guh, R.-S.: Learning-based multi-pass adaptive scheduling for a dynamic manufacturing cell environment. In: *Robotics and Computer-Integrated Manufacturing.* 2006, Band 22, Seiten 203–216.

Shmueli, G.; Patel, R. N. und Bruce, C. P.: *Data mining for business intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner*. Wiley-Interscience, Hoboken, NJ, 2007.

Storer, R. H.; Wu, S. D. und Vaccari, R.: New search spaces for sequencing problems with application to job shop scheduling. In: *Management Science*, Band 38(10), Seiten 1495–1509, 1992.

Stosik, D.: Mehrzielorientierte Ablaufplanung bei auftragsorientierter Werkstattfertigung. In: , Band 17, 2005.

Suhl, L. und Mellouli, T.: *Optimierungssyteme: Modelle, Verfahren, Software, Anwendungen*. Springer, Berlin, u. a., 2006.

Tan, P.-N.; Steinbach, M. und Kumar, V.: *Introduction to Data Mining*. Pearson Education, Boston, u. a., 2006.

Tavakkoli-Moghaddam, R. und Daneshmand-Mehr, M.: A computer simulation model for job shop scheduling problems minimizing makespan. In: *Selected Papers from The 30th International Conference on Computers Industrial Engineering*. 2005, Seiten 811–823.

Teich, T.: *Optimierung von Maschinenbelegungsplanungsplänen unter Benutzung heuristischer Verfahren*. Josef Eul Verlag, 1998.

Tempelmeier, H.: Flexible Fertigungstechniken. In: Kern, W.; Schröder, H.-H. und Weber, J. (Hrsg.) *Handwörterbuch der Produktionswirtschaft*, Schäffer-Poeschel, Stuttgart, Seiten 501–512. 2., völlig neu gestaltete Auflage, 1998.

van Rijsbergen, C.: *Information Retrieval*. Butterworth, London, 2. Auflage, 1979.

Vieira, G. E.; Herrmann, J. W. und Lin, E.: Rescheduling manufacturing systems: A framework of strategies, policies, and methods. In: *Journal of Scheduling*, Band 6(1), Seiten 39–62, 2003.

Vitiello, M.: *Fallbasierte Materialflußsteuerung - Ein Verfahren zur wissensbasierten Materialflußsteuerung*, Band 8: Meß-, Steuerungs- und Regelungstechnik. VDI-Verlag, Düsseldorf, 1997.

Wan, Y.-W.: Which is better, off-line or real-time scheduling? In: *International Journal of Production Research*, Band 33(7), Seiten 2053–2059, 1995.

Wang, H.: Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. In: *Expert Systems*, Band 22(2), Seiten 78–85, 2005.

Wang, K.: Applying data mining to manufacturing: the nature and implications. In: *Journal of Intelligent Manufacturing*, Band 18(4), Seiten 487–495, 2007.

Wiendahl, H.-P.: *Betriebsorganisation für Ingenieure*. Hanser, München, u. a., 6., aktualisierte Auflage, 2008.

Witten, I. H. und Frank, E.: *Data Mining: Praktische Werkzeuge und Techniken für das maschinelle Lernen*. Hanser Fachbuch, München, u. a., 2001.

Witten, I. H. und Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Amsterdam, u. a., 2. Auflage, 2005.

Wu, S.-Y. D. und Wysk, R. A.: An application of discrete-event simulation to online control and scheduling in flexible manufacturing. In: *International Journal of Production Research*, Band 27(9), Seiten 1603–1623, 1989.

Xu, Y.; Sannomiya, N. und Tian, Y.: A new method of shifting bottleneck with tabu search for solving flexible flow shop problems. In: *Transactions of the Society of Instrument and Control Engineers*, Band 39(9), Seiten 865–871, 2003.

Yoo, T.; Kim, D. und Cho, H.: A new approach to multi-pass scheduling in shop floor control. In: Ingalls, R. G.; Rossetti, M. D.; Smith, J. S. und Peters, B. A. (Hrsg.) *WSC '04: Proceedings of the 36th conference on Winter simulation*. Winter Simulation Conference, Washington, D.C., 2004, Seiten 1109–1114.

Zäpfel, G.: Grundlagen und möglichkeiten der gestaltung dezentraler pps-systeme. In: Corsten, H. und Gössinger, R. (Hrsg.) *Dezentrale Produktionsplanungs- und -steuerungs-Systeme: Eine Einführung in zehn Lektionen*, Kohlhammer, Stuttgart, u. a., Seiten 12–53. 1998.

Zäpfel, G. und Braune, R.: *Moderne Heuristiken der Produktionsplanung*. Vahlen, München, 1. Auflage, 2005.

Zhao, L.-h. und Deng, F.-q.: A new heuristic operator for job shop scheduling problems. In: *ICCT '06. International Conference on Communication Technology*. 2006, Seiten 1–6.

A. Anhang

A.1. Taxonomie Lösungsverfahren

Abbildung A.1 illustriet die Taxonomie zur Klassifikation von Verfahren des Shop-Schedulings nach *Jain und Meeran*.

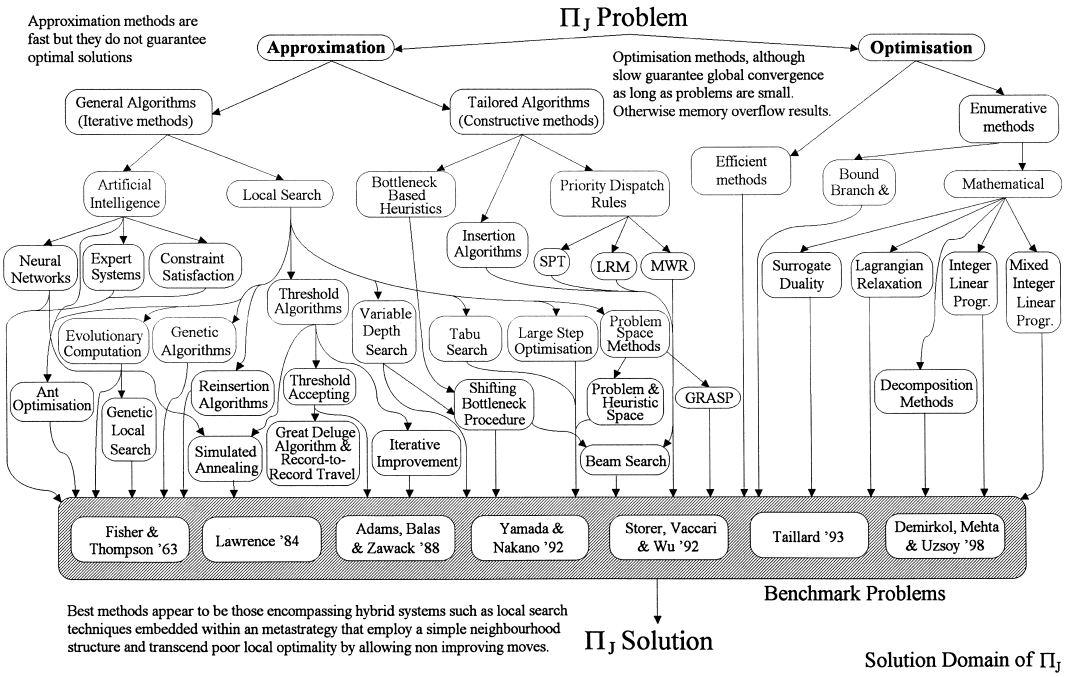


Abbildung A.1.: Taxonomie von Shop-Scheduling-Verfahren[1]

Die Taxonomie nach *Jain und Meeran* wurde beispielhaft gewählt. Weitere Taxonomien zur Klassifikation von Shop-Scheduling Verfahren sind z. B. in *Schoner*, *Dréo et al.*, *Mathirajan und Sivakumar*, *Evers* und *Domschke et al.* zu finden.[2]

[1] Jain und Meeran (1998)

[2] Vgl. (Schoner, 2008; Dréo et al., 2006; Mathirajan und Sivakumar, 2006; Evers, 2002; Domschke et al., 1997).

A.2. Taxonomie Benchmark-Probleme

Eine große Sammlung von Testadatensätzen für den Bereich des Operations Research ist in der OR-Library zu finden.[3] Für die Evaluierung des entwickelten WBSGT-Verfahrens zur Ablaufplanung und -steuerung wurden Testdaten aus *Beasley* verwendet.[4] Nachfolgend ist der einleitende Teil dieser Datei mit Benchmark-Problemen aufgeführt. Dieser beschreibt vor allem die Herkunft der Probleminstanzen – die eigentlichen Probleminstanzen werden in der Datei aufgelistet:

```
This file contains a set of 82 JSP test instances.

These instances are contributed to the OR-Library by
Dirk C. Mattfeld (email dirk@uni-bremen.de) and
Rob J.M. Vaessens (email robv@win.tue.nl).

o abz5-abz9 are from
        J. Adams, E. Balas and D. Zawack (1988),
        The shifting bottleneck procedure for job shop
        scheduling, Management Science 34, 391-401.

o ft06, ft10, and ft20 are from
        H. Fisher, G.L. Thompson (1963),
        Probabilistic learning combinations of local job-shop
        scheduling rules, J.F. Muth, G.L. Thompson (eds.),
        Industrial Scheduling,
        Prentice Hall, Englewood Cliffs, New Jersey, 225-251.

o la01-la40 are from
        S. Lawrence (1984),
        Resource constrained project scheduling:
        an experimental investigation of heuristic
        scheduling techniques (Supplement),
        Graduate School of Industrial Administration,
        Carnegie-Mellon University, Pittsburgh, Pennsylvania.

o orb01-orb10 are from
```

[3] Vgl. (Beasley, 1990b).
[4] Vgl. (Beasley, 1990a).

```
        D. Applegate, W. Cook (1991),
        A computational study of the job-shop scheduling
        instance, ORSA Journal on Computing 3, 149-156.
        (they were generated in Bonn in 1986)

o swv01-swv20 are from
        R.H. Storer, S.D. Wu, R. Vaccari (1992),
        New search spaces for sequencing instances with
        application to job shop scheduling,
        Management Science 38, 1495-1509.

o yn1-yn4 are from
        T. Yamada, R. Nakano (1992),
        A genetic algorithm applicable to large-scale job-shop
        instances, R. Manner, B. Manderick (eds.),
        Parallel instance solving from nature 2,
        North-Holland, Amsterdam, 281-290.

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Each instance consists of a line of description, a line
containing the number of jobs and the number of machines,
and then one line for each job, listing the machine number
and processing time for each step of the job. The machines
are numbered starting with 0.

[...]
```

A.3. Benchmark-Probleme

M_1 *or pt*	M_2 *or pt*	M_3 *or pt*	M_4 *or pt*	M_5 *or pt*	M_6 *or pt*	M_7 *or pt*	M_8 *or pt*	M_9 *or pt*	M_{10} *or pt*	M_{11} *or pt*	M_{12} *or pt*	M_{13} *or pt*	M_{14} *or pt*	M_{15} *or pt*
2 4	3 12	9 17	4 27	0 21	6 25	8 27	7 26	1 30	5 31	11 18	14 16	13 39	10 19	12 26
6 30	3 15	12 20	11 19	1 24	13 15	10 28	2 36	5 26	7 15	0 11	8 23	14 20	9 26	4 28
6 35	0 22	13 23	7 32	2 20	3 12	12 19	10 23	9 17	1 14	5 16	11 29	8 16	4 22	14 22
9 20	6 29	1 19	7 14	12 33	4 30	0 32	5 21	11 29	10 24	14 25	2 29	3 13	8 20	13 18
11 23	13 20	1 28	6 32	7 16	5 18	8 24	9 23	3 24	10 34	2 24	0 24	14 28	12 15	4 18
8 24	11 19	14 21	1 33	7 34	6 35	5 40	10 36	3 23	2 26	4 15	9 28	13 38	12 13	0 25
13 27	3 30	6 21	8 19	12 12	4 27	2 39	9 13	14 12	5 36	10 21	11 17	1 29	0 17	7 33
5 27	4 19	6 29	9 20	3 21	10 40	8 14	14 39	13 39	2 27	1 36	12 12	11 37	7 22	0 13
13 32	11 29	8 24	3 27	5 40	4 21	9 26	0 27	14 27	6 16	2 21	10 13	7 28	12 28	1 32
12 35	1 11	5 39	14 18	7 23	0 34	3 24	13 11	8 30	11 31	4 15	10 15	2 28	9 26	6 33
10 28	5 37	12 29	1 31	7 25	8 13	14 14	4 20	3 27	9 25	13 31	11 14	6 25	2 39	0 36
0 22	11 25	5 28	13 35	4 31	8 21	9 20	14 19	2 29	7 32	10 18	1 18	3 11	12 17	6 15
12 39	5 32	2 36	8 14	3 28	13 37	0 38	6 20	7 19	11 12	14 22	1 36	4 15	9 32	10 16
8 28	1 29	14 40	12 23	4 34	5 33	6 27	10 17	0 20	7 28	11 21	2 21	13 20	9 33	3 27
9 21	14 34	3 30	12 38	0 11	11 16	2 14	5 14	1 34	8 33	4 23	13 40	10 12	6 23	7 27
9 13	14 40	7 36	4 17	0 13	5 33	8 25	13 24	10 23	3 36	2 29	1 18	11 13	6 33	12 13
3 25	5 15	2 28	12 40	7 39	1 31	8 35	6 31	11 36	4 12	10 33	14 19	9 16	13 27	0 21
12 22	10 14	0 12	2 20	5 12	1 18	11 17	8 39	14 31	3 31	7 32	9 20	13 29	4 13	6 26
5 18	10 30	7 38	14 22	13 15	11 20	9 16	3 17	1 12	2 13	12 40	6 17	8 30	4 38	0 13
9 31	8 39	12 27	1 14	5 33	3 31	11 22	13 36	0 16	7 11	14 14	4 29	6 28	2 22	10 17

Tabelle A.1.: Benchmark-Problem: Adams, Balas, and Zawack (ABZ7) 20×15[5]

M_1 *or pt*	M_2 *or pt*	M_3 *or pt*	M_4 *or pt*	M_5 *or pt*
0 456	1 537	2 123	3 214	4 234
0 789	1 854	2 225	3 528	4 123
0 876	1 632	2 588	3 896	4 456
0 543	1 145	2 669	3 325	4 789
0 210	1 785	2 966	3 147	4 876
0 123	1 214	2 332	3 856	4 543
0 456	1 752	2 144	3 321	4 210
0 789	1 143	2 755	3 427	4 123
0 876	1 698	2 322	3 546	4 456
0 543	1 532	2 100	3 321	4 789
0 210	1 145	2 114	3 401	4 876
0 124	1 247	2 753	3 214	4 543

Tabelle A.2.: Benchmark-Problem: Carlier (CAR3) 12×5[6]

[5]Vgl. (Adams et al., 1988).
[6]Vgl. (Carlier, 1978).
[7]Vgl. (Storer et al., 1992).
[8]Vgl. (Fisher und Thompson, 1963).
[9]Vgl. (Lawrence, 1984).
[10]Vgl. (Lawrence, 1984).
[11]Vgl. (Lawrence, 1984).

M_1 *or pt*	M_2 *or pt*	M_3 *or pt*	M_4 *or pt*	M_5 *or pt*	M_6 *or pt*	M_7 *or pt*	M_8 *or pt*	M_9 *or pt*	M_{10} *or pt*
3 19	2 27	1 39	4 13	0 25	8 37	9 40	5 54	7 74	6 93
2 69	0 30	4 1	3 4	1 64	7 71	5 2	9 84	6 31	8 8
4 79	3 80	0 86	2 55	1 54	8 81	6 72	7 86	5 59	9 75
2 76	3 15	1 26	0 17	4 30	8 44	7 91	6 83	5 52	9 68
4 73	3 87	1 74	0 39	2 98	9 100	5 43	8 17	7 7	6 77
1 63	0 49	2 16	3 55	4 9	9 73	5 61	8 34	6 82	7 46
0 87	1 71	4 43	3 80	2 39	7 70	8 18	6 41	9 79	5 44
4 70	2 22	0 73	3 62	1 64	5 25	8 19	6 69	9 41	7 28
3 16	0 84	1 58	4 7	2 9	5 8	6 10	7 17	8 42	9 65
3 8	0 10	1 3	4 41	2 3	7 40	8 56	5 53	9 96	6 13
4 62	1 60	3 64	2 12	0 39	5 2	7 64	6 87	9 21	8 60
2 66	1 71	3 23	4 75	0 78	7 74	6 35	9 24	8 23	5 50
1 5	3 92	4 6	0 69	2 80	7 13	5 17	9 89	6 80	8 47
0 82	3 84	1 24	2 47	4 93	7 85	5 34	6 73	8 28	9 91
4 55	0 57	3 63	2 24	1 40	7 30	6 37	5 99	8 88	9 41
1 75	2 47	3 68	0 7	4 78	7 80	6 2	9 23	8 49	5 50
0 91	4 25	2 10	1 21	3 94	8 6	7 59	5 84	9 75	6 70
2 85	1 31	0 94	4 94	3 11	5 21	9 7	6 61	8 50	7 93
1 27	0 77	4 13	2 30	3 2	5 88	7 4	9 39	6 53	8 54
1 34	2 12	3 31	0 24	4 24	7 16	5 6	9 88	8 81	6 11

Tabelle A.3.: Benchmark-Problem: Storer, Wu, and Vaccari (SWV01) 20×10^7

M_1 *or pt*	M_2 *or pt*	M_3 *or pt*	M_4 *or pt*	M_5 *or pt*	M_6 *or pt*	M_7 *or pt*	M_8 *or pt*	M_9 *or pt*	M_{10} *or pt*
0 29	1 78	2 9	3 36	4 49	5 11	6 62	7 56	8 44	9 21
0 43	2 90	4 75	9 11	3 69	1 28	6 46	5 46	7 72	8 30
1 91	0 85	3 39	2 74	8 90	5 10	7 12	6 89	9 45	4 33
1 81	2 95	0 71	4 99	6 9	8 52	7 85	3 98	9 22	5 43
2 14	0 6	1 22	5 61	3 26	4 69	8 21	7 49	9 72	6 53
2 84	1 2	5 52	3 95	8 48	9 72	0 47	6 65	4 6	7 25
1 46	0 37	3 61	2 13	6 32	5 21	9 32	8 89	7 30	4 55
2 31	0 86	1 46	5 74	4 32	6 88	8 19	9 48	7 36	3 79
0 76	1 69	3 76	5 51	2 85	9 11	6 40	7 89	4 26	8 74
1 85	0 13	2 61	6 7	8 64	9 76	5 47	3 52	4 90	7 45

Tabelle A.4.: Benchmark-Problem: Fisher and Thompson (FT10) 10×10^8

M_1 *or pt*	M_2 *or pt*	M_3 *or pt*	M_4 *or pt*	M_5 *or pt*	M_6 *or pt*	M_7 *or pt*	M_8 *or pt*	M_9 *or pt*	M_{10} *or pt*
2 34	3 55	5 95	9 16	4 21	6 71	0 53	8 52	1 21	7 26
3 39	2 31	0 12	1 42	9 79	8 77	6 77	5 98	4 55	7 66
1 19	0 83	3 34	4 92	6 54	9 79	8 62	5 37	2 64	7 43
4 60	2 87	8 24	5 77	3 69	7 38	1 87	6 41	9 83	0 93
8 79	9 77	2 98	4 96	3 17	0 44	7 43	6 75	1 49	5 25
8 35	7 95	6 9	9 10	2 35	1 7	5 28	4 61	0 95	3 76
4 28	5 59	3 16	9 43	0 46	8 50	6 52	7 27	2 59	1 91
5 9	4 20	2 39	6 54	1 45	7 71	0 87	3 41	9 43	8 14
1 28	5 33	0 78	3 26	2 37	7 8	8 66	6 89	9 42	4 33
2 94	5 84	6 78	9 81	1 74	3 27	8 69	0 69	7 45	4 96
1 31	4 24	0 20	2 17	9 25	8 81	5 76	3 87	7 32	6 18
5 28	9 97	0 58	4 45	6 76	3 99	2 23	1 72	8 90	7 86
5 27	9 48	8 27	7 62	4 98	6 67	3 48	0 42	1 46	2 17
1 12	8 50	0 80	2 50	9 80	3 19	5 28	6 63	4 94	7 98
4 61	3 55	6 37	5 14	2 50	8 79	1 41	9 72	7 18	0 75

Tabelle A.5.: Benchmark-Problem: Lawrence (LA21) 15×10^9

M_1 *or pt*	M_2 *or pt*	M_3 *or pt*	M_4 *or pt*	M_5 *or pt*	M_6 *or pt*	M_7 *or pt*	M_8 *or pt*	M_9 *or pt*	M_{10} *or pt*
8 52	7 26	6 71	9 16	2 34	1 21	5 95	4 21	0 53	3 55
4 55	5 98	3 39	9 79	0 12	8 77	6 77	7 66	2 31	1 42
5 37	4 92	2 64	6 54	1 19	7 43	0 83	3 34	9 79	8 62
1 87	5 77	0 93	3 69	2 87	7 38	8 24	6 41	9 83	4 60
2 98	5 25	6 75	9 77	1 49	3 17	8 79	0 44	7 43	4 96
1 7	4 61	0 95	2 35	9 10	8 35	5 28	3 76	7 95	6 9
5 59	9 43	0 46	4 28	6 52	3 16	2 59	1 91	8 50	7 27
5 9	9 43	8 14	7 71	4 20	6 54	3 41	0 87	1 45	2 39
1 28	8 66	0 78	2 37	9 42	3 26	5 33	6 89	4 33	7 8
4 96	3 27	6 78	5 84	2 94	8 69	1 74	9 81	7 45	0 69
4 24	7 32	9 25	2 17	3 87	8 81	5 76	6 18	1 31	0 20
8 90	5 28	1 72	7 86	2 23	3 99	6 76	9 97	4 45	0 58
2 17	4 98	3 48	1 46	8 27	6 67	7 62	0 42	9 48	5 27
0 80	8 50	3 19	7 98	5 28	2 50	4 94	6 63	1 12	9 80
9 72	0 75	4 61	8 79	6 37	2 50	5 14	3 55	7 18	1 41
3 96	2 14	5 57	0 47	7 65	4 75	8 79	1 71	6 60	9 22
1 31	7 47	8 58	3 32	4 44	5 58	6 34	0 33	2 69	9 51
1 44	7 40	2 17	0 62	8 66	6 15	3 29	9 38	5 8	4 97
2 58	3 50	4 63	9 87	0 57	6 21	7 57	8 32	1 39	5 20
1 85	0 84	5 56	3 61	9 15	7 70	8 30	2 90	6 67	4 20

Tabelle A.6.: Benchmark-Problem: Lawrence (LA26) 20×10^{10}

M_1 *or pt*	M_2 *or pt*	M_3 *or pt*	M_4 *or pt*	M_5 *or pt*	M_6 *or pt*	M_7 *or pt*	M_8 *or pt*	M_9 *or pt*	M_{10} *or pt*
3 60	4 48	5 95	0 87	1 72	9 5	8 35	7 39	6 54	2 66
7 37	6 34	0 97	5 55	2 21	3 20	4 59	9 46	8 19	1 46
4 45	2 73	1 24	8 28	0 28	3 25	5 23	7 83	9 5	6 78
0 53	2 12	9 12	1 37	8 33	3 71	6 55	5 29	7 87	4 38
4 90	2 49	9 27	7 65	5 7	6 23	0 48	3 83	8 17	1 40
3 85	4 25	2 84	6 64	9 13	1 66	7 46	8 59	0 62	5 19
5 88	6 67	4 14	0 41	1 73	7 57	2 53	3 80	9 47	8 74
1 78	5 64	4 63	6 46	3 84	0 84	8 28	9 52	7 26	2 41
1 11	0 64	6 97	9 38	2 17	4 85	5 73	3 10	8 95	7 67
3 93	2 95	7 43	1 65	8 32	0 59	6 85	5 46	9 85	4 60
2 61	3 41	5 49	4 23	0 66	7 49	8 70	9 99	1 90	6 17
4 13	7 7	1 98	8 57	0 73	3 73	2 68	5 40	9 98	6 9
9 86	6 76	4 14	3 41	1 85	0 37	8 19	2 17	7 54	5 79
1 40	2 53	7 97	5 87	8 96	4 84	3 16	6 66	9 52	0 95
6 33	1 33	3 87	0 18	2 55	8 13	4 77	7 60	9 42	5 74
7 92	5 91	8 79	2 54	4 69	6 79	3 33	1 61	9 39	0 16
6 82	1 41	4 28	5 64	2 78	3 76	7 6	8 49	9 47	0 58
0 52	5 42	8 24	9 91	3 47	6 88	4 91	7 52	2 28	1 35
5 82	2 76	3 86	6 93	4 84	7 38	8 95	9 37	1 21	0 23
9 77	4 8	6 42	7 64	0 70	2 45	8 45	5 28	3 67	1 86

Tabelle A.7.: Benchmark-Problem: Lawrence (LA27) 20×10^{11}

A.4. Giffler/ Thompson-Heuristik

Beispiel A.1 Giffler-Thompson
Das folgende Beispiel in Abbildung A.2 zeigt den Ablauf eines Giffler-Thompson-Verfahrens, wobei die Konfliktmengenauflösung mittels der Regel SPT durchgeführt wird.

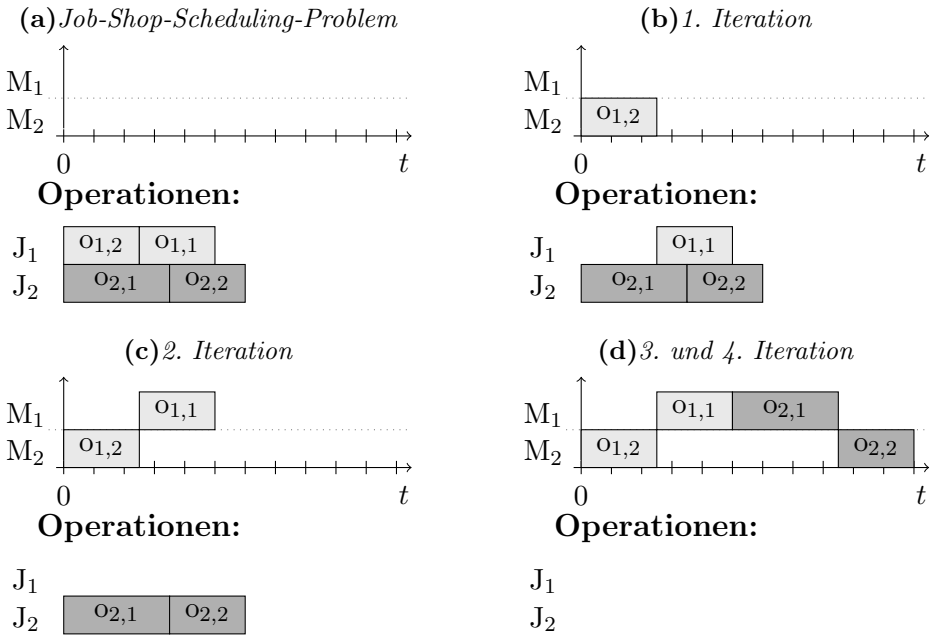


Abbildung A.2.: Lösung eines Job-Shop-Problems mit Giffler-Thompson-Heuristik

Das vorgestellte Scheduling-Problem besteht aus zwei Maschinen und zwei Aufträgen. In den Teilabbildungen sind sowohl der Schedule mit den bereits eingeplanten Operationen als auch die noch ausstehenden Operationen dargestellt. Die noch ausstehenden Operationen sind derart eingezeichnet, dass der früheste Startzeitpunkt erkennbar ist. Abbildung A.2(a) zeigt den noch leeren Schedule und die zwei Aufträge mit ihren in technologischer Reihenfolge sortierten Operationen. Im 1. Iterationsschritt (Abbildung A.2(b)) ermittelt die Heuristik diejenige Operation mit dem frühesten Fertigstellungszeitpunkt (hier $o_{1,2}$). Es kann kein weiterer Auftrag auf Maschine M_2 ausgeführt werden; deshalb ist $o_{1,2}$ direkt einplanbar. Beim 2. Iterationsschritt (Abbildung A.2(c)) ist Operation $o_{1,1}$ diejenige, die den frühestmöglichen Fertigstellungszeitpunkt hat. Hier ist allerdings auch Operation $o_{2,1}$ vor der Fertigstellung der ausgewählten Operation $o_{1,1}$ auf derselben Maschine bearbeitbar. Dieser Konflikt wird mit der Prioritätsregel SPT aufgelöst und somit Operation $o_{1,1}$ eingeplant. Abbildung A.2(d) zeigt die letzten

beiden Iterationsschritte. Innerhalb dieser werden die noch ausstehenden Operationen der vorliegenden Scheduling-Probleminstanz eingeplant. □

Beispiel A.2 Giffler-Thompson für inhomogene Operationszeiten
Das folgende Beispiel zeigt den Ablauf des Giffler-Thompson-Verfahrens für eine Probleminstanz mit inhomogenen Operationszeiten. Die Konfliktmengenauflösung wird mittels der Regeln SPT und LPT durchgeführt. Der erste Konflikt wird mit der SPT Regel gelöst, der zweite mit der LPT Regel, der dritte mit der SPT Regel usw.

	Maschine 1		Maschine 2		Maschine 3	
	or	*pt*	*or*	*pt*	*or*	*pt*
Job 1	I	2	II	4	III	2
Job 2	I	3	III	7	II	3
Job 3	II	3	III	6	I	9
Job 4	III	1	II	3	I	4

Tabelle A.8.: Beispielhafte Scheduling-Probleminstanz

Die Scheduling-Probleminstanz besteht aus 3 Maschinen und 4 Aufträgen (siehe Tabelle A.8). Reihenfolgenummer und Prozesszeit der Operationen eines Jobs beschreiben die beiden Variablen *or* und *pt*. In den Teilabbildungen (siehe Abbildung A.2(a) bis Abbildung A.2(j)) des Beispiels sind die ersten 5 Iterationsschritte dargestellt. Einzelne Operationen sind als $o_{j,m}$ dargestellt, wobei j die Jobnummer und m die Maschinennummer abbilden. Die Gantt-Charts werden von Iteration zu Iteration sukzessive erweitert.

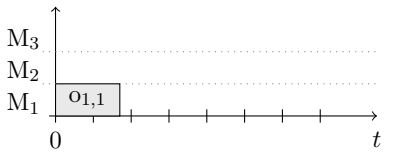
□

Beispiel A.2 zeigt, dass trotz korrekter Ausführung des Giffler-Thompson-Verfahrens ungültige Schedules erzeugt werden können. Obwohl für die Operation $o_{1,3}$ der frühestmögliche Startzeitpunkt im Sinne des Verfahrens immer korrekt angepasst wurde, läuft das Verfahren dennoch in einen Fehler. Maschine M_3 wird durch die Operationen $o_{3,3}$ und $o_{1,3}$ doppelt belegt. Ist das Scheduling-Problem inhomogen, d. h. es sind Operationen mit langen und kurzen Operationszeiten vorhanden, kann es zu diesen Problemen kommen. Wenn also die Differenz zwischen den Bearbeitungszeiten der Jobs nicht hinreichend groß ist, kann es zu nicht ausreichenden Anpassungen der frühestmöglichen Starttermine einzelner Operationen kommen. Im dargestellten Beispiel wäre Operation $o_{1,3}$, obwohl nicht in der Konfliktmenge, bei der Einplanung von Operation $o_{3,3}$ mit zu aktualisieren gewesen.

(a) *1. Iterationsschritt*

Einpl. Operationen:	$o_{1,1}$; $o_{2,1}$; $o_{3,3}$; $o_{4,3}$
Op. mit min F-Zeit:	$o_{1,1}$
Konfliktmenge:	$o_{1,1}$, $o_{2,1}$
SPT wählt aus:	$o_{1,1}$

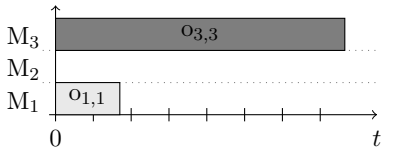
(b) *Gantt-Chart nach 1. Iteration*



(c) *2. Iterationsschritt*

Einpl. Operationen:	$o_{1,2}$; $o_{2,1}$; $o_{3,3}$; $o_{4,3}$
Op. mit min F-Zeit:	$o_{4,3}$
Konfliktmenge:	$o_{4,3}$, $o_{3,3}$
LPT wählt aus:	$o_{3,3}$

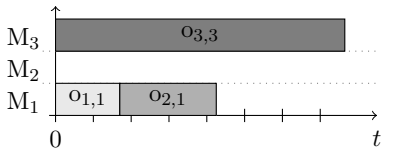
(d) *Gantt-Chart nach 2. Iteration*



(e) *3. Iterationsschritt*

Einpl. Operationen:	$o_{1,2}$; $o_{2,1}$; $o_{3,1}$; $o_{4,3}$
Op. mit min F-Zeit:	$o_{2,1}$
Konfliktmenge:	$o_{2,1}$
SPT wählt aus:	$o_{2,1}$

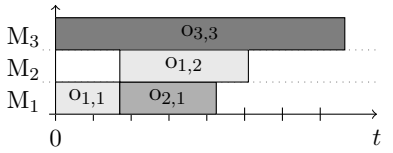
(f) *Gantt-Chart nach 3. Iteration*



(g) *4. Iterationsschritt*

Einpl. Operationen:	$o_{1,2}$; $o_{2,3}$; $o_{3,1}$; $o_{4,3}$
Op. mit min F-Zeit:	$o_{1,2}$
Konfliktmenge:	$o_{1,2}$
LPT wählt aus:	$o_{1,2}$

(h) *Gantt-Chart nach 4. Iteration*



(i) *5. Iterationsschritt*

Einpl. Operationen:	$o_{1,3}$; $o_{2,3}$; $o_{3,1}$; $o_{4,3}$
Op. mit min F-Zeit:	$o_{2,3}$
Konfliktmenge:	$o_{2,3}$; $o_{1,3}$; $o_{4,3}$
SPT wählt aus:	$o_{1,3}$

(j) *Gantt-Chart nach 5. Iteration*

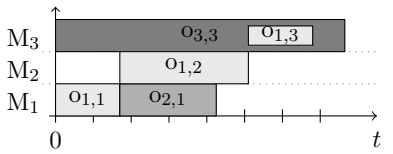


Abbildung A.3.: Fehlerhafte Giffler-Thompson-Lösung eines Job-Shop-Problems

A.5. Precision, Recall und F-Maß

Als Resultat auf eine Suchanfrage bei *Internetsuchmaschine A* werden beispielsweise 100 Ergebnisse zurückgeliefert, von denen 40 dem gewünschten Inhalt/Resultat entsprechen. Bei *Internetsuchmaschine B* werden 400 Ergebnisse zurückgeliefert. Hiervon entsprechen 160 dem gewünschten Inhalt. Weil beide eine Klassifikationsgenauigkeit von 40% ausweisen, stellt sich die Frage, welche der beiden Suchmaschinen (Klassifizierer) das bessere Ergebnis liefert.[12]

Beispiel A.3 Precision, Recall und F-Maß

Die gegebene Menge von 150 Testinstanzen ordnet ein Klassifizierer den drei Klassen a, b

[12] Vgl. (Witten und Frank, 2005, Seite 155 ff.).

und c zu. In der Menge sind die drei Klassen gleichverteilt, d. h. jeweils 50 Testinstanzen gehören zu einer der drei Klassen. Als Klassifikationsergebnis sei angenommen, dass alle Testinstanzen der Klassen a und c korrekt zugeordnet wurden. Die Testinstanzen der Klasse b werden alle falsch klassifiziert. Der Klassifizierer ordnet zwei b-Testinstanzen der Klasse a und die restlichen 48 der Klasse c zu. Tabelle A.4 zeigt das Ergebnis übersichtlich.

Zuordnung des Klassifizierers	a	b	c
a	50	0	0
b	2	0	48
c	0	0	50

Abbildung A.4.: Beispielhaftes Klassifikationsergebnis

Tabelle A.5 ist so zu lesen, dass die Zeile die korrekte Klasse einer Testinstanz und die Spalte das Ergebnis des Klassifizierers angibt. Eintrag (i, j) gibt die Anzahl der Testinstanzen aus i, die j zugeordnet werden an. Sind alle Wertepaare auf der Hauptdiagonalen der Tabelle angeordnet, dann hat der Klassifizierer alle Instanzen richtig zugeordnet, wodurch die Klassifikationsgenauigkeit bei 100% liegt. Die Klassifikationsgenauigkeit im Beispiel liegt bei ca. 67%. Die Werte für Precision, Recall und das resultierende F-Maß sind in Tabelle A.5 aufgelistet.

Klasse	Precision	Recall	F-Mass
a	0,962	1,000	0,980
b	0,000	0,000	0,000
c	0.510	1,000	0,676

Abbildung A.5.: Beispielhafte Precision-, Recall- und F-Maß-Werte

Obwohl sämtliche Testinstanzen der Klassen a und b korrekt klassifiziert wurden, d. h. $rec(a) = rec(c) = 1$, ist das F-Maß kleiner als eins. Das ist damit zu erklären, dass der Klassifizierer die Testinstanzen der Klasse b den Klassen a und c zuordnet und sich so die Precision-Werte verschlechtern. Das Ergebnis für die Klassen a und c ist also durch die falsche Zuordnung der Instanzen von b verschmutzt. Unter dem Ausschluss der Klasse b und deren Instanzen wären die Werte für das F-Maß mit $F(a) = F(c) = 1$ perfekt.

Mit den jetzt zur Verfügung stehenden Maßzahlen sind die beiden eingangs beschrie-

benen Klassifizierer der Internetsuchmaschine genauer zu vergleichen. Dazu werden in Abbildung A.6 zunächst die genauen Verteilungen der jeweils zurückgelieferten Suchergebnisse dargestellt.

(a) *Klassifizierer A mit einer Genauigkeit von 40 Prozent*

Zuordnung des Klassifizierers	a	b	c
a	40	0	0
b	5	0	20
c	5	30	0

(b) *Klassifizierer B mit einer Genauigkeit von 40 Prozent*

Zuordnung des Klassifizierers	a	b	c
a	120	0	0
b	10	30	0
c	90	140	10

Abbildung A.6.: Verteilung des Klassifikationsergebnisses zweier Klassifizierer

Mittels der detaillierten Suchergebnisse für die beiden Klassifizierer A & B sind jetzt die genaueren Vergleichszahlen berechenbar. Abbildung A.7 zeigt die Precision, Recall und F-Maß Werte für die beiden Klassifizierer.

(a) *Klassifizierer A*

Klasse	Precision	Recall	F-Mass
a	0,80	1,00	0,89
b	0,00	0,00	0,00
c	0,00	0,00	0,00

(b) *Klassifizierer B*

Klasse	Precision	Recall	F-Mass
a	0,56	1,00	0,72
b	0,18	0,75	0,29
c	1,00	0,04	0,08

Abbildung A.7.: Precision-, Recall- und F-Maß-Werte für zwei Klassifizierer

Hiermit wird deutlich, dass, obwohl beide Klassifizierer eine Genauigkeit von 40% erreichen, Klassifizierer B der bessere von beiden ist. Er ordnet nicht nur alle Ergebnisse des Typs *a* richtig zu, das macht Klassifizierer A ebenso, zusätzlich trifft B jedoch einige Typen der Klassen *b* und *c*, was Klassifizierer A nicht gelingt.

A.6. Ergebnis: Zwei-Klassifizierer-System

Tabelle A.9 zeigt die Ergebnisse der Evaluation des Zwei-Klassifizierer-Systems. Die Ergebnisse der einzelnen Problemvarianten wurden je Problemtyp zu Mittelwerten zusammengefasst.

Problem	Auftragswahl	Arbeitssystem-wahl	Makespan (Mittelwert)	Abweichung (Mittelwert)
1×3	wissensbasiert	wissensbasiert	284,217	–
	SPT	fastest	295,417	4,05 %
	SPT	slowest	295,338	4,02 %
	LPT	fastest	284,217	0 %
	LPT	slowest	284,036	-0,07 %
	zufällig	zufällig	290,118	1,53 %
2×4	wissensbasiert	wissensbasiert	121,27	–
	SPT	fastest	174,054	46,13 %
	SPT	slowest	198,042	69,83 %
	LPT	fastest	123,962	2,32 %
	LPT	slowest	123,514	1,89 %
	zufällig	zufällig	127,355	5,17 %
3×2	wissensbasiert	wissensbasiert	715,714	–
	SPT	fastest	913,626	27,74 %
	SPT	slowest	931,557	30,26 %
	LPT	fastest	752,077	5,33 %
	LPT	slowest	749,999	5,19 %
	zufällig	zufällig	716,92	0,4 %
3×3	wissensbasiert	wissensbasiert	95,41	–
	SPT	fastest	100,47	5,8 %
	SPT	slowest	124,79	32,77 %
	LPT	fastest	104,5	10,04 %
	LPT	slowest	108,092	14,39 %
	zufällig	zufällig	105,115	10,61 %
4×4	wissensbasiert	wissensbasiert	111,702	–
	SPT	fastest	143,435	28,07 %
	SPT	slowest	219,575	97,2 %
	LPT	fastest	124,292	11,15 %
	LPT	slowest	142,842	29,51 %
	zufällig	zufällig	117,528	5,41 %

Tabelle A.9.: Makespan Zwei-Klassifizierer-System: Übersicht

A.7. Ergebnis: Ein-Klassifizierer-System

Beim Experiment zum Ein-Klassifizierer-System wurde ein Flexible-Flow-Shops mit unverwandten Maschinen auf den Stufen verwendet ($FR5||C_{max}$). Das Benchmark-Problem besteht aus 5 Stufen mit jeweils 3 unverwandten Maschinen auf den Stufen und insgesamt 10 zu bearbeitenden Jobs. Die Ober- und Untergrenzen der Standard-Bearbeitungszeit der Jobs liegen bei 20 und 200, die relativen Maschinengeschwindigkeiten sind im Intervall von 50 bis 200 Prozent gleichverteilt (Schrittgröße 10 Prozent).

Zum Training und zur Evaluierung wurden jeweils 200 unterschiedliche Probleminstanzen verwendet. Die Trainingsbeispiele sind mittels eines Genetischen Algorithmus simuliert worden.[13] Abbildung A.8 illustriert die erreichten Ergebnisse, Tabelle A.10 zeigt diese im Detail.

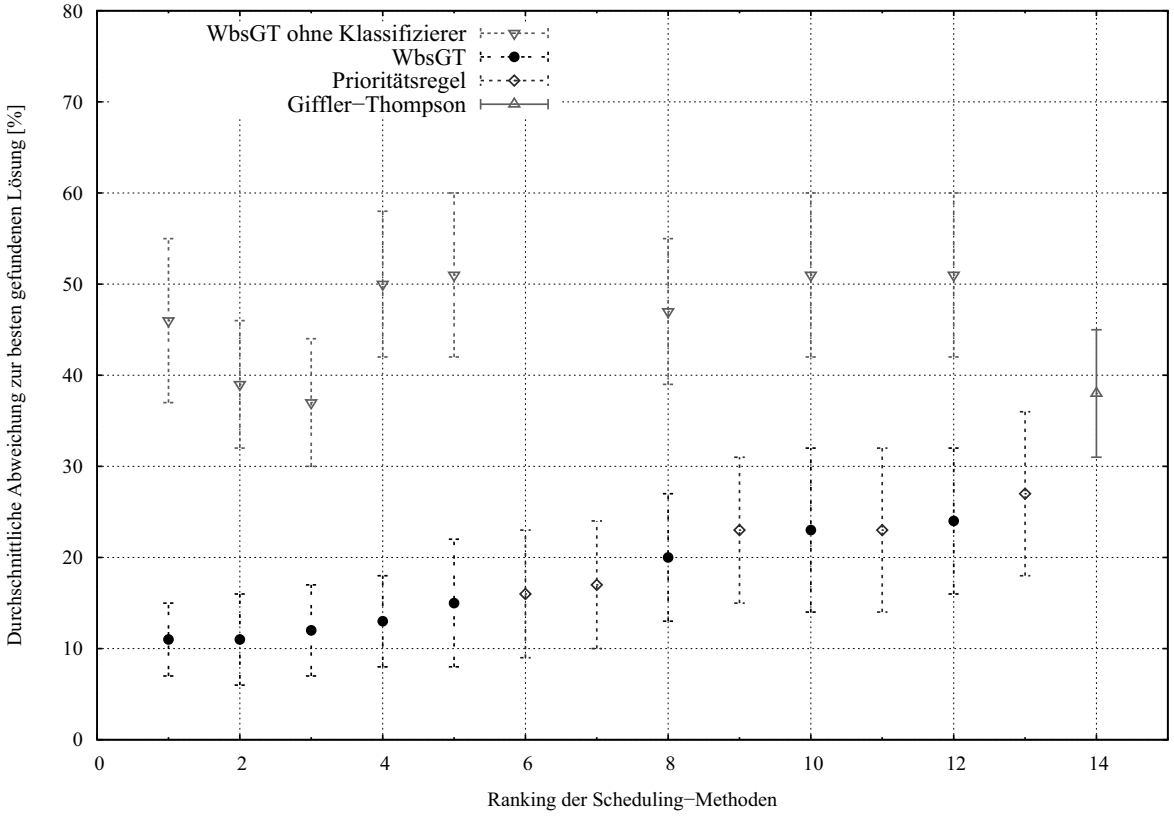


Abbildung A.8.: Ergebnis Ein-Klassifizierer-System bei einem 10×5 Problem

Die Ergebnisse der einfachen Regelanwendung sind wie folgt: $FSFO-\mathrm{Min}\, v$ ($\mu = 90,38$; $\sigma = 17,51$), $\mathrm{Min}\, v$ ($\mu = 99,22; \sigma = 19,37$), $FSFO - \mathrm{Max}\, p$ ($\mu = 105,07; \sigma = 18,68$), $\mathrm{Max}\, p$ ($\mu = 115,10; \sigma = 21,03$), $\mathrm{Max}\, c$ ($\mu = 118,10; \sigma = 21,00$). Da die Ergebnisse extrem schlecht ausfallen, wurden diese nicht mit in Abbildung A.8 und Tabelle A.10 aufgenommen. Es ist festzustellen, dass das WBSGT-Verfahren der zufälligen Auswahl von Prioritätsregeln überlegen ist (Rang 1-5 wird von WBSGT-Ausprägungen belegt). Nicht nur die erreichten Ergebnisse sind besser, auch ist die Varianzen geringer. Das beste Ergebnis wird von WBSGT (*Alle Regeln*) mit einer Abweichung von 11,50 Prozent und einer Varianz von 4,75 erreicht. Die gleiche Ausprägung mit Zufallsauswahl

[13] Implementiert wurde der Genetische Algorithmus in der Ausprägung von *Ruiz und Maroto* (vgl. Ruiz und Maroto, 2006). Als Selektionskomponente wurde das Verfahren von *Jungwattanakit et al.* umgesetzt (vgl. Jungwattanakit et al., 2008).

erreicht eine deutlich schlechtere Abweichung von 46,94 Prozent, bei einer höheren Varianz von 9,43. Den zweiten Rang belegt WbsGT ($FSFO - \text{Min}\,p$, $FSFO - \text{Max}\,p$, $FSFO - \text{Min}\,v$, $FSFO - \text{Max}\,v$) bei einer 11,71 prozentigen Abweichung und einer Varianz von 5,05. Die Ausprägung mit zufälliger Regelauswahl erreicht in diesem Fall lediglich eine Abweichung von 39,31 Prozent bei einer um 2,15 Punkte höheren Varianz. Die erste Ausprägung mit einer einfachen Prioritätsregel liegt mit einer Abweichung von 16,13 Prozent und einer Varianz von 7,29 auf Rang 6. Die Giffler-Thompson-Heuristik bei gänzlich zufälliger Auswahl der Operationen liegt mit einer Abweichung von 38,02 Prozent und einer Varianz von 7,10 auf Rang 14 und ist damit schlechter als alle WbsGT-Varianten. Insgesamt ist festzustellen, dass das WbsGT-Verfahren sowohl der zufälligen Auswahl von Prioritätsregeln als auch der einfachen Regelanwendung und der einfachen Giffler-Thompson-Heuristik überlegen ist.

Rang	Verfahren	Abweichung & Varianz		Regelmenge	Abw. & Var. ohne Klassifizierer	
1	WbsGT	11,50	4,75	Alle Regeln	46,96	9,43
2	WbsGT	11,71	5,05	$FSFO - \text{Min}\,p$, $FSFO - \text{Max}\,p$, $FSFO - \text{Min}\,v$, $FSFO - \text{Max}\,v$	39,31	7,20
3	WbsGT	12,77	5,76	$\text{Min}\,p$, $\text{Min}\,v$, $\text{Min}\,c$, $FSFO - \text{Min}\,p$, $FSFO - \text{Min}\,v$	37,82	7,66
4	WbsGT	13,66	5,99	$\text{Min}\,c$, $\text{Max}\,c$, $\text{Min}\,p$, $\text{Max}\,p$, $\text{Min}\,v$, $\text{Max}\,v$	50,10	8,84
5	WbsGT	15,58	7,06	$\text{Min}\,c$, $\text{Max}\,c$, $\text{Min}\,p$, $\text{Max}\,p$	51,34	9,09
6	$FSFO - \text{Min}\,p$	16,13	7,29	–	–	–
7	$FSFO - \text{Max}\,v$	17,14	7,25	–	–	–
8	WbsGT	20,51	7,13	$\text{Min}\,v$, $\text{Max}\,v$	47,73	8,71
9	$\text{Max}\,v$	23,09	8,06	–	–	–
10	WbsGT	23,97	9,31	$\text{Min}\,c$, $\text{Max}\,c$	51,86	9,57
11	$\text{Min}\,c$	23,97	9,31	–	–	–
12	WbsGT	24,61	8,34	$\text{Min}\,p$, $\text{Max}\,p$	51,62	9,77
13	$\text{Min}\,p$	27,29	9,54	–	–	–
14	Giffler-Thompson	38,02	7,10	–	–	–

Tabelle A.10.: Makespan Ein-Klassifizierer-System bei 10×5 Problem: Übersicht

A.8. Ergebnis: Adaptionsmechanismus

Bei der Evaluierung des Adaptionsmechanismus sollen zwei wesentliche Punkte untersucht werden. Zum einen die Steigerung der Lösungsgüte in Bezug auf die Zielfunktion – Minimierung des Makespans. Zum anderen die so erreichte WbsGT-Lösungsgüte insgesamt. Dazu wurden zwei Testreihen durchgeführt.

In der *ersten Testreihe* werden identische Probleme jeweils mit und ohne Adaptionsmechanismus gelöst.[14] Für den Test werden das FT10-, ABZ7- und ABZ7-8-Benchmark-

[14]Die Variationen des Benchmark-Problems sind für die Lösung mit Adaptionsmechanismus sowie für

Problem verwendet. Hierbei wurde das ABZ7-8 neu entwickelt. Es basiert auf dem ABZ7 Problem, welches zu einem dynamischen Job-Shop-Problem erweitert wurde.[15] Hierzu werden während des Lösungsvorgangs dynamisch weitere Jobs zum Problem hinzugefügt. Die neu hinzugefügten Jobs stammen aus dem ABZ8-Benchmark-Problem und erweitern das Problem zwischen den Zeitpunkten 50 und 500 im Abstand von je 50 Zeiteinheiten um jeweils einen Job.

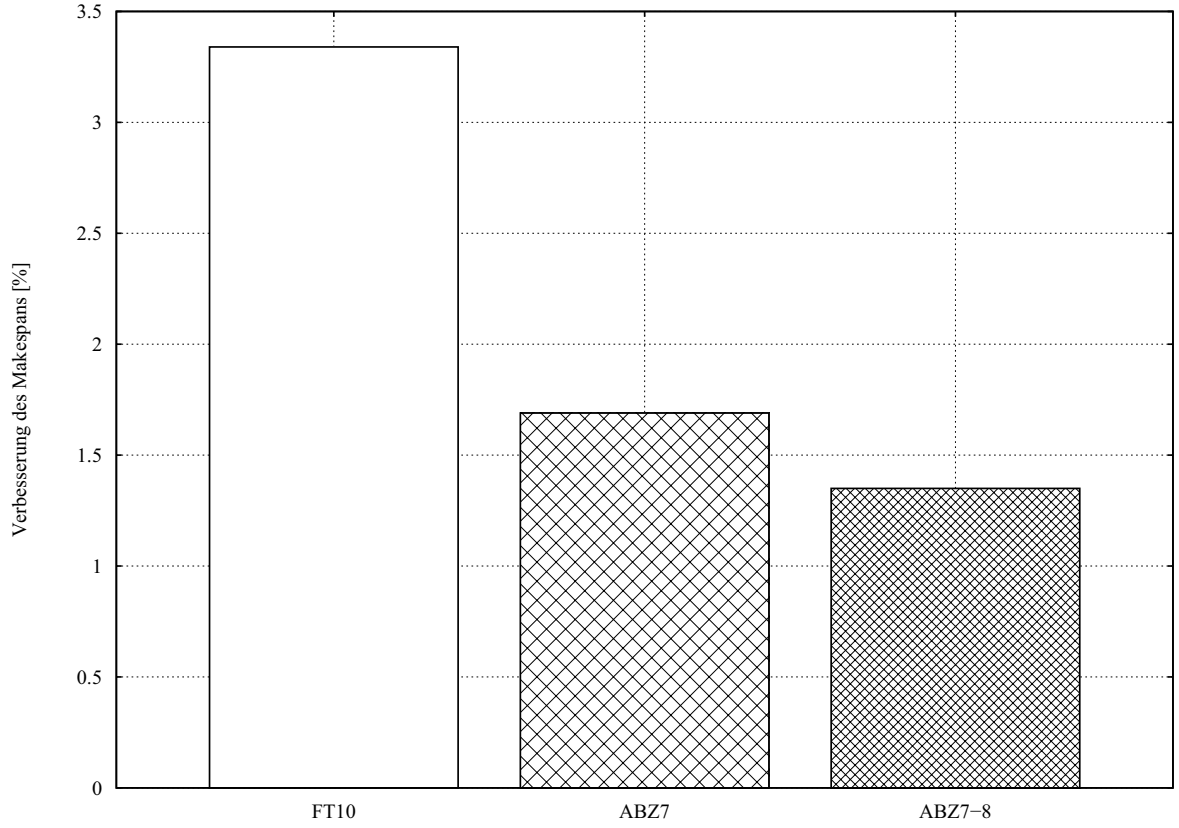


Abbildung A.9.: Ergebnis Adaptionsmechanismus absolut

Abbildung A.9 zeigt die prozentuale Verbesserung der generierten Schedules beim Einsatz des Adaptionsmechanismus. Es ist zu erkennen, dass der Adaptionsmechanismus in allen drei Testreihen zu einer Verbesserung der erzeugten Lösungen führt. Die Makespans der Schedules, welche bei dem Einsatz des Adaptionsmechanismus gefunden werden, liegen in den Testreihen 1,4 bis 3,4 Prozent unter den Lösungen, die ohne Adaptionsmechanismus generiert werden.

In der *zweiten Testreihe* werden dieselben Probleme wie in der vorherigen Testreihen

die Lösung ohne identisch. Es wurden 100 Varianten generiert. Der verwendete Mittelwert war $\mu = 0$, die Standardabweichung $\sigma = 50$.

[15] Zu Details zum FT10- und ABZ7-Benchmark-Problem siehe Tabelle A.4 und A.1.

verwendet. Es kommen die gleichen Lösungsverfahren wie in Kapitel 6.2 zum Einsatz. Abbildung A.10 zeigt die von den einzelnen Verfahren erreichten durchschnittlichen Makespans.[16]

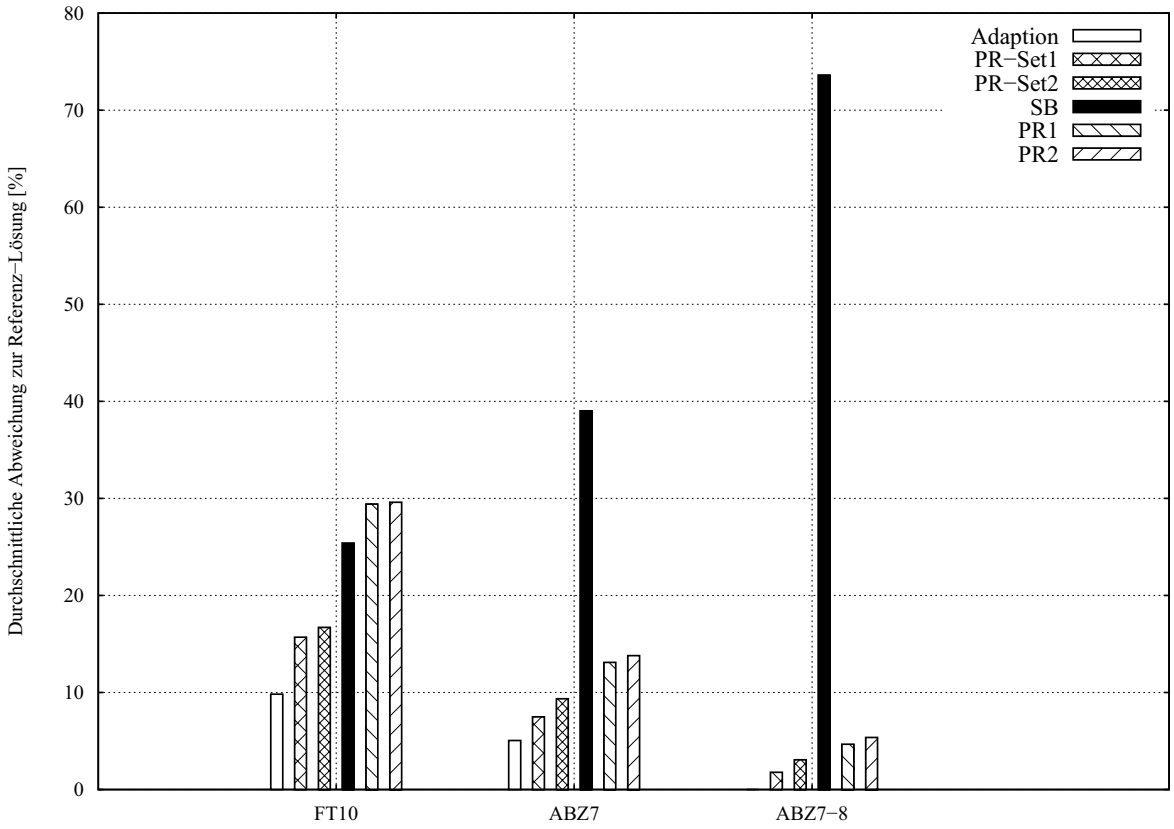


Abbildung A.10.: Ergebnis Adaptionsmechanismus

Es zeigt sich, dass die WbsGT-Lösungsgüte mit Adaptionsmechanismus steigt und WbsGT den anderen Verfahren noch deutlicher überlegen ist als ohne Adaption. Die erreichten Makespans liegen im Durchschnitt nur 5 bis maximal 10 Prozent über der Referenzlösung des Branch-and-Bound-Verfahrens. Für die Giffler-Thompson-Heuristik mit Einsatz einer festen Prioritätsregel liegt die Differenz zur Referenzlösung bei 13 bis 30 Prozent – bei Einsatz der beiden bestgeeigneten Prioritätsregeln (PR1 und PR2) für das jeweilige Problem.[17] Die Differenz ist damit mehr als doppelt so hoch wie die

[16] Als Referenzwert für die Benchmark-Probleme wird die nach einer Stunde Berechnungszeit erreichte Branch-and-Bound-Lösung verwendet. Die Lösungen wurden mittels der Brucker et al.-Variante der LiSA-Scheduling-Umgebung erzeugt (vgl. Bräsel et al., 2003). Da die hierbei die Verarbeitung von Bereitstellungszeiten nicht möglich ist, wurde für dynamische Probleme die beste übrige Lösung als Referenzwert verwendet. Dieser wird von WbsGT erzeugt.

[17] Die bestgeeigneten Prioritätsregeln waren beim FT10-Benchmark-Problem LSO (29,4%) und MWR (29,6%), beim ABZ7-Benchmark-Problem MWR (13,1%) und LSO (13,8%) sowie MWR (4,67%) und LSO (5,37%) beim ABZ7-8-Benchmark-Problem.

der WBSGT-Lösung. Auch der Wert der besten Prioritätsregelmengen (PR-Set1 und PR-Set2) liegt noch deutlich über dem des entwickelten Verfahrens.[18] Die Lösungen des Shifting-Bottleneck-Verfahrens[19] sind nur beim FT10-Benchmark-Problem konkurrenzfähig und liegen ansonsten weit hinter den Lösungen, die auf der Giffler-Thompson-Heuristik basieren.

Die Berechnungszeiten für eine Lösung liegen bei allen Heuristiken unter 10 Sekunden. Der Branch-and-Bound-Algorithmus wurde nach einer Stunde (pro Lösung) abgebrochen und die bis dahin gefundene zeitoptimale Lösung als Referenzlösung verwendet. Abbildung A.10 zeigt, das WBSGT mit Adaptionsmechanismus gute Schedules im Sinne der Zielfunktion erzeugt. Die generierten Schedules sind der bestgeeignetsten Prioritätsregel um 9 bis 20 Prozent überlegen. Im Vergleich zu den Lösungen des Shifting-Bottleneck-Algorithmus werden sogar Verbesserungen von bis zu 73 Prozent erreicht. Die Branch-and-Bound-Lösungen konnten auf 5 bis maximal 10 Prozent approximiert werden. Die zur Erzeugung des Schedules benötigte Zeit liegt dabei jedoch im Gegensatz zu der für das Branch-and-Bound-Verfahren benötigten Zeit im Bereich weniger Sekunden. Zusätzlich kann WBSGT Bereitstellungszeiten verarbeiten und ist somit für dynamische Job-Shops und Flexible-Flow-Shops geeignet.

	FT10	ABZ7
Differenz zur Branch-and-Bound-Referenzlösung	8,9%	5,0%
Absolute Verbesserung durch den Adaptionsmechanismus	3,3%	2,7%
Relative Verbesserung durch den Adaptionsmechanismus (Absolute Verbesserung · 100/Differenz ohne Adaption)	25,1%	35,1%

Tabelle A.11.: Relative Verbesserung durch mit Adaptionsmechanismus

Durch den Einsatz des Adaptionsmechanismus wurden in allen Tests Verbesserungen der WBSGT-Lösungen erreicht. Die Verbesserungen liegen dabei im Bereich von 1,4 bis 3,4 Prozent. Wie dargestellt, liegt die generierte Lösung weniger als zehn Prozent über der, mit hohem Zeit- und Rechenaufwand erstellten Branch-and-Bound-Lösung. Eine Verbesserung im Bereich von wenigen Prozent bedeutet eine starke Annäherung an die Qualität der Trainingslösungen, mit denen die Lernkomponente vorab trainiert wird.

[18] Die bestgeeigneten Prioritätsregelmengen waren beim FT10-Benchmark-Problem {SPT LSO MWR} (15,7%) und {SPT LWR LPT MWR} (16,7%), beim ABZ7-Benchmark-Problem {LSO MWR} (7,5%) und {LWR MWR} (9,35%) sowie {LSO MWR} (1,77%) und {LWR MWR} (3,07%) beim ABZ7-8-Benchmark-Problem.

[19] Eingesetzt wurde die Shifting-Bottleneck-Implementierung der LiSA-Scheduling-Umgebung (vgl. Bräsel et al., 2003).

Tabelle A.11 zeigt die durch die Adaption erreichte relative Verbesserung in Bezug auf die Referenzlösungen.

Zusammenfassend lässt sich feststellen, das WBSGT bei einem geringen Zeitbedarf Lösungen mit einem guten Zielfunktionswert erzeugt. Die erstellten Lösungen sind den Vergleichslösungen der Giffler-Thompson-Heuristik mit festen Prioritätsregeln und den Shifting-Bottleneck-Lösungen überlegen. Die Lösungsgüten des Branch-and-Bound-Verfahrens wird annähernd erreicht. Mit dem entwickelten Adaptionsmechanismus konnte der Abstand um ca. 25 Prozent verringert werden.

A.9. Klassifikation von Schedulingproblemen

Als Klassifikation von Scheduling-Problemen wird die nachfolgend angeführte Drei-Felder-Notation verwendet. „[...] A notation proposed by Graham et al. and Blazewicz et al. will be presented [...].[20]

The notation is composed of three fields $\alpha|\beta|\gamma$. They have the following meaning: The first field $\alpha = \alpha_1, \alpha_2$ describes the processor environment.

Parameter $\alpha_1 \in \{\emptyset, P, Q, R, O, F, J\}$ characterizes the type of processor used:

- $\alpha_1 = \emptyset$: single processor[21],
- $\alpha_1 = P$: identical processors,
- $\alpha_1 = Q$: uniform processors,
- $\alpha_1 = R$: unrelated processor,
- $\alpha_1 = O$: dedicated processor: open shop system,
- $\alpha_1 = F$: dedicated processor: flow shop system,
- $\alpha_1 = J$: dedicated processor: job shop system.

Parameter $\alpha_2 \in \{\emptyset, k\}$ denotes the number of processors in the problem:

- $\alpha_2 = \emptyset$: the number of processors is assumed to be variable,
- $\alpha_2 = k$: the number of processors is equal to k (k is a positive integer).

[20] Vgl. (Graham et al., 1979; Blazewicz et al., 1983).
[21] In this notation $\emptyset$ denotes an empty symbol which will be omitted in presenting problems.

The second field $\beta = \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8$ describes task and resource characteristics. Parameter $\beta_1 \in \{\emptyset, pmtn\}$ indicates the possibility of task preemption:

- $\beta_1 = \emptyset$: no preemption is allowed,
- $\beta_1 = pmtn$: preemptions are allowed.

Parameter $\beta_2 \in \{\emptyset, res\}$ characterizes additional resources:

- $\beta_2 = \emptyset$: no additional resources exist,
- $\beta_2 = res$: there are specified resource constraints [...].

Parameter $\beta_3 \in \{\emptyset, prec, uan, tree, chains\}$ reflects the precedence constraints:

- $\beta_3 = \emptyset, prec, uan, tree, chains$: denotes respectively independent tasks, general precedence constraints, uniconnected activity networks, precedence constraints forming a tree or a set of chains.

Parameter $\beta_4 \in \{\emptyset, r_j\}$ describes ready times:

- $\beta_4 = \emptyset$: all ready times are zero,
- $\beta_4 = r_j$: ready times differ per task.

Parameter $\beta_5 \in \{\emptyset, p_j = p, \underline{p} \leq p_j \leq \overline{p}\}$ describes task processing times:

- $\beta_5 = \emptyset$: tasks have arbitrary processing times,
- $\beta_5 = (p_j = p)$: all tasks have processing times equal to p units,
- $\beta_5 = (\underline{p} \leq p_j \leq \overline{p})$: no p_j is less than $\underline{p}$ or greater than $\overline{p}$.

Parameter $\beta_6 \in \{\emptyset, \tilde{d}\}$ describes deadlines:

- $\beta_6 = \emptyset$: no deadlines are assumed in the system (however, due dates may be defined if a due date involving criterion is used to evaluate schedules),
- $\beta_6 = \tilde{d}$: deadlines are imposed on the performance of a task set.

Parameter $\beta_7 \in \{\emptyset, n_j \leq k\}$ describes the maximal number of tasks constituting a job in case of job shop systems:

- $\beta_7 = \emptyset$: the above number is arbitrary or the scheduling problem is not a job shop problem,
- $\beta_7 = (n_j \leq k)$: the number of tasks for each job is not greater than k.

Parameter $\beta_8 \in \{\emptyset, \text{no-wait}\}$ describes a no-wait property in the case of scheduling on dedicated processors:

- $\beta_8 = \emptyset$: buffers of unlimited capacity are assumed
- $\beta_8 =$ no-wait: buffers among processors are of zero capacity and a job after finishing its processing on one processor must immediately start on the consecutive processor.

The third field, γ, denotes an optimality criterion (performance measure), i. e. $\gamma \in \{C_{max}, \sum C_j, \sum w_j C_j, L_{max}, \sum D_j, \sum w_j D_j, \sum E_j, \sum w_j E_j, \sum U_j, \sum w_j U_j, -\}$, where $\sum C_j = \overline{F}$, $\sum w_j C_j = \overline{F}_w$, $\sum D_j = \overline{D}$, $\sum w_j D_j = \overline{D}_w$, $\sum E_j = \overline{E}$, $\sum w_j E_j = \overline{E}_w$, $\sum U_j = U$, $\sum w_j U_j = U_w$ and „–“ means testing for feasibility whenever scheduling to meet deadlines is considered. [...]“[22]

[22] Vgl. (Blazewicz et al., 2007, Seite 68 f.).