Integrierte Steuerungsansätze für komplexe Produktionssysteme mit automatischem Transport

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

Fakultät für Mathematik und Informatik
der FernUniversität in Hagen

vorgelegt von
Dipl.-Wirt.-Inf. René Drießel

Hagen, 2011

Vorwort

Die vorliegende Arbeit ist das Ergebnis von Forschungsarbeiten an der FernUniversität in Hagen. Sie ist innerhalb meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Unternehmensweite Softwaresysteme entstanden.

Ich möchte mich besonders bei meinem Betreuer Prof. Dr. Lars Mönch für die fachliche und persönliche Unterstützung bedanken. Ohne ihn wäre die vorliegende Arbeit nicht möglich gewesen. Er weckte während meines Studiums an der Technischen Universität Ilmenau das Interesse an Steuerungsansätzen für komplexe Produktionssysteme im Halbleiterumfeld. Ebenso möchte ich mich bei Prof. Dr. Udo Buscher (Technische Universität Dresden) für die Übernahme des Zweitgutachtens bedanken.

Eine Reihe weiterer Personen haben mich bei der Erstellung der Arbeit unterstützt. Ich möchte mich bei den Mitarbeitern des Lehrgebietes Unternehmensweite Softwaresysteme, hier besonders bei Christoph Habla, Jens Zimmermann und Ralf Sprenger, für die fachliche Unterstützung bedanken. Weiterhin bedankte ich mich auch ganz herzlich bei allen, die Korrektur gelesen haben. Hier sind besonders Dr. Udo Hönig, Sandra Ficht sowie Anja und Alexander Pölck zu erwähnen. Außerdem möchte ich auch André Riemann und Kai Hübner für die große Hilfe beim Setzen der Arbeit in LaTeX danken.

Ein weiterer Dank geht auch an meine Familie und alle anderen in meinem privaten Umfeld für den Rückhalt und die Geduld. Ohne diese Unterstützung hätte ich diese Arbeit weder angefangen noch zum Abschluss gebracht.

Hagen, 2011 René Drießel

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Liste der Algorithmen

1 Einleitung

Seit dem Beginn der industriellen Fertigung ist ein fortschreitender Automatisierungsgrad innerhalb der Produktion festzustellen. Die Halbleiterfertigung ist hier ein Beispiel für eine Branche, die einen großen Automatisierungsgrad erreicht hat. In modernen Halbleiterfabriken erfolgt aufgrund der Größe und des Gewichts der zu bearbeitenden Siliziumscheiben (Wafer) der Transport mit automatischen Transportsystemen (vgl. u. a. Agrawal und Heragu (2006), Montoya-Torres (2006) sowie Foster und Pillai (2008) für einen Überblick über automatische Transportsysteme in der Halbleiterfertigung).

Zur effizienten Nutzung der Produktionssysteme muss die Maschinenbelegung und die Verwendung der Fahrzeuge sorgfältig aufeinander abgestimmt werden. In klassischen Steuerungsansätzen werden die Transportentscheidungen allerdings häufig vernachlässigt bzw. unabhängig von den Maschinenbelegungsentscheidungen getroffen. Eine gemeinsame Behandlung von Maschinenbelegungs- und Transportentscheidungen in einem integrierten Modell führt hier zu einer besseren Abbildung der Realität.

In dieser Arbeit soll ein integriertes Steuerungsverfahren, das sowohl Maschinenbelegungsentscheidungen als auch Transportentscheidungen vornimmt, entwickelt werden. Um die Anwendbarkeit des entwickelten Verfahrens zu überprüfen, erfolgt die Leistungsbewertung mit Hilfe von Simulationsmodellen, die das Produktionssystem emulieren (vgl. Mönch und Rose (2004), Mönch und Drießel (2005), Mönch u. a. (2007) und Sourirajan und Uzsoy (2007) für die Notwendigkeit der Leistungsbewertung auf Basis von Simulationsmodellen). Die dabei betrachtete Anwendungsdomäne ist die Halbleiterfertigung.

1.1 Ziele der Arbeit

Zur Entwicklung und Leistungsbewertung eines integrierten Steuerungsverfahrens werden in dieser Arbeit die folgenden Ziele verfolgt:

1. Untersuchung möglicher Verfahrensansätze zur integrierten Steuerung von komplexen Produktionssystemen mit automatischem Transport,

2. Entwicklung eines integrierten Verfahrens,

3. prototypische Implementierung des Verfahrens,

4. Entwurf und Implementierung eines Rahmenwerks zur Bewertung von integrierten Steuerungsverfahren mit Hilfe von Simulationsmodellen,

5. Einbettung des implementierten Verfahrens in das entwickelte Rahmenwerk und

6. Leistungsbewertung des entwickelten Verfahren mit Hilfe des Simulationsrahmenwerkes am Beispiel der Steuerung einer Halbleiterfertigung.

Im ersten Schritt werden mögliche Verfahrensansätze daraufhin untersucht, inwieweit sie für die integrierte Steuerung von komplexen Produktionssystemen mit automatischem Transport geeignet erscheinen. Danach erfolgt die Entwicklung und prototypische Implementierung eines integrierten Steuerungsverfahrens. Das Verfahren muss in der Lage sein, in kurzer Zeit gute Lösungen zu liefern.

Zur Leistungsbewertung des entwickelten Verfahrens ist die Implementierung eines entsprechendes Rahmenwerks notwendig. Hierbei wird das bereits existierende Rahmenwerk aus Mönch u. a. (2002) entsprechend erweitert und angepasst.

1.2 Aufbau der Arbeit

Die Arbeit besteht aus insgesamt acht Kapiteln. Zum besseren Verständnis sind neu eingeführte Begriffe **fett** markiert. Klassen, Attribute und Variablen sind *kursiv* hervorgehoben.

Im zweiten Kapitel erfolgt eine prinzipielle Beschreibung von komplexen Systemen und deren Steuerung. Es wird zunächst ein allgemeiner System- und Prozessbegriff eingeführt. Darauf aufbauend wird ein komplexes System als System von Systemen definiert. Produktionssysteme sind nach diesem Verständis ebenfalls komplexe Systeme. Es folgt ein Überblick über komplexe Produktionssysteme mit automatischem Transport und eine Betrachtung der einzelnen Teilsysteme und -prozesse. Das Kapitel schließt mit der Beschreibung der betrachteten Anwendungsdomäne.

Im dritten Kapitel werden verschiedene Steuerungsverfahren für komplexe Produktionssysteme mit automatischem Transport vorgestellt. Zunächst werden die in der Praxis am häufigsten verwendeten Verfahren auf Basis von Prioritätsregeln behandelt. Danach wird das Konzept des deterministischen Schedulings vorgestellt. Im weiteren Verlauf des Kapitels werden verschiedene exakte und heuristische Verfahren zur Lösung von Scheduling-Problemen beschrieben. Das Kapitel endet mit der Auswahl eines Verfahrensansatzes auf Basis der Shifting-Bottleneck-Heuristik (SBH) und einer Formulierung der Anforderungen an das zu entwickelnde Verfahren.

Das vierte Kapitel beschreibt zunächst die SBH und den zugrunde liegenden disjunktiven Graphen. Danach werden die Erweiterungen für komplexe Produktionssysteme mit automatischem Transport erörtert. Hierzu wird auf die notwendigen Erweiterungen des disjunktiven Graphenmodells zur Berücksichtigung der Informationen aus der Transportdomäne eingegangen, bevor mögliche Varianten des Algorithmus entwickelt werden. Im zweiten Teil des Kapitels wird beschrieben, wie die SBH zur Steuerung von komplexen Produktionssystemen mit automatischem Transport eingesetzt werden kann. Wesentlich hierbei ist, wie der disjunktive Graph in Abhängigkeit des aktuellen Systemzustandes aufzubauen ist.

Die innerhalb der SBH entstehenden Teilprobleme werden im fünften Kapitel behandelt. Zunächst wird ein allgemeiner Teilproblemlöser auf Basis von Prioritätsregeln vorgestellt. Danach werden für das neu zu betrachtende Transportteilproblem verschiedene Verfahren

auf Basis der variablen Nachbarschaftssuche vorgeschlagen. Das Kapitel endet mit einer umfangreichen Leistungsbewertung der einzelnen Verfahren auf Basis einer großen Anzahl von Testinstanzen.

Innerhalb des sechsten Kapitels wird ein Rahmenwerk zur Leistungsbewertung von Steuerungsansätzen für komplexe Produktionssysteme mit automatischem Transport vorgestellt. Die einzelnen Teilsysteme des Produktionssystems werden durch Softwarekomponenten emuliert. Hierdurch ist die Leistungsbewertung der vorgeschlagenen Verfahren in einem praxisnahen – das heißt dynamischen – Umfeld möglich.

Mit Hilfe des entwickelten Rahmenwerkes erfolgt im siebten Kapitel die Leistungsbewertung der SBH. Es werden zwei verschiedene Simulationsmodelle mit unterschiedlichen Umgebungsparametern und unterschiedlichen Steuerungsansätzen betrachtet.

Eine Zusammenfassung der erzielten Ergebnisse und ein Ausblick auf weitere Forschungsarbeiten schließen die Arbeit ab.

2 Komplexe Produktionssysteme mit automatischem Transport

In diesem Kapitel erfolgt eine Beschreibung komplexer Produktionssysteme mit automatischem Transport. In vielen komplexen Produktionssystemen werden mehr oder weniger automatisierte Transportsysteme eingesetzt. Ein Beispiel hierfür sind modernen Halbleiterfabriken.

Zunächst erfolgt eine Definition des verwendeten System- und Prozessbegriffs. Im weiteren Verlauf des Kapitels werden komplexe Produktionssysteme beschrieben, bevor auf automatische Transportsysteme eingegangen wird. Danach erfolgt eine Konkretisierung der betrachteten Anwendungsdomäne. Diese Beschreibung ist erforderlich, um in den nachfolgenden Kapiteln Steuerungs- und Ablaufplanungsverfahren zu spezifizieren.

2.1 System- und Prozessbegriff

Ein **System** besteht allgemein aus einer **Menge von Elementen** E, die über eine **Menge von Relationen** R miteinander verbunden sind (vgl. Ackoff (1971) und Simon (1962)). Jedes Element eines Systems verfügt über **Eingangs-**, **Ausgangs-** und **Zustandsgrößen**. Die Ausgangsgrößen eines Elements können wiederum Eingangsgrößen eines anderen Elements sein. Die Verbindung des Systems zur **Systemumwelt** wird ebenfalls über Ein- und Ausgangsgrößen hergestellt. Je nach Häufigkeit der Wechselwirkung mit der Umwelt wird von **offenen**, **relativ isolierten** oder **autonomen** Systemen gesprochen. Offene Systeme stehen in starker Wechselwirkung mit der Umwelt, autonome Systeme haben keine Wechselwirkung und bei relativ isolierten Systemen ist die Wechselwirkung beschränkt. In Abbildung 2.1 ist schematisch ein System mit seinen Elementen, Beziehungen, den Ein- und Ausgangsgrößen sowie der Systemumwelt dargestellt.

Der einem System zugehörige **Prozess** stellt eine Abbildung $\lambda\colon X \times Z \to Y$ zwischen den Eingangsgrößen X und den Zustandsgrößen Z auf die Ausgangsgrößen Y des Systems dar (vgl. Engelien und Stahn (1989)). Weiterhin stellt ein Prozess eine Folge von **Zustandsübergängen** $\delta\colon X \times Z \to Z$ innerhalb eines Systems dar, die durch die Eingangsgrößen verursacht werden. Unter einem Zustandsübergang wird hierbei die Veränderung einer oder mehrerer Zustandsgrößen der Elemente verstanden. Ein **Komplexes System** ist als System von Systemen definiert, das heißt, die Systemelemente der Menge E sind ihrerseits wieder Systeme (vgl. Simon (1962) und Mesarovic und Takahara (1989)). Es lässt sich demzufolge in seine Teilsysteme zerlegen.

Im weiteren Verlauf dieser Arbeit werden Systeme betrachtet, die im Zeitablauf ihren Zustand ändern und deren Zustand von außen beeinflussbar, das heißt **steuerbar**, ist. Nach Reinisch (1974) werden diese Systeme als **kybernetische Systeme** bezeichnet. Die Steuerung eines kybernetischen Systems erfolgt mit Hilfe eines **Steuerungssystems**. Das

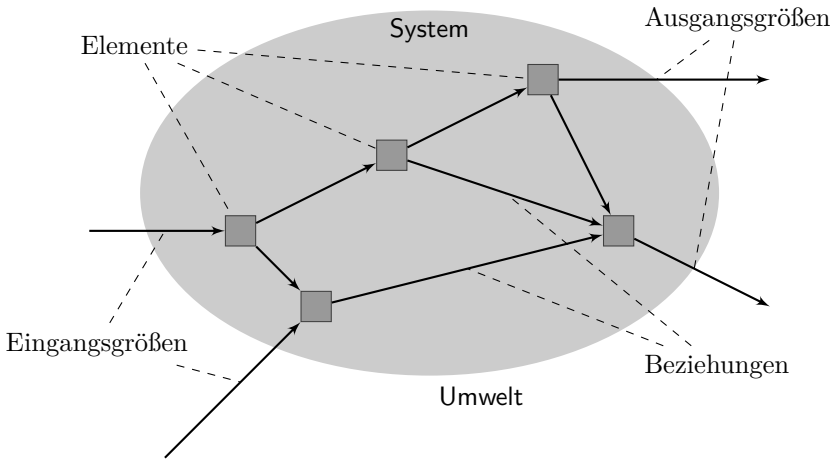

Abbildung 2.1: Schematische Darstellung eines Systems

kybernetische System wird in diesem Zusammenhang auch als **Basissystem** bezeichnet. Die Notwendigkeit der Steuerung ergibt sich aus der Tatsache, dass kybernetische Systeme zu einem bestimmten Zweck geschaffen wurden. Der zugehörige **Steuerungsprozess** versucht, den **Basisprozess** so zu steuern, dass der Zweck, zu dem das Basissystem geschaffen wurde, erfüllt wird. **Steuerungsgrößen** sind in diesem Zusammenhang alle Ausgangsgrößen des Steuerungssystems, die gleichzeitig Eingangsgrößen des Basissystems sind. Mit ihnen erfolgt die Steuerung des Basissystems.

Für die Steuerung im engeren Sinne ist dabei keine Rückmeldung des Basisprozesses erforderlich. Erfolgt vom Basisprozess eine Rückmeldung an den Steuerungsprozess, spricht man innerhalb des Regelkreisparadigmas von **Regelung** (Reinisch (1974) und Van de Vegte (1993)). Diese ist notwendig, wenn bestimmte Eingangsgrößen der Systemumwelt das Basissystem beeinflussen können. Diese Eingangsgrößen werden auch als **Störgrößen** bezeichnet. Hierbei erfolgt eine Messung und Rückmeldung der Ausgangsgrößen des Basissystems an das Steuerungssystem. Auf dieser Basis ist es dem Steuerungssystem möglich, die Steuerungsgrößen anzupassen und den Einfluss der Störgrößen zu verringern. Im weiteren Verlauf der Arbeit wird jedoch unabhängig von der Richtung der Informationsflüsse von Steuerung gesprochen.

In Abbildung 2.2 erfolgt eine schematische Darstellung der Zusammenhänge zwischen Steuerungs- und Basissystem innerhalb eines Regelkreises. Neben den Eingangsgrößen X und den Ausgangsgrößen Y sind die Steuerungsgrößen C und die Störgrößen W dargestellt.

2.2 Steuerung komplexer Systeme

In Reinisch (1974) erfolgt eine Klassifikation von Steuerungsarten nach dem Einfluss des Menschen auf die Steuerung. Der Einfluss des Menschen nimmt in der folgenden Aufzählung von oben nach unten ab. Es wird unterschieden zwischen

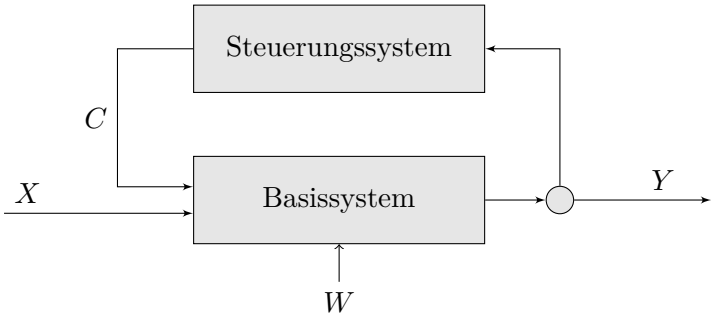

Abbildung 2.2: Steuerungs- und Basissystem in einem Regelkreis

- der **Handsteuerung** durch den Menschen,
- einer **automatisierten Steuerung** unter Beteiligung des Menschen und
- der **automatischen Steuerung** ohne Beteiligung des Menschen.

Bei der automatischen Steuerung kann wiederum zwischen einer **voreingestellten automatischen Steuerung** und der **adaptiven automatischen Steuerung** (Regelung) unterschieden werden. In einer voreingestellten automatischen Steuerung sind alle Steuerungsgrößen fest eingestellt und werden vom Steuerungssystem nicht verändert. Das Steuerungssystem einer adaptiven Steuerung entscheidet hingegen selbstständig über die Anpassung der Steuerungsgrößen.

Im weiteren Verlauf werden nur adaptive automatische Steuerungen betrachtet. Hierzu ist es notwendig, die Ausgangsgrößen zu messen und dem Steuerungssystem zur Verfügung zu stellen. Der zugehörige Steuerungsprozess wandelt diese Informationen in entsprechende Steuerungsgrößen für das Basissystem um. In diesem Zusammenhang müssen Entscheidungen automatisch getroffenen werden. Die **Steuerungsentscheidung** bewirkt die Anpassung von Steuerungsgrößen über **Steuerungsparameter**. Hierbei liegt typischerweise eine Menge von möglichen Entscheidungen, die **Steuerungsalternativen**, vor. Das Problem besteht nun darin, aus dieser Menge eine Steuerungsalternative auszuwählen, damit der Zweck des Systems bestmöglich erfüllt wird.

Zur Beurteilung einer einzelnen Steuerungsalternative ist es notwendig, dass das Steuerungssystem ein **internes Modell** des Basissystems und des Basisprozesses besitzt. Das interne Modell enthält entsprechende Repräsentationen der Systemelemente, die für die Bewertung der Steuerungsalternative notwendig sind (vgl. Bierwirth (2000)). Anhand des internen Modells, das auch als dynamisches Prozessmodell bezeichnet wird, ist es möglich, die Auswirkungen einer gewählten Steuerungsalternative bezüglich der Zweckerfüllung zu überprüfen, bevor sie zur Steuerungsentscheidung wird. Gleichzeitig kann geprüft werden, ob die gewählten Größen für die Steuerungsparameter zulässig sind, das heißt, nicht im Widerspruch zu dem Modell stehen.

Im Vergleich zur Steuerung gewöhnlicher kybernetischer Systeme (siehe auch Reinisch (1974), Van de Vegte (1993) und Bierwirth (2000)) lässt sich bei der Steuerung komplexer

Systeme die Tatsache ausnutzen, dass diese aus verschiedenen **Teilsystemen** bestehen. Die Dekomposition in einzelne Teilsysteme führt dazu, dass das interne Modell des Basissystems ebenfalls in Teilmodelle zerlegt werden kann. Jedes dieser Teilmodelle bildet unterschiedliche Aspekte des Basissystems ab.

Nach Mönch (2006) kann ein komplexes kybernetisches System als dezentralisiert aufgefasst werden, da es per Definition unterschiedliche Teilsysteme enthält, die in Interaktion stehen, um einen gemeinsames Zweck zu erfüllen. Abhängig vom Dezentralisierungsgrad können unterschiedliche Steuerungsstrategien eingesetzt werden. Eine fast vollständig zentrale Steuerung lässt den beteiligten Teilsystemen nur einen geringen Grad von Autonomie. Im Gegensatz dazu ist bei einer vollständig dezentralen Steuerung jedes Teilsystem eigenständig für seine Steuerung verantwortlich. Zwischen beiden Extremen stehen verteilte hierarchische Steuerungssysteme, bei denen auf Basis einer übergeordneten Steuerung die einzelnen Teilsysteme gesteuert werden. Allen dezentralen Systemen ist gemein, dass die einzelnen Teilsysteme geeignet koordiniert werden müssen, um die Ziele für das Gesamtsystem zu erreichen.

2.3 Komplexe Produktionssysteme mit Transport

Komplexe Produktionssysteme sind komplexe Systeme zur Produktion von **Gütern** (vgl. Corsten (2007)). Ein komplexes Produktionssystem mit Transport kann in ein **Bearbeitungssystem** und ein **Transportsystem** zerlegt werden, die ihrerseits wieder komplexe Systeme sind und in jeweils ein Basis- und ein Steuerungssystem zerlegt werden können. Das **Produktionssteuerungssystem** besteht aus dem **Bearbeitungs-** und dem **Transportsteuerungssystem**. Das **Produktionsbasissystem** besteht analog dazu aus dem **Bearbeitungs-** und dem **Transportbasissystem**. Abbildung 2.3 stellt die Zusammenhänge zwischen Bearbeitungs- und Transportsystem im Rahmen des gesamten Produktionssystems dar. Im weiteren Verlauf erfolgt eine Beschreibung des Bearbeitungs- und Transporbasissystems, bevor näher auf das Produktionssteuerungssystem eingegangen wird.

2.3.1 Beschreibung des Bearbeitungsbasissystems

Die Systemelemente des Bearbeitungsbasissystems stellen die **Maschinen** dar. Die Herstellung eines Gutes ist durch einen **Arbeitsplan** und eine **Stückliste** beschrieben. Die Stückliste beschreibt die für die Herstellung des Gutes notwendigen Ausgangsgüter. Der Arbeitsplan besteht aus einer Folge von **Arbeitsgängen**, welche durch die Maschinen ausgeführt werden, um das Gut aus den Ausgangsgütern zu erstellen. Jeder Arbeitsplan bestimmt die Reihenfolge der Arbeitsgänge, die durchzuführen sind, um ein Gut herzustellen. Neben den Maschinen kann ein Arbeitsgang weitere **sekundäre Ressourcen**, wie Werkzeuge und andere Hilfsmittel, benötigen. Außerdem müssen alle weiteren notwendigen Prozessbedingungen wie das Einhalten einer bestimmten Temperatur, das Vorhandsein eines Reinraums oder eine maximale Anzahl gleichzeitig bearbeitbarer Güter beachtet werden. Mit jedem Arbeitsgang ist eine **Bearbeitungsdauer** assoziert, welche angibt wie lange die Bearbeitung auf einer Maschine dauert. Im Falle von **flexiblen**

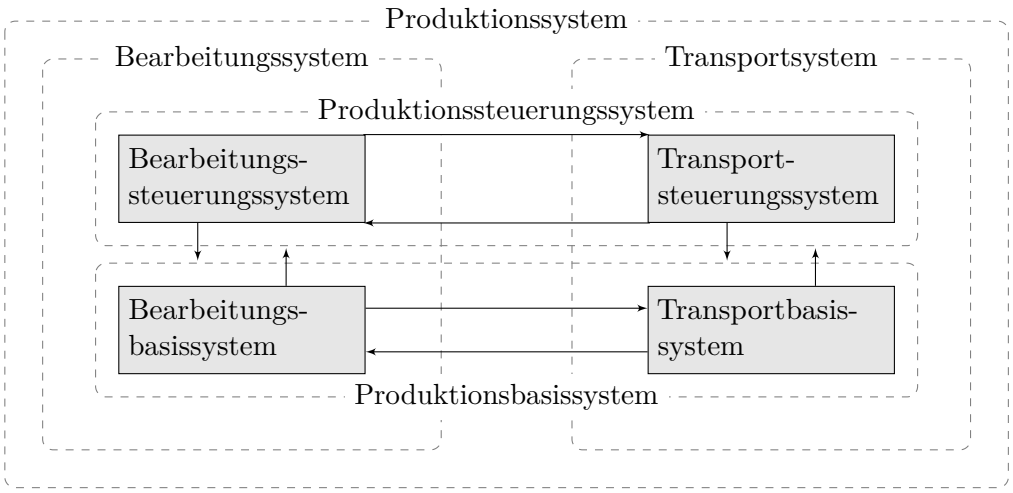

Abbildung 2.3: Einordnung des Bearbeitungs- und Transportsystems

Maschinen lassen sich unterschiedliche Arbeitsgänge auf einer Maschine ausführen. Hierbei treten **Rüstzeiten** auf, die vom jeweiligen **Rüstzustand**, das heißt, von der Konfiguration der Maschine, abhängig sind.

Typischerweise werden mehrere gleichartige Güter nicht einzeln produziert, sondern zu **Losen** zusammengefasst, um die benötigten Rüstzeiten zu verringern. Im Rahmen der Herstellung eines Loses ist eine Folge von **Prozessschritten** innerhalb des Produktionsbasissystems durchzuführen. Die Prozessschritte innerhalb des Bearbeitungsbasissystems werden im weiteren Verlauf als **Bearbeitungsschritte** bezeichnet. Jeder Bearbeitungsschritt ist einem Arbeitsgang zugeordnet und wird durch diesen beschrieben. Die Reihenfolge der einzelnen Bearbeitungsschritte ergibt sich durch den Arbeitsplan, welcher die Herstellung der Güter beschreibt.

Zu den Eingangsgrößen des Bearbeitungsbasissystems zählen unter anderem Terminvorgaben, die zu produzierenden Lose sowie die für die Produktion notwendigen Elemente. Je nach Produktionssystem können dies zur Produktion notwendige Rohstoffe, vorgefertigte Baugruppen oder Energie sein. Die Ausgangsgrößen sind neben den fertigen Produkten und Halbfertigprodukten Nebenprodukte und Abfallstoffe.

Bezüglich der Organisationsform von Produktionssystemen lassen sich nach Corsten (2007) die folgenden drei Fertigungsarten unterscheiden: **Fließfertigung**, **Werkstattfertigung** und **flexible Werkstattfertigung**. Bei der Fließfertigung sind die Maschinen entsprechend der zu bearbeitenden Arbeitspläne angeordnet. Diese Anordnung der Maschinen ist bei der Massenproduktion mit immer gleichen Arbeitsplänen sinnvoll.

Im Rahmen der Werkstattfertigung werden die Maschinen unabhängig vom Arbeitsplan nach dem Verrichtungsprinzip angeordnet. Diese Anordnung der Maschinen ist dann sinnvoll, wenn unterschiedliche Güter mit verschiedenen Arbeitsplänen hergestellt werden und die Kosten für die Aufstellung zusätzlicher Maschinen zu hoch sind, um eine Fließfertigung für jeden Arbeitsplan zu realisieren. Speziell bei der Fertigung einzelner

Güter und Kleinserien ist diese Organisationsform sinnvoll. Allerdings sind die Kosten zur Organisation des Transports zwischen den einzelnen Maschinen höher als bei der Fließfertigung.

Flexible Werkstattfertigungssysteme versuchen, einen Kompromiss zwischen der reinen Fließfertigung und der Werkstattfertigung zu erreichen. In flexiblen Werkstattfertigungssystemen sind die Maschinen nach ihrer Funktionalität in **Maschinengruppen** angeordnet (siehe auch Pinedo (2001)). Hierdurch ist es möglich, dass ein bestimmter Arbeitsgang eines Loses auf unterschiedlichen Maschinen ausgeführt werden kann. Man spricht in diesem Zusammenhang auch von Maschinengruppen mit **parallelen Maschinen**.

Komplexe Produktionssysteme sind häufig Werkstattfertigungssysteme und nach Mönch (2006) durch die folgenden Eigenschaften gekennzeichnet:

- eine große Anzahl von Maschinen und Maschinengruppen und
- die Vielfalt und große Anzahl der Arbeitspläne und Lose.

Im weiteren Verlauf werden komplexe flexible Werkstattfertigungssysteme näher betrachtet. Moderne Halbleiterfertigungssysteme sind Beispiele für flexible Werkstattfertigungssysteme (siehe auch Mason u. a. (2002)).

2.3.2 Beschreibung des Transportbasissystems

Das Transportsystem kann analog zum Bearbeitungssystem in ein Transportsteuerungssystem und ein Transportbasissystem zerlegt werden. Es organisiert während der Produktion den Transport zwischen den einzelnen Maschinen. Dies bedeutet, dass nach jedem durchgeführten Bearbeitungsschritt das Los von der aktuellen Maschine zu der Maschine transportiert werden muss, die den nachfolgenden Bearbeitungsschritt ausführt. Die dabei durchzuführenden Prozessschritte werden im weiteren Verlauf **Transportschritte** genannt. In Anlehnung an Reinisch (1974) können bezüglich des Automatisierungsgrades des Transportbasissystems die folgenden Stufen unterschieden werden:

- **manueller Transport**: Hierbei erfolgt der Transport vollständig durch den Menschen.
- **automatisierter Transport**: Der Transport wird durch technische Hilfsmittel (zum Beispiel Gabelstapler oder Hubwagen) unterstützt, die von Menschen gesteuert werden.
- **automatischer Transport**: Der Transport erfolgt durch technische Hilfsmittel, ohne menschliche Steuerung.

Im Folgenden werden automatische Transportsysteme betrachtet, wie sie in modernen Halbleiterfabriken zu finden sind. Prinzipiell lassen sich dabei zwei Arten von automatischen Transportsystemen unterscheiden (siehe auch Nazzal und Bodner (2003)):

- Transportsysteme, die den Transport mittels **Förderbändern** durchführen und

- Transportsysteme, die den Transport mit Hilfe von **Fahrzeugrobotern** organisieren.

Im ersten Fall sind die Maschinen mittels eines Förderbandes miteinander verbunden. Nach der Bearbeitung auf einer Maschine wird das Los auf das Förderband gelegt und von diesem transportiert. Wenn es an der nächsten Maschine angekommen ist, wird es heruntergenommen. Einen Überblick über bestehende Arbeiten zu Transportsystemen auf Basis von Förderbändern in Halbleiterfabriken findet man in Nazzal und El-Nashar (2007).

Bei fahrzeugbasierten Transportsystemen erfolgt der Transport mittels Fahrzeugrobotern, die das Los am Ausgangsort abholen, zum Zielort transportieren und dort abladen. Die Fahrzeuge fahren hierbei typischerweise entlang von festgelegten **Strecken**. Diese Strecken können beispielsweise Schienen, optische Spuren oder Leitdrähte sein.

Die Wahl des richtigen Transportsystems hängt stark vom betrachteten Produktionssystem und der Organisationsform innerhalb des Bearbeitungsbasissystems ab. Die Auswahl eines geeigneten Transportsystems für Halbleiterfabriken sowie Fragen des **Produktionssystemlayouts** werden in der Literatur intensiv untersucht (vgl. Nazzal (2006), Cardarelli und Pelagagge (1995), Kurosaki u. a. (1997), Plata (1997) und Mackulak u. a. (1998)). Im weiteren Verlauf werden **Einschienensysteme** betrachtet, wie sie häufig in modernen 300-mm-Halbleiterfabriken zum Einsatz kommen (siehe auch Lin u. a. (2004), Chung und Jeng (2003), Agrawal und Heragu (2006) und Liao und Fu (2004)).

Die Systemelemente des Transportbasissystems stellen in diesem Zusammenhang die einzelnen Fahrzeuge und die möglichen Strecken dar. Das Transportbasissystem kann aus technischen Gründen, zum Beispiel aufgrund der Energieversorgung der Fahrzeuge, in einzelne Transportbereiche unterteilt sein. Diese Transportbereiche sind physisch abgegrenzte Bereiche innerhalb des Produktionssystems. Abhängig vom Layout des Produktionssystems können die einzelnen Fahrzeuge dabei nur bestimmte Transportbereiche befahren.

Zwischen zwei Bearbeitungsschritten eines Loses gibt es mindestens einen, häufig jedoch mehrere Transportschritte. Jedem Transportschritt sind dabei der Ausgangsort, der Zielort sowie die für den Transport verwendbaren Fahrzeuge zugeordnet. Als Orte werden in diesem Zusammenhang Belade- und Entladevorrichtungen an den einzelnen Maschinen sowie Zwischenlager für die Lose verstanden. Innerhalb der Halbleiterfertigung werden diese Zwischenlager auch **Stocker** genannt. Jeder dieser Stocker besitzt eine entsprechende **Kapazität**, die angibt, wie viele Lose aufgenommen werden können.

Zu den Eingangsgrößen des Transportbasissystems zählen die durchzuführenden Transportschritte, das heißt die zu transportierenden Lose mit den Ausgangs- und Zielorten. Hierbei ist zu beachten, dass im Rahmen einer flexiblen Werkstattfertigung das Bearbeitungssystem dem Transportsystem die durchzuführenden Transportschritte vorgibt, da diese erst feststehen, nachdem entschieden wurde, auf welcher Maschine das Los bearbeitet werden soll. Die Ausgangsgrößen sind die transportierten Lose an ihren Zielorten. Die fertig transportierten Lose stellen wiederum Eingangsgrößen für das Bearbeitungssystems dar.

2.3.3 Beschreibung des Produktionssteuerungssystems

Wie bereits in Abschnitt 2.2 beschrieben, kann ein Steuerungssystem in Teilsysteme zerlegt werden, die sich an der Dekomposition des Basissystems orientieren. Im Falle der Steuerung komplexer Produktionssysteme mit automatischem Transport kann das Steuerungssystem zunächst in ein Bearbeitungssteuerungssystem und ein Transportsteuerungssystem dekomponiert werden (siehe auch Abbildung 2.3). Das Bearbeitungssteuerungssystem lässt sich dabei in einzelne **Maschinengruppensteuerungssysteme** und diese wiederum in **Maschinensteuerungssysteme** dekomponieren.

Innerhalb des Modells des Transportsteuerungssystems können die einzelnen Transportbereiche als **Transportbereichssteuerungssysteme** getrennt behandelt werden. In diesen kann eine Dekomposition in einzelne **Fahrzeugsteuerungssysteme** vorgenommen werden.

Hierbei ist zu bedenken, dass die Maschinengruppenunterteilung des Bearbeitungsbasissystems zunächst nur eine logische und keine physische Unterteilung innerhalb des Bearbeitungssystems darstellt. Es ist jedoch häufig so, dass die Maschinengruppen mit den einzelnen Transportbereichen korrespondieren bzw. dass nur ein Transportbereich für alle Maschinengruppen existiert.

Innerhalb des internen Modells eines Steuerungssystems sind die entscheidungsrelevanten Systemelemente abzubilden. Bei der Betrachtung von komplexen Produktionssystemen mit automatischem Transport gehören dazu unter anderem

- die einzelnen Maschinen und Maschinengruppen mit ihren Rüstzuständen und Verfügbarkeiten,
- die herzustellenden Güter und Arbeitspläne,
- die einzelnen Lose mit ihren als nächstes auszuführenden Prozessschritten,
- die Strecken und die Stocker sowie
- die Fahrzeuge mit ihren aktuellen Positionen.

Das Produktionssystem als Ganzes erhält externe Vorgaben in Form von **Führungs-** und **Zielgrößen** (siehe Abbildung 2.4). Während Führungsgrößen feste Werte enthalten, die einzuhalten ist, sind Zielgrößen Größen, die es zu minimieren oder zu maximieren gilt. Beide Größenarten beeinflussen die konkreten Entscheidung des Steuerungssystems und werden aus den Zielen des Produktionssystems abgeleitet. Die Ziele selbst ergeben sich wiederum aus dem Wirtschaftlichkeitsprinzip. Mögliche Ziele, die durch das Steuerungssystem verfolgt werden können, sind beispielsweise (siehe auch Kurbel (2003))

- Kostenziele,
- Zeitziele und
- Mengenziele.

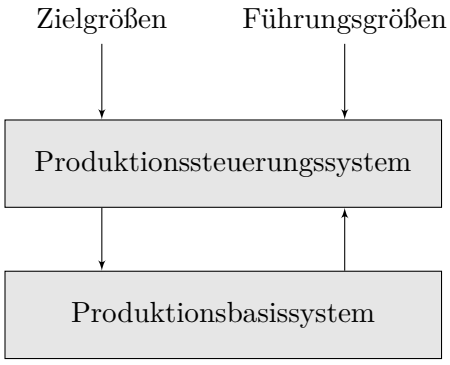

Abbildung 2.4: Führungs- und Zielgrößen

Welche Zielgrößen zur Operationalisierung der Ziele verwendet werden, ist von der Art des Produktionssystems abhängig. Die Kostenziele werden meistens indirekt durch eine Maximierung der Kapazitätsauslastung abgebildet. Als Beispiele für Zielgrößen, die Zeitziele verfolgen, gelten unter anderem die Minimierung von Rüst-, Warte und Durchlaufzeiten sowie die Minimierung von Terminüberschreitungen (Verspätungen). Im Falle von Mengenzielen lässt sich die Minimierung von Lagerbeständen und Fehlmengen sowie die Erhöhung des Durchsatzes anführen. Mit den genannten Zielgrößen lassen sich in Abhängigkeit vom Produktionssystem geeignete Ziele beschreiben. Hierbei ist jedoch zu beachten, dass die einzelnen Zielgrößen in der Regel gegenläufig sind. Zu den mögliche Führungsgrößen, die aus den Zielen abgeleitet werden können, gehören

- Mengenvorgaben, wenn die Mengen nicht bereits als Zielgrößen vorgegeben werden.
- Terminvorgaben und
- Qualitätsvorgaben.

Die Mengenvorgaben umfassen unter anderem die Anzahl der zu fertigenden Produkte und die Limitierung der Menge der Halbfabrikate innerhalb des Produktionssystems. Die Terminvorgaben dienen zur Einhaltung der gewünschten Kundenliefertermine.

2.4 Bearbeitungs- und Transportsyteme in Halbleiterfabriken

Die im weiteren Verlauf der betrachtete Anwendungsdomäne ist die Halbleiterfertigung. In diesem Abschnitt wird zunächst das Produktionsbasissystems, welches zur Herstellung von integrierten Schaltkreisen notwendig ist, beschrieben. Im Anschluss daran erfolgt eine Beschreibung des internen Modells des Produktionssteuerungssystems und darauf aufbauend eine Klassifikation des zu lösenden Steuerungsproblems.

2.4.1 Herstellung integrierter Schaltkreise

Integrierte Schaltkreise werden durch die Kombination verschiedener technischer und chemischer Verfahren aus kreisrunden Siliziumscheiben (**Wafer**) hergestellt. Mehrere Wafer werden hierbei zu einem Los zusammengefasst. Da integrierte Schaltkreise typischerweise aus mehreren Schichten bestehen, werden immer wieder dieselben oder ähnliche Verfahren auf die einzelnen Wafer angewendet. Aufgrund der immer wiederkehrenden Arbeitsgänge, der hohen Kosten der einzelnen Maschinen sowie der großen nachgefragten Stückzahlen erfolgt die Produktion typischerweise als flexible Werkstattfertigung. Der gesamte Herstellungsprozess wird grob in folgende Schritte

- Waferfertigung,
- Wafertest,
- Montage oder Verpackung und
- Abschlusstest

unterteilt (siehe auch Uzsoy u. a. (1992)). Die Waferfertigung ist hierbei der komplexeste und kapitalintensivste Teil und bildet den Kernprozess. Die Struktur der integrierten Schaltkreise wird in einem Reinraum auf die einzelnen Wafer aufgebracht. Auf einem Wafer werden damit mehrere hundert Schaltkreise erstellt. Beim Wafertest werden die produzierten Schaltkreise einem stichprobenartigen Test unterzogen. Fehlerhafte Schaltkreise werden dabei markiert. Diese zwei Teilschritte werden als Front-End-Operationen bezeichnet. Bei den restlichen beiden Back-End-Operationen werden die funktionierenden Schaltkreise aus den Wafern herausgebrochen, mit Plastik- bzw. Keramikgehäusen versehen und einem Abschlusstest unterzogen. Danach können die fertigen Schaltkreise an die Kunden versendet werden.

In Abbildung 2.5 ist ein vereinfachtes Layout einer Halbleiterfabrik dargestellt. Weiterhin ist ein vereinfachter Losdurchlauf durch dieses Produktionssystem eingezeichnet. Die einzelnen Maschinen sind entsprechend der flexiblen Werkstattfertigung nach Maschinengruppen zusammengefasst.

Um den Verschmutzungsgrad möglichst gering zu halten, werden die Wafer in sogenannten **FOUPs** (FOUP = engl. *Front-Open-Unified-Pods*) transportiert. Innerhalb eines FOUPs herrschen Reinraumbedingungen vor. Über spezielle Belade- und Entladevorrichtungen werden die Wafer automatisch aus den FOUPs genommen und in die Maschine geladen bzw. entladen. Da in den Maschinen ebenfalls Reinraumbedingungen vorliegen, müssen die Reinraumanforderungen an das gesamte Produktionssystem nicht mehr so strikt sein. Zur zwischenzeitlichen Lagerung der FOUPs werden die Stocker innerhalb des Transportsystems verwendet. Zur Nachverfolgung innerhalb des Steuerungssytems ist jedem FOUP eine eindeutige Identifikationsnummer zugeordnet. Diese Identifikationsnummer ist als Label sowohl maschinenlesbar (meist in Form von Barcodes bzw. mittels RFID) als auch lesbar durch den Menschen (meist in alphanumerischer Form) an dem FOUP angebracht. Auf Basis der Identifikationsnummer lässt sich die aktuelle

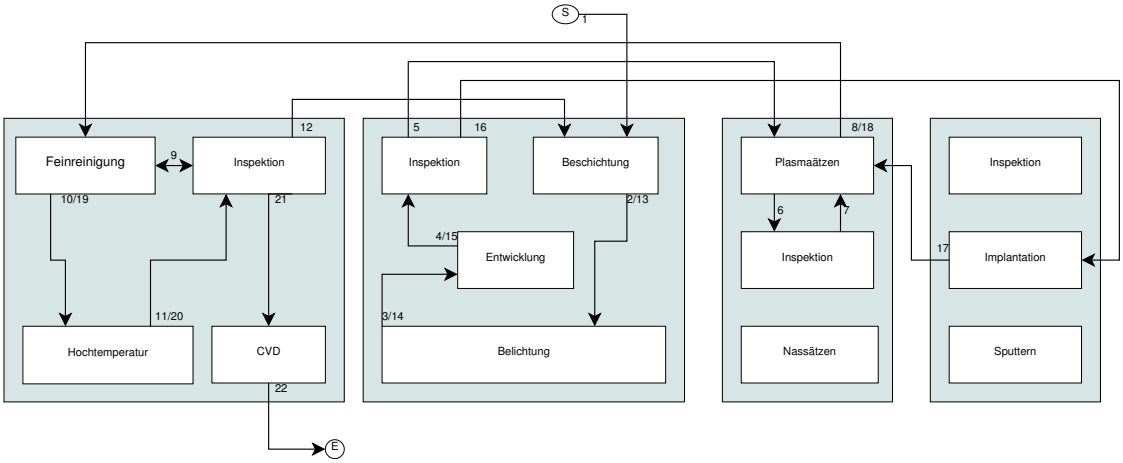

Abbildung 2.5: Layout einer Halbleiterfabrik (in Anlehnung an Mönch und Gmilkowsky (2001))

Position eines FOUPs innerhalb des Transportbasissystems nachverfolgen (siehe hierzu auch SEMATECH (2003) sowie Foster und Pillai (2008)).

Innerhalb des letzten Jahrzehnts erfolgte die Umstellung von Wafern mit 200 Millimeter Durchmesser auf Wafer mit einem Durchmesser von 300 Millimeter. Die Standardlosgröße von 25 Wafern pro Los kann nun bis zu 15 Kilogramm wiegen (siehe auch Mason u. a. (2004)). Die einzelnen Lose sind damit zu schwer für den sicheren manuellen bzw. automatisierten Transport geworden. Aus diesem Grund kommen in solchen Produktionssystemen automatische Transportsysteme zum Einsatz. In den letzten Jahren haben sich hier schienenbasierte Transportsysteme durchgesetzt. Die Transportfahrzeuge fahren dabei auf Einschienenbahnen zwischen den einzelnen Stockern (siehe auch Chung und Jeng (2003) und Liao und Fu (2004)). Die Schienen sind aus Platzgründen meist an der Decke der Fabrik angebracht. Jeder einzelne FOUP hat eine mechanische Kupplung, mit der er von einem Transportfahrzeug aufgenommen und transportiert werden kann. Abbildung 2.6 zeigt ein Foto eines solchen Fahrzeugs zusammen mit einem FOUP. Neben dem Schienensystem ist hier auch erkennbar, wie der FOUP über die mechanische Kupplung mit dem Fahrzeug verbunden ist.

In einer typischen Halbleiterfabrik sind alle Maschinen einer Maschinengruppe räumlich benachbart. Da die Maschinen sehr eng stehen und häufig nur ein kleiner Gang für die Arbeiter existiert, wird diese Anordnung auch Tunnel (engl. *Bay*) genannt. Jeder Maschinengruppe ist weiterhin ein zentraler Stocker zugeordnet, in dem die Lose so lange warten, bis sie von einer Maschine zur Bearbeitung ausgewählt werden. Nachdem ein Los ausgewählt wurde, wird es durch ein Fahrzeug von dort zu der entsprechenden Maschine gebracht. Dieser zentrale Stocker steht typischerweise am Anfang eines solchen Maschinentunnels (engl. *Intrabay*). Die einzelnen Stocker der Maschinentunnel sind wiederum durch einen Fabriktunnel (engl. *Interbay*) miteinander verbunden. Der Transport von Losen von einer Maschinengruppe zur nächsten erfolgt durch den Fabriktunnel. Das Transportsystem ist häufig hierarchisch organisiert. Jedem Tunnel ist hierbei ein Transportbereich zugeordnet, der für die Steuerung der jeweiligen Fahrzeuge innerhalb

Abbildung 2.6: Foto eines Transportfahrzeugs aus Foster und Pillai (2008)

des Bereiches verantwortlich ist. Es existieren jedoch auch Transportsysteme, in denen es nur einen einzigen Transportbereich gibt und die einzelnen Fahrzeuge die gesammten Fabrik befahren können.

Für weitere Details zur Halbleiterfertigung sei auf die Arbeiten von Uzsoy u. a. (1992), Uzsoy u. a. (1994) und Pfund u. a. (2006) sowie Foster und Pillai (2008) verwiesen.

2.4.2 Beschreibung des Steuerungsproblems

An dieser Stelle erfolgt eine formale Beschreibung der Modellannahmen für das betrachtete Produktionssystem. Im Anschluss daran wird das innerhalb dieser Arbeit betrachteten Steuerungsproblem klassifiziert.

Beschreibung des Modells für das Bearbeitungssteuerungssystems

Innerhalb des Bearbeitungssystems liegen m Maschinen vor, die jeweils einer Maschinengruppe zugeordnet werden können. Eine Maschinengruppe besteht dabei aus identischen Maschinen, die alle dieselben Arbeitsgänge durchführen können. Auf den Maschinen sollen n Lose $J_1, \ldots, J_n$ bearbeitet werden (J = engl. *Job*). Jedes Los besitzt einen **Einsteuertermin** r_j (r = engl. *Release Date*), einen **geplanten Aussteuertermin** d_j ($d =$ (engl. *Due Date*)) sowie ein **Gewicht** w_j ($w =$ engl. *Weight*), welches die Wichtigkeit des Loses angibt. Der **Fertigstellungszeitpunkt** eines Loses J_j wird mit c_j ($c =$ engl. *Completion Time*) und die **Verspätung** mit $T_j := \max(c_j - d_j, 0)$ ($T =$ engl. *Tardiness*) bezeichnet. Weiterhin ist jedem Los eine Folge mit n_j Bearbeitungsschritten O_{ij} ($i = 1, \ldots, n_j$) zugeordnet ($O =$ engl. *Operation*). Die Reihenfolge dieser Bearbeitungs-

schritte ist durch das Gut und dessen Arbeitsplan, zu dem das Los gehört, vorgegeben. Da integrierte Schaltkreise aus mehreren Ebenen bestehen, muss ein Los mehrmals von derselben Maschinengruppe bearbeitet werden. In diesem Zusammenhang spricht man von **schleifenförmigen Arbeitsplänen**.

Aufgrund der Arbeitspläne ergeben sich **Vorrangbeziehungen** zwischen den einzelnen Bearbeitungsschritten eines Loses. In diesem Fall bedeutet $O_{ij} \rightarrow O_{uv}$, dass O_{ij} fertiggestellt sein muss, bevor O_{uv} angefangen werden kann. In diesem Zusammenhang kann zwischen **direkten** und **indirekten Vorgängern** bzw. **Nachfolgern** eines Bearbeitungsschritts unterschieden werden. Ein Bearbeitungsschritt O_{ij} ist ein direkter Vorgänger eines anderen Bearbeitungsschritts O_{uv}, wenn eine Vorrangbeziehung $O_{ij} \rightarrow O_{uv}$ existiert. Der Bearbeitungsschritt O_{uv} ist analog dazu der direkte Nachfolger von O_{ij}. Existiert eine zusätzliche Vorrangbeziehung $O_{uv} \rightarrow O_{lk}$ zwischen O_{uv} und einem weiteren Bearbeitungsschritt O_{lk}, dann ist O_{ij} ein indirekter Vorgänger von O_{lk} und O_{lk} ein indirekter Nachfolger O_{ij}. Für einen Bearbeitungsschritt lassen sich alle indirekten Vorgänger und Nachfolger rekursiv ermitteln.

Jeder Bearbeitungsschritt ist aufgrund des zugehörigen Arbeitsgang einer Maschinengruppe zugeordnet, die diesen bearbeiten kann. Mehrere aufeinanderfolgende Bearbeitungsschritte eines Loses J_j können durch ein und dieselbe Maschinengruppe bedient werden. Ein Los kann dabei nicht auf mehreren Maschinen gleichzeitig bearbeitet werden.

Es wird angenommen, dass die Bearbeitungszeit, die zur Durchführung eines Bearbeitungsschritts auf einer Maschine benötigt wird, deterministisch ist und die Bearbeitung nicht unterbrochen werden kann. Des Weiteren wird angenommen, dass das Produktionssystem **einstufig** ist. Dies bedeutet, dass Stücklisten für einzelne Lose nicht berücksichtigt werden.

In einzelnen Maschinengruppen können mehrere Lose gemeinsam in einem **Batch** auf einer Maschine bearbeitet werden. Die **maximale Batch-Größe** gibt die maximale Anzahl von Losen an, die von einer Maschine gleichzeitig bearbeitet werden kann. Es kann zwischen **simultaner** und **sequentieller** Abarbeitung unterschieden werden (Potts und Kovalyov (2000)). Bei der simultanen Abarbeitung werden die Lose gleichzeitig bearbeitet. Die sequentielle Abarbeitung wird nach Potts und Kovalyov (2000) noch einmal in zwei Varianten unterschieden. Im Fall von **Batch-Verfügbarkeit** können die Lose des Batches erst weiterbearbeitet werden, wenn der gesamte Batch beendet ist. Bei **Los-Verfügbarkeit** kann ein Los jedoch direkt nach Fertigstellung weiterbearbeitet werden und muss nicht auf die anderen Lose warten. Lose, die für die gemeinsame Bearbeitung bestimmte Voraussetzungen (zum Beispiel den gleichen Rüstzustand) benötigen, werden zu einer **Losfamilie** zusammengefasst. Wenn mehr als eine Losfamilie vorliegt, wird von inkompatiblen Losfamilien gesprochen.

Die Umrüstzeit der Maschinen ist sehr oft abhängig von der Reihenfolge, in der die Lose bearbeitet werden. Jeder Bearbeitungsschritt eines Loses benötigt einen bestimmten Rüstzustand der bearbeitenden Maschine. Die für den Rüstvorgang benötigte Rüstzeit hängt vom aktuellen Rüstzustand der Maschine ab.

Beschreibung des Modells für das Transportsteuerungssystems

Es wird angenommen, dass die Maschinen räumlich in Tunnel angeordnet sind. Zur Zwischenlagerung existieren innerhalb des Transportbasissystems P_p ($p = 1, \ldots, s$) Stocker ($P =$ dt. *Puffer*). Die Kapazität eines Stockers P_p wird mit K_p bezeichnet und gibt an, wie viele Lose gleichzeitig in den Stocker eingestellt werden können. Ist die Kapazität für einen einzelnen Stocker zu klein, kann es zu Staus innerhalb des Transportsystems kommen. In dieser Arbeit wird allerdings davon ausgegangen, dass die Kapazität hinreichend groß ist ($K_p = \infty$), so dass keine Staus auftreten können.

Jede Maschine besitzt dabei einen **Maschinenstocker**, über den die Maschine beladen und entladen werden kann. Weiterhin ist jeder Maschinengruppe ein **Maschinengruppenstocker** zugeordnet, in dem die Lose warten, bevor sie von einer Maschine selektiert werden.

Der Transport der Lose erfolgt durch b Fahrzeuge R_r, mit $r = 1, \ldots, b$ ($R =$ engl. *Robot*). Es wird nur ein Transportbereich betrachtet. Dies bedeutet, dass die Fahrzeuge die Lose innerhalb des gesamten Transportsystems transportieren können.

Der aktuelle Bearbeitungsschritt eines Loses wird mit O_{kj} bezeichnet. Nachdem ein Los J_j mit dem aktuell durchzuführenden Bearbeitungsschritt O_{kj} von einer Maschine selektiert wurde, wird es zunächst vom Maschinengruppenstocker zum Maschinenstocker transportiert. Ist der Transport abgeschlossen, wird die Maschine beladen und gerüstet. Danach erfolgt die eigentliche Bearbeitung. Nachdem das Los bearbeitet wurde, wird die Maschine entladen und das Los zum Maschinengruppenstocker des nächsten Bearbeitungsschritts $O_{k+1,j}$ transportiert. Zur Abbildung dieser Transporte werden für jeden Bearbeitungsschritt O_{ij} zwei Transportschritte T_{ij0} und T_{ij1} eingeführt, die den Transport eines Loses zwischen den **Ausgangsstockern** und **Zielstockern** repräsentieren.

Der Transportschritt T_{ij0} repräsentiert den Transport eines Loses von diesem Maschinengruppenstocker zu der Maschine, auf der der folgende Bearbeitungsschritt O_{ij} durchgeführt wird. Der zweite Transportschritt T_{ij1} repräsentiert den Transport desselben Loses von der Maschine zum Stocker der Maschinengruppe von $O_{i+1,j}$. Um die Transportschritte in den Arbeitsplan eines Loses einzubinden, werden die zusätzlichen Vorrangbeziehungen $T_{ij0} \rightarrow O_{ij} \rightarrow T_{ij1} \rightarrow T_{i+1,j,0} \rightarrow O_{i+1,j}$ eingeführt falls O_{ij} einen Nachfolgebearbeitungsschritt $O_{i+1,j}$ besitzt. Hierbei ist zu beachten, dass die einzelnen Transportschritte jedoch erst bekannt sind, nachdem entschieden wurde, auf welcher Maschine der zugehörige Bearbeitungsschritt durchgeführt werden soll, da andernfalls der Zielstocker für T_{ij0} bzw. der Ausgangstocker für T_{ij1} nicht bekannt ist.

In einer Halbleiterfabrik erfolgt der Transport typischerweise über Schienen. Die Zeit, die ein Fahrzeug benötigt, um von einem Stocker zu einem anderen zu gelangen, hängt von der Position des Ausgangs- und des Zielstockers ab. Jedes Fahrzeug kann nur ein Los gleichzeitig transportieren. Die benötigte Zeit für einen Transportschritt T_{ijt} wird mit $t_{ijt} := t'_{kh} + t_{hl}$ bezeichnet. Sie setzt sich zusammen aus der Zeit für eine eventuell notwendige **Leerfahrt** t'_{kh} vom **Fahrzeugstocker** P_k zum **Ausgangsstocker** des Loses P_h und der **reinen Transportzeit** t_{hl} vom Ausgangsstocker P_h zum **Zielstocker** P_l. Es wird davon ausgegangen, dass für alle Stocker P_p $t'_{pp} := 0$ gilt. Dies bedeutet, dass, wenn sich das Fahrzeug bereits am Ausgangsstocker des Loses befindet, keine Leerfahrt

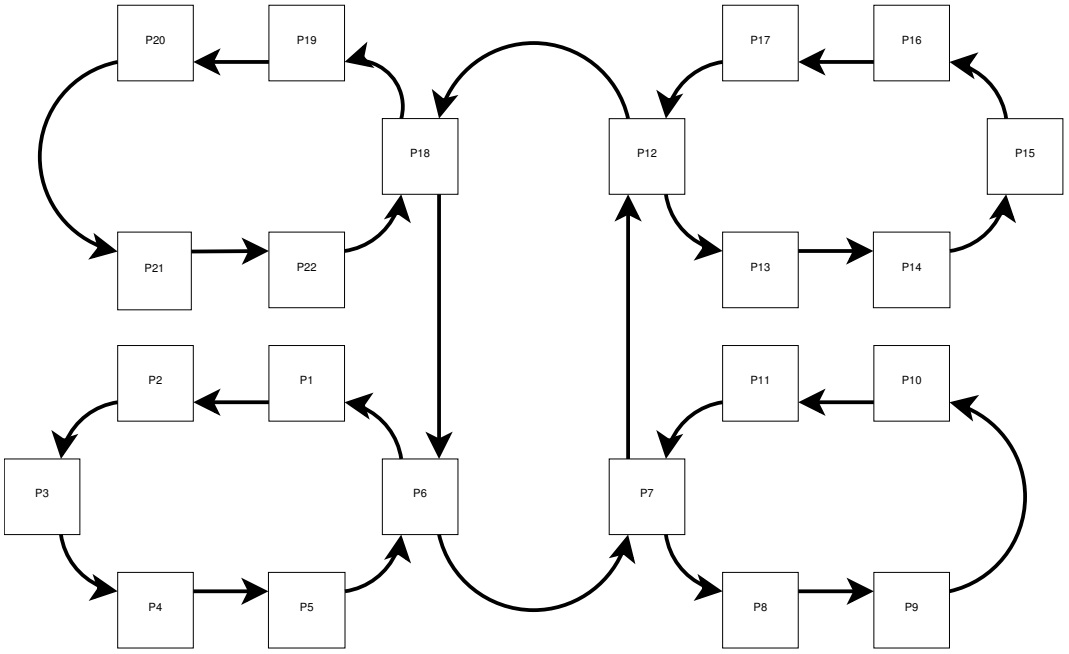

Abbildung 2.7: Layout für vier Maschinengruppen

stattfindet.

Abbildung 2.7 stellt schematisch die räumliche Anordnung der Maschinen- und Maschinengruppenstocker für ein Produktionssystem mit vier Maschinengruppen dar. Jede einzelne Maschinengruppe besitzt einen eigenen Tunnel (Intrabay). Die einzelnen Tunnel sind durch einen weiteren Tunnel (Interbay) miteinander verbunden.

2.4.3 Klassifikation des betrachteten Steuerungsproblems

Zur Klassifikation von Steuerungsproblemen für Produktionssysteme hat sich die $\alpha|\beta|\gamma$-Notation nach Graham u. a. (1979) durchgesetzt. Im α-Feld wird die Maschinenumgebung beschrieben (zum Beispiel Einzelmaschine, Maschinengruppe oder Werkstattfertigung). Das β-Feld beschreibt die Prozessbedingungen (zum Beispiel reihenfolgeabhängige Umrüstzeiten, Berücksichtigung von Batch-Bedingungen usw.). Innerhalb des γ-Feldes erfolgt die Darstellung der Ziele.

Eine Erweiterung dieser Notation für die Transportdomäne erfolgt in Knust (1999). Die Beschreibung der Fahrzeugumgebung wird hierbei in das α-Feld aufgenommen. Das β-Feld wird um die Beschreibung des Transportprozesses erweitert.

Das Steuerungsproblem in einer Halbleiterfabrik ohne Berücksichtigung des Transports kann nach Mönch und Rose (2004) wie folgt klassifiziert werden:

$$FJ_m|r_j, s_{lj}, \text{recrc}, \text{batch}, \text{incompatible}|\,TWT. \tag{2.1}$$

Durch FJ_m ist eine flexible Werkstattfertigung (engl. *Flexible Job-Shop*) beschrieben. Es

müssen unterschiedliche Einsteuerungszeitpunkte (r_j), reihenfolgeabhängige Umrüstzeiten (s_{lj}), schleifenförmige Arbeitspläne (recrc) und simultanes Batching mit inkompatiblen Losfamilien (batch, incompatible) berücksichtigt werden. Im Rahmen dieser Arbeit wird davon ausgegangen, dass das Rüsten nicht antizipativ ist, das heißt, das Rüsten erfolgt erst, wenn sowohl die Lose als auch die Maschine verfügbar sind. Die Zielfunktion der totalen gewichteten Verspätung $TWT := \sum w_j T_j$, mit $T_j := \max(c_j - d_j, 0)$ wird häufig eingesetzt, um eine kundenspezifische Fertigung zu berücksichtigen (TWT = engl. *Total Weighted Tardiness*). Die geplanten Aussteuertermine d_j der Lose werden auf Basis der gewünschten Fertigstellungszeitpunkte der Kunden gesetzt. Durch die Minimierung der TWT wird eine hohe Liefertreue sichergestellt. Aus diesem Grund hat das TWT-Leistungsmaß große Bedeutung in der terminorientierten Fertigung. Durch die Betrachtung des Transports erweitert sich die ursprüngliche Problemklassifikation (2.1) zu:

$$FJ_m, R_r | r_j, s_{lj}, t'_{kh}, t_{hl}, \text{batch}, \text{incompatible}, \text{recr} | \, TWT. \tag{2.2}$$

Durch R_r werden zusätzlich mehrere Fahrzeuge innerhalb des Transportsystems bezeichnet. Die Zeit t_{hl}, die ein Fahrzeug benötigt, um ein Los von einem Stocker zu einem anderen zu transportieren, hängt von der Position des Ausgangs- und des Zielstockers ab. Falls das Fahrzeug sich nicht an derselben Position wie das zu transportierende Los befindet, müssen zusätzlich Zeiten für Leerfahrten (t'_{kh}) berücksichtigt werden.

3 Steuerung von komplexen Produktionssystemen mit automatischem Transport

Dieses Kapitel beschäftigt sich mit der Steuerung von komplexen Produktionssystemen mit automatischem Transport. Zunächst wird auf die in der Praxis in Halbleiterfabriken verwendeten Steuerungssysteme und deren Defizite eingegangen. Im Anschluss daran erfolgt die Vorstellung unterschiedlicher Verfahrensansätze zur Lösung von Steuerungs- und Ablaufplanungsproblemen. Danach erfolgt ein Überblick über aktuelle Forschungsarbeiten bzgl. des zu behandelnden Steuerungsproblems (2.2). Zum Ende des Kapitels werden Anforderungen für ein entsprechendes integriertes Steuerungsverfahren entwickelt.

3.1 Steuerungssysteme in Halbleiterfabriken

Die Steuerung des Bearbeitungsbasissystems einerseits und des Transportbasissystems andererseits erfolgt in Halbleiterfabriken typischerweise getrennt. Das Bearbeitungssteuerungssystem wird dabei durch ein **Manufacturing Excecution System** (MES) umgesetzt. Das Transportsteuerungssystems wird durch ein **Material Control System** (MCS) umgesetzt (vgl. Foster und Pillai (2008)). In der deutschsprachigen Literatur wird anstelle des Begriffs MES auch häufig der Begriff Fertigungsmanagementsystem verwendet (vgl. VDI (2007)).

Die Eingangsgrößen des MES werden hierbei über eine **Betriebsdatenerfassung** (BDE), eine **Maschinendatenerfassung** (MDE) sowie eine **Personaldatenerfassung** ermittelt. Hierzu gehören unter anderem die Erfassung der Start- und Fertigstellungszeitpunkte der Bearbeitungsschritte der Lose, die Lose, die in den einzelnen Stockern lagern, und die aktuellen Rüstzustände der Maschinen. Weiterhin enthält das MES alle weiteren Informationen, die für die Steuerung des Bearbeitungsbasissystems notwendig sind, wie Arbeitspläne, mögliche Rüstzustände der Maschinen, die Kapazitäten der Stocker, die Zuordnung der Maschinen zu Maschinengruppen und Stockern sowie (zukünftige) geplante Einsteuertermine und geplante Aussteuertermine der Lose. Ausgangsgrößen des MES sind Bearbeitungslisten zur Steuerung der Maschinen. In einer Bearbeitungsliste steht, welcher Batch als nächstes auf einer Maschine bearbeitet werden kann.

Im Unterschied zu darüberliegenden Systemen wie **Enterprise Resource Planning** (ERP-System) zeichnet sich ein MES durch direkte Anbindung an das Bearbeitungsbasissystem aus. Hierdurch ist eine größere Automatisierung des Bearbeitungsprozesses und eine zeitnahe Steuerung möglich.

Zur Steuerung des gesamten Produktionssystems ist es notwendig, dass das MES und

das MCS gekoppelt werden. Das MES ist hierbei das führende System und übergibt dem Transportsystem nach erfolgter Maschinenbelegungsentscheidung einen Transportwunsch, bestehend aus Los und Zielstocker. Hierzu ist es notwendig, dass auf Seiten des MES die Zuordnung von Stockern zu Maschinen bzw. Maschinengruppen bekannt ist. Weitere Eingangsgrößen des MCS werden durch die **Fahrzeug- und Stockercontroller** bereitgestellt. Ausgangsgrößen sind Anweisungen an die Fahrzeuge und Stocker, welches Los transportiert sowie ausgelagert bzw. eingelagert werden soll. Nachdem der Transport durchgeführt wurde, erfolgt eine Benachrichtigung des MES durch das MCS.

Wie bereits in Abschnitt 2.4 erwähnt, ist ein MCS oft hierarchisch organisiert. Es existiert häufig für jeden Tunnel des Bearbeitungssystems ein eigenes automatisches Transportsystem mit einem eigenen Basis- und Steuerungssystem. Da eine autonome Energieversorgung der Fahrzeuge aufgrund ihrer Größe technisch nicht möglich ist, erfolgt die Energieversorgung über das Schienensystem. Aus diesem Grund ist das gesamte Schienennetz in einzelne Zonen für die Energieversorgung eingeteilt. Pro Zone kann nur eine bestimmte Anzahl von Fahrzeugen mit Energie versorgt werden. Deswegen erfolgt oft eine weitere Zerlegung des Bereichssteuerungssystems in mehrere Zonensteuerungssysteme, welche die Steuerung der Fahrzeuge innerhalb dieser Zonen übernehmen. Das Bereichssteuerungssystem stellt dabei die Koordination zwischen den einzelnen Zonensteuerungssystemen sicher. Neben der Verwaltung der Energieversorgung erfolgt die Vermeidung von Kollisionen und die Auflösung von Stausituationen innerhalb des Transportsystems auf diesen beiden Ebenen. Auf Ebene des MCS erfolgt die Auswahl der zu transportierenden Lose sowie die Berechnung des zu fahrenden Weges für die Fahrzeuge.

Die Steuerung der Fahrzeuge durch das MCS erfolgt auf Basis der bekannten Positionen der FOUPs und der Fahrzeuge. Bei einer Transportanforderung durch das MES wird dem MCS das Los, der Zielstocker, ein Maschinenstocker bzw. ein Maschinegruppenstocker, und eventuell weitere Informationen (z. B. eine Lospriorität), die für den Transport relevant sind, mitgegeben. Das MCS veranlasst nun, dass das Los an dem Ort, an dem es sich befindet, in einen FOUP umgeladen wird. Danach wird ein Fahrzeug angewiesen, den FOUP aufzunehmen und zum Zielstocker zu transportieren. Hierbei kann es notwendig sein, dass der FOUP über mehrere zwischengelagerte Maschinengruppenstocker transportiert werden muss. Nachdem das Los aus dem FOUP entnommen wurde, wird die zugehörige Maschine beladen bzw. das Los in den Stocker eingelagert, bevor es wieder weitertransportiert werden muss.

Wie bereits beschrieben, sind MES und MCS getrennte Systeme, welche dadurch gekoppelt sind, dass das MCS durch das MES Vorgabem erhält, welche Lose transportiert werden sollen. Aus diesem Grund ist ein geringer und wenig streuende **Durchlaufzeit** $c_j - r_j$ das wichtigste in der Praxis angewendete Leistungsmaße innerhalb des MCS (vgl. auch Foster und Pillai (2008)). Dies führt jedoch bei einer terminorientierten Fertigung nicht immer zu zufriedstellenden Ergebnissen. Dies liegt einerseits daran, dass innerhalb des MES die Restriktionen des MCS nicht berücksichtigen werden. Andererseits hat auch das MCS nur ungenügende Informationen über die Restriktionen des MES und die globalen Ziele des Produktionssystems. Ein stärker integriertes Steuerungssystem, das in der Lage ist die globalen Ziele des Produktionssystems direkter zu berücksichtigen, erscheint in

diesem Zusammenhang wünschenswert.

3.2 Verfahren zur Steuerung komplexer Produktionssysteme mit automatischem Transport

Im Folgenden werden Verfahren zur Steuerung vorgestellt. Die Ausführungen in diesem Abschnitt sind an Mönch (2006) angelehnt. Ein Steuerungsverfahren ist der Teil des Steuerungssystems, der die Entscheidung trifft, welche Lose in welcher Reihenfolge auf einer bestimmten Maschine bearbeitet werden sollen. Eng verbunden mit der Wahl des Steuerungsverfahrens ist die Wahl der Einsteuerstrategie für die Lose. Aus diesem Grund erfolgt zunächst ein Überblick über Einsteuerungsstrategien für komplexe Produktionssysteme. Danach wird auf die Steuerung mittels Prioritätsregeln und deterministischem Scheduling eingegangen.

3.2.1 Einsteuerstrategien für komplexe Produktionssysteme

Innerhalb eines Produktionssystems gibt es einen spezifischen Zusammenhang zwischen der Anzahl der Lose und dem Durchsatz und der durchschnittlichen Durchlaufzeit der Lose, der durch eine charakteristische Kennlinie beschrieben wird (vgl. auch Nyhuis und Wiendahl (1999)).

Unter einer **Einsteuerstrategie** werden in diesem Zusammenhang die Regeln verstanden, mit denen die Lose in das Produktionssystem eingelastet, das heißt, wann und wie die einzelnen Kundenaufträge durch Lose untersetzt werden. Die Lose werden nicht sofort eingelastet, nachdem der Kundenauftrag bekannt ist, sondern erst später. Dadurch wird versucht, die Einlastung so zu steuern, dass die Anzahl der Lose innerhalb des Produktionssystems stabil bleibt und der Mittelwert sowie die Varianz der Durchlaufzeiten minimiert wird. Durch eine spätere Einlastung der Lose verschlechtert sich aber unter Umständen die Termintreue.

Einsteuerstrategien in der Werkstattfertigung werden in Bergamaschi u. a. (1997) behandelt. Eine Übersicht über Einsteuerstrategien innerhalb der Halbleiterindustrie gibt Fowler u. a. (2002). Grundlegend kann bei der Einsteuerung von Losen zwischen **Push-** und **Pull-Konzepten** unterschieden werden (vgl. Mönch (2006)).

Push-Ansätze steuern die Lose entsprechend der Vorgaben aus dem ERP-System ein. Da innerhalb dieser Systeme die endlichen Kapazitäten des Produktionssystems nach wie vor nur unzureichend berücksichtigt werden (vgl. Drexl u. a. (1994)), führt das zu hohen Beständen und langen Durchlaufzeiten in den Produktionssystemen. Push-Methoden sind dadurch gekennzeichnet, dass sie festlegen, wie viel produziert werden soll.

Pull-Methoden demgegenüber gehen davon aus, wie viel produziert werden kann. Sie wurden stark durch die Einführung von Just-in-time- sowie engpassorientierten Steuerungskonzepten wie der Optimized Production Technology (OPT) (vgl. Goldratt und Cox (1992) und Jacobs (1984)) beeinflusst. Dadurch werden die kapazitiven Möglichkeiten des Produktionssystems besser berücksichtigt.

Pull-Ansätze sind Push-Ansätzen oft überlegen (vgl. Roderick u. a. (1992) und Ragatz

und Mabert (1988)), werden jedoch in der Praxis häufig nicht eingesetzt (Fowler u. a. (2002)), da sie als zu kompliziert angesehen werden.

3.2.2 Prioritätsregelbasierte Steuerung komplexer Produktionssysteme

Prioritätsregeln werden dazu verwendet, um zu einem bestimmten Zeitpunkt zu entscheiden, welche Aktion aus einer Menge von zulässigen Aktionen als nächstes ausgeführt werden soll. Hierbei können jeweils zwei Sichtweisen eingenommen werden. Aus Ressourcensicht wird das Lose gewählt, welches als nächstes bearbeitet bzw. transportiert werden soll, wenn eine **Ressource** frei wird. Hierzu wird für jedes Los ein **Prioritätswert** auf Basis einer Prioritätsregel berechnet. Mit diesem Prioritätswert wird dann entschieden, welche Lose als nächstes bearbeitet werden. Unter einer Ressource wird im Rahmen dieser Arbeit entweder eine Maschine des Bearbeitungssystems oder ein Fahrzeug des Transportsystems verstanden. Aus Lossicht wird entschieden, welche Ressource als nächstes belegt werden soll, wenn das Los den nächsten Bearbeitungs- bzw. Transportschritt erreicht. Hier wird der Prioritätswert für die einzelnen Ressourcen berechnet. Innerhalb des Bearbeitungssystems wird normalerweise nur die Ressourcensicht eingenommen, da eine Vielzahl von zu bearbeitenden Losen nur einer kleinen Zahl von Maschinen gegenübersteht. Innerhalb des Transportsystems sind beide Sichten üblich.

In der Literatur werden Prioritätsregeln ausführlich in Blackstone u. a. (1982) und Haupt (1989) behandelt. Spezielle Prioritätsregeln für Halbleiterfabriken werden in den Arbeiten Rose (2001), Rose (2002) und Rose (2003) für das Bearbeitungssystem untersucht. Eine Diskussion von Prioritätsregeln für das Transportsystem in Halbleiterfabriken erfolgt in den Arbeiten Lin u. a. (2001), Liao und Wang (2006) sowie Liao und Fu (2004).

Varadarajan und Sarin (2006) geben einen umfassenden Überblick über Prioritätsregeln für das Bearbeitungs- und das Transportsystem innerhalb von Halbleiterfabriken. Überlegungen zu integrierten Prioritätsregeln für das Bearbeitungs- und Transportsystem finden sich in den Arbeiten Tyan u. a. (2004) und Drießel und Mönch (2007).

Einfache Prioritätsregeln für das Bearbeitungssystem sind zum Beispiel die FIFO-Regel (FIFO = First-In, First-Out), die gewichtete kürzeste Bearbeitungszeitregel und die Schlupfzeitregel. Innerhalb des Transportsystems sind die NLF-Regel (NLF = Nearest Lot First) und die kürzeste Transportzeitregel Beispiele für einfache Prioritätsregeln. Nach Mönch (2006) lassen sich neben den einfachen Regeln

- **zusammengesetzte Prioritätsregeln**,
- **bedingte Prioritätsregeln** und
- **Multi-Level-Prioritätsregeln**

unterscheiden. Zusammengesetzte Prioritätsregeln fassen mehrere Regeln zu einer einzigen zusammen. Ein Beispiel für diesen Ansatz ist die ATC-Regel (ATC = Apparent Tardiness Cost) (Vepsalainen und Morton (1987)), welche die gewichtete kürzeste Bearbeitungszeit und den Schlupf eines Loses miteinander kombiniert.

Bedingte Prioritätsregeln verwenden, abhängig vom Zustand des Produktionssystems, unterschiedliche Prioritätsregeln. So kann zum Beispiel in Abhängigkeit von der Auslastung des Produktionssystems zwischen der gewichteten kürzesten Bearbeitungszeitregel und der Schlupfzeitregel gewechselt werden.

Bei Multi-Level-Prioritätsregeln werden einzelne Prioritätsregeln nacheinander angewendet. So ist es möglich, zunächst alle Lose zu ermitteln, die geringe Rüstkosten verursachen. In einem zweiten Schritt wird dann das Los mit der geringsten Schlupfzeit gewählt. Multi-Level-Regeln für den Einsatz in einer Halbleiterfabrik sind in Thiel u. a. (1998) untersucht worden.

Aufgrund der Komplexität der betrachteten Probleme sind Prioritätsregeln in der Praxis der am weitesten verbreitete Steuerungsansatz. Dies liegt daran, dass sie sich relativ leicht implementieren lassen. Außerdem zeichnen sie sich durch eine leichte Anpassbarkeit an die aktuelle Situation des Produktionssystems aus. Hauptvorteil gegenüber anderen Verfahren ist jedoch die geringe Rechenzeit für die Entscheidungsfindung. Weiterhin ist ihre Leistungsbewertung mit Hilfe von Simulationsmodellen relativ leicht möglich (vgl. Mönch (2006)).

Der wesentliche Nachteil von Prioritätsregeln gegenüber anderen Verfahren besteht darin, dass bei der Entscheidungsfindung nur zeitlich und räumlich lokale Informationen einbezogen werden. Da die einzelnen Lose isoliert voneinander betrachtet werden, ist es nur schwer möglich, mehrere Ziele zu berücksichtigen. Weiterhin ist eine automatische Anpassung an veränderte Bedingungen des Bearbeitungs- und Transportsystems nur mit hohem Aufwand zu realisieren.

3.2.3 Deterministisches Scheduling

Scheduling bzw. **Ablaufplanung** beschäftigt sich mit der Zuordnung von Aktivitäten zu Ressourcen und umgekehrt von Ressourcen zu Aktivitäten unter Beachtung vorgegebener Restriktionen und Zielsetzungen (vgl. Brucker (2004) sowie Pinedo (2001)). Bei der Zuordnung muss gleichzeitig auch eine Reihenfolgeentscheidung über die Abarbeitungsreihenfolge der einzelnen Aktivitäten getroffen werden. Die Gesamtheit aller Abarbeitungsreihenfolgen für alle Ressourcen wird **Ablaufplan** genannt. Im Rahmen des Schedulings von Produktionssystemen umfasst der Begriff Aktivität alle Tätigkeiten, die für die Durchführung eines Prozesschrittes mit einem Los ausgeführt werden müssen, wie zum Beispiel Rüsten, Bearbeiten, Transportieren, Laden und Entladen. Es wird zwischen **deterministischen** und **stochastischen Scheduling-Problemen** unterschieden. Deterministische Scheduling-Probleme sind dadurch gekennzeichnet, dass alle Daten diskret, deterministisch und vorab bekannt sind. Innerhalb von stochastischen Scheduling-Problemen sind einige oder alle Daten in Form von Wahrscheinlichkeitsverteilungen gegeben. Im weiteren Verlauf werden deterministische Scheduling-Probleme betrachtet.

Im Unterschied zur prioritätsregelbasierten Steuerung erfolgt hier die Steuerungsentscheidung nicht zu jedem Zeitpunkt, wenn eine Ressource frei wird, sondern in größeren Zeitabständen, zum Beispiel einmal pro Schicht. Zunächst wird auf Basis des aktuellen Zustands des Basissystems eine Scheduling-Problem formuliert, das durch ein Scheduling-

Verfahren gelöst wird. Die Lösung des Scheduling-Problems ist ein Ablaufplan, mit dem die Ressourcen des Basissystems gesteuert werden können. Bei der Erzeugung des Ablaufplans wird typischerweise versucht, bestimmte Zielgrößen des Produktionssystems zu verbessern. Nachdem der Ablaufplan erzeugt wurde, wird das Produktionssystem auf dessen Basis gesteuert. Störungen wie zum Beispiel Maschinenausfälle oder Verzögerungen bei der Bearbeitung müssen bei der Umsetzung separat behandelt werden.

Die meisten Scheduling-Probleme sind **kombinatorische Optimierungsprobleme** (siehe auch Brucker und Knust (2006)). Ein kombinatorisches Optimierungsproblem lässt sich allgemein wie folgt beschreiben:

$$\min\{c(s) \mid s \in S\}. \tag{3.1}$$

Hierbei ist $s \in S$ eine Lösung innerhalb des **Lösungsraums** S und $c : S \to \mathbb{R}$ die Bewertungsfunktion. Der Lösungsraum ist endlich und enthält alle **zulässigen Lösungen**. Eine Lösung ist zulässig, wenn alle Nebenbedingungen erfüllt sind.

Im Zusammenhang mit der Steuerung eines Basissystems muss zwischen statischem und dynamischem Scheduling unterschieden werden. Meistens werden in der Literatur nur statische Probleme betrachtet. Beim statischen Scheduling wird nur ein Scheduling-Problem betrachtet. Es erfolgt im Unterschied zum dynamischen Scheduling keine Steuerung des Basissystems. Statisches Scheduling dient dazu, die Leistungsfähigkeit von verschiedenen Verfahren miteinander zu vergleichen.

Von einem statischen Experiment lässt sich jedoch nicht unbedingt auf die Leistungsfähigkeit im dynamischen Umfeld, das heißt bei der Steuerung des Produktionssystems, schließen. Zur Steuerung eines Produktionssystems müssen viele einzelne Scheduling-Probleme in einem rollierenden Ansatz nacheinander gelöst werden, um den jeweils aktuellen Zustand des Basissystems berücksichtigen zu können. Aus diesem Grund müssen unterschiedliche Ankunftszeiten der Lose berücksichtigt werden. Weiterhin kann es aufgrund der Gegenläufigkeit der Zielgrößen zu unerwünschten Effekten kommen. Bei einer reinen Verbesserung der totalen gewichteten Verspätung werden nicht verspätete Lose verlangsamt und verspätete Lose beschleunigt. Dies kann zu einem etwas niedrigeren Durchsatz führen. Innerhalb eines einzelnen Problems ist dies nicht problematisch. Tritt dies jedoch bei der Steuerung des Produktionssystems häufiger auf, so erhöht sich bei gleichbleibender Einsteuerung die Anzahl der Lose innerhalb des Produktionssystems. Dies wiederum führt langfristig zu einer Erhöhung der totalen gewichteten Verspätung, obwohl dieser Effekt bei den einzelnen Scheduling-Problemen kaum messbar ist.

In neueren Arbeiten wie Mönch und Rose (2004), Mönch u. a. (2007) sowie Sourirajan und Uzsoy (2007) wird aus diesem Grund häufig die diskrete Simulation zur Bewertung von Ablaufplanungsverfahren für komplexe Werkstattfertigungssysteme herangezogen. Hierbei ersetzt ein Simulationsmodell das reale Produktionssystem.

Gegenüber der Steuerung mit Prioriätsregeln hat Scheduling den Vorteil, dass abhängig vom Verfahren nicht nur lokale Informationen berücksichtigt werden, sondern auch globale. Allerdings benötigen Schedulingansätze im Allgemeinen mehr Rechenzeit und sind auch sonst in der Anwendung komplizierter zu handhaben. Im weiteren Verlauf

werden verschiedene exakte und heuristische Verfahren zur Lösung von Scheduling-Problemen in komplexen Produktionssystemen mit automatischem Transport behandelt.

3.3 Exakte Verfahren zur Lösung von Scheduling-Problemen

Exakte Verfahren ermitteln die optimale Lösung eines Scheduling-Problems in endlich vielen Schritten. Viele Scheduling-Probleme lassen sich als **gemischt-ganzzahlige lineare Optimierungsprobleme** formulieren (vgl. Brucker und Knust (2006)). Aus diesem Grund wird zunächst auf gemischt-ganzzahlige lineare Optimierungsprobleme eingegangen. Im Anschluss daran werden zwei allgemeine Verfahren zur exakten Lösung von Optimierungsproblemen vorgestellt.

3.3.1 Modellierung als gemischt-ganzzahliges lineare Optimierungsproblem

Die **lineare Optimierung** beschäftigt sich mit der Lösung von Optimierungsproblemen mit linearer Zielfunktion unter Berücksichtigung von Nebenbedingungen, die als lineares Ungleichungssystem modelliert werden können. Während die Variablen bei der linearen Optimierung reelle Werte annehmen können, dürfen bei der **ganzzahligen linearen Optimierung** die Variablen nur ganzzahlig sein. Im Falle, dass die Ganzzahligkeitsbedingung nur für einige Variablen gilt, spricht man von gemischt-ganzzahliger linearer Optimierung.

Ein gemischt-ganzzahliges lineares Optimierungsproblem kann wie folgt formuliert werden:

Minimiere

$$\sum_{j=1}^{n} c_j x_j \tag{3.2}$$

unter den Nebenbedingungen

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m) \tag{3.3}$$

$$x_j \in D_j \qquad (j = 1, \ldots, n) \tag{3.4}$$

Der Vektor $x := (x_1, \ldots, x_n)$ enthält die Entscheidungsvariablen. Die Parameter c_j, a_{ij} und b_i sind gegebene reelle Zahlen. Eine zulässige Lösung des Optimierungsproblems ist eine Belegung des Vektors x, die die **Nebenbedingungen** in (3.3) erfüllt sowie innerhalb des geforderten Definitionsbereichs D_j aus (3.4) liegt. Das Ziel ist das Finden eines Vektors x, bei dem die **Zielfunktion** (3.2) minimiert wird. Jedes **Maximierungsproblem** lässt

sich dabei in ein **Minimierungsproblem** umwandeln, indem die Zielfunktion (3.2) mit -1 multipliziert wird.

Für die lineare Optimierung ohne Ganzzahligkeitsbedingung existieren mit dem Simplexverfahren (vgl. Dantzig (1963)) sowie verschiedenen Innere-Punkte-Verfahren exakte Algorithmen, die in den meisten Fällen ein Problem mit polynomiellem Zeitaufwand in Abhängigkeit von der Problemgröße lösen können. Für gemischt-ganzzahlige Optimierungsprobleme ist bisher kein derartiges exaktes Verfahren bekannt. Hierbei werden durch die ganzahligen Entscheidungsvariablen die Reihenfolge- und Zuordnungsentscheidungen des Scheduling-Problems abgebildet.

Die Nebenbedingungen in dem oben vorgestellten linearen Ungleichungssystem sind alle konjunktiv verknüpft, das heißt, alle Nebenbedingungen müssen gleichzeitig erfüllt sein. Eine andere Möglichkeit zur Abbildung der Reihenfolge und Zuordnungsentscheidungen in Scheduling-Problemen ist durch die **disjunktive Programmierung** gegeben (vgl. Pinedo (2001)). Bei der disjunktiven Programmierung können die Nebenbedingungen disjunktiv verknüpft sein, das heißt, es kann nur eine Bedingung erfüllt sein. Auf beide Modellierungsarten lassen sich dieselben Techniken zur Lösung anwenden. Im weiteren Verlauf wird eine Auswahl von exakten Lösungsverfahren für Scheduling-Probleme vorgestellt.

3.3.2 Branch-and-Bound-Verfahren

Ein in der Literatur häufig verwendeter Ansatz zum exakten Lösen von Scheduling-Problemen ist das **Branch-and-Bound-Verfahren** . Dieses wurde erstmals von Land und Doig (1960) formuliert.

Grundlegende Idee ist die strukturierte Durchsuchung des Lösungsraums anhand eines Suchbaums. Dazu wird das Problem (3.1) in eine Menge von Teilproblemen aufgeteilt. Ein Teilproblem betrachtet dabei eine Teilmenge $S_i \subseteq S$. Diese Aufteilung wird rekursiv fortgesetzt, bis alle Lösungen betrachtet wurden. Gemischt-ganzzahlige lineare Problemformulierungen werden häufig mittels des Branch-and-Bound-Verfahrens gelöst. Hierbei werden die Teilprobleme innerhalb des Suchbaums dadurch erzeugt, dass einer ganzahligen Entscheidungsvariablen ein konkreter Wert zugewiesen wird. Dies wird innerhalb des Suchbaums rekursiv so lange fortgesetzt, bis jede ganzahlige Entscheidungsvariable mit einem konkreten Wert belegt ist. Das verbleibende lineare Optimierungsproblem kann dann mit einem Simplexverfahren gelöst werden. Dies wird so lange wiederholt, bis alle Variable-Wert-Kombinationen betrachtet wurden.

Die Durchsuchung des Lösungsraums kann abgekürzt werden, wenn für bestimmte Äste des Baums klar ist, dass eine weitere Untersuchung keine Verbesserung erzielt. Hierzu ist ein Algorithmus notwendig, der eine **untere Schranke** für den Zielfunktionswert auf Basis des aktuellen Teilproblems liefern kann. Eine **obere Schranke** für den Zielfunktionswert ist durch die bisher beste gefundene Lösung gegeben. Im Algorithmus 3.1 ist ein generisches Branch-and-Bound-Verfahren dargestellt. Die Differenz zwischen oberer und unterer Schranke wird als **Optimalitätslücke** bezeichnet. Sie gibt die maximale Entfernung von einem optimalen Zielfunktionswert an. Im Fall einer gemischt-ganzzahligen linearen Problemformulierung kann die untere Schranke dadurch ermittelt werden, dass

die Ganzzahligkeitsbedingung aufgehoben wird (Relaxierung) und das entstehende lineare Optimierungsproblem gelöst wird. Ist das relaxierte Problem nicht lösbar, kann die Suche in diesem Teilbaum abgebrochen werden.

In der Anwendung des Branch-and-Bound-Verfahrens ergeben sich jedoch im Wesentlichen zwei Schwierigkeiten (vgl. Brucker und Knust (2006)):

1. Es ist bei vielen Optimierungsproblemen schwierig, einen Algorithmus zu finden, der eine gute untere Schranke liefert.

2. Die Reihenfolge, in der der Suchbaum aufgebaut wird, ist entscheidend für die Laufzeit des Verfahrens.

Diese zwei Schwierigkeiten sorgen meistens dafür, dass zu viele Lösungen evaluiert werden müssen und die Laufzeit des Verfahrens dadurch nicht akzeptabel ist.

Algorithmus 3.1 : Generisches Branch-and-Bound-Verfahren (vgl. Brucker und Knust (2006))

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $s \in S$ mit der besten Bewertung $c(s)$

```
1  s ← Erzeuge Startlösung aus S;
2  UB ← c(s);
3  Branch & Bound(S);
4  Beginn Branch & Bound(S)
5      Partitioniere S in die Teilmengen S = ⋃_{i=1}^{k} S_i;
6      für alle S_i ⊆ S führe aus
7          Berechne untere Schranke LB_i für S_i;
8          wenn LB_i < UB dann
9              wenn S_i enthält nur eine Lösung s_i dann
10                 wenn c(s_i) < UB dann
11                     s ← s_i;
12                     UB ← c(s_i);
13                 Ende
14             sonst
15                 Branch & Bound(S_i);
16             Ende
17         Ende
18     Ende
19 Ende
```

3.3.3 Dynamische Programmierung

Die **dynamische Programmierung** (Bellman (1957)) ist ein weiterer Ansatz, um Optimierungsprobleme exakt zu lösen. Sie kann angewendet werden, wenn die optimale

Lösung des Problems sich aus den optimalen Lösungen der Teilprobleme zusammensetzen lässt. Die dynamische Programmierung gliedert sich prinzipiell in die drei Teile:

- Initialbedingung,
- rekursive Funktion und
- Zielfunktion.

Innerhalb der Rekursion löst der Algorithmus eine Reihe von Teilprobleme, bis die Lösung für das Originalproblem gefunden ist. Einmal berechnete Teilergebnisse werden in einer Tabelle gespeichert, um bei nachfolgenden Berechnungen auf diese Zwischenlösungen zurückzugreifen. Der Algorithmus ermittelt die optimale Lösung für jedes Teilproblem und den Beitrag des Teilproblems zur global optimalen Lösung. Bei jeder Iteration wird ein Teilproblem gelöst, das größer ist als alle vorher gelösten Teilprobleme. Die Lösung für das aktuelle Teilproblem wird dabei mit Hilfe der Lösungen der vorher gelösten Teilprobleme gefunden.

3.4 Heuristische Verfahren zur Lösung von Scheduling-Problemen

Es ist bekannt, dass Optimierungsprobleme bezüglich der Laufzeit der Lösungsverfahren in Abhängigkeit von der Problemgröße in unterschiedliche **Komplexitätsklassen** eingeteilt werden können (vgl. Brucker und Knust (2006), Pinedo (2001) und Domschke u. a. (1997)). In der Literatur gilt ein Problem als effizient und häufig auch praktisch optimal lösbar, wenn es einen Algorithmus gibt, der die Lösung mit polynomialem Zeitaufwand in Abhängigkeit von der Problemgröße erzeugen kann.

Die Komplexitätsklasse P umfasst dabei alle Probleme, bei denen es ein optimales Lösungsverfahren mit polynomieller Laufzeit auf einer deterministischen Turingmaschine gibt. Die Komplexitätsklasse NP enthält neben den Problemen aus P alle Probleme, für die ein Algorithmus mit polynomieller Laufzeit auf einer nichtdeterministischen Turingmaschine existiert (NP steht für nichtdeterministisch polynomial). NP-schwere Probleme wiederum sind Probleme, die nicht notwendigerweise in NP liegen müssen, allerdings mit Hilfe eines polynomiellen Algorithmus auf ein Problem aus der Komplexitätsklasse NP reduziert werden können.

Alle bekannten deterministischen Verfahren für Probleme die aus NP bzw. NP-schwer sind benötigen einen exponentiellen Aufwand. Da die zur Zeit existierenden Computer deterministischer Natur sind, lassen sich diese Probleme vermutlich nicht effizient, das heißt in Polynomialzeit, lösen.

Ein Beispiel für Probleme, die mit polynomiellem Zeitaufwand lösbar sind, sind Scheduling-Probleme des Typs $1||\sum w_j c_j$ (vgl. auch Lawler (1978)). Hier liefert eine absteigende Sortierung der Bearbeitungsschritte nach der gewichteten kürzeste Bearbeitungszeit $\frac{w_j}{p_{ij}}$ die optimale Lösung. Ein Scheduling-Problem für eine Maschine mit der Zielfunktion der totalen gewichteten Verspätung $1||TWT$ ist jedoch bereits NP-schwer

(vgl. Lenstra u. a. (1977) und Lawler (1977)). Auch im Bereich der Transportplanung sind die meisten Probleme *NP*-schwer (vgl. Lenstra und Kan (1981)). Das betrachtete Problem 2.2 ist ebenfalls *NP*-schwer, da es das Optimierungsproblem für eine Maschine mit totaler gewichteter Verspätung beinhaltet. Einen Überblick über verschiedene Scheduling-Probleme und ihrer Klassifikation bezüglich der Komplexität liefert Brucker und Knust (2011).

Für *NP*-schwere Probleme ist es allgemein sehr zeitaufwendig, optimale Lösungen zu ermitteln. Der weiter oben vorgestellte Branch-and-Bound-Ansatz kann häufig nur bei kleinen Problemen die optimale Lösung ermitteln. Für die Behandlung von größeren Problemen müssen **Heuristiken** angewendet werden. Eine Heuristik liefert allerdings nur eine zulässige Lösung ohne Garantie der Optimalität. Beim Vorliegen einer unteren Schranke ist es jedoch möglich, den Abstand zur optimalen Lösung abzuschätzen. Im Folgenden werden verschiedene heuristische Ansätze zur Lösung von Scheduling-Problemen vorgestellt.

3.4.1 Verfahren auf Basis von Prioritätsregeln

Auf Basis der weiter oben vorgestellten Prioritätsregeln lassen sich auch Lösungen für Scheduling-Problem ermitteln. Wie bereits beschrieben, werden Prioritätsregeln dazu verwendet, um zu entscheiden, welches Los als nächstes bearbeitet bzw. transportiert werden soll, wenn eine Ressource frei wird. Bei der Erzeugung einer Lösung für ein Scheduling-Problem wird dabei der zeitliche Ablauf innerhalb des Basissystem simuliert.

Im Algorithmus 3.2 ist das prinzipielle Vorgehen bei der Erzeugung einer Lösung dargestellt. Basis des Verfahren bilden dabei verschiedene Listen. Die Ressourcen (Maschinen bzw. Fahrzeuge) werden in einer Ressourcenliste R und die Prozessschritte (Bearbeitungs- bzw. Transportschritte) in einer Prozessschrittliste P verwaltet. Der Ablaufplan wird dabei wiederum durch Abarbeitungslisten $S(r)$ für jede einzelne Ressource abgebildet. Eine Abarbeitungsliste legt dabei die Reihenfolge fest, in der die Prozessschritte später abgearbeitet werden sollen. Aus diesem Grund wird das Verfahren auch als listenbasiertes Scheduling (engl. *list based scheduling*) bezeichnet.

Algorithmus 3.2 : Erzeugung einer Lösung auf Basis einer Prioritätsregel

Daten : Scheduling-Problem p
Ergebnis : Lösung s für Problem p

```
1 solange P ≠ ∅ führe aus
2     r ← Ermittle die nächste verfügbare Ressource aus der Ressourcenliste;
3     Ermittle Prioritätswerte für alle verfügbaren Prozessschritte;
4     Wähle auf dieser Basis die nächsten durchzuführenden Prozessschritte;
5     Füge die gewählten Prozessschritte dem Ablaufplan S(r) der Ressource r hinzu;
6     Entferne die Prozessschritte aus der Prozessschrittliste P;
7 Ende
```

Der Vorteil von Verfahren auf Basis von Prioritätsregeln gegenüber anderen Verfahren besteht darin, dass sie in der Lage sind, eine zulässige Lösung in kurzer Zeit zu liefern.

Für bestimmte Problemklassifikationen liefern Prioritätsregeln sogar optimale Lösungen. So erzeugt die kürzeste Bearbeitungszeitregel $\frac{1}{p_{ij}}$ für Probleme des Typs $1||\sum c_j$ die optimale Lösungen. Im Allgemeinen unterscheidet sich die Lösungsqualität allerdings sehr stark, da Prioritätsregeln nur zeitlich und räumlich lokale Informationen berücksichtigen. Um diesen Nachteil auszugleichen, ist es üblich, mehrere Lösungen mit verschiedenen Prioritätsregeln zu erzeugen. Die Lösung mit dem besten Zielfunktionswert wird am Ende gewählt.

3.4.2 Genetische Algorithmen

Das Konzept der **Genetischen Algorithmen** (GA) wurde zuerst in Holland (1975) vorgestellt. Es handelt sich um eine generische Suchtechnik, die die natürliche Evolution imitiert. Ein **Chromosom** ist innerhalb eines Genetischen Algorithmus die Kodierung einer Lösung als Folge von Symbolen. Während einer Iteration eines Genetischen Algorithmus werden immer mehrere Lösungen betrachtet. Alle pro Iteration verwendeten Chromosomen werden als **Population** bezeichnet. Die Bewertung einer Lösung $s \in S$ erfolgt mit Hilfe einer **Fitnessfunktion**. Diese Fitnessfunktion berücksichtigt typischerweise die Zielfunktion, kann jedoch auch andere Kriterien enthalten. Zur Erzeugung der Ausgangspopulation werden andere Heuristiken, wie zum Beispiel die eingangs beschrieben Prioritätsregeln, verwendet.

Algorithmus 3.3 zeigt einen allgemeinen Genetischen Algorithmus. Ausgehend von einer Ausgangspopulation werden zwei Elternchromosomen gewählt. Mittels eines **Kreuzungsverfahrens** wird aus beiden Elternlösungen eine Kindlösung erzeugt. Die Kreuzung verwendet dabei typischerweise Teillösungen der Elternlösungen zur Erzeugung der Kindlösung. Die **Mutation** verändert Teile einer Lösung. Die resultierende Kindlösung wird der Population hinzugefügt. Auf Basis der Fitnessfunktion erfolgt zum Schluss noch eine **Selektion** der Lösungen, die in die nächste **Generation**, das heißt in die nächste Iteration, übernommen werden sollen. Das Verfahren wird so lange wiederholt, bis eine Abbruchbedingung erfüllt ist.

Algorithmus 3.3 : Allgemeiner Genetischer Algorithmus (vgl. Brucker und Knust (2006))

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $s \in S$ mit der besten gefunden Bewertung $c(s)$

```
1 Erzeuge eine initiale Population POP;
2 wiederhole
3     Wähle zwei Eltern s_m, s_f ∈ POP;
4     Erzeuge ein Kind s_c mittels Kreuzung von s_m und s_f;
5     Mutiere s_c;
6     POP ← POP ∪ s_c;
7     Reduziere POP mittels Selektion;
8 bis Abbruchbedingung erfüllt ;
9 s ← beste Lösung aus POP;
```

Unterschiedliche Varianten des Algorithmus 3.3 sind möglich. Beispielsweise können bei der Kreuzung mehr als eine Kindlösung erzeugt werden. Die Auswahl der Eltern kann zufällig erfolgen oder durch eine geeignete Suchmethode. Es existieren verschiedene Kreuzungsverfahren, abhängig von der Kodierung der Chromosomen. Mutationen wiederum können einfach nur Teile einer Lösung zufällig ändern oder bestimmte lokale Suchverfahren verwenden, um die erzeugte Lösung zu verbessern.

Problematisch bei Genetischen Algorithmen ist der benötigte Zeit- und Speicheraufwand, da Genetische Algorithmen populationsbasierte Verfahren sind und auf mehreren Lösungen operieren. Weiterhin kann es schwierig sein, Kreuzungs- und Mutationsverfahren zu finden, die für das betrachtete Problem günstig sind. Dies gilt insbesondere dann, wenn es aufgrund der Nebenbedingungen schwierig ist zulässige Lösungen zu erzeugen bzw. zu reparieren.

3.4.3 Lokale Suchverfahren

Eine **lokale Suche** ist ein iteratives Verfahren, das eine Lösung $s \in S$ des Problems (3.1) in eine Lösung $\dot{s} \in S$ überführt. Im Unterschied zu populationsbasierten Verfahren wird hier immer nur eine Lösung pro Iteration betrachtet. Das Verfahren wird so lange angewendet, bis eine Abbruchbedingung erreicht ist. Einen Überblick über verschiedene lokale Suchverfahren liefern die Arbeiten Aarts und Lenstra (2003) sowie Blum und Roli (2003).

Damit die Suche systematisch erfolgt, werden **Nachbarschaftsstrukturen** eingeführt. Die Menge $N(s)$ enthält dabei die Menge aller Nachbarlösungen von s, die in der aktuellen Iteration erreicht werden können. Eine Nachbarschaftsstruktur N kann als gerichteter Graph $G = (V, E)$ mit $V = S$ und $(s, \dot{s}) \in E$ gesehen werden (vgl. Brucker und Knust (2006)). Im Allgemeinen ist es jedoch nicht möglich, diesen Graphen komplett zu speichern. Aus diesem Grund wird eine Menge M von erlaubten **Zügen** $m : S \to S$ eingeführt. Ein Zug m erzeugt aus der Lösung $s \in S$ eine neue Lösung $\dot{s} \in S$, so dass gilt: $\dot{s} = m(s)$. Die Nachbarschaftstruktur kann nun mit Hilfe der erlaubten Züge als $N(s) = \{m(s) \mid m \in M\}$ beschrieben werden. Bei der Festlegung der Nachbarschaftsstrukturen muss beachtet werden, dass jede Lösung innerhalb des Lösungsraums nach Möglichkeit durch die erlaubten Züge erreicht werden kann.

Die Anwendung einer lokalen Suche ist besonders geeignet, wenn die Auswertung der Zielfunktion für eine Lösung schnell möglich ist, die einzelnen Züge keinen großen Aufwand verursachen und leicht sicherzustellen ist, dass ein Zug immer eine zulässige Lösung erzeugt. Dies hängt von den verwendeten Datenstrukturen innerhalb der Implementierung, aber auch von dem untersuchten Problem an sich ab.

Verfahren des steilsten Abstiegs

Eine einfache lokale Suche ist das Verfahren des **steilsten Abstiegs** und kann wie folgt beschrieben werden. Ausgehend von einer Ausgangslösung wird in einer gegebenen Nachbarschaft nach einer Lösung gesucht, welche die gegebene Startlösung soweit wie möglich verbessert. Wird eine bessere Lösung gefunden, wird mit dieser Lösung als neue

Ausgangslösung weitergesucht, ansonsten terminiert das Verfahren. Algorithmus 3.4 stellt diesen Ansatz dar.

Algorithmus 3.4 : Methode des steilsten Abstiegs

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $s \in S$ mit der besten gefunden Bewertung $c(s)$

```
1 s ← erzeuge Startlösung aus S;
2 wiederhole
3 |   ṡ ← wähle besten Nachbarn aus N(s);
4 |   wenn c(ṡ) < c(s) dann  s ← ṡ;
5 bis keine Verbesserung mehr gefunden ;
```

Der Algorithmus 3.4 endet mit einer Lösung $s \in S$. Im Allgemeinen ist s jedoch nur ein lokales Optimum und kann sich vom globalen Optimum unterscheiden. Das Verfahren verfügt jedoch über keinen Mechanismus, um aus diesem lokalen Optimum zu entkommen. Eine Möglichkeit, dieses Problem zu beheben, ist es, mit unterschiedlichen Lösungen zu starten. Weiterhin ist es möglich, Verschlechterungen während der Suche zuzulassen. Jedoch tritt dann das Problem auf, dass dieselbe Lösung immer wieder besucht wird, die Suche also im Kreis läuft. Im weiteren Verlauf wird auf einige Verfahren eingegangen werden, die dieses Problem behandeln.

Simulated Annealing

Simulated Annealing wurde erstmals in Kirkpatrick u. a. (1983) beschrieben und basiert auf einer Analogie mit dem Abkühlungsprozess in der Physik. Die grundsätzliche Idee besteht darin, dass zufällig eine Lösung $\dot{s} \in N(s)$ gewählt wird und mit einer bestimmten, immer geringer werdenden Wahrscheinlichkeit auch Verschlechterungen des Zielfunktionswertes akzeptiert werden. Verbesserungen werden jedoch immer akzeptiert. Genauer formuliert wird eine Lösung, zum Beispiel nach Brucker und Knust (2006), mit der Wahrscheinlichkeit

$$\min \left\{ 1, e^{-\frac{c(\dot{s})-c(s)}{t_i}} \right\} \tag{3.5}$$

akzeptiert. Hierbei ist t_i ein Parameter, der in Abhängigkeit von der Anzahl der bereits durchgeführten Iterationen immer kleiner wird. Es gilt $\lim_{i\to\infty} t_i = 0$. Falls $c(\dot{s}) < c(s)$ ist, liefert eine Auswertung von Beziehung (3.5) immer eins. Falls $c(\dot{s}) > c(s)$ ist, wird die Lösung mit einer bestimmten Wahrscheinlichkeit akzeptiert. Diese Wahrscheinlichkeit ist zum einen vom neuen Zielfunktionswert $c(\dot{s})$ und zum anderen vom Parameter t_i abhängig. Aufgrund des iterationszahlabhängigen Parameters t_i werden im Laufe der Zeit Verschlechterungen mit abnehmender Wahrscheinlichkeit akzeptiert.

Durch diese Strategie sucht das Verfahren zu Beginn in einer größeren Umgebung des Lösungsraums und kann auch lokale Optima überwinden. Algorithmus 3.5 zeigt eine

generische Prozedur des Simulated Annealing. In der Akzeptanzregel liefert die Funktion rand$(0, 1)$ eine Zufallszahl aus dem Intervall $[0, 1]$.

Algorithmus 3.5 : Simulated Annealing (vgl. Brucker und Knust (2006))

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $\tilde{s} \in S$ mit der besten gefundenen Bewertung $c(\tilde{s})$

1 $i \leftarrow 0$;
2 $s \leftarrow$ Erzeuge Startlösung aus S;
3 $\tilde{s} \leftarrow s$;
4 **wiederhole**
5 $\quad \dot{s} \leftarrow$ wähle zufällige Lösung aus $N(s)$;
6 $\quad$ **wenn** rand$(0, 1) < \min\left\{1, e^{\frac{c(\dot{s})-c(s)}{t_i}}\right\}$ **dann** $s \leftarrow \dot{s}$;
7 $\quad$ **wenn** $c(\dot{s}) < c(\tilde{s})$ **dann** $\tilde{s} \leftarrow \dot{s}$;
8 $\quad i \leftarrow i + 1$;
9 **bis** *Abbruchbedingung erfüllt* ;

Der Parameter t_i wird häufig durch $t_i \leftarrow \alpha t_{i-1}$ mit einem Abkühlungsfaktor $0 < \alpha < 1$ berechnet. Als Abbruchkriterien werden häufig eine maximale Anzahl von Iterationen oder das Verstreichen einer bestimmten Zeit verwendet.

Der Vorteil von Simulated Annealing im Vergleich zum Verfahren des steilsten Abstiegs ist die größere Diversifizierung bei der Durchsuchung des Lösungsraums. Die Möglichkeit des Kreisens innerhalb des Lösungsraums wird durch die zufällige Wahl einer Lösung $\dot{s} \in N(s)$ verringert. Allerdings hat Simulated Annealing keinen Mechanismus, um zu garantieren, dass eine Lösung nicht mehrmals besucht wird.

Tabu-Suche

Eine weitere lokale Suchstrategie ist die **Tabu-Suche** (vgl. Glover (1989), Glover (1990) und Glover und Laguna (1997)). Die grundlegende Idee ist dabei, dass nicht nur die aktuelle Nachbarschaft der Lösung betrachtet wird, sondern auch der bisherige Verlauf der Suche. Hierzu werden innerhalb eines **Gedächtnisses** Informationen über den bisherigen Suchverlauf mitgeführt. Auf Basis dieser Informationen wird die Nachbarschaft einer Lösung eingeschränkt, um das Kreisen im Lösungsraum zu verhindern.

Im einfachsten Fall werden die bereits besuchten Lösungen in einer Liste gespeichert und für alle folgenden Suchschritte verboten. Somit kann die Suche nicht in einem lokalen Optimum stecken bleiben, da neue Bereiche des Lösungsraums untersucht werden müssen. Eine einfache Tabu-Suche ist im Algorithmus 3.6 dargestellt.

Das Speichern der vollständigen Lösungen in der Tabu-Liste führt jedoch in vielen Fällen zu einem erheblichen Speicheraufwand. Aus diesem Grund ist die Länge der Tabu-Liste häufig begrenzt. Dadurch lässt sich jedoch nicht mehr sicher verhindern, dass Zyklen bei der Suche auftreten. Ist die Tabu-Liste jedoch hinreichend lang, ist die Wahrscheinlichkeit dafür entsprechend gering.

Algorithmus 3.6 : Einfache Tabu-Suche

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $\tilde{s} \in S$ mit der besten gefundenen Bewertung $c(\tilde{s})$

```
1 s ← Erzeuge Startlösung aus S;
2 s̃ ← s;
3 TL ← ∅;
4 wiederhole
5 |   ṡ ← wähle besten Nachbarn aus N(s) \ TL;
6 |   wenn c(ṡ) < c(s̃) dann  s̃ ← ṡ;
7 |   TL ← TL ∪ {ṡ};
8 |   s ← ṡ;
9 bis Abbruchbedingung erfüllt ;
```

Weiterhin erfordert das Vergleichen der Lösungen häufig viel Rechenzeit. Hier ist es möglich, nur die Züge, die zur Lösung geführt haben, zu speichern. Um nun zu verhindern, dass Züge nicht durchgeführt werden, obwohl die Lösungen noch nie besucht wurden, werden sogenannte Aspiranzkriterien eingeführt. Ein Zug, der diese Kriterien erfüllt, wird ausgeführt, obwohl er in der Tabu-Liste enthalten ist.

Auch bei der Tabu-Suche sind unterschiedliche Abbruchkriterien denkbar. Häufig wird nach einer bestimmten Anzahl von Iterationen ohne Verbesserung oder nach einer vorgegebenen Zeit abgebrochen.

Weitere Anpassungen der Suche können bei der Wahl der Nachbarlösung und dem Verwalten des Gedächtnisses vorgenommen werden. Bei einer kleinen Nachbarschaft ist es sinnvoll, immer den besten Nachbarn zu wählen. Ist die Nachbarschaft jedoch groß, kann es sinnvoll sein, nur den erstbesten oder gar nur einen zufälligen Nachbarn zu untersuchen.

Bezüglich der Verwaltung des Gedächtnisses während der Suche wird in Glover und Laguna (1997) vorgeschlagen, die folgenden Informationen zu speichern:

- Das **Kurzzeitgedächtnis** wird durch die bereits erwähnte Tabu-Liste realisiert. Es verhindert hauptsächlich das Kreisen um lokale Optima. Wird die Listenlänge dynamisch angepasst, ist es möglich, zu steuern, wie intensiv ein bestimmtes Gebiet des Lösungsraums untersucht wird.

- Im **Langzeitgedächtnis** wird gespeichert, wie oft welche Züge ausgeführt wurden. Mit der Berücksichtigung dieser Informationen ist es möglich, die Suche breiter zu streuen. Eine Möglichkeit besteht darin, immer nur den Zug zu verwenden, der bisher am seltensten verwendet wurde.

- In einer separaten Liste können besonders gute Lösungen gespeichert werden, um sie zu einer späteren Zeit intensiver zu untersuchen.

- Weiterhin sind Lösungen interessant, die eine große Verbesserung gegenüber ihren Vorgängerlösungen erzielt haben. Diese Lösungen können als Meilensteine interpre-

tiert werden und später dazu dienen, von einem vorherigen Zustand aus in eine andere Richtung weiterzusuchen.

Schwierigkeiten bei der Anwendung der Tabu-Suche bereiten vor allem der Aufbau der Tabu-Liste, die Festlegung der zahlreich Parameter sowie die Festlegung geeigneter Lösungsrepräsentationen, wenn auf die vollständige Speicherung der Lösungen verzichtet werden soll.

3.4.4 Ameisenalgorithmen

Ameisenalgorithmen (auch ACO = engl. *Ant Colony Optimization* genannt) gehören ebenso wie die Genetischen Algorithmen zur Klasse der populationsbasierten Metaheuristiken und basieren in der Grundidee auf dem Verhalten von Ameisen bei der Futtersuche. Sie wurden erstmals in den Arbeiten Colorni u. a. (1991), Dorigo u. a. (1991) und Dorigo (1992) vorgestellt. Aufgrund der Tatsache, dass die Lösung von vielen einzelnen autonomen Akteuren konstruiert wird, werden Verfahren wie Ameisenalgorithmen auch als Schwarmalgorithmen bezeichnet.

Jede Ameise scheidet bei der Futtersuche entlang des Weges einen Duftstoff (Pheromon) aus. Die Ameisen laufen hierbei zufällig umher, folgen aber mit erhöhter Wahrscheinlichkeit den Pheromonspuren anderer Ameisen. Da das Pheromon mit der Zeit verdunstet, ist die Pheromonspur auf kurzen Wegen intensiver als auf langen Wegen. Somit werden nach einiger Zeit kürzere Wege bevorzugt.

Innerhalb eines Ameisenalgorithmus wird dieses Vorgehen bei der Lösung von Optimierungsproblemen genutzt. Von jeder künstlichen Ameise wird dabei auf Basis von heuristischen Informationen (Prioritätsregel) sowie den historischen Erfahrungen anderer Ameisen der Kolonie eine Lösung konstruiert. Die historischen Erfahrungen sind in einer künstlichen Pheromonmatrix abgespeichert. Hier wird abgespeichert, wie erfolgreich eine Aktion einer Ameise in Bezug auf die Zielfunktion war. Der Wert aus der Pheromonmatrix wird meist multiplikativ mit dem Wert der Prioritätsregel verknüpft. Auf diese Weise ergeben sich für die Ameise die Auswahlwahrscheinlichkeiten für die nächsten möglichen Aktionen (Roulette Wheel).

Nachdem alle Ameisen eine Lösung konstruiert haben, werden diese anhand der Zielfunktion evaluiert. Gute Entscheidungen werden in der Pheromonmatrix verstärkt. Gleichzeitig werden die anderen Pheromonwerte reduziert (Verdampfung). Zu einer Steigerung der Leistungsfähigkeit des Ansatzes wird häufig zusätzlich eine lokale Suche verwendet. Im Algorithmus 3.7 ist das prinzipielle Vorgehen dargestellt.
Wie bei den Genetischen Algorithmen sind verschiedene Varianten des Algorithmus 3.7 möglich. Zunächst muss festgelegt werden, wieviele Ameisen innerhalb des Algorithmus verwendet werden. Eine größere Anzahl von Ameisen sorgt für eine stärkere **Diversifikation** der erzeugten Lösungen innerhalb des Lösungsraums, allerdings steigt damit auch der Rechenaufwand. Desweiteren ist der Anfangspheromonwert innerhalb der Pheromonmatrix festzulegen.

Bezüglich der Lösungskonstruktion sind unterschiedliche Prioritätsregeln denkbar. Eine Anpassung der Verdunstungsrate hat Einfluss auf die Konvergenz des Algorithmus. Eine niedrige Verdunstungsrate führt hierbei zu einer geringeren Konvergenz des Algorithmus.

Algorithmus 3.7 : Prinzipielles Vorgehen eines Ameisenalgorithmus

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $s \in S$ mit der besten gefunden Bewertung $c(s)$

```
1  wiederhole
2      für alle a ∈ A führe aus
3          s_a ← Ermittle eine Lösung durch die Ameise a;
4          wenn c(s_a) < c(s) dann  s ← s_a;
5      Ende
6      Verbessere s mit einer lokalen Suche;
7      Reduziere alle Pheromonwerte in der Pheromonmatrix;
8      für alle a ∈ A führe aus
9          Aktualisiere Pheromonmatrix;
10     Ende
11 bis Abbruchbedingung erfüllt ;
```

Bei der Aktualisierung der Pheromonmatrix existieren verschiedene Möglichkeiten. So kann festgelegt werden, dass statt allen Ameisen nur die bisher beste Ameise oder die iterationsbeste Ameise die Pheromonmatrix ändern darf. Weiterhin ist auch denkbar, dass einzelne Ameisen zusätzlich zur Verdunstung Pheromone wieder aus der Pheromonmatrix entfernen können. In Stützle und Hoos (2000) werden Ober- und Untergrenzen für die Pheromonwerte innerhalb der Pheromonmatrix eingeführt. Hierdurch kann eine Balance zwischen Exploration und zielgerichteter Suche sichergestellt werden. Einen aktuellen Überblick über die verschiedenen Varianten liefert Dorigo und Stütze (2004).

Neben den Ameisenalgorithmens gibt es noch weitere Schwarmalgorithmen wie beispielsweise der Particle-Swarm-Ansatz (vgl. Kennedy und Eberhart (1995)). Allen Schwarmalgorithmen ist dabei gemeinsam, dass eine Anzahl von Individuen simuliert wird, welche untereinander Informationen austauschen. Die Entscheidungen der Individuen berücksichtigen dabei die Informationen, die durch die anderen Individuen bereitgestellt wurden.

3.4.5 Variable Nachbarschaftssuche

Die **Variable Nachbarschaftssuche** (VNS) wurde erstmals in Mladenović und Hansen (1997) beschrieben. Sie basiert auf einer lokalen Suche, die systematisch verschiedene Nachbarschaftsstrukturen verwendet, um den Lösungsraum zu durchsuchen (vgl. Mladenović und Hansen (1997) und Hansen und Mladenović (2001) für eine genaue Beschreibung). Ausgehend von einer Initiallösung wird zunächst in einer (nahen) Nachbarschaft gesucht. Wird dort keine bessere Lösung gefunden, wird in einer anderen (größeren) Nachbarschaft weiter gesucht. Größere Nachbarschaften enthalten dabei typischerweise die kleineren Nachbarschaften. Algorithmus 3.8 stellt eine einfache variable Nachbarschaftssuche auf Basis des steilsten Abstiegs (**Variable Neighborhood Descent** (VND)) dar. Bemerkenswert hierbei ist, dass VND versucht, nur durch das Wechseln der Nachbarschaftsstrukturen aus einem lokalen Optimum zu entkommen.

Algorithmus 3.8 : Variable Neighborhood Descent (VND) (vgl. Hansen und Mladenović (2001))

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $s \in S$ mit der besten gefunden Bewertung $c(s)$

```
1  k_max ← Anzahl der Nachbarschaftsstrukturen für S;
2  s ← Erzeuge Startlösung aus S;
3  wiederhole
4  |  k ← 1;
5  |  solange k ≤ k_max führe aus
6  |  |  ṡ ← wähle besten Nachbarn aus N_k(s);
7  |  |  wenn c(ṡ) < c(s) dann
8  |  |  |  s ← ṡ, k ← 1;
9  |  |  sonst
10 |  |  |  k ← k + 1;
11 |  |  Ende
12 |  Ende
13 bis keine Verbesserung mehr gefunden ;
```

Ausgehend von dieser Grundidee lässt sich mittels verschiedener Erweiterungen die Suchstrategie anpassen. So ist es möglich, wie bei anderen lokalen Suchverfahren unterschiedliche Strategien zu zu verwenden, um einen Nachbarn auszuwählen.

Eine Variante von VND ist **Reduced Variable Neighborhood Search** (R-VNS). Hier wird anstatt des besten Nachbarn eine zufällige Lösung gewählt, um eine möglichst breite Streuung über den Lösungsraum zu erhalten. Dieser Schritt wird auch als Shaking bezeichnet. Die Variante **Basic Variable Neighborhood Search** (B-VNS) erweitert wiederum den R-VNS-Ansatz um eine lokale Suche, um die Suche intensivieren zu können (vgl. Algorithmus 3.9). Bei der Variante **General Variable Neighborhood Search** (G-VNS) wird VND (vgl. Algorithmus 3.8) als lokale Suche eingesetzt.
Neben diesen Basisvarianten existieren noch verschiedene andere VNS-Ausprägungen. Allen diesen Ansätzen ist gemein, dass im Unterschied zu anderen lokalen Suchverfahren nur sehr wenige Parameter benötigt werden. Schwierigkeiten ergeben sich bei der Bestimmung der Nachbarschaftsstrukturen sowie bei der Festlegung, in welcher Reihenfolge diese anzuwenden sind.

3.4.6 Dekompositionsverfahren

Ein **Dekompositionsverfahren** zerlegt das Gesamtproblem in mehrere (leichter lösbare) **Teilprobleme**. Nachdem die Teilprobleme gelöst wurden, werden die ermittelten **Teillösungen** nacheinander zur **Gesamtlösung** zusammengesetzt. Das Verfahren der dynamischen Programmierung ist ein Beispiel für ein Dekompositionsverfahren. Im Rahmen von Scheduling-Problemen lassen sich unter anderem die folgenden Dekompositionsarten unterscheiden (vgl. Ovacik und Uzsoy (1997)):

Algorithmus 3.9 : Basic VNS (B-VNS) (vgl. Hansen und Mladenović (2001))

Daten : Problem mit Lösungsraum S
Ergebnis : Lösung $s \in S$ mit der besten gefunden Bewertung $c(s)$

```
1  k_max ← Anzahl der Nachbarschaftsstrukturen für S;
2  s ← Erzeuge Startlösung aus S;
3  wiederhole
4  |  k ← 1;
5  |  solange k ≤ k_max führe aus
6  |  |  ṡ ← wähle zufälligen Nachbarn aus N_k(s) (Shaking);
7  |  |  s̈ ← führe ausgehend von ṡ eine lokale Suche aus;
8  |  |  wenn c(s̈) < c(s) dann
9  |  |  |  s ← s̈, k ← 1;
10 |  |  sonst
11 |  |  |  k ← k + 1;
12 |  |  Ende
13 |  Ende
14 bis Abbruchbedingung erfüllt ;
```

zeitliche Dekomposition: Die **zeitliche Dekomposition** macht sich den Umstand zunutze, dass bestimmte Prozessschritte nur zu bestimmten Zeiten ausgeführt werden können. So braucht ein Prozessschritt, der erst in zwei Tagen durchgeführt werden kann, nicht in einer Entscheidung für die nächsten zwei Stunden berücksichtigt zu werden.

Die zeitliche Dekomposition lässt sich wiederum in die **lineare** und **hierarchische zeitliche Dekomposition** unterscheiden. Bei der linearen zeitlichen Dekomposition erfolgen die Schedulingentscheidungen zu bestimmten Zeitpunkten. Der Scheduling-Horizont wird in mehrere kleinere Zeitintervalle aufgeteilt. Zu Beginn eines jeden Zeitintervalls erfolgt eine einzelne Scheduling-Entscheidung. Diese Art der Vorgehensweise wird auch als **Rolling Horizon Procedure** (RHP) bzw. **Zeitfensterdekomposition** bezeichnet (vgl. Morton (1981)). Das weiter oben vorgestellte Verfahren auf Basis von Prioritätsregeln ist ein Beispiel für eine einfache zeitliche, lineare Dekomposition. Zu jeder Zeit, in der eine Maschine frei wird, wird eine Scheduling-Entscheidung gefällt.

Die hierarchische zeitliche Dekomposition trifft Scheduling-Entscheidungen auf verschiedenen zeitlichen Ebenen. Bei den höheren Ebenen ist die Frequenz der Zeitpunkte, zu denen Entscheidungen getroffen werden, niedriger als bei den unteren Ebenen. Höhere und langfristigere Ebenen geben Randbedingungen für die niedrigeren Ebenen vor. Typischerweise erfolgt auf den oberen Ebenen eine grobe Terminierung durch das Setzen von Meilensteinen. Die unteren Ebenen wiederum versuchen, diese Meilesteine einzuhalten.

Dekomposition nach Entitäten: Hier erfolgt die Dekomposition anhand der **Entitäten**

des Scheduling-Problems. Beispiele hierfür sind die **los-basierte Dekomposition** oder die **Dekomposition auf Basis von Maschinengruppen** . Auch hier ist eine hierarchische Staffelung, zum Beispiel auf Grundlage von Produkten und Produktgruppen oder Fertigungsbereichen, denkbar.

Bei der los-basierten Dekomposition werden zunächst nur die Prozessschritte eines Loses geplant. Die Teillösungen der Lose werden dabei üblicherweise nacheinander in die Lösung eingebracht. Bei der Dekomposition auf Basis von Maschinengruppen werden Teillösungen für die einzelnen Maschinengruppen erzeugt und diese nacheinander zusammengesetzt.

Die Lösung eines Teilproblems erzeugt typischerweise neue Nebenbedingungen für die anderen Teilprobleme. Aus diesem Grund sind bei der Erzeugung und Zusammensetzung der einzelnen Teillösungen die Beziehungen zwischen den Teilproblemen geeignet zu berücksichtigen, damit die Gesamtlösung zulässig bleibt. Weiterhin gilt es bei hierarchischen Dekompositionsansätzen sicherzustellen, dass die Vorgaben, die durch obere Ebenen bereitgestellt werden, in den unteren Ebenen auch einhaltbar sind. Für eine weitere Behandlung von Dekompositionsverfahren sei auf Ovacik und Uzsoy (1997) verwiesen.

3.5 Integrierte Ablaufplanung komplexer Produktionssysteme mit automatischem Transport

Das betrachtete Problem (2.2) umfasst neben einem Scheduling-Problem für die flexible Werkstattfertigung auch ein Scheduling-Problem für das Transportsystem. Innerhalb dieses Abschnitts erfolgt zunächst ein Überblick über bereits durchgeführte Forschungsarbeiten. Zunächst wird ein Überblick über bestehende Ansätze für reine Werkstattfertigungen und für reine Transportprobleme gegeben. Danach werden Ansätze für integrierte Probleme vorgestellt. Im Anschluss erfolgt eine Ableitung von Anforderungen für ein integriertes Steuerungsverfahren.

3.5.1 Diskussion relevanter Literatur

Einen Überblick über das Scheduling von Maschinen mit terminorientierten Zielfunktionen liefert Sen u. a. (2003). Ansätze zum Scheduling von Batch-Maschinen werden in Potts und Kovalyov (2000) behandelt. Lösungsverfahren für allgemeine Transportprobleme werden in den Arbeiten Laporte (1992), Parragh u. a. (2008a) und Parragh u. a. (2008b) vorgestellt. Im letzten Jahrzehnt sind Ansätze vorgeschlagen worden (u. a. Ovacik und Uzsoy (1997) und Mason u. a. (2002)), um Ablaufplanungsprobleme für große komplexe Werkstattfertigungssysteme zu behandeln. Einen Überblick über das Scheduling für Werkstattfertigungen liefern die Arbeiten Pinson (1995) und Jain und Meeran (1999).

Automatische Transportsysteme für die Halbleiterfertigung sind in Agrawal und Heragu (2006), Montoya-Torres (2006) sowie Foster und Pillai (2008) beschrieben. Spezielle Steuerungsansätze für automatische Transportsysteme auf Basis von Prioritätsregeln finden sich unter anderem in Jimenez u. a. (2002), Lin u. a. (2001) und Jeong und Randhawa

(2001). Unterschiedliche Aspekte des Designs und der Steuerung von automatischen Transportsystemen sowie Prioritätsregeln zur Steuerung der Fahrzeuge werden in Le-Anh und De Koster (2006) diskutiert. Ein Nachteil dieser Ansätze ist jedoch, dass die Steuerungsentscheidungen für das Transportsystem unabhängig von den Steuerungsentscheidungen für das Bearbeitungssystem getroffen werden.

In Strusevich (1999) werden zusätzliche Verzögerungszeiten berücksichtigt, um Transportzeiten innerhalb des Produktionssystems abzubilden. Anwar und Nagi (1998) schlagen eine integrierte Heuristik für das simultane Scheduling von Bearbeitungssystem und von automatischem Transportsystem für die Montage vor. Der Ansatz basiert auf dem Konzept des kritischen Pfads mit Rückwärtsterminierung. Als Leistungsmaß wurde der Makespan betrachtet. Aufgrund der unterschiedlichen Anforderungen ist dieser Ansatz jedoch nur schwer auf die betrachteten komplexen Produktionssysteme übertragbar. Smith u. a. (1999) schlagen einen Zwei-Phasen-Ansatz vor. In der ersten Phase erfolgt die Berechnung eines Ablaufplans für das Bearbeitungssystem, ohne die zusätzlichen Restriktionen des Transportsystems zu berücksichtigen. In der zweiten Phase wird der Ablaufplan repariert, um die Restriktionen des Transportsystems zu erfüllen. Als Leistungsmaß wurde ebenfalls der Makespan betrachtet. Innerhalb der Phasen werden lokale Suchverfahren eingesetzt. Die notwendigen Reparaturen scheinen allerdings für komplexe Produktionssysteme aufgrund der komplizierteren Prozessbedingungen schwierig zu sein.

Deroussi u. a. (2008) betrachteten das Problem $J_m R_r | t'_{kh}, t_{hl} | C_{\max}$. Hierbei wird mit $C_{\max} \coloneqq max\{c_j | j = 1, \ldots, n\}$ die **maximale Zykluszeit** bezeichnet. Die vorgeschlagene Heuristik basiert auf einer deterministischen Vorwärtssimulation sowie auf lokalen Suchverfahren, um die anfänglichen Ablaufpläne zu verbessern. Der Ansatz scheint für das betrachtete Problem (2.2) nicht geeignet, da die zusätzlichen Prozessbedingungen wie Rüstzeiten, Batch-Maschinen und zyklische Losdurchläufe nur schwer zu integrieren sind.

Fließfertigung mit einem Fahrzeug wird in Hurink und Knust (2001b) behandelt. Einen Überblick über Arbeiten zu zyklischem Scheduling für die Fließfertigung mit Transport liefert Crama u. a. (2000). In Brucker u. a. (2003) wird die Fließfertigung mit Puffern betrachtet.

In Knust (1999) werden verschiedene Werkstattfertigungsprobleme mit Transportzeiten für ein Fahrzeug behandelt. In Hurink und Knust (2002) und Hurink und Knust (2005) wird eine Tabu-Suche für die Werkstattfertigung mit einem Fahrzeug vorgeschlagen. Lacomme u. a. (2005) stellen eine Heuristik auf Basis eines Branch-and-Bound-Ansatzes für die Werkstattfertigung mit einem Fahrzeug vor. In Caumond u. a. (2006) erfolgt dafür eine Formulierung als gemischt-ganzzahliges Optimierungsproblem. Lacomme und Larabi (2007) erweitert die Arbeiten um die Behandlung von mehreren Fahrzeugen. Basis aller Arbeiten ist ein entsprechend modifizierter disjunktiver Graph. Als Zielfunktion wurde in den bisherigen Arbeiten $C_{\max}$ verwendet und nicht TWT. Weiterhin erfolgte keine Behandlung von Batch-Entscheidungen und parallelen Maschinen im Zusammenhang mit den Transportrestriktionen. Die betrachteten Probleme sind außerdem sehr klein (maximal vier Maschinen, sechs Lose und vier Fahrzeuge). Es werden ausschließlich statische Probleme betrachtet.

In Qu u. a. (2004) werden Ideen zur Berücksichtigung von automatischen Transportsystemen im Rahmen der Shifting-Bottleneck-Heuristik auf konzeptioneller Ebene diskutiert.

Eine Realisierung sowie eine experimentelle Überprüfung findet jedoch nicht statt. Es werden Erweiterungen am disjunktiven Graphen zur Berücksichtigung der Transportzeiten vorgeschlagen. Anpassungen der Heuristik selbst wurden nicht vorgenommen.

Um die Evaluierung von Verfahren für die Werkstattfertigung in einem dynamischen Umfeld zu untersuchen, wird in Mönch u. a. (2003) ein System zur Leistungsbewertung von Schedulingansätzen beschrieben. Hierbei wird das Bearbeitungsbasissystem durch ein Simulationsmodell abgebildet. Das Bearbeitungssteuerungssystem wird über eine Kopplungsarchitektur mit dem Simulationsmodell verbunden. In Drießel und Mönch (2007) erfolgt die Erweiterung um die Berücksichtigung von Daten aus dem Transportsystem für das Problem (2.2).

In Huang u. a. (2007) werden Prioritätsregeln für das Bearbeitungs- und Transportsystem für 300-mm-Halbleiterfabriken vorgestellt. Im Unterschied zu den anderen Veröffentlichungen wurden Routing-Entscheidungen zur Stauvermeidung auf Basis von Wahrscheinlichkeiten berechnet, die durch Markov-Ketten ermittelt wurden. In Tyan u. a. (2004) und Drießel und Mönch (2007) werden integrierte Prioritätsregeln zur Steuerung verwendet. Es wird die Auswirkung von aufeinander abgestimmten Prioritätsregeln für das Bearbeitungs- und Transportsystem untersucht. Der Nachteil von Prioritätsregeln, dass die globalen Auswirkungen der lokalen Entscheidungen nicht abgeschätzt werden können, bleibt jedoch bestehen.

3.5.2 Anforderungen an ein integriertes Steuerungsverfahren

Es wurde bisher in der Literatur kein integriertes Schedulingverfahren für das Problem (2.2) vorgeschlagen. Aufgrund der fortschreitenden Automatisierung werden automatische Transportsysteme in modernen komplexen Produktionssystemen jedoch immer häufiger eingesetzt. Damit steigt auch der Bedarf nach Steuerungsansätzen für die Werkstattfertigung mit Berücksichtigung der innerbetrieblichen Transportentscheidungen. Hierzu strebt das zu entwickelnde Verfahren eine simultane Steuerung des Bearbeitungs- und Transportsystem an. Ausgehend von den vorigen Überlegungen hat das zu entwickelnde Verfahren den folgenden Anforderungen zu genügen:

- Funktionalitäten des Bearbeitungs- und des Transportsteuerungssystems sind zu integrieren. Daraus folgt, dass das interne Modell des Produktionssteuerungssystems Informationen aus beiden Basissystemen abbilden muss.
- Das Verfahren muss innerhalb einer dynamischen Umgebung evaluiert werden. Hierzu muss das Verfahren auch für realitätsnahe Problemstellungen hinreichend schnell sein. Die Umgebung muss die Prozessbedingungen von Problem (2.2) abbilden können.
- Das Verfahren muss neben lokalen Informationen auch globale Informationen berücksichtigen. Bei einzelnen Entscheidung sollten die globalen Auswirkungen anhand des internen Modells mit berücksichtigt werden können.
- Eine möglichst geringe Anzahl von vorzugebenden Verfahrensparametern scheint wünschenswert, um Anpassungen an neue Prozessumgebungen zu vereinfachen.

Als Basis für die softwaretechnische Implementierung und die Evaluation des zu entwickelnden Steuerungsverfahrens werden die Vorarbeiten aus Mönch u. a. (2003) und Drießel und Mönch (2007) verwendet.

Aufgrund der NP-Schwere des Ablaufplanungsproblems sind exakte Verfahren zur Lösung ungeeignet. Verfahren, die auf ganzen Lösungen operieren, erscheinen in diesem Zusammenhang ebenfalls als nicht geeignet, da die Bewertung einer Lösung sowie die Überführung einer Lösung s in eine Lösung $\dot{s}$ sehr zeitaufwändig ist. Ebenso ist es in diesem Fall nicht trivial, die Zulässigkeit der Lösung $\dot{s}$ von vornherein sicherzustellen, das heißt ohne die Lösung $\acute{s}$ zu erzeugen und danach zu evaluieren.

Entitätsbasierte Dekompositionsverfahren haben den Vorteil, dass sie einerseits die natürlichen Dekompositionsmöglichkeiten des Problems (2.2) ausnutzen und anderseits durch die Zerlegung des Gesamtproblems kleinere, leichter zu beherrschende Teilprobleme entstehen. Die Shifting-Bottleneck-Heuristik als ein Vertreter entitätsbasierter Dekompositionsverfahren hat in der Vergangenheit für verschiedene Werkstattfertigungsprobleme mit terminorientierter Zielfunktion gute Ergebnisse geliefert (siehe auch Pinedo und Singer (1999), Mason u. a. (2002) und Mönch und Drießel (2005)). Die entstehenden Teilprobleme sind Ablaufplanungsprobleme für Maschinengruppen. Die Abhängigkeiten zwischen den Teilproblemen werden mittels eines disjunktiven Graphen sichergestellt. Die datentechnische Modellierung einer Lösung für das Problem (2.2) ist nicht trivial, da Batch-Entscheidungen, parallele Maschinen und mehrere Fahrzeuge betrachtet werden. Das Modell des disjunktiven Graphen scheint hier jedoch geeignet zu sein, um diese Bedingungen abzubilden (vgl. auch u. a. Knust (1999), Oey und Mason (2001), Qu u. a. (2004) und Lacomme und Larabi (2007)).

Neuere Scheduling-Ansätze für Halbleiterfabriken basieren auf Varianten der Shifting-Bottleneck-Heuristik. In Mason u. a. (2002) wird eine Shifting-Bottleneck-Heuristik für Halbleiterfabriken mit dem Leistungsmaß TWT vorgeschlagen. Mönch und Drießel (2005) stellten eine verteilte Shifting-Bottleneck-Heuristik vor, die in ein hierarchisches Scheduling-Rahmenwerk eingebunden ist. Die Leistungsfähigkeit von Shifting-Bottleneck-Ansätzen hängt stark von der Leistungsfähigkeit der verwendeten Teilproblemlöser ab. Unterschiedliche Teilproblemlöser wurden unter anderem in Demirkol u. a. (1997) und Mönch u. a. (2007) vorgeschlagen.

In Upasani u. a. (2006) wird ein modifizierter disjunktiver Graph verwendet, der nur Knoten von Engpassmaschinen enthält, um den Rechenaufwand der Heuristik zu reduzieren. Während die ersten Veröffentlichungen nur statische Probleme untersuchten, berücksichtigen jüngere Publikationen dynamische Umgebungen (siehe auch Mönch und Drießel (2005), Mönch u. a. (2007) und Sourirajan und Uzsoy (2007)). Die begrenzte Kapazität des Transportsystems wurde bisher jedoch nach Kenntnis des Autors noch in keiner Veröffentlichung berücksichtigt.

4 Shifting-Bottleneck-Heuristik für die integrierte Ablaufplanung komplexer Produktionssysteme mit automatischem Transport

Die Shifting-Bottleneck-Heuristik (SBH) ist eine entitätsbasierte Dekompositionsheuristik für Job-Shop-Probleme, bei der das Gesamtproblem in eine Folge von Scheduling-Problemen für einzelne Maschinengruppen zerlegt wird. Diese werden sukzessive gelöst, um einen Ablaufplan für das Gesamtproblem zu ermitteln. Die Heuristik verwendet einen disjunktiven Graphen, mit dessen Hilfe die Beziehungen zwischen den einzelnen Teilproblemen berücksichtigt werden können. Hierdurch lässt sich das Gesamtproblem schneller lösen, da die Teilprobleme im Allgemeinen deutlich kleiner sind als das Gesamtproblem. Weiterhin lassen sich für die einzelnen Maschinengruppen einfach unterschiedliche Strategien implementieren.

Innerhalb dieses Kapitels erfolgt zunächst Diskussion der Vorarbeiten. Im Anschluss wird die disjunktive Graphenrepräsentation mit den notwendigen Erweiterungen beschrieben. Danach wird die Heuristik mit den Anpassungen zur Berücksichtigung des Transportsystems vorgestellt.

4.1 Diskussion von Vorarbeiten

Das Konzept des disjunktiven Graphen wurde erstmals von Roy und Sussmann (1964) für das klassische Job-Shop-Problem vom Typ $(Jm|\,|C_{\max})$ eingeführt. Es wurde in vielen Arbeiten als Basis für die Entwicklung von sowohl exakten als auch heuristischen Verfahren verwendet und an weitere Prozessbedingungen angepasst (zum Beispiel in Adams u. a. (1988), Applegate und Cook (1991), Balas (1969), Knust (1999) sowie Oey und Mason (2001)).

In Knust (1999), Brucker und Knust (2011) und Lacomme und Larabi (2007) wurden mehrere Erweiterungen des disjunktiven Graphen vorgeschlagen, mit denen Transportschritte abgebildet werden können. Es wurden sowohl Probleme mit einem Fahrzeug als auch mit mehreren Fahrzeugen modelliert. Batch-Maschinen und Maschinengruppen fanden in diesen Arbeiten jedoch keine Berücksichtigung.

Die SBH wurde in Adams u. a. (1988) für Probleme des Typs $Jm|\,|C_{\max}$ vorgestellt. Für jeden Bearbeitungsschritt werden mit Hilfe des disjunktiven Graphen früheste Verfügbarkeitszeitpunkte und gewünschte Fertigstellungstermine ermittelt. Zur Lösung der Teilprobleme von Typ $1|r_j|L_{\max}$ wird ein Branch-and-Bound-Verfahren von Carlier (1982)

verwendet. Hierbei wird mit $L_{\max} = max\{c_j - d_j | j = 1, \ldots, n\}$ die maximale Terminabweichung bezeichnet. Weiterhin werden in einer Reoptimierungsphase alle bisher gelösten Teilprobleme auf Basis der geänderten Zeiten erneut gelöst. Diese Reoptimierungsphase ist für die Erzeugung des Ablaufplans nicht zwingend notwendig, jedoch hat sich gezeigt, dass die Reoptimierungsphase zu einer deutlichen Ergebnisverbesserung führt.

In Dauzere-Peres und Lasserre (1993) und Balas u. a. (1995) wurden verzögernde Vorrangbeziehungen mit Übergangszeiten für die Teilprobleme eingeführt, um Kreise innerhalb des disjunktiven Graphen zu vermeiden. Das Verfahren von Carlier (1982) wurde entsprechend verändert. In Balas und Vazacopoulos (1998) wurde die Reoptimierungsphase durch eine lokale Suche ersetzt.

Eine Vielzahl von praktischen Anwendungsfällen (u. a. Integration von Rüstzeiten, die Abbildung von Losgrößen, Stücklisten, Montageschritten, Lossplits, überlappenden Bearbeitungsschritten, parallelen Maschinen, geplanten Aussteuerterminen und Berücksichtigung des aktuellen Arbeitsvorrats) wurde in Ivens und Lambrecht (1996) diskutiert. In Balas, Lancia, Serafini und Vazacopoulos (1998) erfolgte die Berücksichtigung von Deadlines für Probleme vom Typ $Jm|r_j, deadline|C_{\max}$.

Die SBH wurde weiterhin auf verschiedene Maschinenumgebungen und Prozessbedingungen angewendet (vgl. Demirkol u. a. (1997), Ovacik und Uzsoy (1997) und Uzsoy und Wang (2000)). Die ursprünglich für die Zielfunktion $C_{\max}$ entwickelte Heuristik wurde in Pinedo und Singer (1999) an die Zielfunktion der totalen gewichteten Verspätung angepasst.

In Ovacik und Uzsoy (1992) wurde die SBH zur Erzeugung von Ablaufplänen aus dem Testbereich von Halbleiterfabriken verwendet. Weitere Fortschritte zur Anwendbarkeit auf reale Problemstellungen brachte die Arbeit Mason u. a. (2002) durch die Behandlung von Batch-Maschinen und zyklischen Losdurchläufen. Diese Erweiterungen ermöglichen eine Anwendung der Heuristik im Front-End-Bereich von Halbleiterfabriken. In Mönch u. a. (2003) wurde die Heuristik erstmalig in einem dynamischen Umfeld getestet. In Mönch und Drießel (2005) wurde darauf basierend eine verteilte SBH vorgestellt. Das Gesamtproblem wurde hierbei in Probleme für einzelne Produktionsbereiche aufgeteilt. Die Probleme der Produktionsbereiche wurden verteilt mit Hilfe der SBH gelöst und geeignet synchronisiert. In Qu u. a. (2004) wurde auf konzeptioneller Ebene eine Erweiterung der SBH für automatische Transportsysteme diskutiert. Eine Implementierung und Leistungsbewertung fand jedoch nicht statt.

Ein kritischer Punkt für die Leistungsfähigkeit der Heuristik ist die Reihenfolge, in der die einzelnen Teilprobleme eingeplant werden. Hierzu fanden umfangreiche Untersuchungen statt (vgl. Applegate und Cook (1991), Dorndorf und Pesch (1995), Holtsclaw und Uzsoy (1996), Lee und Pinedo (1997), Ovacik und Uzsoy (1997), Uzsoy und Wang (2000), Aytug u. a. (2002) und Osisek und Aytug (2004)).

Für die Lösung der entstehenden Teilprobleme wurden unterschiedliche Verfahren wie Branch-and-Bound, Prioritätsregeln und Genetische Algorithmen eingesetzt (u. a. in Pinedo und Singer (1999), Mönch und Drießel (2005) und Mönch u. a. (2007)). Es stellte sich dabei heraus, dass nur bei bestimmten Teilproblemen der Einsatz von hochwertigen Lösungsverfahren notwendig ist.

4.2 Disjunktive Graphenformulierung

Ein Job-Shop-Problem kann als ein **disjunktiver Graph** $G = (V, E_c, E_d)$ bestehend aus den folgenden Mengen beschrieben werden:

Knoten V: Jedem einzelnen Bearbeitungsschritt O_{ij} eines Loses wird ein Knoten $v \in V$ zugeordnet. Dieser Knoten wird als **Bearbeitungsknoten** bezeichnet. Weiterhin wird ein **Startknoten** s und ein **Endknoten** e_j für jedes Los angelegt.

Konjunktive Kanten E_c: Eine konjunktive Kante ist eine gerichtete Kante (u, v) zwischen zwei Knoten $u \in V$ und $v \in V$. Konjunktive Kanten stellen festgelegte Vorrangbeziehungen zwischen den einzelnen Knoten dar. Diese Vorrangbeziehungen repräsentieren den Arbeitsplan eines Loses bzw. Reihenfolgeentscheidungen auf den Maschinen (vgl. auch Unterabschnitt 2.4.2). Jeder konjunktiven Kante ist ein Gewicht $p(u, v)$ zugeordnet, dass den zeitlichen Abstand zwischen den Bearbeitungsschritten, die mit den beiden Knoten u und v assoziiert sind, darstellt.

Das Gewicht $p(u, v)$ einer konjunktiven Kante (u, v) wird wie folgt gesetzt:

- Eine konjunktive Kante (u, v) mit $u = s$ enthält als Gewicht den geplanten Einsteuertermin r_j des Loses J_j, das heißt $p(u, v) \coloneqq r_j$. Hierbei ist v der Knoten, der den ersten Bearbeitungsschritt von J_j repräsentiert.
- Eine konjunktive Kante (u, v) mit $u \neq s$ erhält als Gewicht $p(u, v) \coloneqq p(u) \coloneqq p_{ij}$. Hierbei stellt p_{ij} die Bearbeitungszeit für O_{ij} dar.

Disjunktive Kanten E_d: Eine disjunktive Kante (u, v) ist eine ungerichtete Kante zwischen den Bearbeitungsknoten $u \in V$ und $v \in V$. Disjunktive Kanten repräsentieren noch nicht getroffene Reihenfolgeentscheidungen. Von allen Knoten, die durch ungerichtete Kanten verbunden sind, werden die zugeordneten Bearbeitungsschritte auf den Maschinen derselben Maschinengruppe ausgeführt. Eine disjunktive Kante (u, v) repräsentiert die folgenden Möglichkeiten:

1. Es existiert eine konjunktive Kante (u, v) oder
2. eine konjunktive Kante (v, u) oder
3. gar keine Kante.

Im ersten Fall wird der Bearbeitungsschritt, der mit dem Knoten u assoziiert ist, direkt vor dem mit dem Knoten v assoziierten Bearbeitungsschritt auf derselben Maschine ausgeführt. Im zweiten Fall ist die Bearbeitungsfolge umgekehrt. Im dritten Fall gibt es keine direkte Vorrangbeziehung zwischen den beiden Knoten.

Abbildung 4.1 zeigt einen disjunktiven Graphen für drei Lose, die auf vier Maschinengruppen bearbeitet werden. Die Maschinengruppen A, B und C bestehen aus je einer Maschine. Die Maschinengruppe D besteht aus zwei Maschinen. Die Maschinen sind alle sofort verfügbar. Jedem einzelnen Bearbeitungsschritt ist ein Knoten zugeordnet. Weiterhin sind ein Startknoten s und für jedes Los ein Endknoten e_j eingefügt. Die konjunktiven Kanten sind durch durchgezogene Linien und die disjunktiven Kanten

durch gestrichelte Linien dargestellt. Die Bearbeitungsreihenfolgen und -zeiten sind in Tabelle 4.1 gegeben. Weiterhin ist der Arbeitsplan, das Gewicht, der Einsteuertermin r_j sowie der geplante Aussteuertermin d_j gegeben. Der Arbeitsplan ist dabei vereinfacht in Form der einzelnen Maschinengruppen, die benötigt werden, um das Los fertigzustellen, angegeben.

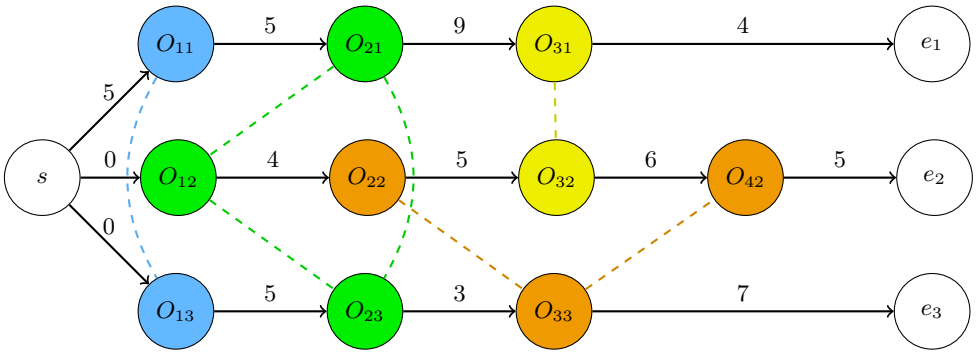

Abbildung 4.1: Graphenrepräsentation für drei Lose und vier Maschinengruppen

Tabelle 4.1: Job-Shop-Problem mit drei Losen und vier Maschinengruppen

Los	w_j	r_j	d_j	Arbeitsplan	Bearbeitungszeiten
J_1	1	5	24	A,B,C	$p_{11} = 5$, $p_{21} = 9$, $p_{31} = 4$
J_2	2	0	18	B,D,C,D	$p_{12} = 4$, $p_{22} = 5$, $p_{32} = 6$, $p_{42} = 5$
J_3	2	0	16	A,B,D	$p_{13} = 5$, $p_{23} = 3$, $p_{33} = 7$

Auf Basis des Graphen lässt sich für jeden Knoten $v \in V$ ein **frühesten Verfügbarkeitszeitpunkt** $r(v)$ aus dem längsten Pfad des Knotens s zu dem Knoten v ermitteln:

$$r(v) := \begin{cases} 0 & \text{wenn } v = s \\ \max\{r(u) + p(u,v) \mid u \in V, (u,v) \in E_c\} & \text{sonst.} \end{cases} \tag{4.1}$$

Weiterhin kann ein **frühester Fertigstellungszeitpunkt** $c(v)$ eines Knotens v ermittelt werden:

$$c(v) := \begin{cases} 0 & \text{wenn } v = s \\ r(v) & \text{wenn } v = e_j \\ r(v) + p(v) & \text{sonst.} \end{cases} \tag{4.2}$$

Handelt es sich bei v um einen Bearbeitungsknoten, so gibt $r(v)$ den frühesten Verfügbarkeitszeitpunkt $r_{ij} := r(v)$ und $c(v)$ den frühesten Fertigstellungszeitpunkt $c_{ij} := c(v)$ des

zugehörigen Bearbeitungsschritts O_{ij} an. Diese zwei Zeitpunkte geben an, wann der zugehörige Bearbeitungsschritt frühestens gestartet werden kann bzw. wann er beendet beendet ist. Der Fertigstellungszeitpunkt eines Loses J_j ergibt sich durch $c_j \coloneqq c(e_j) \coloneqq r(e_j)$.

Das Ersetzen einer disjunktiven Kante durch eine konjunktive Kante führt zusätzliche Restriktionen in das Gesamtproblem ein, die eine Eröhung der frühesten Verfügbarkeits- und Fertigstellungszeitpunkte verursachen können. Weiterhin können die zusätzliche Restriktionen dafür sorgen, dass ein Knoten die Fertigstellung von unterschiedlichen Loses verzögert. Ein Knoten u hat dann Einfluss auf den Fertigstellungszeitpunkt Loses J_k, wenn ein Pfad zu dem zugehörigen Endknoten e_k existiert. Auf Basis des längsten Pfades zwischen u und e_k lässt sich ein **gewünschter Fertigstellungstermin** $d(u,k)$ des Knotens u im Bezug auf das Los J_k ermitteln (vgl. auch Pinedo und Singer (1999)). Ist $c(u) \leq d(u,k)$, so wird die Verspätung von J_k nicht erhöht. Im Falle von $c(u) > d(u,k)$ erhöht sich die Verspätung von J_k mindestens um $c(u) - d(u,k)$. Handelt es sich bei u um einen Bearbeitungsknoten so gibt $d(u,k)$ den gewünschten Fertigstellungstermin $d_{ij}^k \coloneqq d(u,k)$ des zugehörigen Bearbeitungsschritts O_{ij} mit Bezug auf das Los J_k an. Der gewünschte Fertigstellungstermin eines Knoten u bezüglich des Loses J_k wird wie folgt berechnet:

$$d(u,k) \coloneqq \begin{cases} \infty & \text{wenn } u = e_j \text{ und } j \neq k \\ \max\{c(e_j), d_j\} & \text{wenn } u = e_j \text{ und } j = k \\ \min\{d(v,k) - p(v) \mid v \in V, (u,v) \in E_c\} & \text{sonst.} \end{cases} \tag{4.3}$$

Die Ermittlung der gewünschten Fertigstellungstermine bezüglich der einzelnen Lose kann sehr zeit- und speicherintensiv werden, da für jeden Knoten für mehrere bis alle Lose ein gewünschter Fertigstellungstermin gespeichert und berechnet werden muss. Aus diesem Grund wird in einigen Arbeiten nur der kleinste gewünschte Fertigstellungstermin für jeden Bearbeitungsschritt ermittelt, das heißt $d_{ij} \coloneqq \min\{d_{ij}^k) | k = 1, \ldots, n\}$ (vgl. u. a. Mason u. a. (2002), Mönch (2006)). Die Laufzeitersparnis wird allerdings mit einer schlechteren Abschätzung des Beitrags der Verspätung eines Knotens zur totalen gewichteten Verspätung über alle Lose erkauft. Wie stark diese Verschlechterung ausfällt, hängt von der Anzahl der Endknoten ab, die mit dem Knoten u verbunden sind, dem Gewicht der zugehörigen Lose und davon, für wie viele Lose $c(u) > d(u,k)$ gilt.

4.2.1 Abbildung von Ablaufplänen innerhalb des disjunktiven Graphen

Das Scheduling-Problem ist gelöst, wenn alle disjunktiven Kanten in konjunktive Kanten umgewandelt wurden. Hierzu ist es notwendig, dass für jede Maschinengruppe die Bearbeitungsschritte den Maschinen zugeordnet werden und die Abarbeitungsreihenfolge auf den Maschinen festgelegt wird. Werden zwei Bearbeitungsschritte direkt hintereinander auf einer Maschine bearbeitet, wird die disjunktive Kante zwischen den zugehörigen Knoten durch eine entsprechende konjunktive Kante ersetzt. Die disjunktiven Kanten zwischen Bearbeitungsknoten, deren Bearbeitungsschritte nicht direkt hintereinander auf derselben Maschine bearbeitet werden, werden aus G entfernt. Es entsteht pro Maschine eine Kette

von Kanten, die in G eingefügt wird. Der Knoten, der dem Bearbeitungsschritt zugeordnet ist, der als erstes auf einer Maschine M_m bearbeitet wird, sei nun u_m. Um sicherzustellen, dass der früheste Verfügbarkeitszeitpunkt des Knotens u_m korrekt berechnet wird, wird eine zusätzliche konjunktive Kante (s, u_m) mit dem Gewicht $p(s, u_m) := a_m$ eingefügt, wobei a_m die früheste Verfügbarkeit der Maschine M_m ist. Anderenfalls kann es passieren, dass ein zu früher Verfügbarkeitszeitpunkt für diesen Knoten berechnet wird.

Nach dem Einfügen der konjunktiven Kanten muss der Graph weiterhin azyklisch, das heißt kreisfrei, bleiben. Enthält er einen Kreis, ist der durch den Graphen repräsentierte Ablaufplan unzulässig, da in diesem Fall zwei Bearbeitungsschritte wechselseitig auf die Fertigstellung warten.

Tabelle 4.2 zeigt einen möglichen Ablaufplan für das Scheduling-Problem aus Tabelle 4.1. Für jede Maschine wurde eine Reihenfolge festgelegt, in der die einzelnen Bearbeitungsschritte der Lose bearbeitet werden sollen.

Tabelle 4.2: Bearbeitungsreihenfolge auf den einzelnen Maschinen

Maschine	Bearbeitungsreihenfolge
$M_{1,A}$	O_{11}, O_{13}
$M_{1,B}$	O_{12}, O_{21}, O_{23}
$M_{1,C}$	O_{31}, O_{32}
$M_{1,D}$	O_{22}, O_{33}
$M_{2,D}$	O_{42}

Die Verfügbarkeits- und Fertigstellungszeiten für jeden einzelnen Bearbeitungsschritt lassen sich auf Basis des disjunktiven Graphen aus Abbildung 4.2 ermitteln. Alle disjunktiven Kanten sind aus Abbildung 4.1 entfernt und die entsprechenden konjunktiven Kanten zur Abbildung der Bearbeitungsreihenfolgen eingefügt. Da alle Maschinen M_m zum Zeitpunkt $a_m := 0$ verfügbar sind, wurde auf die Darstellung der konjunktiven Kanten zur Abbildung der Maschinenverfügbarkeit aus Übersichtsgründen verzichtet.

Die sich ergebenden frühesten Verfügbarkeits- und Fertigstellungszeitpunkte der einzelnen Bearbeitungsschritte sowie die gewünschten Fertigstellungstermine bezüglich der einzelnen Lose sind in Tabelle 4.3 dargestellt. Aufgrund der Vorrangbeziehungen, die sich aus den Arbeitsplänen ergeben, können die einzelnen Bearbeitungsschritte nicht direkt hintereinander abgearbeitet werden. Dadurch treten Wartezeiten an den einzelnen Maschinen auf. Der betrachtete Ablaufplan ergibt für J_2 eine Verspätung von 16 Zeiteinheiten und für J_3 eine Verspätung von 13 Zeiteinheiten. Das Los J_1 ist nicht verspätet. Insgesamt haben alle Lose eine totale gewichtete Verspätung von 56 Zeiteinheiten.

4.2.2 Behandlung von Rüstzeiten

Maschinen, die unterschiedliche Bearbeitungsschritte ausführen, müssen gegebenfalls an diese angepasst, also gerüstet werden. Wenn eine solche Anpassung notwendig ist, treten typischerweise Rüstzeiten auf. Das Vorhandensein von Rüstzeiten erfordert Änderungen an den Kantengewichten. Bei nicht reihenfolgeabhängigen Rüstzeiten wird für eine Kante

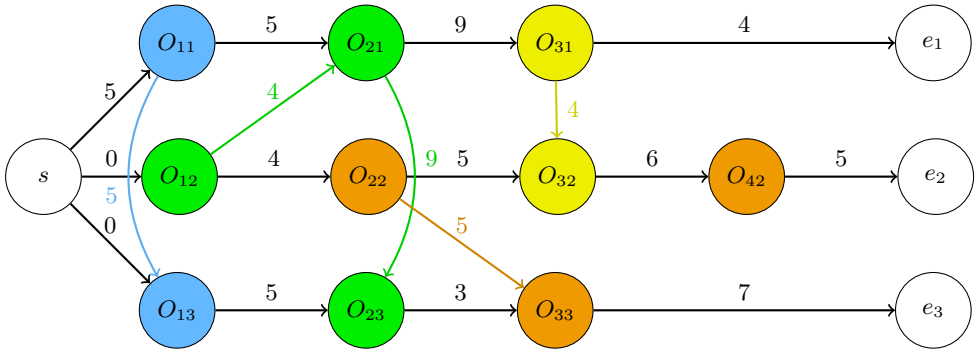

Abbildung 4.2: Graphenrepräsentation eines Ablaufplans für drei Lose und vier Maschinengruppen

(u, v) das Kantengewicht $p(u, v) := s_{ij} + p_{ij}$ gesetzt, falls u den Bearbeitungsschritt O_{ij} repräsentiert.

Im Fall von reihenfolgeabhängigen Umrüstzeiten muss unterschieden werden zwischen den Bearbeitungsschritten, die einer Maschine bereits zugeordnet wurden, und denen, wo die Zuordnung noch nicht erfolgte. Das Gewicht $p(u, v)$ einer konjunktiven Kante (u, v) wird dabei wie folgt gesetzt:

- Solange noch kein Ablaufplan für eine Maschine vorliegt, können reihenfolgeabhängige Umrüstzeiten nicht ermittelt werden. Aus diesem Grund werden sie nicht berücksichtigt.
- Eine ausgehende Kante (u, v), wobei u den Bearbeitungsschritt O_{ij} repräsentiert, erhält als Gewicht $p(u, v) = s_{lj} + p_{ij}$. Hierbei stellt s_{lj} die notwendige Rüstzeit dar, um die Maschine von dem vorherigen Los J_l auf das aktuelle Los J_j umzurüsten.

4.2.3 Berücksichtigung von Batch-Maschinen

Erweiterungen des disjunktiven Graphen zur Behandlung von Batch-Maschinen werden in Mason u. a. (2002) beschrieben. Wesentliche Idee hierbei ist das Einfügen von **Batch-Bearbeitungsknoten**. Es wird davon ausgegangen, dass eine simultane Batch-Verarbeitung bzw. eine sequentielle Batch-Verarbeitung mit Batch-Verfügbarkeit vorliegt. In diesen Fällen ist es möglich, den ausgehenden Kanten des Batch-Bearbeitungsknotens eine gemeinsame Bearbeitungszeit $p(u)$ zuzuweisen.

Jeder Batch-Bearbeitungsknoten repräsentiert die Bearbeitung eines Batches auf einer Maschine. Die Knoten der Bearbeitungsschritte, die innerhalb des Batches zusammengefasst sind, werden mit dem Batch-Bearbeitungsknoten durch konjunktive Kanten mit dem Gewicht $p(u, v) := 0$ verbunden. In Abbildung 4.3 werden die Bearbeitungsschritte O_{22} und O_{33} zu einem Batch zusammengefasst. Die Lose können erst dann auf den anderen Maschinen weiterverarbeitet werden, wenn der Batch fertig bearbeitet ist. Aus diesem Grund haben die vom Batch-Bearbeitungsknoten wegführenden Kanten das Gewicht $p(B_1) := 7$.

Tabelle 4.3: Verfügbarkeits-, Fertigstellungszeitpunkte und Fertigstellungstermine für Abbildung 4.2

Knoten	Maschine	Vorgänger	$r(v)$	$p(v)$	$c(v)$	$d(v,1)$	$d(v,2)$	$d(v,3)$
s			0		0	6	-11	-8
O_{11}	$M_{1,A}$	s	5	5	10	11	-6	-3
O_{21}	$M_{1,B}$	O_{11}, O_{12}	10	9	19	20	3	6
O_{31}	$M_{1,C}$	O_{21}	19	4	23	24	7	∞
e_1		O_{31}	23		23	24	∞	∞
O_{12}	$M_{1,B}$	s	0	4	4	11	2	4
O_{22}	$M_{1,D}$	O_{12}	4	5	9	∞	7	9
O_{32}	$M_{1,C}$	O_{22}, O_{31}	23	6	29	∞	13	∞
O_{42}	$M_{2,D}$	O_{32}	29	5	34	∞	18	∞
e_2		O_{42}	34		34	∞	18	∞
O_{13}	$M_{1,A}$	s, O_{11}	10	5	15	∞	∞	6
O_{23}	$M_{1,B}$	O_{13}, O_{21}	19	3	22	∞	∞	9
O_{33}	$M_{1,D}$	O_{23}, O_{22}	22	7	29	∞	∞	16
e_3		O_{33}	29		29	∞	∞	16

Die Ermittlung des gewünschten Fertigstellungstermins für einen Bearbeitungsschritt muss modifiziert werden. Zwei Fälle sind zu unterscheiden. Bei einem Bearbeitungsschritt, der einem Batch zugeordnet ist, wird nicht der gewünschte Fertigstellungstermin des zugeordneten Bearbeitungsknotens, sondern der gewünschte Fertigstellungstermin des zugeordneten Batch-Bearbeitungsknotens $b(v)$ verwendet. Bei Bearbeitungsschritten, die nicht in einem Batch bearbeitet werden, ändert sich die Ermittlung nicht. Der gewünschte Fertigstellungstermin d_{ij}^k wird auf Basis des O_{ij} zugeordneten Knotens v wie folgt ermittelt:

$$d_{ij}^k := \begin{cases} d(b(v),k) & \text{wenn } v \text{ einem Batch zugeordnet ist} \\ d(v,k) & \text{wenn } v \text{ keinem Batch zugeordnet ist.} \end{cases} \tag{4.4}$$

4.2.4 Berücksichtigung von Transportzeiten

Zur Berücksichtigung von Transportzeiten innerhalb des disjunktiven Graphen gibt es zwei Möglichkeiten. Der einfachste Ansatz besteht darin, eine durchschnittliche Transportzeit dem Gewicht der disjunktiven Kanten hinzuzufügen, die zwei aufeinanderfolgende Bearbeitungsschritte desselben Loses verbinden. Ein Nachteil dieses Ansatzes ist jedoch, dass er die vorhandene Kapazität des Transportsystems nicht berücksichtigt. Es wird dabei davon ausgegangen, dass eine unendliche Transportkapazität vorliegt und Transportkonflikte nicht auftreten.

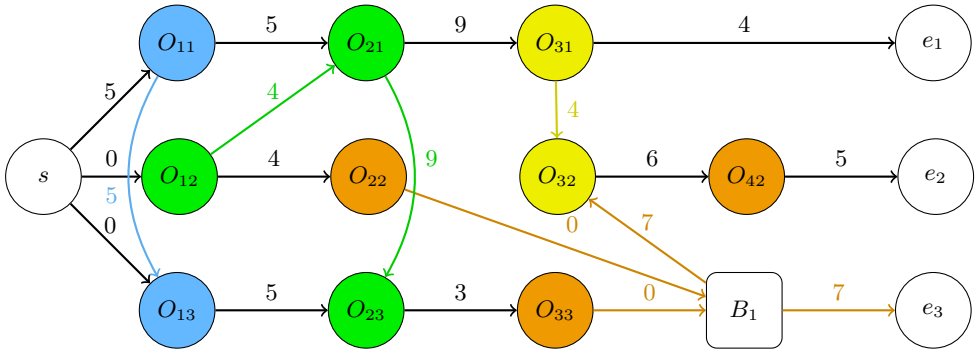

Abbildung 4.3: Graphenrepräsentation eines Ablaufplans mit Batch-Maschinen

Um Transportkonflikte zu modellieren, ist es üblich, die Transportschritte über zusätzliche **Transportknoten** abzubilden (vgl. u. a. Knust (1999), Qu u. a. (2004) und Brucker und Knust (2011)). Jeder Transportknoten repräsentiert einen Transportschritt und stellt den Transport des Loses zwischen den Ausgangs- und Zielstockern des Transportschritts dar.

Existieren mehrere Maschinen pro Maschinengruppe, sind die Transportschritte T_{ijt}, die den Transport von einem Maschinengruppenstocker zu einem Maschinenstocker bzw. von einem Maschinenstocker zu einem Maschinengruppenstocker abbilden, erst bekannt, nachdem der Bearbeitungsschritt O_{ij} einer Maschine zugewiesen wurde, das heißt ein Ablaufplan für diese Maschine erzeugt wurde. Die Transportknoten werden aus diesem Grund auch erst zusammen mit den disjunktiven Kanten, die sich aus dem Ablaufplan ergeben, eingefügt.

Abbildung 4.4 zeigt den Graphen aus Abbildung 4.1, bei dem ein Ablaufplan für die Maschinengruppen B und C implementiert wurde und die zusätzlichen Transportknoten hinzugefügt wurden. Der Transportknoten vor einem Bearbeitungsknoten repräsentiert den Transport des Loses vom Maschinengruppenstocker zur Maschine. Der Transportknoten nach einem Bearbeitungsknoten modelliert den Transport des Loses zum Maschinengruppenstocker des nachfolgenden Bearbeitungsschritts.

Das Einfügen eines Ablaufplans für ein Fahrzeug in den Graphen erfolgt analog zum Vorgehen bei den Maschinengruppen. Werden zwei Tranportschritte direkt hintereinander von einem Fahrzeug durchgeführt, wird die disjunktive Kante zwischen den zugehörigen Knoten durch eine entsprechende konjunktive Kante ersetzt. Die disjunktiven Kanten zwischen Transportknoten, deren Transportschritte nicht direkt nacheinander von demselben Fahrzeug transportiert werden, werden aus dem Graphen entfernt. Pro Fahrzeug wird eine Kette von Kanten in den Graphen eingefügt. Für den Anfangsknoten einer solchen Kette, wird durch eine zusätzliche konjunktive Kante vom Startknoten sichergestellt, dass der früheste Verfügbarkeitstermin korrekt berechnet wird.

Bezüglich der Abbildung der Transportzeiten muss analog zum Vorgehen bei den reihenfolgeabhängigen Umrüstzeiten unterschieden werden, ob das Transportteilproblem bereits gelöst wurde oder nicht. Das Gewicht $p(u, v)$ einer ausgehenden konjunktiven

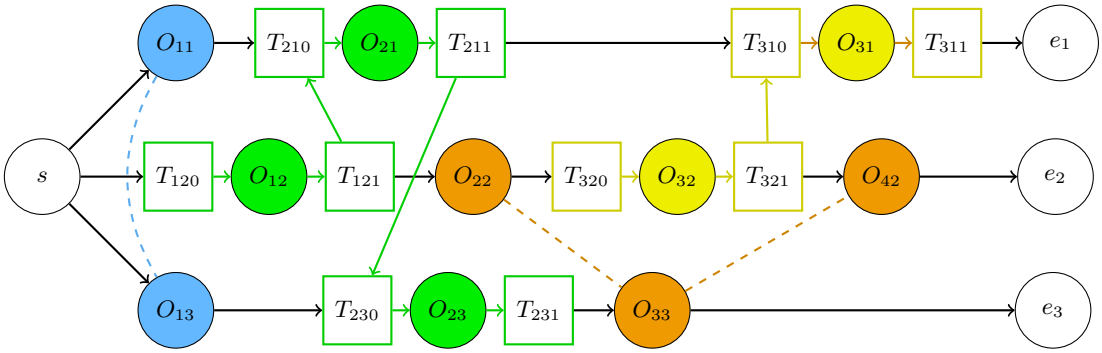

Abbildung 4.4: Graphenrepräsentation mit Transportknoten

Kante (u, v) für einen Transportknoten u wird dabei wie folgt gesetzt:

- Solange das Transportteilproblem noch nicht gelöst ist, können die Zeiten für eventuell benötigte Leerfahrten nicht ermittelt werden. Aus diesem Grund werden sie nicht berücksichtigt. Der Kante wird somit das Gewicht $p(u, v) := p(u) := t_{hl}$ zugewiesen. Hierbei stellt t_{hl} die Zeit dar, die benötigt wird, um das Los vom Ausgangsstocker P_h zu seinem Zielstocker P_l zu transportieren.

- Nachdem das Transportteilproblem gelöst ist, erhält eine ausgehende Kante (u, v) für einen Transportknoten u als Gewicht $p(u, v) = t'_{kh} + t_{hl}$. Die notwendige Zeit für die Leerfahrt, um das Fahrzeug von seinem vorherigen Stocker zum Stocker des Transportknotens zu bewegen, wird durch t'_{kh} dargestellt.

Wie bei den Bearbeitungsschritten können der früheste Verfügbarkeitszeitpunkt $r(v)$ und die gewünschten Fertigstellungstermine $d(v, k)$ eines Transportknoten v verwendet werden, um den frühesten Verfügbarkeitszeitpunkt und die gewünschten Fertigstellungstermine für den zugehörigen Transportschritt zu ermitteln, das heißt , $r_{ijt} := r(v)$ und $d^k_{ijt} := d(v, k)$.

Bei der Berücksichtigung von Transportentscheidungen muss die Behandlung von Batch-Maschinen innerhalb des disjunktiven Graphen modifiziert werden, damit $r(v)$ und $d(v, k)$ für jeden Transportknoten v korrekt berechnet werden. Es wird wieder davon ausgegangen, dass eine simultane bzw. eine sequentielle Batch-Verarbeitung mit Batch-Verfügbarkeit vorliegt. Aus diesem Grund können die Lose auch erst transportiert werden, nachdem der Batch fertig bearbeitet wurde. Gleichzeitig sind die Transportschritte, die den Transport eines Loses vom Maschinengruppenstocker zur Maschine repräsentieren, erst bekannt, nachdem die Selektion durch die jeweilige Maschine erfolgt ist.

Zur Abbildung der Verfügbarkeitsdaten der Transportschritte wird zusätzlich zum Batch-Bearbeitungsknoten ein **Batch-Bildungsknoten** für jeden Batch eingeführt. Der Batch-Bildungsknoten wird mit dem Transportknoten T_{ij0}, der den Transport vom Maschinengruppenstocker zum Maschinenstocker repräsentiert, durch eine konjunktive Kante mit dem Gewicht $p(u, v) := 0$ verbunden. Der Bearbeitungsknoten u des Bearbeitungsschritts, der vorher mit dem Transportknoten v des Transportschritt T_{ij0} über eine

konjunktive Kante verbunden war, wird nun mit dem Batch-Bildungsknoten über eine konjunktive Kante mit Gewicht $p(u, v)$ verbunden. Danach wird die Kante (u, v) entfernt.

Das Einfügen des Batch-Bearbeitungsknotens entspricht dem bereits beschriebenen Vorgehen. Die Knoten der Bearbeitungsschritte, die innerhalb des Batches zusammengefasst sind, werden mit dem Batch-Bearbeitungsknoten durch konjunktive Kanten mit dem Gewicht $p(u, v) := 0$ verbunden. Ausgehend vom Batch-Bearbeitungsknoten werden die Transportschritte, die den Weitertransport der Lose darstellen, durch konjunktive Kanten mit dem Gewicht $p(u, v) := p(u)$ verbunden, wobei $p(u)$ die gemeinsame Bearbeitungszeit der Lose des Batches darstellt. Dadurch ist sichergestellt, dass die frühesten Verfügbarkeitszeitpunkte und gewünschten Fertigstellungstermine der Transportknoten für die Lose eines Batches gleich sind. Die bereits beschriebene Modifikation bei der Ermittlung der gewünschten Fertigstellungstermine für die Bearbeitungsschritte bleibt gleich.

In Abbildung 4.5 repräsentieren die Rechtecke mit abgerundeten Ecken die Knoten für die Batchbildung und die Batchbearbeitung (BF = engl. *Batch Formation* und BP = engl. *Batch Processing*). Die Bearbeitungsschritte O_{22} und O_{33} wurden zu einem Batch zusammengefasst, wohingegen O_{42} allein bearbeitet wird. Da beide Batches nacheinander auf derselben Maschine bearbeitet werden, ist der Batch-Bearbeitungsknoten BP_1 mit dem Batch-Bildungsknoten BF_2 durch eine konjunktive Kante verbunden.

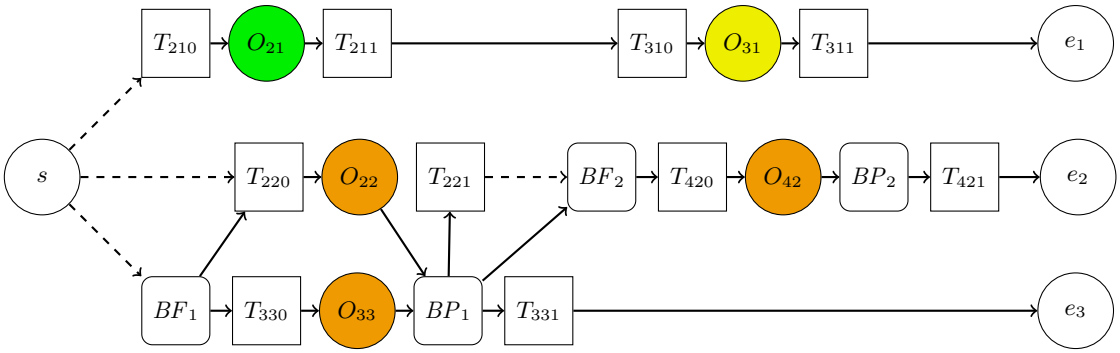

Abbildung 4.5: Disjunktiver Graph mit Batch- und Transportknoten

4.2.5 Erzeugung von disjunktiven Teilgraphen

Im Rahmen der SBH wird das Gesamtproblem in eine Folge von kleineren Teilproblemen zerlegt. Diese Teilprobleme betrachten nur einen Teil der Knoten von G. Für jeden Knoten, der vom jeweiligen Teilproblem betrachtet wird, lassen sich früheste Verfügbarkeitszeitpunkte und gewünschte Fertigstellungstermine ermitteln. Es hat sich jedoch herausgestellt, dass Teilproblemformulierungen auf Basis von frühesten Verfügbarkeitszeitpunkten und gewünschte Fertigstellungtermine nicht ausreichend sind, um zu garantieren, dass die erzeugten Teillösungen zu einer zulässigen Gesamtlösung zusammengesetzt werden können. Es müssen zusätzlich verzögernde Vorrangbeziehungen zwischen den Knoten innerhalb

des Teilproblems beachtet werden. Diese ergeben sich einerseits aus den Arbeitsplänen der Lose und andererseits aus bereits getroffenen Schedulingentscheidungen für die anderen Teilprobleme.

Eine verzögernde Vorrangbeziehung zwischen zwischen zwei für das Teilproblem relevanten Knoten u und v existiert, wenn innerhalb von G ein Pfad zwischen diesen Knoten existiert. Für jede verzögernde Vorrangbeziehung kann eine Verzögerungszeit $t(u,v) \coloneqq r(v) - (r(u) + p(u))$ ermittelt werden. Die Verzögerungszeit gibt an, wieviele Zeiteinheiten v nach der Beendigung von u startet. Durch die Berücksichtigung der verzögernden Vorrangbeziehungen wird sichergestellt, dass die zu erzeugende Teillösung bei der Einfügung in den G keinen Kreis erzeugt (vgl. Balas u. a. (1995)).

Die Ermittlung der Vorrangbeziehungen für ein Teilproblem p ist in Algorithmus 4.1 dargestellt. Zu jedem Knoten u existiert hierzu eine Menge $PRED(u)$, welche die Vorgänger des Knotens aufnimmt. Basis für die Ermittlung der Vorrangbeziehungen ist eine topologische Sortierung von G.

Algorithmus 4.1 : Ermittlung der Vorrangbeziehungen für ein Teilproblem

```
   Daten : disjunktiver Graph des Gesamtproblems G und Teilproblem p
 1 für alle u innerhalb einer topologischen Sortierung von G führe aus
 2 |   für alle ausgehenden Kanten (u, v) führe aus
 3 |   |   wenn u ∈ p dann
 4 |   |   |   PRED(v) := PRED(v) ∪ PRED(u) ∪ {u};
 5 |   |   sonst
 6 |   |   |   PRED(v) := PRED(v) ∪ PRED(u);
 7 |   |   Ende
 8 |   Ende
 9 |   wenn u ∉ p dann
10 |   |   PRED(u) := ∅;
11 |   Ende
12 Ende
```

Zur Abbildung der verzögernden Vorrangbeziehungen innerhalb eines Teilproblem kann ein **disjunktiver Teilgraph** erzeugt werden, der alle Bearbeitungs- bzw. Transportkonten, die für das Teilproblems relevant sind, beinhaltet. Dieser Teilgraph wird im weiteren Verlauf mit SG bezeichnet. Weiterhin enthält SG ebenfalls den Startknoten s sowie einen Endknoten e_j für jedes Los J_j, dessen Fertigstellungszeitpunkt durch diese Knoten beeinflusst wird. Die verzögernden Vorrangbeziehung zwischen zwei Knoten u und v werden als konjunktive Kanten (u,v) in SG eingefügt. Das Gewicht der Kante (u,v) ergibt sich auf Basis des längsten Pfades innerhalb von G mit $p(u,v) \coloneqq t(u,v) + p(u)$. Redundante Kanten werden nicht in SG eingefügt. Wenn für drei Knoten u, v und z eine Vorrangbeziehung zwischen den Knoten u und v sowie den Knoten v und z existiert, ist es nicht notwendig, die Kante (u,z) einzufügen, da das Gewicht der Kante (u,z) nicht größer sein kann als die Summe der Gewichte der beiden anderen Kanten (u,v) und (v,z). Hierdurch ist sichergestellet, dass die Berechnung der frühesten Verfügbarkeitszeitpunkte

auf Basis von SG dasselbe Ergebnis liefert wie auf Basis von G. Für die korrekte Ermittlung der gewünschten Fertigstellungstermine auf Basis von SG muss die Berechnung allerdings angepasst werden, um die Verzögerungszeiten korrekt zu berücksichtigen. Die Berechnung des gewünschten Fertigstellungstermins eines Knoten u bezüglich des Loses J_k ergibt sich innerhalb von SG mit

$$d(u,k) := \begin{cases} \infty & \text{wenn } u = e_j \text{ und } j \neq k \\ \max\{c(e_j), d_j\} & \text{wenn } u = e_j \text{ und } j = k \\ \min\{d(v,k) - p(v) - t(u,v) \mid v \in V, (u,v) \in E_c\} & \text{sonst.} \end{cases} \tag{4.5}$$

In Abbildung 4.6 ist der disjunktive Teilgraph für die Maschinengruppe D, basierend auf dem disjunktiven Graphen für das Gesamtproblem aus Abbildung 4.1, dargestellt. Neben den Knoten für die Bearbeitungsschritte O_{22}, O_{33} und O_{42}, die auf der Maschinengruppe bearbeitet werden, sind noch der Startknoten s sowie die Endknoten e_2 und e_3 dargestellt. Der Endknoten für das Los J_1 ist nicht mit übernommen, da die betrachteten Bearbeitungsschritte den Fertigstellungszeitpunkt des Loses nicht beeinflussen. Die dargestellten Kanten und ihre Gewichte ergeben sich auf Basis der längsten Pfade zwischen den einzelnen Knoten des disjunktiven Graphen für das Gesamtproblem aus Abbildung 4.1. So ergibt sich beispielsweise das Gewicht der Kante (O_{22}, O_{42}) aus der Summe der Gewichte der Kanten (O_{22}, O_{32}) sowie (O_{32}, O_{42}) $(11 = 5 + 6)$. Wie bereits beschrieben, wird auf das Einfügen von redundanten Kanten verzichtet. So kann in Abbildung 4.6 auf eine Kante (s, O_{42}) verzichtet werden, da es bereits die Kanten (s, O_{22}) und (O_{22}, O_{42}) gibt.

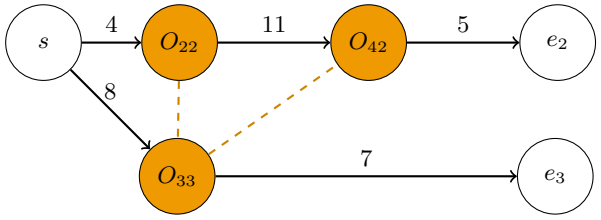

Abbildung 4.6: Disjunktiver Graph für das Teilproblem einer Maschinengruppe

4.3 Shifting-Bottleneck-Heuristik für komplexe Produktionssysteme

Innerhalb dieses Abschnitts wird die SBH für komplexe Produktionssysteme ohne Berücksichtigung von Transportentscheidungen vorgestellt. Basis der Heuristik ist ein disjunktiver Graph für das Gesamtproblem, mit dem die Abhängigkeiten der einzelnen Bearbeitungsschritte modelliert werden können. Der hier beschriebene Ansatz basiert auf den Arbeiten von Pinedo und Singer (1999) und Mason u. a. (2002).

In Algorithmus 4.2 ist das Verfahren dargestellt. Zunächst wird der disjunktive Graph G für das Gesamtproblem erzeugt. Die Menge M enthält alle zu planenden Maschinengruppen. Die Menge M_0 beinhaltet alle Maschinengruppen, für die bereits ein Ablaufplan in G eingeplant wurde.

Algorithmus 4.2 : Shifting-Bottleneck-Heuristik nach Adams u. a. (1988)

Daten : Job-Shop-Problem p
Ergebnis : Ablaufplan s für p
1 $M \leftarrow$ ermittle alle Maschinengruppen aus p, $M_0 \leftarrow \emptyset$;
2 Erzeuge einen disjunktiven Graphen G aus p;
3 **wiederhole**
4 Ermittle früheste Verfügbarkeitszeitpunkte und gewünschte Fertigstellungstermine für G;
5 $\forall m \in \{M \setminus M_0\}$: Erzeuge das Teilproblem p_m und ermittle einen zulässigen Ablaufplan s_m;
6 Ermittle die kritischen Maschinengruppe k aus $M \setminus M_0$;
7 Füge den Ablaufplan s_k in G ein und setze $M_0 \leftarrow M_0 \cup \{k\}$;
8 Reoptimiere die Maschinengruppen $(M_0 \setminus \{k\})$ unter Berücksichtigung der neuen Teillösung s_k;
9 **bis** $M_0 = M$;
10 $s \leftarrow$ ermittle Gesamtablaufplan aus G;

Auf Basis von G werden früheste Verfügbarkeitszeitpunkte und gewünschte Fertigstellungstermine sowie die verzögernden Vorrangbeziehungen ermittelt. Danach werden die Teilprobleme für die noch nicht geplanten Maschinengruppen erzeugt und gelöst. Die einzelnen Teilprobleme sind dabei unabhängig voneinander und können aus diesem Grund parallel gelöst werden. Dies wurde bisher allerdings noch nicht ausgenutzt. Für die Lösung der Teilprobleme sind unterschiedliche Verfahren wie Branch-and-Bound, Prioritätsregeln und Genetische Algorithmen möglich (vgl. auch Pinedo und Singer (1999), Mönch und Drießel (2005) und Mönch u. a. (2007)).

Um die Reihenfolge, in der die Teillösungen der einzelnen Maschinengruppen in G eingebracht werden, zu ermitteln, ist es notwendig, zu bestimmen, wie kritisch die Teillösung der jeweiligen Maschinengruppe für die Gesamtlösung ist. Kritische Maschinengruppen werden früher eingeplant als unkritische. Hierzu wurden in der Literatur unterschiedliche Strategien und Maße diskutiert (vgl. Adams u. a. (1988), Dorndorf und Pesch (1995), Holtsclaw und Uzsoy (1996), Pinedo und Singer (1999), Aytug u. a. (2002) sowie Mönch und Zimmermann (2007)).

Nachdem die kritische Maschinengruppe gewählt und die Teillösung in G eingeplant wurde, kann optional eine Reoptimierung der bereits eingeplanten Maschinengruppen unter Berücksichtigung der aktuellen Teillösung erfolgen. Hierbei wird die Teillösung einer bereits eingeplanten Maschinengruppe aus G entfernt. Es werden aktualisierte früheste Verfügbarkeitszeitpunkte und gewünschte Fertigstellungstermine ermittelt. Für die zu reoptimierende Maschinengruppe wird ein neues Teilproblem erzeugt, gelöst und in G

eingebracht. Verbessert sich der globale Zielfunktionswert nicht, so wird die alte Teillösung wiederhergestellt. In Algorithmus 4.3 ist dieser Reoptimierungsansatz dargestellt. Andere Verfahren sind denkbar. So wurde beispielsweise in Balas und Vazacopoulos (1998) die Reoptimierungsphase durch eine lokale Suche ersetzt. Bei der Leistungsbewertung in Kapitel 7 wird allerdings auf die Reoptimierungsphase verzichtet, da der Berechnungsaufwand hierfür relativ groß ist.

Algorithmus 4.3 : Reoptimierung im Rahmen der Shifting-Bottleneck-Heuristik

```
   Daten : Job-Shop-Problem p
 1 für alle m ∈ {M_0 \ {k}} führe aus
 2 |   K_alt ← ermittle Zielfunktionswert aus G;
 3 |   s_alt ← speichere aktuelle Teillösung;
 4 |   Entferne s_alt aus G;
 5 |   Ermittle früheste Verfügbarkeitszeitpunkte und gewünschte
   |   Fertigstellungstermine für G;
 6 |   Erzeuge das Teilproblem p_m und ermittle einen Ablaufplan s_neu;
 7 |   Füge s_neu in G ein;
 8 |   K_neu ← ermittle Zielfunktionswert aus G;
 9 |   wenn K_neu > K_alt dann
10 |   |   Entferne s_neu aus G;
11 |   |   Füge s_alt in G ein;
12 |   Ende
13 Ende
```

4.3.1 Erzeugung eines Teilproblems für eine Maschinengruppe

Im Rahmen der SBH für komplexe Produktionssysteme wird ein Job-Shop-Problem vom Typ (2.1) in eine Folge von Teilproblemen vom Typ

$$Pm|r_{ij}, s_{lj}, \text{batch}, \text{incompatible}, \text{prec}|\,TWT \tag{4.6}$$

zerlegt. Die Teilprobleme sind Scheduling-Probleme für Maschinenguppen mit identischen parallelen Maschinen (Pm). Es müssen dynamische Ankunftszeiten r_{ij}, reihenfolgeabhängige Umrüstzeiten (s_{lj}) und inkompatible Batch-Familien (batch, incompatible) berücksichtigt werden. Maschinen, die nur Einzellose verarbeiten, sind hierbei ein Spezialfall, bei dem für die maximale Batch-Größe $B = 1$ gilt.

Wie in Unterabschnitt 4.2.5 beschrieben, wird mit dem Teilproblem auch ein disjunktiver Teilgraph SG erzeugt, der alle Bearbeitungsknoten enthält, deren Bearbeitungsschritte auf den Maschinen der Maschinengruppe bearbeitet werden können. Innerhalb des Teilproblems sind Entscheidungen bezüglich

- der Zuordnung der Bearbeitungsschritte zu den einzelnen Maschinen,
- der Bildung von Batches für einzelne Bearbeitungsschritte auf einer Maschine und

- der Reihenfolge, in der die Batches auf den einzelnen Maschinen ausgeführt werden

zu treffen. Hierbei sind neben den Informationen, die durch SG gegeben sind, die Verfügbarkeitszeitpunkte der Maschinen, Losfamilien und reihenfolgeabhängige Umrüstzeiten zu berücksichtigen.

Das Ziel ist die Minimierung der totalen gewichteten Verspätung, wobei das Leistungsmaß nur für die Lose berechnet wird, deren Endknoten in SG enthalten sind. Die Menge dieser Lose sei mit $J(SG)$ bezeichnet. Nachdem das Problem gelöst ist, sind alle disjunktiven Kanten in SG in konjunktive umgewandelt und die Fertigstellungszeitpunkte für die Lose können mit Hilfe von SG ermittelt werden. Die Verspätung eines Loses J_k ist nun mindestens $T_k \coloneqq \max(c_k - d_k, 0)$. Mit c_k wird hier der Fertigstellungszeitpunkt des Lose J_k auf Basis des Endkotens e_k in SG bezeichnet. Da das Teilproblem nur die Lose innerhalb $J(SG)$ beeinflusst, erfolgt die Berechnung der totalen gewichteten Verspätung mit

$$TWT \coloneqq \sum_{J_k \in J(SG)} w_k T_k. \tag{4.7}$$

Die Verspätung eines Bearbeitungsschritts O_{ij} im Bezug auf das Los J_k ist $T_{ij}^k \coloneqq \max(c_{ij} - d_{ij}^k, 0)$. Da alle Bearbeitungsschritte geplant werden müssen, ist die Verspätung des Loses J_k mindestens

$$T_k \coloneqq \max_{ij}(T_{ij}^k), \tag{4.8}$$

das heißt für jedes Los J_k wird T_{ij}^k über alle Bearbeitungsschritte des Teilproblems maximiert. Aus diesem Grund ist es sinnvoll Formel (4.7), wie von Pinedo und Singer (1999) vorgeschlagen, in

$$TWT \coloneqq \sum_{J_k \in J(SG)} w_k \max_{ij}(T_{ij}^k) \tag{4.9}$$

umzuformulieren. Diese Umformulierung ist im weiteren Verlauf der Arbeit hilfreich bei der Motivation eines Prioritätsindexes zur Lösung des Teilproblems für eine Maschinengruppe.

4.3.2 Berücksichtigung von Transportentscheidungen innerhalb der Shifting-Bottleneck-Heuristik

Bei der Berücksichtigung von Transportentscheidungen innerhalb der SBH für Probleme vom Typ (2.2) wird das Transportsystem als zusätzliches Teilproblem des Typs

$$R_r | r_{ij}, t'_{kh}, t_{hl}, \text{prec} | TWT \tag{4.10}$$

betrachtet. Durch R_r wird ein Transportsystem mit r Fahrzeugen beschrieben. Die Zeit für den Transport eines Loses ist abhängig von dessen Ausgangs- und Zielstockern (t_{hl}).

Falls das Fahrzeug sich an einem anderen Stocker befindet als das zu transportierende Los, müssen Leerfahrten berücksichtigt werden (t'_{kh}). Wie bei den Teilproblemen für die Maschinengruppen müssen auch hier früheste Verfügbarkeitszeitpunkte (r_{ij}) berücksichtigt werden. Das Leistungsmaß ist ebenfalls die totale gewichtete Verspätung. Auch hier dient die Berücksichtigung von verzögernden Vorrangbeziehungen (prec) dazu, die Entstehung von Kreisen innerhalb von G zu verhindern.

Da ein Bearbeitungsschritt üblicherweise auf unterschiedlichen Maschinen einer Maschinengruppe ausgeführt werden kann, ist ein Transportschritt, der den Transport des Loses von bzw. zu einem Maschinenstocker abbildet, erst bekannt, nachdem der zugehörige Bearbeitungsschritt einer Maschine zugeordnet wurde. Aus diesem Grund ist während der SBH nur ein Teil des Transportteilproblems bekannt. Vollständig bekannt ist das Teilproblem für das Transportsystem erst, nachdem alle Maschinengruppen geplant wurden. Es werden zwei Möglichkeiten vorgeschlagen, wie das Teilproblem für das Transportsystem berücksichtigt werden kann:

- Das Teilproblem für das Transportsystem wird gelöst, nachdem alle Maschinengruppenteilprobleme in G implementiert wurden. Dieser Ansatz wird als **hierarchische Berücksichtigung von Transportzeiten** (SBH-HT) bezeichnet.

- Nachdem ein Ablaufplan für ein Maschinengruppenteilproblem in G eingefügt wurde, ist ein Teil des Transportteilproblems bekannt. Zunächst werden alle vorher eingefügten Transportteillösungen aus G entfernt. Danach erfolgt die Lösung des teilweise bekannten Teilproblems für das Transportsystem und die Einfügung dieses Ablaufplans in G. Dieser Ansatz wird als **simultane Berücksichtigung von Transportzeiten** (SBH-ST) bezeichnet.

Es ist leicht ersichtlich, dass der zweite Ansatz gegenüber dem ersten einen deutlich höheren Rechenaufwand verursacht. Andererseits wird jedoch auch mit einem besseren Ergebnis gerechnet, da im simultanen Fall genauere Abschätzungen für die frühesten Verfügbarkeitszeitpunkte und gewünschten Fertigstellungstermine für die restlichen Maschinengruppen vorliegen.

4.3.3 Erzeugung des Teilproblems für das Transportsystem

Auch bei der Erzeugung des Teilproblems für das Transportsystem wird ein disjunktiver Teilgraph SG auf Basis von G erzeugt. Dieser Teilgraph enthält alle aktuell bekannten Transportknoten sowie den Startknoten s und die Endknoten e_j für jedes Los J_j. Analog dem Teilproblem für eine Maschinengruppe müssen neben den Informationen, die durch SG gegeben sind, noch die Verfügbarkeitszeitpunkte der Fahrzeuge berücksichtigt werden.

In der Literatur werden Transportplanungsprobleme umfangreich diskutiert. Einen Überblick liefern die Arbeiten Laporte (1992), Parragh u. a. (2008a, b). Die betrachteten Teilprobleme vom Typ (4.10) können als eine Generalisierung von Vehicle-Routing-Problemen mit Pickup, Delivery und Zeitfenstern betrachtet werden. Für jeden Transportschritt existieren prinzipiell mehrere Zeitfenster $I(T_{ijt}, k)$ und Vorrangbeziehungen,

die berücksichtigt werden müssen. Innerhalb des Teilproblems für das Transportsystem sind Entscheidungen bezüglich

- der Zuordnung der Lose zu den einzelnen Fahrzeugen,
- der Reihenfolge, in der die Lose auf einem einzelnen Fahrzeug transportiert werden, und
- des Transportweges, den die Fahrzeuge fahren (Routing),

zu treffen. Da die Fahrzeuge auf Schienen fahren, können zwei Fahrzeuge sich typischerweise nicht überholen. Aus diesem Grund sind Staus in realen Halbleiterfabriken innerhalb des Transportsystems nicht selten. Weiterhin gibt es häufig mehrere Wege, um ein Los von einem Stocker zu einem anderen zu transportieren. Durch eine entsprechende Auswahl des Transportweges ließen sich die Auswirkungen von Staus innerhalb des Transportsystems verringern.

Viele MCS treffen die Bestimmung des Transportweges autonom und sehen eine Vorgabe des Transportweges von außerhalb nicht vor (vgl. auch Foster und Pillai (2008)). Auch der in Kapitel 6 verwendete Transportsystemsimulator lässt eine direkte Steuerung des Fahrzeuges bzgl. der Wegfindung nicht zu. Aus diesem Grund erfolgt im Rahmen dieser Arbeit keine Bestimmung des Transportweges. Es wird davon ausgegangen, dass die Routing-Entscheidung – und damit auch die direkte Staubehandlung – durch das MCS erfolgt. Da ein Routing nicht erfolgt, ist es notwendig, die Zeiten für eine Fahrt zwischen zwei Stockern abzuschätzen. Hierzu werden die Transportzeiten zwischen den einzelnen Stockern gemessen und der gleitende Durchschnitt für die letzten fünf Transporte ermittelt. Sind noch keine vorherigen Transporte bekannt, wird die benötigte Zeit anhand des kürzesten Weges und der Maximalgeschwindigkeit des Fahrzeuges abgeschätzt. Hierdurch passen sich die Zeiten für Leerfahrten t'_{kh} und die reinen Transportzeiten t_{hl} an die Stausituation des Transportbasissystems an.

Die Vernachlässigung der Wegentscheidung ermöglicht die Formulierung von Problemen des Typs (4.10) als Belegungsprobleme für parallele Maschinen vom Typ

$$P_m|r_j, s_{lj}, \text{prec}|\, TWT. \tag{4.11}$$

Hierbei entspricht jedes Fahrzeug einer Maschine und jeder Stocker innerhalb des Transportsystems einem Rüstzustand (vgl. auch Knust (1999) sowie Lacomme und Larabi (2007) für ein ähnliches Vorgehen). Die Transportschritte werden als Bearbeitungsschritte interpretiert. Der Stocker, an dem sich ein Fahrzeug aktuell befindet, wird durch den aktuellen Rüstzustand einer Maschine repräsentiert. Die benötigte Zeit für eine Leerfahrt t'_{kh} entspricht dabei der Rüstzeit auf einer Maschine. Die eigentliche Transportzeit des Loses t_{hl} entspricht der Bearbeitungszeit. Da sich das Fahrzeug bei einem Lostransport von einem Stocker zum anderen bewegt, ist es wichtig zu erwähnen, dass sich der Rüstzustand der zugeordneten Maschine im Laufe der Bearbeitung ebenfalls ändert, das heißt, zu Beginn der Bearbeitung besitzt die Maschine einen anderen Rüstzustand als nach

der Bearbeitung. Die frühesten Verfügbarkeitszeitpunkte und gewünschten Fertigstellungstermine entsprechen den frühesten Verfügbarkeitszeitpunkten und gewünschten Fertigstellungsterminen der entsprechenden Transportschritte.

Die Umwandlung in ein paralleles Maschinenbelegungsproblem hat den Vorteil, dass für alle Teilprobleme prinzipiell dieselben Lösungsverfahren angewendet werden können. Die umgewandelten Probleme des Typs (4.11) sind Spezialfälle von Teilproblemen des Typs (4.6), bei denen $B = 1$ gilt.

4.3.4 Bestimmung der Reihenfolge der Einplanung der Teilproblemlösungen

Bei der Lösung der Teilprobleme werden typischerweise mehrere Lösungen erzeugt. Hierbei wird eine Minimierung der TWT für die Gesamtlösung angestrebt, das heißt, es wird von zwei gegebenen Lösungen die gewählt, die den geringeren TWT-Wert aufweist.

Nachdem die einzelnen Teilprobleme gelöst wurden, muss innerhalb der SBH (Algorithmus 4.2) bestimmt werden, welches der Teilproblemlösungen in G eingefügt wird. Hierzu wird das Teilproblem gewählt, das den größten TWT-Wert aufweist. Es hat sich jedoch herausgestellt, dass die Verwendung von TWT als alleiniges Leistungsmaß im dynamischen Umfeld keine zufriedenstellenden Ergebnisse liefert, da unter Umständen die Auslastung der Maschinen bzw. Fahrzeuge nicht hoch genug ist oder in manchen Situationen der TWT-Wert keine Unterscheidung der Teilprobleme ermöglicht.

In dem Fall, dass zwei Teilprobleme denselben TWT-Wert besitzten, erfolgt die Bewertung zusätzlich auf Basis der **maximalen gewichteten Terminabweichung** $WL_{\max} := \max\{ WL_k | k = 1, \ldots, n\}$ ($WL_{\max}$ = engl. *Maximum Weighted Lateness*) mit

$$WL_j := \begin{cases} w_j L_j & \text{wenn } L_j := c_j - d_j > 0 \\ \frac{L_j}{w_j} & \text{sonst.} \end{cases} \tag{4.12}$$

.

Hierbei wird mit w_j das Gewicht des Loses J_j bezeichnet. Der Fertigstellungstermin c_j des Loses J_j wird mit Hilfe von SG berechnet. Die Lösung des Teilproblems mit dem größten Wert für $WL_{\max}$ wird in G eingefügt. In dem Fall, dass zwei Teilprobleme ebenfalls denselben Wert für $WL_{\max}$ aufweisen, wird die Teillösung gewählt, welche die höchste Zykluszeit $C_{\max} := \max\{c_j | k = 1, \ldots, n\}$ auf Basis von SG hat.

Hierbei ist zu beachten, dass die oben genannten Leitungsmaße auch innerhalb der einzelnen Teilprobleme als Zielfunktionen eingesetzt sind. Dort ist die Logik allerdings umgekehrt, d.h., es wird versucht jeweils eine Teillösung zu finden, die diese Leistungsmaße in der angegeben Reihenfolge minimiert.

4.4 Einsatz der Shifting-Bottleneck-Heuristik innerhalb des Steuerungssystems

Beim Einsatz der SBH zur Steuerung des Produktionsbasissystems erfolgt der Aufruf der Heuristik rollierend alle τ_Δ Zeiteinheiten. Innerhalb dieser Arbeit wird τ_Δ als **Aufrufintervall** bezeichnet. Beim Aufruf der Heuristik wird der aktuelle Zustand der Lose,

der Fahrzeuge sowie der Maschinen des Produktionsbasissystems berücksichtigt. Weiterhin erfolgt die Planung für einen **Planungshorizont** der Länge $\tau_\Delta + \alpha_\Delta$, wobei α_Δ als **zusätzlicher Planungshorizont** bezeichnet wird. Beim Aufruf der Heuristik werden nur die Lose berücksichtigt, die innerhalb des Planungshorizonts verfügbar sind. Von diesen zu planenden Losen werden weiterhin auch nur die Bearbeitungsschritte betrachtet, die voraussichtlich innerhalb des Planungshorizonts verfügbar sind. Durch das Aufrufintervall τ_Δ ist sichergestellt, dass alle τ_Δ Zeiteinheiten der aktuelle Status des Produktionsbasissystems erneut berücksichtigt wird.

Um zu ermitteln, welche Bearbeitungsschritte innerhalb des Planungshorizonts liegen, ist es notwendig, für jeden Bearbeitungsschritt O_{ij} einen **voraussichtlichen Starttermin** s_{ij} und einen **voraussichtlichen Fertigstellungstermin** f_{ij} abzuschätzen.

4.4.1 Abschätzung der voraussichtlichen Start- und Fertigstellungstermine

Zur Abschätzung der voraussichtlichen Start- und Fertigstellungstermine wird das **Infinite-Capacity-Algorithm**-Verfahren (ICA) aus Atherton und Atherton (1995) um die Berücksichtigung der durchschnittlichen Wartezeiten vor den einzelnen Maschinengruppen erweitert. Das einfache ICA-Verfahren ermittelt die voraussichtlichen Start- und Fertigstellungstermine ohne Berücksichtigung der Kapazitäten und Auslastungen der einzelnen Maschinengruppen. Es zeichnet sich durch eine hohe Ausführungsgeschwindigkeit aus, da die Bearbeitungsschritte eines Loses unabhängig von den Bearbeitungsschritten der anderen Lose terminiert werden. Das **Modifizierte ICA-Verfahren** (MICA) berücksichtigt die aktuelle Auslastung der Maschinengruppen auf Basis der durchschnittlichen Wartezeiten vor den einzelnen Maschinengruppen. Hierdurch kann bei geringer längerer Laufzeit eine fairere Abschätzung der voraussichtlichen Start- und Fertigstellungstermine erreicht werden.

Der nächste auszuführende Bearbeitungsschritt für ein Los J_j zum Zeitpunkt t wird im weiteren Verlauf mit $O_{n_k j}$ bezeichnet. Zunächst wird für J_j ein **globaler Flussfaktor**

$$GFF_j = \max\left(\frac{d_j - \max(t, r_j)}{\sum_{i=n_k}^{n_j} p_{ij}}, 1\right) \tag{4.13}$$

ermittelt. Der Flussfaktor gibt an, wie viel Wartezeit in der zur Verfügung stehenden Restzeit enthalten ist. Innerhalb des einfachen ICA-Verfahrens werden die voraussichtlichen Start- und Fertigstellungstermine rekursiv mit

$$s_{ij} \coloneqq \begin{cases} \max(r_j, t) & \text{wenn } i = n_k \\ f_{i-1,j} & \text{wenn } i > n_k \end{cases} \tag{4.14}$$

und

$$f_{ij} \coloneqq s_{ij} + GFF_j \cdot p_{ij} \tag{4.15}$$

festlegt. Zur Berücksichtigung der Auslastung der einzelnen Maschinengruppen wird davon ausgegangen, dass für jeden Bearbeitungsschritt neben der Bearbeitungszeit p_{ij} eine aktuelle durchschnittliche Wartezeit w_{ij} bekannt ist. Diese Wartezeit stellt die durchschnittliche Zeit dar, die ein Bearbeitungsschritt warten muss, bevor er von der Maschinengruppe zur Bearbeitung ausgewählt wird. Die Ermittlung dieser Wartezeiten erfolgt durch das Bearbeitungssteuerungssystem auf Basis der tatsächlichen Verfügbarkeits- und Fertigstellungszeitpunkte der einzelnen Bearbeitungsschritte. Es werden dabei die letzten fünf Lose betrachten, die den jeweiligen Bearbeitungsschritt des Arbeitsplans auf der Maschinengruppe durchlaufen haben. Damit lässt sich nun für jedes Los ein **globaler Wartezeitfaktor**

$$GWF_j = \frac{\sum_{i=n_k}^{n_j}(p_{ij} + w_{ij})}{\sum_{i=n_k}^{n_j} p_{ij}} \tag{4.16}$$

sowie für jeden Bearbeitungsschritt ein **lokaler Wartezeitfaktor**

$$LWF_{ij} = \frac{p_{ij} + w_{ij}}{p_{ij}} \tag{4.17}$$

berechnen. Mit Hilfe der Faktoren (4.13), (4.16) und (4.17) kann nun ein **lokaler Flussfaktor**

$$LFF_{ij} = \frac{GFF_j}{GWF_j} \cdot LWF_{ij} \tag{4.18}$$

für jeden Bearbeitungsschritt ermittelt werden, der die zur Verfügung stehende Restzeit des Loses und die aktuelle Auslastung der Maschinengruppe berücksichtigt. Je höher die Auslastung der jeweiligen Maschinengruppe ist, desto größer ist typischerweise die Warteschlange vor der Maschinengruppe und damit auch die durchschnittliche Wartezeit.

Eine größere durchschnittliche Wartezeit führt in MICA zu einem höheren lokalen Flussfaktor. Bei Maschinengruppen mit einer geringen durchschnittlichen Wartezeit ist der lokale Flussfaktor entsprechend kleiner. Die voraussichtlichen Start- und Fertigstellungstermine für einen Bearbeitungsschritt lassen sich nun analog zu (4.14) und (4.15) mit

$$s_{ij} \coloneqq \begin{cases} \max(r_j, t) & \text{wenn } i = n_k \\ f_{i-1,j} & \text{wenn } i > n_k \end{cases} \tag{4.19}$$

und

$$f_{ij} \coloneqq s_{ij} + LFF_{ij} \cdot p_{ij} \tag{4.20}$$

festlegen. Die Ermittlung der voraussichtlichen Start- und Fertigstellungstermine erfolgt synchron mit dem Aufruf der SBH rollierend alle τ_Δ Zeiteinheiten. Durch die Verwendung von lokalen Flussfaktoren passt sich das Verfahren an die Auslastung der einzelnen Maschinengruppen an.

4.4.2 Erzeugung des disjunktiven Graphen in einem dynamischen Umfeld

Bei der Erzeugung des disjunktiven Graphen G zum Zeitpunkt t werden nur die Lose berücksichtigt, deren Einsteuertermin innerhalb des Planungshorizonts liegt, das heißt, $r_j < t + \tau_\Delta + \alpha_\Delta$. Weitehin werden nur die Bearbeitungsschritte eines Loses betrachtet, deren voraussichtlicher Starttermin ebenfalls innerhalb des Planungshorizonts liegt, das heißt, $s_{ij} < t + \tau_\Delta + \alpha_\Delta$. Die Vorgehensweise ist in Abbildung 4.7 dargestellt. Die voraussichtlichen Start- und Fertigstellungstermine für die Bearbeitungsschritte von vier Lose werden gezeigt. Die Bearbeitungsschritte, die innerhalb des Planungshorizonts liegen, sind hervorgehoben.

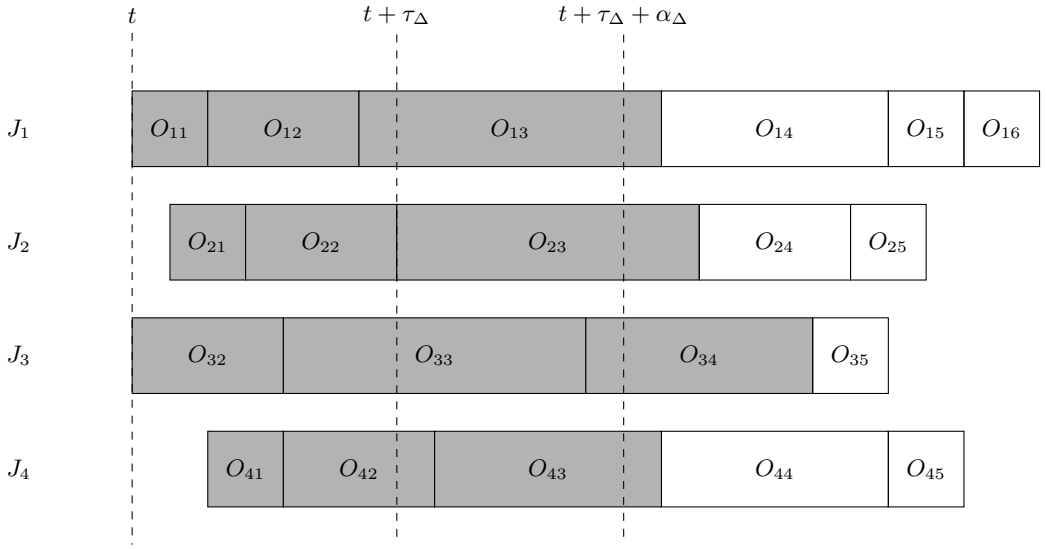

Abbildung 4.7: Bearbeitungsschritte innerhalb des Planungshorizont

Für jeden der Bearbeitungsschritte wird ein Knoten innerhalb von G erzeugt. Entsprechend des Arbeitsplans des Loses werden diese Knoten durch Ketten von konjunktiven Kanten verbunden. Der bezüglich des Arbeitsplans letzte innerhalb von G berücksichtigte Bearbeitungsschritt des Loses J_j sei an dieser Stelle mit O_{ij} bezeichnet. Der Endknoten des Loses J_j wird nun ebenfalls in G eingefügt und über eine konjunktive Kante (O_{ij}, e_j) mit dem Knoten des Bearbeitungsschritts O_{ij} verbunden. Als geplanten Aussteuertermin d_j für den Endknoten e_j wird der geplante Fertigstellungstermin f_{ij} des letzten hinzugefügten Bearbeitungsschritts O_{ij} verwendet. Durch dieses Vorgehen wird der zu verwaltende Graph und die zu lösenden Teilprobleme kleiner. Entsprechend der Wahl der Parameter τ_Δ und α_Δ führt dies zu einem geringeren Speicherbedarf und einer größeren Ausführungsgeschwindigkeit der Heuristik.

Abhängig vom aktuellen Zustand des Loses werden zusätzliche Knoten und Kanten am Anfang der Knotenkette eingefügt. Im Folgenden werden die Zustände aufgeführt, die berücksichtigt werden müssen:

Das Los J_j wartet auf die Selektion durch eine Maschine: Das Los befindet sich hierbei in einem Maschinengruppenstocker. Der Startknoten s wird direkt mit dem Bearbeitungsknoten für O_{n_kj} durch eine konjunktive Kante mit dem Gewicht $\max(r_j, t)$ verbunden. Da noch keine Selektion erfolgt ist, können auch noch keine Transportknoten erzeugt werden.

Das Los J_j wird auf einer Maschine bearbeitet: Da der Transport zum nachfolgenden Maschinengruppenstocker bereits bekannt ist, wird der entsprechende Transportknoten mit Kanten erzeugt. Die konjunktive Kante (s, v), wobei v den Transportknoten bezeichnet, besitzt als Gewicht den Fertigstellungszeitpunkt des Loses auf der Maschine.

Das Los J_j wartet bei einer Maschine auf den Bearbeitungsbeginn: Der Maschine ist hierbei bereits ein Batch zur Bearbeitung bekannt. Es werden zunächst ein entsprechender Batch-Bearbeitungsknoten sowie die zugehörigen Kanten innerhalb von G angelegt. Da der Transport zum nachfolgenden Maschinengruppenstocker bereits bekannt ist, wird auch der entsprechende Transportknoten mit den dazugehörigen Kanten erzeugt. Der Startknoten s wird mit dem Bearbeitungsknoten für O_{n_kj} durch eine konjunktive Kante mit dem Gewicht t verbunden.

Das Los J_j wartet auf die Selektion durch ein Fahrzeug: Das Los befindet sich dabei in einem Stocker. Handelt es sich bei dem Stocker um einen Maschinengruppenstocker, dann hat eine Maschine das Los bereits selektiert und wartet darauf, die Bearbeitung zu starten. In diesem Fall wird zunächst der notwendige Transportknoten eingefügt. Danach erfolgt im Falle eine Batch-Maschine die Anlage des Batch-Bearbeitungsknoten mit den dzugehörigen Kanten. Da auch hier der Transport zum nachfolgenden Maschinengruppenstocker bereits bekannt ist, wird der entsprechende Transportknoten mit den notwendigen Kanten erzeugt. Die Reihenfolge der eingefügten Knoten ist $T_{n_kj0} \rightarrow O_{n_kj} \rightarrow b(O_{n_kj}) \rightarrow T_{n_kj1}$. Hierbei hat die Kante zwischen dem Bearbeitungsknoten und dem Batch-Bearbeitungsknoten das Gewicht $p(O_{n_kj}, b(O_{n_kj})) := 0$. Das Gewicht der Kante $(b(O_{n_kj}), T_{n_kj1})$ ist auf $p_{n_kj} + s_{lj}$ gesetzt. Handelt es sich bei der Maschine um keine Batch-Maschine bzw. ist die Batchgröße 1 wird auf die Anlage eines zusätzlichen Batch-Bearbeitungsknotens verzichtet.

Befindet sich das Los in einem Maschinenstocker, so erfolgt der Transport nur bis zum nächsten Maschinengruppenstocker. Aus diesem Grund wird nur der Transportknoten mit den entsprechenden Kanten für diesen Transport angelegt. Der Startknoten s wird in beiden Fällen mit dem ersten Transportknoten durch eine konjunktive Kante verbunden, die als Gewicht die aktuelle Zeit t enthält.

Das Los J_j wird gerade auf einem Fahrzeug transportiert: Abhängig vom Zielstocker werden hierbei ebenfalls zusätzliche Batch-Bearbeitungsknoten erzeugt, wenn es sich um eine Batch-Maschine handelt. Ist der Zielstocker ein Maschinenstocker, so wird das Los nach dem Transport auf dieser Maschine bearbeitet. Aus diesem Grund wird der entsprechende Batch-Bearbeitungsknoten mit den zugehörigen

Kanten angelegt. Da der Transport zum nachfolgenden Maschinengruppenstocker auch hier bereits bekannt ist, wird der entsprechende Transportknoten mit den notwendigen Kanten erzeugt. Im Falle einer Nicht-Batch-Maschine wird auch hier auf die Anlage des Batch-Bearbeitungsknotens verzichtet. Der Startknoten s wird mit dem Batch-Bearbeitungsknoten durch eine konjunktive Kante mit dem voraussichtlichen Transportendtermin als Gewicht verbunden.

Ist der Zielstocker ein Maschinengruppenstocker, so wird das Los von einer Maschine weg transportiert. Hier ist die Selektionsentscheidung noch nicht getroffen. Die konjunktive Kante zwischen dem Startknoten und dem Bearbeitungsknoten O_{n_kj} hat hier ebenfalls den voraussichtlichen Transportendtermin als Gewicht. Ein Batch-Bildungsknoten ist in beiden Fällen nicht notwendig, da die korrekten Verfügbarkeitszeiten durch die ausgehenden konjunktiven Kanten von s sichergestellt werden.

Das Los J_j wartet auf Aufnahme durch ein Fahrzeug: Das Los wurde hier bereits von einem Fahrzeug selektiert. Nun wartet das Los, bis es vom Fahrzeug aufgenommen wird. Die Behandlung ist dieselbe, als würde das Los gerade transportiert. Ist der Zielstocker ein Maschinenstocker, wird ein Batch-Bearbeitungsknoten mit Kanten angelegt. Wenn der Zielstocker ein Maschinengruppenstocker ist, wird kein weiterer Knoten angelegt. Die konjunktive Kante, die vom Startknoten wegführt, erhält den voraussichtlichen Transportendtermin als Gewicht.

4.4.3 Steuerung der Maschinen und Fahrzeuge

In Abbildung 4.8 ist dargestellt, wie sich das vorgeschlagene Verfahren auf Basis der SBH in das Produktionssteuerungssystem einfügt.

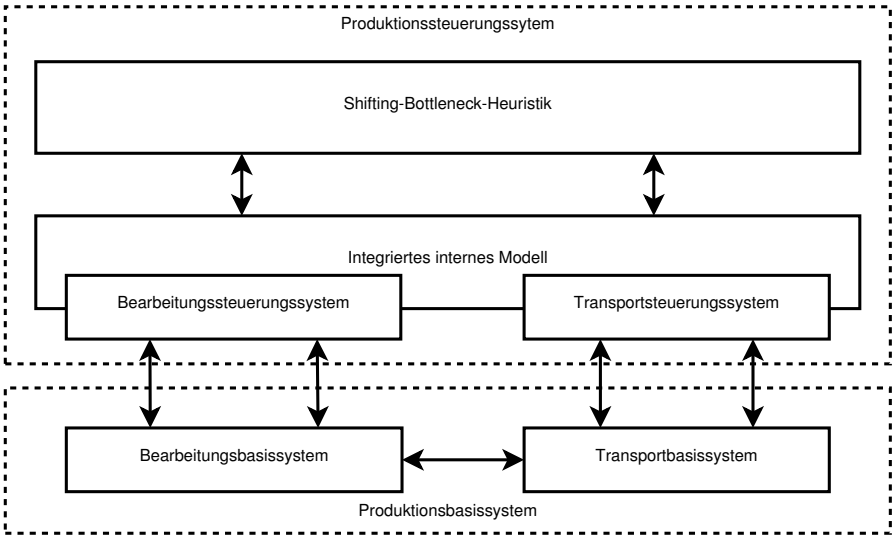

Abbildung 4.8: Einordnung der Shifting-Bottleneck-Heuristik in das Produktionssteuerungssystem

Für die integrierte Ablaufplanung ist es notwendig, dass das interne Modell sowohl Informationen aus dem Bearbeitungssystem als auch aus dem Transportsystem enthält

Nach dem Aufruf der SBH liegt eine Liste von Batches bzw. Losen für jede Ressource, das heißt alle Maschinen und Fahrzeuge, vor. Die Listen werden verwendet, um die einzelnen Ressourcen zu steuern. Wenn eine Ressource frei wird, so wird aus der zugehörigen Liste der erste Batch bzw. das erste Los gelesen. Sind alle Lose bereits in der Warteschlange vor der Ressource vorhanden, so werden sie selektiert. Sind noch nicht alle Lose vorhanden, wird solange gewartet, bis alle notwendigen Lose verfügbar sind.

Der Ansatz entspricht im Wesentlichen dem Steuerungsansatz aus Narasimhan u. a. (2005) und hat den Vorteil, dass Schwankungen in den zugrunde liegenden Prozess- und Transportzeiten den Plan nicht unzulässig werden lassen. Da keine Maschinenausfälle und andere Störungen des Bearbeitungsbasis- und Transportbasissystems betrachtet werden, ist eine Reparaturstrategie für die ermittelten Ablaufpläne nicht notwendig.

5 Effiziente Verfahren zur Lösung von Teilproblemen innerhalb der Shifting-Bottleneck-Heuristik

Innerhalb der SBH wird das gesamte Scheduling-Problem in kleinere Teilprobleme für jede Maschinengruppe zerlegt. In der Literatur werden unterschiedliche Teilproblemlöser für die SBH untersucht (vgl. u. a. Demirkol u. a. (1997), Mason u. a. (2002) und Mönch u. a. (2007)). Hierbei werden verschiedene Verfahren wie Branch-and-Bound, Prioritätsregeln und Genetische Algorithmen eingesetzt. Die Auswahl der Teilproblemlöser hat einen großen Einfluss auf die Leistungsfähigkeit der SBH. Da die einzelnen Teilproblemlöser gleichzeitig sehr häufig aufgerufen werden, ist es deswegen notwendig, effiziente Verfahren für die Teilprobleme einzusetzten, die sowohl einen geringen Rechenaufwand verursachen als auch eine gute Lösung für das Gesamtproblem liefern können. Es erfolgt zuerst, eine Beschreibung der in diesem Kapitel betrachteten Scheduling-Probleme, bevor im weiteren Verlauf des Kapitels auf passende Lösungverfahren eingegangen wird.

Teilproblemlöser auf Basis von Prioritätsregeln sind in der Lage, schnell einen Ablaufplan für das Teilproblem zu erzeugen. Es hat sich herausgestellt, dass zum Beispiel Teilproblemlöser auf Basis der Apparant-Tardiness-Cost-Regel (ATC-Regel) schnell sehr gute Ergebnisse innerhalb der SBH liefern (vgl. u. a. Mason u. a. (2002) sowie Mönch und Drießel (2005)). Aus diesem Grund wird zunächst ein Teilproblemlöser auf Basis von Prioritätsregeln vorgestellt.

Aufgrund der hohen Anzahl von Transportschritten und der starken wechselseitigen Abhängigkeit des Transportteilproblems mit den Maschinengruppenteilproblemen wird für das Transportsystem zusätzlich ein Ansatz auf Basis von VNS vorgeschlagen. Es hat sich gezeigt, dass VNS für ähnliche Optimierungsprobleme in der Lage ist, schnell gute Ergebnisse zu generieren (vgl. Almeder und Mönch (2009), Klemmt u. a. (2009) sowie Wang und Tang (2009)). Weiterhin zeichnet sich VNS im Allgemeinen durch einen geringeren Parametrisierungsaufwand im Vergleich zu anderen Metaheuristiken aus (vgl. hierzu auch Kapitel 3). Es werden verschiedene VNS-Ansätze entwickelt und mit Hilfe umfangreicher numerischer Experimente miteinander verglichen. Weiterhin wird ein exaktes Verfahren sowie ein Ansatz auf Basis von Genetischen Algorithmen für Vergleichszwecke kurz vorgestellt.

5.1 Betrachtete Scheduling-Probleme

Die Zerlegung im Rahmen der SBH führt sowohl zu Scheduling-Problemen für parallele identische Maschinen also auch zu einem Scheduling-Problem für das Transportsystem.

Wie bereits in Unterabschnitt 4.3.3 beschrieben, lässt sich das Transportteilproblem ebenfalls in ein Scheduling-Problem für parallele Maschinen umwandeln. Aus diesem Grund werden im Rahmen dieses Kapitels nur Probleme für parallele identische Maschinen vom Typ

$$Pm|r_j, s_{lj}, \text{batch}, \text{incompatible}, \text{prec}|\, TWT \tag{5.1}$$

betrachtet. Es werden n Lose auf identischen parallelen Maschinen einer Maschinenguppe (Pm) bearbeitet. Für die Lose müssen dynamische Ankunftszeiten r_j, reihenfolgeabhängige Umrüstzeiten (s_{lj}) und inkompatible Batch-Familien (batch, incompatible) berücksichtigt werden. Es wird davon ausgegangen, dass auf den Maschinen jedes Los nur einmal bearbeitet wird und somit auch nur einen Bearbeitungsschritt besitzt. Zwischen den Bearbeitungsschritten der verschiedenen Lose existieren verzögernde Vorrangbeziehungen, die als disjunktiver Graph SG abgebildet sind. Als Zielfunktion dient die totale gewichtete Verspätung $TWT \coloneqq \sum w_j \max(c_j - d_j, 0)$. Im Falle des Transportproblems verarbeiten die Maschinen nur Einzellose. Dieser Fall ist ein Spezialfall, bei dem $B = 1$ gilt. Hierdurch reduziert sich der betrachtete Problemtyp auf

$$Pm|r_j, s_{lj}, \text{prec}|\, TWT. \tag{5.2}$$

5.2 Verfahren auf Basis von Prioritätsregeln

Verfahren auf Basis von Prioritätsregeln können als zeitliche lineare Dekompositionsverfahren betrachtet werden. Das Problem wird dabei in eine Menge von Auswahlproblemen für einzelne Maschinen zerlegt. Innerhalb eines dieser Auswahlprobleme werden die Lose, die vor der Maschine warten, bewertet. Auf Basis dieser Bewertung wird entschieden, welche Lose zu einem Batch zusammengefasst und von der Maschine bearbeitet werden sollen. Das in diesem Abschnitt vorgestellte Verfahren ist für Scheduling-Probleme des Typs (5.1) geeignet und soll sowohl für die Maschinengruppenteilprobleme als auch für das Transportteilproblem eingesetzt werden.

Wie in Unterabschnitt 3.4.1 beschrieben, liefern Verfahren auf Basis von Prioritätsregeln in kurzer Zeit einen zulässigen Ablaufplan. Allerdings werden nur zeitlich und räumlich lokale Informationen berücksichtigt. Um dies zu behandeln, werden mehrere Ablaufpläne mit unterschiedlichen Prioritätsregeln generiert. In diesem Abschnitt wird zuerst ein Rahmenwerk für die Lösung von parallelen Maschinenproblemen auf Basis von Prioritätsregeln vorgestellt. Im Anschluss erfolgt eine Beschreibung, wie die einzelnen Ablaufpläne bewertet werden.

5.2.1 Rahmenwerk zur Lösung von parallelen Maschinenprobleme auf Basis von Prioritätsregeln

Die verzögernden Vorrangbeziehungen der Bearbeitungsschritte sind als disjunktiver Teilgraph $SG \coloneqq (V, E_c, E_d)$ mit der Menge der Knoten V, der Menge der konjunktiven Kanten E_c und der Menge der disjunktiven Kanten E_d, abgebildet. Jeder zu planende

Bearbeitungsschritt ist einem Knoten innerhalb von SG zugeordnet. Weiterhin existiert ein Startknoten sowie für jedes Los, das für das Teilproblem relevant ist, ein Endknoten. Die Knoten sind durch gerichtete Kanten aus der Menge E_c verbunden. Diese Kanten repräsentieren die verzögernden Vorrangbeziehungen, die sich durch Arbeitspläne bzw. vorige Teillösungen ergeben. Die ungerichteten Kanten der Menge E_d repräsentieren die zu treffenden Schedulingentscheidungen (vgl. auch Abbildung 4.6 aus Kapitel 4 für ein Beispiel).

Zunächst wird die als nächste verfügbare Maschine ermittelt. Das Maximum aus dem Verfügbarkeitszeitpunkt dieser Maschine und dem Minimum der Verfügbarkeitszeitpunkte aller Lose ist der **Entscheidungszeitpunkt** t. Die Menge $PRED(O_{ij}, SG)$ enthält alle Bearbeitungsschritte, die aufgrund von SG vor O_{ij} durchgeführt werden müssen. Es werden nun alle Bearbeitungsschritte berücksichtigt, die auf Basis von SG zum Zeitpunkt t verfügbar sind und keine Vorgänger haben, das heißt $PRED(O_{ij}, SG) = \emptyset$. Da hierbei verschiedene Losfamilien berücksichtigt werden müssen, ist eine entsprechende Gruppierung notwendig. Mit

$$M(f,t) \coloneqq \{O_{ij} \mid r_{ij} \leq t \wedge PRED(O_{ij}, SG) = \emptyset \wedge J_j \text{ gehört zur Losfamilie } f\} \quad (5.3)$$

werden die Bearbeitungsschritte bezeichnet, die zur Losfamilie f gehören und zum Zeitpunkt t verfügbar sind. Jeder Bearbeitungsschritt in $M(f,t)$ wird nun mittels eines **Prioritätsindexes** $I(O_{ij}, t)$ bewertet. Die Mengen $M(f,t)$ werden nun so sortiert, dass die Bearbeitungsschritte mit der höchsten Priorität am Anfang der resultierenden Liste $L(f,t)$ stehen.

Es wird nun der Bearbeitungsschritt aus allen Listen $L(f,t)$ ausgewählt, der die höchste Priorität hat. Es wird nun zunächst ein Batch mit dem Los des Bearbeitungsschritts O_{ij} mit der Batch-Größe eins gebildet. Danach wird dieser Batch durch die Lose der nächsten Bearbeitungsschritte aus der zugehörigen Liste $L(f,t)$ aufgefüllt. Die Auffüllung erfolgt, solange die Batch-Größe kleiner oder gleich der maximalen Batch-Größe ist oder das Ende der Liste erreicht ist.

Für jede Maschine existiert ein Ablaufplan als Liste von Batches $S(m)$, der die Abarbeitungsreihenfolge der Batches auf der Maschine M_m repräsentiert (S = engl. *Schedule*). Der ermittelte Batch wird an das Ende der Liste $S(m)$ der aktuell ermittelten Maschine M_m angefügt. Mit r_b wird der Zeitpunkt bezeichnet, an dem alle Lose des Batches verfügbar sind. Der neue Verfügbarkeitszeitpunkt der Maschine wird auf $a_m \coloneqq \max(a_m, r_b) + s_{bl} + p_b$ gesetzt. Hierbei wird mit s_{bl} die notwendige Rüstzeit bezeichnet und mit p_b die Bearbeitungszeit des Batches auf der Maschine. Danach müssen die Verfügbarkeitszeitpunkte der Bearbeitungsschritte angepasst werden, die direkter Nachfolger der in dem Batch b enthaltenen Bearbeitungsschritte sind. Der neue Verfügbarkeitszeitpunkt eines Nachfolgers wird auf $r_{ij} \coloneqq \max(r_{ij}, a_m + t_{uv}^{ij})$ aktualisiert. Mit t_{uv}^{ij} wird hierbei die Verzögerungszeit zwischen dem Vorgänger O_{uv} und dem Nachfolger O_{ij} bezeichnet, die sich aus SG ergibt. Danach werden die Knoten der in dem Batch enthaltenen Bearbeitungsschritte aus SG entfernt. Algorithmus 5.1 fasst das Rahmenwerk zur Erzeugung einer Ablaufplans für eine Maschinengruppe auf Basis von Prioritätsregeln zusammen.

Algorithmus 5.1 : Erzeugung eines Ablaufplans auf Basis einer Prioritätsregel

Daten : paralleles Maschinenproblem p
Ergebnis : Ablaufplan s für das Teilproblem p

solange $G \neq \emptyset$ **führe aus**
  $M_m \leftarrow$ Ermittle die Maschine mit dem frühesten Verfügbarkeitszeitpunkt a_m;
  $M(f,t) \leftarrow \{O_{ij} \mid r_{ij} \leq t \wedge PRED(O_{ij}, SG) = \emptyset \wedge J_j$gehört zur Losfamilie $f\}$;
  Berechne $I(O_{ij}, t)\ \forall O_{ij} \in M(f,t)$ und erzeuge $L(f,t)$;
  Erzeuge einen Batch b auf Basis der sortierten Liste $L(f,t)$;
  Füge den Batch b dem Ablaufplan $S(m)$ der Maschine M_m hinzu;
  $a_m \leftarrow \max(a_m, r_b) + s_{bl} + p_b$;
  Für nachfolgende Bearbeitungsschritte setze $r_{ij} \coloneqq \max(r_{ij}, a_m + t_{uv}^{ij})$;
  Entferne die Knoten der Bearbeitungsschritte des Batches b aus SG;
Ende

Im Folgenden soll ein Beispiel das Vorgehen verdeutlichen. Ausgehend von dem Beispiel aus Kapitel 4 wird ein Ablaufplan für die Maschinengruppe D auf Basis einer Schlupfzeitregel mit $I(O_{ij}, t) \coloneqq d_{ij} - p_{ij} - t$ mit $d_{ij} = min\{d_{ij}^k | k = 1, \ldots, n\}$ erzeugt. Hierbei wird der Bearbeitungsschritt mit dem kleinsten Wert für $I(O_{ij}, t)$ gewählt. Wie bereits beschrieben, besteht die Maschinengruppe D aus den zwei Maschinen $M_{1,D}$ und $M_{2,D}$, die beide zum Zeitpunkt 0 verfügbar sind. Die Maschinen der Maschinengruppe sind in diesem Beispiel keine Batch-Maschinen, das heißt das $B = 1$ gilt. Weiterhin wird davon ausgegangen, dass es nur eine Losfamilie gibt. Tabelle 5.1 zeigt die frühesten Verfügbarkeitszeitpunkte r_{ij}, die Bearbeitungszeiten p_{ij}, die zusätzliche Verzögerungszeit t_{uv}^{ij} mit den zugehörigen Nachfolgern sowie die einzelnen gewünschten Fertigstellungstermine d_{ij}^k, wie sie sich aus dem Graphen in Abbildung 4.6 ergeben. Weiterhin sind noch die Bearbeitungszeiten aus Tabelle 4.1 mit angeben.

Innerhalb der ersten Iteration sind beide Maschinen zum Zeitpunkt 0 verfügbar. Der Teilgraph SG beinhaltet Knoten für die Bearbeitungsschritte O_{22}, O_{33}, O_{42}. Aufgrund der Vorrangbeziehung zwischen O_{22} und O_{42} wird O_{42} zunächst nicht berücksichtigt. Die Maschine $M_{1,D}$ wird gewählt und der aktuelle Entscheidungszeitpunkt ist aufgrund der Verfügbarkeitszeitpunkte der Bearbeitungsschritte $t = min(r_{22}, r_{33}) = 4$. Die Menge $M(f,t)$ enthält entsprechend nur O_{22} und O_{33}. Im nächsten Schritt erfolgt eine Bewertung auf Basis der Schlupfzeitregel. Für O_{22} ergibt sich ein Wert von $I(O_{22}, t) = 7 - 5 - 4 = -2$ und für O_{33} ein Wert von $I(O_{33}, t) = 16 - 7 - 8 = 1$. Auf Basis der daraus resultierenden Liste $L(f,t) \coloneqq (O_{22}, O_{33})$ kann der Ablaufplan von $M_{1,D}$ wie folgt angepasst werden: $S(M_{1,D}) \coloneqq (O_{22})$. Der neue Verfügbarkeitszeitpunkt für die Maschine beträgt $a_{M_{1,D}} \coloneqq 9$. Am frühesten Verfügbarkeitszeitpunkt von O_{42} ändert sich nichts, da keine Wartezeit für O_{22} aufgetreten ist. Der Knoten für den Bearbeitungsschritte O_{22} wird am Ende der Iteration aus dem Teilgraphen SG entfernt.

In der zweite Iteration enthält der Teilgraph SG nur noch die Knoten für O_{33} und O_{42}. Die nächste verfügbare Maschine ist nun $M_{2,D}$ und der Entscheidungszeitpunkt ist $t = 8$. Aufgrund der Prioritätswerte $I(O_{33}, t) = 16 - 7 - 8 = 1$ und $I(O_{42}, t) = 18 - 5 - 8 = 5$ wird

der Ablaufplan von $M_{2,D}$ auf $S(M_{2,D}) \coloneqq (O_{33})$ gesetzt. Der neue Verfügbarkeitszeitpunkt beträgt $a_{M_{2,D}} \coloneqq 15$. Der Knoten für den Bearbeitungsschritt O_{33} wird wieder aus dem Teilgraphen SG entfernt.

In der dritten und letzten Iteration ist die früheste verfügbare Maschine wieder die Maschine $M_{1,D}$, der Entscheidungszeitpunkt ist $t = 15$. Der letzte verbleibende Bearbeitungsschritt wird an den zugehörigen Ablaufplan angehängt: $S(M_{1,D}) \coloneqq (O_{22}, O_{42})$. Der zugehörige Knoten wird ebenfalls wieder aus dem Teilgraphen SG entfernt. Da der Teilgraph nun keine weiteren Bearbeitungsknoten enthält, terminiert das Verfahren.

Tabelle 5.1: Teilproblem für Maschinengruppe D aus Abbildung 4.6

Bearbeitungsschritt	r_{ij}	p_{ij}	t_{uv}^{ij}	d_{ij}^{2}	d_{ij}^{3}	Nachfolger
O_{22}	4	5	6	7	∞	O_{42}
O_{33}	8	7	0	∞	16	
O_{42}	15	5	0	18	∞	

5.2.2 Bewertung der generierten Ablaufpläne

Wie bereits beschrieben, werden mit Hilfe von Algorithmus 5.1 für das Teilproblem typischerweise mehrere Ablaufpläne auf Basis von unterschiedlichen Prioritätsregeln erzeugt. Ein Ablaufplan liegt zunächst als Liste von Batches $S(m)$ für jede einzelne Maschine M_m vor. Zur Bewertung wird dieser Ablaufplan in SG eingefügt und alle disjunktiven Kanten durch konjunktive Kanten ersetzt (vgl. auch Abschnitt 4.2). Nachdem der Ablaufplan in SG eingebracht wurde, können die Fertigstellungszeitpunkte der Lose berechnet werden.

Die Bewertung der Ablaufpläne erfolgt auf Basis derselben Leistungsmaße, die auch für die Bestimmung der Einplanungsreihenfolge der Teilproblemlösungen in den disjunktiven Graphen G für das Gesamtproblem verwendet werden (siehe Unterabschnitt 4.3.4).

Das Hauptunterscheidungskriterium ist die TWT. Von den erzeugten Ablaufplänen wird derjenige mit der geringeren TWT gewählt. Falls die Entscheidung aufgrund der TWT nicht eindeutig ist, das heißt es bleiben mehrere Ablaufpläne mit gleicher TWT übrig, so erfolgt die Bewertung anhand der maximalen gewichteten Terminabweichung $WL_{\max}$. Hier wird der Ablaufplan mit der kleinsten maximalen gewichteten Terminabweichung gewählt. Sollte auch danach noch keine eindeutige Entscheidung möglich sein, wird der Ablaufplan gewählt, der die kleinste Zykluszeit $C_{\max}$ aufweist.

Die Bewertung muss im Zusammenhang mit der Bewertung eines Teilproblems innerhalb der SBH aus Unterabschnitt 4.3.4 betrachtet werden. Während innerhalb des Teilproblems eine Verbesserung der oben genannten Leistungsmaße erfolgt, wird in Unterabschnitt 4.3.4 das kritischste Teilproblem gesucht. Aus diesem Grund wird das Teilproblem mit den schlechtesten dieser Leistungsmaße gewählt.

5.3 Verfahren zur Lösung des Transportteilproblems auf Basis von VNS

Innerhalb dieses Abschnitts werden Scheduling-Probleme des Typs (5.2) betrachtet. Das Verfahren soll für das Transportteilproblem eingesetzt werden. Aus diesem Grund müssen Batch-Maschinen nicht berücksichtigt werden. Jedoch müssen für jeden Bearbeitungsschritt der Maschinengruppenteilprobleme innerhalb des Transportteilproblems zwei Transportschritte berücksichtigt werden. Damit enthält das Transportteilproblem doppelt so viele Prozessschritte wie alle Maschinengruppenteilprobleme zusammen. Ein weiterer Unterschied zu den Maschinengruppenteilproblemen ist die Anzahl der Vorrangbeziehungen. Da bereits alle Teillösungen für die Maschinengruppen in G eingefügt wurden, existieren viel mehr Vorrangbeziehungen innerhalb des Transportteilproblems als innerhalb der einzelnen Maschinengruppenteilprobleme.

Diese Unterschiede sorgen dafür, dass der Aufwand zum Erzeugen eines Ablaufplans für das Transportteilproblem deutlich größer ist als für die einzelnen Maschinengruppenteilprobleme. Andererseits hat das Transportteilproblem aufgrund der Größe und der starken Abhängigkeiten unter den Transportschritte einen relativ großen Einfluss auf das Gesamtproblem, obwohl die einzelnen Transportzeiten sehr kurz sind. Dies muss innerhalb des Lösungsverfahrens geeignet berücksichtigt werden.

Auf Basis früherer Arbeiten (vgl. u. a. Mönch und Drießel (2005) und Mönch u. a. (2007)) ist bekannt, dass die Verwendung aufwändigerer Teilproblemlöser für kritische Teilprobleme zu guten Ergebnissen führt. Der Grund hierfür liegt darin, dass bei kritische Teilproblemen der Einfluss auf das Gesamtproblem sehr stark ist und so auch kleine Verbesserungen in den Teilproblemen zu Verbesserungen für das Gesamtproblem führen.

Ansätze auf Basis von VNS werden in der neueren Literatur für viele schwere Optimierungsprobleme eingesetzt und haben zu guten Ergebnissen geführt. Aus diesem Grund wird ein Ansatz auf Basis von VNS für Maschinenbelegungsprobleme vom Typ (5.2) vorgeschlagen.

5.3.1 Diskussion relevanter Literatur

An dieser Stelle wird auf relevante Literatur mit Bezug auf Maschinenbelegungsprobleme mit reihenfolgeabhängigen Umrüstzeiten, Berücksichtigung von frühesten Verfügbarkeitszeitpunkten, Vorrangbeziehungen eingegangen. Desweiteren werden VNS-Ansätze für Scheduling-Probleme vorgestellt. Maschinenbelegungsprobleme mit reihenfolgeabhängigen Umrüstzeiten werden intensiv in der Literatur untersucht. Für einen aktuellen Überblick über den Forschungsstand sei auf Allahverdi u. a. (2008) verwiesen.

Für $Pm|r_j, s_{lj}|L_{\max}$ wird in Ovacik und Uzsoy (1995) ein zeitbasierter Dekompositionsansatz vorgeschlagen. Mit $L_{\max} \coloneqq \max\{L_j|j = 1, \ldots, n\}$ wird hierbei die maximale Terminabweichung für alle Lose bezeichnet. Es werden Prioritätsregeln verwendet, um die Lose einzelnen Zeitfenstern zuzuweisen. Danach werden verschiedene Branch-and-Bound-Verfahren und lokale Suchverfahren angewendet, um die Lose, die innerhalb eines Zeitfensters eintreffen, zu planen. Das Problem $Pm|r_j|L_{\max}$ wird in Dessouky (1998) diskutiert. Ein Branch-and-Bound-Verfahren wird zur Lösung vorgeschlagen. Mittels

einfacher Heuristiken werden dabei obere Schranken ermittelt.

Lee und Pinedo (1997) schlagen eine Modifikation der ATC-Regel (vgl. Vepsalainen und Morton (1987)) vor, die reihenfolgeabhängige Umrüstzeiten berücksichtigt (ATCS = engl. *Apparent Tardiness Cost with Setup Times*). Es werden verschiedene Ansätze vorgeschlagen, um, abhängig von der jeweiligen Situation, geeignete Parameter für die ATCS-Regel zu finden. In Park u. a. (2000) werden die Parameter von ATCS mit Hilfe eines neuronalen Netzes bestimmt.

Eine Erweiterung für ATCS zur Berücksichtigung von Verfügbarkeitsinformationen wird in Pfund u. a. (2008b) vorgeschlagen. Die Parameter werden mit Hilfe einer Gittersuche sowie eines Regressionsansatzes bestimmt. Liao und Juan (2007) schlagen einen ACO-Ansatz zur Lösung von $1|s_{lj}|TWT$ vor. Innerhalb des Ansatzes wird die ATCS-Regel verwendet, um heuristische Informationen für die einzelnen Lossequenzen zu berechnen. Der vorgeschlagene ACO-Ansatz liefert bessere Ergebnisse als die ATCS-Regel allein. Der Ansatz wird durch Anghinolfi und Paolucci (2008) verbessert. Anghinolfi und Paolucci (2009) schlagen außerdem einen Particle-Schwarm-Ansatz für dasselbe Problem vor.

In Lou u. a. (2006) werden Dominanzregeln für $1|s_{lj}|TT$ mit $TT := \sum T_j$ und $1|s_{lj}|\max\{T_j\}$ vorgestellt. Diese können verwendet werden, um Branch-and-Bound-Ansätze zu beschleunigen. Ein Ansatz auf Basis von Greedy Randomized Adaptive Search Procedure (GRASP) wird von Gupta und Smith (2006) für $1|s_{lj}|TWT$ vorgeschlagen. Unterschiedliche Metaheuristiken (u. a. Genetische Algorithmen, Tabu-Suche und Simulated Annealing) werden in Lin und Ying (2007) für das Problem $1|s_{lj}|TWT$ untersucht. Kim u. a. (2006) schlägt eine Tabu-Suche für $Pm|r_j, s_{lj}|TT$ vor, allerdings wurden keine Vorrangbeziehungen berücksichtigt. Das Problem $Pm|r_j, q_j|C_{\max}$ wird durch Gharbi und Haouari (2002) untersucht. Hierbei steht q_j für die Transportzeit von einem Los J_j, die benötigt wird, um das Los nach der Bearbeitung auf einer Maschine aus dem System zu transportieren. Zur Lösung werden Branch-and-Bound-Ansätze eingesetzt.

Neben den bereits in Kapitel 4 erwähnten Arbeiten existieren noch weitere Arbeiten mit Bezug auf Scheduling von parallelen Maschinen mit Vorrangbeziehungen. In Hermann u. a. (1997) wird für die Minimierung der Zykluszeit $C_{\max}$ von Losen auf parallelen Maschinen mit Vorrangbeziehungen mit einfachen Zuordnungsheuristiken und einem Simulated-Annealing-Ansatz gelöst. Das Scheduling-Problem $Pm|prec, p_j = 1|C_{\max}$ wird von Aho und Mäkinen (2006) untersucht. Es wird ein exakter Algorithmus vorgeschlagen, der die Lösung in polynomieller Zeit liefert, wenn die Größe des Vorranggraphs durch den maximale Grad und den Durchmesser begrenzt sowie die Anzahl der Maschinen m fest ist. List-Scheduling-Heuristiken zur Lösung eines Blockmontageproblems in einer Werft werden in Hu u. a. (2010) vorgestellt. Dieses Problem kann als paralleles Maschinenbelegungsproblem mit Vorrangbeziehungen, Maschinenzuordnungsrestriktionen und $C_{\max}$-Zielfunktion modelliert werden. Das Problem $Pm|prec, s_{lj}|C_{\max}$ wird in Hurink und Knust (2001a) diskutiert. Es wird gezeigt, dass im Unterschied zu verschiedenen anderen Scheduling-Problemen in dieser Situation eine Menge von dominanten Ablaufplänen nicht effizient durch prioritätsbasierte Ansätze berechnet werden kann. Eine Menge von Ablaufplänen wird hierbei dominant genannt, wenn sie mindestens einen optimalen Ablaufplan enthält.

Wang und Tang (2009) beschreiben einen populationsbasierten VNS-Ansatz für $1||TWT$.

Rocha u. a. (2007) schlagen einen VNS- und GRASP-Ansatz für das parallele Maschinenbelegungsproblem $Pm|s_{lj}|C_{\max}+TWT$ vor. Der VNS-Ansatz liefert hierbei besonders bei größeren Problemen bessere Ergebnisse als der GRASP-Ansatz. Liao und Cheng (2007) wenden VNS auf ein Einmaschinenproblem mit einem gemeinsamen geplanten Aussteuertermin für alle Lose und gewichteten Verfrühungs- und Verspätungskosten an. Almeder und Mönch (2009) sowie Klemmt u. a. (2009) schlagen VNS-Ansätze für Scheduling-Probleme mit identischen parallelen Batch-Maschinen und sowohl mit als auch ohne Berücksichtigung von Verfügbarkeitszeiten der Lose vor. Anghinolfi und Paolucci (2007) verwenden VNS in Kombination mit anderen Metaheuristiken für das Problem $Pm|s_{lj}|TWT$. Von Sevkli und Aydin (2006) wird ein VNS-Ansatz für das Problem $Jm||C_{\max}$ vorgeschlagen. Probleme des Typs (5.2) wurde bisher allerdings noch nicht behandelt.

5.3.2 Rahmenwerk auf Basis von VNS

Wie bereits beschrieben, ist VNS ein relativ neues lokales Suchverfahren. Die grundsätzliche Idee besteht in der Verwendung unterschiedlicher Nachbarschaftsstrukturen, um den Lösungsraum zu durchsuchen. Die Leistungsfähigkeit von VNS hängt im Wesentlichen von

- der Qualität der Initiallösung,
- den verwendeten Nachbarschaftsstrukturen,
- den verwendeten lokalen Suchverfahren und
- der Reihenfolge, in der die Nachbarschaften verwendet werden,

ab. Im Unterabschnitt 5.3.3 erfolgt die Beschreibung, wie die Initiallösung ermittelt werden kann. Die verwendeten Nachbarschaftsstrukturen werden im Unterabschnitt 5.3.4 vorgestellt. Innerhalb der Literatur werden Vergleiche zwischen verschiedenen VNS-Ansätzen selten diskutiert. Es existieren jedoch, wie in Unterabschnitt 5.5.2 zu sehen ist, teilweise beträchtliche Unterschiede zwischen den einzelnen VNS-Varianten. Ein grundlegendes Verständnis der einzelnen Varianten und ihrer Leistungsfähigkeit ist wichtig für das Design der Nachbarschaftsstrukturen und der Auswahl der verwendeten lokalen Suchverfahren.

5.3.3 Ermittlung der Initiallösung

Zur Erzeugung der Initiallösung wird das Rahmenwerk auf Basis von Prioritätsregeln aus Abschnitt 5.2 eingesetzt.

Bei gleichbleibender Laufzeit für VNS führt eine bessere Initiallösung im Allgemeinen zu einem besseren Gesamtergebnis, da bei einer schlechteren Initiallösung zusätzlich Zeit benötigt wird, bis dieselbe Lösungsgüte der besseren Initiallösung erreicht wird. Aus diesem Grund ist es sinnvoll, zur Erzeugung der Initiallösung bessere und damit

gegebenenfalls langsamere Verfahren einzusetzen. Es ist bekannt, dass die Klasse der ATC-basierten Prioritätsregeln für das Leistungsmaß der totalen gewichteten Verspätung im allgemeinen bessere Ergebnisse liefern als viele andere Prioritätsregeln (siehe hierzu auch Pfund u. a. (2008b)). Weiterhin haben Teilproblemlöser auf ATC-Basis auch innerhalb der Shifting-Bottleneck-Heuristk bereits gute Ergebnisse geliefert (vgl. Pinedo und Singer (1999), Mönch und Drießel (2005) und Mönch u. a. (2007)).

An dieser Stelle wird der ATC-Index mit Rüstzeiten und Verfügbarkeitsinformationen ATCSR (engl. *ATC with Setup and Release Times*) aus Pfund u. a. (2008b) zur Ermittlung des Prioritätsindexes verwendet. Der ATCSR-Index für ein Los J_j mit dem durchzuführenden Bearbeitungsschritt O_{ij} wird wie folgt berechnet:

$$I(O_{ij},t) := \frac{w_j}{p_{ij}} e^{-\frac{\max(d_j - p_{ij} - \max(r_j,t),0)}{k_1 \bar{p}} - \frac{s_{lj}}{k_2 \bar{s}} - \frac{\max(r_j - t,0)}{k_3 \bar{p}}}. \tag{5.4}$$

Mit w_j wird das Gewicht des Loses bezeichnet, p_{ij} repräsentiert die Bearbeitungszeit für den aktuellen Bearbeitungsschritt des Loses und s_{lj} stellt die notwendige Rüstzeit dar, um die Maschine vom Los J_l auf das Los J_j umzurüsten. Mit $\bar{p}$ wird die durchschnittliche Bearbeitungszeit und mit $\bar{s}$ die durchschnittliche Rüstzeit aller Lose vor der Maschine bezeichnet. Mit r_j wird der Einsteuertermin und mit d_j der geplante Aussteuertermin des Loses J_j bezeichnet.

Der Prioritätsindex (5.4) ist ein zusammengesetzter Prioritätsindex, um die Dringlichkeit des betrachteten Bearbeitungsschritts zu berechnen. Je höher der berechnete Prioritätsindex, umso dringlicher ist die Abarbeitung des betrachteten Bearbeitungsschritts. Die einzelnen Terme werden mit Hilfe der Skalierungsparameter k_1, k_2 und k_3 gewichtet. Die Skalierung des Schlupfterms erfolgt mit Hilfe des Parameters k_1, der Rüstzeitterm wird mit Hilfe des Parameters k_2 skaliert und der Verfügbarkeitsterm mit Hilfe des Parameters k_3.

Mit Hilfe der Skalierungsparameter k_1, k_2 und k_3 erfolgt die Anpassung der Prioritätsregel an unterschiedliche Umgebungssituationen. Abhängig vom der jeweiligen Problem sind unterschiedliche Parameterkombination sinnvoll. Die Wahl der Skalierungsparameter hat dadurch einen großen Einfluss auf die Qualität des berechneten Ablaufplans. Aus diesem Grund ist es häufig üblich, mit Hilfe einer Gittersuche zunächst eine Menge von zulässigen Ablaufplänen zu bestimmen. Aus dieser Menge wird dann der beste Ablaufplan ermittelt (vgl. u. a. Pinedo und Singer (1999) sowie Pfund u. a. (2008b)). Die Bewertung der erzeugten Ablaufpläne erfolgt dabei wie in Unterabschnitt 5.2.2 beschrieben.

Vier unterschiedliche Varianten des Prioritätsindex (5.4) werden in diesem Abschnitt untersucht. Sie unterscheiden sich im Wesentlichen dadurch, wie die Skalierungsparameter $(k_1, k_2, k_3) \in [0{,}0; 7{,}2]^3$ gewählt werden. Das Raster für die erste Variante ATCSR-1 ist sehr fein und hat eine kleine Schrittweite. Es werden 22 Werte für k_1, elf Werte für k_2 und 13 Werte für k_3 betrachtet. Die Werte basieren auf den Werten aus Pfund u. a. (2008b):

- $k_1 \in \{0{,}2; 0{,}6; 0{,}8; 1{,}0; 1{,}2; 1{,}4; 1{,}6; 1{,}8; 2{,}0; \ldots; 5{,}2; 5{,}6; 6{,}0; 6{,}4; 6{,}8; 7{,}2\}$,
- $k_2 \in \{0{,}1; 0{,}3; 0{,}5; 0{,}7; 0{,}9; 1{,}1; 1{,}3; 1{,}5; 1{,}7; 1{,}9; 2{,}1\}$ sowie

- $k_3 \in \{0{,}001; 0{,}002\,5; 0{,}004; 0{,}005; 0{,}025; 0{,}04; 0{,}05; 0{,}25; 0{,}4; 0{,}6; 0{,}8; 1{,}0; 1{,}2\}$.

Die Verwendung eines solchen engmaschigen Gitters ist sehr zeitaufwändig. Insgesamt werden 3 146 Ablaufpläne erzeugt und bewertet. Aus diesem Grund wird zusätzlich ein gröberes Raster mit sieben Werten für k_1, vier Werten für k_2 und fünf Werten für k_3 betrachtet. Die Werte der Skalierungsparameter, die sich durch die entsprechende Ausdünnung des Raster von ATCSR-1 ergeben, sind wie folgt gewählt:

- $k_1 \in \{0{,}2; 1{,}0; 1{,}6; 2{,}4; 3{,}6; 4{,}8; 6{,}0\}$,
- $k_2 \in \{0{,}1; 0{,}7; 1{,}3; 1{,}9\}$ und
- $k_3 \in \{0{,}001; 0{,}005; 0{,}05; 0{,}6; 1{,}2\}$.

Diese Variante wird im weiteren Verlauf mit ATCSR-2 bezeichnet. Der dritte Ansatz berücksichtigt den Verfügbarkeitsterm bei der Berechnung des Prioritätsindex (5.4) nicht. Stattdessen wird nur der ATCS-Index aus Lee und Pinedo (1997) verwendet. Der ATCS-Index wird wie folgt berechnet:

$$I(O_{ij}, t) := \frac{w_k}{p_{ij}} e^{-\frac{\max(d_j - p_{ij} - \max(r_j, t), 0)}{k_1 \bar{p}} - \frac{s_{lj}}{k_2 \bar{s}}}. \tag{5.5}$$

Die hier verwendeten Gitterpunkte für (5.5) werden ebenfalls in einer früheren Arbeit (vgl. Mönch und Drießel (2005)) verwendet:

- $k_1 \in \{0{,}01; 0{,}15; 0{,}5; 1{,}0; 1{,}5\}$,
- $k_2 \in \{0{,}1; 0{,}7; 1{,}3; 1{,}9\}$.

In diesem Fall werden nur noch 20 verschiedene Ablaufpläne betrachtet. In der letzten betrachteten Variante wird der Rüstzeitterm im Prioritätsindex (5.4) nicht berücksichtigt. Der Index wird mit ATCR bezeichnet und wird wie folgt berechnet:

$$I(O_{ij}, t) := \frac{w_k}{p_{ij}} e^{-\frac{\max(d_j - p_{ij} - \max(r_j, t), 0)}{k_1 \bar{p}} - \frac{\max(r_j - t, 0)}{k_3 \bar{p}}}. \tag{5.6}$$

Wie im Falle von ATCR werden 20 Ablaufpläne berechnet. Hier werden folgende Gitterpunkte verwendet:

- $k_1 \in \{0{,}01; 0{,}15; 0{,}5; 1{,}0; 1{,}5\}$,
- $k_3 \in \{0{,}01; 0{,}1; 0{,}5; 1{,}0\}$.

5.3.4 Verwendete Nachbarschaftsstrukturen

Ein Ablaufplan für Probleme des Typs (5.2) wird als Liste von Bearbeitungsschritten I_i für jede Maschine $i = 1, 2, \ldots, m$ der betrachteten Maschinengruppe dargestellt. Die Anzahl der Bearbeitungsschritte innerhalb der Liste $I_i = 0, \ldots, n_{i,max} - 1$ wird mit $n_{i,\max}$ bezeichnet. Die Menge aller zulässigen Ablaufpläne für ein Problem des Typs (5.2) sei mit S bezeichnet. Ausgehend von einem zulässigen Ablaufplan $x = (I_1, I_2, \ldots, I_m) \in S$ können verschiedene Nachbarschaftsstrukturen $N_k(x) \subset S$ definiert werden. Wie in Unterabschnitt 3.4.3 beschrieben, werden die Nachbarschaftsstrukturen durch Züge definiert. Ein Zug gibt an, wie der aktuelle Ablaufplan verändert wird, um einen neuen zulässigen Ablaufplan zu erhalten. Die Bewertung zweier Ablaufpläne erfolgt mit der in Unterabschnitt 5.2.2 beschrieben Strategie. Es werden vier Basisnachbarschaftsstrukturen mit den Parametern r und l betrachtet. Innerhalb der Nachbarschaftsstrukturen sei p die aktuelle Position eines zufällig gewählten Bearbeitungsschritts und i die dem Bearbeitungsschritt zugeteilte Maschine. Die Nachbarschaftsstrukturen sind wie folgt definiert:

TransferOnOneMachine(r, l): Für jeden Bearbeitungsschritt gehe wie folgt vor: Alle zulässigen Transfers dieses Bearbeitungsschritts von seiner aktuellen Position auf eine Position innerhalb des Intervalls $[\max(p - r, 0), \min(p + r, n_{i,\max})]$ auf der Maschine i werden betrachtet. Bei dem Transfer wird der Bearbeitungsschritt an der jeweiligen Position eingefügt. Die anderen Bearbeitungsschritte werden entsprechend um eine Position nach hinten oder nach vorn verschoben. Die Prozedur wird l mal wiederholt.

SwapOnOneMachine(r, l): Für jeden Bearbeitungsschritt gehe wie folgt vor: Alle zulässigen Vertauschungen dieses Bearbeitungsschritts mit Bearbeitungsschritten, die eine Position innerhalb des Intervalls $[\max(p - r, 0), \min(p + r, n_{i,\max} - 1)]$ auf der Maschine i haben, werden betrachtet. Die Prozedur wird l mal wiederholt.

TransferAcrossMachines(r, l): Für jeden Bearbeitungsschritt gehe wie folgt vor: Alle zulässigen Transfers dieses Bearbeitungsschritts von seiner aktuellen Position auf eine Position innerhalb des Intervalls $[\max(p - r, 0), \min(p + r, n_{k,\max})]$ auf alle Maschinen $k \neq i$ werden betrachtet. Im Falle von $n_{k,\max} < p$ wird der Bearbeitungsschritt an Position $n_{k,\max} + 1$ eingefügt. Die Prozedur wird l mal wiederholt.

SwapAcrossMachines(r, l): Für jeden Bearbeitungsschritt gehe wie folgt vor: Alle zulässigen Vertauschungen des Bearbeitungsschritts mit Bearbeitungsschritten, die eine Position innerhalb des Intervalls $[\max(p - r, 0), \min(p + r, n_{k,\max} - 1)]$ auf einer Maschine $k \neq i$ haben, werden betrachtet. Im Falle von $n_{k,\max} < p$ wird keine Vertauschung ausgeführt. Die Prozedur wird l mal wiederholt.

Diese Nachbarschaftstrukturen berücksichtigen den Aspekt, dass günstige Transferpositionen bzw. günstige Tauschpartner mit hoher Wahrscheinlichkeit nahe beieinander liegen. Der Grund hierfür liegt zum einen darin, dass der Initiallöser vermutlich bereits eine gute Vorsortierung der Bearbeitungsschritte vorgenommen hat. Zum anderen ist die

mögliche Position der einzelnen Bearbeitungsschritte aufgrund der Verfügbarkeitszeiten sowie der Vorrangbeziehungen eingeschränkt. Der Parameter r bildet diese Umstände ab.

Die Größe der Nachbarschaften auf Basis der ersten und der zweiten Nachbarschaftsstruktur ist $O(2rln)$, die auf Basis der zwei nachfolgenden Nachbarschaftsstrukturen $O(2rln^2)$. Hierbei wird mit n die Anzahl der Bearbeitungsschritte bezeichnet. Wie in Abschnitt 4.3 beschrieben, dienen die verzögernden Vorrangbeziehungen dazu, die Kreisfreiheit des disjunktiven Graphen G für das Gesamtproblem sicherzustellen. Aus diesem Grund müssen an dieser Stelle die Vorrangbeziehungen zwischen den Bearbeitungsschritten innerhalb der Transfers bzw. Vertauschungen so berücksichtigt werden, dass der neue Ablaufplan weiterhin zulässig ist. Der neue Ablaufplan ist dann zulässig, wenn der aktuelle Bearbeitungsschritt nicht vor einen direkten oder indirekten Vorgänger bzw. nach einen direkten oder indirekten Nachfolger verschoben wurde. Bei einem gegebenen Ablaufplan kann für jeden Bearbeitungsschritt und jede Maschine eine **obere und untere Grenze** ermittelt werden. Bei Transfer innerhalb dieser Grenzen ist sichergestellt, dass der neue Ablaufplan zulässig bleibt, da der Bearbeitungsschritt nicht hinter einen direkten oder indirekte Nachfolger bzw. vor einen direkten oder indirekten Vorgänger verschoben wird. In Algorithmus 5.2 ist das Verfahren für die Ermittlung der unteren Grenze dargestellt. Die Ermittlung der oberen Grenze erfolgt analog.

Algorithmus 5.2 : Rekursive Ermittlung der unteren Grenze

Daten : Maschine m, Bearbeitungsschritt O_{ij} sowie der disjunktive Teilgraph SG
Ergebnis : Untere Grenze p auf der Maschine m

```
1  p ← 0;
2  Lower Bound(m, O_ij, SG);
3  Beginn Lower Bound(m, O_ij, SG)
4  |  für alle O_uv ∈ PRED(O_ij, SG) führe aus
5  |  |  ṁ ← ermittle Maschine von O_uv;
6  |  |  ṗ ← ermittle Position von O_uv auf ṁ;
7  |  |  wenn ṁ ≠ m dann
8  |  |  |  p ← max(p, Lower Bound(m, O_uv, SG));
9  |  |  sonst
10 |  |  |  p ← max(p, ṗ);
11 |  |  Ende
12 |  Ende
13 Ende
```

Da eine Vertauschung durch zwei Transfers abgebildet werden kann, ist es ausreichend, sicherzustellen, dass die Position des zweiten Bearbeitungsschritts zwischen der oberen und unteren Grenze des ersten liegt und umgekehrt die Position des ersten Bearbeitungsschritts zwischen der oberen und unteren Grenze des zweiten liegt.

Je größer eine Nachbarschaft ist, umso größer ist auch die Abdeckung des Lösungsraums durch diese Nachbarschaft. Eine kleinere Nachbarschaft lässt sich jedoch schneller durchsuchen. In Hansen und Mladenović (2001) wird vorgeschlagen, dass im Allgemeinen

zunächst mit kleinen Nachbarschaften begonnen werden soll. Wenn dort keine weiteren Verbesserungen erzielt werden, soll die Suche in größeren Nachbarschaften fortgesetzt werden. Zunächst werden die vier Basisnachbarschaften mit den Parametern $r = 2$ und $l = 1$ verwendet. Danach folgen größere Nachbarschaften mit den Parametern $r = 5$, $r = 10$ bzw. $r = n_{i,\max}$ und $l = 1$. Auf diese Weise ergeben sich zunächst 16 verschiedene Nachbarschaften.

Weiterhin wird jede dieser Nachbarschaften durch das Setzen des Parameters l vergrößert. Dazu werden die Züge innerhalb der Nachbarschaft zwei bzw. dreimal wiederholt ($l = 2$ und $l = 3$), um größere Nachbarschaften zu erzeugen. Auf diese Weise entstehen $k_{\max} = 48$ unterschiedliche Nachbarschaften. Die einzelnen Nachbarschaften, die verwendeten Parameter und ihre Reihenfolge sind in Tabelle 5.2 dargestellt. Die Tabellenfelder enthalten dabei die Nachbarschaftsbezeichnungen mit den Reihenfolgenummer. Hierbei ist zu erwähnen, dass die verschiedenen Nachbarschaften und ihre Reihenfolge das Ergebnis von umfangreichen Vorabexperimenten sind. Es ist wichtig, dass mit kleinen Werten für r und l gestartet wird, da in diesem Fall die Nachbarschaften klein sind und entsprechend wenig Aufwand im Rahmen der lokalen Suchen zum Finden eines lokalen Optimums anfällt.

Tabelle 5.2: Verwendete Nachbarschaften

	$r = 2$	$r = 5$	$r = 10$	$r = n_{i,\max}$
$l = 1$				
`TransferOnOneMachine`	N_1	N_5	N_9	N_{13}
`SwapOnOneMachine`	N_2	N_6	N_{10}	N_{14}
`TransferAcrossMachines`	N_3	N_7	N_{11}	N_{15}
`SwapAcrossMachines`	N_4	N_8	N_{12}	N_{16}
$l = 2$				
`TransferOnOneMachine`	N_{17}	N_{21}	N_{25}	N_{29}
`SwapOnOneMachine`	N_{18}	N_{22}	N_{26}	N_{30}
`TransferAcrossMachines`	N_{19}	N_{23}	N_{27}	N_{31}
`SwapAcrossMachines`	N_{20}	N_{24}	N_{28}	N_{32}
$l = 3$				
`TransferOnOneMachine`	N_{33}	N_{37}	N_{41}	N_{45}
`SwapOnOneMachine`	N_{34}	N_{38}	N_{42}	N_{46}
`TransferAcrossMachines`	N_{35}	N_{39}	N_{43}	N_{47}
`SwapAcrossMachines`	N_{36}	N_{40}	N_{44}	N_{48}

Es ist zu beachten, dass Transfer- und Vertauschungszüge unterschiedlich sind und nicht ein Spezialfall des jeweils anderen. Aus diesem Grund sind die Nachbarschaften nur teilweise verschachtelt. Weiterhin ist zu erwähnen, dass die Nachbarschaften in Tabelle 5.2 nicht durchgängig größer werden. So nimmt die Größe ab beim Übergang von N_4 zu N_5, beim Übergang von N_8 zu N_9, beim Übergang von N_{12} zu N_{13} und so weiter. Die Vorteile von nicht verschachtelten und eventuell schrumpfenden Nachbarschaften wurden in den Besten und Stützle (2001) für VND-Ansätze für ein Einmaschinenproblem untersucht.

5.3.5 Betrachtete VNS-Ansätze

Wie bereits in Abschnitt 3.4.5 beschrieben, existieren viele verschiedene VNS-Varianten, wie zum Beispiel VND oder R-VNS. In diesem Abschnitt werden unterschiedliche VNS-Ansätze auf ihre Leistungsfähigkeit hin untersucht. Ausgangspunkt ist das allgemeine VNS-Schema aus Algorithmus 3.9. Charakteristisch ist hierbei, dass zunächst ein zufälliger Ablaufplan aus der aktuellen Nachbarschaft gewählt wird (Shaking) und danach dieser Ablaufplan durch eine lokale Suche intensiver untersucht wird.

Der R-VNS-Ansatz kann in diesem Zusammenhang als allgemeines VNS-Schema ohne lokale Suche betrachtet werden. Der Lösungsraum wird nur durch das Erzeugen von zufälligen Ablaufplänen innerhalb der Nachbarschaften durchsucht.

Im weiteren Verlauf werden unterschiedliche lokale Suchverfahren für das allgemeine VNS-Schema beschrieben. Hierbei ist zu beachten, dass der Gesamtansatz möglichst schnell sein soll und gleichzeitig möglichst gute Ergebnisse im Hinblick auf die totale gewichtete Verspätung liefern soll.

In der Variante **Best Neighbor VNS** (BN-VNS) wird innerhalb der lokalen Suche der beste Nachbar aus der aktuellen Nachbarschaft gewählt. Im Unterschied dazu wird bei **Simple VNS** (S-VNS) das Verfahren des steilsten Abstiegs (siehe Algorithmus 3.4 als lokale Suche verwendet.

General VNS (G-VNS) verwendet VND (siehe Algorithmus 3.8) als lokale Suche. **Fast VNS** (F-VNS) basiert auf G-VNS, ersetzt VND allerdings durch **Variable Neighborhood First Descent** (VNFD), das ähnlich arbeitet. Im Unterschied zu VND wird bei VNFD die Nachbarschaft nur solange durchsucht, bis der erste bessere Nachbar gefunden wurde. Dadurch wird vermieden, dass die komplette Nachbarschaft durchsucht werden muss. In Algorithmus 5.3 ist der VNFD-Ansatz dargestellt.

Algorithmus 5.3 : Variable Neighborhood First Descent (VNFD)

Daten : $s \in S$
Ergebnis : Ablaufplan $s \in S$ mit der besten gefunden Bewertung $c(s)$

```
1  k_max ← Anzahl der Nachbarschaftsstrukturen für S;
2  wiederhole
3  |  k ← 1;
4  |  solange k ≤ k_max führe aus
5  |  |  ṡ ← suche ersten besseren Nachbarn aus N_k(s);
6  |  |  wenn c(ṡ) < c(s) dann
7  |  |  |  s ← ṡ, k ← 1;
8  |  |  sonst
9  |  |  |  k ← k + 1;
10 |  |  Ende
11 |  Ende
12 bis keine Verbesserung mehr gefunden ;
```

Zwei weitere Heuristiken werden untersucht: **Fast Skewed VNS** (FS-VNS) und **General Skewed VNS** (GS-VNS). FS-VNS variiert den Ansatz von F-VNS, indem zusätzlich auch

Verschlechterungen akzeptiert werden. GS-VNS arbeitet wie G-VNS, akzeptiert jedoch ebenfalls Verschlechterungen.

Innerhalb von FS-VNS und GS-VNS basiert die Akzeptanz eines neuen Ablaufplans darauf, wie stark er sich von dem aktuellen Ablaufplan unterscheidet. Genauer formuliert wird der neue Ablaufplan dann akzeptiert, wenn $c(\dot{s}) - \alpha\rho(s, \dot{s}) < c(s)$. Der Parameter α wird dabei in Abhängigkeit vom aktuellen Zug m sowie der maximalen Anzahl der Züge $m_{\max}$ gewählt ($\alpha := 1,0 - m/m_{\max}$). Mit $\rho(s, \dot{s})$ wird die Hamming-Distanz der zwei Ablaufpläne s und $\dot{s}$ bezeichnet.

Die Hamming-Distanz ist ein Maß für den Unterschied zwischen zwei gegebenen Zeichenketten, indem die Anzahl unterschiedlichen Zeichen in zwei Zeichenketten ermittelt werden. Die Berechnung für zwei Ablaufpläne erfolgt hier auf Basis einer Permutationsdarstellung derselben (vgl. auch Bierwirth u. a. (1996) für einen ähnlichen Ansatz für Job-Shop-Scheduling-Probleme). Ein Ablaufplan $s \in S$ kann als binärer Vektor mit der Größe $n \times (m+n-1)$ dargestellt werden, wobei mit n die Anzahl der Bearbeitungsschritte und mit m die Anzahl der Maschinen bezeichnet wird. Zur Erzeugung des binären Vektors wird zunächst jedem Bearbeitungsschritt und jeder Maschine eine eindeutige Nummer zugeordnet. Danach kann der folgende Vektor erzeugt werden:

$$v := (\underbrace{\underbrace{\pi_1, \ldots, \pi_m}_{\text{Maschinen}}, \underbrace{\pi_{m+1}, \ldots, \pi_{m+n-1}}_{\text{Nachfolger}}}_{\text{Bearbeitungsschritt 1}}, \underbrace{\underbrace{\pi_{m+n}, \ldots, \pi_{2m+n}}_{\text{Maschinen}}, \underbrace{\pi_{2m+n+1}, \ldots, \pi_{2m+n}}_{\text{Nachfolger}}}_{\text{Bearbeitungsschritt 2}}, \ldots). \quad (5.7)$$

Für jeden Bearbeitungsschritt existiert in dem Vektor (5.7) ein Feld von $m+n-1$ Einträgen. Die ersten m Einträge stellen die Maschinenzuteilung dar. Hierbei ist für jede Maschine ein Eintrag π_p reserviert. Dieser ist auf $\pi_p := 1$ gesetzt, wenn der Bearbeitungsschritt auf der entsprechenden Maschine bearbeitet wird, andernfalls ist $\pi_p := 0$. Die restlichen Einträge eines Bearbeitungsschritts repräsentieren die Nachfolger auf dieser Maschine. Ist O_{ij} ein unmittelbarer Nachfolger von O_{uv}, so ist der O_{ij} repräsentierende Eintrag $\pi_p := 1$, ansonsten ist $\pi_p := 0$.

5.4 Exakte und heuristische Vergleichsverfahren

Zur Überprüfung der Korrektheit der Ergebnisse der VNS-Verfahren wurden ein optimales Verfahren auf Basis eines Enumerationsansatzes und ein Genetischer Algorithmus gewählt.

An dieser Stelle soll zunächst das optimale Verfahren für das Problem (5.2) vorgestellt werden. Da es sich um ein *NP*-schweres Problem handelt, kann nicht erwartet werden, dass der Ansatz auf mittlere und große Probleme anwendbar ist. Es wird der Fall von $m = 2$ identischen parallelen Maschinen betrachtet. In diesem Fall kann eine Lösung des Scheduling-Problems als Liste mit $n+1$ Elementen abgebildet werden, wobei n die Anzahl der Bearbeitungsschritte darstellt. Mittels des zusätzlichen Eintrags $n+1$ wird die Zuteilungsentscheidung auf die einzelnen Maschinen abgebildet. Das Scheduling-Problem kann nun so aufgefasst werden, dass eine zulässige Permutation $\sigma = (\sigma[1], \ldots, \sigma[n+1])$ zu finden ist, die zu dem kleinsten TWT-Wert führt. Die Bearbeitungsschritte $\sigma[1], \ldots, \sigma[k-1]$, mit

$k = \sigma^{-1}[n+1]$ werden der ersten Maschine zugeteilt, wohingegen die Bearbeitungsschritte $\sigma[k+1], \ldots, \sigma[n+1]$ der zweiten Maschine zugeteilt werden. Die Reihenfolge innerhalb des Ablaufplans ergibt sich aus der Reihenfolge in der Permutation. Dieser Ansatz wird verwendet, um die Korrektheit der VNS-Implementierung sicherzustellen. Im weiteren Verlauf wird dieser Ansatz PA genannt.

In einem weiteren Ansatz wird ein GA verwendet, um die Bearbeitungsschritte zunächst den Maschinen zuzuteilen. Wie bereits in Kapitel 3 beschrieben, ist ein GA im Unterschied zu VNS eine populationsbasiertes Verfahren. Ein Chromosom ist in diesem Fall als ein Vektor der Größe n in der folgenden Form kodiert:

$$(m_1, \ldots, m_n). \tag{5.8}$$

Hierbei repräsentiert jedes Feld $m_i \in \{1, \ldots, m\}$ die Maschinenzuordnung für einen der n Bearbeitungsschritte. Als Operatoren werden One-Point-Crossover und Swap-Mutationen eingesetzt (vgl. Balasubramanian u. a. (2004) für eine genauere Beschreibung für ein ähnliches paralleles Maschinenbelegungsproblem). Die Implementierung erfolgte mit Hilfe der GALib (vgl. Wall (1999)).

Zur Bewertung jedes einzelnen Chromosoms wird eine Modifikation des Verfahrens auf Basis von Prioritätsregeln verwendet (vgl. Algorithmus 5.1). Im Unterschied zu Algorithmus 5.1 muss dabei sichergestellt werden, dass die einzelnen Bearbeitungsschritte auch nur auf den durch den GA vorgegebenen Maschinen bearbeitet werden. Aufgrund der Vorrangbeziehungen zwischen den Bearbeitungsschritten ist es nicht mehr möglich, wie in Balasubramanian u. a. (2004) mehrere Einmaschinenprobleme zu lösen. Stattdessen muss das parallele Maschinenproblem unter Berücksichtigung der bereits erfolgten Maschinenzuteilung gelöst werden. Dabei kann es passieren, dass zu einen bestimmten Zeitpunkt kein Bearbeitungsschritt für eine gegebene Maschine verfügbar ist. In diesem Fall wird eine zusätzliche Übergangszeit innerhalb des Ablaufplans für diese Maschine eingefügt.

Als Prioritätsregel wird an dieser Stelle ATCS eingesetzt. Die Verwendung von ATCS gegenüber ATCSR-1 oder ATCSR-2 ist dadurch begründet, dass die ATCS-Variante deutlich schneller ist als die anderen Varianten. Für eine bessere Vergleichbarkeit mit den VNS-Ansätzen wird der GA nach der Erzeugung von 1000 Individuen beendet und mit den VNS-Ergebnissen nach 1000 Zügen verglichen.

5.5 Numerische Experimente zur Leistungsbewertung

In diesem Abschnitt erfolgt die Beschreibung des Experimentdesigns. Es wird zunächst beschrieben, wie die einzelnen Testinstanzen erzeugt wurden, danach erfolgt eine Vorstellung des Experimentdesigns sowie eine Diskussion der Ergebnisse.

5.5.1 Erzeugung der Testinstanzen und Experimentdesign

Zur Generierung der Testinstanzen wird ein ähnlicher Ansatz wie in Lee und Pinedo (1997) und Pfund u. a. (2008b) gewählt. Die einzelnen Testinstanzen werden durch sieben

Faktoren charakterisiert. Im Unterschied zu den Arbeiten Lee und Pinedo (1997) und Pfund u. a. (2008b) wird noch ein Faktor hinzugefügt, um die Vorrangbeziehungen abzubilden.

Betrachtet werden $m = 5$ Maschinen. Jedes Los wird nur einmal bearbeitet und hat damit einen Bearbeitungsschritt. Die Anzahl der Lose wird auf Basis des Los-Maschinen-Faktors $\mu \in \{11; 19; 27\}$ durch $n := m \times \mu$ festgelegt. Die Bearbeitungszeiten der Bearbeitungsschritte p_{ij} werden zufällig entsprechend der Gleichverteilung $U[50, 150]$ generiert. Die Gewichte der Lose w_j werden entsprechend der diskreten Gleichverteilung $DU[1, 10]$ erzeugt. Die Ermittlung der durchschnittlichen Rüstzeit $\bar{s} = \eta \cdot \bar{p}$ erfolgt mittels des Rüstzeitfaktors $\eta \in \{0{,}020; 1{,}010; 2{,}000\}$. Dieser charakterisiert den Einfluss der Rüstzeiten innerhalb des Problems. Mit $\bar{p}$ ist dabei die durchschnittliche Bearbeitungszeit aller Bearbeitungsschritte bezeichnet.

Die Zykluszeit wird durch den Ausdruck $\widehat{C_{\max}} := (\beta\bar{s} + \bar{p})\mu$ geschätzt. Der Koeffizient β dient hierbei der Abschätzung des Einflusses der Rüstzeiten auf die Zykluszeit und wird mittels $\beta := 0.4 + 10\mu^2 - \eta 7$ berechnet (vgl. Lee und Pinedo (1997)). Der durchschnittliche geplante Aussteuertermin der Lose $\bar{d} := \widehat{C_{\max}}(1 - \tau)$ wird mittels des Faktors $\tau \in \{0{,}300; 0{,}600; 0{,}900\}$ ermittelt. Dieser beschreibt, wie nah der durchschnittliche geplanten Aussteuertermin an der geschätzten Zykluszeit liegt. Auf Basis des durchschnittlichen geplanten Aussteuertermins und des Faktors $R \in \{0{,}250; 0{,}630; 1{,}000\}$ werden die geplanten Aussteuertermine der Lose mit der Wahrscheinlichkeit τ gleichverteilt aus dem Intervall $[(1 - R)\bar{d}, \bar{d}]$ sowie mit der Wahrscheinlichkeit $(1 - \tau)$ gleichverteilt aus dem Intervall $[\bar{d}, \bar{d} + (\widehat{C_{\max}} - \bar{d})R]$ erzeugt.

Der Losverfügbarkeitsfaktor $J_a \in \{0{,}2; 0{,}5; 0{,}8\}$ wird verwendet, um den Einsteuertermin des Loses zu bestimmen. Mit einer Wahrscheinlichkeit von J_a erhält das aktuell erzeugte Los den Einsteuertermin $r_j := 0$. Mit der Wahrscheinlichkeit $(1 - J_a)$ erfolgt die Generierung des Einsteuertermins gleichverteilt aus dem Intervall $U[\max(d_j - r_\tau p_{ij}, 0), d_j]$. Hierbei wird mit $r_\tau \in \{1{,}0; 5{,}5; 10{,}0\}$ gesteuert, wie weit der Einsteuer- und der geplante Aussteuertermin auseinanderliegen.

Die Vorrangbeziehungen werden auf Basis des Vorrangfaktors $\pi \in \{0{,}25; 0{,}50; 0{,}75\}$ hinzugefügt. Es werden keine Vorrangbeziehungen erzeugt, wenn die Realisierung einer $U[0, 1]$-verteilten Zufallsvariable größer als π ist. Ist die Realisierung kleiner als π, dann wird die Anzahl der direkten Vorgänger entsprechend $DU[0, \min(3, \lfloor \pi n_{\text{cur}} \rfloor)]$ erzeugt. Hierbei wird mit n_{cur} die Anzahl der bis dahin erzeugten Lose bezeichnet. Die Vorgänger werden zufällig aus der Menge der bereits erzeugten Lose ausgewählt. Als Verzögerungszeit wird $t_{uv}^{ij} := 0$ zwischen den Bearbeitungsschritten der ausgewählten Lose und dem Bearbeitungsschritt des aktuellen Loses verwendet. Die frühesten Verfügbarkeitszeitpunkte und gewünschten Fertigstellungstermine werden entsprechend berechnet. Das Gesamtdesign ist in Tabelle 5.3 zusammengefasst.

Jede Kombination der Faktoren wird betrachtet. Insgesamt werden $3^7 = 2\,187$ unterschiedliche Testinstanzen für alle Algorithmen getestet. Zur Überprüfung der Korrektheit der VNS-Implementierungen mit Hilfe des GA-Ansatzes wird nur eine Untermenge der oben genannten Faktoren verwendet. Für den Los-Maschinen-Faktor wird nur $\mu = 19$ betrachtet. Bei den anderen Faktoren werden nur die niedrigen und mittleren Ausprä-

Tabelle 5.3: Design der Experimente

Faktor	Niedrig	Mittel	Hoch
μ	11,000	19,000	27,000
η	0,020	1,010	2,000
τ	0,300	0,600	0,900
R	0,250	0,630	1,000
J_a	0,200	0,500	0,800
r_τ	1,000	5,500	10,000
π	0,250	0,500	0,750

gungen aus Tabelle 5.3 berücksichtigt. Insgesamt müssen $2^6 = 64$ verschiedene Probleme gelöst werden.

5.5.2 Diskussion der Ergebnisse

Die Qualität von VNS hängt hauptsächlich von den gewählten Nachbarschaften und der Reihenfolge, in der diese Nachbarschaften angewendet werden, ab. In vorigen Untersuchungen wurde bereits die Eignung der Nachbarschaften und der Reihenfolge aus Tabelle 5.2 untersucht. Weiterhin ist die Qualität von der verwendeten lokale Suche abhängig. In dieser Arbeit werden die in Unterabschnitt 5.3.5 vorgestellten Ansätze untersucht. Die Bewertung der einzelnen erzeugten Ablaufpläne erfolgt wie in Unterabschnitt 5.2.2 beschrieben.

Um die Korrektheit der Implementierung der einzelnen VNS-Ansätze zu überprüfen, wurden 20 Testinstanzen mit $m = 2$ und $n = 4, \ldots, 8$ erzeugt und mittels PA optimal gelöst. Diese wurden nach maximal 1000 Zügen ebenfalls durch die unterschiedlichen VNS-Ansätze gefunden. Weiterhin wurden eine Untermenge der Experimente zusätzlich unter Verwendung des GAs gelöst. Es stellte sich heraus, dass die besten VNS-Ansätze GA um ca. 9% schlagen. Die GA-Ergebnisse selbst sind dabei im Mittel leicht besser (ca. $0,5\%$) als die Ergebnisse, die durch ATCSR-1 ermittelt wurden. Diese Ergebnisse ähneln den Ergebnisse aus Almeder und Mönch (2009) für ein anderes paralleles Maschinenbelegungsproblem.

Da das Verfahren für das Transportteilproblem innerhalb der SBH eingesetzt werden soll, besteht das Hauptinteresse daran, einen VNS-Ansatz zu finden, der mit relativ wenigen Zügen ein gutes Ergebnis bezüglich der totalen gewichteten Verspätung liefert.

Untersucht wird der Einfluss der maximal erlaubten Anzahl von Zügen. Ausgehend von einem initialen Ablaufplan, der durch ATCSR-1, ATCSR-2, ATCS bzw. ATCR erzeugt wurde, werden die einzelnen VNS-Ansätze mit 100, 500, 1 000, 5 000 sowie 10 000 Zügen gestartet. Für jede ATC-Variante werden alle VNS-Ansätze mit jeder Anzahl von Züge getestet. Insgesamt werden damit 144 unterschiedliche Varianten geprüft. Alle präsentierten Ergebnisse sind auf Basis des TWT-Wertes eines Ablaufplans normiert, der mit Hilfe Hilfe der FIFO-Prioritätsregel erzeugt wurde. Eine Normierung ist notwendig, um eine Vergleichbarkeit zwischen den einzelnen Testinstanzen zu ermöglichen.

Tabelle 5.4 zeigt den Mittelwert, die Standardabweichung, den besten und den schlech-

testen Fall bezüglich der totale gewichtete Verspätung über alle Testinstanzen in Abhängigkeit der maximalen Zuganzahl für jeden VNS-Ansatz. R-VNS schlägt hier für längere Laufzeiten die anderen Ansätze. Allerdings scheint die Variante F-VNS bzw. FS-VNS für sehr kurze Laufzeiten besser geeignet zu sein. Der Grund hierfür liegt in der verwendeten lokalen Suche. Innerhalb von F-VNS bzw. FS-VNS, das im Wesentlichen dieselbe Strategie wie F-VNS verwendet, wird versucht, möglichst schnell zu Verbesserungen zu gelangen, wohingegen R-VNS stärker zufällig im Lösungsraum sucht. Jedoch ist bei R-VNS im Falle von langen Laufzeiten eine bessere Abdeckung des Lösungsraums möglich, wodurch langfristig bessere Lösungen gefunden werden. Die anderen VNS-Varianten verbringen dagegen zu viel Zeit bei der Durchsuchung der Nachbarschaften. Dies deckt sich auch mit den Ergebnissen im Vergleich zu PA, wo die intelligenteren lokalen Suchen die optimale Lösung sehr häufig gefunden haben, jedoch nicht der R-VNS Ansatz. Bei größeren Problemen scheint dies jedoch aus dem oben genannten Grund nicht mehr so gut zu funktionieren. Die Skewed-Varianten FS-VNS und GS-VNS liefern hierbei leicht bessere Ergebnisse als die Nicht-Skewed-Varianten F-VNS und G-VNS. Der Grund hierfür ist, dass zu Beginn breiter gesucht wird als bei den anderen Varianten.

Weiterhin ist interessant, dass die Anwendung des VNS-Ansatzes zu einer Reduzierung der Standardabweichung bezüglich der totale gewichtete Verspätung führt. Daraus kann gefolgert werden, dass VNS im Allgemeinen eine Verbesserung gegenüber der Initiallösung erzielen kann. Der Grund hierfür ist der beständige Verbesserungsdruck, der auch bei schweren Problemen zu kleinen Verbesserungen führt, wie man an den Werten für die schlechtesten Fälle sehen kann. Während sich die besten Ergebnisse mit steigender Zügeanzahl kaum verbessert haben, sind die schlechtesten Ergebnisse besser geworden. Durch die kleinere Spanne ergibt sich auch eine geringere Standardabweichung.

Tabelle 5.4: Ergebnisse in Abhängigkeit von der Anzahl der Züge

Züge / VNS	Durchschnitt	Standard-abweichung	Minimum	Maximum
100 Züge				
FS-VNS	0,593	0,272	0,082	1,291
F-VNS	0,598	0,272	0,082	1,291
GS-VNS	0,600	0,276	0,082	1,380
G-VNS	0,605	0,276	0,082	1,380
BN-VNS	0,610	0,276	0,082	1,380
R-VNS	0,604	0,271	0,082	1,341
S-VNS	0,607	0,276	0,082	1,380
500 Züge				
FS-VNS	0,575	0,257	0,082	1,186
F-VNS	0,580	0,257	0,082	1,186
GS-VNS	0,579	0,259	0,082	1,184

Fortsetzung auf der nächsten Seite ...

Züge VNS	Durchschnitt	Standard-abweichung	Minimum	Maximum
G-VNS	0,584	0,259	0,082	1,184
BN-VNS	0,592	0,261	0,082	1,342
R-VNS	0,571	0,247	0,082	1,276
S-VNS	0,587	0,259	0,082	1,337
1 000 Züge				
FS-VNS	0,562	0,247	0,082	1,126
F-VNS	0,566	0,248	0,082	1,126
GS-VNS	0,564	0,247	0,082	1,136
G-VNS	0,569	0,248	0,082	1,136
BN-VNS	0,579	0,251	0,082	1,317
R-VNS	0,550	0,235	0,082	1,169
S-VNS	0,572	0,248	0,082	1,317
5 000 Züge				
FS-VNS	0,518	0,224	0,081	1,071
F-VNS	0,522	0,224	0,081	1,071
GS-VNS	0,522	0,225	0,081	1,047
G-VNS	0,526	0,225	0,081	1,047
BN-VNS	0,545	0,232	0,081	1,233
R-VNS	0,481	0,215	0,079	1,003
S-VNS	0,531	0,226	0,081	1,047
10 000 Züge				
FS-VNS	0,497	0,216	0,080	1,004
F-VNS	0,500	0,216	0,080	1,004
GS-VNS	0,501	0,218	0,080	1,013
G-VNS	0,505	0,218	0,080	1,013
BN-VNS	0,526	0,224	0,081	1,132
R-VNS	0,464	0,213	0,076	0,984
S-VNS	0,511	0,218	0,081	1,013

Tabelle 5.5 zeigt die durchschnittlichen TWT-Werte für alle VNS-Ansätze und alle Initiallöser. Hier ist interessant, dass ATCS in Kombination mit den verschiedenen VNS die besten Ergebnisse erzielt hat, obwohl es eine schlechtere Initiallösung geliefert hat als ATCSR-1. Gleichzeitig ist der Zeitaufwand für ATCS deutlich geringer ist als für ATCSR-1. Die bessere Initiallösung führt nicht zwangsläufig zu einer besseren Gesamtlösung. Bei genauerer Betrachtung dieses Phänomens ist erkennbar, dass bei 100, 500 und 1 000 Zügen alle ATCS-Initiallöser alle anderen Varianten schlagen. Bei 5 000 Zügen trifft dies nur noch für FS-VNS, F-VNS, GS-VNS und R-VNS zu und bei 10 000 Zügen trifft dies nur noch auf R-VNS zu. Der letzte Umstand lässt sich dadurch erklären, dass die initiale

Lösung mit längeren Laufzeiten immer mehr an Bedeutung verliert. Dies ist auch daran ersichtlich, dass die Unterschiede zwischen bei den einzelnen VNS-Ansätzen bezüglich der Initiallöser geringer werden, wohingegen die Unterschiede zwischen den VNS-Ansätzen größer werden. Desweiteren führen Ansätze auf Basis von ATCSR-Prioritätsregeln oft zu zusätzlichen Wartezeiten durch die Berücksichtigung von später verfügbaren Bearbeitungsschritten. Dies führt häufig zu Verbesserungen der totalen gewichteten Verspätung gegenüber den anderen Initiallösern, allerdings scheint auch die Position der einzelnen Bearbeitungsschritte innerhalb des Ablaufplans für die Nachbarschaftssuchen ungünstig zu liegen, so dass die VNS-Ansätze zu Beginn in einem lokalen Optimum verbleiben. Erst durch eine größere Anzahl von Shaking-Schritten ist es möglich, diesen lokalen Optima zu entkommen.

Tabelle 5.5: Ergebnisse in Abhängigkeit von den initialen Lösungsverfahren

Züge VNS	ATCSR-1	ATCSR-2	ATCS	ATCR
0 Züge				
Initiallösung	0,535	0,548	0,547	0,902
100 Züge				
FS-VNS	0,533	0,521	0,504	0,862
F-VNS	0,533	0,526	0,506	0,861
GS-VNS	0,535	0,524	0,506	0,876
G-VNS	0,534	0,532	0,511	0,874
BN-VNS	0,533	0,544	0,519	0,875
R-VNS	0,533	0,540	0,518	0,854
S-VNS	0,534	0,536	0,515	0,875
500 Züge				
FS-VNS	0,525	0,512	0,496	0,807
F-VNS	0,526	0,518	0,498	0,807
GS-VNS	0,528	0,515	0,497	0,810
G-VNS	0,527	0,523	0,503	0,810
BN-VNS	0,527	0,538	0,512	0,818
R-VNS	0,522	0,527	0,506	0,745
S-VNS	0,526	0,528	0,507	0,811
1 000 Züge				
FS-VNS	0,520	0,506	0,490	0,767
F-VNS	0,520	0,511	0,492	0,768
GS-VNS	0,522	0,509	0,491	0,762
G-VNS	0,521	0,517	0,496	0,763
BN-VNS	0,522	0,532	0,506	0,776
R-VNS	0,514	0,518	0,497	0,684
S-VNS	0,521	0,522	0,500	0,764
5 000 Züge				

Fortsetzung auf der nächsten Seite ...

Züge VNS	ATCSR-1	ATCSR-2	ATCS	ATCR
FS-VNS	0,501	0,486	0,470	0,636
F-VNS	0,502	0,490	0,472	0,638
GS-VNS	0,503	0,488	0,471	0,643
G-VNS	0,503	0,495	0,474	0,645
BN-VNS	0,509	0,516	0,491	0,678
R-VNS	0,472	0,472	0,457	0,526
S-VNS	0,504	0,501	0,480	0,653
10 000 Züge				
FS-VNS	0,488	0,474	0,460	0,580
F-VNS	0,492	0,478	0,461	0,580
GS-VNS	0,492	0,477	0,461	0,589
G-VNS	0,492	0,482	0,464	0,590
BN-VNS	0,499	0,504	0,482	0,630
R-VNS	0,461	0,459	0,446	0,491
S-VNS	0,494	0,489	0,469	0,602

Tabelle 5.6 zeigt die Verbesserungen aller VNS-Ansätze im Bezug auf die Initiallösung von ATCSR-1. Die Daten wurden entsprechend der einzelnen Faktoren gruppiert. Dies bedeutet zum Beispiel für $\mu = 11$, dass alle anderen Faktoren variiert wurden, der Los-Maschinen-Faktor jedoch konstant bei 11 gehalten wurde. Tabelle 5.7 zeigt dieselbe Gruppierung normiert mit Hilfe der FIFO-Ergebnisse. Die Betrachtung von ATCSR-1 anstelle von ATCS an dieser Stelle erfolgt, um die Verbesserungen der VNS-Ansätze gegenüber der besten Intitiallösung zu untersuchen. Da die Durchschnittswerte über alle VNS-Ansätze betrachted werden, sind die Ergebnisse bezüglich ATCS sehr ähnlich.

Tabelle 5.6: Verbesserungen der Lösung von ATCSR-1

Faktor	100	500	1 000	5 000	10 000
$\mu = 11$	1,35 %	2,93 %	4,01 %	6,77 %	8,29 %
$\mu = 19$	0,67 %	1,39 %	2,05 %	4,41 %	5,81 %
$\mu = 27$	0,48 %	1,48 %	2,44 %	6,83 %	9,63 %
$\eta = 0{,}020$	1,11 %	3,17 %	4,83 %	10,20 %	12,79 %
$\eta = 1{,}010$	0,62 %	1,00 %	1,23 %	2,65 %	3,72 %
$\eta = 2{,}000$	0,64 %	0,91 %	1,09 %	2,45 %	3,37 %
$\tau = 0{,}300$	1,65 %	3,68 %	5,34 %	10,35 %	13,27 %
$\tau = 0{,}600$	0,74 %	1,65 %	2,32 %	4,49 %	5,80 %
$\tau = 0{,}900$	0,37 %	0,79 %	1,13 %	2,38 %	3,11 %
$R = 0{,}250$	0,91 %	1,96 %	2,86 %	5,44 %	6,90 %

Fortsetzung auf der nächsten Seite ...

Faktor	100	500	1 000	5 000	10 000
$R =$ 0,630	0,82 %	1,81 %	2,59 %	5,13 %	6,56 %
$R =$ 1,000	1,03 %	2,36 %	3,37 %	6,79 %	8,88 %
$J_a =$ 0,200	0,98 %	2,06 %	2,93 %	5,64 %	7,23 %
$J_a =$ 0,500	0,92 %	2,07 %	3,03 %	5,98 %	7,63 %
$J_a =$ 0,800	0,87 %	1,99 %	2,85 %	5,74 %	7,47 %
$r_\tau =$ 1,000	0,94 %	2,08 %	2,98 %	5,80 %	7,47 %
$r_\tau =$ 5,500	0,90 %	2,03 %	2,93 %	5,87 %	7,54 %
$r_\tau =$ 10,000	0,93 %	2,02 %	2,90 %	5,69 %	7,33 %
$\pi =$ 0,250	0,95 %	2,18 %	3,22 %	6,45 %	8,20 %
$\pi =$ 0,500	0,87 %	2,00 %	3,00 %	5,84 %	7,61 %
$\pi =$ 0,750	0,95 %	1,95 %	2,59 %	5,06 %	6,52 %

Wie in Tabelle 5.6 ersichtlich, haben einige Faktoren einen großen Einfluss auf die Ergebnisse der einzelnen VNS-Ansätze. Dies sind der Vorrangfaktor π, der Faktor τ zur Ermittlung des geplanten Aussteuertermins, der Rüstzeitfaktor η sowie der Los-Maschinen-Faktor μ. Mit steigenden Werten von π liefern die VNS-Ansätze bessere Ergebnisse. Dies liegt daran, dass die Vorrangbeziehungen direkt innerhalb der Nachbarschaftsstrukturen berücksichtigt werden. Größere Werte für τ und η sorgen dafür, dass das Scheduling-Problem schwerer zu lösen ist. Aus diesem Grund sind die relativen Verbesserungen gegenüber der initialen Lösung bei größeren Werten nicht so stark wie bei kleineren Werten. Dasselbe gilt für die Anzahl der Lose. Mit größerem μ werden die Nachbarschaften größer. Damit ist es schwieriger, bei gleichbleibendem Zeitaufwand dieselben Verbesserungen zu erreichen.

Tabelle 5.7: Ergebnisse für ATCSR-1 relativ zu FIFO

Faktor	0	100	500	1 000	5 000	10 000
$\mu = 11$	0,522	0,516	0,506	0,499	0,479	0,472
$\mu = 19$	0,470	0,466	0,461	0,457	0,440	0,430
$\mu = 27$	0,767	0,766	0,758	0,752	0,719	0,697
$\eta =$ 0,020	0,817	0,809	0,792	0,779	0,736	0,715
$\eta =$ 1,010	0,369	0,367	0,365	0,365	0,360	0,357
$\eta =$ 2,000	0,299	0,298	0,297	0,297	0,292	0,290
$\tau =$ 0,300	0,439	0,437	0,424	0,413	0,378	0,361
$\tau =$ 0,600	0,518	0,515	0,508	0,504	0,487	0,479
$\tau =$ 0,900	0,654	0,652	0,649	0,646	0,635	0,629
$R =$ 0,250	0,528	0,527	0,520	0,514	0,496	0,487
$R =$ 0,630	0,533	0,531	0,524	0,519	0,501	0,492
$R =$ 1,000	0,548	0,543	0,534	0,527	0,500	0,487

Fortsetzung auf der nächsten Seite ...

Faktor	0	100	500	1 000	5 000	10 000
$J_a =$ 0,200	0,530	0,530	0,522	0,516	0,497	0,486
$J_a =$ 0,500	0,539	0,535	0,527	0,521	0,499	0,488
$J_a =$ 0,800	0,536	0,536	0,528	0,523	0,501	0,491
$r_\tau =$ 1,000	0,532	0,532	0,524	0,518	0,497	0,486
$r_\tau =$ 5,500	0,537	0,536	0,528	0,522	0,501	0,490
$r_\tau =$ 10,000	0,535	0,533	0,526	0,520	0,499	0,489
$\pi =$ 0,250	0,550	0,547	0,538	0,532	0,508	0,497
$\pi =$ 0,500	0,531	0,530	0,522	0,516	0,494	0,483
$\pi =$ 0,750	0,525	0,524	0,517	0,513	0,495	0,485

In den Tabellen 5.6 und 5.7 ist zu sehen, das VNS einige der Einschränkungen der ATC-Heuristiken überwinden kann. ATC-basierte Heuristiken liefern relativ schlechte Ergebnisse, wenn die geplanten Aussteuerungstermine der Lose sehr spät liegen oder wenn die Auslastung der Maschinen sehr niedrig ist. In diesen Situationen wird der Schlupfzeitterm bei der Berechnung des Prioritätsindex oft negativ und der Index entsprechend klein. Dadurch haben ATC-basierte Verfahren häufig das Problem, dass die Unterscheidung zwischen den einzelnen Losen zu undifferenziert wird. Dies führt dazu, dass Entscheidungen, die sehr früh durch die Heuristik getroffen werden, oft nicht effektiv sind. Wenn mehr Lose betrachtet werden und der Entscheidungszeitpunkt näher an die geplanten Aussteuerungstermine rückt, wird die Unterscheidung differenzierter. Die früheren Entscheidungen lassen sich jedoch nicht mehr zurücknehmen. VNS-Ansätze haben hier den Vorteil, dass sie diese früheren Entscheidungen korrigieren können und das Leistungsmaß TWT direkt berücksichtigen.

In Tabelle 5.6 ist weiterhin zu sehen, dass sich für kürzere Laufzeiten mit weniger als 1 000 Zügen Verbesserungen des initialen Ablaufplans von 2 % bis 5 % ergeben. Für längere Laufzeiten beträgt die Verbesserung zwischen 5 % und 14 %. Wie stark die Verbesserung ausfällt, ist dabei abhängig von den einzelnen Faktoren des verwendeten Designs. In Abbildung 5.1 sind die durchschnittlichen Laufzeiten für die einzelnen VNS-Varianten dargestellt. Die Laufzeit für ein Experiment mit ATCSR-1 und 10 000 Zügen beträgt im Durchschnitt weniger etwas mehr als 15 Minuten. Bei der Verwendung von ATCS mit F-VNS mit nur 500 Zügen reduziert sich die Laufzeit auf weniger als eine Minute. Die Experimente wurden auf Computern mit AMD Opteron Prozessor mit 2,2 bzw. 2,4 GHz sowie 4 GB RAM durchgeführt.

Welcher der vorgeschlagenen VNS-Ansätze für ein gegebenes Scheduling-Problem zu verwenden ist, ist abhängig von den einzelnen Faktoren und von der zur Verfügung stehenden Zeit. Ist nicht viel Zeit verfügbar, d.h. sind nicht mehr als 1000 Züge möglich, ist die Verwendung von F-VNS bzw. FS-VNS sinnvoll. Für längere Laufzeiten scheint R-VNS besser geeignet zu sein. Bezüglich der Faktoren lässt sich sagen, dass die erzielbaren Verbesserungen bei einer großen Anzahl von Bearbeitungsschritten geringer sind, als bei einer kleinen Anzahl. Der Grund hierfür liegt im Wesentlichen in den größeren Nach-

barschaften, die bei mehr Bearbeitungsschritten durchsucht werden müssen. Dadurch sind bei gleicher zur Verfügung stehenden Zeit tendenziell geringere Verbesserungen möglich als bei kleineren Nachbarschaften. Auch im Falle von großen Rüstzeiten und engen geplanten Aussteuerterminen der Lose können die VNS-Ansätze nur relativ geringe Verbesserungen erzielen, da hier die verwendeten Intiallöser bereits sehr gute Ergebnisse liefern. Weiterhin eignen sich die VNS-Ansätze besonders bei weiten geplanten Aussteuerterminen, da einerseits die ATC-basierten Initiallöser tendenziell schlechtere Ergebnisse liefern und es andererseits bei den einzelnen Transfers und Vertauschungen genügend Raum für Verbesserungen gibt.

Zusammenfassend lässt sich sagen, dass VNS-Ansätze zu sehr guten Ergebnissen für einfache Problemen führen. Dies wird auch in Almeder und Mönch (2009) für ein Batch-Maschinenproblem bestätigt. In Kombination mit ATC-basierten Initiallösern, die dafür bekannt sind, gute Ergebnisse für schwere Probleme zu liefern, sind relativ gute und robuste Gesamtergebnisse für alle Umgebungssituationen mit geringem zusätzlichen Rechenaufwand möglich.

5.6 Anpassungen zur Verwendung der Verfahren in der Shifting-Bottleneck-Heuristik

Basierend auf den vorhergehenden Untersuchungen wird sowohl für die Maschinengruppenteilprobleme als auch für das Transportteilproblem Teilproblemlöser auf Basis von ATCS vergeschlagen. Das Transportteilproblem wird dabei, wie in Unterabschnitt 4.3.3 beschrieben, in ein entsprechendes Maschinengruppenproblem umgewandelt, so dass für alle Teilprobleme das Rahmenwerk aus Abschnitt 5.2 eingesetzt werden kann. Für das Transportteilproblem wird zusätzlich vorgeschlagen, den erzeugten Ablaufplan optional noch mit F-VNS zu verbessern.

Für die Verwendung innerhalb der SBH ist eine Anpassung der verwendeten Prioritätsregel sinnvoll. Aufgrund der verzögernden Vorrangbeziehungen hat die Einplanung eines Bearbeitungsschritts O_{ij} auf einer Maschine nicht nur Einfluss auf den Fertigstellungszeitpunkt c_j des zugehörigen Loses J_j, sondern auch auf alle Fertigstellungszeitpunkte c_k der Lose J_k, für die ein Pfad zwischen dem Bearbeitungsknoten und dem Endknoten v_k innerhalb des disjunktiven Graphen G für das Gesamtproblem existiert. Aus diesem Grund wird die ATCS-Regel so modifiziert, dass die gewünschten Fertigstellungstermine d_{ij}^k des Bearbeitungsschrittes bezüglich der Lose J_k berücksichtigt werden (vgl. auch Pinedo und Singer (1999), Abschnitt 4.2 sowie Abschnitt 4.3). Der Prioritätsindex für O_{ij} zum Zeitpunkt t wird nun wie folgt berechnet:

$$I(O_{ij},t) := \sum_{k=1}^{n} \frac{w_k}{p_{ij}} e^{-\frac{\max(d_{ij}^k - p_{ij} - \max(r_{ij},t),0)}{k_1 \bar{p}} - \frac{s_{lj}}{k_2 \bar{s}}}. \tag{5.9}$$

Mit w_k wird das Gewicht des Loses J_k bezeichnet, p_{ij} repräsentiert die Bearbeitungszeit des Bearbeitungsschritts und s_{lj} stellt die notwendige Rüstzeit dar, um die Maschine

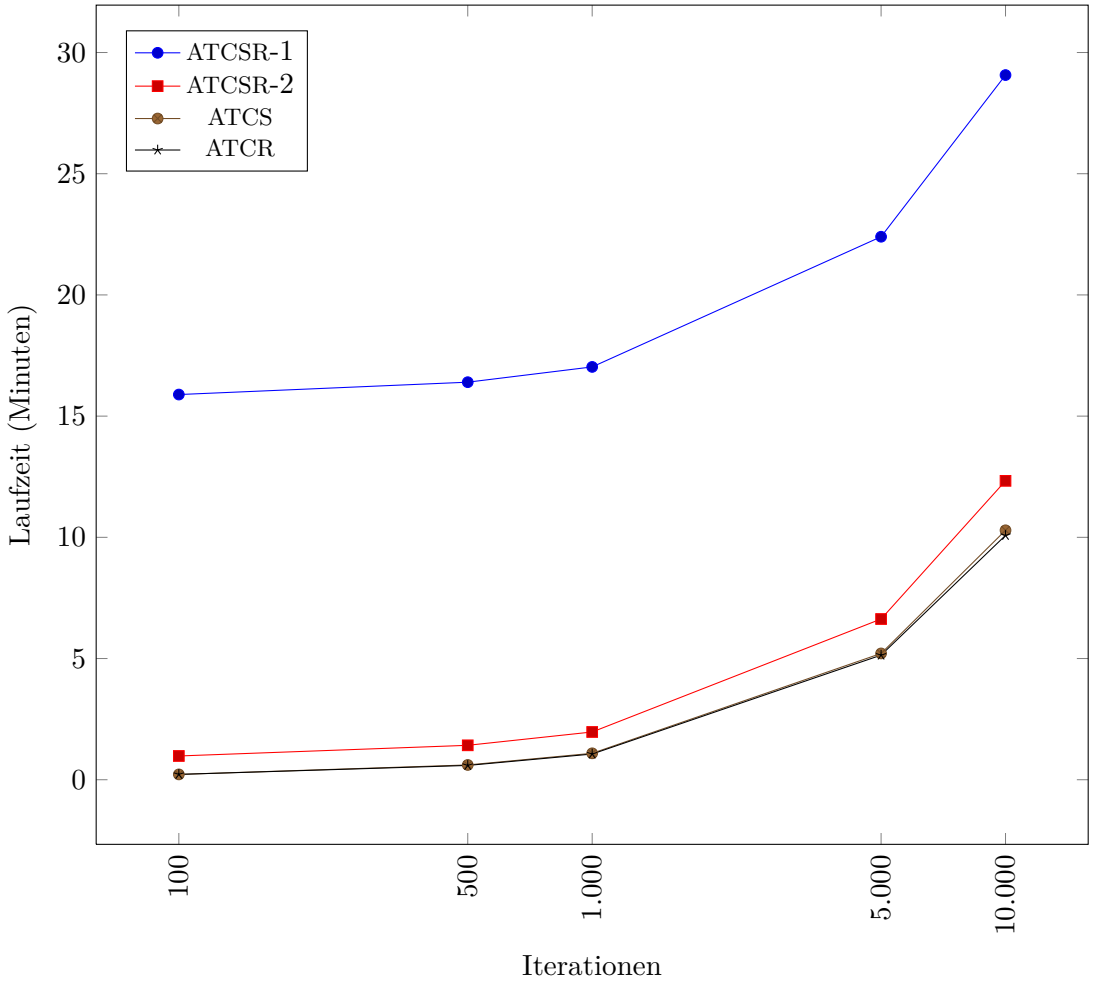

Abbildung 5.1: Mittleres Laufzeitverhalten der vorgeschlagen VNS-Ansätze

vom bisherigen Los J_l auf das Los J_j umzurüsten. Mit $\bar{p}$ wird die durchschnittliche Bearbeitungszeit und mit $\bar{s}$ die durchschnittliche Rüstzeit aller betrachteten Bearbeitungsschritte bezeichnet. Der früheste Verfügbarkeitstermin r_{ij} wird wie die gewünschten Fertigstellungstermine d_{ij}^k bezüglich eines Loses J_k auf Basis von G wie in Abschnitt 4.2 beschrieben berechnet.

Der Prioritätsindex (5.9) ist als Summe einzelner ATCS-Prioritätswerte für die unterschiedlichen gewünschten Fertigstellungstermine d_{ij}^k formuliert. Für den Fall, dass O_{ij} keinen Einfluss auf den Fertigstellungszeitpunkt von J_k hat, ist $d_{ij}^k = \infty$. Der entsprechende Summand wird dann Null und hat damit keinen Einfluss auf die Priorität. Die einzelnen Terme werden mit Hilfe der Skalierungsparameter k_1 und k_2 gewichtet. Die Skalierung des Schlupfterms erfolgt mit Hilfe des Parameters k_1 und der Rüstzeitterm wird mit Hilfe des Parameters k_2 skaliert.

Die Summenformulierung resultiert aus derselben Überlegung wie in Pinedo und Singer

(1999). Folgendes Beispiel soll diese verdeutlichen. Es werden zwei Bearbeitungsschritte O_{11} und O_{12} sowie zwei Lose J_1 und J_2 betrachtet. Beide Bearbeitungsschritte haben die gleichen Parameter und unterscheiden sich nur darin, dass aufgrund von Entscheidungen auf anderen Maschinengruppen O_{11} Einfluss auf den Fertigstellungszeitpunkt von J_1 und O_{12} Einfluss auf die Fertigstellungszeitpunkte von J_1 und J_2 hat. Wenn davon ausgegangen wird, dass die gewünschten Fertigstellungstermine $d_{11}^1 = 5$, $d_{11}^2 = \infty$, $d_{12}^1 = 5$ und $d_{12}^2 = 10$ sowie die Fertigstellungszeitpunkte $c_{11} = c_{12} = 15$ sind, dann hat O_{12} eine höhere Priorität, da sich dadurch zwei Lose verspäten, wohingegen durch O_{11} nur ein Los verspätet wird. In Abbildung 5.2 wird das prinzipielle Verhalten des Prioritätsindex (5.9) in Abhängigkeit von der Rüstzeit s_{lj} und dem Entscheidungszeitpunkt t gezeigt.

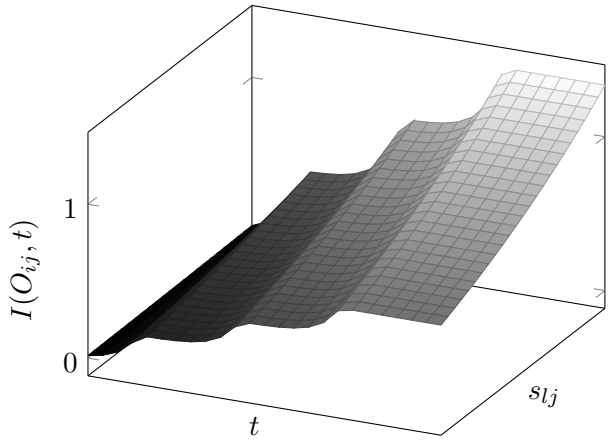

Abbildung 5.2: Prioritätsindex $I(O_{ij}, t)$ in Anlehnung an Pinedo und Singer (1999)

6 Rahmenwerk zur Leistungsbewertung von Ansätzen zur integrierten Ablaufplanung komplexer Produktionssysteme mit automatischem Transport

Während in der Literatur eine Leistungsbewertung der SBH häufig mit Hilfe von statischen Testfällen vorgenommen wird (siehe z. B. Pinedo (2001), Ovacik und Uzsoy (1997), Demirkol u. a. (1997) sowie Mason u. a. (2002)), ist aus neueren Forschungsarbeiten (u. a. Mönch und Rose (2004), Mönch und Drießel (2005), Mönch u. a. (2007) und Sourirajan und Uzsoy (2007)) bekannt, dass die Leistungsfähigkeit von Heuristiken mit Hilfe von Simulationsstudien überprüft werden muss, um zu Aussagen bzgl. der praktischen Anwendbarkeit bei der Steuerung von komplexen Produktionssystemen zu gelangen. Im Unterschied zu statischen Testfällen müssen hier im Rahmen eines rollierenden Ansatzes viele Ablaufplanungsprobleme hintereinander gelöst werden (vgl. auch Abschnitt 4.4). Die Ablaufplanungsprobleme, die zu einem früheren Zeitpunkt gelöst werden, haben dabei einen Einfluss auf die später zu lösenden Ablaufplanungsprobleme. Diese Problematik kann bei der Betrachtung von statischen Testfällen nicht berücksichtigt werden.

In diesem Kapitel wird ein Rahmenwerk zur Bewertung von integrierten Ablaufplanungsverfahren für das Bearbeitungssystem und das Transportsystem eines komplexen Produktionssystems vorgestellt. Zunächst werden die Anforderungen an ein solches Rahmenwerk ausgearbeitet und die Gesamtarchitektur zur Leistungsbewertung vorgestellt. Es werden die wesentlichen Komponenten und ihr Zusammenspiel kurz vorgestelt. Danach wird auf die Kopplung zwischen den einzelnen Teilen des Produktionssystems eingegangen (vgl. auch Abbildung 2.3). Daran anschließend wird das verwendete Datenmodell der Datenschicht als Kern des Rahmenwerks vorgestellt. Das Kapitel schließt mit einer Beschreibung, wie die einzelnen Ablaufplanungsverfahren des Produktionssteuerungssystems implementiert sind.

6.1 Anforderungen an das Rahmenwerk

Das in Mönch u. a. (2002) vorgestellte Rahmenwerk zur Leistungsbewertung von Ablaufplanungsansätzen für komplexen Produktionssystemen wurde bereits erfolgreich in mehreren Arbeiten zur Leistungsbewerzung von Ablaufplanungsansätzen angewendet (unter anderen in Mönch und Drießel (2005), Mönch u. a. (2007) und Mönch und Zimmermann (2007)). Allerdings berücksichtigt dieses Rahmenwerk kein Transportsystem.

Aufbauend auf den Anforderungen für das integrierte Steuerungsverfahren aus Unterabschnitt 3.5.2 hat das Rahmenwerk den folgenden Anforderungen zu genügen:

- Die zugrundeliegenden Bearbeitungs- und Transportbasissysteme sind mit Hilfe von geeigneter Simulationssoftware zu imitieren, wobei es möglich sein soll, das Rahmenwerk mit relativ wenig Aufwand an ein reales MES und MCS anzupassen, um die entwickelten Planungsverfahren in der Realität einzusetzen zu können.
- Da prinzipiell unterschiedliche Ablaufplanungsverfahren untersucht werden sollen, wird eine allgemeine Schnittstelle benötigt, um diese anbinden zu können.
- Hierzu ist es notwendig, dass das interne Modell des Produktionssteuerungssystems durch eine Datenschicht realisiert wird, die vom eigentlichen Planungsverfahren getrennt werden kann.
- Innerhalb der Datenschicht muss ein integriertes Datenmodell vorgehalten werden, das alle zur Steuerung der beiden Basissysteme notwendigen Informationen enthält.
- Es ist sicherzustellen, dass das Datenmodell den jeweils aktuellen Zustand des Produktionsbasissystems widerspiegelt.
- Die Steuerung der zwei Basissysteme hat anhand der durch die Planungsverfahren erzeugten Ablaufpläne zu erfolgen.

6.2 Gesamtarchitektur zur Leistungsbewertung

Das Rahmenwerk ist eine Erweiterung und Anpassung des in Mönch u. a. (2002) vorgestellten Rahmenwerks um die Berücksichtigung des Transportsystems. Es besteht aus den folgenden vier Komponenten, die miteinander gekoppelt sind:

- einem integriertes Planungsverfahren, das Ablaufpläne für die Maschinen und Fahrzeuge des Produktionsbasissystems erzeugt,
- einem Simulationsmodell, welches das Bearbeitungsbasissystem emuliert,
- einem Simulationsmodell zur Emulation des Transportbasissystems und
- einer Datenschicht, die das Bearbeitungsbasissystem, das Transportbasissystem und das Planungsverfahren über ein integriertes Datenmodell miteinander koppelt.

Das Bearbeitungsbasissystem ist innerhalb des Simulators AutoSched AP und das Transportbasissystem mit Hilfe des Simulators AutoMod realisiert. Beide Simulatoren werden von der Firma Brooks Inc. vertrieben. Die Implementierung erfolgte im Wesentlichen in der Programmiersprache C++ und in kleinen Teilen innerhalb der AutoMod-Programmiersprache. Im Fall des Transportbasissystems handelt es sich um ein fahrzeugbasiertes Transportsystem, bei dem die Fahrzeuge an einem Schienensystem an der Decke

fahren. Das Transportbasissystem verbindet die einzelnen Tunnel des Bearbeitungsbasissystems miteinander und übernimmt auch den Transport innerhalb der Tunnel. Das Produktionssteuerungssystem wird durch die Datenschicht und das darauf aufbauende Planungsverfahren gebildet.

Das entwickelte Rahmenwerk setzt auf den Arbeiten von Mönch u. a. (2003) und Drießel und Mönch (2007) auf. Es wurden Datenstrukturen angepasst bzw. hinzugefügt, um das Transportbasissystem geeignet zu berücksichtigen. Das integrierte Datenmodell der Datenschicht stellt ein Objektmodell zur Verfügung, mit dem alle für die Ablaufplanung notwendigen Geschäftsobjekte des Produktionsbasissystems repräsentiert werden können. Die einzelnen Objekte werden dazu ereignisgesteuert aus dem Produktionsbasissystem aktualisiert.

6.3 Kopplung zwischen Bearbeitungsbasissystem, Transportbasissystem und Datenschicht

Zur Kopplung zwischen beiden Simulatoren AutoSched AP und AutoMod wird durch die Firma Brooks Inc. das Model Communication Module (MCM) bereitgestellt. Dies ist im Wesentlichen eine socket-basierte Schnittstelle, über die standardisierte Nachrichten zwischen den beiden Simulatoren ausgetauscht werden (siehe auch Brooks Automation (2001) für eine detaillierte Beschreibung).

Wenn ein Los innerhalb des Bearbeitungsbasissystems von einem Stocker zu einem anderen transportiert werden soll, sendet AutoSched AP eine entsprechende Anfrage an AutoMod. Danach simuliert AutoMod den Transport des Loses. Nachdem das Los erfolgreich transportiert wurde, wird eine entsprechende Nachricht an AutoSched AP gesendet. Während der Simulation des Transports innerhalb des AutoMod-Simulators wird die Simulation innerhalb von AutoSched AP gestoppt.

Die Datenschicht ist als Bibliothek implementiert, die als Erweiterung in den Simulator AutoSched AP geladen wird. Die Datenversorgung erfolgt ereignisgesteuert aus den beiden Simulatoren heraus. Für die Kopplung mit dem Bearbeitungsbasissystem existiert ein entsprechender Adapter, der Callback-Funktionen registriert, welche die folgenden Ereignisse von AutoSched AP behandeln:

- Start der Simulation,
- Einsteuerung eines Loses in das Produktionsbasissystem,
- Einlagerung eines Loses in einen Stocker,
- Hinzufügen eines Loses zur Arbeitsliste einer Maschinengruppe,
- Steuerungsanfrage einer Maschine zur Auswahl von Losen aus der Arbeitsliste,
- Start der Bearbeitung eines Loses auf einer Maschine,
- Beendigung der Bearbeitung eines Loses auf einer Maschine,

- Auslagerung eines Loses aus einem Stocker,
- Aussteuerung eines Loses aus dem Produktionsbasissystem und
- Ende der Simulation.

Die Callback-Funktionen aktualisieren entsprechend der Ereignisse die Daten innerhalb der Datenschicht. Auf diese Weise ist sichergestellt, dass die Datenschicht immer den aktuellen Zustand des Bearbeitungsbasissystems repräsentiert. Es ist zu beachten, dass die Arbeitsliste einer Maschinengruppe alle Lose enthält, welche sich innerhalb des Maschinengruppenstockers befinden und deren nächster Bearbeitungsschritt auf einer der Maschinen dieser Maschinengruppe durchgeführt wird. Innerhalb der Ablaufplanungsliteratur wird dieser Maschinengruppenstocker deshalb auch als Warteraum vor der Maschinengruppe bezeichnet.

Die Kopplung mit dem Transportbasissystem wird ebenfalls über einen entsprechenden Adapter hergestellt, der ereignisgesteuert den Zustand der zugehörigen Geschäftsobjekte innerhalb der Datenschicht aktualisiert. Der Adapter besteht aus einer Erweiterung auf Seiten des AutoMod-Simulators und aus einer Erweiterung auf Seiten der Datenschicht. Die Erweiterung innerhalb des AutoMod-Simulators behandelt die Ereignisse

- Initialisierung des Simulators,
- Start der Fahrt eines Fahrzeugs zum Ausgangsstocker eines Loses (innerhalb AutoMod als retrieve job bezeichnet),
- Start des Transports eines Loses zum Zielstocker (innerhalb AutoMod als deliver job bezeichnet),
- Beendigung der Fahrt eines Fahrzeugs,
- Einlagerung des Loses in den Ausgangsstocker,
- Start der Fahrt eines Fahrzeugs zu einer Parkstation und
- Fahrzeug ist ohne Tätigkeit

und sendet sie per Socket-Verbindung an einen Thread innerhalb der Datenschicht. Hier werden die entsprechenden Objekte der Datenschicht aktualisiert. Die Kopplung der einzelnen Bestandteile innerhalb der Gesamtarchitektur ist in Abbildung 6.1 schematisch dargestellt.

In den beiden Adaptern ist auch die Steuerung des Bearbeitungs- und Transportbasissystems implementiert. Im Falle des Bearbeitungsbasissystems wird immer dann, wenn eine Maschine frei wird bzw. wenn ein Lose in der Arbeitsliste einer Maschinengruppe ankommt für die gewählte Maschine innerhalb des zugehörigen Ablaufplans nachgesehen, ob der nächste Batch bearbeitet werden kann. Wenn die Lose, die den Batch bilden, innerhalb der Arbeitsliste verfügbar sind, wird dem Bearbeitungsbasissystem mitgeteilt, dass die Lose bearbeitet werden sollen. Sind die Lose noch nicht verfügbar,

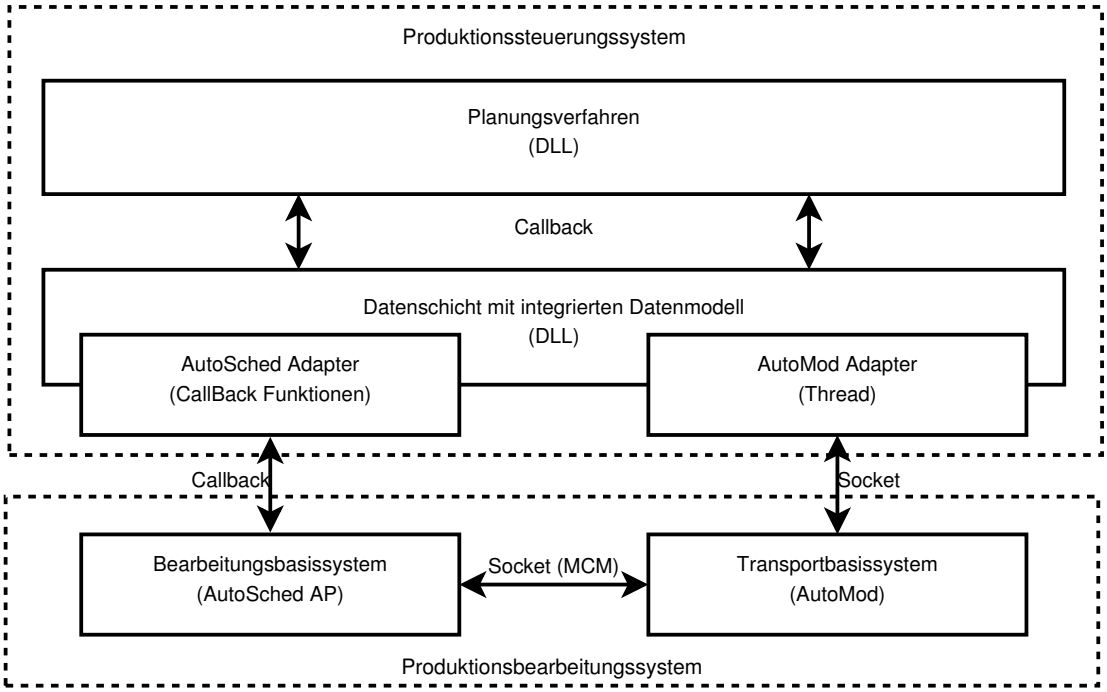

Abbildung 6.1: Architektur zur Leistungsbewertung

dann muss die Maschine warten, bis alle Lose zur Bearbeitung bereit stehen. Im Falle des Transportbasissystems erfolgt dann eine Steuerungsanfrage, sobald ein neues Los in einen Ausgangsstocker eingelagert wird oder ein Fahrzeug den Transport eines Loses zum Zielstocker abgeschlossen hat. Es wird wie beim Bearbeitungssystem innerhalb des Ablaufplans des Fahrzeugs das nächste zu transportierende Los gesucht. Ist dieses bereits verfügbar, wird ein entsprechender Transportauftrag ausgelöst. Ist es nicht verfügbar, muss das Fahrzeug warten. Hierbei ist wichtig zu erwähnen, dass es keine Möglichkeit gibt, innerhalb des Simulators den Ort eines Fahrzeugs direkt zu bestimmen. Das heißt zum einen, dass der Simulator selbstständig entscheidet, welche Route das Fahrzeug wählt und ob das Fahrzeug zu einer Parkstation fährt oder nicht. Zum anderen kann dadurch auch nur abgeschätzt werden, wann der Transport zu Ende ist.

6.4 Datenmodell der Datenschicht

Die Datenschicht stellt Datenstrukturen zur Beschreibung des Produktionsbasissystems zur Verfügung und stellt den Kern der Implementierung dar. Die Klassen zur Beschreibung des Bearbeitungsbasissystems sind in Abbildung 6.2 als UML-Klassendiagramm dargestellt.

Die Klasse Step stellt einen Arbeitsgang innerhalb eines Arbeitsplans dar. Ein Arbeitsplan enthält eine Liste von einzelnen Bearbeitungsschritten, die von bestimmten Maschinen durchgeführt werden können, und wird durch die Klasse Route repräsentiert. Es ist hierbei denkbar, dass ein Arbeitsgang in unterschiedlichen Arbeitsplänen verwendet

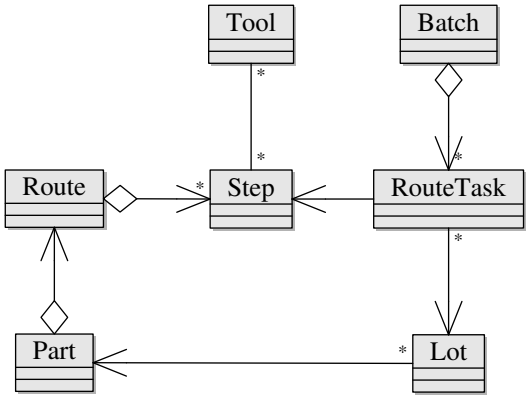

Abbildung 6.2: Entitäten des Bearbeitungssystems

wird. Die Klasse Part repräsentiert ein herzustellendes Gut und ist immer mit einem Arbeitsplan verbunden, der die Herstellung beschreibt. Die Klasse Lot repräsentiert ein Los. Jedes Los umfasst eine bestimmte Menge eines Gutes, das hergestellt werden soll. Die Bearbeitungsschritte der Lose werden durch die Klasse RouteTask abgebildet. Neben dem Verweis auf das jeweilige Los ist auch ein Verweis auf den Arbeitsgang innerhalb des Arbeitsplans vermerkt.

Eine Maschine des Bearbeitungsbasissystems wird durch die Klasse Tool repräsentiert. Jede Maschine ist in der Lage, bestimmte Arbeitsgänge auszuführen. Aus diesem Grund können mehrere Arbeitsgänge einer einzelnen Maschine zugeordnet werden. Ein Ablaufplan für eine Maschine wird als Liste von Batches abgebildet. Jeder Batch wird durch die Klasse Batch beschrieben und kann mehrere Lose umfassen. Für die Abbildung der inkompatiblen Losfamilienen werden die Arbeitsgänge innerhalb der Arbeitspläne verwendet. Ein Batch kann nur gebildet werden, wenn die durchzuführenden Bearbeitungsschritte der enthaltenen Lose zum selben Arbeitsgang (Klasse Step) gehören. Die Bearbeitungszeit für den Batch wird auf Basis des zugehörigen Arbeitsgangs ermittelt. Die Klasse Tool enthält eine Methode, um die Rüstzeit beim Wechsel zwischen verschiedenen Arbeitsgängen zu berechnen. Maschinengruppen sind nicht direkt in dem Datenmodell abgebildet. Sie lassen sich indirekt dadurch ermitteln, indem alle Maschinen, die dieselben Arbeitsgänge ausführen können, zusammengefasst werden. Eine solche Modellierung wird dadurch begründet, nicht identische Maschinen für spätere Forschungen leicht abbilden zu können.

Die verschiedenen Arten von Entitäten des Transportsystems sind in Abbildung 6.3 ebenfalls als Klassendiagramm dargestellt. Ein Transportsystem, repräsentiert durch die Klasse TransportSystem, umfasst Fahrzeuge, Schienen und Stocker. Die Fahrzeuge werden durch die Klasse Vehicle beschrieben. Die Klasse Storage repräsentiert einen Stocker. Innerhalb der Klasse TransportSystem werden die Schienen durch eine Menge von Paaren von Stockern abgebildet. Die Klasse TransportTask entspricht einem Transportschritt und verweist auf den Ausgangs-, den Zielstocker und auf das dazugehörige Los. Die Leer- und Transportfahrten der Fahrzeuge werden durch die Klasse TransportActivity abgebildet. Zur Unterscheidung, welcher Fahrtentyp mit einer einzelnen Instanz be-

schrieben wird, gibt es innerhalb der Klasse TransportActivity ein Attribut type, das entweder den Wert `TransportActivity::Retrieve` für eine Leerfahrt oder den Wert `TransportActivity::Deliver` für eine Transportfahrt annehmen kann. Der Ablaufplan für ein Fahrzeug wird dementsprechend durch eine Liste von einzelnen Fahrten abgebildet. Hierbei ist zu beachten, dass der Transport eines Loses entweder ausschließlich nur durch eine Transportfahrt oder durch eine Leerfahrt mit einer nachfolgenden Transportfahrt durchgeführt wird.

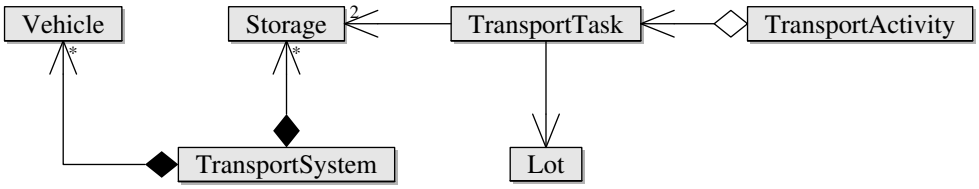

Abbildung 6.3: Entitäten des Transportsystems

6.5 Implementierung der Ablaufplanungsverfahren

Die Ablaufplanungsverfahren sind innerhalb derselben Bibliothek implementiert wie die Datenschicht. Dadurch ist ein direkter und schneller Zugriff auf die entsprechenden Datenstrukturen mittels Zeiger möglich. Es existieren für die einzelnen Problemtypen Klassen, um das jeweilige Problem und die zugehörigen Lösungen zu beschreiben. Desweiteren sind einzelnen Lösungsverfahren ebenfalls in Klassen gekapselt. In diesem Abschnitt wird zunächst die generische Infrastruktur für Optimierungsprobleme vorgestellt. Darauf aufbauend wird auf die Implementierung des Ablaufplanungsproblems für das gesamte Produktionssystem und der SBH eingegangen. Danach erfolgt eine Beschreibung der Implementierung der einzelnen Teilprobleme sowie der zugehörigen Lösungsverfahren.

6.5.1 Generische Infrastruktur für Optimierungsprobleme

Die generische Infrastruktur für Optimierungsprobleme umfasst vier Klassen. Die Templateklasse Problem stellt eine Vorlage für **Problemklassen** dar, die ein Optimierungsproblem beschreiben. Zu jeder dieser Problemklassen existiert auch eine Klasse, welche die Lösung des zugehörigen Optimierungsproblems abbildet. Diese Klasse wird über den Templateparameter *`Solution`* der konkreten Problemklasse mitgeteilt. Innerhalb der Templateklasse Problem existiert in diesem Zusammenhang eine Funktion, mit der sich neue Instanzen der zugeordneten **Lösungsklassen** erzeugen lassen.

Die Templateklasse Solution dient wiederum als Vorlage für die konkreten Lösungsklassen. Jeder Lösung ist ein entsprechendes Problem zugeordnet. Der Templateparameter *`Problem`* dient zur Mitteilung der zugehörigen Problemklasse. Die Templateklasse Solver dient als Vorlage für die einzelnen **Verfahrensklassen**. Jedes Verfahren wird dabei mit der entsprechenden Problem- und Lösungsklasse verknüpft. Die drei virtuellen Funktionen *`initialize`*, *`solve`* und *`finalize`* werden in den Verfahrensklassen überschrieben und

implementieren darin das jeweilige Lösungsverfahren. Dadurch, dass die Klassen Problem, Solution und Solver Templateklassen sind, wird ein typsicherer Zugriff auf die jeweils beteiligten Spezialklassen zur Problem- und Lösungformulierung ermöglicht.

Zur Abbildung der Zielfunktionswerte einer Lösung dient die Klasse Objective. Mit Hilfe dieser Klasse können mehrere Zielfunktionswerte einfach verwaltet werden. In Abbildung 6.4 sind die besprochenen Klassen dargestellt.

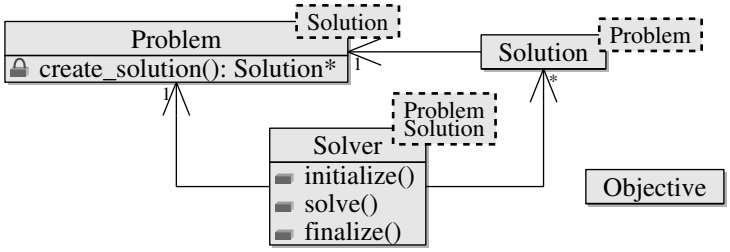

Abbildung 6.4: Generische Infrastruktur für Optimierungsprobleme

6.5.2 Ablaufplanungsproblem für das Produktionssystem

Die betrachteten Probleme vom Typ (2.2) werden mit Hilfe der Klasse FactoryProblem beschrieben. Die Klasse FactorySolution wiederum beschreibt die Ablaufpläne der einzelnen Maschinen und Fahrzeuge. Mit Hilfe dieser Ablaufpläne erfolgt die Steuerung des Bearbeitungs- und Transportbasissystems.

Innerhalb der Datenschicht existiert eine globale Instanz der Klasse FactoryProblem, die das zentrale Element des Objektmodells innerhalb der Datenschicht bildet. Diese Instanz dient dazu, den Zugriff auf alle relevanten Systemobjekte bereitzustellen. Hierzu hat die Klasse FactoryProblem Verweise auf die entsprechenden Instanzen der Klassen Lot, Tool, Vehicle und Storage. Innerhalb der Klasse FactoryProblem wird für jedes Los, jede Maschine und jedes Fahrzeug dessen aktueller Zustand gespeichert. Um dies zu modellieren, werden entsprechende Statusklassen eingeführt, die den einzelnen Entitäten zugeordnet werden. Abbildung 6.5 stellt diesen Zusammenhang dar.

Für jeden Status innerhalb des Produktionsbasissystems existiert eine spezielle Klasse, die von einer der drei Statusklassen abgeleitet ist. Die Losstatusklassen sind die folgenden:

***LotStorageState*:** Hierdurch wird modelliert, dass das zugehörige Los in einen Stocker eingelagert ist.

***LotWaitForProcessSelectionState*:** Dieser Zustand beschreibt, dass das Los in der Arbeitsliste einer Maschinengruppe angekommen ist und auf die Auswahl durch eine Maschine wartet. Innerhalb des Simulators AutoSched AP ist die Arbeitsliste immer einem Stocker zugeordnet. Der Stocker wird als zusätzliches Attribut innerhalb der Klasse modelliert.

***LotWaitForProcessState*:** Ist ein Los, das zu einem Batch gehört, bereits an der Maschine angekommen und wartet allerdings noch auf die restlichen Lose des gewählten

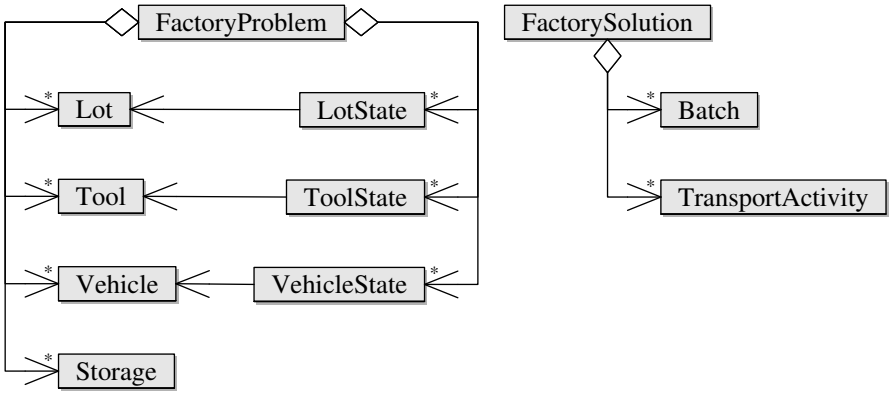

Abbildung 6.5: Ablaufplanungsproblem für das Produktionssystem

Batches, so ist dieses Los im Zustand *LotWaitForProcessState*. Für den Batch existiert ein entsprechendes Attribut innerhalb dieser Klasse.

***LotProcessingState*:** Wird ein Los auf einer Maschine bearbeitet, so ist es im Zustand *LotProcessingState*.

***LotWaitForTransportSelectionState*:** Nachdem das Los von einer Maschine selektiert oder auf einer Maschine fertiggestellt wurde, wechselt es in den Zustand *LotWaitForTransportSelectionState*.

***LotWaitForTransportState*:** Ein Los ist im Zustand *LotWaitForTransport\State*, wenn es zwar bereits einem Fahrzeug zugeordnet ist, allerdings noch nicht auf das Fahrzeug aufgeladen wurde.

***LotTransportingState*:** Diese Klasse repräsentiert den Zustand, dass das Los auf einem Fahrzeug zum Zielstocker transportiert wird.

Abbildung 6.6 zeigt für ein Los die möglichen Übergänge zwischen den einzelnen Zuständen. Es ist dabei zu beachten, dass es für den Zustand *LotStorageState* mehrere Folgezustände geben kann. Befindet sich das Los in einem Maschinengruppenstocker, dann wechselt es sofort in den Zustand *LotWaitForProcessSelectionState*. Ist das Los gerade zur Bearbeitung zu einer Maschine transportiert worden, wechselt es in den Zustand *LotWaitForProcessState* bis die restlichen Lose des zu bearbeitenden Batches ebenfalls zur Maschine transportiert wurden. Befindet sich das Los im Maschinenstocker, nachdem die Bearbeitung auf der Maschine beendet wurde, wechselt das Los sofort in den Zustand *LotWaitForTransportSelectionState*.
Neben der Lossicht werden noch die Sichten für die Ressourcen, das heißt Maschinen und Fahrzeuge, abgebildet. Für eine Maschine existieren die folgenden Statusklassen:

***ToolIdleState*:** Eine Maschine ist im Zustand *ToolIdleState*, wenn sie auf Arbeit wartet.

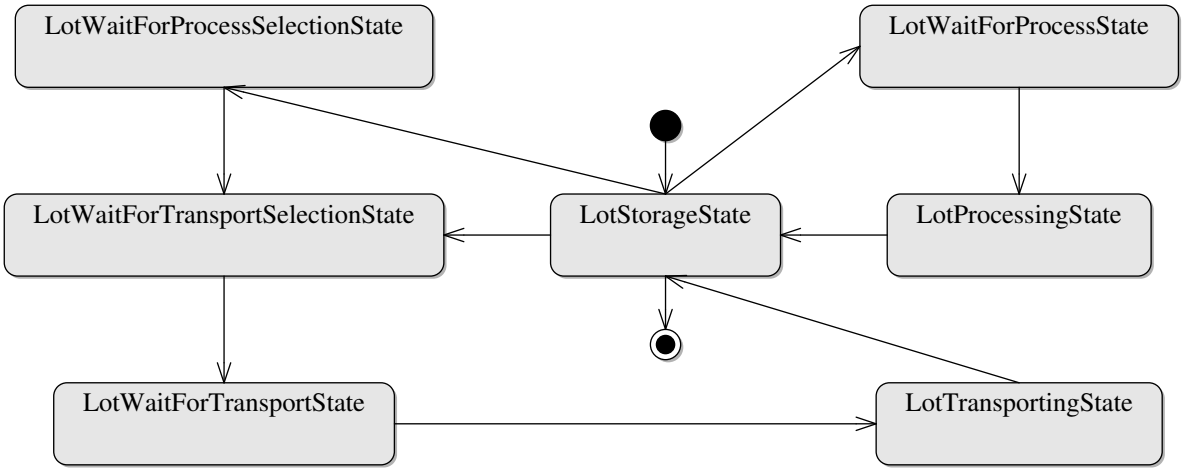

Abbildung 6.6: Zustandsdiagramm für ein Los

***ToolProcessState*:** Die Klasse *ToolProcessState* repräsentiert die Bearbeitung eines Batches auf der Maschine und ist das Gegenstück zum Zustand *LotProcessingState*. Auch hier existiert für den Batch ein entsprechendes Attribut.

***ToolWaitState*:** Hat eine Maschine einen Batch aus der Auftragsliste zur Bearbeitung ausgewählt und wartet darauf, dass die Lose zur Maschine transportiert werden, so ist sie im Zustand *ToolWaitState*. Auf den Batch wird ebenfalls durch ein Attribute verwiesen.

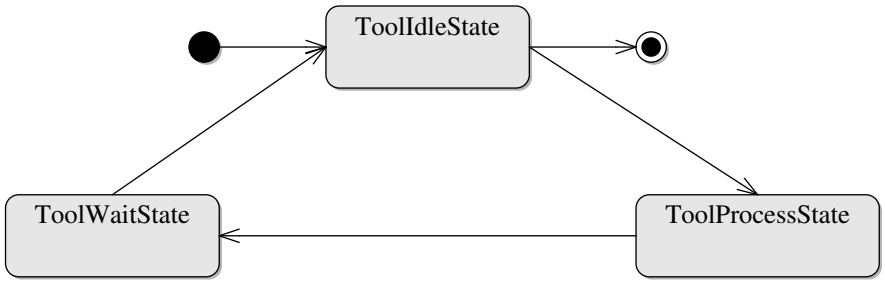

Abbildung 6.7: Zustandsdiagramm für eine Maschine

Abbildung 6.7 zeigt für die möglichen Übergänge zwischen den einzelnen Maschinenzuständen. Das Rüsten, Beladen und Entladen wurde nicht explizit innerhalb der Datenschicht modelliert, da die Abbildung dieser Zustände für das Produktionssteuerungssystem nicht benötigt wird und die Berücksichtigung die Laufzeit der Simulationsstudien verlängert hätte. Sie dem Zustand *ToolProcessState* zugeordnet. Die folgenden Fahrzeugstatusklassen werden verwendet:

***VehicleIdleState*:** Ähnlich des Zustands *ToolIdleState* für eine Maschine ist ein Fahrzeug im Zustand *VehicleIdleState*, wenn es nicht für den Transport eines Loses benötigt wird.

***VehicleParkState*:** Fährt ein Fahrzeug zu seiner Parkstation oder befindet sich dort, dann ist es im Zustand *VehicleParkState*. Dieser Zustand tritt immer dann auf, wenn das Fahrzeug einige Zeit im Zustand *VehicleIdleState* war.

***VehicleRetrieveState*:** Die Klasse *VehicleRetrieveState* stellt eine Leerfahrt des Fahrzeugs zur Aufnahme eines Loses dar. Das zugehörige Los ist hierbei im Zustand *LotWaitForTransportState*.

***VehicleDeliverState*:** Der eigentliche Transport des Loses vom Ausgangs- zum Zielstocker wird durch die Klasse *VehicleDeliverState* abgebildet. Hier befindet sich das zugehörige Los dann im Zustand *LotTransportingState*.

Auch für das Transportsystem wurde mit Rücksicht auf die Laufzeit der Simulationsstudien auf die explizite Abbildung von einzelnen Vorgängen innerhalb der Datenschicht verzichtet. So ist der Zustand des Be- und Enladens der Fahrzeuge und der Stocker dem Zustand *VehicleDeliverState* bzw. dem Zustand *VehicleRetrieveState* zugeordnet. Die möglichen Übergänge zwischen den einzelnen Fahrzeugzuständen sind die Abbildung 6.8 dargestellt.

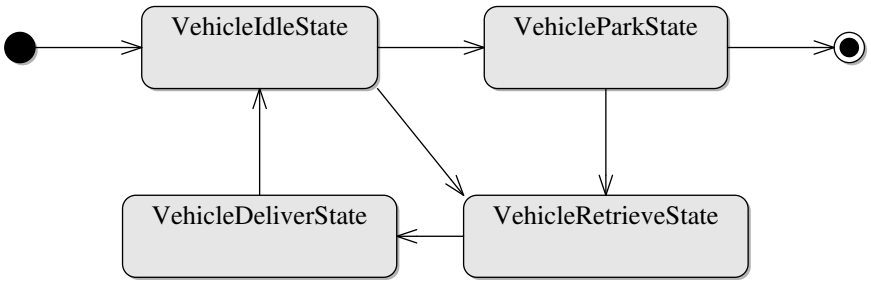

Abbildung 6.8: Zustandsdiagramm für ein Fahrzeug

Der Zusammenhang der einzelnen Status für Lose, Maschinen und Fahrzeuge soll an dieser Stelle durch ein Beispiel erläutert werden. Ausgangspunkt ist das Produktionssystemlayout in Abbildung 6.9 mit zwei Maschinengruppen. Die Maschinen $M1$ und $M2$ bilden die Maschinengruppe A und die Maschinen $M3$ und $M4$ die Maschinengruppe B. Im Stocker $S1$ lagern die noch nicht bearbeiteten Lose. Aufgrund des Arbeitsplans müssen die Lose zunächst von einer Maschine der Maschinengruppe A und danach von einer Maschine der Maschinengruppe B bearbeitet werden. Nach der Bearbeitung auf beiden Maschinen erfolgt eine Einlagerung in den Stocker $S3$. Der Stocker $S1$ ist der Maschinengruppenstocker für die Maschinengruppe A. Der Stocker $S2$ ist der Maschinengruppenstocker für die Maschinengruppe B. Der Transport zwischen den Maschinen wird über ein Schienensystem mit mehreren Fahrzeugen hergestellt. Die möglichen Wege für die Fahrzeuge sind schematisch durch Pfeile dargestellt. Die Parkstation der Fahrzeuge befindet sich bei Stocker $S3$.

Ein Los befindet sich zunächst in dem Stocker $S1$ und ist im Zustand *LotStorageState*. Alle Maschinen sind in dem Zustand *ToolIdleState* und alle Fahrzeuge sind in dem Zustand *VehicleParkState*, das heißt, sie befinden sich in der Parkstation.

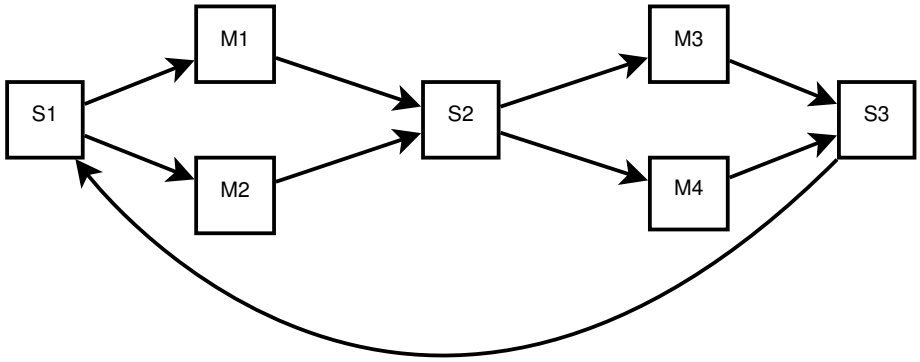

Abbildung 6.9: Varianten des Durchlaufs eines Loses durch das Produktionssystem

Da das Los zunächst auf einer der Maschinen der Maschinengruppe A bearbeitet werden soll, wechselt es unmittelbar in den Zustand *LotWaitForProcessSelectionState*. Damit kann es entweder von der Maschine $M1$ oder der Maschine $M2$ ausgewählt werden. In diesem Beispiel wird davon ausgegangen, dass die Maschine $M1$ als nächstes entscheiden kann, welche Lose sie bearbeitet. Die Maschine wählt aufgrund einer entsprechenden Auswahlregel einen Batch aus einem oder mehreren Losen aus. Nach der Auswahl wechselt die Maschine $M1$ in den Zustand *ToolWaitState*. Für jedes der ausgewählten Lose wird ein Auftrag für den Transport vom Maschinengruppenstocker $S1$ zum zugehörigen Maschinenstocker an das Transportsystem gesendet. Die Lose wechseln in diesem Zusammenhang in den Zustand *LotWaitForTransportSelectionState*.

Nun können die Lose durch die Fahrzeuge des Transportsystems transportiert werden. Nachdem ein Los von einem Fahrzeug selektiert wurde, fährt das Fahrzeug zu dem Maschinengruppenstocker. Das Fahrzeug wechselt dazu in den Zustand *VehicleRetrieveState*. Währenddessen hat das Los den Zustand *LotWaitForTransportState*. Mit dem Aufladen des Loses und während des Transports zum Maschinenstocker ist das Los im Zustand *LotTransportingState*. Das Fahrzeug, welches das Los während dieser Zeit transportiert, ist Zustand *VehicleDeliverState*.

Nach dem Abladen wechselt das Los in den Zustand *LotWaitForProcessState*, bis alle anderen Lose des Batches ebenfalls zum Maschinenstocker transportiert wurden. Das Fahrzeug, welches das Los transportiert hat, wechselt in den Zustand *VehicleIdleState*. Nachdem alle Lose des Batches zur Maschine transportiert sind, wird die Bearbeitung der Lose gestartet. Die Maschine wechselt dabei in den Zustand *ToolProcessState* und die Lose in den Zustand *LotProcessingState*. Die Fahrzeuge, die im Zustand *VehicleIdleState* sind, fahren zu der Parkstation beim Stocker $S3$ und wechseln dadurch in den Zustand *VehicleParkState*.

Nachdem die Bearbeitung beendet ist, werden die Lose wieder in den Maschinenstocker transferiert und sind wieder im Zustand *LotStorageState*. Die Maschine wechselt in den Zustand *ToolIdleState*. Kurz danach werden wieder ein Transportaufträge an das Transportsystem gesendet, um die Lose zum Maschinengruppenstocker $S2$ der Maschinengruppe B zu transportieren. Die Lose wechseln dazu wieder in den Zustand *LotWaitForTransportSelectionState*. Nachdem sie durch die Fahrzeug, die

im Zustand *VehicleIdleState* bzw. *VehicleParkState* sind, selektiert wurden, wechseln sie wieder in den Zustand *LotWaitForTransportState* während die Fahrzeuge im Zustand *VehicleRetrieveState* sind. Während des eigentlichen Transports sind die Lose wieder im Zustand *LotTransportingState* und die Fahrzeug im Zustand *VehicleDeliverState*. Nach dem Abladen der Lose am Maschinengruppenstocker $S2$ sind sie im Zustand *LotStorageState* und die Fahrzeuge im Zustand *VehicleIdleState*.

Die Auswahl, der Transport und die Bearbeitung der Lose für die Maschinengruppe B erfolgt analog zur Maschinengruppe A. Nachdem die Lose fertig bearbeitet wurden, werden sie in den Stocker $S3$ eingelagert. Die einzelnen Zustandsobjekte für die Lose, Maschinen und Fahrzeuge werden auf Basis der in Abschnitt 6.3 beschriebenen Ereignisse erzeugt.

6.5.3 Implementierung der Shifting-Bottleneck-Heuristik

Die Implementierung der SBH erfolgt innerhalb der Klasse SBHSolver, die von der in Unterabschnitt 6.5.1 beschriebenen Templateklasse Solver abgeleitet ist. Der Klasse SBHSolver wird ein Objekt der Klasse FactoryProblem übergeben. Das Ergebnis des Verfahrens ist ein Objekt der Klasse FactorySolution, mit dem das Produktionsbasissystem gesteuert wird.

Innerhalb der Implementierung des Verfahrens wird auf einige Hilfsklassen zugegriffen, wovon die wichtigsten an dieser Stelle kurz erläutert werden sollen. Zentrale Komponente ist die Klasse SBHGraph, die den disjunktiven Graphen G repräsentiert. Zur Realisierung wurde die Boost-Graphenbibliothek verwendet. Diese Templatebibliothek ist kompatibel mit der C++-Standard-Template-Bibliothek und stellt Datenstrukturen für unterschiedliche Graphentypen zur Verfügung. Desweiteren werden häufig benötigte Algorithmen für Breitensuche, Tiefensuche, topologische Sortierung, Berechnungen kürzester Wege usw. bereitgestellt. Einen genauen Überblick zu der Bibliothek liefert Siek u. a. (2002).

Die Klasse SBHGraph setzt auf den Funktionen der Boost-Graphenbibliothek auf, kapselt diese und bietet alle Funktionen an, die zur Manipulation einer disjunktiven Graphenstruktur im Rahmen der SBH notwendig sind. Dazu gehören unter anderem die Berechnung der frühesten Verfügbarkeitszeitpunkte, der gewünschten Fertigstellungstermine sowie die Berechnung der Vorrangbeziehungen zur Zyklenvermeidung (vgl. auch Algorithmus 4.1 in Unterabschnitt 4.2.5). In diesem Zusammenhang stellt die Klasse SBHGraph auch Funktionen zur Verfügung, um die in Abschnitt 4.3 behandelten Teilgraphen für die einzelnen Teilprobleme zu erzeugen. Diese werden im Fall der Maschinengruppenteilprobleme durch die Klasse RouteTaskPlan und im Fall des Transportteilproblems durch die Klasse TransportTaskPlan beschrieben. Intern verwenden diese zwei Klassen ebenfalls Datenstrukturen der Boost-Graphenbibliothek.

Die verschiedenen Arten von Knoten des disjunktiven Graphen sind in der Klasse SBHNode gekapselt. Es wird dabei jeweils auf das entsprechende Objekt in der Datenschicht verwiesen. Ein Knoten repräsentiert dabei entweder

- einen Bearbeitungsschritt (Verweis auf ein Objekt der Klasse RouteTask),
- einen Transportschritt (Verweis auf ein Objekt der Klasse TransportTask),

- einen Batch-Bildungsknoten (Verweis auf ein Objekt der Klasse Batch),
- einen Batch-Bearbeitungsknoten (Verweis auf ein Objekt der Klasse Batch),
- einen Startknoten oder
- einen Endknoten (Verweis auf ein Objekt der Klasse Batch).

Auf die Knoten gibt es prinzipiell zwei Arten von Sichten. Zum einen ist dies die Lossicht in Form einer Liste aller Knoten, die zu einem Los gehören. Diese Sicht wird in der Klasse SBHRoute gekapselt. Diese Klasse repräsentiert den Teil des Arbeitsplans eines Loses, der innerhalb von G berücksichtigt wird. Zum anderen lassen sich die Knoten entsprechend ihrer Zugehörigkeit zu einem Teilproblem gruppieren. Diese Sicht wird durch die Klasse SBHProblem gekapselt.

Innerhalb der SBH werden auf Basis des G Teilprobleme erzeugt und durch die jeweiligen Verfahren gelöst (vgl. Abschnitt 4.3). Die Teilprobleme für eine Maschinengruppe werden hierbei durch die Klasse ProductionProblem abgebildet. Die zugehörigen Ablaufpläne werden durch die Klasse ProductionSolution beschrieben. Die entsprechenden Klassen für das Transportteilproblem heißen TransportProblem und TransportSolution. Abbildung 6.10 stellt den Zusammenhang zwischen der Klasse SBHSolver und den anderen Klassen dar.

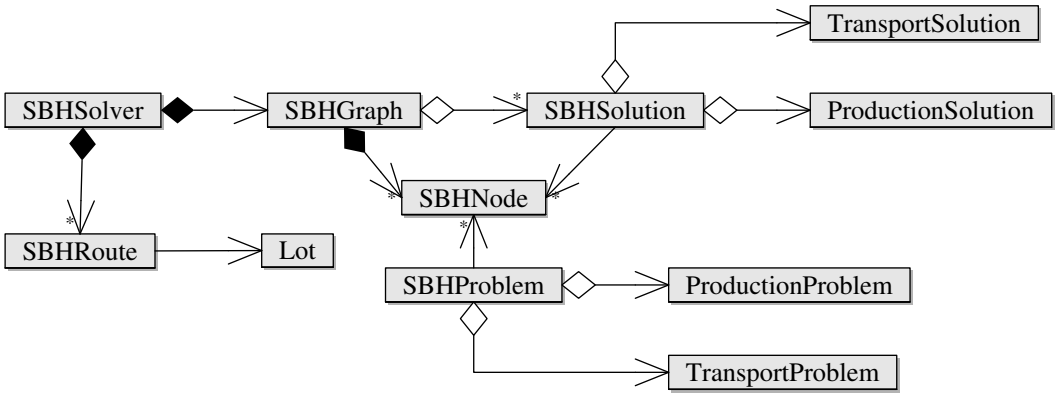

Abbildung 6.10: Verfahrensklassen für die Shifting-Bottleneck-Heuristik

Die Erzeugung eines SBHSolver-Objektes erfolgt immer nach Ablauf eines gegebenen Aufrufintervalls im Rahmen der Steuerungsanfrage für eine Maschine. Beträgt das Aufrufintervall zum Beispiel zwei Stunden, erfolgt zu Beginn der Simulation zunächst eine Berechnung der Ablaufpläne für die Maschinen und Fahrzeuge mit Hilfe eines SBHSolver-Objektes. Hierzu wird bei der Erzeugung das Objekt mit Hilfe des Datenmodells initialisiert. Nach erfolgreicher Berechnung liegt eine Instanz der Klasse FactorySolution vor. Mit Hilfe der Ablaufpläne in dieser Instanz erfolgt danach die Steuerung des Produktionsbasissystems für die nächsten zwei Stunden. Nach Ablauf der zwei Stunden erfolgt wiederum eine neue Planung.

Die SBH, die durch die Klasse SBHSolver implementiert ist, lässt sich durch verschiedene Parameter anpassen. Ist der Parameter *consider_transport* gesetzt, so werden die Verfahrenserweiterungen für das Transportsystem verwendet. Der Parameter *simultaneous_transport* legt in diesem Fall fest, ob die simultane oder die hierarchische Variante gewählt wird. Durch den Parameter *forecast* wird bestimmt, wie groß der Planungshorizont ist (siehe auch Abschnitt 4.4).

Zur Lösung der Maschinengruppenteilprobleme wird das Verfahren aus Abschnitt 5.2 verwendet. Die einzelnen zu verwendenden Prioritätsregeln werden der Klasse über den Parameter *ssp_dispatcher* mitgeteilt. Die Prioritätsregeln werden sowohl für alle Maschinengruppenteilprobleme als auch für das Transportteilproblem verwendet. Für das Transportteilproblem gibt der Parameter *improvement_iterations* die Anzahl der Züge für VNS an.

6.5.4 Teilprobleme innerhalb der Shifting-Bottleneck-Heuristik

Wie bereits beschrieben, erfolgt die Abbildung des Teilproblems für die einzelnen Maschinengruppen mit Hilfe der Klasse ProductionProblem. Die zugehörigen Ablaufpläne der einzelnen Maschinen sind in der Klasse ProductionSolution zusammengefasst. In Abbildung 6.11 sind die beteiligten Klassen dargestellt.

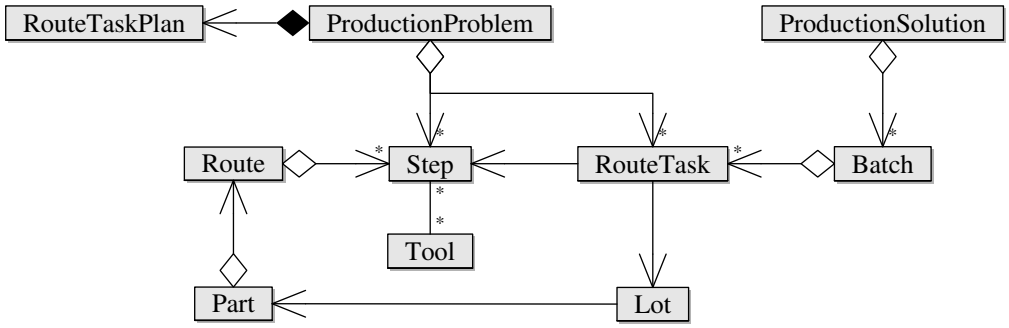

Abbildung 6.11: Belegungsproblem für parallele Maschinen

Einem Objekt der Klasse ProductionProblem sind alle Arbeitsplanschritte zugeordnet, die zu der Maschinengruppe gehören. Auf Basis dieser Information können ebenfalls die zugehörigen Maschinen ermittelt werden. Die Klasse enthält für jede Maschine den nächsten Verfügbarkeitszeitpunkt sowie den aktuellen Rüstzustand. Weiterhin enthält die Klasse den disjunktiven Teilgraphen aller zugehörigen Bearbeitungsschritte auf Basis der Klasse RouteTaskPlan. Die Klasse ProductionSolution enthält, wie die Klasse FactorySolution, für jede Maschine eine Liste von Batches.

Zur Beschreibung des Transportteilproblems dient die Klasse TransportProblem. Die zugehörige Lösungsklasse ist TransportSolution. Das zugehörige Transportsystem wird innerhalb der Instanz der Klasse TransportProblem registriert. Über dieses ist der Zugriff auf die Fahrzeuge, Stocker und Schienen möglich. Ähnlich der Klasse ProductionProblem wird für jedes Fahrzeug der nächste Verfügbarkeitszeitpunkt sowie der aktuelle Stocker gespeichert. Weiterhin werden alle Transportschritte als disjunktiven Teilgraphen auf

Basis der Klasse TransportTaskPlan abgebildet. Abbildung 6.12 zeigt die beteiligten Klassen. Die Klasse TransportSolution speichert den Ablaufplan für jedes Fahrzeug als Liste von Instanzen der Klasse TransportActivity.

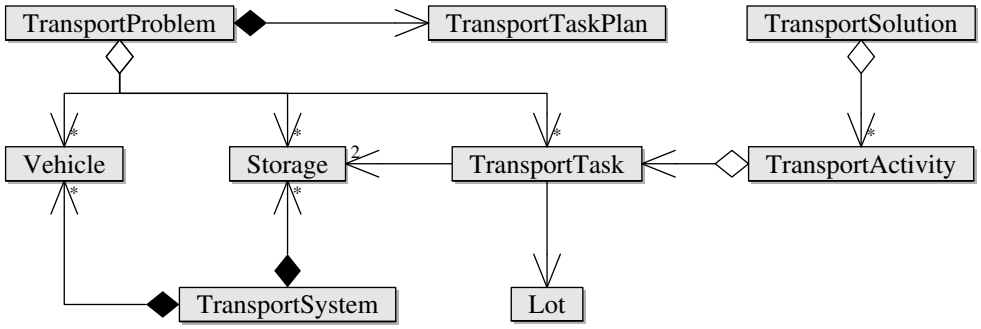

Abbildung 6.12: Transportteilproblem für mehrere Fahrzeuge

Innerhalb des Produktionssteuerungssystems existiert eine Funktion, um ein Objekt vom Typ TransportProblem in ein Objekt vom Typ ProductionProblem umzuwandeln. Weiterhin gibt es eine Funktion zur Umwandlung eines ProductionSolution-Objektes in ein entsprechendes TransportSolution-Objekt. Hierduch ist es möglich, wie in Kapitel 5 beschrieben, die Verfahren für die Maschinengruppenteilprobleme auch auf das Transportteilproblem anzuwenden.

6.5.5 Implementierung der Verfahren für die Teilprobleme

Innerhalb des Produktionssteuerungssystems sind die Verfahren aus Abschnitt 5.2 und 5.3 implementiert. Aufgrund der Möglichkeit der Umwandlung eines Transportteilproblems in ein Maschinengruppenteilproblem wurden nur Verfahren für die Klassen ProductionProblem und ProductionSolution implementiert. Es ist jedoch ohne weiteres möglich, entsprechende Verfahren direkt für die anderen Problem- und Lösungsklassen zu implementieren.

Die Klasse MultiSolver stellt eine einfache Möglichkeit bereit, für ein gegebenes Problem verschiedene Verfahren parallel rechnen zu lassen. Die erzeugten Lösungen werden miteinander verglichen und die beste Lösung wird als Ergebnis zurückgegeben. Um die Lösung zu bewerten muss, die virtuelle Funktion *objective()* in der Kindklasse überschrieben werden. Dieser Ansatz wird zusammen mit der Klasse ProductionSimulationSolver verwendet, um die Maschinengruppenteilprobleme zu lösen. Die Klasse ProductionSimulationSolver implementiert dabei Algorithmus 5.1. Die Prioritätsregeln sind dabei in Klassen implementiert, die von der Klasse ToolDispatcher abgeleitet sind. So werden zum Beispiel die unterschiedlichen ATC-Varianten durch die Klasse ToolATCDispatcher implementiert. In Abbildung 6.13 sind die beteiligten Klassen dargestellt.

Abbildung 6.14 stellt die Verfahrensklassen für VNS dar. Zentrale Verfahrensklasse ist VNSProductionSolver, welche die einzelnen VNS-Varianten implementiert. Wie in Unterabschnitt 3.4.3 beschrieben, setzt VNS auf lokalen Suchverfahren auf. In allen lokalen Suchverfahren wird, ausgehend von einer gegebenen Lösung, innerhalb einer bestimmten

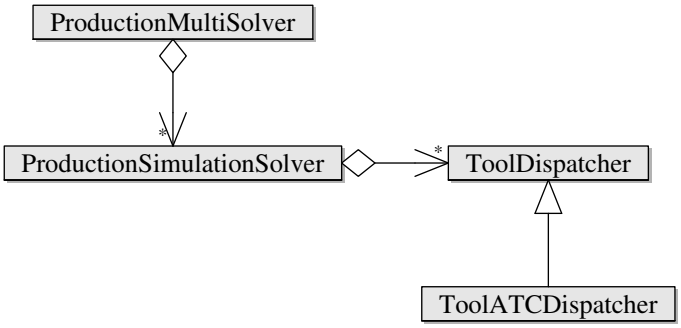

Abbildung 6.13: Verfahrensklassen für die Lösung von Maschinengruppenteilproblemen mit Hilfe von Prioritätsregeln

Nachbarschaft nach besseren Lösungen gesucht. Die Nachbarschaft ist typischerweise dadurch definiert, dass in einem Zug bestimmte Modifikationen an einer Lösung durchgeführt werden. Die Lösungen, das heißt Ablaufpläne, sind in der Klasse VNSProductionSolution gekapselt. Diese Klasse stellt entsprechende Modifikationsmöglichkeiten zur Verfügung.

Die einzelnen VNS-Varianten sind als Templatefunktionen implementiert, um sie möglichst generisch verwenden zu können. Die Modifikationen, die innerhalb der jeweiligen Templatefunktionen an einer Lösung durchgeführt werden können, werden durch Klassen beschrieben, die von der Templateklasse Movement abgeleitet sind. Der Templateparameter *Solution* übernimmt hier die verwendete Lösungsklasse VNSProductionSolution. Die zwei Modifikationen, die in Unterabschnitt 5.3.4 vorgeschlagen wurden, sind in den Klassen VNSMoveTask sowie VNSSwapTask implementiert.

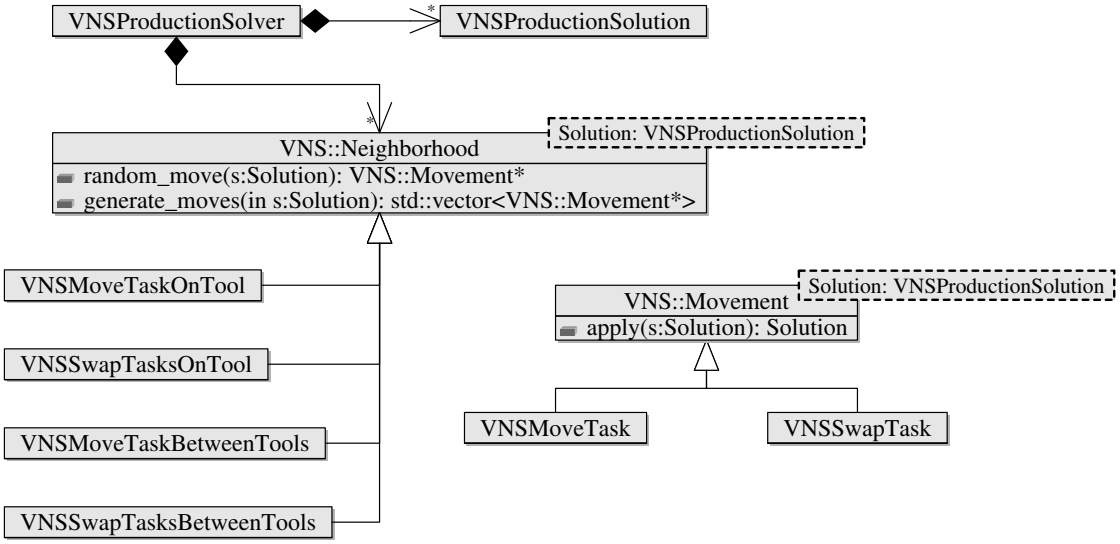

Abbildung 6.14: Verfahrensklassen für VNS

Innerhalb der Templateklasse Movement existiert die virtuelle Funktion *apply*, die von den beiden Klassen VNSMoveTask und VNSSwapTask überschrieben wird, um die

Modifikation an der übergebenen Lösung auszuführen. Die Templateklasse Neighborhood repräsentiert die einzelnen Nachbarschaften. Wie bei der Templateklasse Movement nimmt der Templateparameter die Lösungsklasse VNSProductionSolution auf. Die implementierten Nachbarschaften sind ebenfalls in Abbildung 6.14 dargestellt. Zentral sind die beiden virtuellen Funktionen *random_move* und *generate_moves*. Diese müssen von den abgeleiteten Klassen überschrieben werden, damit die entsprechenden Modifikationen erzeugt werden.

7 Leistungsbewertung der vorgeschlagenen Verfahren

In diesem Kapitel erfolgt die Leistungsbewertung der SBH für komplexe Produktionssysteme mit automatischem Transport innerhalb eines dynamischen Umfelds auf Basis von Simulationsmodellen. Die Abbildung des Produktionsbasissystems erfolgt, wie bereits in Kapitel 6 beschrieben, mit Hilfe der Simulatoren AutoSched AP und AutoMod. Die Aktualisierung des Datenmodells und die Steuerung der Simulatoren erfolgt ereignisgesteuert.

Im Unterschied zu klassischen Job-Shop-Scheduling-Problemen sind für komplexe Produktionssysteme keine Testinstanzen mit bekannten Lösungen verfügbar (vgl. hierzu auch die Diskussion in Mönch u. a. (2011)). Aufgrund dieser Tatsache erfolgt die Leistungsbewertung mit Hilfe von Prioritätsregeln. Zunächst werden die verwendeten Simulationsmodelle und die Versuchsplanung beschrieben. Im Anschluss werden die Ergebnisse der Simulationsstudien dargestellt. Es lässt sich zeigen, dass die vorgestellte SBH den Vergleichsregeln in vielen Fällen überlegen ist.

7.1 Verwendete Simulationsmodelle

Es werden zwei unterschiedliche Simulationsmodelle betrachtet. Das erste Modell basiert auf einem erweiterten MiniFab-Modell, das von der Intel Corporation vorgeschlagen wurde und in Spier und Kempf (1996) beschrieben ist. Dieses Modell wird im weiteren Verlauf als **Modell A** bezeichnet. Jede Maschinengruppe des MiniFab-Modells wurde für das Modell A dreimal dupliziert. Insgesamt liegen innerhalb des Modells 24 Maschinen vor, die in neun Maschinengruppen organisiert sind. Wie im Originalmodell wird ein Produkt gefertigt. Der Arbeitsplan ist allerdings viermal so lang wie der Originalarbeitsplan des MiniFab-Modells und umfasst 24 Arbeitsgängen mit Bearbeitungszeiten zwischen zehn Minuten und 4,25 Stunden (vgl. Tabelle 7.1). Jede Maschinengruppe wird hierbei zwischen zwei- und viermal von den Losen besucht. Die Maschinen jeder Maschinengruppe befinden sich jeweils in einem eigenen Tunnel. Das Layout des Transportsystems ist in Abbildung 7.1 dargestellt. Die Rechtecke repräsentieren die Position der einzelnen Stocker bzw. Maschinen. Alle Maschinen sind durch ein Schienentransportsystem an der Decke verbunden. Der Transport der Lose erfolgt durch fünf identische Fahrzeuge.

Das zweite Modell ist eine verkleinerte Variante des MIMAC-Testdatensatzes 1 (vgl. Fowler und Robinson (1995)). Es enthält zwei stark zyklische Arbeitspläne mit 56 bzw. 66 Arbeitsgängen. Die Lose werden auf 45 Maschinen bearbeitet, die in zwölf Maschinengruppen unterteilt sind. Die einzelnen Maschinengruppen werden dabei bis zu 16-mal in ein und demselben Arbeitsplan benötigt. Die Bearbeitungszeiten für die Arbeitsgänge

Tabelle 7.1: Arbeitsplan für Modell A

Arbeitsgang	Maschinengruppe	Arbeitsgang	Maschinengruppe
S01	TGA–1	S13	TGA–1
S02	TGA–2	S14	TGA–2
S03	TGA–3	S15	TGA–3
S04	TGA–2	S16	TGA–2
S05	TGA–1	S17	TGA–1
S06	TGA–3	S18	TGA–3
S07	TGB–1	S19	TGC–1
S08	TGB–2	S20	TGC–2
S09	TGB–3	S21	TGC–3
S10	TGB–2	S22	TGC–2
S11	TGB–1	S23	TGC–1
S12	TGB–3	S24	TGC–3

bewegen sich zwischen ca. einer Minute und bis zu 24 Stunden. Unter den Maschinen sind sowohl Batch-Maschinen als auch Maschinen, die nur ein Los gleichzeitig bearbeiten können. Das Transportsystem ähnelt dem ersten Modell. Wie im Modell A sind die einzelnen Maschinen durch ein Schienentransportsystem verbunden. Der Transport der Lose erfolgt hier allerdings durch zehn identische Fahrzeuge. Dieses Modell wird im weiteren Verlauf als **Modell B** bezeichnet. Die wichtigsten Parameter der beiden Modelle sind in Tabelle 7.2 dargestellt.

Tabelle 7.2: Gegenüberstellung der verwendeten Simulationsmodelle

Elemente	Modell A (Anzahl)	Modell B (Anzahl)
Maschinengruppen	9	12
davon Batch-Maschinengruppen	3	7
Maschinen	24	46
Anzahl der Produkte	1	2
Arbeitsgänge der Produkte	24	56 und 66
Anzahl der Fahrzeuge	5	10

Innerhalb der Transportbasissysteme wurden die AutoMod-Standardwerte verwendet. So haben die Fahrzeuge eine Maximalgeschwindigkeit von 1 m/s und eine Beschleunigung bzw. Abbremsverzögerung von $0{,}3\,\mathrm{m/s^2}$. Die Maschinen stehen in beiden Modellen zwischen 3 m und 15 m voneinander entfernt. Ausgehend von einer gegebenen Anzahl von Losen innerhalb des Bearbeitungssystems (WIP = Work in Progress) werden insgesamt 40 Tage vom beginnend vom 1. Januar des Jahres 2010 simuliert. Alle Modelle sind komplett deterministisch. Die Maschinen laufen 24 h am Tag. Es werden außerdem keine Stillstandzeiten der Maschinen oder Fahrzeuge berücksichtigt.

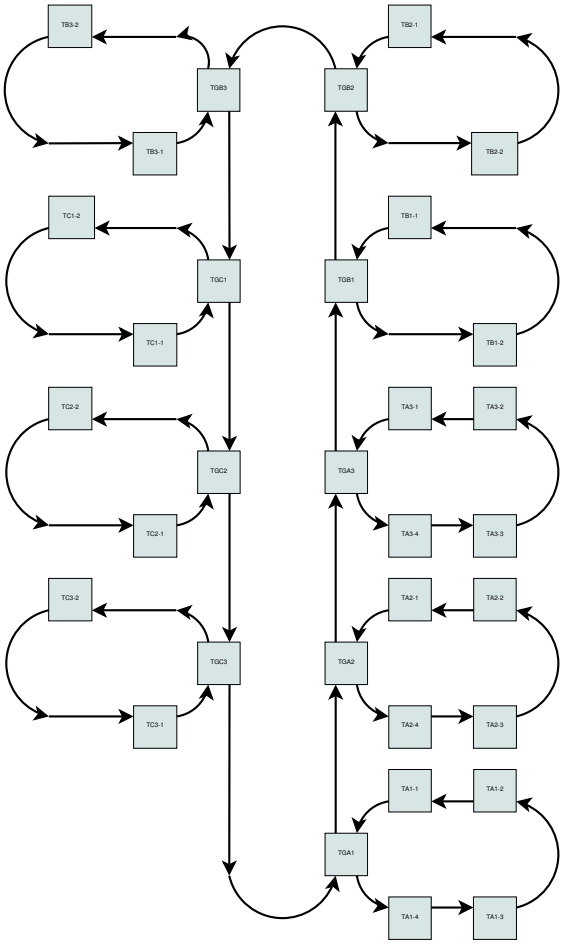

Abbildung 7.1: Layout des Transportsystems für Modell A

7.2 Betrachtete Leistungsmaße und verwendete Teilproblemlöser

Es werden verschiedene Leistungsmaße betrachtet. Das primäre Leistungsmaß ist, wie bereits in Kapitel 2 beschrieben, die TWT. Als zweites Leistungsmaß ist die durchschnittliche Zykluszeit (ACT = Average Cycle Time) der einzelnen Lose von Interesse. Die durchschnittliche Zykluszeit wird wie folgt berechnet:

$$ACT = \frac{\sum_{j=1}^{n} c_j - r_j}{n}. \tag{7.1}$$

In Formel 7.1 stellt n die Anzahl der fertiggestellten Lose nach Abschluss eines Simulationslaufs dar. Weiterhin wird noch der Durchsatz, das heißt die Anzahl der fertiggestellten Lose (TP = Throughput) betrachtet. Die zu Beginn der Simulation bereits vorhandenen Lose innerhalb des Bearbeitungssystems werden in die Berechnung der Leistungsmaße

nicht mit einbezogen. Es werden nur die neu eingesteuerten Lose während des Simulationslaufes berücksichtigt.

Es wird sowohl für die Maschinengruppenteilprobleme als auch für das Transportteilproblem ein Teilproblemlöser auf Basis der in Abschnitt 5.6 beschrieben modifizierten ATCS-Prioritätsregel verwendet. Dieser wird im weiteren Verlauf mit ATC-SSP bezeichnet (SSP = Sub Solution Procedure). Die Ermittlung des Ablaufplans erfolgt dabei mit Hilfe einer Gittersuche über die Parameter k_1 und k_2. Aufgrund der Ergebnisse aus Kapitel 5 werden die folgenden Parameterwerte verwendet:

- $k_1 \in \{0{,}01; 0{,}15; 0{,}5; 1{,}0; 1{,}5\}$,
- $k_2 \in \{0{,}1; 0{,}7; 1{,}3; 1{,}9\}$.

Neben ATC-SSP wird zu Vergleichszwecken ein weiterer Teilproblemlöser auf Basis der FIFO-Prioritätsregel mit $I(O_{ij}, t) \coloneqq t - r_{ij}$ verwendet (FIFO-SSP). Für das Transportteilproblem wird der erzeugte Ablaufplan optional noch mit F-VNS (vgl. Unterabschnitt 5.3.5) verbessert. Die einzelnen Unterproblemlöser ATCS sowie F-VNS wurden dabei aufgrund der Ergebnisse des Kapitels 5 gewählt.

7.3 Versuchsplanung

Auf Basis früherer Untersuchungen und vorangegangen Experimenten wird erwartet, dass das Aufrufintervall τ_Δ bzw. der Planungshorizont $\tau_\Delta + \alpha_\Delta$, die verwendeten Teilproblemlöser, die Art der Berücksichtigung des Transportteilproblems sowie die zur Verfügung gestellte Zeit für das VNS-Verfahren für das Transportteilproblem einen Einfluss auf die Gesamtleistung des Steuerungsansatzes haben (vgl. auch Mönch und Drießel (2005), Mönch u. a. (2007) sowie Unterabschnitt 5.5.2).

Die zwei Varianten der SBH SBH-ST und SBH-HT werden untersucht. Es werden drei Aufrufintervalle $\tau_\Delta = 2\,\text{h}$, $\tau_\Delta = 4\,\text{h}$ und $\tau_\Delta = 6\,\text{h}$ mit einem zusätzlichen Planungshorizont von $\alpha_\Delta = 1\,\text{h}$ betrachtet. Die Abschätzung der voraussichtlichen frühesten Start- und Fertigstellungstermine erfolgt mit Hilfe des in Abschnitt 4.4 vorgestellten MICA-Verfahrens.

Bezüglich der Unterproblemlöser wird zunächst zwischen FIFO-SSP und ATC-SSP unterschieden. Ob für das Transportteilproblem optional noch F-VNS verwendet wird, hängt von der Wahl des Parameters *improvement_iterations* ab, der die Anzahl der Züge, die für die Verbesserung vorgesehen sind, angibt. Dieser Parameter wird entweder auf 0 gesetzt oder auf 100. Ist dieser Parameter auf 0 gesetzt, findet keine zusätzliche Verbesserung durch F-VNS statt und das Teilproblem wird nur unter Verwendung der Prioritätsregeln gelöst.

Es wird erwartet, dass neben den Parametern der SBH die geplanten Aussteuertermine, die Gewichtung der Lose und die Last innerhalb des Bearbeitungssystems einen Einfluss auf die Ergebnisse haben. Die geplanten Aussteuertermine der Lose werden über einen gegebenen Flussfaktor berechnet. Hierbei wird zum geplanten Einsteuerzeitpunkt die erwartete Durchlaufzeit hinzugefügt. Diese erwartete Durchlaufzeit eines Loses wird durch

den Flussfaktor als Vielfaches der Bearbeitungszeit des Loses angegeben. Der geplante Aussteuertermin ergibt sich wie folgt:

$$d_j = FF \sum_{i=1}^{n_j} p_{ij} + r_j. \tag{7.2}$$

Mit p_{ij} wird die Bearbeitungszeit des jeweiligen Bearbeitungsschritts und mit n_j die Anzahl der Bearbeitungsschritte des Loses J_j bezeichnet. Innerhalb der Experimente werden zwei verschiedene Flussfaktoren verwendet. Im ersten Fall werden enge geplante Aussteuertermine ($FF = 1{,}6$) und im zweiten Falle werden weite geplante Aussteuertermine ($FF = 2{,}0$) betrachtet.

Für das Gewicht der Lose werden zwei Schemen verwendet. Die diskrete Verteilung D_1 stellt eine Situation dar, in der eine große Menge von Losen ein kleines bzw. mittleres Gewicht haben. Die Verteilung D_1 ist wie folgt definiert:

$$D_1 := \begin{cases} w_j = 1 & \text{mit } p_1 = 0{,}50 \\ w_j = 5 & \text{mit } p_2 = 0{,}35 \\ w_j = 10 & \text{mit } p_3 = 0{,}15. \end{cases} \tag{7.3}$$

Die Verteilung D_2 beschreibt eine Situation, in der nur sehr wenige Lose ein sehr hohes Gewicht haben. D_2 ist wie folgt definiert:

$$D_2 := \begin{cases} w_j = 1 & \text{mit } p_1 = 0{,}50 \\ w_j = 2 & \text{mit } p_2 = 0{,}45 \\ w_j = 10 & \text{mit } p_3 = 0{,}05. \end{cases} \tag{7.4}$$

Die zwei Verteilungen wurden bereits in Mönch und Drießel (2005) angewendet und dienen dazu, einem Los bei dessen Erzeugung ein Gewicht mit der entsprechenden Wahrscheinlichkeit auf Basis der Verteilungen zuzuweisen.

Es werden zwei unterschiedliche Lastsituationen betrachtet. Die Auslastung lässt sich dabei über die Menge der eingesteuerten Lose regulieren. In einer Hochlastsituation werden innerhalb eines gegebenen Zeitintervalls mehr Lose in das Bearbeitungssystem eingesteuert als in einer Niedriglastsituation. Die Anzahl der anfänglichen Lose innerhalb des Bearbeitungssystems beträgt bei Modell A 47 Lose für die Niedriglastsituation und 99 Lose für die Hochlastsituation. Für Modell B liegt die Last bei 80 bzw. 129 Losen.

Für jede Kombination der Faktoren, welche das Produktionsbasissystem beschreiben, werden zu Vergleichszwecken Experimente auf Basis der FIFO- bzw. der ATC-Prioritätsregel für die Maschinen und der NLF-Prioritätsregel für die Fahrzeuge durchgeführt. Bei der NLF-Prioritätsregel wird das Los als nächstes für den Transport selektiert, das der aktuellen Fahrzeugposition am nächsten ist. Es ist bekannt, dass die NLF-Prioritätsregel gute Ergebnisse im Hinblick auf den Durchsatz in die Transportzeit liefert (vgl. Liao und Fu (2004)). Der verwendete ATC-Index wird wie folgt berechnet:

$$I(O_{ij},t) := \frac{w_j}{p_{ij}} e^{-\frac{\max(d_{ij}-p_{ij}-\max(r_{ij},t),0)}{k\bar{p}}}. \tag{7.5}$$

Hierbei wird der früheste Verfügbarkeitszeitpunkt r_{ij} und der gewünschte Fertigstellungstermin d_{ij} mit Hilfe von MICA ermittelt. Mit $\bar{p}$ wird die durchschnittliche Bearbeitungszeit der durchzuführenden Bearbeitungsschritte der wartenden Lose vor der Maschinengruppe bezeichnet. Der verwendete Skalierungsparameter ist $k := 0,5$. Die ATC-Regel wird an dieser Stelle verwendet, da aus Mönch und Zimmermann (2011) bekannt ist, dass diese Prioritätsregel zu kleinen TWT-Werten führt. Die resultierenden kombinierten Steuerungsstrategien werden mit FIFO-NLF bzw. ATC-NLF bezeichnet.

Für das Simulationmodell A wird für jede Faktorkombination ein Experiment durchgeführt. Aufgrund des hohen Rechenbedarfs wird für das Simulationsmodell B auf einige Experimente verzichtet. Es werden nur die diskrete Verteilung D_2 und die hierarchische SBH-Variante untersucht. Das faktorielle Design der Versuchsplanung für das Simulationsmodell A ist in Tabelle 7.3 zusammengefasst. Das Design für Simulationsmodell B findet sich in Tabelle 7.4.

Tabelle 7.3: Versuchsplanung für Simulationsmodell A

Faktor	Level	Anzahl
Parameter des Steuerungssystems		
Aufrufintervall τ_Δ	2 h; 4 h; 6 h	3
Teilproblemlöser	ATC-SSP; FIFO-SSP	2
Anzahl der VNS-Züge	0; 100	2
SBH-Variante	SBH-HT, SBH-ST	2
Faktoren zur Beschreibung des Produktionsbasissystems		
Flussfaktor FF	1,6; 2,0	2
Prioritätsverteilung	$D_1; D_2$	2
Systemlast	niedrig, hoch	2
Summe		192

7.4 Diskussion der Ergebnisse

Zur Durchführung der Experimente wurden Lenovo Thinkpad Laptops mit Intel Core Duo Prozessoren mit 2 Ghz und 2GB RAM verwendet. Die Laufzeiten der einzelnen Simulationsexperimente variieren sehr stark in Abhängigkeit von den einzelnen Faktoren. Für das kleinere Modell A mit niedriger Last, $\tau_\Delta = 6$ h, SBH-HT mit FIFO-SSP für alle Teilprobleme benötigt die Simulation fünf Stunden. Im Falle von SBH-ST mit ATC-SSP und VNS für das Transportsystem benötigt ein Simulationslauf von 40 bis zu 200 Stunden abhängig vom Planungshorizont. Die Ursache für die lange Laufzeit liegt hier

Tabelle 7.4: Versuchsplanung für Simulationsmodell B

Faktor	Level	Anzahl
Parameter des Steuerungssystems		
Aufrufintervall τ_Δ	2 h; 4 h; 6 h	3
Teilproblemlöser	ATC-SSP; FIFO-SSP	2
Anzahl der VNS-Züge	0; 100	2
SBH-Variante	SBH-HT	1
Faktoren zur Beschreibung des Produktionsbasissystems		
Flussfaktor FF	1,6; 2,0	2
Prioritätsverteilung	D_2	1
Systemlast	niedrig, hoch	2
Summe		48

hauptsächlich darin, dass bei der simultanen Variante das sehr große Transportteilproblem immer wieder gelöst werden muss. Tabelle 7.5 zeigt die Laufzeiten für einen Aufruf der unterschiedlichen SBH-Varianten der für das Simulationsmodell A mit niedriger Last, weiten geplanten Aussteuerterminen und Gewichtsverteilung D_2.

Tabelle 7.5: durchschnittliche Laufzeiten für einen SBH-Aufruf (Simulationsmodel A mit niedriger Last, weiten geplanten Aussteuerterminen und Gewichtsverteilung D_2)

Algorithmus	VNS-Züge	τ_Δ	ATC-SSP		FIFO-SSP	
		(h)	SBH-HT (min)	SBH-ST (min)	SBH-HT (min)	SBH-ST (min)
SBH	0	2	1,61	12,39	0,14	0,55
SBH	0	4	2,62	13,24	0,23	0,93
SBH	0	6	5,30	50,95	0,40	19,71
SBH	100	2	3,11	15,12	1,64	7,75
SBH	100	4	5,11	26,94	2,96	12,13
SBH	100	6	8,99	77,82	4,04	19,20

Tabelle 7.6 zeigt die Ergebnisse für Modell A mit hoher Last und engen geplanten Aussteuerterminen und vielen hochpriorisierten Losen. Es werden die TWT-Werte für die unterschiedlichen SBH-Varianten, für FIFO-NLF sowie für ATC-NLF gezeigt.

Bei der Verwendungen der FIFO-Teilproblemlöser liefert die SBH in dieser Situation nur leicht bessere Ergebnisse als FIFO-NLF und wird eindeutig von ATC-NLF geschlagen. Der Grund ist, dass die SBH bei hoher Last und engen geplanten Aussteuerterminen sehr stark von der Leistungsfähigkeit der Teilproblemlöser abhängt. Bei der Verwendung von ATC-SSP werden wesentlich bessere Ergebnisse erzielt. Ein größerer Planungshorizont führt hierbei zu besseren Ergebnissen. Der Grund hierfür ist, dass der Algorithmus

Tabelle 7.6: TWT für Simulationsmodel A mit hoher Last, engen geplanten Aussteuerterminen und Gewichtsverteilung D_1

Algorithmus	VNS-Züge	τ_Δ	ATC-SSP		FIFO-SSP	
			SBH-HT	SBH-ST	SBH-HT	SBH-ST
		(h)	(h)	(h)	(h)	(h)
FIFO-NLF						166 337
ATC-NLF						56 959
SBH	0	2	53 008	52 518	164 679	167 615
SBH	0	4	44 617	48 263	164 923	163 418
SBH	0	6	40 212	41 137	165 675	165 750
SBH	100	2	52 575	52 855	165 399	165 427
SBH	100	4	45 867	50 150	164 519	164 883
SBH	100	6	37 235	42 356	164 119	164 580

bei einem größeren Planungshorizont mehr globale Informationen verwenden kann. Bei kürzeren Planungshorizonten ist die Menge der berücksichtigten Informationen und der mögliche Raum zur Verbesserung kleiner. In diesem Fall verhält sich die SBH eher wie eine Prioritätsregel. Die Verwendung von VNS für das Transportteilproblem führt zu leicht besseren Ergebnissen. Es ist allerdings etwas überraschend, dass die simultane Variante der SBH kaum bessere Ergebnisse liefert als die hierarchische Variante. Das nur teilweise Lösen des Transportteilproblems führt zu ungünstigen Vorgaben der Maschinengruppenteilproblemlöser, da speziell in der frühen Phase der SBH nur wenig Informationen für das Transportsystem vorhanden sind und hierdurch schlechtere Entscheidungen getroffen werden.

Aus früheren Arbeiten, zum Beispiel Mönch und Drießel (2005), ist bekannt, dass die SBH nicht immer gute Ergebnisse liefert. Speziell in Niedriglastsituationen schwankt die Qualität der erzeugten Ablaufpläne stark. Tabelle 7.7 zeigt die Ergebnisse für dasselbe Modell bei niedriger Last, weiten geplanten Aussteuerterminen sowie wenigen hochpriorisierten Losen. Es werden die TWT-Werte für die unterschiedlichen SBH-Varianten und die Prioritätsregeln gezeigt. Im Vergleich zu der Situation in Tabelle 7.6 liegen im Falle von Tabelle 7.7 deutlich niedrigere TWT-Werte vor.

Bei niedriger Last und weiten geplanten Aussteuerterminen ist es besser, für die Maschinengruppenteilprobleme FIFO-SSP zu verwenden und für das Transportteilproblem VNS. Die Verwendung von ATC-SSP führt nur zu leichten Verbesserungen im Vergleich zu FIFO-NLF. Ein Grund liegt darin, dass ATC-Heuristiken bei niedriger Last und mit späten Fertigstellungsterminen d_{ij}^k relativ schlechte Ergebnisse liefern, da der Schlupf $d_{ij}^k - p_{ij} - \max(r_{ij}, t)$ in diesen Fällen oft sehr groß ist und dadurch der Schlupfterm

$$\exp\left(-\frac{\max(d_{ij}^k - p_{ij} - \max(r_{ij}, t), 0)}{k_1 \bar{p}}\right)$$

Tabelle 7.7: TWT für Simulationsmodel A mit niedriger Last, weiten geplanten Aussteuerterminen und Gewichtsverteilung D_2

Algorithmus	VNS-Züge	τ_Δ	ATC-SSP		FIFO-SSP	
			SBH-HT	SBH-ST	SBH-HT	SBH-ST
		(h)	(h)	(h)	(h)	(h)
FIFO-NLF						1 834
ATC-NLF						2 130
SBH	0	2	597	2 463	1 075	1 842
SBH	0	4	1 727	1 625	868	876
SBH	0	6	1 556	2 408	1 234	717
SBH	100	2	1 599	2 539	593	2 022
SBH	100	4	1 869	1 843	494	1 664
SBH	100	6	1 274	3 626	402	1 305

entsprechend klein ist. In diesem Fall ist es schwierig, die Unterschiede zwischen den einzelnen Bearbeitungsschritten zu ermitteln. Die ersten Reihenfolge- und Zuteilungsentscheidungen sind aus diesem Grund häufig nicht effektiv. Wenn einige Bearbeitungsschritte bereits eingeplant sind und t größer wird, lassen sich die restlichen Bearbeitungsschritte besser unterscheiden. Die bereits getroffenen früheren Entscheidungen lassen sich allerdings nicht mehr korrigieren. Ein weiterer Grund liegt darin, wie die Bewertung der einzelnen Maschinengruppen vorgenommen wird. In Niedriglastsituationen ist die TWT für einzelne Teilprobleme null. Da die Bewertung, wie kritisch eine Maschinengruppe ist, hauptsächlich auf Basis der TWT erfolgt (vgl. Unterabschnitt 4.3.4), kann es hier zu ungünstigen Entscheidungen kommen. Die im Unterabschnitt 4.3.4 sowie Unterabschnitt 5.2.2 vorgeschlagenen Strategien, scheinen nicht zu genügen. Allerdings zeigten vorhergehende Experimente, dass eine alleinige Verwendung der totalen gewichteten Verspätung als Leistungsmaß in dieser Situation zu noch schlechteren Ergebnissen führt. Es ist in diesem Zusammenhang zu bemerkten, dass auch ATC-NLF in dieser Situation schlechtere Ergebnisse als FIFO-NLF liefert.

Interessant in diesem Zusammenhang ist, dass die Verwendung von FIFO als Teilproblemlöser zusammen mit VNS für das Transportsystem sehr gute Ergebnisse liefert. In Niedriglastsituationen scheint das Transportsystem im Vergleich zum Bearbeitungssystem deutlich wichtiger zu sein als in Hochlastsituationen. Wie bereits in Almeder und Mönch (2009) und Mönch und Drießel (2010) beschrieben (vgl. auch Unterabschnitt 5.5.2), eignet sich VNS besonders für Situationen mit weiten geplanten Fertigstellungsterminen, bei denen es entsprechenden Verbesserungsspielraum gibt, wohingegen ATC-basierte Heuristiken in diesen Situationen eher schlechte Ergebnisse liefern. Mit Hilfe des FIFO-Teilproblemlösers wird in diesem Fall zunächst ein stabiler Ablaufplan für die Maschinengruppen erzeugt wird. Unter Stabilität versteht man, dass der Durchsatz im Zeitverlauf nicht stark schwankt. Durch den VNS-Teilproblemlöser werden im Anschluss

die Leerfahrten der Fahrzeuge unter Berücksichtigung der TWT minimiert. Zusammen resultiert dies in einen guten Durchsatz und einer niedrigen TWT.

Die Tabellen 7.8 und 7.9 zeigen die Ergebnisse für das Simulationsmodell B. Tabelle 7.8 zeigt die Ergebnisse für die hohe Lastsituation mit engen geplanten Aussteuerterminen. Tabelle 7.9 zeigt die Ergebnisse für die niedrige Lastsituation mit weiten geplanten Aussteuerterminen. In beiden Fällen wird die Gewichtsverteilung D_2 betrachtet.

Tabelle 7.8: TWT und ACT für Simulationsmodel B mit hoher Last, engen geplanten Aussteuerterminen und Gewichtsverteilung D_2

Algorithmus	VNS-Züge	τ_Δ	ATC-SSP		FIFO-SSP	
			TWT	ACT	TWT	ACT
		(h)	(h)	(h)	(h)	(h)
FIFO-NLF					38 062	134,74
ATC-NLF					25 004	122,44
SBH	0	2	23 175	131,56	38 034	133,01
SBH	0	4	44 089	160,33	40 716	138,81
SBH	0	6	67 505	197,40	46 946	149,34
SBH	100	2	23 303	130,16	39 951	134,78
SBH	100	4	58 731	181,07	42 783	141,45
SBH	100	6	73 953	205,05	45 616	148,12

Tabelle 7.9: TWT und ACT für Simulationsmodel B mit niedriger Last, weiten geplanten Aussteuerterminen und Gewichtsverteilung D_2

Algorithmus	VNS-Züge	τ_Δ	ATC-SSP		FIFO-SSP	
			TWT	ACT	TWT	ACT
		(h)	(h)	(h)	(h)	(h)
FIFO-NLF					27 851	126,15
ATC-NLF					32 001	118,22
SBH	0	2	27 457	127,27	27 269	127,89
SBH	0	4	22 133	130,85	31 551	129,62
SBH	0	6	18 929	134,53	27 249	135,92
SBH	100	2	25 901	128,14	31 151	125,21
SBH	100	4	27 198	130,96	28 930	131,48
SBH	100	6	18 167	135,43	26 742	137,74

Für das Simulationsmodell B mit hoher Last werden die besten Ergebnisse bei einem Aufrufintervall von zwei Stunden erzielt (siehe Tabelle 7.8). Größere Aufrufintervalle führen hier zu deutlichen Verschlechterungen. Bei der Betrachtung der Ergebnisse für

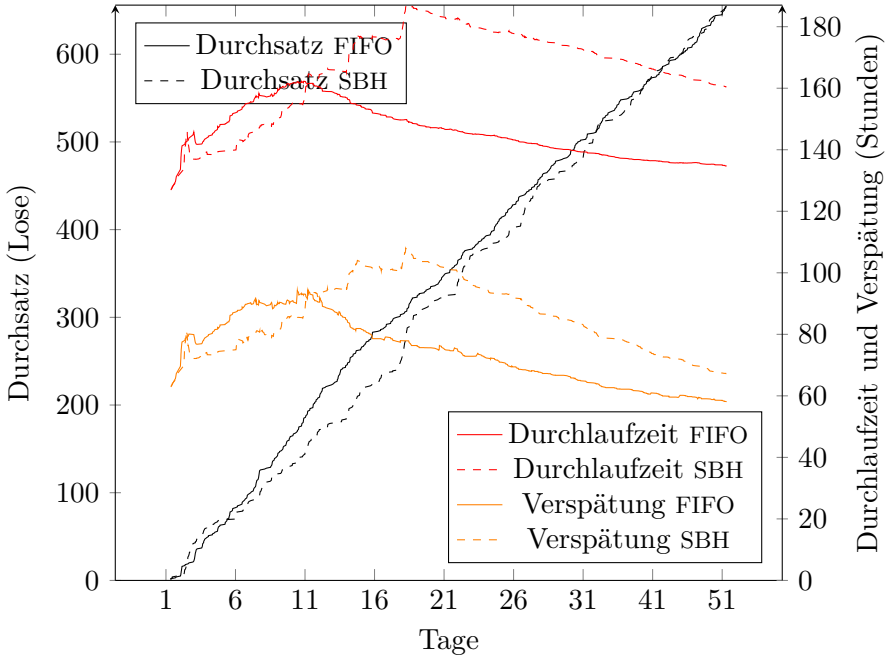

Abbildung 7.2: TP und AWT für Modell B mit hoher Last, engen geplanten Aussteuerterminen und Gewichtsverteilung D_2

Niedriglast (siehe Tabelle 7.9) tritt dies jedoch nicht auf. Hier verbessern sich mit größeren Aufrufintervallen die Ergebnisse. Die Ergebnisse für die Niedriglastsituation des Modells B (vgl. Tabelle 7.9) entsprechen damit den Ergebnissen für die Hochlastsituation des Modells A (vgl. Tabelle 7.6). Grund hierfür ist das bereits in Unterabschnitt 3.2.3 erwähnte Verhalten. Eine Verbesserung der TWT geht fast immer mit einer steigenden durchschnittlichen Durchlaufzeit der einzelnen Lose einher. Eine Erhöhung der Durchlaufzeit verringert auch den Durchsatz durch das Produktionssystem. In einer starken Hochlastsituation führt dies zu einer Verschlechterung der TWT. In Abbildung 7.2 ist die Entwicklung des Durchsatzes und die Entwicklung der gewichteten Verspätung sowie der Durchlaufzeit für SBH-HT mit vier Stunden Abrufintervall und ATC-SSP ohne VNS im Vergleich zu FIFO-NLF während eines Simulationslaufs dargestellt. Der Durchsatz ist dabei als Summe der fertiggestellten Lose im Zeitablauf dargestellt. Die Durchlaufzeit und die Verspätung beziehen sich immer auf das aktuell fertiggestellte Los.

Es ist erkennbar, dass der Durchsatz der SBH zunächst höher als der Durchsatz von FIFO ist, danach jedoch deutlich hinter FIFO zurückfällt. In diesem Zuge verschlechtert sich auch die SBH-Durchlaufzeit gegenüber der FIFO-Durchlaufzeit. Daraus resultiert dann ebenfalls eine Verschlechterung der Verspätungen der SBH gegenüber FIFO. Der Grund für dieses Verhalten liegt darin, dass bei jedem Aufruf der SBH das Hauptoptimierungskriterium die totale gewichtete Verspätung ist. Hierbei ist es notwendig, dass einige Lose verlangsamt und andere beschleunigt werden. In einigen Fällen erhöht sich dadurch die durchschnittliche Durchlaufzeit, da durch das Verlangsamen und Beschleunigen der Lose

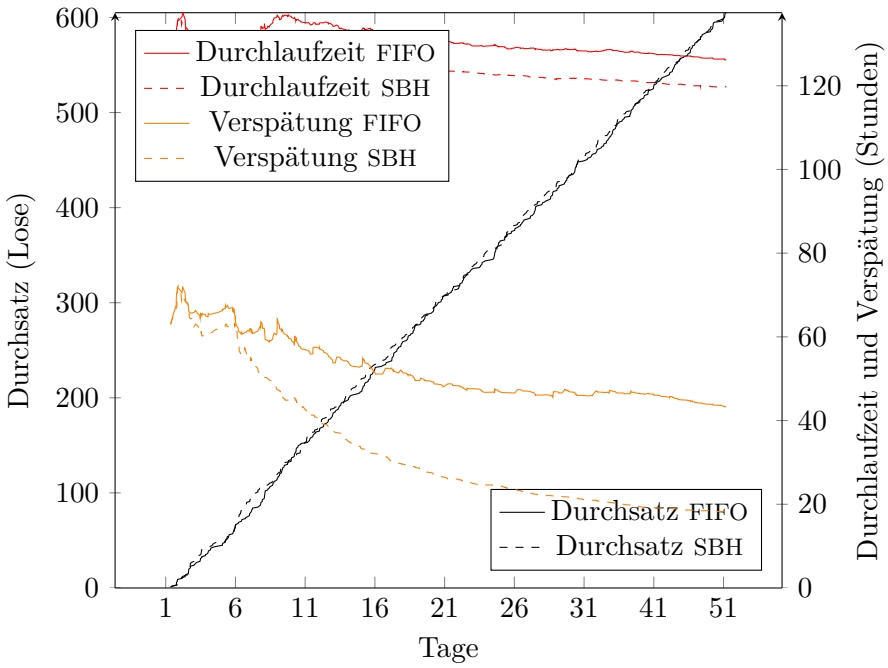

Abbildung 7.3: TP und AWT für Modell B mit niedriger Last, engen geplanten Aussteuerterminen und Gewichtsverteilung D_2

Lücken innerhalb des Ablaufplans und damit Wartezeiten entstehen. Damit reduziert sich Auslastung der Maschinen und der Durchsatz des Produktionssystems. Bei gleichbleibender Einsteuerung der Lose erhöht sich dabei die durchschnittliche Anzahl der Lose innerhalb des Produktionssystems. Dieses Verhalten ist in Abbildung vom 5.01. bis zum 20.01. zu sehen. Dies sorgt wiederum dafür, dass die gewichtete Verspätung langsam ansteigt, da in Zukunft mehr Lose betrachtet werden müssen. Dieses Verhalten scheint jedoch nur in starken Hochlastsituationen aufzutreten. Wie in Abbildung 7.3 zu sehen ist, tritt in niedriger ausgelasteten Systemen keine Erhöhung der Durchlaufzeit ein, da hier genügend Möglichkeiten bestehen, Lose zu verlangsamen oder zu beschleunigen, ohne das diese sich gegenseitig behindern.

In Mönch (2005) wurde bei einer ähnlichen Situation die Einsteuerungsstrategie der Lose entsprechend angepasst. Eine anderer Ansatz könnte eine gleichzeitige Berücksichtigung des Durchsatzes bzw. der Durchlaufzeit im Rahmen eines multikriteriellen Ansatzes durch die Teilproblemlöser sein (vgl. auch Pfund u. a. (2008a)).

Um die Ergebnisse für Simulationsmodell A und Simulationsmodell B vergleichen zu können, werden die Ergebnisse der unterschiedlichen Varianten der SBH mit den FIFO-NLF-Werten in Tabelle 7.10 normiert. Verbesserungen bis zu über 70 % gegenüber FIFO-NLF sind möglich. Meistens liegen die Verbesserungen jedoch zwischen 40 % und 50 %. In einigen Fällen gibt es Verschlechterungen, deren Gründe bereits diskutiert wurden.

Weiterhin ist erkennbar, dass bei beiden Modellen in vielen Fällen ähnliche relative Verbesserungen erzielt werden. Dies deutet darauf hin, dass die Ergebnisse auch auf

andere Modelle übertragbar sind. Für genauere Aussagen sind jedoch weitere Experimente mit größeren Modellen erforderlich. Diese sind allerdings mit der aktuellen Implementierung aufgrund der langen Zeiten für einen Simulationslauf derzeit nicht möglich. Einen Ausweg bietet hier die Ausnutzung der Möglichkeiten zur Parallelisierung der SBH. Ohne Berücksichtigung des Transportteilproblems müssen innerhalb der SBH bei n Maschinengruppen

$$\sum_{k=1}^{n}(n-k+1) = \frac{n(n+1)}{2} \tag{7.6}$$

Teilprobleme gelöst werden. Allerdings sind die $n-k+1$ Teilprobleme innerhalb der k-ten Iteration voneinander unabhängig und können gleichzeitig berechnet werden. Erste Untersuchungen in diese Richtung sind vielversprechend (vgl. Drießel u. a. (2010)).

Zusammenfassend lässt sich sagen, dass die SBH für komplexe Produktionssysteme mit automatischem Transport im dynamischen Umfeld gute Ergebnisse im Vergleich zu FIFO-NLF und ATC-NLF liefert. Hierbei scheint, im Bezug auf Laufzeit und Lösungsqualität, der hierarchische Ansatz dem simultanen Ansatz überlegen.

Tabelle 7.10: TWT-Werte für ATC-SSP relativ zu FIFO-NLF

Modell	Last	τ_Δ	$FF = 1{,}6$		$FF = 2{,}0$	
		(h)	0 VNS-Züge	100 VNS-Züge	0 VNS-Züge	100 VNS-Züge
A	hoch	2	0,667	0,691	0,683	0,681
		4	0,657	0,616	0,683	0,673
		6	0,448	0,453	0,498	0,286
	niedrig	2	0,544	0,501	0,326	0,872
		4	0,490	0,494	0,942	1,019
		6	0,777	0,664	0,848	0,695
B	hoch	2	0,666	0,632	0,754	0,736
		4	1,393	1,911	1,053	1,026
		6	2,196	2,406	1,584	1,474
	niedrig	2	0,421	0,398	0,614	0,623
		4	0,430	0,478	0,631	0,628
		6	0,640	0,596	0,675	0,704

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Verfahren zur Ablaufplanung von komplexen Produktionssystemen mit automatischem Transport auf Basis der SBH vorgestellt.

Es erfolgte zunächst eine Analyse des Zusammenspiels zwischen dem Transport- und Bearbeitungssystems innerhalb komplexer Produktionssysteme mit automatischem Transport und eine Beschreibung der Anwendungsdomäne Halbleiterfertigung. Mögliche Verfahrensansätze mit Vor- und Nachteilen wurden vorgestellt. Dekompositionsverfahren wie die SBH erscheinen besonders geeignet, da das Gesamtproblem hier in kleinere und leichter lösbare Teilprobleme zerlegt wird.

Darauf aufbauend wurde ein hierarchischer und ein simultaner Ansatz zur Integration des Transportplanungsproblems in die SBH vorgeschlagen. Hierzu wurden zunächst die notwendigen Erweiterungen des disjunktiven Graphenmodells herausgearbeitet. Danach erfolgte eine Formulierung der zu betrachtenden Teilprobleme. Für das sich neu ergebende Transportteilproblem wurden verschiedene Lösungsansätze auf Basis von VNS vorgeschlagen. Es wurde gezeigt, dass die Ansätze in der Lage sind, schnell sehr gute Ergebnisse liefern und damit als Teilproblemlöser innerhalb der SBH geeignet erscheinen.

Eine simulationsbasierte Leistungsbewertung der vorgeschlagenen Heuristik wurde durchgeführt. Hierzu erfolgte eine Implementierung der Verfahren und die Erstellung eines Rahmenwerks für die Leistungsbewertung. Weiterhin wurden Simulationsmodelle von Halbleiterfabriken beschrieben, mit deren Hilfe die Leistungsbewertung durchgeführt wurde.

Die durchgeführten Arbeiten bieten Ansatzpunkte für weitere Forschungsarbeiten. Im Rahmen des Transportteilproblems ist eine Untersuchung weiterer Nachbarschaften interessant. Anstatt nur einzelne Bearbeitungsschritte zu tauschen bzw. zu transferieren wird eine bessere Ergebnisqualität erwartet, wenn ganze Blöcke von Bearbeitungsschritten bewegt werden, da bereits gefundene gute Sequenzen nicht zerstört werden. Hierzu ist zu klären, wie diese Blöcke zu bilden und gegebenfalls zu identifizieren sind.

Weiterhin wird erwartet, dass die Berücksichtigung von Dominanzregeln (vgl. Lou u. a. (2006)) innerhalb der lokalen Suchen die Zeiten für die Suche stark verringern. Ebenso wird erwartet, dass Nachbarschaften basierend auf Vertauschungen von aufeinanderfolgenden Bearbeitungsschritten ebenfalls gut funktionieren.

Ansätze auf Basis der Zeitfensterdekomposition, wie in Ovacik und Uzsoy (1995) für ein ähnlichen Optimierungsproblem vorgeschlagen, erscheinen vielversprechend und können einfach in den bisherigen Ansatz eingearbeitet werden. Diese Technik wurde bereits erfolgreich für ein Batch-Scheduling-Problem mit Verfügbarkeitszeiten angewandt (siehe Mönch u. a. (2005) und Klemmt u. a. (2009)).

Im Rahmen des Gesamtproblems müssen weitere Simulationsstudien mit komplexeren Modellen durchgeführt werden. Hierzu ist der bisherige Ansatz allerdings zu langsam.

Einen Ausweg kann hierbei die Parallelisierung der Berechnung der Unterprobleme bieten, wie in Drießel u. a. (2010) prototypisch gezeigt wurde. Bisher wurden ebenfalls aus Laufzeitgründen nur für das Transportteilproblem fortgeschrittenere Verfahren eingesetzt. Es ist jedoch prinzipiell möglich, wie in Mönch u. a. (2007) gezeigt, fortgeschrittenere Verfahren auch für die Maschinengruppenteilprobleme einzusetzten.

Weiterhin ist intensiver zu untersuchen, wie die Ergebnisse in einer starken Hochlastsituation stabilisiert werden können. Dies kann zum einen durch die Berücksichtigung weiterer Zielgrößen oder durch eine Anpassung der Einsteuerstrategie wie in Mönch (2005) erfolgen.

Das bisher betrachtete Modell besitzt zur Zeit noch einige Einschränkungen. So wird von unendlicher Kapazität der Stocker ausgegangen. In Modellen mit endlicher Stockerkapazität kann es passieren, dass Maschinen blockiert sind, wenn nicht genügend Platz in einen oder mehreren Stockern verfügbar ist. In Brucker und Knust (2006) wurde diese zusätzliche Bedingung mittels zusätzlicher Knoten innerhalb des disjunktiven Graphen behandelt. Dies hat jedoch längere Laufzeiten zur Folge.

In dieser Arbeit wurde nur der Fall eines einzelnen Transportsystems für das gesamte Produktionssystem betrachtet. In der Praxis existieren jedoch manchmal mehrere Transportsysteme, die nur durch Stocker miteinander verbunden sind. In diesem Fall werden ebenfalls weitere Knoten innerhalb des disjunktiven Graphen notwendig.

Des Weiteren erfolgen die Routing-Entscheidungen im Rahmen der Steuerung der Fahrzeuge. Innerhalb der SBH wird aus diesem Grund zur Zeit angenommen, dass die Fahrzeuge den kürzesten Weg fahren. In der Realität sind jedoch Staus innerhalb der Transportsysteme typisch. Eine Behandlung der Staus innerhalb der SBH könnte an dieser Stelle zu Verbesserungen führen. Die Behandlung von Stauung kann vermutlich analog der Behandlung der endlichen Stockerkapazität erfolgen. Dies bedingt jedoch neben dem Einfügen einer Vielzahl weiterer Knoten einen Austausch des Transportsimulators.

Weiterhin ist die Durchführung von Simulationsstudien mit größeren Simulationsmodellen notwendig. Dies ist jedoch aufgrund der langen Laufzeiten nur eingeschränkt möglich. In diesem Zusammenhang sollten auch die Reaktion auf unterschiedliche Kapazitäten des Transportbasissystems bezüglich der Anzahl der Fahrzeuge und Strecken untersucht werden.

Literaturverzeichnis

Aarts und Lenstra 2003 Aarts, E.; Lenstra, J.K.: *Local search in combinatorial optimization.* Princeton University Press, 2003

Ackoff 1971 Ackoff, R.: Towards a system of systems concepts. In: *Management Science* 17 (1971), S. 661–671

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

Agrawal und Heragu 2006 Agrawal, G.; Heragu, S.: A survey of automated material handling systems in 300-mm semiconductor fabs. In: *IEEE transactions on semiconductor manufacturing* 19 (2006), S. 112–120

Aho und Mäkinen 2006 Aho, I.; Mäkinen, E.: On a parallel machine scheduling problem with precedence constraints. In: *Journal of Scheduling* 9 (2006), S. 493–495

Allahverdi u. a. 2008 Allahverdi, A.; Ng, C.; Cheng, T.; Kovalyov, M.: A survey of scheduling problems with setup times or costs. In: *European Journal of Operational Research* 187 (2008), S. 985–1032

Almeder und Mönch 2009 Almeder, C.; Mönch, L.: Variable neighborhood search for parallel batch machine scheduling. In: *Proceedings of the 8th Metaheuristic International Conference (MIC 2009)*, 2009

Anghinolfi und Paolucci 2007 Anghinolfi, D.; Paolucci, M.: Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. In: *Computers & Operations Research* 34 (2007), S. 3471–3490

Anghinolfi und Paolucci 2008 Anghinolfi, D.; Paolucci, M.: A new ant colony optimization approach for the single machine total weighted tardiness scheduling problem. In: *International Journal of Operations Research* 5 (2008), Nr. 1, S. 1–17

Anghinolfi und Paolucci 2009 Anghinolfi, D.; Paolucci, M.: A new discrete particle swarm optimization approach for the total weighted tardiness scheduling problem. In: *European Journal of Operational Research* 193 (2009), Nr. 1, S. 73–85

Anwar und Nagi 1998 Anwar, M.; Nagi, R.: Integrated scheduling of material handling and manufacturing activities for just-in-time production of complex assemblies. In: *International Journal of Production Research* 36 (1998), Nr. 3, S. 653–681

Applegate und Cook 1991 Applegate, D.; Cook, W.: A computational study of job shop scheduling. In: *ORSA Journal of Computing* 3 (1991), S. 149–156

Atherton und Atherton 1995 Atherton, L.; Atherton, R.: *Wafer Fabrication: Factory Performance and Analysis.* Boston, Dordrecht, London : Kluwer Academic Publishers, 1995

Aytug u. a. 2002 Aytug, H.; Kempf, K.; Uzsoy, R.: Measures of subproblem criticality in decomposition algorithms for shop scheduling. In: *International Journal of Production Research* 41 (2002), S. 865–882

Balas 1969 Balas, E.: Machine sequencing via disjunctive graphs: an implicit enumeration approach. In: *Operations Research* 17 (1969), S. 941–957

Balas u. a. 1998 Balas, E.; Lancia, G.; Serafini, P.; Vazacopoulos, A.: Job shop scheduling with deadlines. In: *Journal of Combinatorial Optimization* 1 (1998), S. 329–353

Balas u. a. 1995 Balas, E.; Lenstra, J.; Vazacopoulos, A.: The one machine problem with delayed precedence constraints and its use in job shop scheduling. In: *Management Science* 41 (1995), S. 94–109

Balas und Vazacopoulos 1998 Balas, E.; Vazacopoulos, A.: Guided local search with shifting bottleneck for job shop scheduling. In: *Management Science* 44 (1998), Nr. 2, S. 262–275

Balasubramanian u. a. 2004 Balasubramanian, H.; Mönch, L.; Fowler, J.; Pfund, M.: Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. In: *International Journal of Production Research* 42 (2004), Nr. 8, S. 1621–1638

Bellman 1957 Bellman, R.: *Dynamic Programming.* Princeton University Press, 1957

Bergamaschi u. a. 1997 Bergamaschi, D.; Cigolini, R.; Perona, M.; Portioli, A.: Order review and release Strategies in a job shop environment: a review and a classification. In: *International Journal of Production Research* 35(2) (1997), S. 339–420

Bierwirth 2000 Bierwirth, C.: *Adaptive search and the management of logistic systems - base models for learning agents.* Dordrecht : Kluwer Academic Publishers, 2000

Bierwirth u. a. 1996 Bierwirth, C.; D.Mattfeld; Kopfer, H.: On permutation representations for scheduling problems. In: *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature.* London, UK : Springer-Verlag, 1996 (PPSN IV), S. 310–318

Blackstone u. a. 1982 Blackstone, J.; Phillips, D.; Hogg, G.: A state-of-the-art survey of dispatching rules for manufacturing job shop operations. In: *International Journal of Production Research* 20(1) (1982), S. 27–45

Blum und Roli 2003 Blum, C.; Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. In: *ACM Computing Surveys* 35 (2003), Nr. 3, S. 268–308

Brooks Automation 2001 Brooks Automation, Inc.: *AutoMod User's Manual.* 1. Utah : Brooks Automation, Inc., 2001

Brucker 2004 Brucker, P.: *Scheduling Algorithms.* Springer, 2004

Brucker u. a. 2003 Brucker, P.; Heitmann, S.; Hurink, J.: Flow-shop problems with intermediate buffers. In: *OR Spectrum* 25 (2003), S. 549–574

Brucker und Knust 2006 Brucker, P.; Knust, S.: *Complex Scheduling.* 2. Springer, 2006

Brucker und Knust 2011 Brucker, P.; Knust, S.: *Complexity results for scheduling problems.* http://www.informatik.uni-osnabrueck.de/knust/class/. 01 2011. – Zugriff 01.02.2011

Cardarelli und Pelagagge 1995 Cardarelli, G.; Pelagagge, P.: Simulation tool for design and management optimization of automated interbay material handling and storage systems for large wafer fab. In: *IEEE Transactions on Semiconductor Manufacturing* Bd. 8, 1995, S. 44–49

Carlier 1982 Carlier, J.: The one-machine sequencing problem. In: *European Journal of Operational Research* 11 (1982), S. 42–47

Caumond u. a. 2006 Caumond, A.; Lacomme, P.; Moukrim, A.; Tchernev, N.: A MILP for scheduling problems in an FMS with one vehicle. In: *European Journal of Operational Research* (2006)

Chung und Jeng 2003 Chung, S.; Jeng, M.: An overview of semiconductor fab automation systems. In: *Proccediogs of the2003 IEEE International Conference on Robotics & Automation* Bd. 1, 2003, S. 1050–1055

Colorni u. a. 1991 Colorni, A.; Dorigo, M.; Maniezzo, V.: Distributed optimization by ant colonies. In: *Proceedimgs of ECAL'91, European Conference on Artifical Life, Berlin*, 1991, S. 134–142

Corsten 2007 Corsten, H.: *Produktionswirtschaft: Einführung in das industrielle Produktionsmanagement.* 11. Oldenburg Wissenschaftsverlag GmbH, 2007

Crama u. a. 2000 Crama, Y.; Kats, V.; Van Knludert, J.; Levner, E.: Cyclic scheduling in robotic flowshop. In: *Annals of Operations Research* 96 (2000), Nr. 1, S. 97–124

Dantzig 1963 Dantzig, G. B.: *Linear Programming and Extensions.* Princeton University Press, 1963

Dauzere-Peres und Lasserre 1993 Dauzere-Peres, S.; Lasserre, J. B.: A modified shifting bottleneck procedure for job-shop scheduling. In: *International Journal of Production Research* 31/4 (1993), S. 923–932

Demirkol u. a. 1997 Demirkol, E.; Metha, S.; Uzsoy, R.: A computational study of shifting bottleneck procedures for shop scheduling problems. In: *Journal of Heuristics* 3 (1997), S. 111–137

den Besten und Stützle 2001 den Besten, M.; Stützle, T.: Neighborhoods revisited: An experimental investigation into the effectiveness of variable neighborhood descent for scheduling. In: *Proceedings of the 4th Metaheuristic International Conference (MIC 2001)*, 2001, S. 545–549

Deroussi u. a. 2008 Deroussi, L.; Gourgand, M.; Tchernev, N.: A simple metaheuristic approach to simultaneous scheduling of machines and automated guided vehicles. In: *International Journal of Production Research* 46 (2008), Nr. 8, S. 2143–2164

Dessouky 1998 Dessouky, M.: Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness. In: *Computers and Industrial Engineering* 34 (1998), Nr. 4, S. 793–806

Domschke u. a. 1997 Domschke, W.; Scholl, A.; Voß, S.: *Produktionsplanung - Ablauforganisatorische Aspekte*. Berlin : Springer, 1997 (2. Auflage)

Dorigo 1992 Dorigo, M.: *Optimization, learning and natural algorithms*, Politecnico di Milano, Italien, Dissertation, 1992

Dorigo u. a. 1991 Dorigo, M.; Maniezzo, V.; Colorni, A.: Ant System: An autocatalytic optimizing process - Technical Report 91-016. / Politecnico di Milano. 1991. – Forschungsbericht. Italien

Dorigo und Stütze 2004 Dorigo, M.; Stütze, T.: *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004

Dorndorf und Pesch 1995 Dorndorf, U.; Pesch, E.: Evolution based learning in a job shop scheduling environment. In: *Computers and Operations Research* 22 (1995), S. 25–40

Drexl u. a. 1994 Drexl, A.; Fleischmann, B.; Günther, H.; Stadtler, H.; Tempelmeier, H.: Konzeptionelle Grundlagen kapazitätsorientierter PPS-Systeme. In: *Zeitschrift für betriebswirtschaftliche Forschung* 46 (1994), S. 1022–1045

Drießel u. a. 2010 Drießel, R.; Hönig, U.; Mönch, L.; Schiffmann, W.: A parallel shifting bottleneck heuristic for scheduling complex job shops: architecture and performance Assessment. In: *CASE*, 2010, S. 81–86

Drießel und Mönch 2007 Drießel, R.; Mönch, L.: Performance evaluation of lot dispatching and automated material handling approaches by discrete event simulation.

In: Ottjes, Jaap (Hrsg.); Veeke, Hans (Hrsg.): *Proceedings 5th Industrial Simulation Conference*, 04 2007, S. 230–235

Engelien und Stahn 1989 Engelien, M.; Stahn, H.: *Software-Engineering: CAMRAS-Technologie.* Akademie-Verlag, Berlin, 1989

Foster und Pillai 2008 Foster, Leonard; Pillai, Devadas: Kap. 300mm wafer fab logistics and automated material handling systems. In: Doering, R. (Hrsg.); Nishi, Y. (Hrsg.): *Handbook of Semiconductor Manufacturing Technology*, CRC Press, 2008

Fowler u. a. 2002 Fowler, F.; Hogg, G.; Mason, S.: Workload control in the semiconductor industry. In: *Production Planing & Control* 13(7) (2002), S. 568–578

Fowler und Robinson 1995 Fowler, J.; Robinson, J.: Measurement and improvement of manufacturing capacities (MIMAC): Final report. / SEMATECH. 1995 (95062861A-TR). – Forschungsbericht. Austin, Texas, USA

Gharbi und Haouari 2002 Gharbi, A.; Haouari, M.: Minimizing makespan on parallel machines subject to release dates and delivery times. In: *Journal of Scheduling* 5 (2002), S. 329–355

Glover 1989 Glover, F.: Tabu Search, Part I. In: *ORSA Journal of Computing 1* (1989), S. 190–206

Glover 1990 Glover, F.: Tabu Search, Part II. In: *ORSA Journal of Computing 2* (1990), S. 4–32

Glover und Laguna 1997 Glover, F.; Laguna, M.: *Tabu Search.* Kluwer, Dordrecht, 1997

Goldratt und Cox 1992 Goldratt, D.; Cox, J.: *The Goal.* North River Press, 1992

Graham u. a. 1979 Graham, R.; Lawler, E.; Lenstra, J.; Kann, A. R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: *In: Annals of Discrete Mathematics* (1979), S. 343–362

Gupta und Smith 2006 Gupta, S.; Smith, J.: Algorithms for single machine total tardiness scheduling with sequence dependent setups. In: *European Journal of Operational Research* 175 (2006), Nr. 2, S. 722–739

Hansen und Mladenović 2001 Hansen, P.; Mladenović, N.: Variable neighborhood search: principles and applications. In: *European Journal of Operational Research* 130 (2001), S. 449–467

Haupt 1989 Haupt, R.: A survey of priority rule-based-scheduling. In: *OR Spektrum* 11 (1989), S. 3–16

Hermann u. a. 1997 Hermann, J.; Proth, J.; Sauer, N.: Heuristics for Unrelated Machine Scheduling with Precedence Constraints. In: *European Journal of Operational Research* 102 (1997), S. 528–537

Holland 1975 Holland, J.: *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975

Holtsclaw und Uzsoy 1996 Holtsclaw, H.; Uzsoy, R.: Machine criticality measures and subproblem solution procedures in shifting bottleneck methods: A computational study. In: *Journal of Operational Research Society* 47 (1996), S. 666–677

Hu u. a. 2010 Hu, X.; Bao, J.; Jin, Y.: Minimizing makespan on parallel machines with precedence constraints and machine eligibility restrictions. In: *International Journal of Production Research* 48 (2010), Nr. 6, S. 1639–1651

Huang u. a. 2007 Huang, H.; Lu, C.; Fu, L.: Lot dispatching and scheduling integrating OHT traffic information in the 300mm Wafer Fab. In: *IEEE International Conference on Automation Science and Engineering.*, September 2007, S. 495–500

Hurink und Knust 2001a Hurink, J.; Knust, S.: List scheduling in a parallel machine environment with precedence constraints and setup times. In: *Operations Research Letters* 29 (2001), S. 231–239

Hurink und Knust 2001b Hurink, J.; Knust, S.: Makespan minimization for flow-shop problems with transportation times and a single robot. In: *Discrete Applied Mathematics* 112 (2001), September, S. 199–216. – Issues 1-3, 15 September 2001

Hurink und Knust 2002 Hurink, J.; Knust, S.: A tabu search algorithm for scheduling a single robot in a job-shop environment. In: *European Journal of Operations Research* 119 (2002), Nr. 1-2, S. 181–203

Hurink und Knust 2005 Hurink, J.; Knust, S.: Tabu search algorithms for job-shop problems with a single transport robot. In: *European Journal of Operations Research* 162 (2005), Nr. 1, S. 99–111

Ivens und Lambrecht 1996 Ivens, P.; Lambrecht, M.: Extending the shifting bottleneck procedure to real-life applications. In: *European Journal of Operational Research* 90 (1996), S. 252–268

Jacobs 1984 Jacobs, F.: OPT uncovered. In: *Industrieal Engineering* 16 (1984), S. 32–35

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

Jeong und Randhawa 2001 Jeong, B.; Randhawa, S.: A multi-attribute dispatching rule for automated guided vehicle system. In: *International Journal of Production Research* 39 (2001), Nr. 13, S. 2817–2832

Jimenez u. a. 2002 Jimenez, J.; Kim, B.; Fowler, J.; Mackulak, G.; Choung, Y.; Kim, D.: Operational modeling and simulation of an inter-bay AMHS in semiconductor

wafer fabrication. In: *Proceedings of the 2002 Winter Simulation Conference*, 2002, S. 377–382

Kennedy und Eberhart 1995 Kennedy, J.; Eberhart, R. C.: Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks*. Piscataway, NJ., 1995, S. 1942–1948

Kim u. a. 2006 Kim, S.; Choi, H.; Lee, D.: Tabu Search heuristics for parallel machine scheduling with sequence-dependent setup and ready times. In: *Proceedings ICCSA 2006, LNCS 3982*, 2006, S. 728–737

Kirkpatrick u. a. 1983 Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P.: Optimization by simulated annealing. In: *Science* (1983), S. 671–680

Klemmt u. a. 2009 Klemmt, A.; Weigert, G.; Almeder, C.; Mönch, L.: A comparison of MIP-based decomposition techniques and VNS approaches for batch scheduling problems. In: *Proceedings of the Modeling and Analysis of Semiconductor Manufacturing Conference (MASM 2009)*, 2009, S. 1686–1694

Knust 1999 Knust, S.: *Shop-scheduling problems with transportation*, Fachbereich Mathematik/Informatik, Universität Osnabrück, Dissertation, 10 1999

Kurbel 2003 Kurbel, K.: *Produktionsplanung und -steuerung*. 5. München, Wien : Oldenburg Wissenschaftsverlag GmbH, 2003

Kurosaki u. a. 1997 Kurosaki, R.; Nagao, N.; Komada, H.; Watanabe, Y.; Yano, H.: AMHS for 300-mm wafer. In: *IEEE International Symposium on Semiconductor Manufacturing Conference*, 1997, S. 13–16

Lacomme und Larabi 2007 Lacomme, P.; Larabi, M.: A disjunctive graph for the job-shop with several robots. In: Baptiste, Philippe (Hrsg.); Kendall, Graham (Hrsg.); Sourd, Alix M. (Hrsg.): *MISTA 2007 - Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Application*, 2007

Lacomme u. a. 2005 Lacomme, P.; Moukrim, A.; Tchernev, N.: Simultaneously job input sequencing and vehicle dispatching in a single vehicle AGVS: a heuristic branch and bound approach coupled with a discrete events simulation model. In: *International Journal of Production Research* 43 (2005), Nr. 9, S. 1911–1942

Land und Doig 1960 Land, A. H.; Doig, A. G.: An automatic method of solving discrete programming problems. In: *Econometrica* 28 (1960), S. 497–520

Laporte 1992 Laporte, G.: The vehicle routing problem: an overview of exact and approximate algorithms. In: *European Journal of Operational Research* 59 (1992), S. 345–358

Lawler 1978 Lawler, E.: Sequencing jobs to minimize total weighted completion time subject to precedence constraints. In: *Annals of Discrete Mathematics* 2 (1978), S. 75–90

Lawler 1977 Lawler, E. L.: A "pseudopolymomialälgorithm for sequencing jobs to minimize total tardiness. In: *Annals of Discrete Mathematics* 1 (1977), S. 331–342

Le-Anh und De Koster 2006 Le-Anh, T.; De Koster, M.: A review of design and control of automated guided vehicle systems. In: *European Journal of Operational Research* 121 (2006), S. 1–23

Lee und Pinedo 1997 Lee, Y.; Pinedo, M.: Scheduling jobs on parallel machines with sequence-dependent setup times. In: *European Journal of Operational Research* 100 (1997), S. 464–474

Lenstra u. a. 1977 Lenstra, J.; Kan, A. R.; Brucker, P.: Complexity of machine scheduling problems. In: *Annals of Discrete Mathematics* 1 (1977), S. 343–362

Lenstra und Kan 1981 Lenstra, J. K.; Kan, A.H.G. R.: Complexity of vehicle routing and scheduling problems. In: *Networks* 11 (1981), S. 211–227

Liao und Cheng 2007 Liao, C.; Cheng, C.: A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. In: *Computers & Industrial Engineering* 52 (2007), S. 404–413

Liao und Juan 2007 Liao, C.; Juan, H.: An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. In: *Computers & Operations Research* (2007), S. 1899–1909

Liao und Fu 2004 Liao, D.; Fu, H.: Speedy delivery - dynamic OHT allocation and dispatching in large-scale, 300-mm AMHS management. In: *Robotics & Automation Magazine, IEEE* 11 (2004), S. 22–32

Liao und Wang 2006 Liao, D.; Wang, C.: Differentiated preemptive dispatching for automatic materials handling services in 300 mm semiconductor foundry. In: *International Journal of Advanced Manufacturing Technology* 29 (2006), S. 890–896

Lin u. a. 2001 Lin, J.; Wang, F.; Yen, P.: Simulation analysis of dispatching rules for an automated interbay material handling system in wafer fab. In: *International Journal of Production Research* 39 (2001), April, Nr. 6, S. 1221–1238

Lin u. a. 2004 Lin, J.; Wang, F.; Young, J.: Virtual vehicle in the connecting transport automated material-handling system (AMHS). In: *International Journal of Production Research* 42 (2004), S. 2599–2610

Lin und Ying 2007 Lin, S.; Ying, K.: Solving single machine total weighted tardiness problems with sequence dependent setup times by meta-heuristics. In: *International Journal of Advanced Manufacturing Technology* 34 (2007), S. 1183–1190

Lou u. a. 2006 Lou, X.; Chu, C.; Wang, C.: Some dominance properties for single-machine tardiness problems with sequence-dependent setup. In: *International Journal of Production Research* 44 (2006), Nr. 17, S. 3367–3378

Mackulak u. a. 1998 Mackulak, G.; Lawrence, F.; Rayter, J.: Simulation analysis of 300mm intrabay automation vehicle capacity alternatives. In: *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 1998, S. 445–450

Mason u. a. 2002 Mason, S.; Fowler, J.; Carlyle, W. M.: A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. In: *Journal of Scheduling* 5 (2002), Nr. 3, S. 247–262

Mason u. a. 2004 Mason, S.; Qu, P.; Kutanoglu, E.; Fowler, J.: Abstract The single machine multiple orders per job scheduling problem. In: *IIE Transactions on Scheduling and Logistics* (2004). – Zur Begutachtung eingereicht

Mesarovic und Takahara 1989 Mesarovic, M.; Takahara, Y. ; Thoma, M. (Hrsg.); Wyner, A. (Hrsg.): *Abstract Systems Theory. Lecture Notes in Control and Information Sciences*. Berlin, Heidelberg, New York, London, Paris, Tokyo : Springer Verlag, 1989

Mladenović und Hansen 1997 Mladenović, N.; Hansen, P.: Variable neighborhood search. In: *Computers and Operations Research* 24 (1997), S. 1097–1100

Montoya-Torres 2006 Montoya-Torres, J.: A literature survey on the design approaches and operational issues of automated wafer-transport systems for wafer fabs. In: *Production Plannng & Control* 17 (2006), Nr. 6, S. 648–663

Morton 1981 Morton, T.: Forward algorithms for forward-thinking-managers. In: *Applications of Management* (1981), S. 1–55

Mönch 2005 Mönch, L.: Simulation-based asseessment of order release strategies for a distributed shifting bottleneck heuristic. In: *Proceedings of the 2005 Winter Simulation Conference*, 2005, S. 2186–2193

Mönch 2006 Mönch, L.: *Agentenbasierte Produktionssteuerung komplexer Produktionssysteme*. Wiesbaden : Gabler, DUV-Verlag, 2006

Mönch u. a. 2005 Mönch, L.; Balasubramanian, H.; Fowler, J.; Pfund, M.: Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. In: *Computers & Operations Research* 32 (2005), S. 2731–2750

Mönch und Drießel 2005 Mönch, L.; Drießel, R.: A distributed shifting bottleneck heuristic for complex job shops. In: *Computers & Industrial Engineering* 49 (2005), November, Nr. 3, S. 363–380

Mönch und Drießel 2010 Mönch, L.; Drießel, R.: Variable Neighborhood Search approaches for scheduling jobs on parallel machines with sequence dependent setup times, precedence constraints, and ready times. In: *Computers & Industrial Engineering* (2010). – angenommen zur Veröffentlichung

Mönch u. a. 2011 Mönch, L.; Fowler, J.; Mason, S.; Dauzere-Peres, S.; Rose, O.: A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. In: *Journal of Scheduling* (2011). – Akzeptiert zur Veröffentlichung

Mönch und Gmilkowsky 2001 Mönch, L.; Gmilkowsky, P.: Steuerung des Waferfertigungsprozesses: ein agentenorientierter Ansatz. In: *Industrie Management, Themenheft Agententechnologie* 6 (2001), S. 17–20

Mönch und Rose 2004 Mönch, L.; Rose, O.: Shifting-Bottleneck-Heuristik für komplexe Produktionssysteme: softwaretechnische Realisierung und Leistungsbewertung. In: Suhl, L. (Hrsg.); Voö, S. (Hrsg.): *Proceedings Multi-Konferenz Wirtschaftsinformatik, Teilkonferenz „Quantitative Methoden in ERP und SCM", DSOR Beiträge zur Wirtschaftsinformatik 2*, 2004, S. 145–159

Mönch u. a. 2002 Mönch, L.; Rose, O.; Sturm, R.: Framework for the performance assessment of Shop-Floor Control Systems. In: Fowler, J. (Hrsg.); Cochran, J. (Hrsg.): *Proceedings of the 2002 Modeling and Analysis of Semiconductor Manufacturing Conference (MASM 2002)*, 2002, S. 95–100

Mönch u. a. 2003 Mönch, L.; Rose, O.; Sturm, R.: A simulation framework for performance assessment of shop-floor control systems. In: *SIMULATION* 79 (2003), 03, Nr. 3, S. 163–170

Mönch u. a. 2007 Mönch, L.; Schabacker, R.; Pabst, D.; Fowler, J.: Genetic algorithm-based subproblem solution procedures for a modified shifting bottleneck heuristic for complex job shops. In: *European Journal of Operational Research* 177 (2007), März, Nr. 3, S. 2100–2118

Mönch und Zimmermann 2007 Mönch, L.; Zimmermann, J.: Simulation-based assessment of machine criticality measures for a shifting bottleneck scheduling approach in complex manufacturing systems. In: *Computers in Industry* 58 (2007), Nr. 7, S. 644–655

Mönch und Zimmermann 2011 Mönch, L.; Zimmermann, J.: A computational study of a shifting bottleneck heuristic for multi-product complex job shops. In: *Production Planning & Control* 22 (2011), Nr. 1, S. 25–40

Narasimhan u. a. 2005 Narasimhan, A. Barua R.; Upasani, A.; Uzsoy, R.: Implementing global factory schedules in the face of stochastic disruptions. In: *International Journal of Production Research* 43 (2005), S. 793–818

Nazzal 2006 Nazzal, D.: *Analytical approach to estimating AMHS performance in 300mm fabs.*, Georgia Institute of Technology, Dissertation, 2006

Nazzal und Bodner 2003 Nazzal, D.; Bodner, D. A.: A simulation-based design framework for automated material handling systems in 300mm fabrication facilities.

In: Chick, S. (Hrsg.); Sánchez, P. J. (Hrsg.); Ferrin, D. (Hrsg.); Morrice, D. J. (Hrsg.): *Proceedings of the 2003 Winter Simulation Conference*, 2003

Nazzal und El-Nashar 2007 Nazzal, D.; El-Nashar, A.: Survey of research in modeling conveyor-based automated material handling systems in wafer fabs. In: Henderson, S. G. (Hrsg.); B. Biller, M.-H. H. (Hrsg.); Shortle, J. (Hrsg.); Tew, J. D. (Hrsg.); Barton, R. R. (Hrsg.): *Proceedings of the 2007 Winter Simulation Conference*, 2007

Nyhuis und Wiendahl 1999 Nyhuis, P.; Wiendahl, H.: *Logistische Kennlinien: Grundlagen, Werkzeuge und Anwendungen.* Berlin-Heidelberg : Springer, 1999

Oey und Mason 2001 Oey, K.; Mason, S.: Scheduling batch processing machines in complex job shops. In: *Proceedings of the 2001 Winter Simulation Conference*, 2001, S. 1200–1207

Osisek und Aytug 2004 Osisek, V.; Aytug, H.: Discovering subproblem prioritization rules for shifting bottleneck algorithms. In: *Journal for Intelligent Manufacturing* 15 (2004), S. 55–67

Ovacik und Uzsoy 1992 Ovacik, I.; Uzsoy, R.: The shifting bottleneck algorithm for scheduling semiconductor testing operations. In: *Journal of Electronics Manufacturing* 2 (1992), S. 119–134

Ovacik und Uzsoy 1995 Ovacik, I.; Uzsoy, R.: Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent setup times. In: *International Journal of Production Research* 33 (1995), S. 3173–3292

Ovacik und Uzsoy 1997 Ovacik, I. M.; Uzsoy, R.: *Decomposition methods for complex factory scheduling problems.* 1. Auflage. Boston/Dordrecht/London : Kluwer Academic Publishers, 1997

Park u. a. 2000 Park, Y.; Kim, S.; Lee, Y.: Scheduling jobs on parallel machines applying neural networks and heuristic rules. In: *Computers & Industrial Engineering* 38 (2000), S. 189–202

Parragh u. a. 2008a Parragh, S.; Doerner, K.; Hartl, R.: A survey on pickup and delivery problems: Part I: Transportation between customers and depot. In: *Journal für Betriebswirtschaft* 58 (2008), Nr. 1, S. 21–51(31)

Parragh u. a. 2008b Parragh, S.; Doerner, K.; Hartl, R.: A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. In: *Journal für Betriebswirtschaft* 58 (2008), Nr. 2, S. 81–117(31)

Pfund u. a. 2008a Pfund, M.; Balasubramanian, H.; Fowler, J.; Mason, S.; Rose, O.: A multi-criteria approach for scheduling semiconductor wafer fabrication facilities. In: *Journal of Scheduling* 11 (2008), Nr. 1, S. 29–47

Pfund u. a. 2008b Pfund, M.; Fowler, J.; Gadkari, A.; Chen, Y.: Scheduling jobs on parallel machines with setup times and ready times. In: *Computers & Industrial Engineering* 54 (2008), S. 764–782

Pfund u. a. 2006 Pfund, M.; Mason, S.; Fowler, J.: Kap. Dispatching and scheduling in semiconductor manufacturing, S. 213–241. In: Herrmann, J. (Hrsg.): *Handbook of Production Scheduling.* Heidelberg : Springer, 2006

Pinedo 2001 Pinedo, M.: *Scheduling: Theory, Algorithm, and Systems.* Second Edition. Prentice Hall, Englewood Cliffs, 2001

Pinedo und Singer 1999 Pinedo, M.; Singer, M.: A shifting-bottleneck-heuristic for minimizing the total weighted tardiness in a job shop. In: *Naval Research Logistics* 46 (1999), S. 1–17

Pinson 1995 Pinson, E.: The job shop scheduling problem: a concise survey and some recent developments. In: Chrétienne, P. (Hrsg.); Coffman, E. (Hrsg.); Lenstra, J. (Hrsg.): *Scheduling Theory and its Applications.* Wiley, 1995, S. 277–293

Plata 1997 Plata, J.: 300-mm fab design – a total factory perspective. In: *Proceedings of the IEEE International symposium on Semiconductor Manufacturing Conference*, 1997, S. 5–8

Potts und Kovalyov 2000 Potts, C.; Kovalyov, M.: Scheduling with batching: a review. In: *European Journal of Operational Research* 120 (2000), S. 228–249

Qu u. a. 2004 Qu, P.; Steinmiller, B.; Mason, S.: Incorporating automated material handling systems into a disjunctive graph. In: *Proceedings of the 2004 Industrial Engineering Research Conference*, 2004

Ragatz und Mabert 1988 Ragatz, G.; Mabert, V.: An evaluation of order release mechanisms in a job-shop environment. In: *Decision Sciences* 19 (1988), Nr. 1, S. 167–189

Reinisch 1974 Reinisch, Karl: *Kybernetische Grundlagen und Beschreibung kontinuierlicher Systeme.* Berlin : VEB Verlag Technik, 1974

Rocha u. a. 2007 Rocha, M.; Ravetti, M.; Mateus, G.; Pardalos, P.: Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. In: *IMA Journal of Management Mathematics* 18 (2007), S. 101–115

Roderick u. a. 1992 Roderick, L.; Phillips, D.; Hogg, G.: A comparison of order release strategies in production control systems. In: *International Journal of Production Research* 30 (1992), Nr. 3, S. 611–626

Rose 2001 Rose, O.: The shortest processing time first (SPTF) dispatch rule and some variants in semiconductor manufacturing. In: Peters, B. (Hrsg.); Smith, J. (Hrsg.);

Medeiros, D. (Hrsg.); Rohrer, M. (Hrsg.): *Proceedings of the 2001 Winter Simulation Conference*, 2001, S. 1220–1224

Rose 2002 Rose, O.: Some issues of the critical ratio dispatch rule in semiconductor manufacturing. In: Yücesan, E. (Hrsg.); Chen, C. (Hrsg.); Snowdon, J. (Hrsg.); Charnes, J. (Hrsg.): *Proceedings of the 2002 Winter Simulation Conference*, 2002, S. 1401–1405

Rose 2003 Rose, O.: Comparison of due-date oriented dispatch rules in semiconductor manufacturing. In: *In Proceedings of the 2003 Industrial Engineering Research Conference*, 2003, S. 18–20

Roy und Sussmann 1964 Roy, B.; Sussmann, B.: Les problèmes d'ordonnancement avec contraintes disjonctives. / SEMA. 1964 (9). – Forschungsbericht. Note D.S

SEMATECH 2003 SEMATECH: *SEMI International Standard E47.1: Provisional Mechanical Specification for Boxes and Pods Used to Transport and Store 300 mm Wafers.* 03 2003

Sen u. a. 2003 Sen, T.; Sulek, J.; Dileepan, P.: Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. In: *International Journal of Production Economics* 83 (2003), S. 1–12

Sevkli und Aydin 2006 Sevkli, M.; Aydin, M. E.: A variable neighbourhood search algorithm for job shop scheduling problems. In: *Proceedings EvoCOP 2006, LNCS 3906*, 2006, S. 261–271

Siek u. a. 2002 Siek, J.; Lee, L.; Lumsdaine, A.: *The Boost Graph Library - User Guide and Reference Manual.* Addison Wesley, 2002

Simon 1962 Simon, H.: The architecture of complexity. In: *Proceedings of the American Philosophical Society* Bd. 106, 1962, S. 467–482

Smith u. a. 1999 Smith, J.; Peters, B.; Srinivasan, A.: Job shop scheduling considering material handling. In: *International Journal of Production Research* 37 (1999), Nr. 7, S. 1541–1560

Sourirajan und Uzsoy 2007 Sourirajan, K.; Uzsoy, R.: Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication. In: *Journal of Scheduling* 10 (2007), S. 41–65

Spier und Kempf 1996 Spier, J.; Kempf, K.: Simulation of emergent behavior in manufacturing systems. In: *Proceedings of the IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 1996, S. 90–94

Strusevich 1999 Strusevich, V.: A heuristic for the two-machine open-shop scheduling problem with transportation time. In: *Discrete Applied Mathematics* 93 (1999), Nr. 2-3, S. 287–304

Stützle und Hoos 2000 Stützle, T.; Hoos, H.: MAX–MIN ant system. In: *Future Generation Computer Systems* 16 (2000), Nr. 8, S. 889–914

Thiel u. a. 1998 Thiel, M.; Schulz, R.; Gmilkowsky, P.: Simulation-based production control in the semiconductor industry. In: Medeiros, D. (Hrsg.); Watson, E. (Hrsg.); Carson, J. (Hrsg.); Manivannan, M. (Hrsg.): *Proceedings of the 1998 Winter simulation conference*, 1998

Tyan u. a. 2004 Tyan, J.; Du, T.; Chen, J.; Chang, I.: Multiple response optimization in a fully automated FAB: an integrated tool and vehicle dispatching strategy. In: *Computers & Industrial Engineering* 46 (2004), S. 121–131

Upasani u. a. 2006 Upasani, A.; Uzsoy, R.; Sourirajan, K.: A problem reduction approach for scheduling semiconductor wafer fabrication facilities. In: *IEEE Transactions on Semiconductor Manufacturing* 19 (2006), Nr. 2, S. 216–225

Uzsoy u. a. 1992 Uzsoy, R.; Lee, C.; Martin-Vega, L.: A review of production planning and scheduling models in the semiconductor industry part I: system characteristics, performance evaluation and production planning. In: *IIE Transactions on Scheduling and Logistics*, 1992 (24 4), S. 47–61

Uzsoy u. a. 1994 Uzsoy, R.; Lee, C.; Martin-Vega, L.: A review of production planning and scheduling models in the semiconductor industry part II: shop floor control. In: *IIE Transactions on Scheduling and Logistics*, 1994 (26 5), S. 44–55

Uzsoy und Wang 2000 Uzsoy, R.; Wang, C.: Performance of decomposition procedures for job-shop scheduling problems with bottleneck machines. In: *International Journal of Production Research* 38 (2000), S. 1271–1286

Van de Vegte 1993 Van de Vegte, J.: *Feedback Control Systems*. Englewood Cliffs : Prentice Hall, 1993

Varadarajan und Sarin 2006 Varadarajan, A.; Sarin, S.: A Survey of dispatching rules for operational control in wafer fabrication. In: *12th IFAC Symposium on Information Control Problems in Manufacturing* Bd. 12, 2006

VDI 2007 VDI: *Richtlinie 5600: Fertigungsmanagementsysteme – Manufacturing Execution Systems (MES)*. Berlin : Beuth Verlag, 2007

Vepsalainen und Morton 1987 Vepsalainen, A.; Morton, T.: Priority rules for job shops with weighted tardiness costs. In: *Management Science* 33 (1987), Nr. 8, S. 1035–1047

Wall 1999 Wall, M.: *Galib: A C++ library of genetic algorithms components*. http://lancet.mit.edu/ga/. 1999

Wang und Tang 2009 Wang, X.; Tang, L.: A population-based variable neighborhood search for the single machine total weighted tardiness problem. In: *Computers & Operations Research* 36 (2009), Nr. 6, S. 2105–2110

Abkürzungen

ACO	Ant Colony Optimization
ACT	Average Cycle Time
AMD	Advanced Micro Devices *ein US-amerikanischer Chip-Entwickler*
ATC	Apparent Tardiness Cost
ATC-NLF	Kombiniertes Steuerungsverfahren auf Basis von ATC und NLF
ATCR	Apparent Tardiness Cost with Release Times
ATCS	Apparent Tardiness Cost with Setups
ATCSR	Apparent Tardiness Cost with Setup and Release Times
ATC-SSP	SSP auf Basis von ATC
AWT	Average Weighted Tardiness
BDE	Betriebsdatenerfassung
B-VNS	Basic VNS
BN-VNS	Best Neighbor VNS
CT	Cycle Time
DLL	Dynamic Link Library
MICA	Modifizierter Infinite Capacity Algorithm
ERP	Enterprise Resource Planning
FIFO	First-In, First-Out
FIFO-NLF	Kombiniertes Steuerungsverfahren auf Basis FIFO und NLF
FIFO-SSP	SSP auf Basis von FIFO
FOUP	Front-Open-Unified-Pod
F-VNS	Fast VNS
FS-VNS	Fast Skewed VNS
GA	Genetischer Algorithmus
GRASP	Greedy Randomized Adaptive Search Procedure
G-VNS	General VNS
GS-VNS	General Skewed VNS
ICA	Infinite Capacity Algorithm
MES	Manufacturing Execution System
MCS	Manufacturing Control System
MCM	Model Communication Module
MDE	Maschinendatenerfassung
MIMAC	Measurement and Improvement of Manufacturing Capacity
NLF	Nearest Lot First
OPT	Optimized Production Technology

PA	Permutations-Ansatz
RAM	Random-Access Memory
RFID	Radio-Frequency Identification
RHP	Rolling Horizon Procedure
R-VNS	Reduced VNS
SBH	Shifting-Bottleneck-Heuristik
SBH-HT	SBH mit hierarchischer Berücksichtigung von Transportzeiten
SBH-ST	SBH mit simultaner Berücksichtigung von Transportzeiten
SSP	Sub Solution Procedure
S-VNS	Simple VNS
TP	Throughput
TWT	Total Weighted Tardiness
UML	Unified Modeling Language
VNFD	Variable Neighborhood First Descent
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
WIP	Work in Progress

Index

Curriculum Vitae

Persönliche Angaben

Name:	René Drießel
Geburtsdatum:	29. März 1980
Geburtsort:	Vieselbach

Schulausbildung

1990–1998	Von-Bülow-Gymnasium Neudietendorf *Abschluss:* Abitur
1986–1990	Grundschule Ingersleben

Wehrdienst

1998–1999	Grundwehrdienst

Hochschulausbildung

1999–2004	Studiengang Wirtschaftsinformatik, Technische Universität Ilmenau, *Abschluss:* Diplom-Wirtschaftsinformatiker

Berufliche Tätigkeit

ab 2009	Betriebskoordinator SAP-Basis, Hessische Zentrale für Datenverarbeitung, Wiesbaden
2006–2009	Wissenschaftlicher Mitarbeiter, FernUniversität in Hagen
2004–2006	SAP-Berater Systemintegration, X-CASE GmbH, Ilmenau
2001–2004	Praktikant und Freier Unternehmensberater, X-CASE GmbH, Ilmenau
2000–2004	Studentische Hilfskraft, Technische Universität Ilmenau