Strukturuntersuchungen für Shop-Scheduling-Probleme:

Anzahlprobleme, potentielle Optimalität und neue Enumerationsalgorithmen

DISSERTATION

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.),

genehmigt durch
die Fakultät für Mathematik
der Otto-von-Guericke-Universität Magdeburg

von Diplommathematiker Martin HARBORTH

geb. am:	21. Oktober 1968
in	Braunschweig

Gutachter:	Prof. Dr. Heidemarie Bräsel
	Prof. Dr. Johannes Terno
	Prof. Dr. Peter Brucker

Eingereicht am:	21.05.1999
Verteidigung am:	31.08.1999

Abstract

Structural analysis of shop scheduling problems: number problems, potential optimality, and new enumeration algorithms.

This thesis deals with the number and the structure of the solutions for shop scheduling problems. The basic notation and preliminaries which are relevant for the problems in consideration can be found in Chapter 2.

We are starting the structural investigation from the classical open shop problem with n jobs and m machines. A feasible solution of such a problem is represented by a sequence graph (directed acyclic graph) or a sequence (certain Latin rectangle). In Chapter 3 we discuss the modeling concepts and we prove some statements about sequence graphs and sequences.

Chapter 4 contains the general determination of the number of sequences. Here, exact formulae for the number of sequences for open shop problems with up to three machines and an arbitrary number of jobs or vice versa have been found. Furthermore, new upper and lower bounds for the number of $n \times m$–sequences are developed. These results are mainly based on the estimation of the chromatic polynomial of the Hamming graph $K_n \times K_m$.

The main topic of Chapter 5 is a new enumeration method which gives us the ability to generate all $n \times m$–sequences efficiently at least for small values of n and m. This procedure results from the construction of equivalence classes in which we collect sequences with the same basic structure.

In Chapter 6 we present and compare two concepts which serve for the determination of optimality criteria for sequences. With the aid of the concept of irreducibility we are able to reduce the set of all sequences to a set of potentially-optimal sequences, in which there is always an optimal sequence regardless of the given processing times. By means of the concept of stability we can characterize optimal sequences for given processing times, which remain optimal if there are certain deviations from these processing times. Such sequences are called stable sequences.

In Chapter 7 we discuss various methods for the enumeration of potentially-optimal sequences. These methods allow us to generate a comparatively small set of potentially-optimal sequences only instead of the whole set of $n \times m$–sequences for a given problem. Thus we can restrict to such a small set if we are searching for an optimal sequence for a given open shop problem with n jobs and m machines.

From the ratio of the total number of sequences and the corresponding number of potentially-optimal sequences we can deduce interesting statements about the differences in the hardness of the classical job shop problem with different machine orders of the jobs.

Danksagung

Für ihre fachkundige und freundliche Betreuung und Unterstützung beim Erstellen dieser Arbeit bin ich Prof. Dr. Heidemarie Bräsel und Dr. Thomas Tautenhahn sehr dankbar. Die Anfertigung dieser Dissertation wurde mir anhand eines Stipendiums im Rahmen der Graduiertenförderung des Landes Sachsen-Anhalt sowie durch das vom Land Sachsen-Anhalt finanzierte Projekt „Lateinische Rechtecke in der Schedulingtheorie“ ermöglicht. Mein ganz besonderer Dank gilt Dr. Eva Nuria Müller und Dipl.-Math. Per Willenius, denn ihre Korrekturvorschläge waren mir bei der Durchsicht der Arbeit eine große Hilfe. Ebenso möchte ich mich bei meinen Eltern und Freunden für die Unterstützung in vielen Bereichen bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die *Schedulingtheorie* befaßt sich mit der Optimierung der zeitlichen Zuordnung von knappen Ressourcen zu bestimmten Aktivitäten und kann als ein Teilgebiet des *Operations Research* aufgefaßt werden. Wegen der großen praktischen Relevanz in der betrieblichen Produktionsplanung erfährt die Schedulingtheorie seit den fünfziger Jahren eine enorme Entwicklung, die unter anderem an der Vielzahl von Publikationen in diesem Bereich abgelesen werden kann. Durch die obige Beschreibung der Optimierungsprobleme in der Schedulingtheorie ergibt sich eine große Anzahl verschiedener Problemtypen. Zusätzlich sind die aus den praktischen Gegebenheiten abgeleiteten Modellierungen dieser Probleme durch die verschiedenen Restriktionen und Anforderungen sehr vielfältig. Viele Schedulingprobleme treten auch in Lebensbereichen auf, die nicht direkt aus dem Anwendungsgebiet der Produktionsplanung stammen. Beispielsweise seien hier Probleme bei der Stundenplanerstellung in Schulen oder bei der Optimierung des zeitlichen Zusammenspiels von Computerprozessoren genannt.

Typischerweise existiert für Schedulingprobleme eine sehr große Anzahl zulässiger Zuordnungen bzw. Lösungen. Beim Auffinden einer optimalen Lösung kann man sich zwar in der Regel auf eine bestimmte endliche Menge von zulässigen Lösungen beschränken, aber die Anzahl dieser Lösungen ist meistens immer noch so groß, daß sich deren Untersuchung selbst mit Hilfe moderner Computertechnik als zu aufwendig herausstellt. In diesem Zusammenhang ist eine Unterscheidung zwischen einfachen und schwierigen Problemen bezüglich der Laufzeit entsprechender Lösungsalgorithmen wünschenswert.

Die Anwendung der *Komplexitätstheorie* für Schedulingprobleme ist zu einem wichtigen Instrument geworden, denn sie erlaubt eine formale Interpretation der empirischen Unterscheidung zwischen einfachen und schwierigen kombinatorischen Optimierungsproblemen. Diese Unterscheidung beruht auf einer Identifizierung der *einfachen Probleme* mit Problemen, deren Lösung eine Zeit in Anspruch nimmt, die nur durch eine polynomiale Funktion der Problemgröße beschränkt ist, während

es für die *schweren* bzw. $\mathcal{NP}$*-vollständigen Probleme* unwahrscheinlich ist, daß ein polynomialer Lösungsalgorithmus existiert.

Problemstellung und Modellierung

Bei der gängigen Terminologie für Schedulingprobleme ist eine Menge von *Aufträgen* und eine Menge von *Maschinen* gegeben, wobei die Aktivitäten den Aufträgen entsprechen und die Ressourcen durch die Maschinen repräsentiert werden. Ein Schedulingproblem mit mehr als einem Auftrag und mehr als einer Maschine, bei dem zu gegebenen Aufträgen und Maschinen jeder Auftrag auf jeder Maschine für eine bestimmte Zeit zu bearbeiten ist, wird als *Shop-Scheduling-Problem* bezeichnet. Es gelten die für die meisten Schedulingprobleme üblichen Bedingungen, daß zu jedem Zeitpunkt jede Maschine höchstens einen Auftrag gleichzeitig bearbeitet, und jeder Auftrag auf höchstens einer Maschine gleichzeitig bearbeitet werden kann.

Ein spezielles Shop-Scheduling-Problem, das sogenannte *Open-Shop-Problem*, ist zentraler Ausgangspunkt der hier vorgestellten Untersuchungen. Bei diesem Problem ist die Reihenfolge, in der ein Auftrag von den Maschinen bearbeitet wird und in der eine Maschine die Aufträge bearbeitet, für alle Aufträge und Maschinen beliebig wählbar. Die Optimierungsaufgabe liegt in der Bestimmung dieser Reihenfolgen, so daß eine zulässige Lösung entsteht und dabei eine gegebene Zielfunktion minimiert wird, die für gewöhnlich als eine Funktion in den Fertigstellungszeiten der Aufträge definiert ist.

Diese Art von Schedulingproblemen tritt in vielen Lebensbereichen auf. Stellvertretend wird ein Beispiel aus der Praxis gegeben: In Kraftfahrzeugwerkstätten kann die Ablaufplanung für die anfallenden Inspektionen als ein bestimmtes Open-Shop-Problem aufgefaßt werden. Die Aufträge entsprechen hierbei den Fahrzeugen und die Abteilungen übernehmen die Rolle der Maschinen. In den Abteilungen werden die bezüglich ihrer Reihenfolge voneinander unabhängigen Wartungsarbeiten (z. B. Ölwechsel, Prüfung der Bremsen, Prüfung der Elektrik, usw.) durchgeführt, wobei die Bearbeitungszeiten für die einzelnen Wartungsarbeiten vorher bekannt sind. Es wird angenommen, daß zu jedem Zeitpunkt immer nur höchstens eine Wartungsarbeit an einem Fahrzeug vorgenommen, und höchstens ein Fahrzeug in einer Abteilung gewartet werden kann. Das Ziel ist eine möglichst geringe Gesamtbearbeitungszeit für alle anfallenden Inspektionen. Auf diese Weise sind alle Fahrzeuge wieder so früh wie möglich einsatzbereit.

Sehr anschaulich können Shop-Scheduling-Probleme anhand von Graphen modelliert werden. Ein *Graph* G ist ein geordnetes Paar $G = (V, E)$, bestehend aus einer Menge $V \neq \varnothing$ von *Knoten* und einer Menge E von *Kanten*, wobei jede Kante eine zweielementige Teilmenge $\{u, v\}$ mit $u, v \in V$ ist. Bei einem *Digraphen* bzw. *gerichteten Graphen* $G' = (V', E')$ besteht die Menge E' aus geordneten Knoten-

paaren (u, v) mit $u, v \in V'$. In diesem Fall wird häufig auch von *gerichteten* bzw. *orientierten Kanten* gesprochen. Zur Illustration einer zulässigen Lösung kann eine bestimmte Klasse von Digraphen verwandt werden. Diese *Ablaufgraphen* stehen mit den in der Schedulingtheorie häufig benutzten *disjunktiven Graphen* in engem Zusammenhang, auf den in Kapitel 3 näher eingegangen wird.

Zur Modellierung der Lösungen von Shop-Scheduling-Problemen bietet sich neben den Ablaufgraphen auch das *Blockmatrizenmodell* (siehe BRÄSEL [7]) an, bei dem jede zulässige Lösung eines Problems durch ein eindeutig bestimmtes lateinisches Rechteck repräsentiert ist. Ein *lateinisches Rechteck*[1] ist eine Matrix mit Einträgen aus einer endlichen Menge S, wobei jedes Element aus S höchstens einmal in jeder Zeile und höchstens einmal in jeder Spalte vorkommt.

Die lateinischen Rechtecke, die eine zulässige Lösung beschreiben, haben spezielle Eigenschaften und werden als *Pläne* bezeichnet. Jedem Eintrag eines Planes entspricht eine *Operation*, d. h. einem Bearbeitungsschritt eines Auftrags auf einer Maschine, und jeder Eintrag gibt gleichzeitig die Position seiner zugehörigen Operation innerhalb der Reihenfolge des gesamten Produktionsablaufs wieder.

Strukturuntersuchungen

Die Anzahl der Pläne eines Shop-Scheduling-Problems wird mit steigender Auftrags- und Maschinenanzahl schnell sehr groß und unhandlich. Man kann sich dieses Wachstum zum Beispiel an der Anzahl der Möglichkeiten veranschaulichen, für alle Maschinen die Reihenfolgen der auf ihnen zu bearbeitenden Aufträge festzulegen. Diese Anzahl beträgt $(n!)^m$ bei einem Shop-Scheduling-Problem mit n Aufträgen und m Maschinen, und im Falle $n = 6,\ m = 3$ sind das bereits 373 248 000 Möglichkeiten! Deshalb wird bei vielen *Approximationsalgorithmen* zur Lösung eines Shop-Scheduling-Problems auf eine vollständige Plan-Enumeration verzichtet. Auf diese Weise nähert man den optimalen Zielfunktionswert zwar in der Regel nur bis auf einen bestimmten Faktor an, aber der entsprechende Algorithmus besitzt dafür eine wesentlich kürzere Laufzeit.

Trotzdem ist die Enumeration aller Pläne eines Shop-Scheduling-Problems häufig von Nutzen, denn es können damit allgemeine Aussagen über die Schwierigkeit des betrachteten Problems getroffen werden. Offensichtlich hängt die Auswahl der Pläne, die hinsichtlich der Optimalität günstiger als andere sind, in starkem Maße von der Struktur des zugrunde liegenden Ablaufgraphen ab. Da bereits während der Enumeration das Erkennen von ungünstigen Strukturen möglich ist, kann man viele Pläne schon vor ihrer vollständigen Erzeugung von der weiteren Betrachtung

[1] Im 18. Jh. hat EULER [34] wahrscheinlich als erster den Begriff *lateinisches Quadrat* geprägt, denn er benutzte damals lateinische Buchstaben als Einträge für entsprechende quadratische Matrizen.

ausschließen. Bei der Plan-Enumeration in der vorliegenden Arbeit spielen die vorgegebenen Zeiten für die Bearbeitung eines Auftrags auf einer Maschine zunächst keine Rolle. Die Untersuchungen für Shop-Scheduling-Probleme, die sich auf die rein kombinatorische Frage nach der Art und Anzahl der zugehörigen Pläne bzw. Ablaufgraphen unabhängig von den gegebenen Bearbeitungszeiten beziehen, heißen *Strukturuntersuchungen*.

Die ersten Ergebnisse auf dem Gebiet der Strukturuntersuchungen stammen von AKERS und FRIEDMAN [1] sowie von CONWAY, MAXWELL und MILLER [26]. Die Autoren ermitteln in diesen Arbeiten für bestimmte Shop-Scheduling-Probleme günstige Grundstrukturen zulässiger Lösungen. Auf diese Weise kann man sich bei der Suche nach optimalen Lösungen auf einen kleineren Suchraum beschränken.

In [14, 15] haben BRÄSEL und M. KLEINAU die Resultate aus [1] und [26] für Open-Shop-Probleme verallgemeinert. Darauf aufbauend sind im Rahmen der vorliegenden Arbeit neue Enumerationstechniken entwickelt worden. Mit Hilfe dieser Techniken ist die Plan-Enumeration und die Charakterisierung günstiger Planstrukturen für größere Formate als bisher möglich. Auf diese Weise kann man zum Beispiel alle Pläne mit bis zu 6 Aufträgen und 3 Maschinen vollständig enumerieren und dabei die Pläne mit günstigen Strukturen identifizieren. Eine Reihe von neuen theoretischen Ergebnissen im Bereich der Anzahlproblematik bestätigen und ergänzen die durch die Enumeration entstandenen numerischen Werte. Insgesamt können die hier präsentierten Resultate auch als Weiterentwicklung der Überlegungen und Algorithmen zur Plan-Enumeration aus der Dissertation von M. KLEINAU [55] aufgefaßt werden. Die in [55] benutzten Begriffe und Konzepte werden dabei erweitert und an die gängige graphentheoretische Terminologie angepaßt.

Kapitelübersicht

In Kapitel 2 werden die Grundlagen aus dem Bereich der Schedulingtheorie bereitgestellt, die für die hier untersuchten Probleme relevant sind. Kapitel 3 enthält die Beschreibung der Konzepte und Ergebnisse im Zusammenhang mit der Modellierung von Schedulingproblemen durch Ablaufgraphen und Pläne.

In Kapitel 4 werden bekannte und neue Ergebnisse im Bereich der Anzahlproblematik für Pläne eines gegebenen Formats vorgestellt. Neben einer ausführlichen Übersicht über den aktuellen Stand auf dem Gebiet der Pläne, die den klassischen lateinischen Rechtecken entsprechen, werden in diesem Kapitel neue exakte Werte sowie obere und untere Schranken für die Anzahl allgemeiner Pläne entwickelt. Der zentrale Gegenstand in Kapitel 5 ist ein neues Enumerationsverfahren zur Erzeugung und Anzahlbestimmung der Pläne eines gegebenen Formats. Vorbereitend dazu werden Methoden beschrieben, durch die man Pläne mit gleichartigen Eigenschaften zusammenfassen kann. Weiterhin enthält dieses Kapitel Anzahlbestimmungen, bei

denen jeweils ausschließlich die Reihenfolge, in der ein Auftrag von den Maschinen bearbeitet wird, eine Rolle spielt.

Kapitel 6 ist der Charakterisierung von Plänen mit günstigen Grundstrukturen gewidmet. Es handelt sich um *potentiell-optimale* Pläne, unter denen unabhängig von den gegebenen Bearbeitungszeiten stets ein Plan existiert, der eine optimale Lösung des entsprechenden Shop-Scheduling-Problems repräsentiert. Weiterhin werden in diesem Kapitel Zusammenhänge zwischen potentieller Optimalität und sogenannter Stabilität von Plänen hergestellt. Verschiedene Verfahren zur Enumeration der potentiell-optimalen Pläne werden in Kapitel 7 behandelt. Abschließende Bemerkungen und Einstufungen der erzielten Ergebnisse sollen diese Arbeit in Kapitel 8 abrunden.

Kapitel 2

Grundlagen

Dieses Kapitel behandelt einige Grundbegriffe aus der Scheduling- und Komplexitätstheorie, die notwendig für das Verständnis der Problemstellungen sind.

In der vorliegenden Arbeit werden ausschließlich *deterministische Scheduling-probleme* behandelt, bei denen im Gegensatz zu den *stochastischen Problemen* alle problemdefinierenden Parameter vor dem Optimierungsprozeß bereits bekannt sind. Diese Probleme können nochmals in *Single-Stage-* und *Multi-Stage-Probleme* unterteilt werden, je nachdem ob zur Fertigstellung eines Auftrags eine oder mehr als eine Maschine benötigt wird. Es wird sich hier mit der Klasse der Multi-Stage-Probleme beschäftigt, zu der auch die *Shop-Scheduling-Probleme* gehören, die im anschließenden Abschnitt definiert werden und in der Regel schwerer als vergleichbare Single-Stage-Probleme sind.

2.1 Problem-Klassifikation

Parallel zur Entwicklung der Schedulingtheorie gestaltet sich die Verfeinerung einer detaillierten Problemklassifikation, dessen erste Grundlage im Buch von Conway, Maxwell und Miller [26] geschaffen wurde. Das Klassifikationsschema von Graham *et al.* [41] ist eine Weiterentwicklung des Schemas [26] und umfaßt eine sehr große Anzahl der Schedulingprobleme. Im folgenden werden nur die in der vorliegenden Arbeit benötigten Begriffe und Symbole in Anlehnung an das Klassifikationsschema [41] eingeführt.

Bei einem *Shop-Scheduling-Problem* wird für n, $m \in \mathbb{N}$ mit $n, m \geq 2$ eine Menge von n *Aufträgen (jobs)* $\{J_1, \ldots, J_n\}$ betrachtet, die auf einer Menge von m *Maschinen* $\{M_1, \ldots, M_m\}$ zu bearbeiten sind. Dabei bearbeitet jede Maschine höchstens einen Auftrag gleichzeitig, und jeder Auftrag kann höchstens auf einer Maschine gleichzeitig bearbeitet werden. Jeder Auftrag J_i, $i = 1, \ldots, n$, besteht aus einer Menge von m *Operationen* $\{o_{i1}, \ldots, o_{im}\}$, wobei jede Operation o_{ij}, $j = 1, \ldots, m$,

des Auftrags J_i auf der Maschine M_j während der *Bearbeitungszeit (processing time)* $p_{ij} > 0$ ausgeführt werden muß.

Zwischen je zwei Operationen o_{ik}, o_{il} eines Auftrags J_i oder je zwei Operationen o_{kj}, o_{lj} einer Maschine M_j können *Vorrangbedingungen (precedence constraints)* gegeben sein. Beispielsweise bedeutet eine Vorrangbedingung der Form $o_{i1} \rightarrow o_{i2}$, daß die Operation o_{i1} bereits bearbeitet sein muß, bevor mit der Bearbeitung der Operation o_{i2} des Auftrags J_i begonnen werden darf.

Bei einer Zuordnung der Maschinen zu den Aufträgen gibt die *Fertigstellungszeit (completion time)* $c_{ij} > 0$ der Operation o_{ij} den Zeitpunkt an, bei dem die Bearbeitung der Operation o_{ij} beendet ist. Die *Fertigstellungszeit* C_i des Auftrags J_i bezeichnet den Zeitpunkt, nachdem die Bearbeitung der letzten Operation dieses Auftrags beendet ist, also $C_i = \max_j(c_{ij})$.

Ziel eines Shop-Scheduling-Problem ist das Finden einer zulässigen Lösung, d. h. einer zulässigen zeitlichen Zuordnung von Maschinen und Aufträgen, so daß die Fertigstellungszeiten C_i der Aufträge J_i einem bestimmten Optimalitätskriterium genügen. Mit Hilfe der drei Felder $\boldsymbol{\alpha}|\boldsymbol{\beta}|\boldsymbol{\gamma}$ werden im Klassifikationsschema in [41] die verschiedenen Maschinen- (α), Auftrags- (β) und Optimalitätsmerkmale (γ) von Shop-Scheduling-Problemen wiedergegeben.

Maschinenumgebung ($\boldsymbol{\alpha}$)

Das erste Klassifikationsfeld $\boldsymbol{\alpha = \alpha_1 \alpha_2}$ spezifiziert die Maschinenmerkmale, wobei beim ersten Teilfeld α_1 hier vier Symbole von Interesse sind: $\boldsymbol{\alpha_1 \in \{J, F, O, G\}}$. Diese Symbole bezeichnen Sonderfälle des Shop-Scheduling-Problems, für die ganz bestimmte oder überhaupt keine Vorrangbedingungen bestehen:

$\boldsymbol{\alpha_1 = J}$: Beim *Job-Shop-Problem* sind für jeden Auftrag J_i bezüglich seiner Operationen o_{ij} $(j = 1, \ldots, m)$ Vorrangbedingungen der Form $o_{i,j_1} \rightarrow o_{i,j_2} \rightarrow \cdots \rightarrow o_{i,j_m}$ festgelegt.

$\boldsymbol{\alpha_1 = F}$: Das *Flow-Shop-Problem* ist ein Job-Shop-Problem, bei dem für jeden Auftrag die Vorrangbedingungen dieselben sind, also $o_{i1} \rightarrow o_{i2} \rightarrow \cdots \rightarrow o_{im}$ für alle $i = 1, \ldots, n$.

$\boldsymbol{\alpha_1 = O}$: Beim *Open-Shop-Problem* bestehen zwischen den Operationen o_{ij} keinerlei Vorrangbedingungen.

$\boldsymbol{\alpha_1 = G}$: Beim *General-Shop-Problem* existieren Vorrangbedingungen zwischen beliebigen Operationen einer Maschine bzw. eines Auftrags.

$\boldsymbol{\alpha_2 \in \{m, \circ\}}$: Das Symbol α_2 bezeichnet die *Maschinenanzahl*. Für $\alpha_2 = m$ mit $m \in \mathbb{N}$ wird eine konstante Anzahl m der Maschinen vorausgesetzt, während

man für das leere Symbol $\alpha_2 = \circ$ eine variable Maschinenanzahl annimmt, die dann als Teil der Eingabe eines Algorithmus zur Lösung des betreffenden Shop-Scheduling-Problems aufzufassen ist.

Auftragseigenschaften (β)

Das zweite Klassifikationsfeld $\boldsymbol{\beta = \beta_1, \beta_2, \ldots}$ stellt eine Kombination unterschiedlicher Typen von Nebenbedingungen für die Aufträge J_i dar. Die in der vorliegenden Arbeit betrachteten Nebenbedingungen sind:

$\boldsymbol{\beta_1 \in \{p_{ij} = 1, \circ\}}$**:** Der Eintrag $p_{ij} = 1$ bedeutet, daß jede Operation o_{ij} die Bearbeitungszeit 1 hat, d. h. es bestehen sogenannte *Einheitsbearbeitungszeiten*.

$\boldsymbol{\beta_2 \in \{\underline{p} \leq p_{ij} \leq \overline{p}, \circ\}}$**:** Im Fall $\beta_2 = \underline{p} \leq p_{ij} \leq \overline{p}$ gibt es *konstante untere und obere Schranken* für die Bearbeitungszeiten p_{ij} der Operationen o_{ij}.

$\boldsymbol{\beta_3 \in \{n = k, \circ\}}$**:** Der Eintrag β_3 kann weitere Symbole mit naheliegender Interpretation wie z. B. $n = 2$ für ein Problem mit zwei Aufträgen haben.

Optimalitätskriterium (γ)

Das dritte Klassifikationsfeld $\boldsymbol{\gamma \in \{C_{\max}, \sum C_i, \ldots\}}$ gibt als Optimalitätskriterium die zu minimierende Zielfunktion γ an. Die Zielfunktion ist immer eine Funktion in den Fertigstellungszeiten C_i der Aufträge J_i, also $\gamma = \gamma(C_1, C_2, \ldots, C_n)$. Eine Zielfunktion γ heißt *regulär*, wenn γ nicht-fallend in den C_i ist. Das heißt, wenn bei zwei verschiedenen Lösungen A und B eines Shop-Scheduling-Problems für die Fertigstellungszeiten C_i^A und C_i^B die Beziehungen $C_i^A \leq C_i^B$ für alle $i = 1, \ldots, n$ gelten, folgt für jede reguläre Zielfunktion die Ungleichung

$$\gamma(C_1^A, \ldots, C_n^A) \leq \gamma(C_1^B, \ldots, C_n^B).$$

Es gibt eine Reihe von möglichen Optimalitätskriterien, wobei hier nur die folgenden regulären Zielfunktionen betrachtet werden:

$\boldsymbol{\gamma = C_{\max}}$**:** Das Symbol $C_{\max}$ bezeichnet die zu minimierende *Gesamtbearbeitungszeit (makespan)*. Dies ist die größte Fertigstellungszeit C_i über allen Aufträgen J_i, also $C_{\max} = \max(C_1, \ldots, C_n)$.

$\boldsymbol{\gamma = \sum C_i}$**:** Dieses Symbol bezeichnet als Optimalitätskriterium die *Summe der Fertigstellungszeiten (total flow time)* über allen Aufträgen J_i.

2.2 Sequenzen und Schedules

Bei den in dieser Arbeit betrachteten Shop-Scheduling-Problemen wird jeder Auftrag J_i stets genau einmal auf jeder Maschine M_j bearbeitet. Also kann eine Operation o_{ij} stets mit der Bearbeitung des Auftrags J_i auf der Maschine M_j für alle $i = 1, \ldots, n$ und $j = 1, \ldots, m$ identifiziert werden. Eine einzelne *Reihenfolge* $o_{i,j_1} \prec o_{i,j_2}$ bedeutet, daß mit der Bearbeitung der Operation o_{i,j_2} erst begonnen wird, nachdem die Bearbeitung der Operation o_{i,j_1} abgeschlossen ist. In einer Lösung für ein Shop-Scheduling-Problem wird für einen Auftrag J_i die Reihenfolge der Maschinen, auf denen J_i bearbeitet wird, als *technologische Reihenfolge* des Auftrags J_i bezeichnet. Analog dazu heißt für eine Maschine M_j die Reihenfolge der auf ihr bearbeiteten Aufträge die *organisatorische Reihenfolge* der Maschine M_j. Offensichtlich kann eine technologische bzw. organisatorische Reihenfolge auch als eine Operationen-Reihenfolge $o_{i,j_1} \prec o_{i,j_2} \prec \cdots \prec o_{i,j_m}$, bzw. $o_{i_1,j} \prec o_{i_2,j} \prec \cdots \prec o_{i_n,j}$ aufgefaßt werden.

Definition 2.2.1 Die *Technologie* ist die Menge der technologischen Reihenfolgen für alle Aufträge J_i, $i = 1, \ldots, n$. Die *Organisation* ist die Menge der organisatorischen Reihenfolgen für alle Maschinen M_j, $j = 1, \ldots, m$.

Beim Open-Shop-Problem sind sowohl die Technologie als auch die Organisation frei wählbar, während beim Job-Shop- und Flow-Shop-Problem die Technologie durch die gegebenen Vorrangbedingungen der Form „$\rightarrow$“ bereits festgelegt ist, und bei der Suche nach einer optimalen Lösung nur noch verschiedene Organisationen betrachtet werden.

Definition 2.2.2 Eine *Sequenz* eines Shop-Scheduling-Problems ist eine zulässige Kombination von Technologie und Organisation. Zulässig bedeutet hierbei, daß der durch die Kombination charakterisierte Produktionsablauf endlich ist.

Es handelt sich bei einer auf diese Weise definierten Sequenz eigentlich um eine „Multi-Sequenz“, denn es werden gleichzeitig stets alle technologischen und organisatorischen Reihenfolgen durch eine gegebene Sequenz beschrieben.

Ein *Schedule* ist die Realisierung einer Sequenz, bei der jeder Operation o_{ij} des gegebenen Shop-Scheduling-Problems ein fester Startzeitpunkt zugeordnet ist. Ein Schedule heißt *zulässig*, wenn neben dem durch die Sequenz garantierten endlichen Produktionsablauf alle weiteren problemspezifischen Bedingungen erfüllt sind. Zu einer gegebenen Sequenz kann ein Schedule anhand der vorgegebenen Bearbeitungszeiten p_{ij} der Operationen o_{ij} wie folgt bestimmt werden: Unter Berücksichtigung der durch Technologie und Organisation gegebenen Bearbeitungsreihenfolgen wird jede Operation o_{ij} so früh wie möglich ausgeführt. Die Laufzeit eines entsprechenden Algorithmus zur Berechnung der Start- und Fertigstellungszeiten der Operationen

wird später bei der Beschreibung der Matrix der Fertigstellungszeiten (Matrix C) im Blockmatrizenmodell auf Seite 23 behandelt. Ein Schedule, der auf die beschriebene Weise einer Sequenz eindeutig zugeordnet ist, heißt *semiaktiver Schedule*. Es liegt also ein semiaktiver Schedule vor, wenn keine Bearbeitung einer Operation früher beendet werden kann, ohne dabei eine technologische oder organisatorische Reihenfolge zu verändern oder eine der weiteren problemspezifischen Bedingungen zu verletzen.

In dieser Arbeit werden ausschließlich reguläre Zielfunktionen betrachtet. Offensichtlich genügt es, zur Minimierung solcher Zielfunktionen den untersuchten Lösungsbereich auf semiaktive Schedules zu beschränken (siehe FRENCH [35]).

Existenz optimaler Schedules

Die technologische Reihenfolge eines Auftrags J_i kann als Permutation der Indizes $j = 1, \ldots, m$ der Maschinen M_j aufgefaßt werden. Daher gibt es $m!$ mögliche technologische Reihenfolgen für einen Auftrag J_i. Dementsprechend kann eine gesamte Technologie für alle n Aufträge auf $(m!)^n$ verschiedene Weisen gebildet werden. Analog dazu gibt es bei einem Shop-Scheduling-Problem mit n Aufträgen und m Maschinen $(n!)^m$ mögliche Organisationen.

Für feste Werte von n und m ist die Anzahl aller möglichen Kombinationen der Technologien und Organisationen offensichtlich endlich. Da die Menge der Sequenzen für ein Problem mit n Aufträgen und m Maschinen gleichzeitig die Menge der zulässigen Kombinationen von Technologien und Organisationen darstellt, ist auch die Anzahl der Sequenzen endlich. Die oben beschriebene Zuordnung eines semiaktiven Schedules zu einer Sequenz ist eindeutig, daher folgt die Endlichkeit auch für die Anzahl der zu betrachtenden semiaktiven Schedules. Diese Endlichkeit sichert für ein Shop-Scheduling-Problem die Existenz eines Schedules, der bezüglich der gegebenen Zielfunktion optimal ist.

2.3 Komplexitätsergebnisse

Die Werkzeuge der Komplexitätstheorie sind wichtige Hilfsmittel zur Einschätzung der Schwierigkeit von kombinatorischen Optimierungsproblemen. Diese Theorie findet seinen Ursprung in Arbeiten von COOK [27] und KARP [51] Anfang der siebziger Jahre.

Eine ausführliche Übersicht zum Themenbereich der Komplexität im Zusammenhang mit rechnergestützten Lösungen von Entscheidungs- und Optimierungsproblemen ist dem Buch von GAREY und JOHNSON [36] zu entnehmen. In der vorliegenden Arbeit werden Begriffe wie z. B. *polynomialer Algorithmus*, *$\mathcal{NP}$-Vollständigkeit*, *polynomiale Reduktion* und *polynomiale Äquivalenz* als bekannt vorausgesetzt und

meist ohne weitere Erklärung benutzt. An dieser Stelle sei darauf hingewiesen, daß mit $\mathcal{P}$ bzw. $\mathcal{NP}$ die Klasse der Entscheidungsprobleme bezeichnet wird, zu deren Lösung deterministische bzw. nichtdeterministische Algorithmen mit polynomialer Laufzeit bekannt sind. Weiterhin heißt ein Optimierungsproblem *$\mathcal{NP}$-schwer ($\mathcal{NP}$-hard)*, wenn das zugehörigen Entscheidungsproblem $\mathcal{NP}$-vollständig ist, und damit im Falle der bisher nicht bestätigten Hypothese $\mathcal{P} \neq \mathcal{NP}$ für ein solches Problem kein deterministischer Algorithmus mit polynomialer Laufzeit existiert.

Die Terminologie für die Komplexität von Enumerationsproblemen ist nicht so weit verbreitet wie die von Entscheidungs- und Optimierungsproblemen. Daher wird auf diese Terminologie näher in Kapitel 5 bei der Betrachtung der Komplexität der Plan-Enumeration eingegangen.

Ziel bei der Lösung eines Shop-Scheduling-Problems ist das Finden einer optimalen und zulässigen Kombination aus Technologie und Organisation. Eine Einstufung eines solchen Problems in die Klasse $\mathcal{P}$ oder in die Klasse der $\mathcal{NP}$-schweren Probleme stellt ein Komplexitätsergebnis dar. Bei Brucker und Knust [20] ist eine ausführliche Übersicht bekannter Komplexitätsergebnisse für verschiedene Schedulingprobleme zu finden. Diese Zusammenstellung wird regelmäßig aktualisiert und ist im World-Wide-Web unter

`http://www.mathematik.uni-osnabrueck.de/research/OR/class/`

abrufbar.

Viele Schedulingprobleme lassen sich mittels elementarer polynomialer Reduktionen (siehe z. B. [18]) bezüglich ihrer Komplexität paarweise vergleichen. Bei den Definitionen im Anschluß werden die Begriffe „schwerere“ und „einfachere Probleme“ stets gemäß dieser Reduktionen aufgefaßt. Ein Problem aus der Klasse $\mathcal{P}$ heißt *maximal polynomial lösbar*, wenn alle schwereren Probleme $\mathcal{NP}$-schwer oder offen sind. Ein Problem heißt *minimal $\mathcal{NP}$-schwer*, wenn es $\mathcal{NP}$-schwer ist und alle einfacheren Probleme aus $\mathcal{P}$ oder offen sind. Ein Problem heißt *minimal offen*, wenn sein Komplexitätsstatus unbekannt ist, aber alle einfacheren Probleme aus $\mathcal{P}$ sind. Schließlich wird ein Problem *maximal offen* genannt, wenn sein Komplexitätsstatus unbekannt ist, aber alle schwereren Probleme bereits $\mathcal{NP}$-schwer sind. Offensichtlich genügt die Aufzählung aller bekannten Probleme aus diesen vier Bereichen zur vollständigen Beschreibung des Standes der Forschung bei der komplexitätstheoretischen Einstufung von Schedulingproblemen. Daher enthält die Übersicht [20] ausschließlich Komplexitätsergebnisse für Schedulingprobleme aus diesen Problemklassen.

Im Rahmen eines Projekts zur Entwicklung des Programmpakets LiSA (siehe Bräsel *et al.* [9]) wurde auf der Basis der Daten aus [20] eine Datenbank im BibTeX-Format erstellt. Die Datensätze dieser Datenbank umfassen die Literaturquellen, in denen die Komplexität (entweder maximal polynomial lösbar oder minimal $\mathcal{NP}$-schwer) der Schedulingprobleme nachgewiesen ist. Mit Hilfe des Pro-

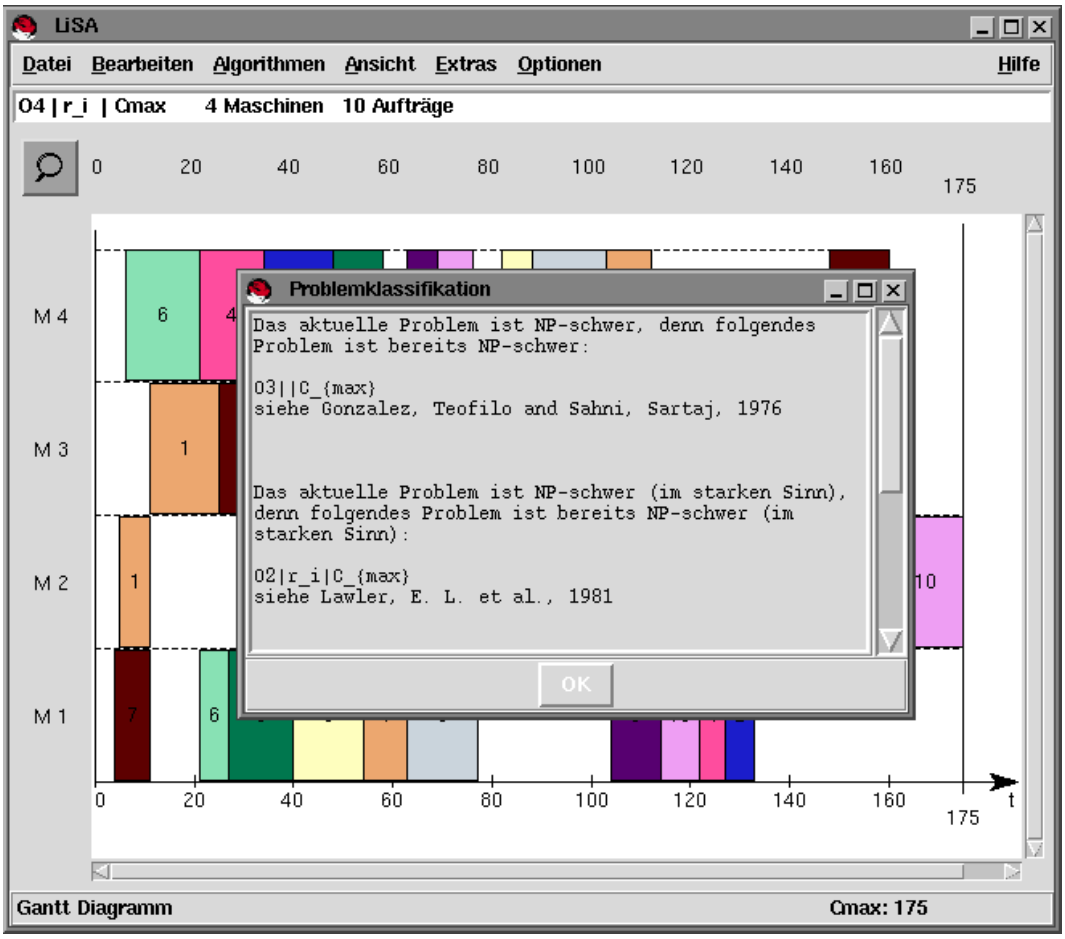

Abbildung 2.1: Die Klassifikation von Schedulingproblemen in LiSA.

grammpakets LiSA ist unter anderem eine bequeme und interaktive Benutzung dieser BibTeX-Datenbank möglich.

Beispielsweise ist in Abbildung 2.1 die Programmausgabe bei der Problemklassifikation für das Open-Shop-Problem $O4|r_i|C_{\max}$ dargestellt. Ein eigenes LiSA-Fenster enthält die Literaturquellen, die die Zugehörigkeit dieses Open-Shop-Problems zur Klasse der $\mathcal{NP}$-schweren Probleme zeigen. Die eindeutigen Identifikationsschlüssel der entsprechenden BibTeX-Datensätze (vgl. Abbildung 2.2) setzen sich in der Regel aus der entsprechenden „Mathematical Reviews Number“ oder „Zentralblatt-Nummer“ zusammen. Im `ANNOTE`-Feld wird jeweils codiert, welche Schedulingprobleme in der gegebenen Literaturquelle vorkommen, und zu welchen Komplexitätsklassen diese Probleme zugeordnet werden können.

```
@article {MR55:2108,
   AUTHOR = {Gonzalez, Teofilo and Sahni, Sartaj},
    TITLE = {Open shop scheduling to minimize finish time},
  JOURNAL = {J. Assoc. Comput. Mach.},
   VOLUME = 23,
     YEAR = 1976,
   NUMBER = 4,
    PAGES = {665--679},
   ANNOTE = {$O2||C_{\max}$ is in $P$;\\ $O3||C_{\max}$ is
                  $\NP$-hard.}
}

@article {MR82m:90091,
   AUTHOR = {Lawler, E. L. and Lenstra, J. K. and Rinnooy Kan, A. H. G.},
    TITLE = {Minimizing maximum lateness in a two-machine open shop},
  JOURNAL = {Math. Oper. Res.},
   VOLUME = 6,
     YEAR = 1981,
   NUMBER = 1,
    PAGES = {153--158},
     ISSN = {0364-765X},
   ANNOTE = {$O2|p_{ij}=1,prec,r_i|L_{\max}$ is in $P$;\\
                  $O2|r_i|C_{\max}$ is $*\NP$-hard;\\ $O2||L_{\max}$
                  is $*\NP$-hard;\\ $O2|pmtn|\sum{U_i}$ is
                  $\NP$-hard.}
}
```

Abbildung 2.2: Zwei Datensätze der BibTeX-Datenbank von LiSA.

Kapitel 3

Konzepte der Modellierung

Um eine strukturelle Vorstellung der untersuchten Schedulingprobleme gewinnen zu können, sind geeignete Modellierungen von großem Nutzen. In diesem Kapitel werden einige Konzepte und Ergebnisse behandelt, mit denen die Strukturuntersuchungen für Shop-Scheduling-Probleme in sinnvoller Weise realisiert werden können.

Viele Lösungsverfahren basieren auf der Veranschaulichung der Probleme durch geeignete Graphen. In der vorliegenden Arbeit wird auf die Terminologie der Graphentheorie im Buch von HARARY [44] zurückgegriffen, solange nicht abweichende oder zusätzliche Bezeichnungen definiert sind. Es werden stets Graphen $G = (V, E)$ ohne Schlingen und Mehrfachkanten betrachtet.[1]

Während des Optimierungsprozesses bei einem Shop-Scheduling-Problem sind stets für bestimmte Paare von Operationen Vorrangbedingungen bzw. Reihenfolgen festgelegt. Zur Illustration der verschiedenen Lösungsstrategien werden sogenannte disjunktive Graphen benutzt. Ein *disjunktiver Graph* $G^* = (V, C \cup D)$ zu gegebenen Vorrangbedingungen eines Shop-Scheduling-Problems ist anhand der Mengen V, C und D definiert:

- $V = \{\, o_{ij} \mid 1 \leq i \leq n,\ 1 \leq j \leq m \,\}$

  Die Knotenmenge besteht aus allen Operationen o_{ij}. Jeder Knoten o_{ij} besitzt als Gewicht die zugehörige Bearbeitungszeit p_{ij}.

- Die Menge C der *konjunktiven (gerichteten) Kanten* mit

  $$C = \Big\{\ (o_{ij}, o_{kl}) \ \Big|\ o_{ij}, o_{kl} \in V, \big((i = k) \vee (j = l)\big) \wedge \big(o_{ij} \rightarrow o_{kl}\big)\ \Big\}.$$

  Die konjunktiven Kanten repräsentieren die gegebenen Vorrangbedingungen $(o_{ij} \rightarrow o_{kl})$ zwischen je zwei Operationen (o_{ij} und o_{kl}), die zu einem Auftrag J_i bzw. zu einer Maschine M_j gehören.

[1] Eine *Schlinge* eines Graphen $G = (V, E)$ ist eine Kante $\{v, w\} \in E$ mit $v, w \in V$ und $v = w$. Wenn in G mehr als eine Kante aus E zwei Knoten $v, w \in V$ verbindet, werden diese Kanten $\{v, w\}$ als *Mehrfachkanten* von G bezeichnet.

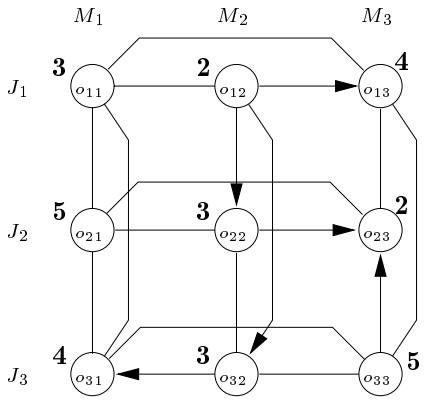

Abbildung 3.1: Ein disjunktiver Graph mit 9 Operationen.

- Die Menge D der *disjunktiven (ungerichteten) Kanten* mit

$$D = \Big\{ \{o_{ij}, o_{kl}\} \;\Big|\; o_{ij}, o_{kl} \in V, \\ \big((i = k) \vee (j = l)\big) \;\wedge\; \big((o_{ij}, o_{kl}) \notin C \;\wedge\; (o_{kl}, o_{ij}) \notin C\big) \Big\}.$$

  Die disjunktiven Kanten bestehen zwischen Operationen des gleichen Auftrags J_i oder der gleichen Maschine M_j, zwischen denen keine konjunktive Kante existiert.

Diese auf Roy und Sussmann [84] zurückgehenden Graphen besitzen gerichtete sowie ungerichtete Kanten und eignen sich gut zur schrittweisen Konstruktion zulässiger Lösungen für Shop-Scheduling-Probleme. Zum Beispiel wird in Abbildung 3.1 ein General-Shop-Problem mit 3 Aufträgen und 3 Maschinen und den Vorrangbedingungen $o_{12} \to o_{13},\ o_{12} \to o_{22},\ o_{12} \to o_{32},\ o_{22} \to o_{23},\ o_{32} \to o_{31},\ o_{33} \to o_{23}$ mit Hilfe eines disjunktiven Graphen modelliert.

Zur sukzessiven Bestimmung einer vollständigen und zulässigen Zuordnung zwischen den Maschinen M_j und den Aufträgen J_i müssen den Kanten $d \in D$ in $G^* = (V, C \cup D)$ Orientierungen zugewiesen werden, so daß keine gerichteten Kreise entstehen. Disjunktive Kanten werden in konjunktive umgewandelt, d. h. für je zwei Operationen o_{ij}, o_{il} oder o_{ij}, o_{kj}, die entweder zu einem Auftrag J_i oder zu einer Maschine M_j gehören, wird auf diese Weise eine Reihenfolge („$\prec$“) festgelegt, da solche Operationen nicht parallel bearbeitet werden können.

Am Ende dieses Prozesses existieren ausschließlich gerichtete Kanten, d. h. es gilt $D = \varnothing$. Der dadurch entstandene azyklische Digraph[2] beschreibt eine Sequenz,

[2]Ein *azyklischer Digraph* ist ein Digraph, der keinen gerichteten Kreis enthält. In Harary [44] wird ein solcher Digraph als *kreisloser Digraph* bezeichnet.

d. h. eine zulässige Kombination aus Technologie und Organisation. Die Klasse der azyklischen Digraphen, die man auf diese Weise mit Sequenzen assoziiert, werden im folgenden Abschnitt gesondert eingeführt, da sie in dieser Arbeit im Rahmen der Strukturuntersuchungen eine zentrale Rolle spielen.

3.1 Ablaufgraphen

Wenn Technologie und Organisation eines Shop-Scheduling-Problems vollständig vorgegeben sind, ist zwischen je zwei Operationen o_{ij} und o_{kl}, die zu einem Auftrag bzw. zu einer Maschine gehören, entweder $o_{ij} \prec o_{kl}$ oder $o_{kl} \prec o_{ij}$ als Reihenfolge festgelegt. Diese Reihenfolgen sind entweder durch direkte Vorgänger-Nachfolger-Beziehungen oder transitiv bestimmt. So ergeben sich z. B. durch die Reihenfolge $o_{i,j_1} \prec o_{i,j_2} \prec \ldots \prec o_{i,j_m}$ für den Auftrag J_i gleichzeitig transitiv die Reihenfolgen $o_{i,j_1} \prec o_{i,j_m},\ o_{i,j_2} \prec o_{i,j_m},\ \ldots,\ o_{i,j_{m-2}} \prec o_{i,j_m}$.

Definition 3.1.1 Es sei A eine Sequenz. Der azyklische Digraph $G(A) = (V, E)$ mit $V = \{\, o_{ij} \mid 1 \leq i \leq n,\ 1 \leq j \leq m \,\}$, $E = E_{TR} \cup E_{OR}$ und

$$\begin{aligned} E_{TR} &= \{\, (o_{i,j_1}, o_{i,j_2}) \mid o_{i,j_1}, o_{i,j_2} \in V,\ o_{i,j_1} \prec o_{i,j_2} \,\}, \\ E_{OR} &= \{\, (o_{i_1,j}, o_{i_2,j}) \mid o_{i_1,j}, o_{i_2,j} \in V,\ o_{i_1,j} \prec o_{i_2,j} \,\} \end{aligned}$$

heißt *Ablaufgraph (sequence graph)* der Sequenz A. Dabei repräsentieren E_{TR} und E_{OR} die Technologie und Organisation von A.

Dieser Digraph $G(A)$ entspricht einem disjunktiven Graphen, dessen disjunktive Kanten bereits sämtlich auf die im vorangegangenen Abschnitt beschriebene Weise in konjunktive Kanten umgewandelt wurden. In Abbildung 3.2 wird exemplarisch ein Ablaufgraph gezeigt, der aus einer (vollständigen) azyklischen Orientierung des disjunktiven Graphen aus Abbildung 3.1 hervorgeht.

Im Bereich der rein graphentheoretischen Terminologie ist diese Klasse von Digraphen in folgendem Zusammenhang bekannt: Das *kartesisches Produkt* $G = H_1 \times H_2$ zweier Graphen $H_1 = (V_1, E_1)$ und $H_2 = (V_2, E_2)$ ist der Graph $G = (V, E)$ mit $V = V_1 \times V_2$, bei dem zwei Knoten $(v_1, v_2), (w_1, w_2) \in V$ mit $v_i, w_i \in V_i$ genau dann benachbart sind, wenn $v_1 = w_1$ und $(v_2, w_2) \in E_2$ oder $v_2 = w_2$ und $(v_1, w_1) \in E_1$ gilt. Ein *Hamming-Graph* $K_n \times K_m$ ist das kartesische Produkt aus zwei vollständigen Graphen K_n und K_m. Es ist leicht zu sehen, daß der Ablaufgraph $G(A)$ einer Sequenz A für ein Shop-Scheduling-Problem mit n Aufträgen und m Maschinen einer azyklischen Orientierung des indizierten[3] Hamming-Graphen

[3]Ein Graph $G = (V, E)$ heißt *indizierter Graph*, wenn seine Knoten mit festen Indizes identifiziert und durch diese unterschieden werden.

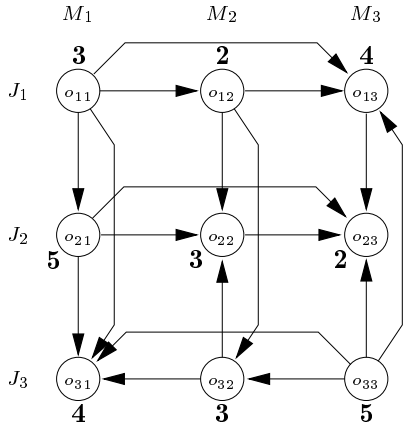

Abbildung 3.2: Ein 3×3–Ablaufgraph.

$K_n \times K_m$ entspricht, dessen Knoten mit den Operationen o_{ij} mit $i = 1, \ldots, n$ und $j = 1, \ldots, m$ identifiziert werden. Daher wird bei diesen azyklischen indizierten Digraphen im weiteren von $n \times m$–*Ablaufgraphen* gesprochen.

Eine Sequenz A läßt sich eindeutig durch den zugehörigen $n \times m$–Ablaufgraphen $G(A)$ beschreiben. Umgekehrt stellt jede azyklische Orientierung des Hamming-Graphen $K_n \times K_m$, dessen Knoten mit den Operationen o_{ij} indiziert sind, eindeutig eine zulässige Kombination von Technologie und Organisation dar, also ist die Zuordnung von Sequenzen zu Ablaufgraphen eineindeutig.

Im Zusammenhang mit Ablaufgraphen ist die sogenannte Prozedur des topologischen Sortierens[4] von großer Bedeutung. Eine *topologische Sortierung* der Knotenmenge V eines Digraphen $G = (V, E)$ mit $|V| = p$ ist eine Abbildung $\varrho : V \to \{1, \ldots, k\}$ mit $k \leq p$, so daß für alle $v, w \in V$ mit $(v, w) \in E$ die Beziehung $\varrho(v) < \varrho(w)$ gilt. Beim topologischen Sortieren der Knoten eines Digraphen G wird versucht, eine solche Abbildung ϱ von V zu finden. Offensichtlich existiert eine topologische Sortierung ϱ von V genau dann, wenn $G = (V, E)$ azyklisch ist.

Für einen azyklischen Digraphen $G = (V, E)$ sei $k_{\min}$ der kleinste Wert, für den eine topologische Sortierung $\varrho : V \to \{1, \ldots, k_{\min}\}$ existiert. Im folgenden wird stets diejenige topologische Sortierung ϱ betrachtet, die jedem Knoten $v \in V$ den kleinsten möglichen Wert aus $\{1, \ldots, k_{\min}\}$ zuordnet. Dann ist ϱ eindeutig bestimmt, und der Wert $\varrho(v)$ heißt *Rang* des Knotens $v \in V$. Der Rang $\varrho(v)$ eines Knotens $v \in V$ entspricht der Knotenanzahl eines längsten Weges in G, der im

[4]Das *topologische Sortieren* wurde erstmals im Zusammenhang mit PERT-Netzwerken behandelt (PERT steht für „Project Evaluation Review Technique“ – siehe Lasser [59] und Kahn [50]).

Knoten v endet, also ist zum Beispiel $\varrho(w) = 1$ für alle Quellen $w \in V$.

Der folgende Satz hilft bei der Beantwortung der Frage, ob es sich bei einem gegebenen Digraphen um den Ablaufgraphen einer Sequenz handelt.

Satz 3.1.2 [12] *Es sei $G = (V, E)$ ein Digraph. Das Problem der Entscheidung, ob die Knoten von G so indiziert werden können, daß G der Ablaufgraph einer Sequenz ist, kann in der Zeit $O(|E|)$ entschieden werden.*

Beweis: Zum Digraphen $G = (V, E)$ wird zunächst der zugrunde liegende Graph[5] $[G]$ betrachtet. Für einen ungerichteten Graphen mit q Kanten kann in der Zeit $O(q)$ festgestellt werden, ob es sich dabei um einen Hamming-Graphen $K_n \times K_m$ handelt (siehe IMRICH und KLAVŽAR [47]). Falls $[G]$ tatsächlich ein Hamming-Graph des Typs $K_n \times K_m$ für natürliche Zahlen n und m ist, liefert der in [47] beschriebene Algorithmus eine entsprechende Indizierung der Knoten von $[G]$. Anschließend kann topologisches Sortieren auf V angewandt werden, um zu testen, ob die Orientierung von G azyklisch ist. Die Laufzeit des Algorithmus zur Erzeugung einer topologischen Sortierung beträgt $O(p+q)$ für einen Digraphen mit p Knoten und q Kanten (siehe z. B. SIMON [87]). Da für einen Hamming-Graphen $K_n \times K_m$ mit $n, m \geq 2$ die Anzahl seiner Knoten niemals größer als die Anzahl seiner Kanten ist, folgt insgesamt die Aussage des Satzes. □

3.2 Pläne

Für eine Vielzahl der hier untersuchten Algorithmen ist es sinnvoll, die Ablaufgraphen in komprimierter Form anhand von speziellen Matrizen darzustellen. Zu diesem Zweck wird in diesem Abschnitt eine eineindeutige Zuordnung von bestimmten lateinischen Rechtecken zu Ablaufgraphen beschrieben, auf der auch das von BRÄSEL [7] eingeführte *Blockmatrizenmodell* basiert.

Es sei $n \leq m \leq r$. Ein *lateinisches Rechteck* $\mathcal{L}_{n,m,r}$ ist eine $n \times m$–Matrix mit Einträgen aus der Belegungsmenge $S = \{1, \ldots, r\}$, wobei jeder Eintrag in jeder Zeile und Spalte höchstens einmal auftritt. Ein lateinisches Rechteck mit $n = m = r$ heißt *lateinisches Quadrat*. Viele Probleme im Zusammenhang mit lateinischen Quadraten und Rechtecken werden bei DÉNES und KEEDWELL [29, 30] behandelt. Eine aktuell erschienene Übersicht von LAYWINE und MULLEN [60] zeigt die vielfältigen Anwendungen lateinischer Rechtecke in verschiedenen Bereichen der Diskreten Mathematik. In diesen Monographien ist jedoch nicht die Anwendung lateinischer Rechtecke in der Schedulingtheorie enthalten, die im folgenden beschrieben wird.

[5] Der einem Digraphen G *zugrunde liegende Graph* $[G]$ ist der ungerichtete Graph, der durch Ersetzen der gerichteten durch ungerichtete Kanten in G entsteht.

Definition 3.2.1 Ein $n \times m$*–Plan* ist ein lateinisches Rechteck $\mathcal{L}_{n,m,r} = (l_{ij})$, in dem zu jedem Eintrag $l_{ij} > 1$ der Wert $l_{ij} - 1$ als Eintrag in der Zeile i oder Spalte j auftritt.

Die Menge aller $n \times m$–Pläne bildet eine Klasse von lateinischen Rechtecken des Typs $\mathcal{L}_{n,m,r}$. Der folgende Satz zeigt, daß die Elemente dieser Klasse von lateinischen Rechtecken den im vorangegangenen Abschnitt eingeführten Digraphen zugeordnet werden können.

Satz 3.2.2 *Jedem $n \times m$–Ablaufgraphen, dessen Knoten mit den Operationen o_{ij} eines Shop-Scheduling-Problems identifiziert werden, kann eineindeutig ein $n \times m$–Plan zugeordnet werden.*

Beweis: Es sei $G = (V, E)$ ein $n \times m$–Ablaufgraph, dessen Knoten mit den Operationen o_{ij}, $i = 1, \ldots, n$ und $j = 1, \ldots, m$ eines Shop-Scheduling-Problems mit n Aufträgen und m Maschinen identifiziert werden. Weiter sei $A = (a_{ij})$ die $n \times m$–Matrix, die aus den Rängen ϱ der Knoten von G besteht, also $a_{ij} = \varrho(o_{ij})$ für alle i, j. Für je zwei Operationen $o_{i,j_1}, o_{i,j_2} \in V$ eines Auftrags J_i gilt $(o_{i,j_1}, o_{i,j_2}) \in E$ oder $(o_{i,j_2}, o_{i,j_1}) \in E$, damit ist $a_{i,j_1} \neq a_{i,j_2}$ für alle $j_1 \neq j_2$. Analoges gilt für je zwei Operationen, die zu einer Maschine M_j gehören, also ist A ein lateinisches Rechteck. Weil G nur gerichtete Kanten zwischen Operationen enthält, die zum gleichen Auftrag oder zur gleichen Maschine gehören, erfüllt A die in Definition 3.2.1 beschriebene zusätzliche Bedingung für die Einträge eines Plans.

Sei umgekehrt ein $n \times m$–Plan $A = (a_{ij})$ gegeben. Zu A wird der Digraph $G = (V, E)$ mit $V = \{o_{ij} | 1 \leq i \leq n, 1 \leq j \leq m\}$ und $E = E_{TR} \cup E_{OR}$ definiert, wobei

$$\begin{aligned} E_{TR} &= \{ (o_{i,j_1}, o_{i,j_2}) \mid o_{i,j_1}, o_{i,j_2} \in V,\ a_{i,j_1} < a_{i,j_2} \}, \\ E_{OR} &= \{ (o_{i_1,j}, o_{i_2,j}) \mid o_{i_1,j}, o_{i_2,j} \in V,\ a_{i_1,j} < a_{i_2,j} \} \end{aligned}$$

ist. Die Beziehungen der Form $a_{ij_1} < a_{ij_2}$ induzieren Reihenfolgen $o_{ij_1} \prec o_{ij_2}$, in denen die Operationen o_{ij_1} und o_{ij_2} bearbeitet werden. Die Gesamtheit dieser Beziehungen repräsentiert eine Technologie. Analog dazu ergibt sich die Organisation. Die Existenz eines gerichteten Weges in G von einem Knoten o_{ij} zu einem Knoten o_{kl} setzt die Bedingung $a_{ij} < a_{kl}$ voraus. Wegen $a_{ij} \neq a_{kl}$ für $i = j \ \wedge \ k = l$ ist G azyklisch. Insgesamt folgt also, daß G ein $n \times m$–Ablaufgraph ist. □

Dieser Satz zeigt, daß jeder Plan mit genau einem Ablaufgraphen korrespondiert. In Abbildung 3.4 (Seite 22) ist beispielsweise ein 3×3–Plan und der zugehörige 3×3–Ablaufgraph zu sehen. Aufgrund dieser Korrespondenz können Pläne auch wie folgt definiert werden (vgl. Definition 3.2.1).

Definition 3.2.3 Ein $n \times m$*–Plan* ist eine $n \times m$–Matrix $A = (a_{ij})$, die aus den Rängen der Knoten o_{ij} $(i = 1, \ldots, n;\ j = 1, \ldots, m)$ eines $n \times m$–Ablaufgraphen besteht, also $a_{ij} = \varrho(o_{ij})$ für alle i, j.

Im folgenden wird bei einem Eintrag a_{ij} eines Plans A häufig auch vom *Rang* $\varrho(o_{ij})$ der Operation o_{ij} gesprochen. Ein Ablaufgraph entspricht einer zulässigen Kombination von Technologie und Organisation und damit der in Abschnitt 2.2 definierten Sequenz eines Shop-Scheduling-Problems. Wegen Satz 3.2.2 können solche Sequenzen ebenfalls anhand von Plänen repräsentiert werden. Die Ränge der Knoten des zugehörigen Ablaufgraphen werden durch topologisches Sortieren ermittelt. In diesem Zusammenhang erhält man folgende komplexitätstheoretische Aussage.

Satz 3.2.4 *Für $n \leq m$ kann der Plan zu einem gegebenen $n \times m$–Ablaufgraphen in in der Zeit $O(m^3)$ berechnet werden.*

Beweis: Der Rang eines Knotens v in einem azyklischen Digraphen $G = (V, E)$ ist die Anzahl der Knoten eines in v endenden längsten Weges von G. Die Bestimmung der Ränge der Knoten in G basiert auf einer topologischen Sortierung ϱ von V. Das topologische Sortieren der Knoten eines azyklischen Digraphen mit p Knoten und q Kanten benötigt $O(p + q)$ Zeit (vgl. Satz 3.1.2). Für einen $n \times m$–Ablaufgraphen $G = (V, E)$ gilt $|V| = nm$ und $|E| = nm(m-1)/2 + mn(n-1)/2$, daher können für $n \leq m$ die Ränge seiner Knoten und damit sein zugeordneter Plan mit dem Zeitaufwand $O(m^3)$ bestimmt werden. □

Definition 3.2.5 Ein *reduzierter Ablaufgraph* ist ein Ablaufgraph ohne transitive Kanten.

Folgerung 3.2.6 *Zu einem gegebenen reduzierten $n \times m$–Ablaufgraphen kann der zugehörige Plan in der Zeit $O(nm)$ berechnet werden.*

Beweis: Diese Aussage ergibt sofort sich aus Satz 3.2.4, denn ein reduzierter $n \times m$–Ablaufgraph besitzt maximal $n(m-1) + m(n-1)$ Kanten. □

Oft wird ein zu einer Sequenz zugehöriger Schedule, der zusätzlich zu den gegebenen Reihenfolgen der Sequenz die Informationen über die Fertigstellungszeiten C_i der Aufträge J_i enthält, anhand des sogenannten *Gantt-Diagramms* dargestellt (vgl. Abbildung 3.3). Erstmals werden derartige Diagramme in Clark [24] und Porter [75] erwähnt. Gantt-Diagramme können je nach der Bedeutung ihrer vertikalen Achse entweder *auftrags-* oder *maschinenorientiert* sein. Das in Abbildung 3.3 dargestellte Gantt-Diagramm ist maschinenorientiert. Die horizontale Achse ist die Zeitachse; an ihr können die Start- und Fertigstellungszeiten jeder Operation o_{ij} abgelesen werden.

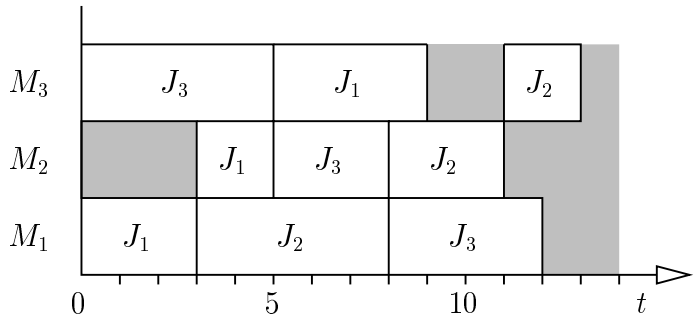

Abbildung 3.3: Das Gantt-Diagramm eines semiaktiven Schedules.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 4 & 3 & 1 \end{pmatrix}$$

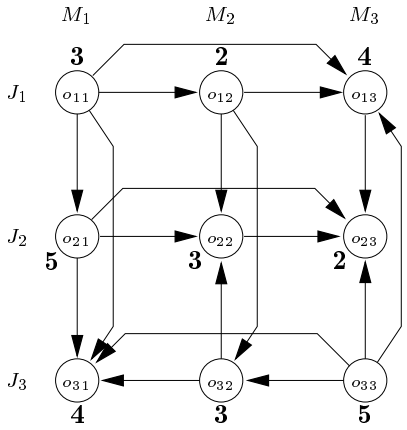

Abbildung 3.4: Ein Plan A und sein zugeordneter Ablaufgraph $G(A)$.

Eine alternative Methode für die Modellierung von Lösungen bzw. Schedules von Shop-Scheduling-Probleme stellt das auf Bräsel [7] zurückgehende *Blockmatrizenmodells* dar. Es sei A ein beliebiger $n \times m$–Plan und $G(A)$ der zugeordnete $n \times m$–Ablaufgraph (siehe Abbildung 3.4). Neben dem Plan A gehören im Blockmatrizenmodell noch vier weitere Typen von $n \times m$–Matrizen zur Beschreibung eines gegebenen Shop-Scheduling-Problems und einer zugehörigen zulässigen Lösung:

$TR = (tr_{ij})$: Die *Technologie-Matrix TR* besteht aus Zeilen, die Permutationen der Zahlen $1, \ldots, m$ sind. Diese Permutationen geben die technologischen Reihenfolgen wieder: Ein Eintrag tr_{ij} bedeutet die Position der Maschine M_j in der technologischen Reihenfolge des Auftrags J_i.

$OR = (or_{ij})$: Die *Organisations-Matrix OR* besteht aus Spalten, die jeweils Permutationen der Zahlen $1, \ldots, n$ sind und damit die organisatorischen Reihen-

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 4 & 3 & 1 \end{pmatrix} \quad TR = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \quad OR = \begin{pmatrix} 1 & 1 & 2 \\ 2 & 3 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

$$P = \begin{pmatrix} 3 & 2 & 4 \\ 5 & 3 & 2 \\ 4 & 3 & 5 \end{pmatrix} \quad C = \begin{pmatrix} 3 & 5 & 9 \\ 8 & 11 & 13 \\ 12 & 8 & 5 \end{pmatrix}$$

Abbildung 3.5: Beispiel-Matrizen im Blockmatrizenmodell.

folgen angeben: Ein Eintrag or_{ij} beschreibt die Position des Auftrags J_i in der organisatorischen Reihenfolge auf der Maschine M_j.

$P = (p_{ij})$: Für alle i, j gibt der Eintrag p_{ij} in der *Bearbeitungszeit-Matrix* P für die Operationen o_{ij} die zugehörige Bearbeitungszeit p_{ij} an.

$C = (c_{ij})$: Der Eintrag c_{ij} in der Matrix C entspricht für alle i, j der *Fertigstellungszeit c_{ij} der Operation o_{ij}*. Die Einträge dieser Matrix C lassen sich aus dem zugrunde liegenden Plan A zusammen mit der Bearbeitungsmatrix P in der Zeit $O(nm)$ bestimmen, wenn der Plan A günstig abgespeichert ist (siehe Bräsel [8]). Bei der Matrixdarstellung $C = (c_{ij})$ eines Schedules kann die Gesamtbearbeitungszeit $C_{\max}$ des zugehörigen Plans A mit Hilfe der Gleichung

$$C_{\max} = \max_{\substack{1 \le i \le n \\ 1 \le j \le m}} (c_{ij}) \tag{3.1}$$

angegeben werden. Neben dem Gantt-Diagramm handelt es sich bei der Matrix C um eine weitere Möglichkeit, einen semiaktiven Schedule darzustellen.

Man stellt fest, daß für Shop-Scheduling-Probleme im Blockmatrizenmodell die Sequenzen durch Pläne und die Schedules durch Matrizen C der Fertigstellungszeiten repräsentiert werden.

Zum Plan A und seinem 3×3–Ablaufgraphen $G(A)$ aus Abbildung 3.4 sind in Abbildung 3.5 die zugehörigen Matrizen TR und OR zu finden. Weiterhin ist die Matrix P mit Bearbeitungszeiten p_{ij} aufgeführt, die mit den Gewichten der Knoten o_{ij} in Abbildung 3.4 übereinstimmen. Für den sich aus A und P ergebenen semiaktiven Schedule (Matrix C in Abbildung 3.5) gilt in diesem Fall $C_{\max} = 13$. Dieser Schedule korrespondiert mit dem in Form eines Gantt-Diagramms gezeichneten Schedule aus Abbildung 3.3. Das Gantt-Diagramm stellt damit den eindeutig

bestimmten semiaktiven Schedule zum Ablaufgraphen $G(A)$ aus Abbildung 3.4 dar, dessen Knoten mit den vorgegebenen Bearbeitungszeiten p_{ij} gewichtet sind.

Kapitel 4

Plan-Anzahlen

In diesem Kapitel werden Ergebnisse über die Anzahl der Elemente in verschiedenen Plan-Klassen erläutert. Jedem Plan entspricht ein spezielles lateinisches Rechteck. Viele der hier vorgestellten Resultate stammen daher aus Arbeiten über die Anzahl lateinischer Rechtecke. Darüber hinaus werden bisher unbekannte Anzahlen für allgemeine Pläne bestimmt, die sich nicht direkt auf die Anzahlen entsprechender lateinischer Rechtecke zurückführen lassen. Zunächst werden im ersten Abschnitt diejenigen Pläne behandelt, die mit den sogenannten *klassischen* lateinischen Rechtecken korrespondieren.

4.1 Rangminimale Pläne

Ein $n \times m$–Plan ($n \leq m$) mit maximalem Eintrag m heißt *rangminimaler $n \times m$–Plan*. Jeder rangminimale $n \times m$–Plan repräsentiert offensichtlich eine optimale Sequenz für das zugehörige Open-Shop-Problem $Om|p_{ij} = 1|\,C_{\max}$ mit n Aufträgen. Dies ist ein Shop-Scheduling-Problem mit Einheitsbearbeitungszeiten. Einen Überblick über die Komplexität von Open-Shop-Problemen mit Einheitsbearbeitungszeiten ist bei Brucker *et al.* [19] und Tautenhahn [95] zu finden.

Es sei $L(n, m, r)$ mit $n \leq m \leq r$ die Anzahl der lateinischen Rechtecke $\mathcal{L}_{n,m,r}$ mit Einträgen aus $\{1, 2, \ldots, r\}$. Im Fall $r = m$ entspricht $L(n, m, m)$ der Anzahl $P(n, m)$ der rangminimalen $n \times m$–Pläne, denn die Bedingung für Pläne aus Definition 3.2.1 ist bei lateinischen Rechtecken $\mathcal{L}_{n,m,m}$ trivialer Weise erfüllt. Die Anzahl $P(n, m)$ ist in der Literatur auch als Anzahl *klassischer lateinischer Rechtecke* ($\mathcal{L}_{n,m,m} := \mathcal{L}_{n,m}$) bekannt.

Die Anzahl $P(n) := P(n, n)$ der quadratischen rangminimalen Pläne entspricht der Anzahl lateinischer Quadrate der Ordnung n. Die Bestimmung dieser Anzahl ist das ursprüngliche und bekannteste Enumerationsproblem in diesem Bereich. Die Berechnung von $P(n)$ erweist sich bereits für $n \geq 6$ als eine nicht-triviale Aufgabe.

n	$\dfrac{P(n)}{n!(n-1)!}$
1	1
2	1
3	1
4	4
5	56
6	9 408
7	16 942 080
8	535 281 401 856
9	377 597 570 964 258 816
10	7 580 721 483 160 132 811 489 280

Tabelle 4.1: Anzahlen rangminimaler quadratischer Pläne.

Im Zusammenhang mit den bis heute bekannten Werten für $n \leq 10$ (vgl. Tabelle 4.1) sind viele Arbeiten verschiedener Autoren entstanden. An dieser Stelle sei erwähnt, daß die Berechnung des bisher größten bekannten Wertes $P(10)$ von McKay und Rogoyski [68] im Jahre 1995 veröffentlicht wurde. Verschiedene Quellen für die Berechnungen der Anzahlen $P(n)$ mit $6 \leq n \leq 9$ können z.B. in [29, 68] nachgeschlagen werden.

Geschlossene Formeln

Auch die Behandlung der Werte $P(n, m)$ für $n \neq m$ tritt in der Literatur vielfach auf. Es sind jedoch keine exakte Formeln für $m \geq n \geq 5$ bekannt. Offensichtlich gilt $P(1, m) = m!$. Mit Hilfe der Rencontre-Zahlen[1] D_m läßt sich $P(2, m)$ darstellen (siehe z. B. [76]):

Satz 4.1.1 *Für alle $m \geq 2$ gilt*

$$P(2, m) = m!\, D_m \qquad mit \qquad D_m = m! \sum_{k=0}^{m} \frac{(-1)^k}{k!}.$$

[1] Die *Rencontre-Zahl* D_m ist die Anzahl der Derangements der Ordnung m. Ein *Derangement* der Ordnung m ist eine fixpunktfreie Permutation der Ordnung m, also eine Permutation, die in keiner Position mit der Identität übereinstimmt.

m	$\frac{P(2,m)}{m!}$	$\frac{P(3,m)}{m!}$	$\frac{P(4,m)}{m!}$
2	1		
3	2	2	
4	9	24	24
5	44	552	1 344
6	265	21 280	393 120
7	1 854	1 073 760	155 185 920
8	14 833	70 299 264	88 390 995 840
9	133 496	5 792 853 248	69 761 852 246 016
10	1 334 961	587 159 944 704	74 175 958 614 030 336

Tabelle 4.2: Anzahlen rangminimaler $n \times m$–Pläne für $n = 2, 3$ und 4.

Die Anzahl $P(3,m)$ ist unter verschiedenen Gesichtspunkten untersucht worden, da ihre Bestimmung die ersten Schwierigkeiten darstellt, siehe BOGART und LONGYEAR [6], JACOB [48], KERAWALA [52, 53], RIORDAN [80, 81, 82], und YAMAMOTO [104]. In [81] gibt RIORDAN eine elegante Formel für $P(3,m)$ an, die auf die Rencontre-Zahlen D_m und die Ménage-Zahlen[2] U_m zurückgeht.

Satz 4.1.2 [81] *Für alle $m \geq 3$ gilt*

$$\begin{aligned} P(3,m) &= m! \sum_{k=0}^{\lfloor m/2 \rfloor} \binom{m}{k} D_k D_{m-k} U_{m-2k} \\ &\quad mit \quad U_m = \sum_{k=0}^{m} (-1)^k \frac{2m}{2m-k} \binom{2m-k}{k} (m-k)! \\ &\quad und \quad U_0 = 1. \end{aligned}$$

Für $n = 4$ hat LIGHT in [61] ein Verfahren zur Enumeration rangminimaler $4 \times m$–Pläne entwickelt, das auf einer Reihe von Rekursionen im Zusammenhang mit bestimmten Diagrammen beruht. Mit deren Hilfe konnte er die Werte $P(4,m)$ für $m \leq 8$ bestimmen. Im Fall $m = 8$ hat LIGHT allerdings irrtümlich einen falschen Wert angegeben, wie ein Vergleich von $P(4,8)$ mit den entsprechenden Resultaten in [68, 71] zeigt. Die Autoren MULLEN und PURDY haben in [71] einige Fehler

[2] Die *Ménage-Zahl* U_m ist die Anzahl der Permutationen der Ordnung m, die jeweils in keiner Position mit der Identität und einem Zyklus der Länge m übereinstimmen.

in der Literatur zur Enumeration lateinischer Rechtecke bzw. rangminimaler Pläne aufgedeckt. Der falsche Wert $P(4,8)$ aus [61] ist ihnen allerdings verborgen geblieben, obwohl sie diesen Wert selbst korrekt auflisten und die Arbeit von LIGHT [61] zitieren.

Eine Übersicht über die Anzahl rangminimaler zwei-, drei- und vierzeiliger Pläne mit jeweils bis zu zehn Spalten gibt Tabelle 4.2. Die Werte für $P(4,m)$ wurden aus [68] übernommen, da die Werte $P(4,m)$ aus [61] nur bis $m = 7$ korrekt angegeben sind.

Die in den oben zitierten Arbeiten angewandten Methoden zur Berechnung der Werte $P(n,m)$ mit festem n sind nicht einheitlich und ergeben für $n \geq 5$ keine befriedigenden Resultate. Eine mögliche Begründung für die Unbrauchbarkeit dieser Methoden bei größeren Formaten liefert der nächste Unterabschnitt über die Erweiterung rangminimaler Pläne.

Es besteht ein Zusammenhang zwischen der Anzahl der rangminimalen Pläne und sogenannten Permanenten; dies zeigt Satz 4.1.4, der eine allgemeine Formel für $P(n,m)$ darstellt. Zur Vorbereitung benötigen wir:

Definition 4.1.3 Es sei $n \leq m$ und $S_m(n)$ die Menge aller n-Permutationen der Elemente $\{1,\ldots,m\}$. Die *Permanente* einer $n \times m$–Matrix $B = (b_{ij})$ ist durch

$$\mathrm{per}(B) = \sum_{\pi \in S_m(n)} b_{1,\pi(1)} b_{2,\pi(2)} \cdots b_{n,\pi(n)} \tag{4.1}$$

definiert.

Bei der Permanente einer Matrix B haben alle Terme im Gegensatz zur Determinante von B positives Vorzeichen. Obwohl die Determinante einer $n \times n$–Matrix effizient (also polynomial) berechnet werden kann, ist zur Berechnung ihrer Permanente unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ die Existenz eines polynomialen Algorithmus nicht zu erwarten (siehe VALIANT [99]). Eine Formel für $P(n,m)$, die auf Permanenten von $(0,1)$–Matrizen beruht, ist bei SHAO und WEI [85] zu finden. Mit Hilfe von Permutationsmatrizen[3] und dem Prinzip der Inklusion und Exklusion wurde der folgenden Satz beweisen.

Satz 4.1.4 [85] *Für alle $n, m \in \mathbb{N}$ mit $n \leq m$ gilt*

$$P(n,m) = m! \sum_{B \in \mathcal{B}_{n,m}} (-1)^{\sigma(B)} \binom{\mathrm{per}(B)}{m}, \tag{4.2}$$

wobei $\mathcal{B}_{n,m}$ die Menge aller $n \times m$–$(0,1)$–Matrizen ist und $\sigma(B)$ die Anzahl der Null-Elemente von B angibt.

[3] Eine *Permutations-Matrix* ist eine $(0,1)$–Matrix, in der jede Zeile und jede Spalte genau ein Eins-Element enthält.

Trotz der Einfachheit von (4.2) ist dieser Ausdruck für eine effiziente Methode zur expliziten Berechnung der Werte $P(n,m)$ für größere n und m nicht geeignet, da die Anzahl der Terme in (4.2) gemäß (4.1) mit n bzw. m exponentiell wächst. Im anschließenden Abschnitt wird deutlich, daß die Permanenten bestimmter $(0,1)$–Matrizen auch bei der Erweiterung rangminimaler Pläne von großer Bedeutung sind.

Erweiterung rangminimaler Pläne

Die Ansätze zur Bestimmung der Anzahlen $P(n,m)$ für $n \leq 4$ beruhen hauptsächlich auf der Abzählung der Möglichkeiten, eine weitere Zeile zu rangminimalen Plänen so hinzuzufügen, daß wiederum rangminimale Pläne des gewünschten größeren Formats entstehen. Daß die Existenz von Erweiterungen rangminimaler Pläne immer gesichert ist, zeigt der anschließende Satz.

Satz 4.1.5 [29] *Es sei $n < m$. Jeder rangminimale $n \times m$–Plan ist zu einem quadratischen rangminimalen $m \times m$–Plan erweiterbar.*

Es ist nun von Interesse, ob das Prinzip der Erweiterung rangminimaler Pläne ebenfalls zur effizienten Berechnung der Anzahl entsprechender Pläne größeren Formats angewandt werden kann. Es wird zunächst anhand eines Beispiels der Zusammenhang zwischen der zeilenweisen Erweiterung rangminimaler $n \times m$–Pläne und perfekten Matchings[4] in $(m-n)$–regulären Teilgraphen des $K_{m,m}$ sowie Permanenten assoziierter $m \times m$–$(0,1)$–Matrizen veranschaulicht.

Beispiel 4.1.6 Es sei ein rangminimaler 3×5–Plan A gegeben mit

$$A = \begin{pmatrix} 1 & 4 & 3 & 5 & 2 \\ 2 & 5 & 1 & 3 & 4 \\ 4 & 3 & 2 & 1 & 5 \end{pmatrix}. \tag{4.3}$$

Zu A wird der 2-reguläre bipartite Graph $G_A = (V,E)$ mit $V = \{v_1, \ldots, v_5\} \cup \{w_1, \ldots, w_5\}$ und

$$E = \{\{v_i, w_j\} \mid \text{Eintrag } i \text{ existiert } \textit{nicht} \text{ in Spalte } j \text{ von } A\},$$

definiert, siehe Abbildung 4.1. Offensichtlich ist die Anzahl der Möglichkeiten, A zu einem rangminimalen 4×5–Plan zu erweitern, gleich der Anzahl perfekter Matchings in G_A, denn jedes perfekte Matching von G_A entspricht einer möglichen 4. Zeile. Die Anzahl der Erweiterungsmöglichkeiten kann ebenso mittels der Permanente einer

[4] Ein *perfektes Matching* eines Graphen $G = (V,E)$ mit $|V| = 2p$ ist eine Menge von p Kanten aus E, in der keine zwei Kanten einen gemeinsamen Knoten besitzen.

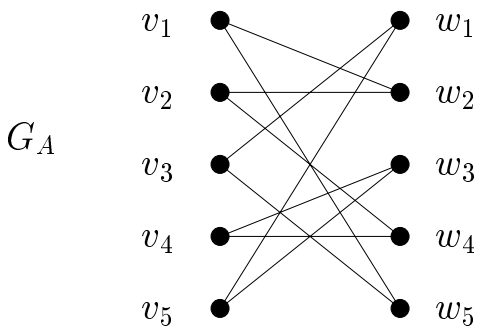

Abbildung 4.1: Der bipartite Graph G_A zu Beispiel 4.1.6.

bestimmten quadratischen $(0,1)$–Matrix ausgedrückt werden. Im quadratischen Fall ergibt sich aus (4.1) die Beziehung

$$\mathrm{per}(B) = \sum_{\pi \in S_m} b_{1,\pi(1)} b_{2,\pi(2)} \cdots b_{m,\pi(m)} \tag{4.4}$$

als Definition der Permanente einer $m \times m$–Matrix B, wobei S_m die Menge aller Permutationen der Zahlen $\{1, \ldots, m\}$ ist. Zu einem Plan A sei die $(0,1)$–Matrix $B = (b_{ij})$ durch

$$b_{ij} = \begin{cases} 1, \text{ falls Eintrag } i \textit{ nicht} \text{ in Spalte } j \text{ von } A \text{ existiert;} \\ 0, \text{ sonst;} \end{cases} \tag{4.5}$$

gegeben. Für den gemäß (4.3) gegebenen Plan A bekommt man also die 5×5–Matrix

$$B = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Offensichtlich betragen die Zeilen- und Spaltensummen von B jeweils $5-3=2$, und der Ausruck $\mathrm{per}(B)$ gibt gerade die Anzahl der Möglichkeiten an, den rangminimalen Plan A zu einem rangminimalen 4×5–Plan zu erweitern. In unserem Beispiel gilt $\mathrm{per}(B) = 2$ und die Matrizen

$$A' = \begin{pmatrix} 1 & 4 & 3 & 5 & 2 \\ 2 & 5 & 1 & 3 & 4 \\ 4 & 3 & 2 & 1 & 5 \\ 3 & 2 & 5 & 4 & 1 \end{pmatrix} \quad \text{und} \quad A'' = \begin{pmatrix} 1 & 4 & 3 & 5 & 2 \\ 2 & 5 & 1 & 3 & 4 \\ 4 & 3 & 2 & 1 & 5 \\ 5 & 1 & 4 & 2 & 3 \end{pmatrix}$$

sind die beiden Erweiterungsmöglichkeiten von A zu rangminimalen 4×5–Plänen.

In [56] hat U. KLEINAU gezeigt, daß die Enumeration der Möglichkeiten, einen gegebenen rangminimalen $n \times m$–Plan mit $n < m$ durch Hinzufügen einer Zeile zu einem rangminimalen $(n+1) \times m$–Plan zu erweitern, $\mathcal{NP}$-schwer ist:

Satz 4.1.7 [56] *Das Problem der Enumeration rangminimaler Pläne durch ein zeilenweise Erweiterung rangminimaler Pläne kleineren Formats ist $\#\mathcal{P}$-vollständig.*

Die Terminologie für die Komplexitätsklassen von Enumerationsproblemen (insbesondere die Klasse der $\#\mathcal{P}$-vollständigen Probleme) wird in Abschnitt 5.5 ausführlich erläutert (siehe dazu auch [99, 100]). Die Aussage dieses Satzes ist in [56] durch eine polynomiale Reduktion des Problems der Enumeration perfekter Matchings in regulären bipartiten Graphen auf das Enumerationsproblem für rangminimale Pläne bewiesen. Für das erste Problem hat U. KLEINAU in [56] die $\#\mathcal{P}$-Vollständigkeit gezeigt.

Mit diesem Resultat wissen wir, daß die Existenz eines polynomialen Algorithmus zur Abzählung der Erweiterungsmöglichkeiten rangminimaler Pläne unwahrscheinlich ist. Das Ergebnis verdeutlicht die Ursache des Scheiterns aller Bemühungen, anhand zeilenweiser Erweiterung von rangminimalen Plänen das Problem der Bestimmung der Anzahl rangminimaler $n \times m$–Pläne allgemein lösen zu können.

Komplettierung partieller lateinischer Rechtecke

Eine zu Satz 4.1.7 verwandte Aussage stammt aus einer Arbeit von COLBOURN [25] von 1984 und wird im folgenden vorgestellt.

Definition 4.1.8 Es sei $n \leq m$. Ein *partielles lateinisches Rechteck* ist ein lateinisches Rechteck $\mathcal{L}_{n,m}$, bei dem einige der nm Einträge aus $\{1, \ldots, m\}$ fehlen.

Satz 4.1.9 [25] *Das Entscheidungsproblem „Ist ein gegebenes partielles lateinisches Rechteck komplettierbar?“ ist $\mathcal{NP}$-vollständig.*

Bei diesem Ansatz wird von partiellen Matrizen ausgegangen, bei denen beliebige Zellen unbesetzt sein können, während in [56] ausschließlich ganze Zeilen hinzugefügt werden. Der Beweis von Satz 4.1.9 wird durch polynomiale Reduktion eines graphentheoretischen Problems geführt, das dem Problem der perfekten Matchings in [56] ähnelt: Besitzt ein gegebener tripartiter Graph eine Partition der Kantenmenge in Dreiecke?

Dieser Satz zeigt, daß im Fall $\mathcal{P} \neq \mathcal{NP}$ keine gute Charakterisierung für komplettierbare partielle lateinische Rechtecke zu erwarten ist. Das mit dem $\mathcal{NP}$-vollständigen Entscheidungsproblem in [25] assoziierte Enumerationsproblem, also das Problem der Enumeration der verschiedenen Möglichkeiten, ein partielles

lateinisches Rechteck zu komplettieren, ist offensichtlich auch $\mathcal{NP}$-schwer. Nichtsdestotrotz ist im Laufe der Zeit eine Vielzahl von notwendigen und hinreichenden Bedingungen für die Komplettierbarkeit partieller lateinischer Rechtecke entwickelt worden, siehe dazu TAUTENHAHN [95] und VAN LINT [101].

In einer 1998 erschienenen Arbeit entwickeln MCKAY und WANLESS [69] die rangminimalen $n \times m$–Pläne, die die meisten Erweiterungen zu entsprechenden $(n+1) \times m$–Plänen besitzen. Bei diesen Untersuchungen wird die Äquivalenz zum Problem der Bestimmung maximaler Permanenten von $m \times m$–$(0,1)$–Matrizen sowie zum Problem der Bestimmung $(m-n)$-regulärer Teilgraphen des $K_{m,m}$ mit maximaler Anzahl perfekter Matchings ausgenutzt. Die Bedingungen für rangminimale Pläne mit maximaler Anzahl der Erweiterungsmöglichkeiten werden meist in Form von Eigenschaften im zugrundeliegenden regulären bipartiten Graphen ausgedrückt. Das Problem der Identifizierung der rangminimalen $n \times m$–Pläne, die eine maximale Anzahl der Erweiterungsmöglichkeiten besitzen, ist in [69] zum einen für $n = 2$, m beliebig und zum anderen für $n \geq 2, k \geq 5$ und $m = kn$ komplett gelöst worden.

Asymptotische Resultate

Da sich die exakte Bestimmung der rangminimalen Pläne für größere Formate als außerordentlich schwierig gestaltet, sind asymptotische Resultate von Interesse. Ein asymptotischer Ausdruck für $P(n,m)$ ist erstmals 1946 in einer Arbeit von ERDÖS und KAPLANSKY [33] erschienen.

Satz 4.1.10 [33] *Für* $n = o((\log m)^{3/2})$ *gilt*

$$P(n,m) \sim (m!)^n \; e^{-n(n-1)/2}. \tag{4.6}$$

In [105] hat YAMAMOTO gezeigt, daß (4.6) sogar für $n = o(m^{1/3})$ gilt. Eine weitere Verbesserung dieses Resultats ist STEIN [93] im Jahr 1978 gelungen.

Satz 4.1.11 [93] *Für* $n = o(m^{1/2})$ *gilt*

$$P(n,m) \sim (m!)^n \; e^{-\binom{n}{2} - \frac{n^3}{6m}}. \tag{4.7}$$

In einer kürzlich erschienenen Arbeit verwendet SKAU [88] Ideen von VAN LINT [101], um zu zeigen, daß die Schranke $n = o(m^{1/2})$ für die Gültigkeit von (4.7) bestmöglich ist. Genauer gesagt wächst $P(n,m)$ für $n > m^{1/2+\varepsilon}$ mit $\varepsilon > 0$ langsamer als die rechte Seite von (4.7). Wie viele Resultate im Bereich der asymptotischen Bestimmung von $P(n,m)$ entsteht auch das Ergebnis in [88] durch die Abschätzung der Permanente $\mathrm{per}(B)$ der $m \times m$–$(0,1)$–Matrix B, die die Anzahl der Möglichkeiten angibt, eine $(n+1)$–te Zeile zu einem rangminimalen $n \times m$–Plan mit $n < m$ hinzuzufügen.

Umfangreiche probabilistische Untersuchungen von GODSIL und MCKAY [38] haben in diesem Zusammenhang eine wesentliche Verbesserung der vorangegangenen Resultate gebracht. Das im anschließenden Satz zusammengefaßte Ergebnis stellt zur Zeit die beste bekannte asymptotische Abschätzung für $P(n,m)$ dar.

Satz 4.1.12 [38] *Für $n = o(m^{6/7})$ gilt*

$$P(n,m) \sim (m!)^n \left(\frac{m(m-1)\cdots(m-n+1)}{m^n}\right)^m \left(1-\frac{n}{m}\right)^{-m/2} e^{-n/2}.$$

Eine interessante Verallgemeinerung des oben erwähnten Ergebnisses von YAMAMOTO [105] ist die Bestimmung der asymptotischen Anzahl der sogenannten B-lateinischen Rechtecke durch GREEN [42], die mit Hilfe der Techniken erzielt wurden, die bereits STEIN [93] benutzt hat.

Es sei B eine feste Menge mit $B \subset \mathbb{N}$. Eine $n \times m$–Matrix mit Einträgen aus $S = \{1, \ldots, m\}$ heißt *B-lateinisches Rechteck*, wenn jeder Eintrag in jeder Zeile genau einmal, und jeder Eintrag aus $B_m = B \cap S$ in jeder Spalte höchstens einmal auftritt. Das heißt, in einem B-lateinischen Rechteck ist im Gegensatz zum gewöhnlichen lateinischen Rechteck nur für einen Teil der Einträge aus S eine Wiederholung innerhalb der Spalten verboten. Für $B = S$ handelt es sich dagegen um ein gewöhnliches lateinisches Rechteck.

Der Ausdruck $L_B(n,m)$ bezeichne die Anzahl der B-lateinischen Rechtecke des Formats $n \times m$.

Satz 4.1.13 [42] *Wenn $1/|B_m| = O(n^{-\alpha})$ für ein festes α mit $\frac{1}{2} < \alpha \leq 1$ ist, dann gilt für $n = o(m^{(2\alpha-1)/3})$ die Abschätzung*

$$L_B(n,m)^{m/|B_m|} \sim (m!)^n e^{n(n-1)/2}.$$

Im Fall gewöhnlicher lateinischer Rechtecke ist $|B_m| = m$. Dieser Satz reduziert sich dann zu Satz 4.1.10 von ERDÖS und KAPLANSKY [33] mit der Schranke $n = o(m^{1/3})$ von YAMAMOTO [105].

4.2 Allgemeine lateinische Rechtecke

Im Fall der Anzahl allgemeiner lateinischer Rechtecke lassen sich nur die Werte $L(1,m,r)$ und $L(2,m,r)$ ohne große Schwierigkeiten exakt bestimmen (siehe z. B. PRANESACHAR [76]). Offensichtlich gilt $L(1,m,r) = r!/(r-m)!$.

Satz 4.2.1 *Für alle $m, r \in \mathbb{N}$ gilt*

$$L(2,m,r) = \frac{r!}{(r-m)!} \sum_{k=0}^{m} (-1)^k \binom{m}{k} \frac{(r-k)!}{(r-m)!}.$$

In [3] haben ATHREYA, PRANESACHAR und SINGHI die Technik der Möbius-Inversion[5] benutzt, um eine einheitliche Methode zur Enumeration von lateinischen Rechtecken zu entwickeln. Mit Hilfe dieser Methode sind die Anzahlen $L(n,m,r)$ für $n = 3,4$ berechnet worden. Die Überlegungen in [3] beruhen auf einer Korrespondenz zwischen allgemeinen lateinischen Rechtecken und sogenannten zulässigen Färbungen bestimmter Graphen.

Eine *zulässige λ-Färbung* eines Graphen $G = (V,E)$ ist eine Funktion $f : V \to \{1,2,\ldots,\lambda\}$ mit $f(v) \neq f(w)$ für alle $v,w \in V$ mit $\{v,w\} \in E$. Das *chromatische Polynom* $\chi(G,\lambda)$ eines ungerichteten Graphen $G = (V,E)$ ist das eindeutig bestimmte Polynom in λ, das für alle $\lambda \in \mathbb{N}$ die Anzahl der zulässigen λ–Färbungen von G angibt.[6] Ein *vollständiger bipartiter Graph* $K_{n,m}$ ist ein Graph, dessen Knotenmenge so in zwei Mengen V_1 und V_2 mit $|V_1| = n, |V_2| = m$ partitioniert werden kann, daß jeder Knoten aus V_1 mit jedem aus V_2 benachbart ist und sonst keine Nachbarschaften bestehen. Der *Kantengraph (line graph)* eines Graphen $G = (V,E)$ ist der Graph $l(G) = (V_l, E_l)$ mit $V_l = E$, bei dem je zwei Knoten aus V_l genau dann benachbart sind, wenn die entsprechenden Kanten in G einen gemeinsamen Endknoten haben.

Hilfssatz 4.2.2 *Für alle $n,m,r \in \mathbb{N}$ gilt $L(n,m,r) = \chi(l(K_{n,m}),r)$.*

Beweis: Offensichtlich entspricht der Kantengraph $l(K_{n,m})$ dem Hamming-Graphen $K_n \times K_m$. In diesem Graphen korrespondiert jeder Knoten mit einem Eintrag des lateinischen Rechtecks, so daß jeweils alle Knoten einer Zeile bzw. einer Spalte paarweise benachbart sind und bei einer zulässigen Färbung verschiedene Farben besitzen. Der Wert $\chi(K_n \times K_m, r)$ gibt die Anzahl der möglichen r-Färbungen des Hamming-Graphen Graphen $K_n \times K_m$ an. Diese Anzahl entspricht damit der Anzahl $L(n,m,r)$, da die Einträge $1,\ldots,r$ als r verschiedene Farben aufgefaßt werden können. □

Eine Spezialisierung der Resultate in [3] ist eine Formel, die das chromatische Polynom von $l(K_{n,m})$, also die Anzahl $L(n,m,r)$, als Linearkombination der chromatischen Polynome bestimmter, durch Partitionen konstruierter Graphen ausdrückt. Auf diese Weise können die Zahlen $L(3,m,r)$ und $L(4,m,r)$ berechnet werden.

Satz 4.2.3 [3] *Für alle $m,r \in \mathbb{N}$ gilt*

$$L(3,m,r) = \frac{r!m!}{((r-m)!)^3} \sum_{\alpha+\beta+\gamma=m} (-1)^\beta 2^\gamma \frac{((r-m+\alpha)!)^2}{\alpha!\gamma!} \binom{3r-3m+3\alpha+\beta+2}{\beta}.$$

[5] Die *Möbius-Inversion* ist eine effiziente Methode zur Berechnung der Summanden, die bei der Anwendung des Prinzips der Inklusion und Exklusion vorkommen (siehe ROTA [83]).

[6] Die Funktion $\chi(G,\lambda)$ ist von BIRKHOFF [4] erstmals 1912 eingeführt worden. Es ist leicht zu zeigen, daß es sich bei $\chi(G,\lambda)$ tatsächlich um ein Polynom in λ handelt (siehe z. B. [77]).

Da die Formel in [3] für $L(4,m,r)$ sehr umfangreich ist, wird auf deren Darstellung hier verzichtet. Für $r = m$ ergeben sich aus Satz 4.2.1 und Satz 4.2.3 die Werte $P(2,m)$ und $P(3,m)$ gemäß Satz 4.1.1 und 4.1.2. Tabelle 4.3 auf Seite 37 und Tabelle 4.4 auf Seite 42 enthalten die Anzahlen $L(2,m,r)$ und $L(3,m,r)$ für $m \leq 10$, jeweils als Summe über alle möglichen Werte r mit $m \leq r \leq nm$.

In [72] hat NECHVATAL, ebenfalls mit Hilfe der Technik der Möbius-Inversion, asymptotische Ergebnisse für $L(n,m,r)$ erzielt. Diese Ergebnisse können als Verallgemeinerung der Resultate von ERDÖS und KAPLANSKY [33] für $P(n,m)$ im vorangegangenen Unterabschnitt aufgefaßt werden.

4.3 Allgemeine Pläne

Es sei $P(n,m,r)$ die Anzahl der $n \times m$–Pläne mit maximalem Eintrag bzw. Rang r, wobei $m \leq r \leq nm$ ist. Aufgrund der Definition der Pläne ist es offensichtlich, daß für alle r die Beziehung $P(n,m,r) \leq L(n,m,r)$ gilt, d. h. die Anzahl lateinischer Rechtecke ist im allgemeinen nur eine grobe obere Schranke für die entsprechende Anzahl der Pläne. Es sind für festes n weniger Anzahlen $P(n,m,r)$ als $L(n,m,r)$ bekannt.

In [14] haben BRÄSEL und M. KLEINAU im Jahr 1992 eine Enumerationsmethode für die Anzahl der $n \times m$–Pläne für kleine Werte von n und m vorgestellt. In Kapitel 5 wird eine Weiterentwicklung des Algorithmus aus [14] behandelt, die zu einer effektiveren Plan-Enumeration führt.

Im Rahmen der folgenden Unterabschnitte wird eine bekannte exakte Formel für die Anzahl aller $2 \times m$–Pläne angegeben, und es werden neue Ergebnisse für die Anzahlen aller $3 \times m$– und $4 \times m$–Pläne entwickelt.

Pläne des Formats $2 \times m$

Für die Gesamtanzahl aller $n \times m$–Pläne wird $P_{n,m} := \sum_{r=m}^{nm} P(n,m,r)$ geschrieben. Ein geschlossener Ausdruck für $P_{2,m}$ erscheint erstmals bei BRÄSEL und M. KLEINAU [13]. Diese Werte geben Aufschluß über die Anzahl aller zulässigen Kombinationen von Technologien und Organisationen des Problems $Om|n = 2|C_{\max}$ bzw. die entsprechende Anzahl für das Problem $O2||C_{\max}$ mit Auftragsanzahl n. Die Beziehung $P_{2,m} = P_{n,2}$ gilt für $n = m$ offensichtlich aus Symmetriegründen.

Satz 4.3.1 [13] *Für alle $m \in \mathbb{N}$ gilt*

$$P_{2,m} = \sum_{r=m}^{2m} P(2,m,r) = m! \sum_{k=0}^{m} \frac{m!}{k!} \binom{m}{k}. \tag{4.8}$$

Beweis: In [1] haben AKERS und FRIEDMAN gezeigt, daß die Anzahl der zulässigen Organisationen zu einer gegebenen Technologie des Problems $Jm|n = 2|C_{\max}$ gleich $m + 1 + \sum_{k=2}^{m} |\pi_k|$ ist, wobei π_k die Menge der geordneten k-Tupel $(j_1, \ldots, j_k)$ von Maschinen $M_{j_1}, \ldots, M_{j_k}$ ist, die sich in derselben technologischen Reihenfolge beider Aufträge befinden, also

$$o_{1,j_1} \prec o_{1,j_2} \prec \cdots \prec o_{1,j_k} \quad \text{und} \quad o_{2,j_1} \prec o_{2,j_2} \prec \cdots \prec o_{2,j_k}.$$

Durch Summation über alle möglichen Technologien entsteht die verallgemeinerte Aussage (4.8) für das Open-Shop-Problem: Die Operationen-Reihenfolge $o_{1,1} \prec o_{1,2} \prec \cdots \prec o_{1,m}$ repräsentiere die technologische Reihenfolge von J_1. Unter den $m!$ technologischen Reihenfolgen von J_2 tritt ein festes, in natürlicher Reihenfolge geordnetes k-Tupel von Maschinen genau $m!/k!$–mal auf. Für die Wahl eines solchen k-Tupels gibt es $\binom{m}{k}$ Möglichkeiten. Der konstante Summand $m + 1$ aus der Formel von AKERS und FRIEDMAN kommt für jede der $m!$ technologischen Reihenfolgen von J_2 hinzu. Durch Multiplikation mit $m!$ als Anzahl der technologischen Reihenfolgen von J_1 ergibt sich

$$P_{2,m} = m!\left[m!(m+1) + \sum_{k=2}^{m} \frac{m!}{k!}\binom{m}{k}\right] = m! \sum_{k=0}^{m} \frac{m!}{k!}\binom{m}{k}.$$

□

In Tabelle 4.3 ist die Anzahl zweizeiliger Pläne gemäß (4.8) im Vergleich mit der Anzahl lateinischer Rechtecke $\mathcal{L}_{2,m,r}$ mit $m \le r \le 2m$ für $m = 2, \ldots, 10$ (jeweils reduziert um den Faktor $m!$) dargestellt.

Azyklische Orientierungen und chromatische Polynome

Ein $n \times m$–Plan entspricht einem $n \times m$–Ablaufgraphen und somit einer azyklischen Orientierung des Hamming-Graphen $K_n \times K_m$. Im folgenden wird gezeigt, daß nicht nur die Bestimmung der Anzahl allgemeiner lateinischer Rechtecke (Abschnitt 4.2), sondern auch die Bestimmung der Anzahl aller $n \times m$–Pläne eng mit der Bestimmung des chromatischen Polynoms des Hamming-Graphen $K_n \times K_m$ verbunden ist.

Es sei $\alpha(G)$ die Anzahl der azyklischen Orientierungen eines Graphen G. In [92] hat STANLEY erstmals $\alpha(G)$ mit dem chromatischen Polynom $\chi(G, \lambda)$ von G in Zusammenhang gebracht. Der entsprechende Satz in [92] macht sogar eine allgemeinere Aussage. Hier wird jedoch ausschließlich die folgende, für die Bestimmung der Anzahlen $P_{n,m}$ interessante Spezialisierung der Aussage in [92] bewiesen.

Satz 4.3.2 [92] *Für jeden Graphen $G = (V, E)$ gilt $\alpha(G) = (-1)^{|V|}\chi(G, -1)$.*

m	$P_{2,m} = \sum_{r=m}^{2m} \frac{P(2,m,r)}{m!}$	$\sum_{r=m}^{2m} \frac{L(2,m,r)}{m!}$
2	7	52
3	34	1 786
4	209	89 334
5	1 546	5 860 548
6	13 327	476 670 186
7	130 922	46 306 142 594
8	1 441 729	5 232 708 447 382
9	17 572 114	674 452 363 859 548
10	234 662 231	97 662 704 169 789 056

Tabelle 4.3: Anzahlen der $2 \times m$–Pläne und der lateinischen Rechtecke $\mathcal{L}_{2,m,r}$.

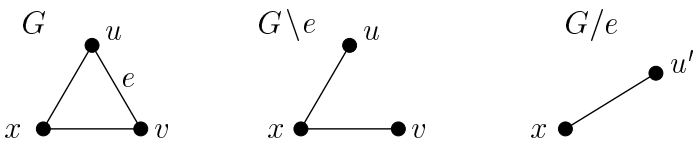

Abbildung 4.2: Zur Definition der Graphen $G\backslash e$ und G/e.

Beweis: Es ist bekannt, daß das chromatische Polynom $\chi(G,\lambda)$ eines Graphen $G = (V,E)$ eindeutig durch die drei folgenden Bedingungen bestimmt ist (siehe z. B. [77], Theorem 2.2, 2.5 und 2.6):

(i) $\chi(G_0,\lambda) = \lambda$, wobei G_0 der Graph ist, der aus einem Knoten besteht,
(ii) $\chi(G \cup H,\lambda) = \chi(G,\lambda)\chi(H,\lambda)$, wobei $G \cup H$ die Vereinigung zweier disjunkter Graphen G und H ist,
(iii) $\chi(G,\lambda) = \chi(G\backslash e,\lambda) - \chi(G/e,\lambda)$ für alle $e \in E$, wobei $G\backslash e$ bzw. G/e der Graph ist, der aus einem Graphen $G = (V,E)$ durch Löschen bzw. Kontraktion der Kante e entsteht (vgl. Abbildung 4.2).

Es reicht also zu zeigen, daß für $\alpha(G) = (-1)^{|V|}\chi(G,-1)$ die entsprechenden Bedingungen

(i') $\alpha(G_0) = 1$,
(ii') $\alpha(G \cup H) = \alpha(G)\alpha(H)$,
(iii') $\alpha(G) = \alpha(G\backslash e) + \alpha(G/e)$

gelten. Die Anzahl azyklischer Orientierungen des trivialen Graphen G_0 ist eins, und die Anzahl der azyklischen Orientierungen eines aus zwei Komponenten bestehenden Graphen ist das Produkt der entsprechenden Zahlen für die Komponenten, da die azyklischen Orientierungen unabhängig voneinander sind. Also gelten offensichtlich die Bedingungen (i') und (ii'). Im folgenden wird nun auch die Gültigkeit von (iii') gezeigt.

Es sei $\mathcal{O}$ eine azyklische Orientierung von $G\backslash e$, wobei $e = \{u, v\}$ die gelöschte Kante ist. Weiterhin sei $\mathcal{O}_1$ die Orientierung von G, die durch Hinzufügen von $u \to v$ zu $\mathcal{O}$ entsteht, und $\mathcal{O}_2$ die entsprechende Orientierung durch Hinzufügen von $v \to u$. Es ist schnell einzusehen (vgl. Abbildung 4.3, linke Hälfte), daß für jede azyklische Orientierung $\mathcal{O}$ von $G\backslash e$ entweder $\mathcal{O}_1$ oder $\mathcal{O}_2$ azyklisch ist, außer in $\alpha(G/e)$ Fällen, in denen sowohl $\mathcal{O}_1$ als auch $\mathcal{O}_2$ azyklisch sind (vgl. Abbildung 4.3, rechte Hälfte). Also gilt $\alpha(G) = \alpha(G\backslash e) + \alpha(G/e)$. □

Einen alternativen Beweis der Aussage dieses Satzes hat Vo in [102] gegeben. Der Beweis beruht auf sogenannten geordneten kantenfreien Partitionen der Knoten eines Graphen, und wird im folgenden skizziert.

Es sei $G = (V, E)$ ein Graph. Eine *Partition* von V ist eine Menge disjunkter Teilmengen von V, deren Vereinigung V ergibt. Eine *kantenfreie Partition* von V ist eine Partition in unabhängige[7] Knotenmengen. Eine *geordnete kantenfreie Partition* von G ist eine kantenfreie Partition, bei der eine Reihenfolge der unabhängigen Knotenmengen festgelegt ist. Ist π_k die Anzahl der kantenfreien Partitionen von G in k unabhängige Knotenmengen ist, so gilt offensichtlich

$$\chi(G, \lambda) = \sum_{k=1}^{|V|} \pi_k \ \lambda(\lambda - 1) \cdots (\lambda - k + 1), \tag{4.9}$$

und mit $\lambda = -1$ erhält man

$$(-1)^{|V|}\chi(G, -1) = \sum_{k=1}^{|V|} (-1)^{|V|-k} \pi_k k! \,. \tag{4.10}$$

Es sei Π_G die Menge aller geordneten kantenfreien Partitionen von V. Es gilt $|\Pi_G| = \sum_{k=1}^{|V|} \pi_k k!$. Wenn man in dieser Summe jedem Element P von Π_G das Vorzeichen $(-1)^{|V|-k}$ zuordnet, wird deutlich, daß beim Summieren nur die Fixpunkte einer vorzeichenumkehrenden Involution[8] $i : \Pi_G \to \Pi_G$ übrig bleiben, da

[7] In einem Graphen $G = (V, E)$ heißt eine Knotenmenge $V' \subseteq V$ *unabhängig*, wenn keine zwei Knoten aus V' benachbart sind.

[8] Eine *Involution* ist eine Abbildung i mit $i(i(a)) = a$ für alle Elemente a, auf denen i definiert ist.

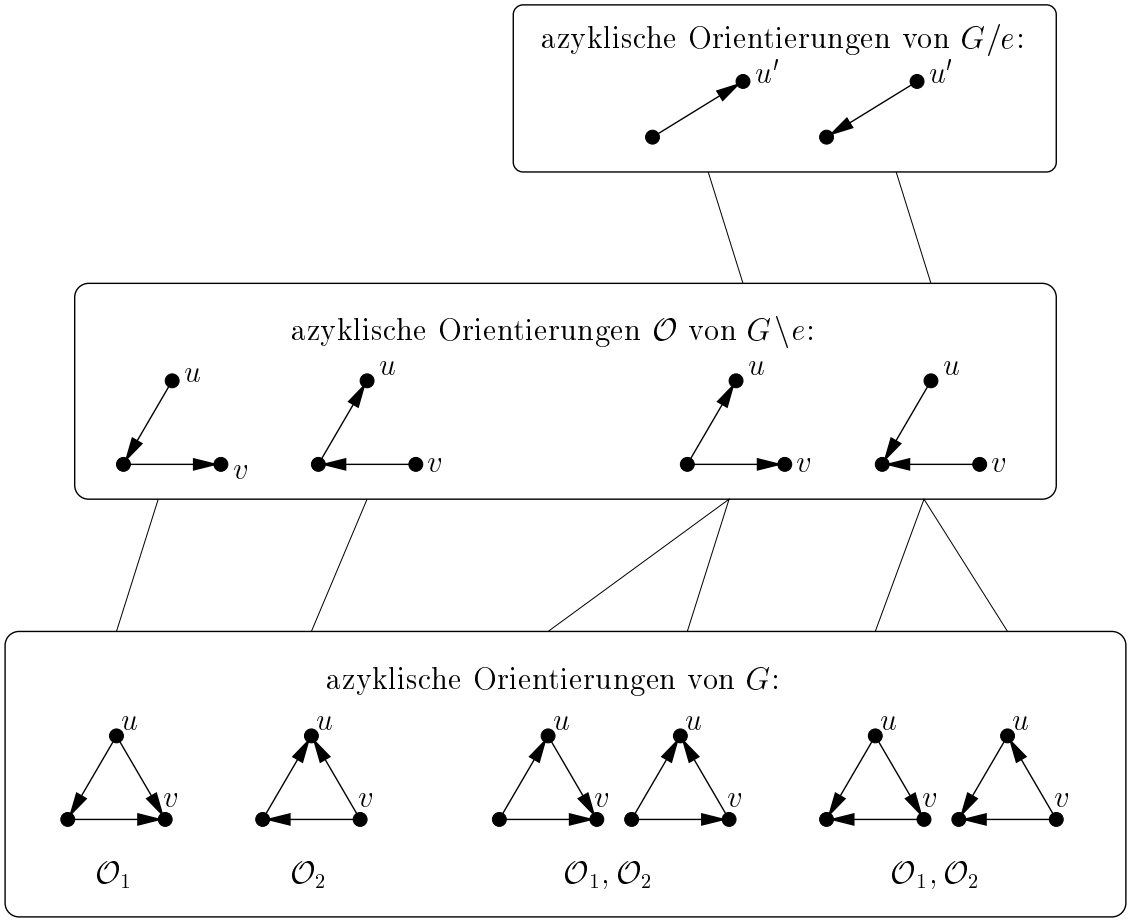

Abbildung 4.3: Zum Beweis von Satz 4.3.2.

sich alle anderen Elemente von Π_G durch Anwendung von i zu Null summieren. In [102] definiert Vo nun eine solche Involution i auf Π_G, deren Fixpunktmenge gerade die sogenannten diskreten Basis-Partitionen[9] von V sind, die wiederum den azyklischen Orientierungen von G eineindeutig zugeordnet werden können. Daher werden auf der rechten Seite von (4.10) die azyklischen Orientierungen von G gezählt, und der alternative Beweis von Satz 4.3.2 ist damit vollständig.

Da jeder $n \times m$–Plan mit einer azyklischen Orientierung des Hamming-Graphen $K_n \times K_m$ eineindeutig korrespondiert, kann man anhand von Satz 4.3.2 die Anzahl der Pläne mit Hilfe des chromatischen Polynoms der Hamming-Graphen darstellen.

Satz 4.3.3 *Für alle $n, m \in \mathbb{N}$ gilt*

$$P_{n,m} = \alpha(K_n \times K_m) = (-1)^{nm}\chi(K_n \times K_m, -1).$$

□

Wenn es eine effiziente Methode zur Berechnung des chromatischen Polynoms von Hamming-Graphen gibt, kann mit ihrer Hilfe die Anzahl aller Pläne eines gegebenen Formats $n \times m$ berechnet werden. Der Komplexitätsstatus dieses Enumerationsproblems ist bis heute unbekannt (siehe Abschnitt 5.5).

Pläne des Formats $3 \times m$ und $4 \times m$

Mit Hilfe des gerade beschriebenen Zusammenhangs und den Ergebnissen aus Abschnitt 4.2 über die Anzahl allgemeiner lateinischer Rechtecke kann neben der Bestimmung der Anzahl $P_{2,m}$ auch eine Formel für die Anzahl der dreizeiligen Pläne erstellt werden, denn das chromatische Polynom der Hamming-Graphen $K_3 \times K_m$ ist bekannt.

Satz 4.3.4 *Für festes $m \in \mathbb{N}$ gilt*

$$P_{3,m} = (-1)^m \frac{\lambda! m!}{((\lambda-m)!)^3} \sum_{\alpha+\beta+\gamma=m} (-1)^\beta \; 2^\gamma \; \frac{((\lambda-m+\alpha)!)^2}{\alpha!\gamma!} \; \binom{3\lambda-3m+3\alpha+\beta+2}{\beta}$$

mit $\lambda = -1$.

Beweis: Aufgrund von Satz 4.3.3 ist $P_{3,m}$ gleich dem Betrag des chromatischen Polynoms $\chi(K_3 \times K_m, \lambda)$ an der Stelle $\lambda = -1$. Hilfssatz 4.2.2 zeigt, daß das

[9] Die Elemente einer Knotenmenge V seien linear geordnet. Eine *diskrete Basis-Partition* ist eine geordnete kantenfreie Partition, bei der alle Teilmengen mit mehr als einem Knoten in 1-elementige Teilmengen aufgeteilt sind, wobei diese bezüglich der linearen Ordnung von V absteigend sortiert sind.

chromatische Polynom $\chi(K_3 \times K_m, \lambda)$ dem Ausdruck $L(3, m, \lambda)$ entspricht. Die Aussage des Satzes folgt dann wegen der Formel für $L(3, m, \lambda)$ aus Satz 4.2.3. □

Bemerkung 4.3.5 Wenn beim Ausdruck $P_{3,m}$ der Term $1/((\lambda-m)!)^2$ in die Summe hineingezogen wird, sieht man, daß es sich auf der rechten Seite um ein Polynom in λ vom Grad $3m$ handelt: Vor der Summe verbleibt mit $\lambda!/(\lambda - m)!$ der Anteil λ^m. In der Summe ergibt

$$\left(\frac{(\lambda - m + \alpha)!}{(\lambda - m)!}\right)^2$$

den Anteil $\lambda^{2\alpha}$ und der Binomialkoeffizient den Anteil λ^β. Wegen $m = \alpha + \beta + \gamma$ gilt $2\alpha + \beta \leq 2m$, also kommt durch die Summe der Anteil λ^{2m} zu λ^m noch hinzu. Beispielsweise lauten die beiden Polynome auf der rechten Seite für

$$\begin{array}{lll} m = 1: & -\lambda^3 + 3\lambda^2 - 2\lambda & \text{und für} \\ m = 2: & \lambda^6 - 9\lambda^5 + 34\lambda^4 - 67\lambda^3 + 67\lambda^2 - 26\lambda. & \end{array}$$

Mit $\lambda = -1$ ergibt sich $P_{3,1} = 6$ und $P_{3,2} = 204$.

Es liegt nun nahe, die gleiche Vorgehensweise auch für die Anzahl $P_{4,m}$ anzuwenden, da analog zu Satz 4.2.3 die Arbeit von Athreya, Pranesachar und Singhi [3] auch eine Formel für $L(4, m, \lambda)$ enthält. Allerdings scheint diese Formel nicht korrekt zu sein, denn schon im einfachsten Fall (für $m = 1$ und $\lambda = 4$) ergibt sich ein Wert, der nicht der Anzahl der zulässigen 4-Färbungen des Hamming-Graphen $K_4 \times K_1$ bzw. der Anzahl der Lateinischen Rechtecke $\mathcal{L}_{4,1,4}$ entspricht. Diese Anzahl $L(4, 1, 4)$ beträgt 4!. Mit der Formel aus [3] ergibt sich jedoch $L(4, 1, 4) = 137952$.

Der Beweis der Formel in [3] ist nur angedeutet. Eine Anfrage an C. R. Pranesachar (einer der Autoren von [3]) blieb bisher ohne klärenden Erfolg.

Analog zu Tabelle 4.3 auf Seite 37 für Pläne und lateinische Rechtecke vom Format $2 \times m$ enthält Tabelle 4.4 die Werte entsprechender Matrizen des Formats $3 \times m$. Tabelle 4.3 und Tabelle 4.4 veranschaulichen deutlich, daß in der betrachteten Menge der lateinischen Rechtecke nur relativ wenig Elemente die zusätzliche Bedingung eines Plans erfüllen. Die Anzahl der lateinischen Rechtecke ist also nur eine sehr unscharfe obere Schranke für die Anzahl der zugehörigen Pläne. Im folgenden Abschnitt werden nun schärfere obere und untere Schranken für die Anzahl $P_{n,m}$ hergeleitet.

m	$P_{3,m} = \sum_{r=m}^{3m} \frac{P(3,m,r)}{m!}$	$\sum_{r=m}^{3m} \frac{L(3,m,r)}{m!}$
3	3 194	9 432 636
4	155 544	32 338 932 048
5	10 736 592	185 278 786 748 496
6	989 958 592	1 602 418 389 749 579 136
7	116 976 844 224	19 524 505 523 383 344 567 936
8	17 177 847 282 048	318 946 995 329 678 929 562 127 360
9	3 061 325 835 300 608	6 730 548 553 292 744 342 990 592 919 680
10	649 679 086 266 011 904	178 253 947 720 328 843 939 901 662 766 677 760

Tabelle 4.4: Anzahlen der $3 \times m$–Pläne und der lateinischen Rechtecke $\mathcal{L}_{3,m,r}$.

4.4 Obere und untere Schranken

Die Formeln für die $3 \times m$– und $4 \times m$–Pläne sind bereits vergleichsweise umfangreich und kompliziert. Aufgrund der erwähnten Resultate im Zusammenhang mit der Komplexität der Erweiterung rangminimaler Pläne sind erst recht keine einfachen Ausdrücke für Formate $n \times m$ mit $n \geq 5$ zu erwarten. Es wird daher in diesem Abschnitt nach oberen und unteren Schranken für die Gesamtanzahl $P_{n,m}$ aller $n \times m$–Pläne gesucht.

In [55] hat M. Kleinau Abschätzungen für die Anzahl zulässiger Lösungen von Job-Shop-Problemen des Typs $Jm||C_{\max}$ mit n Aufträgen gegeben. Im folgenden werden obere und untere Schranken für die Anzahl $P_{n,m}$ aller $n \times m$–Pläne bzw. aller zulässigen Lösungen des Open-Shop-Problems $Om||C_{\max}$ mit n Aufträgen entwickelt.

Zum Auffinden oberer und unterer Schranken für die Anzahl der $n \times m$–Pläne genügt es nach Satz 4.3.3, das chromatische Polynom des Hamming-Graphen $K_n \times K_m$ nach oben und unten entlang der negativen reellen Achse abzuschätzen. In Arbeiten von Dohmen [31, 32] über Schranken für chromatische Polynome $\chi(G,k)$ werden ausschließlich Abschätzungen für positive ganzzahlige λ bzw. reelle Werte $\lambda \geq 1$ entwickelt. Die Interpretation von $\chi(G,\lambda)$ als Anzahl zulässiger λ-Färbungen des Graphen G ist nur für positive ganzzahlige λ sinnvoll. Für negative Werte von λ sind die Ergebnisse aus [31, 32] unbrauchbar und können daher hier im Zusammenhang mit azyklischen Orientierungen nicht verwandt werden. Es wird nun nach Abschätzungen des Polynoms $\chi(G,\lambda)$ gesucht, die auch für negative λ Gültigkeit besitzen.

Obere Schranke durch Gerüst-Anzahl

In [49] haben KAHALE und SCHULMAN eine obere Schranke für die Anzahl $\alpha(G)$ der azyklischen Orientierungen eines Graphen G auf der Grundlage der Gerüste[10] eines zu G verwandten Graphen G' hergeleitet. Diese Schranke stellt eine Verbesserung der vorher bekannten oberen Schranken dar, die ausschließlich auf den Knotengraden der Knoten eines Graphen basieren. Eine Verallgemeinerung dieser Ergebnisse ergibt zusätzlich Abschätzungen für das chromatische Polynom $\chi(G, \lambda)$ für negative reelle Argumente λ.

Für einen ungerichteten Graphen G sei G' der erweiterte Graph, der aus G durch Hinzufügen eines Knoten u entsteht, wobei u zu allen Knoten von G benachbart ist. Weiterhin sei $\tau(G)$ die Anzahl der Gerüste eines Graphen G. Die entwickelten oberen Schranken für $\alpha(G)$ beruhen auf einem in [49] gezeigten Zusammenhang zwischen den Anzahlen $\alpha(G)$ und $\tau(G')$.

Hilfssatz 4.4.1 [49] *Für einen beliebigen Graphen G gilt $\alpha(G) \leq \tau(G')$.*

Die *Admittanzmatrix* $Q(G)$ eines Graphen $G = (V, E)$ mit $V = \{1, \ldots, p\}$ ist die $p \times p$–Matrix $Q(G) = (q_{ij})$ mit

$$q_{ij} = \begin{cases} -1, & \text{falls die Knoten } i \text{ und } j \text{ benachbart sind,} \\ 0, & \text{falls } i \neq j \text{ und } i \text{ ist nicht zu } j \text{ benachbart,} \\ d(i), & \text{falls } i = j, \end{cases}$$

wobei $d(i)$ den Grad des Knoten i angibt.

Es sei $Q(G)_i$ die Matrix, die sich durch Streichung der i-ten Zeile und i-ten Spalte der Admittanzmatrix $Q(G)$ ergibt. Der folgende, ursprünglich auf KIRCHHOFF [54] zurückgehende Matrix-Gerüst-Satz zeigt, daß die Anzahl der Gerüste eines beliebigen Graphen mit Hilfe der Admittanzmatrix bestimmbar ist.

Satz 4.4.2 [44] *Es sei $G = (V, E)$ ein Graph mit $V = \{1, \ldots, p\}$. Dann gilt $\tau(G) = \det(Q(G)_i)$ für beliebiges i mit $1 \leq i \leq p$.*

Die *Kirchhoff-Matrix* von G ist $K(G) = Q(G) + E$, dabei bezeichnet E die Einheitsmatrix. Anhand von Hilfssatz 4.4.1 und durch Anwenden des Matrix-Gerüst-Satzes auf G' (Streichung der zum Knoten u gehörenden Zeile und Spalte) folgt unmittelbar die anschließende Aussage.

Satz 4.4.3 [49] *Es sei G ein beliebiger Graph und $K(G)$ seine Kirchhoff-Matrix. Dann gilt $\alpha(G) \leq \det(K(G))$.*

[10] Ein zyklenfreier Graph mit p Knoten und $p - 1$ Kanten heißt *Baum*. Es sei $G = (V, E)$ ein Graph. Ein Baum $T = (V_T, E_T)$ mit $V_T = V$ und $E_T \subseteq E$ heißt *aufspannender Baum* bzw. *Gerüst* von G.

Dieser Satz liefert zusammen mit einer geeigneten Abschätzung für die Kirchhoff-Matrix von Hamming-Graphen $K_n \times K_m$ eine obere Schranke für $P_{n,m}$.

Satz 4.4.4 *Für alle $n, m \in \mathbb{N}$ gilt*

$$P_{n,m} \leq \left[\frac{(n+m-1)}{e^{\left(\frac{1}{2d} - \frac{1}{2d^2} + \frac{1}{12d^3} - \frac{1}{4d^4} + O(\frac{1}{d^5})\right)}} \right]^{nm}$$

mit $d = n + m - 2$.

Beweis: Wegen Satz 4.3.3 und Satz 4.4.3 ist $P_{n,m} \leq \det(K(K_n \times K_m))$. Weiterhin gilt für jeden d-regulären Graphen $G = (V, E)$ mit $|V| = p$ die Beziehung $\det(K(G)) \leq (d+1)^p \exp(-p(\frac{1}{2d} - \frac{1}{2d^2} + \frac{1}{12d^3} - \frac{1}{4d^4} + O(\frac{1}{d^5})))$, siehe [49]. Die Aussage des Satzes folgt, da der Hamming-Graph $K_n \times K_m$ ein $(n+m-2)$–regulärer Graph mit nm Knoten ist. □

Zwei untere Schranken

In [37] wird von GODDARD *et al.* erstmals eine untere Schranke für die Anzahl azyklischer Orientierungen eines Graphen G in Abhängigkeit der Gradfolge von G bewiesen.

Satz 4.4.5 [37] *Für jeden Graphen $G = (V, E)$ mit $V = \{1, \ldots, p\}$ gilt*

$$\alpha(G) \geq \prod_{i=1}^{p} \left((d_i + 1)!\right)^{\frac{1}{d_i+1}},$$

wobei d_i den Grad des Knoten i bezeichnet.

Im Fall $G = K_n \times K_m$ kann man für die Anzahl der $n \times m$–Pläne eine untere Schranke herleiten.

Folgerung 4.4.6 *Für alle $n, m \in \mathbb{N}$ gilt*

$$P_{n,m} \geq \left((n+m-1)!\right)^{\frac{nm}{n+m-1}}. \tag{4.11}$$

Diese untere Schranke zeigt bereits das enorme Ansteigen der Anzahlen $P_{n,m}$ mit wachsenden Werten n und m. Zur Veranschaulichung ist die Funktion $f(n, m) = \left((n+m-1)!\right)^{nm/(n+m-1)}$ für das Intervall $[0, 1000]$ des Wertebereichs in Abbildung 4.4 dargestellt. In [14] haben BRÄSEL und M. KLEINAU eine andere untere Schranke für $P_{n,m}$ entwickelt, die auf der Analyse eines ersten Enumerationsalgorithmus für Pläne beruht.

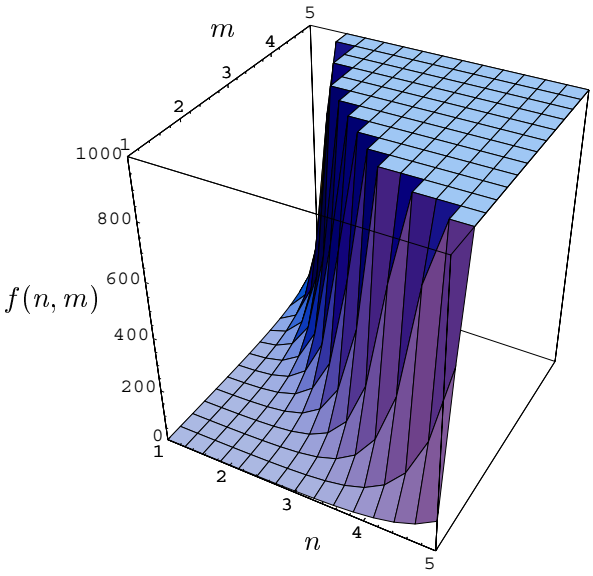

Abbildung 4.4: Untere Schranke (4.11) für die Anzahl aller $n \times m$–Pläne.

Satz 4.4.7 [14] *Für alle $n, m \in \mathbb{N}$ gilt*

$$P_{n,m} \geq \prod_{i=0}^{n-1} \frac{(m+i)!}{i!}. \tag{4.12}$$

Der folgende Satz zeigt, daß die untere Schranke (4.11) schlechter als (4.12) ist.

Satz 4.4.8 *Für alle $n, m \in \mathbb{N}$ gilt*

$$\prod_{i=0}^{n-1} \frac{(m+i)!}{i!} \geq \big((n+m-1)!\big)^{\frac{nm}{n+m-1}}. \tag{4.13}$$

Beweis: Die rechte Seite von (4.13) ist offensichtlich eine symmetrische Funktion von n und m. Die linke Seite ist ebenfalls symmetrisch, denn mit $M = \min(n, m)$ gilt

$$\prod_{i=0}^{n-1} \frac{(m+i)!}{i!} = \prod_{i=0}^{m-1} \frac{(n+i)!}{i!} = \prod_{i=1}^{M} \frac{(m+n-i)!}{(M-i)!}. \tag{4.14}$$

Sei also ohne Beschränkung der Allgemeinheit $n \leq m$, d. h. zu zeigen ist

$$\prod_{i=1}^{n} \frac{(m+n-i)!}{(n-i)!} \geq \big((n+m-1)!\big)^{\frac{nm}{n+m-1}}.$$

Durch Logarithmieren erhält man

$$(m+n-1)\sum_{i=1}^{n}\log(m+n-i)! \geq$$
$$\geq mn\log(m+n-1)! + (m+n-1)\sum_{i=1}^{n}\log(n-i)!\,.$$

Die Anwendung der Logarithmen-Gesetze sowie weitere elementare Umformungen führen schließlich auf

$$n(n-1)\sum_{i=1}^{m}\log(n-1+i) + (m+n-1)\sum_{i=1}^{n-1} i\log i \geq$$
$$\geq mn\sum_{i=1}^{n-1}\log i + (m+n-1)\sum_{i=1}^{n-1} i\log(m+i). \tag{4.15}$$

Es wird nun gezeigt, daß für $n \leq m$ die linke Seite der Ungleichung (4.15) mit m schneller wächst als die rechte Seite von (4.15), d. h. zu zeigen ist die Gültigkeit der Ungleichung

$$\begin{aligned} &n(n-1)\sum_{i=1}^{m+1}\log(n-1+i) + (m+n)\sum_{i=1}^{n-1} i\log i \\ &-n(n-1)\sum_{i=1}^{m}\log(n-1+i) - (m+n-1)\sum_{i=1}^{n-1} i\log i \geq \\ &\geq (m+1)n\sum_{i=1}^{n-1}\log i + (m+n)\sum_{i=1}^{n-1} i\log(m+1+i) \\ &- mn\sum_{i=1}^{n-1}\log i - (m+n-1)\sum_{i=1}^{n-1} i\log(m+i). \end{aligned} \tag{4.16}$$

Vereinfachungen von (4.16) führen auf

$$\sum_{i=1}^{n-1}(m+n-i)\log(m+i) \geq m(n-1)\log(m+n) + \sum_{i=1}^{n-1}(n-i)\log i$$

bzw.

$$\sum_{i=1}^{n-1}\left\{(n-i)\log\left(1+\frac{m}{i}\right) - m\log\left(1+\frac{n-i}{m+i}\right)\right\} \geq 0.$$

Diese Ungleichung ist gültig, wenn für alle $i = 1, \ldots, n-1$ die Ungleichung

$$(n-i)\log\left(1+\frac{m}{i}\right) \geq m\log\left(1+\frac{n-i}{m+i}\right)$$

gilt. Wegen $x > \log(1+x)$ mit $x > 0$ reicht es zu zeigen, daß für alle $i = 1, \ldots, n-1$ die Ungleichung

$$(n-i)\log\left(1+\frac{m}{i}\right) \geq \frac{m(n-i)}{m+i}$$

bzw.

$$\left(1+\frac{i}{m}\right)\log\left(1+\frac{m}{i}\right) \geq 1 \tag{4.17}$$

gilt. Dies ist der Fall, da für festes i, $1 \leq i \leq n-1 \leq m-1$ die erste Ableitung der linken Seite von (4.17) nach m positiv ist und (4.17) für das kleinste m, also $m = i+1$, erfüllt ist, wie sich leicht nachweisen läßt.

Damit ist die Gültigkeit von (4.16) gezeigt. Wegen der bereits erwähnten Symmetrie der beiden Seiten von (4.13) muß nun Beziehung (4.13) bzw. (4.15) nur noch für $n = m$ nachgewiesen werden. Aus (4.15) mit $n = m$ ergibt sich

$$\begin{aligned} n(n-1)\sum_{i=1}^{n}\log(n-1+i) + (2n-1)\sum_{i=1}^{n-1} i\log i \geq \\ \geq n^2\sum_{i=1}^{n-1}\log i + (2n-1)\sum_{i=1}^{n-1} i\log(n+i). \end{aligned} \tag{4.18}$$

Die linke Seite von (4.18) wächst mit n schneller als die rechte Seite, wenn

$$\begin{aligned} &(n+1)n\sum_{i=1}^{n+1}\log(n+i) + (2n+1)\sum_{i=1}^{n} i\log i \\ &-n(n-1)\sum_{i=1}^{n}\log(n-1+i) - (2n-1)\sum_{i=1}^{n-1} i\log i \geq \\ &\qquad \geq (n+1)^2\sum_{i=1}^{n}\log i + (2n+1)\sum_{i=1}^{n} i\log(n-1+i) \\ &\qquad\quad - n^2\sum_{i=1}^{n-1}\log i - (2n-1)\sum_{i=1}^{n-1} i\log(n+i) \end{aligned} \tag{4.19}$$

gilt. Durch elementare Umformungen erhält man

$$\begin{aligned} \sum_{i=1}^{n}(2n-2i+1)\log\left(1+\frac{n}{i}\right) + \sum_{i=1}^{n} 2n\log(n+i) \geq \\ \geq n^2\log(2n+1) + n^2\log(2n). \end{aligned} \tag{4.20}$$

Die beiden Summen auf der linken Seite lassen sich durch

$$\begin{aligned}\sum_{i=1}^{n}(2n-2i+1)\log\left(1+\frac{n}{i}\right) &= \\ &= (2n-1)\log(n+1)+\sum_{i=2}^{n}(2n-2i+1)\log\left(1+\frac{n}{i}\right) \\ &\geq (2n-1)\log(n+1)+[1+3+5+\cdots+(2(n-1)-1)]\log 2 \\ &= (2n-1)\log(n+1)+(n-1)^2\log 2 \\ &= (2n-1)\log\frac{n+1}{2}+n^2\log 2\end{aligned} \tag{4.21}$$

und

$$\begin{aligned}2n\sum_{i=1}^{n}\log(n+i) &\geq 2n\left(n\log n+\frac{n}{2}(\log(2n)-\log n)\right) \\ &= 2n^2\log n+n^2\log 2\end{aligned} \tag{4.22}$$

abschätzen, wobei sich die rechte Seite von (4.22) durch Berechnung der Trapezfläche unterhalb der Funktion $\log x$ im Intervall $[n,2n]$ ergibt. Für die rechte Seite von (4.20) erhält man

$$\begin{aligned}n^2\log(2n+1)+n^2\log(2n) &\leq n^2\log(2(n+1))+n^2\log(2n) \\ &= n^2\log(n+1)+n^2\log n+2n^2\log 2.\end{aligned} \tag{4.23}$$

Schließlich führt die Anwendung von (4.21)–(4.23) auf (4.20) zu

$$(2n-1)\log\frac{n+1}{2}+2n^2\log 2+2n^2\log n\geq n^2\log(n+1)+n^2\log n+2n^2\log 2$$

bzw.

$$(2n-1)\log\frac{n+1}{2}\geq n^2\log\left(1+\frac{1}{n}\right).$$

Wegen $x>\log(1+x)$ mit $x>0$ reicht es zu zeigen, daß $(2n-1)\log\frac{n+1}{2}\geq n$ bzw. $(2-\frac{1}{n})\log\frac{n+1}{2}\geq 1$ gilt, was für $n\geq 3$ offensichtlich der Fall ist. Da Ungleichung (4.19) auch für $n=1,2$ gilt, ist insgesamt die Monotonie von (4.18) für alle $n\in\mathbb{N}$ bewiesen. Schließlich prüft man schnell die Gültigkeit von (4.18) für $n=1,2$ und 3 nach. Damit gilt (4.18) für alle $n\in\mathbb{N}$ und der Satz ist bewiesen. □

Kapitel 5

Plan-Enumeration

In diesem Kapitel geht es um die Enumeration der zulässigen Lösungen eines Open-Shop-Problems mit n Aufträgen und m Maschinen, d. h. es wird für gegebene Werte n und m die Menge aller $n \times m$–Pläne erzeugt.

In [14] haben Bräsel und M. Kleinau 1992 erstmals eine Enumerationsmethode entwickelt, mit der die Werte $P(n, m, r)$ in den Fällen

- $n = 2, \quad 2 \leq m \leq 8 \quad$ und
- $n = 3, \quad 3 \leq m \leq 4$

für $m \leq r \leq nm$ mit Hilfe eines Computers bestimmt werden können. Um die Pläne auch für größere Formate $n \times m$ aufzählen zu können, werden in diesem Kapitel die Verfahren aus [14] in vielfacher Hinsicht modifiziert.

In Abschnitt 5.1 werden verschiedene Äquivalenzrelationen behandelt, die die Menge aller $n \times m$–Pläne jeweils in disjunkte Äquivalenzklassen partitionieren. Weiterhin wird für die Menge der Äquivalenzklassen ein geeignetes Vertretersystem beschrieben. Die daraus gewonnenen Erkenntnisse bilden die Basis für einen neuen effizienten Algorithmus zur Enumeration aller Pläne eines Open-Shop-Problems mit n Aufträgen und m Maschinen.

Die Abschnitte 5.2 und 5.3 dokumentieren diesen Enumerationsalgorithmus, wobei die Enumeration der Technologien eines Open-Shop-Problems vorangestellt ist. Die Beschreibung des gesamten Algorithmus mit den zugehörigen Teilprozeduren ist in Abschnitt 5.3 enthalten. Zum Abschluß wird in Abschnitt 5.4 eine Zusammenfassung und Auswertung der neu erzielten Werte für die Anzahl der $n \times m$–Pläne gegeben.

5.1 Pläne gleicher Struktur

Die in den folgenden Unterabschnitten vorgestellten Äquivalenzrelationen (Isomorphie, Äquivalenz und Struktur-Äquivalenz) sind für eine effiziente Aufzählung und Charakterisierung der Pläne mit gleichen Eigenschaften bzw. Strukturen grundlegend. Ein Plan A eines Shop-Scheduling-Problems mit n Aufträgen und m Maschinen wird mit der $n \times m$–Matrix $A = (a_{ij})$ identifiziert, die aus den Rängen $a_{ij} = \varrho(o_{ij})$ der Knoten bzw. Operationen o_{ij} des zugehörigen $n \times m$–Ablaufgraphen besteht.

Isomorphie von Plänen

Die $n \times m$–Pläne bilden eine spezielle Klasse innerhalb der Menge der lateinischen Rechtecke $\mathcal{L}_{n,m,r}$. Die $n \times m$–Pläne mit $n = m$ heißen *quadratische Pläne*. Die rangminimalen quadratischen Pläne (mit $n = m = r$) sind meist unter dem Namen *lateinische Quadrate* bekannt.

Lateinische Quadrate können als Multiplikationstafeln von Quasigruppen[1] aufgefaßt werden. In [29] bezeichnen DÉNES und KEEDWELL zwei lateinische Quadrate als *isomorph*, wenn sie durch *dieselbe* Permutation von Zeilen, Spalten und Elementen der Belegungsmenge ineinander überführt werden können, d. h. wenn die entsprechenden Quasigruppen isomorph sind (bezüglich des Isomorphie von Quasigruppen)[2].

Die Übertragung des für lateinische Quadrate verwandten Begriffs „isomorph“ auf $n \times n$–Pläne ist nicht sinnvoll, da die Permutation von Elementen der Belegungsmenge für allgemeine quadratische Pläne keine abgeschlossene Operation ist, d. h. wenn Elemente eines Planes vertauscht werden, führt dies zu Matrizen, die nicht notwendig der zusätzlichen Eigenschaft für Pläne (vgl. Definition 3.2.1) genügen müssen.

In [17] bezeichnet BROWN zwei lateinische Rechtecke als *isomorph*, wenn sie durch Permutationen von Zeilen, Spalten und Elementen ineinander überführt werden können (vgl. auch die *Isotopie* von lateinischen Quadraten und zugehörigen Quasigruppen in DÉNES und KEEDWELL [29]). Das heißt, im Fall isomorpher lateinischer Rechtecke müssen im Gegensatz zur obigen Definition der isomorphen lateinischen Quadrate die verwendeten Permutationen nicht identisch sein. Aus dem gleichen Grund wie bei der „Isomorphie von lateinischen Quadraten“ ist die Übertragung des Begriffs „Isomorphie von lateinischen Rechtecken“ auf $n \times m$–Pläne

[1] Eine Menge Q heißt *Quasigruppe*, wenn auf ihr eine Verknüpfung $(\cdot)$ definiert ist, und für jedes Paar $a, b \in Q$ die Gleichungen $a \cdot x = b$ und $y \cdot a = b$ eindeutig nach x bzw. y auflösbar sind.

[2] Zwei Quasigruppen G und G' heißen *isomorph*, wenn es eine bijektive Abbildung $\varphi : G \to G'$ gibt mit $\varphi(ab) = \varphi(a)\varphi(b)$ für alle $a, b \in G$.

ebenfalls unangebracht. Es wird daher die Isomorphie von Plänen folgendermaßen definiert:

Definition 5.1.1 Zwei Pläne A und B heißen *isomorph*, $A \cong B$, wenn A durch eine Permutation ϱ von Zeilen und eine Permutation σ von Spalten in B überführt werden kann.

Ein Isomorphismus (ϱ, σ), der einen Plan A in sich selbst überführt, heißt *Automorphismus* von A. Die Isomorphie von Plänen ist eine Äquivalenzrelation, die die Menge der Pläne in disjunkte *Isomorphieklassen* aufteilt. Der Begriff „Isomorphie" wurde so gewählt, daß die Pläne einer Isomorphieklasse bei passender Indizierung der Aufträge J_i und Maschinen M_j identisch sind.

Äquivalenz von Plänen

Im quadratischen Fall ist bei der Zusammenfassung von Plänen mit gleichen Eigenschaften jeweils nicht nur eine andere Indizierung der Aufträge J_i und Maschinen M_j denkbar, sondern die Rolle der Aufträge kann auch komplett mit der der Maschinen vertauscht werden.

Zu einer gegebenen $n \times n$–Matrix A wird die Matrix, die sich durch *Transposition*, also durch Spiegelung von A an ihrer Hauptdiagonalen, ergibt, mit A^T bezeichnet. Offensichtlich ist A^T genau dann ein $n \times n$–Plan, wenn A einer ist. Analog zur Terminologie bei allgemeinen Matrizen heißt A^T der *transponierte Plan* von A.

Definition 5.1.2 Zwei Pläne A und B heißen *äquivalent*, $A \equiv B$, wenn $A \cong B$ oder $A \cong B^T$ gilt.

Wird für Pläne stets ein festes Format $n \times m$ mit $n \neq m$ betrachtet, so gilt für zwei Pläne A und B genau dann $A \cong B$, wenn $A \equiv B$ ist, d. h. die Begriffe „Isomorphie" und „Äquivalenz" fallen in diesem Fall zusammen. Wie leicht zu sehen ist, muß für zwei quadratische Pläne A und B mit $A \equiv B$ allerdings nicht notwendig $A \cong B$ gelten, d. h. für $n = m$ ist die Anzahl nicht-isomorpher Pläne mindestens so groß wie die Anzahl nicht-äquivalenter Pläne.

Im nächsten Unterabschnitt wird deutlich, daß die Äquivalenz von $n \times m$–Plänen mit der Isomorphie der zugehörigen $n \times m$–Ablaufgraphen gleichbedeutend ist. Zunächst wird jedoch ein Algorithmus entwickelt, der in polynomialer Zeit entscheidet, ob zwei gegebene Pläne A und B äquivalent sind oder nicht.

Definition 5.1.3 Ein Plan $A = (a_{ij})$ ist in *Normalform*, wenn $a_{11} = 1$ gilt, und die Einträge der ersten Zeile und der ersten Spalte jeweils aufsteigend sind.

Algorithmus 5.1.4 *Plan-Äquivalenz*

Eingabe: Zwei beliebige $n \times m$–Pläne $A = (a_{ij})$ und $B = (b_{ij})$.
Ausgabe: Eine Äquivalenz (in Form von Zeilen- und Spaltenpermutationen sowie Transposition), falls eine existiert.

1. Bringe A durch geeignete Zeilen- und Spaltenpermutation ϱ_A und σ_A in eine Normalform $A' = (a'_{ij})$ mit $a'_{11} = 1$.

2. Für alle Einträge $b_{kl} = 1$ in B:

   (a) Bringe B durch geeignete Zeilen- und Spaltenpermutation ϱ_B und σ_B in die Normalform $B' = (b'_{ij})$ mit $b'_{11} = b_{kl}$.

   (b) Wenn $A' = B'$ ist:

   A und B sind äquivalent. Die Äquivalenz wird durch die Zeilenpermutation $\varrho_A \varrho_B^{-1}$ und Spaltenpermutation $\sigma_A \sigma_B^{-1}$ beschrieben.

   (c) Wenn $n = m$ ist:

   i. Setze $B'' := B'^T$.

   ii. Wenn $A' = B''$ ist:

   A und B sind äquivalent. Die Äquivalenz wird durch die Zeilenpermutation $\varrho_A \varrho_B^{-1}$ und Spaltenpermutation $\sigma_A \sigma_B^{-1}$ sowie durch Transposition beschrieben.

Die Korrektheit des Algorithmus 5.1.4 folgt aus den vorangegangenen Betrachtungen, insbesondere aus Definition 5.1.2 und Definition 5.1.3.

Satz 5.1.5 *Für $n \leq m$ kann die Äquivalenz von $n \times m$–Plänen in der Zeit $O(nm^2)$ entschieden werden.*

Beweis: Es wird Algorithmus 5.1.4 betrachtet: Die Sortierung der Zeilen und Spalten, die zur Konstruktion der Normalformen in Schritt 1 und 2a erforderlich ist, benötigt $O(n \log n + m \log m)$ Zeit. Mit $n \leq m$ ergibt sich $O(m \log m)$ als obere Schranke. Die Vergleiche der Elemente zweier Matrizen in Schritt 2b und 2c können für $n \leq m$ jeweils in der Zeit $O(m^2)$ ausgeführt werden. Da B ein lateinisches Rechteck ist, gibt es höchstens n verschiedene Normalformen von B. Also muß der gesamte Schritt 2 maximal n-mal wiederholt werden, d. h. die Zeitkomplexität für den gesamten Algorithmus beträgt $O(nm^2)$. □

Isomorphie-Problem für Ablaufgraphen

Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen *isomorph*, wenn es eine bijektive Abbildung $\varphi : V_1 \to V_2$ gibt, so daß für alle Paare von Knoten $v, w \in V_1$ genau dann $\{v, w\} \in E_1$ gilt, wenn $\{\varphi(v), \varphi(w)\} \in E_2$ ist. Das heißt, zwei Graphen sind isomorph, wenn man die Knoten des einen Graphen auf die des anderen so abbilden kann, daß die Nachbarschaften zwischen den Knoten erhalten bleiben. Eine Bijektion φ mit dieser Eigenschaft heißt *Isomorphismus*. Diese Definition kann für Digraphen angepaßt werden, indem man für die Kanten die ungeordneten Paare von Knoten jeweils durch geordnete ersetzt.

Nach Satz 3.2.2 besteht eine eineindeutige Beziehung zwischen $n \times m$–Plänen und zugehörigen $n \times m$–Ablaufgraphen. Der folgende Hilfssatz gibt eine Charakterisierung äquivalenter Pläne anhand von Isomorphismen zwischen den zugehörigen azyklischen Digraphen. Diese Charakterisierung kann ebenfalls als Definition der Äquivalenz von Plänen benutzt werden.

Hilfssatz 5.1.6 [12] *Zwei Pläne A und B sind genau dann äquivalent, wenn die zugehörigen Ablaufgraphen $G(A)$ und $G(B)$ isomorph sind.*

Beweis: Offensichtlich besteht ein $n \times m$–Ablaufgraph $G(A)$ aus $n + m$ azyklischen Turnieren[3]. Dabei handelt es sich um n Turniere mit m Knoten, die mit den Zeilen des zugehörigen Plans A korrespondieren und m Turniere mit n Knoten entsprechend für die Spalten von A. Zwei Turniere von $G(A)$ sind genau dann knotendisjunkt, wenn sie entweder zu zwei verschiedenen Zeilen oder zu zwei verschiedenen Spalten von A gehören. Deshalb können die Knoten von Zeilen und die Knoten von Spalten permutiert werden, ohne die Grundstruktur der Nachbarschaften zwischen den Knoten des entsprechenden Ablaufgraphen G zu verändern. Für jeden Plan A gibt es offensichtlich auch eine Bijektion zwischen den Knotenmengen der Ablaufgraphen $G(A)$ und $G(A^T)$, die die Nachbarschaften invariant läßt. Die Permutationen und die eventuelle Matrix-Transposition, die den Plan A in den Plan B überführen, legen also auf diese Weise den Isomorphismus zwischen den zugehörigen Ablaufgraphen fest. □

Es wird im weiteren der in diesem Hilfssatz dargestellte Zusammenhang zur Gewinnung einer Komplexitätsaussage für eines der bekanntesten Probleme aus der Graphentheorie benutzt: Das Problem der Entscheidung, ob zwei gegebene Graphen isomorph sind, heißt *Graphen-Isomorphie-Problem*. Die Komplexität dieses Problems ist bis heute ungeklärt, d. h. es ist weder ein polynomialer Lösungsalgorithmus bekannt, noch konnte gezeigt werden, daß es sich um ein $\mathcal{NP}$-vollständiges Problem handelt.

[3]Ein *Turnier* $T = (V, E)$ mit $|V| = n$ ist eine Orientierung des vollständigen Graphen K_n.

Das Graphen-Isomorphie-Problem spielt in der Komplexitätstheorie eine bedeutende Rolle, denn es ist eines der Probleme in der Klasse $\mathcal{NP}$, die möglicherweise weder in der Klasse $\mathcal{P}$ liegen noch $\mathcal{NP}$-vollständig sind. Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, existieren tatsächlich solche Probleme, die zwischen diesen beiden Komplexitätsklassen liegen (siehe LADNER [57]). Die große Bedeutung des Graphen-Isomorphie-Problems in diesem Bereich kommt auch durch die besondere Erwähnung in frühen Arbeiten der Komplexitätstheorie [27, 51, 36] zum Ausdruck.

Die Komplexität des *Digraphen-Isomorphie-Problems*, also des entsprechenden Entscheidungsproblems für gerichtete Graphen, ist polynomial äquivalent zum Graphen-Isomorphie-Problem (siehe MILLER [70]). Während für beliebige Graphen bzw. Digraphen bis heute kein effizienter Algorithmus zur Entscheidung über die Existenz eines Isomorphismus im allgemeinen Fall bekannt ist, hat man bereits einige Graphenklassen bestimmt, in denen dieses Problem in polynomialer Zeit gelöst werden kann: Es sind polynomiale Algorithmen z. B. im Falle von planaren Graphen (HOPCROFT und WONG [46]), Graphen mit beschränkter maximaler Knotenanzahl (LUKS [64]), Graphen mit beschränktem durchschnittlichen Geschlecht (CHEN [22]) und Intervallgraphen (LUEKER und BOOTH [63]) bekannt.

Zur Lösung des Digraphen-Isomorphie-Problems sind polynomiale Algorithmen für minimale serien-parallele Digraphen[4] (VALDES, TARJAN und LAWLER [98]) und für zyklische Turniere[5] (PONOMARENKO [74]) entwickelt worden.

Im anschließenden Satz zeigen wir, daß die Menge aller $n \times m$–Ablaufgraphen eine weitere Klasse von Digraphen ist, in der die Entscheidung über die Existenz eines Isomorphismus in polynomialer Zeit möglich ist.

Satz 5.1.7 [12] *Es ist in polynomialer Zeit entscheidbar, ob ein Isomorphismus zwischen zwei azyklischen Orientierungen des Hamming-Graphen $K_n \times K_m$ existiert.*

Beweis: Die Menge der azyklischen Orientierungen des Hamming-Graphen $K_n \times K_m$ entspricht der Menge der $n \times m$–Ablaufgraphen. Einem gegebenen $n \times m$–Ablaufgraphen kann nach Satz 3.2.2 eindeutig ein $n \times m$–Plan zugeordnet werden. Satz 3.2.4 zeigt, daß es in polynomialer Zeit möglich ist, zu einem gegebenen Ablaufgraphen den zugehörigen Plan zu berechnen. Zusammen mit dem polynomialen Algorithmus 5.1.4 zur Entscheidung, ob zwei Pläne äquivalent sind, und Hilfssatz 5.1.6 folgt, daß in der Klasse der azyklischen Orientierungen des Hamming-Graphen vom Typ $K_n \times K_m$ das Graphen-Isomorphie-Problem polynomial lösbar ist. □

[4] Ein *minimaler serien-paralleler Digraph* (MSP-Digraph) ist ein Digraph, der sich durch eine Folge von seriellen und parallelen Kompositionen rekursiv aus kleineren MSP-Digraphen konstruieren läßt, wobei der triviale Digraph mit nur einem Knoten auch ein MSP-Digraph ist (Rekursionsanfang).

[5] Ein Turnier $T = (V, E)$ heißt *zyklisch*, wenn seine Automorphismengruppe, d. h. die Gruppe der Isomorphismen $\varphi : V \to V$, die zyklische Permutation $(1, 2, \ldots, n)$ enthält.

Struktur-Äquivalenz von Plänen

Es ist sinnvoll, bei der Zusammenfassung von $n \times m$–Plänen mit gleichen Eigenschaften nicht nur die jeweilige Neuindizierung der Aufträge J_i und Maschinen M_j untereinander (Isomorphie) sowie zusätzlich die komplette Vertauschung von Aufträgen und Maschinen im Fall $n = m$ (Äquivalenz) zuzulassen. Die Grundstruktur eines Planes ändert sich ebenfalls nicht, wenn alle Reihenfolgen von Technologie und Organisation vollständig umgekehrt werden. Daher wird in diesem Abschnitt zusätzlich zur Isomorphie „$\cong$“ und zur Äquivalenz „$\equiv$“ eine weitere Äquivalenzrelation für Pläne betrachtet, die auf der Umkehrung aller Reihenfolgen bzw. gerichteten Kanten im entsprechenden Ablaufgraphen basiert. Bei dieser Äquivalenzrelation handelt es sich um die sogenannte Struktur-Äquivalenz.

Es sei A ein Plan. Der Plan, der sich durch Umkehrung der Technologie und Organisation von A ergibt, heißt *Umkehrplan* $\overline{A}$ von A

Definition 5.1.8 Zwei Pläne A und B heißen *struktur-äquivalent*, $A \equiv_S B$, wenn $A \equiv B$ oder $A \equiv \overline{B}$ gilt.

Diese durch „$\equiv_S$“ definierte Äquivalenzrelation ist nicht so streng gefaßt wie die Äquivalenz („$\equiv$“) und die Isomorphie („$\cong$“): Offensichtlich folgt für zwei Pläne A und B aus $A \cong B$ die Beziehung $A \equiv B$, und $A \equiv B$ impliziert die Beziehung $A \equiv_S B$.

Vertretersysteme

In [12] haben Bräsel, Harborth und Willenius die Anzahl der Isomorphie- bzw. Äquivalenzklassen in der Menge der Pläne folgendermaßen bestimmt: Es werden zunächst alle $n \times m$–Pläne mit Hilfe des vollständigen Enumerationsalgorithmus (Bräsel und M. Kleinau [14]) für kleine Werte n und m erzeugt. Danach müssen diese $n \times m$–Pläne paarweise anhand von Algorithmus 5.1.4 verglichen werden, um letztlich nur die nicht-isomorphen bzw. nicht-äquivalenten Pläne zu zählen.

Diese Methode der Enumeration nicht-isomorpher bzw. nicht-äquivalenter Pläne ist aufgrund der großen Anzahl der durchzuführenden paarweisen Vergleiche zur schnellen Berechnung der gewünschten Plan-Anzahl bereits für $m \geq n \geq 4$ unzureichend. Es wird nun die in [11] entwickelte neue Grundlage vorgestellt, mit der alle Pläne eines gegebenen Formats effektiver als in [12] erzeugt bzw. gezählt werden können.

Zur Enumeration der Äquivalenzklassen bestimmter kombinatorischer Objekte wendet man häufig algebraische Methoden an, mit denen in geeigneter Weise nur die Vertreter eines disjunkten Vertretersystems für die Äquivalenzklassen erzeugt oder gezählt werden.

Es seien $A = (a_{ij})$ und $B = (b_{ij})$ zwei $n \times m$–Matrizen mit $a_{ij}, b_{ij} \in \mathbb{N}$ für alle $i = 1, \ldots, n$ und $j = 1, \ldots, m$. Die Matrix A heißt *lexikographisch kleiner* als die Matrix B ($A <_{lex} B$), wenn es ein Indexpaar (k, l) gibt, für das $a_{kl} < b_{kl}$ gilt und für alle Indexpaare (i, j) mit $1 \leq i < k,\ 1 \leq j \leq m$ und $i = k,\ 1 \leq j < l$ die Beziehung $a_{ij} = b_{ij}$ erfüllt ist. Auf diese Weise ist auf der Menge S der $n \times m$–Matrizen mit Einträgen aus $\mathbb{N}$ die *lexikographische Ordnung* der Form $A_1 <_{lex} A_2 <_{lex} \cdots <_{lex} A_r$ für die Matrizen $A_1, A_2, \ldots, A_r$ gegeben. Offensichtlich handelt es sich bei der Menge S mit dieser Ordnung um eine linear geordnete Menge[6].

Definition 5.1.9 Es sei S die Menge aller $n \times m$–Matrizen mit Einträgen aus $\mathbb{N}$. Eine Menge $Q \subseteq S$ werde durch eine Äquivalenzrelation in die disjunkten Teilmengen $Q_1, \ldots, Q_r$ partitioniert, also $Q = Q_1 \cup \cdots \cup Q_r$ und $Q_i \cap Q_j = \varnothing$ für $i \neq j$. Es sei $\min_{lex}(Q_i)$ das lexikographische Minimum der Elemente der Menge $Q_i \subseteq S$. Dann ist die Funktion f mit

$$\begin{aligned} f : \{1, \ldots, r\} &\rightarrow S, \\ i &\mapsto f(i) = \min_{lex}(Q_i) \end{aligned} \tag{5.1}$$

die Auswahlfunktion eines Vertretersystems für die gegebene Äquivalenzrelation auf Q.

Isomorphie, Äquivalenz und Struktur-Äquivalenz stellen Äquivalenzrelationen dar, durch die die Menge aller $n \times m$–Pläne jeweils in disjunkte Äquivalenzklassen partitioniert wird. In den folgenden Abschnitten beruhen die benutzten Vertretersysteme für die Äquivalenzklassen in der Menge der Technologien und Pläne direkt oder indirekt auf der Auswahlfunktion (5.1). Es wird also häufig die lexikographische kleinste Matrix als Repräsentant ihrer Klasse benutzt.

In Abbildung 5.1 sind unter allen 2×2–Plänen alle Vertreter einer Isomorphie- bzw. Äquivalenzklasse hervorgehoben. Außerdem sind in dieser Abbildung die vier verschiedenen Isomorphieklassen zeilenweise angeordnet.

5.2 Technologie-Anzahlen

Im ersten Teil neuen Enumerationsalgorithmus für $n \times m$–Pläne werden zunächst nur Technologien betrachtet. Eine Technologie wird stets mit der Matrix *TR* identifiziert (siehe Abbildung 3.5, Seite 23). Diese $n \times m$–Matrix *TR* besteht für ein Shop-Scheduling-Problem mit n Aufträgen und m Maschinen aus n Zeilen, die jeweils Permutationen der Zahlen $1, \ldots, m$ darstellen.

[6] Eine *linear geordnete Menge* ist eine Menge in der je zwei Elemente anhand einer zweistelligen Relation vergleichbar sind.

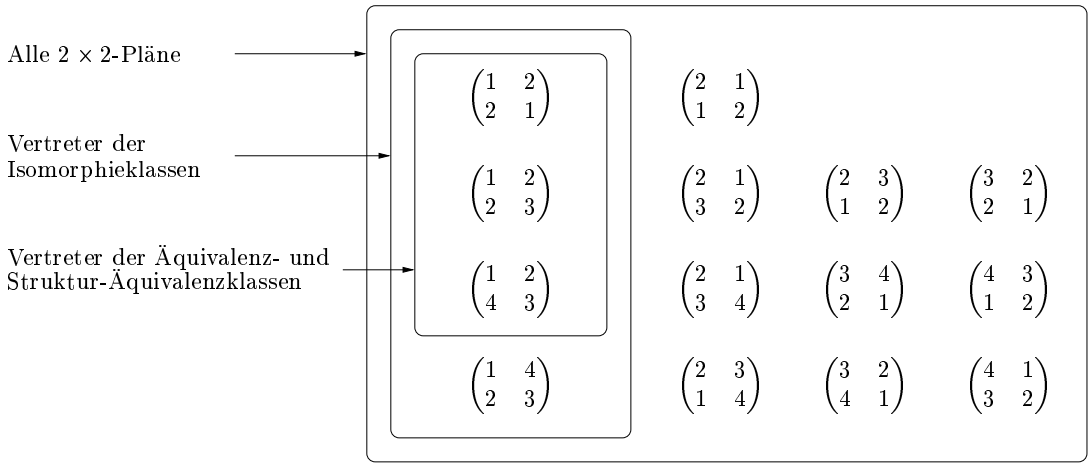

Abbildung 5.1: Ein Vertretersystem der 2×2–Pläne.

$n \backslash m$	2	3	4	5	6
2	4	36	576	14 400	518 400
3	8	216	13 824	1 728 000	373 248 000
4	16	1 296	331 776	207 360 000	268 738 560 000
5	32	7 776	7 962 624	24 883 200 000	193 491 763 200 000
6	64	46 656	191 102 976	2 985 984 000 000	139 314 069 504 000 000
7	128	279 936	4 586 471 424	358 318 080 000 000	100 306 130 042 880 000 000

Tabelle 5.1: Gesamtanzahlen der $n \times m$–Technologien.

Im folgenden wird das in Abschnitt 5.1 angewandte Prinzip der Zusammenfassung von Plänen gleicher Struktur auf die Technologien TR übertragen, um sich unter der Gesamtanzahl der Technologien eines Formats (vgl. Tabelle 5.1) auf die strukturell verschiedenen konzentrieren zu können.

Definition 5.2.1 Zwei Technologien TR^1 und TR^2 heißen *isomorph* ($TR^1 \cong TR^2$), wenn TR^1 durch eine Permutation ϱ von Zeilen und eine Permutation σ von Spalten in TR^2 überführt werden kann.

Zur Abschätzung der Anzahl nicht-isomorpher Technologien ist die Betrachtung sogenannter reduzierter Technologien TR sinnvoll. In der folgenden Definition und im weiteren Teil dieses Kapitels wird die Gesamtheit aller Einträge einer Zeile TR_i der Technologie TR stets als Zeilenvektor $TR_i = (tr_{i1}, tr_{i2}, \ldots, tr_{im})$ aufgefaßt.

Definition 5.2.2 Eine $n \times m$–Technologie TR heißt *reduziert*, wenn

$$TR_1 = (1, 2, \ldots, m)$$

ist, und $TR_i \leq_{lex} TR_{i+1}$ für alle $1 \leq i \leq n-1$ gilt.

Hilfssatz 5.2.3 [11] *Für die Anzahl $I_{TR}(n,m)$ der Isomorphieklassen der Technologien TR des Formats $n \times m$ gilt*

$$\frac{1}{n}\binom{m!+n-2}{n-1} \leq I_{TR}(n,m) \leq \binom{m!+n-2}{n-1}. \tag{5.2}$$

Beweis: Jede Technologie TR kann durch Zeilen- und Spaltenpermutationen in eine reduzierte Form überführt werden. Also ist die Anzahl der reduzierten TR's eine obere Schranke für die Anzahl nicht-isomorpher TR's. Da die erste Zeile einer reduzierten TR festgelegt ist, entspricht die Anzahl der reduzierten TR's der Anzahl der $(n-1)$–Kombinationen der Menge der Permutationen der Länge m mit Wiederholung, und diese Anzahl ist $\binom{m!+n-2}{n-1}$.

Es sei k die Anzahl verschiedener Zeilenvektoren in Technologien TR einer Isomorphieklasse. Dann enthält diese Isomorphieklasse höchstens k verschiedene reduzierte TR's, denn jede der k verschiedenen Zeilen kann durch Anwendung geeigneter Zeilen- und Spaltenpermutation als erste Zeile benutzt werden. Da eine Technologie TR höchstens n verschiedene Zeilen besitzt, gilt die in (5.2) angegebene untere Schranke. □

Ein *Automorphismus* (ϱ, σ) einer Technologie TR ist eine Zeilen- und Spaltenpermutation, die TR in sich selbst überführt. Ein Automorphismus (ϱ, σ) heißt *nichttrivial*, wenn die Spaltenpermutation σ nicht die Identität ist.

Die in Hilfssatz 5.2.3 gegebenen Schranken für die Anzahl nicht-isomorpher Technologien sind nicht scharf. Satz 5.2.5 zeigt, daß sich die Anzahl $I_{TR}(n,m)$ für unendlich viele Paare (n,m) exakt angeben läßt. Zum Beweis dieses Ergebnisses benötigt man folgendes vorbereitendes Resultat:

Hilfssatz 5.2.4 [11] *Es gibt genau dann eine $n \times m$–Technologie TR mit einem nichttrivialen Automorphismus, wenn n durch eine Zahl $p \in \mathbb{N}$ mit $1 < p \leq m$ teilbar ist.*

Beweis: Es sei TR eine $n \times m$–Technologie mit einem nichttrivialen Automorphismus (ϱ, σ). Dann enthält die Spaltenpermutation σ mindestens einen Zyklus der Länge p mit $1 < p \leq m$. Es gilt $TR_{\varrho(i)} = \sigma(TR_i)$ für alle i, da (ϱ, σ) ein Automorphismus ist. Sei r die Länge des Zyklus von ϱ, der i enthält. Dann gilt $\varrho^r(i) = i$ und damit $\sigma^r(TR_i) = TR_i$. Also ist σ^r die Identität und r ein Vielfaches von p. Die gleiche Schlußfolgerung kann für alle Zeilen i, $1 \leq i \leq n$ angewandt werden. Jeder Zyklus in ϱ hat daher als Länge ein Vielfaches von p. Da n die Summe aller Zykluslängen von ϱ ist, ist n ebenfalls Vielfaches von p.

Sei umgekehrt $n = sp$ mit $1 \leq p \leq m$. Dann ist die Technologie $TR = (tr_{ij})$ mit

$$tr_{ij} = \begin{cases} \big((\lceil i/s \rceil + j - 2) \mod p \big) + 1 & \text{für } j \leq p; \\ j & \text{für } p < j \leq m; \end{cases} \tag{5.3}$$

für alle $1 \leq i \leq n$, eine $n \times m$–Technologie, die einen nichttrivialen Automorphismus besitzt. □

Die im Beweis angegebene Matrix ist die einfachste Form einer Technologie mit einem nichttrivialen Automorphismus. Zum Beispiel ergibt sich aus (5.3) mit $n = 6$, $m = 7$ und $p = 3$ die Technologie

$$(tr_{ij}) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 1 & 4 & 5 & 6 & 7 \\ 2 & 3 & 1 & 4 & 5 & 6 & 7 \\ 2 & 3 & 1 & 4 & 5 & 6 & 7 \end{pmatrix}. \tag{5.4}$$

Die Aufzählung der Technologien bereitet nur dann Schwierigkeiten, wenn nichttriviale Automorphismen existieren. Ansonsten kann die Anzahl $I_{TR}(n,m)$ der paarweise nicht-isomorphen Technologien anhand des folgenden Satzes leicht berechnet werden.

Satz 5.2.5 [11] *Es sei n durch keine Zahl p mit $1 < p \leq m$ teilbar. Dann gilt*

$$I_{TR}(n,m) = \sum_{k=1}^{n} \binom{m! - 1}{k - 1} \binom{n-1}{k-1} \frac{1}{k}. \tag{5.5}$$

Beweis: Es wird zunächst die Anzahl der reduzierten Technologien TR mit genau k paarweise verschiedenen Zeilenvektoren $TR_i,\ i = 1, \ldots, k$ betrachtet. Der erste Zeilenvektor einer reduzierten Technologie ist stets $TR_1 = (1, 2, 3, \ldots, m)$, daher verbleiben $\binom{m!-1}{k-1}$ Möglichkeiten, diese Menge von insgesamt k verschiedenen Zeilenvektoren auszuwählen. Weiterhin tritt jeder dieser Zeilenvektoren $TR_i,\ i = 1, \ldots, k$ mit einer Häufigkeit $h_i \geq 1$ auf, wobei offensichtlich $n = h_1 + h_2 + \cdots + h_k$ gilt. Die Anzahl der Möglichkeiten, die Zahl n auf diese Weise in k positive Summanden zu zerlegen, ist $\binom{n-1}{k-1}$. Also existieren insgesamt $\binom{m!-1}{k-1}\binom{n-1}{k-1}$ reduzierte Technologien TR mit genau k verschiedenen Zeilenvektoren. Wie bereits im Beweis von Hilfssatz 5.2.3 gezeigt wurde, enthält jede Isomorphieklasse von Technologien mit k verschiedenen Zeilen maximal k verschiedene reduzierte Technologien. Nach Voraussetzung ist n durch keine Zahl p mit $1 < p \leq m$ teilbar. Wegen Hilfssatz 5.2.4 hat jede solche Isomorphieklasse in diesem Fall genau k verschiedene reduzierte Technologien. Die Aussage des Satzes folgt also, wenn man über k summiert und dabei jeweils den Faktor $1/k$ hinzufügt. □

Anwendung des Cauchy-Frobenius-Hilfssatzes

In diesem Abschnitt wird eine weit verbreitete algebraische Methode angewendet, um eine exakte Formel für die Anzahl $I_{TR}(n, m)$ der Isomorphieklassen der $n \times m$–Technologien herleiten zu können. Das zentrale Hilfsmittel in diesem Zusammenhang ist der sogenannte Cauchy-Frobenius-Hilfssatz, der zunächst zitiert wird. Vorbereitend dazu benötigt man die anschließend erklärten Begriffe.

Es sei G eine Gruppe von Elementen, die auf einer endlichen Menge X operiere, also $G \times X \to X, (g, x) \mapsto g(x)$. Man kann die Gruppe G bezüglich X in sogenannte Bahnen zerlegen: Für ein Element $x \in X$ heißt die Menge $G(x) = \{g(x) \mid g \in G\}$ die *Bahn* von x. Der *Stabilisator* von x ist die Menge $G_x = \{g \in G \mid g(x) = x\}$. Offensichtlich ist der Stabilisator von x eine Untergruppe von G. Weiterhin kann gezeigt werden, daß stets $|G(x)|\ |G_x| = |G|$ gilt.

Hilfssatz 5.2.6 [21] *Sei G eine Gruppe, die auf einer endlichen Menge X operiere. Mit $B(G)$ werde die Menge der Bahnen von G auf X bezeichnet. Es gilt*

$$|B(G)| = \frac{1}{|G|} \sum_{g \in G} |\{x \in X : g(x) = x\}|. \tag{5.6}$$

Dieser Cauchy-Frobenius-Hilfssatz ist irrtümlich auch unter dem Namen „Burnside's Lemma“ bekannt. Dieser Irrtum ist durch folgenden Zusammenhang begründet: Während Burnside in [21] noch die entsprechende Literaturquelle zitiert, ist dies in der 2. Auflage von [21] nicht mehr der Fall. Daher sind viele Autoren später davon

ausgegangen, daß dieser Hilfssatz auf BURNSIDE zurückgeht. Der Hilfssatz ist jedoch CAUCHY und FROBENIUS bereits deutlich früher bekannt gewesen. Historische Anmerkungen hierzu findet man in [5, 28, 73, 103].

Die *symmetrische Gruppe* S_m ist die Gruppe der Ordnung $m!$, die aus allen Permutationen der Zahlen $1, 2, \ldots, m$ besteht. Es sei π ein Element von S_m. Der kleinste Wert k, für den π^k die Identität ist, heißt *Ordnung* der Permutation π. Der Cauchy-Frobenius-Hilfssatz gibt uns im Zusammenhang mit der Enumeration der Pläne von Shop-Scheduling-Problemen die Möglichkeit, die Anzahl der nicht-isomorphen Technologien exakt anzugeben:

Satz 5.2.7 [11] *Es sei $n_{k,m}$ die Anzahl der Permutationen der Ordnung k in der symmetrischen Gruppe S_m. Dann gilt*

$$I_{TR}(n,m) = \frac{1}{m!} \sum_{k|n} n_{k,m} \binom{\frac{m!}{k} + \frac{n}{k} - 1}{\frac{n}{k}}. \tag{5.7}$$

Beweis: Der Cauchy-Frobenius-Hilfssatz wird direkt angewandt. Es sei X die Menge der reduzierten Technologien TR und G die symmetrische Gruppe S_m. Der Wert $I_{TR}(n,m)$ ist dann gerade die Anzahl der Bahnen von G auf X. Jedes Element $\sigma \in S_m$ wird als Spaltenpermutation der Technologie TR aufgefaßt, und es werde zu σ stets automatisch die Zeilenpermutation angewandt, die wieder eine reduzierte Technologie erzeugt. Für alle Spaltenpermutationen σ wird jeweils die Anzahl der Technologien TR gesucht, für die σ ein Automorphismus ist.

Offensichtlich kann σ nur dann ein Automorphismus sein, wenn die Ordnung von σ ein Teiler von n ist. Es sei k die Ordnung von σ in S_m, dann sind alle Zeilenvektoren $TR_i, \sigma(TR_i), \sigma^2(TR_i), \ldots, \sigma^{k-1}(TR_i)$ verschieden. Daher enthält jede Technologie, die unter σ gleich bleibt, entweder alle oder keinen dieser Zeilenvektoren. Das heißt, durch einen ausgewählten Zeilenvektor sind stets automatisch weitere $k-1$ festgelegt, und man erhält als Anzahl verschiedener Technologien TR, für die σ Automorphismus ist, die Anzahl der n/k–Kombinationen aus $m!/k$ solchen Mengen von Zeilenvektoren mit Wiederholung. Durch Summation über alle Spaltenpermutationen σ, deren Ordnung Teiler von n ist, ergibt sich (5.7). □

Wenn die Anzahl $n_{k,m}$ der Permutationen der Ordnung k in der symmetrischen Gruppe S_m für alle k mit $k|n$ bekannt ist, liefert dieser Satz sofort die Anzahl $I_{TR}(n,m)$ nicht-isomorpher Technologien des Formats $n \times m$. Eine Übersicht über die Werte $I_{TR}(n,m)$ für $1 \leq n \leq 9$ und $1 \leq m \leq 6$ gibt Tabelle 5.2. Beim Vergleich mit Tabelle 5.1 wird die erhebliche Reduzierung der Technologie-Anzahlen deutlich, die durch die Beschränkung auf nicht-isomorphe Technologien entsteht.

$n\backslash m$	2	3	4	5	6
2	2	5	17	73	398
3	2	10	111	2 467	86 787
4	3	24	762	76 044	15 688 744
5	3	42	4 095	1 876 255	2 270 743 529
6	4	83	19 941	39 096 565	274 382 326 290
7	4	132	84 825	703 593 825	28 457 281 936 435
8	5	222	329 214	11 169 676 185	2 586 055 570 098 800
9	5	335	1 168 740	158 855 852 180	209 183 155 674 562 575

Tabelle 5.2: Anzahlen nicht-isomorpher $n \times m$–Technologien.

Struktur-Isomorphie von Technologien

In diesem Abschnitt wird in Anlehnung an die Pläne auch für die Technologien eine Erweiterung der einfachen Isomorphie („$\cong$“) gegeben. Da eine Technologie TR durch Transposition nicht notwendig wieder in eine Technologie überführt wird, kann die Äquivalenz von Plänen nicht für Technologien TR angewandt werden. Man kann aber die Technologien als „strukturell gleichwertig“ identifizieren, in denen die technologischen Reihenfolgen aller Aufträge vollständig umgekehrt sind. Die Technologie, die sich durch Umkehrung aller technologischen Reihenfolgen in TR ergibt, heißt *Umkehrtechnologie* $\overline{TR}$ von TR.

Definition 5.2.8 Zwei Technologien TR^1 und TR^2 heißen *struktur-isomorph*, geschrieben $TR^1 \cong_S TR^2$, wenn $TR^1 \cong TR^2$ oder $TR^1 \cong \overline{TR^2}$ gilt.

Für die Anzahl $S_{TR}(n,m)$ der nicht-struktur-isomorphen Technologien TR läßt sich analog zu Satz 5.2.7 der Cauchy-Frobenius-Hilfssatz anwenden, wobei dann die Gruppe $S_m \times \mathbb{Z}_2$ anstelle von S_m zugrunde gelegt werden muß, da jeweils die Operation der Umkehrung aller technologischen Reihenfolgen hinzukommt.[7] Das Abzählen der Technologien TR, die unter einer Operation $(\sigma, u) \in S_m \times \mathbb{Z}_2$ fix bleiben, ist jedoch weitaus schwieriger als im Fall ohne mögliche Umkehrtechnologien, daher ergibt sich eine kompliziertere Formel als (5.7), auf deren Darstellung hier verzichtet wird. In Tabelle 5.3 wird analog zu Tabelle 5.2 eine Übersicht über die Werte $S_{TR}(n,m)$ gegeben. Beim Vergleich mit Tabelle 5.1 (Seite 57) kann wiederum die deutliche Anzahlreduzierung festgestellt werden: Während z. B. die Gesamtanzahl der 7×6–Technologien bei $1,003 \times 10^{20}$ liegt, ist $I^*_{TR}(7,6) \approx 1,423 \times 10^{13}$.

[7] Die Gruppe $\mathbb{Z}_2$ ist die *zyklische Gruppe* der Ordnung 2; und die Gruppe $S_m \times \mathbb{Z}_2$ ist das *direkte Produkt* aus den Gruppen S_m und $\mathbb{Z}_2$.

$n\backslash m$	2	3	4	5	6
2	2	4	13	45	230
3	2	7	67	1 269	43 767
4	3	16	434	38 356	7 854 456
5	3	26	2 175	939 395	1 135 495 745
6	4	50	10 385	19 556 801	137 193 369 114
7	4	76	43 353	351 827 297	14 228 666 657 843
8	5	126	167 102	5 585 002 649	1 293 028 139 377 488
9	5	185	589 648	79 428 476 802	104 591 581 641 412 531

Tabelle 5.3: Anzahlen nicht-struktur-isomorpher $n \times m$–Technologien.

Mit Hilfe der Datenbank *On-Line Encyclopedia of Integer Sequences*, die von SLOANE gepflegt wird und sich im World-Wide-Web unter der Adresse

`http://www.research.att.com/~njas/sequences/`

befindet, läßt sich ein interessanter Zusammenhang zwischen der Anzahl sogenannter Armbänder und den Werten aus Tabelle 5.2 und 5.3 entdecken. Auf diese Korrespondenz wird im folgenden näher eingegangen.

Definition 5.2.9 Ein *k-Armband (k-bracelet)* ist eine Äquivalenzklasse von zyklischen Null-Eins-Folgen der Länge k unter den Operationen der Drehung und Spiegelung.[8]

Die Anzahl der k-Armbänder entspricht der Anzahl der Möglichkeiten, schwarze und weiße Perlen auf ein Armband mit insgesamt k Perlen aufzuziehen. Dabei gehören diejenigen Darstellungen der Armbänder einer Äquivalenzklasse an, die durch Drehung und Achsenspiegelung ineinander überführt werden können. So sind zum Beispiel die zyklischen Folgen 001011 und 001101 gleichwertig (siehe Abbildung 5.2).

Satz 5.2.10 *Die Anzahl der nicht-struktur-isomorphen $n \times 3$–Technologien TR ist gleich der Anzahl der $(n+6)$–Armbänder mit n weißen und 6 schwarzen Perlen.*

Beweis: Es sei $f : S_3 \to \{0, 1, \ldots, 5\}$ eine Funktion mit

$$\begin{array}{l} (1,2,3) \mapsto 0,\ (1,3,2) \mapsto 1,\ (2,3,1) \mapsto 2, \\ (3,2,1) \mapsto 3,\ (3,1,2) \mapsto 4,\ (2,1,3) \mapsto 5, \end{array} \tag{5.8}$$

[8]Ein Armband (bracelet) ist in der Literatur häufig auch unter dem Namen *Perlenkette (necklace)* bekannt (siehe z. B. `http://sue.csc.uvic.ca/~cos/inf/neck/NecklaceInfo.html`).

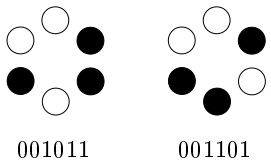

Abbildung 5.2: Ein 6-Armband in zwei verschiedenen Darstellungen.

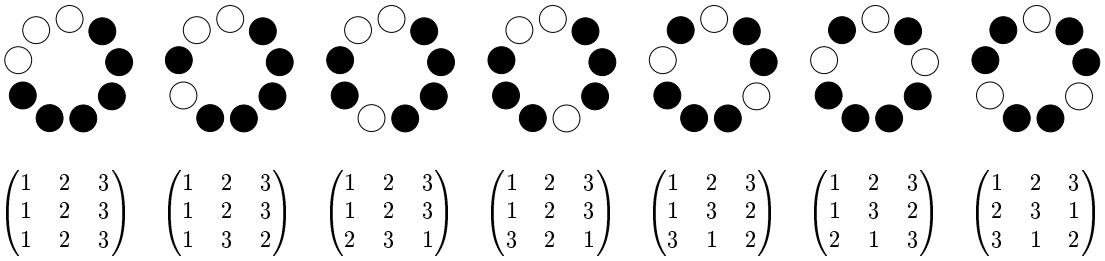

Abbildung 5.3: Eine Bijektion zwischen 9-Armbändern und den Repräsentanten der Struktur-Isomorphie-Klassen von 3×3–Technologien.

wobei S_3 die symmetrische Gruppe der Ordnung 3 ist. Jeder Zeilenvektor TR_i einer $n \times 3$–Technologie TR wird als Permutation der Zahlen $\{1, 2, 3\}$ aufgefaßt. Eine Technologie TR heißt *f-geordnet*, wenn $f(TR_i) \leq f(TR_{i+1})$ für alle $i = 1, \ldots, n-1$ gilt. Offensichtlich kann jede Technologie durch eine geeignete Zeilenpermutation in eine f-geordnete Technologie TR überführt werden. Es läßt sich nun eine Bijektion zwischen den f-geordneten Technologien TR des Formats $n \times 3$ und den Darstellungen der $(n+6)$–Armbänder formulieren.

Eine f-geordnete Technologie TR wird auf eine Darstellung eines Armbandes folgendermaßen abgebildet: Für $i = 1, 2, \ldots, n$ ist die Anzahl der schwarzen Perlen, die sich zwischen der obersten Perle und der i-ten weißen Perle im mathematisch positiven Sinne befinden, gleich $f(TR_i)$, wobei die Funktion f durch (5.8) gegeben ist (siehe z. B. Abbildung 5.3 im Fall $n = 3$). Es ist dabei zu beachten, daß jeweils n der Darstellungen der $(n+6)$–Armbänder nicht unterschieden werden. Und zwar handelt es sich jeweils um diejenigen Darstellungen, für die bezüglich der gerade eingeführten Abbildung kein Urbild definiert ist, da sich zwischen der obersten Perle und einer i-ten Perle > 5 schwarze Perlen befinden (siehe z. B. Abbildung 5.4 im Fall $n = 3$).

Zu jeder der unterschiedenen Darstellungen der $(n+6)$–Armbänder gibt es genau ein Urbild, daher handelt es sich insgesamt bei dieser Abbildung um eine eineindeutige Zuordnung.

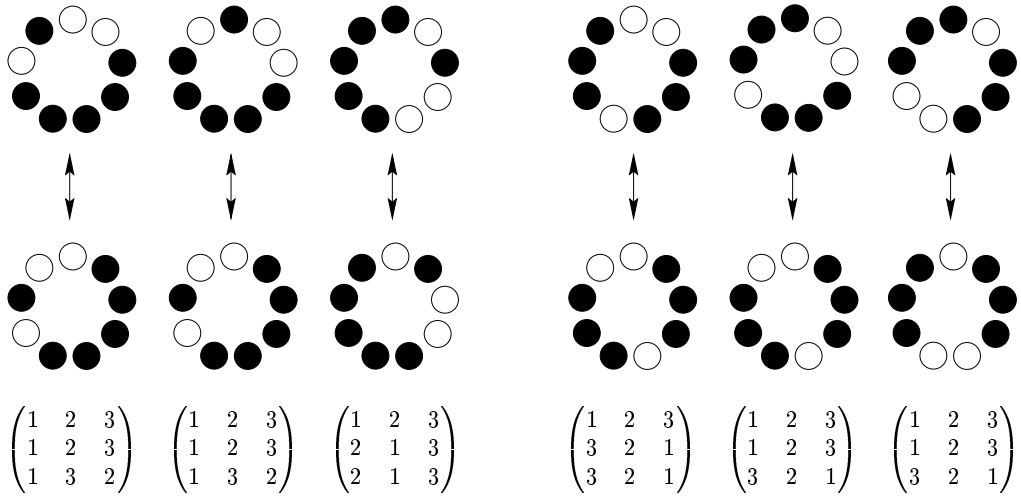

Abbildung 5.4: Zwei Beispiele für die Identifizierung von jeweils 3 Darstellungen der 9-Armbändern (es werden jeweils die obere und untere Darstellung nicht unterschieden).

Da von den $n+6$ Positionen, in die ein $(n+6)$–Armband gedreht werden kann, n nicht unterschieden werden, ist die Gruppe der Drehspiegelungen, die auf dieser Menge von Darstellungen der $(n+6)$–Armbänder operiert, die Diedergruppe[9] D_{12}. Die Gruppe der Operationen (Spaltenpermutationen und Umkehrung), die auf den f-geordneten $n \times 3$–Technologien definiert sind, ist $S_3 \times \mathbb{Z}_2$. Es kann gezeigt werden, daß die Gruppen D_{12} und $S_3 \times \mathbb{Z}_2$ isomorph sind, daher ist die Anzahl der Struktur-Isomorphieklassen von $n \times 3$–Technologien tatsächlich gleich der Anzahl der verschiedenen $(n+6)$–Armbänder mit n weißen und sechs schwarzen Perlen. □

Die gesuchte Anzahl der $(n+6)$–Armbänder mit n weißen und sechs schwarzen Perlen läßt sich auch direkt anhand des Cauchy-Frobenius-Hilfssatzes ermitteln, indem gezählt wird, wieviele Darstellungen der $(n+6)$–Armbänder unter den Operationen der Diedergruppe $D_{2(n+6)}$ fix bleiben (in diesem Fall werden alle $n+6$ „Drehpositionen“ unterschieden).

Der anschließende Satz zeigt, daß für kein $m > 3$ eine Satz 5.2.10 entsprechende Beziehung existiert.

Satz 5.2.11 *Die Anzahl der nicht-struktur-isomorphen $n \times m$–Technologien TR (n variabel) entspricht für kein festes $m > 3$ der Anzahl der $(n+m!)$–Armbänder mit*

[9] Die *Diedergruppe* D_{2n} ist eine Gruppe der Ordnung $2n$, die als Menge von Drehspiegelungen des regelmäßigen n-Ecks aufgefaßt werden kann.

n weißen und $m!$ schwarzen Perlen.

Beweis: Wie im Beweis zu Satz 5.2.10 wird mittels einer geeigneten Funktion $f : S_m \to \{1, 2, \ldots, m!\}$ eine Bijektion zwischen den f-geordneten $n \times m$–Technologien TR und den $(n + m!)$–Armbändern mit n weißen und $m!$ schwarzen Perlen hergestellt, wobei bei den Darstellungen der $(n + m!)$–Armbänder wieder jeweils n Positionen nicht unterschieden werden. Die Gruppe der Drehspiegelungen, die auf diesen Armbändern operiert, ist die Diedergruppe $D_{2(m!)}$. Die Gruppe, die auf den f-geordneten $n \times m$–Technologien TR operiert, ist $S_m \times \mathbb{Z}_2$. Jede Diedergruppe $D_{2(m!)}$ hat ein Element der Ordnung $m!$. Aber kein Element von $S_m \times \mathbb{Z}_2$ hat für $m > 3$ ein Element der Ordnung $m!$, daher sind die Gruppen $D_{2(m!)}$ und $S_m \times \mathbb{Z}_2$ für kein $m > 3$ isomorph, und die im Satz genannten Anzahlen (m fest, n variabel) stimmen nicht überein. □

5.3 Ein neuer Enumerationsalgorithmus

In diesem Abschnitt wird ein neuer Algorithmus zur Enumeration aller $n \times m$–Pläne für fest vorgegebene Werte n und m vorgestellt. Die gesamte Plan-Enumeration gliedert sich zunächst in zwei Teilprozeduren: Lexikographische Technologie-Erzeugung und Technologie-Minimalitätstest. Diese Teilprozeduren werden im anschließenden Unterabschnitt behandelt, bevor die gesamte Plan-Enumeration Mittelpunkt des darauf folgenden Unterabschnitts ist. Einige Ergebnisse und Bestandteile der benutzten Verfahren sind bereits in [11] enthalten. In der vorliegenden Arbeit sind die zur Plan-Enumeration benötigten Algorithmen im Gegensatz zu [11] ausführlich anhand von Pseudocode-Programmen beschrieben.

Erzeugung von Permutationen

Ein Isomorphismus (ϱ, σ) oder Struktur-Isomorphismus (ϱ, σ, u), der eine Technologie TR^1 auf eine Technologie TR^2 abbildet, definiert gleichzeitig eine Bijektion zwischen den beiden Mengen von Plänen, deren Technologien entweder TR^1 oder TR^2 entsprechen. Diese Bijektion bildet einen Plan A mit TR^1 jeweils auf einen Plan B mit TR^2 ab. Daher ist es zur Enumeration nicht-isomorpher Pläne zunächst ausreichend, ausschließlich nicht-isomorphe bzw. nicht-struktur-isomorphe Technologien zu generieren.

Mit Hilfe der anschließend beschriebenen Algorithmen wird ein Vertretersystem für nicht-isomorphe Technologien auf der Grundlage von Auswahlfunktion (5.1) erzeugt. Zuerst werden Technologien TR in lexikographischer Reihenfolge ausgegeben (Algorithmus 5.3.1). Dann erfolgt für alle so konstruierten Technologien TR jeweils

ein Minimalitätstest (Algorithmus 5.3.3), der entscheidet, ob es sich bei der gegebenen Technologie um das lexikographische Minimum ihrer (Struktur-)Isomorphieklasse handelt. Offensichtlich ist das lexikographische Minimum der Technologien TR stets reduziert, daher reicht es bei der *Lexikographischen Technologie-Erzeugung* aus, ausschließlich reduzierte Technologien zu generieren.

Algorithmus 5.3.1 *Lexikographische Technologie-Erzeugung*

Eingabe: Parameter n und m.
Ausgabe: Alle reduzierten $n \times m$–Technologien in lexikographisch aufsteigender Reihenfolge.

1. Setze $TR := TR^0$, wobei TR^0 die lexikographisch kleinste $n \times m$–Technologie ist, also
$$TR^0 = \begin{pmatrix} 1 & 2 & 3 & \cdots & m \\ 1 & 2 & 3 & \cdots & m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2 & 3 & \cdots & m \end{pmatrix}.$$

2. Bestimme den größten Zeilenindex i von TR, so daß der Zeilenvektor TR_i gemäß Schritt 3 noch lexikographisch erhöht werden kann. Falls kein solches i existiert, Stop!

3. Erhöhe den Zeilenvektor $TR_i = (t_{i1}, t_{i2}, \ldots, t_{im})$ wie folgt:

   (a) Bestimme den größten Spaltenindex j mit $t_{i,j-1} < t_{ij}$, d. h. es gilt $t_{ij} > t_{i,j+1} > \cdots > t_{i,m-1} > t_{im}$ oder $j = m$.

   (b) Bestimme den größten Spaltenindex $k \geq j$ mit $t_{i,j-1} < t_{ik}$, d. h. t_{ik} ist die kleinste Zahl unter den $t_{ij}, \ldots, t_{im}$, die größer als t_{ij-1} ist.

   (c) Setze $TR_{i'} := (t_{i1}, \ldots, t_{i,j-2}, t_{ik}, t_{im}, \ldots, t_{i,k+1}, t_{i,j-1}, t_{i,k-1}, \ldots, t_{ij})$, d. h. die Reihenfolge von $t_{ij}, t_{i,j+1}, \ldots, t_{im}$ wird umgekehrt und $t_{i,j-1}$ wird mit t_{ik} vertauscht.

4. Setze $TR' := (TR_1, TR_2, \ldots, TR_{i-1}, TR_{i'}, TR_{i'}, \ldots, TR_{i'})^T$ und gib TR' aus.

5. Setze $TR := TR'$ und gehe zu Schritt 2.

Satz 5.3.2 *Die Prozedur der Erzeugung der lexikographisch nachfolgenden Technologie zu einer gegebenen reduzierten $n \times m$–Technologie kann in der Zeit $O(nm)$ ausgeführt werden.*

Beweis: Die Korrektheit von Algorithmus 5.3.1 ist offensichtlich. Die Schritte der Erhöhung einer Zeile (Schritte 3a-3c) benötigen jeweils $O(m)$ Zeit. Da maximal $n-1$ Zeilen auf die beschriebene Weise lexikographisch erhöht werden müssen, wird für den Schritt von einer reduzierten Technologie TR zur lexikographisch folgenden im schlechtesten Fall $O(nm)$ Zeit benötigt. □

Für jede Technologie TR', die bei der *Lexikographischen Technologie-Erzeugung* ausgegeben wird, kann zur Entscheidung, ob TR' ein Vertreter ihrer Isomorphieklasse ist, der folgende Algorithmus angewandt werden.

Algorithmus 5.3.3 *Technologie-Minimalitätstest*

Eingabe: Eine beliebige $n \times m$–Technologie TR^*.
Ausgabe: Entscheidung, ob die gegebene Technologie TR^* das lexikographische Minimum ihrer Isomorphieklasse ist.

1. Setze $TR := TR^*$
2. Für alle $i := 1, \ldots, n$:
   (a) Setze $TR' := (TR_i, TR_2, \ldots, TR_1, \ldots, TR_n)^T$, d. h. vertausche den ersten mit dem i-ten Zeilenvektor von TR.
   (b) Wende die Spaltenpermutation auf TR' an, die den ersten Zeilenvektor von TR' in $(1, 2, 3, \ldots, m)$ überführt und speichere das Ergebnis in TR''.
   (c) Wende die Zeilenpermutation auf TR'' an, die TR'' in die reduzierte Form TR''' überführt, wobei die erste Zeile von TR'' fix bleibt.
   (d) Wenn $TR''' <_{lex} TR^*$ gilt:
   TR^* ist nicht das lexikographische Minimum, Stop!
3. TR^* ist das lexikographische Minimum, gib TR^* aus, Stop!

Algorithmus 5.3.3 kann neben der Entscheidung über das lexikographische Minimum bezüglich Isomorphie auch zur entsprechenden Entscheidung bezüglich Struktur-Isomorphie benutzt werden. Zu diesem Zweck muß jeweils vor dem Abschluß (Schritt 3) die Beziehung $TR := \overline{TR^*}$ gesetzt und wieder zum Anfang von Schritt 2 gesprungen werden. Da die Umkehrtechnologie $\overline{TR}$ einer $n \times m$–Technologie in $O(nm)$ Zeit berechnet werden kann, beeinträchtigt diese Komplexität nicht den gesamten Zeitaufwand für den Test auf Minimalität bezüglich $<_{lex}$:

Satz 5.3.4 *Mit Hilfe von Algorithmus 5.3.3 kann in der Zeit $O(n^2 m \log n)$ festgestellt werden, ob es sich bei einer gegebenen $n \times m$–Technologie TR um das lexikographische Minimum ihrer Isomorphieklasse handelt.*

Beweis: Die Korrektheit von Algorithmus 5.3.3 ist klar. Da bei der Implementation des Algorithmus zur Realisierung der Spaltenpermutation in Schritt 2b nur Zeiger umgesetzt werden müssen, genügt dafür $O(m)$ Zeit. Die Sortierung der Zeilen in Schritt 2c, durch die wieder eine reduzierte Technologie erzeugt wird, benötigt $O(nm \log n)$ Zeit. Der lexikographische Vergleich (Schritt 2d) kann in der Zeit $O(nm)$ ausgeführt werden. Da man schließlich alle Operationen in Schritt 2 maximal n-mal durchführen muß, folgt als gesamte Zeitkomplexität die Schranke $O(n^2 m \log n)$. □

Mit Hilfe der *Lexikographischen Technologie-Erzeugung* und dem *Technologie-Minimalitätstest* ist es möglich, die theoretisch erzielten Ergebnisse über die Anzahl der Isomorphie- und Struktur-Isomorphieklassen aus Abschnitt 5.2 zu verifizieren. Tatsächlich stammen die Werte aus Tabelle 5.2 und Tabelle 5.3 ursprünglich aus Berechnungen mit Hilfe der gerade beschriebenen Algorithmen. Dabei können die gewünschten Anzahlen z. B. für $n \leq 5$ und $m = 6$ auf einer schnellen Workstation innerhalb von wenigen Sekunden berechnet und ausgegeben werden.

Modifiziertes Einfügeverfahren

Bei dem von Bräsel und M. Kleinau [13, 14] entwickelten Einfügeverfahren zur Enumeration aller $n \times m$–Pläne werden für feste Werte n und m die Operationen o_{ij} sukzessive in partielle Ablaufgraphen bzw. sogenannte Teilpläne eingefügt.

Definition 5.3.5 Ein *Teilplan* $A = (a_{ij})$ ist ein Plan, in dem einige Zellen leer sind, und bei dem zu jedem vorhandenen Eintrag $a_{ij} > 1$ der Wert $a_{ij} - 1$ als Eintrag in der Zeile i oder Spalte j existiert.

In unvollständig besetzten Matrizen kennzeichnen werden die leeren Zellen ohne Einträge stets mit „$\cdot$“ gekennzeichnet.

Beispiel 5.3.6 Es seien die Matrizen

$$A = \begin{pmatrix} 3 & \cdot & 2 \\ \cdot & 2 & 1 \\ 4 & \cdot & 3 \end{pmatrix} \qquad \text{und} \qquad B = \begin{pmatrix} \cdot & 5 & 3 \\ 3 & \cdot & 2 \\ 2 & 1 & \cdot \end{pmatrix}$$

gegeben. Die Matrix A ist ein Teilplan, B ist es nicht, da für den Eintrag $b_{1,2} = 5$ kein Eintrag $b_{1j} = 4$ oder $b_{i2} = 4$ in der ersten Zeile oder zweiten Spalte existiert.

Das anschließend vorgestellte modifizierte Einfügeverfahren baut auf der im vorangegangenen Abschnitt beschriebenen Konstruktion nicht-isomorpher Technologien TR auf. Das heißt, die technologischen Reihenfolgen sind bereits vorgegeben und

dazu werden jeweils vollständige Pläne erzeugt, indem die Operationen o_{ij} so in die partiellen organisatorischen Reihenfolgen eingefügt werden, daß keine Zyklen entstehen. Der Ausdruck $TR(A)$ bezeichne im folgenden stets die Technologie eines Plans A, und in der Variablen $Aut(A)$ wird im Laufe des Algorithmus die Anzahl der Automorphismen des Plans A gespeichert.

Algorithmus 5.3.7 *Plan-Enumeration*

Eingabe: Parameter n und m.
Ausgabe: Vertretersystem für die Isomorphieklassen der $n \times m$–Pläne, Gesamtanzahl $P_{n,m}$ aller $n \times m$–Pläne.

1. Setze $P_{n,m} := 0$.
2. Erzeuge ein Vertretersystem S^* für die Isomorphieklassen der $n \times m$–Technologien TR auf der Grundlage der lexikographischen Minima mittels Kombination aus Algorithmus 5.3.1 und 5.3.3.
3. Für alle Vertreter TR aus S^*, die in Schritt 2 erzeugt wurden:
   (a) Initialisiere eine $n \times m$–Matrix A, die ausschließlich leere Zellen enthält.
   (b) Für alle Operationen $o_{ij},\ i := 1, \ldots, n,\ j := 1, \ldots, m$:
      i. Aktualisiere die Matrix A anhand folgender Prozedur: Füge die Operation o_{ij} gemäß der Technologie TR in die Organisation der Maschine M_j als direkten Nachfolger einer in dieser Organisation bereits existierenden Operation ein, oder falls es noch keine derartige Operation gibt, füge o_{ij} als Quelle der Organisation von M_j ein.
      ii. Prüfe die Zulässigkeit von A, d. h. teste, ob A ein $n \times m$–Teilplan ist.
4. Für alle (vollständigen) Pläne A, die in Schritt 3 rekursiv erzeugt werden:
   (a) Setze $Aut(A) := 0$.
   (b) Für alle nichttrivialen Automorphismen (ϱ, σ) von $TR(A)$:
      i. Vergleiche A lexikographisch mit dem $n \times m$–Plan A', der gemäß (ϱ, σ) zu A isomorph ist (vgl. Algorithmus 5.1.4).
      ii. Wenn (ϱ, σ) ein Automorphismus von A ist:
         Setze $Aut(A) := Aut(A) + 1$.
   (c) Wenn für alle Pläne A' aus Schritt 4b die Beziehung $A \leq_{lex} A'$ gilt:
      i. Gib den Plan A aus.
      ii. Setze $P_{n,m} := P_{n,m} + (n!m!)/Aut(A)$.

5. Gib $P_{n,m}$ aus.

Es folgen einige kurze Erläuterungen zu einzelnen Teilen dieser *Plan-Enumeration*. Weitere Details zur Implementation sind in [11] beschrieben.

Die Einfügungen der Operationen in Schritt 3b wird in einer festgelegten Reihenfolge durchgeführt, die in bestimmten Fällen eventuell noch optimiert werden kann. Der gesamte Schritt 3 erfolgt durch Backtracking, um in systematischer Weise alle möglichen $n \times m$–Pläne erzeugen zu können, die zur gegebenen Technologie TR gehören. Bei jeder Aktualisierung der Matrix A wird versucht, die Ränge der Operationen zu ermitteln (topologisches Sortieren im assoziierten Digraphen). Wenn keine topologische Sortierung existiert, handelt es sich nicht um einen $n \times m$–Teilplan, d. h. die aktuelle Matrix A ist nicht zulässig, und die Weiterführung der Einfügungen von Operationen in A wird abgebrochen.

Das bei der *Plan-Enumeration* erzeugte Vertretersystem für die Isomorphieklassen der $n \times m$–Pläne basiert nicht auf der Auswahlfunktion (5.1), die jeweils das lexikographische Minimum als Repräsentant einer Isomorphieklasse auswählt. Da man die Informationen ausnutzen will, die bereits aus der lexikographischen Technologie-Erzeugung und dem Minimalitätstest in Schritt 2 stammen, wird für das zu erzeugende Vertretersystem der $n \times m$–Pläne die folgende, bezüglich (5.1) leicht modifizierte Auswahlfunktion verwendet:

Definition 5.3.8 Für zwei $n \times m$–Pläne A und B gelte $A \ll_{lex} B$, genau dann wenn $TR(A) <_{lex} TR(B)$ ist oder $TR(A) =_{lex} TR(B)$ und $A <_{lex} B$ gilt. Es sei $\mathcal{P}_{n,m}$ die Menge aller $n \times m$–Pläne, und $\mathcal{P}_1, \ldots, \mathcal{P}_r$ seien ihre Isomorphieklassen, also gilt $\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_r = \mathcal{P}_{n,m}$ und $\mathcal{P}_i \cap \mathcal{P}_j = \varnothing$ für $i \neq j$. Weiterhin sei $\min_{\ll}(\mathcal{P}_i)$ das Minimum der Pläne aus $\mathcal{P}_i$ bezüglich $\ll$. Dann ist die Funktion f mit

$$\begin{aligned} f : \{1, \ldots, r\} &\rightarrow \mathcal{P}_{n,m}, \\ i &\mapsto f(i) = \min_{\ll}(\mathcal{P}_i) \end{aligned} \tag{5.9}$$

die Auswahlfunktion des Vertretersystems der $n \times m$–Pläne, die bei der *Plan-Enumeration* zugrunde gelegt wird.

In Schritt 3 von Algorithmus 5.3.7 werden ausschließlich solche Pläne erzeugt, die für zwei Aufträge mit gleicher technologischer Reihenfolge bereits lexikographisch sortiert sind. Das heißt, wenn in einem Plan zwei Aufträge J_i und J_k mit $i < k$ die gleiche technologische Reihenfolge besitzen, wird die Operation o_{i1} vor o_{k1} ausgeführt. Auf diese Weise müssen in Schritt 4 von Algorithmus aufgrund der zugrunde gelegten Auswahlfunktion (5.9) nur dann lexikographische Vergleiche für einen Plan A ausgeführt werden, wenn $TR(A)$ nichttriviale Automorphismen besitzt.

Die Variable $Aut(A)$ in Schritt 4 dient zur Berechnung der Anzahl der Automorphismen von A. Der Ausdruck $P_{n,m} := P_{n,m} + (n!m!)/Aut(A)$ in Schritt 4(c)ii läßt sich wiederum gruppentheoretisch deuten (vgl. Seite 60): Die Zeilen- und Spaltenpermutationen (ϱ, σ) werden als Elemente der Gruppe $S_n \times S_m$ aufgefaßt, die auf der Menge der $n \times m$–Pläne operiert. Die Bahn eines Plans A entspricht dann der Isomorphieklasse, der A angehört, und der Stabilisator des Plans A ist die Menge der Automorphismen von A. Es ist bekannt, daß die Anzahl der Elemente der Isomorphieklasse, der A angehört, gleich dem Verhältnis von der Anzahl der Gruppenelemente zur Anzahl der Automorphismen von A ist. Das heißt, die wiederholte Ausführung von Schritt 4(c)ii bei der *Plan-Enumeration* bedeutet, daß für alle Isomorphieklassen die Anzahl der zugehörigen Pläne jeweils zur Variablen $\mathcal{P}_{n,m}$ addiert wird. Auf diese Weise wird am Ende der Prozedur neben der Anzahl der Isomorphieklassen auch die Gesamtanzahl aller $n \times m$–Pläne berechnet, obwohl nicht alle Pläne tatsächlich erzeugt wurden.

Durch Anpassung der lexikographischen Tests in Schritt 4 ist es ebenso möglich, neben der Anzahl der Isomorphieklassen auch die Anzahl der Plan-Klassen bezüglich Äquivalenz und Struktur-Äquivalenz mittels Erzeugung entsprechender Vertretersysteme zu bestimmen. Eine Übersicht über die anhand dieses Algorithmus erzielten Werte gibt der folgende Abschnitt.

5.4 Numerische Auswertungen

In diesem Abschnitt werden die Resultate für die verschiedenen Anzahlen der $n \times m$–Pläne zusammengestellt, verglichen und kommentiert. Im folgenden sei stets $P_{n,m}$ die Gesamtanzahl aller $n \times m$–Pläne, $I_{n,m}$ die Anzahl der nicht-isomorphen, $A_{n,m}$ die Anzahl der nicht-äquivalenten und $S_{n,m}$ die Anzahl der nicht-struktur-äquivalenten $n \times m$–Pläne.

In Tabelle 5.4 fällt auf, daß für $n \neq m$ stets $I_{n,m} = A_{n,m}$ gilt, was auf die Gleichbedeutung der Begriffe Isomorphie und Äquivalenz im nicht-quadratischen Fall hinweist. Weiterhin gilt $A_{n,m}/S_{n,m} \leq 2$, da die Anzahl der nicht-äquivalenten Pläne höchstens um die Hälfte reduziert werden kann, wenn zu jedem Plan A sein Umkehrplan $\overline{A}$ als strukturell gleichwertig eingestuft wird. Da allerdings die Wahrscheinlichkeit, daß ein zufällig ausgewählter Plan A zu $\overline{A}$ äquivalent ist, mit wachsenden n und m abnimmt, kann man folgendes feststellen.

Beobachtung 5.4.1 *Für* $(n + m) \to \infty$ *gilt*

$$\frac{A_{n,m}}{S_{n,m}} = 2 - o(1). \tag{5.10}$$

n	m	$P_{n,m}$	$I_{n,m}$	$A_{n,m}$	$S_{n,m}$
2	2	14	4	3	3
2	3	204	17	17	12
3	3	19 164	533	280	147
2	4	5 016	106	106	68
3	4	3 733 056	25 924	25 924	13 100
4	4	6 941 592 576	12 051 574	6 028 059	3 017 369
2	5	185 520	773	773	422
3	5	1 288 391 040	1 789 432	1 789 432	895 388
4	5	26 549 943 275 520	9 218 730 304	9 218 730 304	4 609 489 912
2	6	9 595 440	6 671	6 671	3 495
3	6	712 770 186 240	164 993 112	164 993 112	82 507 654
2	7	659 846 880	65 461	65 461	33 193
3	7	589 563 294 888 960	19 496 140 704	19 496 140 704	9 748 141 078

Tabelle 5.4: Anzahlen der $n \times m$–Pläne im Vergleich.

$n \setminus m$	2	3	4	5	6	7
2	21.4286	5.9113	1.3557	0.2275	0.0364	0.0050
3		0.7671	0.3509	0.0695	0.0116	0.0017
4			0.0435	0.0174	??	??

Tabelle 5.5: Verhältnisse der Anzahlen nicht-struktur-äquivalenter $n \times m$–Pläne zu den jeweiligen Gesamtanzahlen (in %).

Diese Beobachtung wird durch die Werte in Tabelle 5.4 bestätigt.

Ziel der Einführung der Struktur-Äquivalenz ist es, die Plan-Untersuchungen, also die Untersuchungen von Lösungen für Shop-Scheduling-Probleme, auf solche Pläne zu beschränken, die sich in ihrer Grundstruktur bezüglich der Wege in den zugehörigen Ablaufgraphen unterscheiden. Tabelle 5.5 zeigt, daß die Anzahl nicht-struktur-äquivalenter $n \times m$–Pläne deutlich kleiner als die Gesamtanzahl der Pläne des gleichen Formats ist, d. h. es ergibt sich damit wie gewünscht eine erhebliche Reduzierung der zu betrachtenden Pläne.

Die zur Enumeration benötigten Algorithmen wurden in C++ implementiert und auf einem Pentium-PC-133-Mhz-Rechner getestet. Das gesamte Programm ist so angelegt, daß für ein gegebenes Format $n \times m$ in *einem* Durchlauf *alle* Zahlen, also die Gesamtanzahl $P_{n,m}$ sowie die Anzahlen $I_{n,m}, A_{n,m}$ und $S_{n,m}$ für die drei Plan-Äquivalenzklassen bestimmt werden. Die dazu benötigte CPU-Zeit zur

n	m	CPU-Zeit in Sek.	# versuchter Komplettierungen	# erzeugter $n \times m$–Pläne	$P_{n,m}$
2	2	0.01	6	4	14
2	3	0.01	33	15	204
3	3	0.04	892	384	19 164
2	4	0.01	226	84	5 016
3	4	0.66	38 704	15 638	3 733 056
4	4	838.15	14 184 294	6 872 356	6 941 592 576
2	5	0.03	1 546	491	185 520
3	5	35.73	2 430 856	923 073	1 288 391 040
4	5	1118.57	10 247 426 194	4 656 870 459	26 549 943 275 520
2	6	0.23	13 908	3 888	9 595 440
3	6	3617.67	235 692 637	83 310 542	712 770 186 240
2	7	2.08	137 532	34 709	659 846 880
3	7	3537.27	29 593 003 309	9 756 803 163	589 563 294 888 960

Tabelle 5.6: Statistische Werte für die Berechnung der Plan-Anzahlen.

vollständigen Enumeration ist in Tabelle 5.6 enthalten. Weiterhin zeigt diese Tabelle die Anzahl der versuchten Komplettierungen von Teilplänen gemäß Schritt 3 in Algorithmus 5.3.7, sowie die Anzahl der darunter erfolgreichen Komplettierungen, welche der Anzahl der im Algorithmus tatsächlich erzeugten Pläne entspricht. Zu Vergleichszwecken ist nochmals zusätzlich die Gesamtanzahl $P_{n,m}$ aufgeführt.

Während die benötigte CPU-Zeit für kleinere Formate noch unerheblich ist, wächst mit steigenden Werten n und m enorm. Mit dem vorliegenden Programm ist die Berechnung der Anzahlen für das Format 5×5 in vernünftiger Zeit nicht mehr möglich. Die Anzahlbestimmung für die quadratischen Formate ($n = m$) ist allerdings auch besonders aufwendig, da bei den Tests auf Äquivalenz- und Struktur-Äquivalenz jeweils zusätzlich die transponierten Pläne betrachtet werden müssen.

5.5 Komplexität der Plan-Enumeration

Die von Cook [27] und Karp [51] eingeführte Komplexitätstheorie, die häufig zur Einschätzung der Schwierigkeit von kombinatorischen Optimierungsproblemen dient, wurde in Arbeiten von Valiant [99, 100] für Enumerationsprobleme erweitert. Viele der bekannten Entscheidungs- bzw. Optimierungsprobleme sind mit solche Enumerationsproblemen auf natürliche Weise verbunden. Zum Beispiel gehört

zum Problem der Bestimmung eines Hamiltonschen Kreises[10] in einem gegebenen Graphen G das Enumerationsproblem „Wieviele solcher Hamiltonschen Kreise gibt es für G?“

Neben der bereits in Abschnitt 2.3 erwähnten Klasse $\mathcal{P}$ von Entscheidungsproblemen gibt es im Bereich der Anzahlproblematik die Komplexitätsklasse $\#\mathcal{P}$, die aus allen nichtdeterministisch-polynomial-lösbaren Enumerationsproblemen besteht. Die Klasse $\#\mathcal{P}$ wurde von VALIANT definiert, um die zusätzliche Schwierigkeit bei der Enumeration widerzuspiegeln zu können. Analog zur Terminologie der $\mathcal{NP}$-vollständigen Probleme sind die *$\#\mathcal{P}$-vollständigen* Enumerationsprobleme (gesprochen: Anzahl-$\mathcal{P}$-vollständigen Probleme) die schwierigsten Probleme in der Komplexitätsklasse $\#\mathcal{P}$. Genauer gesagt, wird ein Enumerationsproblem Π als *$\#\mathcal{P}$-vollständig* bezeichnet, wenn $\Pi \in \#\mathcal{P}$ ist und alle Probleme $\Pi' \in \#\mathcal{P}$ auf Π polynomial reduzierbar sind (siehe GAREY und JOHNSON [36]).

Die Enumerationsprobleme, die zu $\mathcal{NP}$-vollständigen Entscheidungsproblemen gehören, sind offensichtlich $\mathcal{NP}$-schwer. Allgemein gilt, daß die Enumerationsprobleme mindestens so schwierig sind wie die zugehörigen Entscheidungsprobleme. Es gibt Enumerationsprobleme, die $\#\mathcal{P}$-vollständig sind, obwohl das zugrundeliegende Entscheidungsproblem polynomial lösbar ist. Beispielsweise ist das Problem „Gibt es in einem gegebenen bipartiten Graphen $G = (V, E)$ ein perfektes Matching?“ in der Zeit $O(|V|^{5/2})$ lösbar (siehe HOPCROFT und KARP [45]), während für das zugehörige Enumerationsproblem „Wieviele perfekte Matchings gibt es in einem bipartiten Graphen G?“ von VALIANT [99] die $\#\mathcal{P}$-Vollständigkeit nachgewiesen wurde.

In Abschnitt 4.3 zeigen Satz 4.3.2 und 4.3.3 bereits, daß das Problem der Enumeration aller $n \times m$–Pläne bzw. der Enumeration aller azyklischen Orientierungen des Hamming-Graphen $K_n \times K_m$ mit dem der Bestimmung des chromatischen Polynoms von $K_n \times K_m$ komplexitätstheoretisch korrespondiert. Satz 4.3.2 von STANLEY [92] stammt aus dem Jahr 1973. In [62] hat LINIAL 1986 gezeigt, daß das Problem der Enumeration der azyklischen Orientierungen eines Graphen G zu den schwierigsten Problemen in der Klasse $\#\mathcal{P}$ gehört:

Satz 5.5.1 [62] *Das Problem der Enumeration der azyklischen Orientierungen eines Graphen G ist $\#\mathcal{P}$-vollständig.*

Beweis: Die *Verbindung (join)* zweier Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ mit $V_1 \cap V_2 = \varnothing$ ist definiert als

$$G_1 + G_2 = (V_1 \cup V_2 \ , \ E_1 \cup E_2 \cup \{\{v_1, v_2\} : v_1 \in V_1, v_2 \in V_2\}). \tag{5.11}$$

[10] Ein *Hamiltonscher Kreis* eines Graphen G ist ein Kreis in G, der alle Knoten von G durchläuft.

Es sei $G = (V, E)$ ein Graph mit $|V| = p$. Für das chromatische Polynom der Verbindung $G + K_n$ gilt offensichtlich

$$\chi(G + K_n, \lambda) = \lambda(\lambda - 1) \cdots (\lambda - n + 1)\chi(G, \lambda - n). \tag{5.12}$$

Nach Satz 4.3.2 ist das Problem der Enumeration der azyklischen Orientierungen eines beliebigen Graphen G äquivalent zur Bestimmung des chromatischen Polynoms von G an der Stelle $\lambda = -1$. Das Problem der Bestimmung von $\chi(G, -1)$ läßt sich polynomial auf das Problem der Berechnung von $\chi(G, \lambda)$ transformieren:

Mit $\chi(G, -1)$ haben wir auch $\chi(G + K_n, -1)$ für $n = 1, \ldots, p$. Also kann mittels (5.12) der Ausdruck $\chi(G, -j)$ für $2 \leq j \leq p + 1$ berechnet werden. Damit ist auch das chromatische Polynom von G bestimmt, denn $\chi(G, \lambda)$ ist ein Polynom in λ vom Grad p.

Das Entscheidungsproblem „Hat ein Graph G eine zulässige λ-Färbung?“ ist für $\lambda \geq 3$ $\mathcal{NP}$-vollständig, siehe [36]. Daher ist das assoziierte Enumerationsproblem $\#\mathcal{P}$-vollständig und somit auch das Problem der Bestimmung des chromatischen Polynoms von G. □

Dieser Satz zeigt die Schwierigkeit der Bestimmung der Anzahl azyklischer Orientierungen eines beliebigen Graphen G. Im Zusammenhang mit der Enumeration von Plänen ist man allerdings ausschließlich an der Anzahl der azyklischen Orientierungen von Hamming-Graphen $K_n \times K_m$ interessiert.

Innerhalb einiger Graphenklassen kann das chromatische Polynom auf einfache Weise bestimmt werden. Bei diesen Graphen G existiert wegen Satz 4.3.2 daher auch ein effektiver und exakter Ausdruck für $\alpha(G)$. Es stellt sich nun also die Frage, ob die Bestimmung des chromatischen Polynoms von Hamming-Graphen $K_n \times K_m$ in polynomialer Zeit möglich ist. In diesem Zusammenhang ist ein kürzlich erschienenes Resultat von Rezaie [79] interessant:

Satz 5.5.2 [79] *Es sei P_n ein Weg mit n Knoten. Für alle $n, m \in \mathbb{N}$ gilt*

$$\chi(P_n \times K_m, \lambda) = \chi(K_m, \lambda)^n \left(\sum_{i=0}^{m} \frac{(-1)^i \binom{m}{i}}{\chi(K_i, \lambda)} \right)^{n-1}, \tag{5.13}$$

wobei $\chi(K_m, \lambda) = \lambda(\lambda - 1) \cdots (\lambda - m + 1)$ ist.

Zur rekursiven Berechnung von chromatischen Polynomen wird sehr häufig die folgende, bereits im Beweis zu Satz 4.3.2 verwandte Eigenschaft benutzt.

Hilfssatz 5.5.3 [77] *Es sei e eine Kante eines Graphen G, dann gilt*

$$\chi(G, \lambda) = \chi(G \backslash e, \lambda) - \chi(G/e, \lambda), \tag{5.14}$$

wobei $G\backslash e$ bzw. G/e der Graph ist, der aus G durch Löschen bzw. Kontraktion der Kante e entsteht.

Die Prozedur in [79] zur Bestimmung von $\chi(P_n \times K_m, \lambda)$ macht in jedem Schritt von der Existenz eines Knotens vom Grad 1 Gebrauch, um mit Hilfe der Anwendung von Eigenschaft (5.14) eine rekursive Beziehung herleiten zu können. Auch für allgemeine Bäume T_n mit n Knoten führt dieselbe Vorgehensweise zum Erfolg. Also erhält man für $\chi(T_n \times K_m, \lambda)$ ebenfalls die Formel (5.13).

Es stellt sich heraus, daß sich die rekursive Prozedur in [79] nicht auf Hamming-Graphen $K_n \times K_m$ übertragen läßt, denn in diesem Fall fehlen die in den Wegen P_n bzw. Bäumen T_n enthaltenen Knoten vom Grad 1. Das Problem der Bestimmung von $\chi(K_n \times K_m, \lambda)$ ist bis heute ungelöst. Weiterhin sind bisher weder geschlossene Formeln für $\chi(C_n \times K_m, \lambda)$ noch für $\chi(P_n \times P_m, \lambda)$ bekannt (siehe [23]).

Kapitel 6

Optimalitätskriterien für Pläne

Zu jedem Plan eines Shop-Scheduling-Problems kann der eindeutig zugeordnete semiaktive Schedule $C = (c_{ij})$ mit dem Zeitaufwand $O(nm)$ berechnet werden (siehe Seite 23). Daher ist die Bestimmung des semiaktiven Schedules zu einem gegebenen Plan ein polynomial lösbares Problem. Eine Vielzahl der Shop-Scheduling-Probleme ist jedoch $\mathcal{NP}$-schwer. Die Schwierigkeit dieser Probleme liegt also bereits in der Konstruktion eines optimalen Plans.

In diesem Kapitel werden verschiedene Kriterien behandelt, mit denen „ungünstige“ Pläne bei der Suche nach einem optimalen Plan ausgeschlossen werden können. Die hier vorgestellten Kriterien sind unabhängig bzw. nur bedingt abhängig von den gegebenen Bearbeitungszeiten p_{ij} für die Operationen o_{ij} des gegebenen Shop-Scheduling-Problems.

6.1 Potentiell-optimale Pläne

Bei jedem Job-Shop-Problem ist eine Technologie bereits fest vorgegeben. Zur Lösung des Problems muß eine günstige Organisation gefunden werden. In [2] hat ASHOUR für ein Job-Shop-Problem unter allen Organisationen erstmals zwischen zulässigen, potentiell-optimalen und optimalen Organisationen unterschieden. Bei der in [2] verwandten Terminologie für Job-Shop-Probleme wird im Gegensatz zur vorliegenden Arbeit (vgl. Definition 2.2.1 und 2.2.2) eine Organisation als „sequence“ bezeichnet.

Definition 6.1.1 Für ein gegebenes Shop-Scheduling-Problem sei $\mathcal{S}^*$ eine echte Teilmenge der Menge aller zulässigen Lösungen mit der Eigenschaft, daß $\mathcal{S}^*$ für jede beliebige Bearbeitungszeit-Matrix $P = (p_{ij})$ mindestens eine optimale Lösung enthält. Dann heißt $\mathcal{S}^*$ eine *potentiell-optimale* Menge, und die Elemente von $\mathcal{S}^*$ heißen *potentiell-optimale* Lösungen.

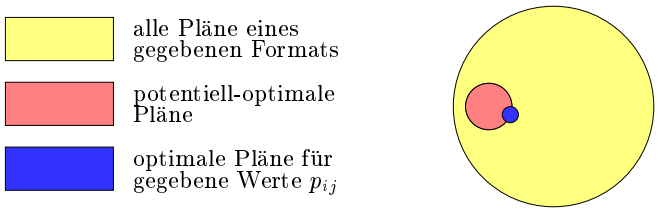

Abbildung 6.1: Die potentiell-optimalen Elemente im Mengensystem der Pläne.

Während die Eigenschaft eines Plans, potentiell-optimal zu sein, unabhängig von der Wahl der Bearbeitungszeiten p_{ij} ist, hängt eine optimale Lösung von den Werten p_{ij} sowie von der gegebenen Zielfunktion ab.

Im folgenden ist zunächst das Open-Shop-Problem Ausgangspunkt für die Strukturuntersuchung der zugehörigen Lösungen. Zur Lösung eines Open-Shop-Problems ist neben der Organisation auch eine Technologie zu bestimmen. Solche Sequenzen werden anhand von Plänen, also durch bestimmte lateinische Rechtecke, modelliert. Eine allgemeine Klassifikation der Pläne hinsichtlich Optimalität ist Abbildung 6.1 zu entnehmen. Ziel von Kapitel 6 und 7 ist es, geeignete Charakterisierungen für eine potentiell-optimale Menge von Plänen zu entwickeln, um „günstige“ Pläne effizient enumerieren zu können.

Für das Problem $J|n = 2|C_{\max}$ haben Akers und Friedman in [1] eine eindeutige Charakterisierung einer potentiell-optimalen Menge von Sequenzen anhand von sogenannten freien Maschinen angegeben. In einer Sequenz für $J|n = 2|C_{\max}$ wird eine Maschine als *freie Maschine* bezeichnet, wenn auf ihr die beiden Aufträge direkt aufeinanderfolgend bearbeitet werden und zur gleichen Zeit auf den anderen Maschinen keine Bearbeitungen stattfinden. In [1] wird gezeigt, daß alle Sequenzen ohne freie Maschinen zusammen für $J|n = 2|C_{\max}$ eine Menge potentiell-optimaler Sequenzen bilden.

Ähnliche Ergebnisse für Flow-Shop-Probleme des Typs $F||C_{\max}$ haben Conway, Maxwell und Miller in [26] erzielt. Es zeigt sich, daß diejenigen Sequenzen eine Menge potentiell-optimaler Sequenzen bilden, bei denen die Aufträge auf den Maschinen M_1 und M_2 in der gleichen organisatorische Reihenfolge stehen, und ebenfalls die Maschinen M_{m-1} und M_m eine gleiche organisatorische Reihenfolge besitzen. In [26] werden daraus auch Resultate für andere reguläre Zielfunktionen abgeleitet.

Eine Verallgemeinerung dieser Ergebnisse für Open-Shop-Probleme mit beliebiger Auftrags- und Maschinenanzahl gestaltet sich schwierig. Eine mögliche allgemeine Charakterisierung einer potentiell-optimalen Menge von Sequenzen ist durch das anschließend vorgestellte und erstmals von M. Kleinau in [55] eingeführte Konzept der *Irreduzibilität* von Plänen gegeben (siehe auch [15]). Im folgenden wird

dieses Konzept erweitert, ergänzt und präzisiert. Weiterhin werden mehrere Begriffe an die gängige Terminologie der Graphentheorie angepaßt, insbesondere im Fall sogenannter Vergleichbarkeitsgraphen.

6.2 Konzept der Irreduzibilität

Auf der Menge aller $n \times m$–Pläne ist eine spezielle Halbordnung definiert, deren minimale Elemente eine Menge potentiell-optimaler Pläne bilden. Zur Beschreibung dieser Halbordnung sind folgende Vorbetrachtungen notwendig.

Es sei A ein $n \times m$–Plan und $G(A) = (V, E)$ der zugehörige Ablaufgraph. Im folgenden handelt es sich bei den betrachteten Wegen in $G(A)$ stets um gerichtete Wege. Die hier dargestellten Aussagen über Pläne gelten für alle Shop-Scheduling-Probleme. Für einen Weg $W = (v_0, v_1, \ldots, v_k)$ in $G(A)$ sei $V_W = \{v_0, v_1, \ldots, v_k\}$ die Menge seiner Knoten. Weiterhin sei $\mathcal{W}_A(v_k)$ die Menge aller Wege in $G(A)$, die im Knoten $v_k \in V$ enden. Aufgrund der Definition des Ablaufgraphen $G(A) = (V, E)$ ist leicht zu sehen, daß sich die Fertigstellungszeit $c_{ij}(A)$ einer Operation o_{ij} im Plan A anhand von

$$c_{ij}(A) = \max_{W \in \mathcal{W}_A(o_{ij})} \left\{ \sum_{o_{kl} \in V_W} p_{kl} \right\} \tag{6.1}$$

berechnen läßt, wobei die p_{kl} die Bearbeitungszeiten der Operationen o_{kl} angeben.

Definition 6.2.1 Ein Weg $W \in \mathcal{W}_A(v_k)$ in $G(A)$ heißt *dominant*, wenn kein anderer Weg $W' \in \mathcal{W}_A(v_k)$ mit $V_W \subset V_{W'}$ existiert.

Es sei $\mathcal{W}_A^*(v_k) \subseteq \mathcal{W}_A(v_k)$ die Menge aller dominanten Wege in $G(A)$, die im Knoten v_k enden. Da die Bearbeitungszeiten p_{ij} für alle $i = 1, \ldots, n$ und $j = 1, \ldots, m$ nicht negativ sind, genügt es, sich bei der Bestimmung der Fertigstellungszeiten $c_{ij}(A)$ auf dominante Wege zu beschränken, d. h. es folgt unmittelbar

$$c_{ij}(A) = \max_{W \in \mathcal{W}_A^*(o_{ij})} \left\{ \sum_{o_{kl} \in V_W} p_{kl} \right\}. \tag{6.2}$$

Wenn Gleichung (3.1) auf einen bestimmten Plan A bezogen wird, so wird für die zugehörige Gesamtbearbeitungszeit $C_{\max}(A)$ die Gleichung

$$C_{\max}(A) = \max_{\substack{1 \le i \le n \\ 1 \le j \le m}} \{c_{ij}(A)\}. \tag{6.3}$$

geschrieben.

Definition 6.2.2 Ein Plan B heißt *reduzierbar* auf einen Plan A, geschrieben $A \preceq B$, wenn für jeden dominanten Weg W_a in $G(A)$ ein dominanter Weg W_b in $G(B)$ mit $V_{W_a} \subseteq V_{W_b}$ existiert. Zwei Pläne A und B heißen *ähnlich*, symbolisiert durch $A \sim B$, wenn $A \preceq B$ und $B \preceq A$ gilt. Ein Plan B wird *streng reduzierbar* auf einen Plan A genannt, wenn $A \preceq B$ und $A \not\sim B$ gilt. In diesem Fall wird $A \prec B$ geschrieben.

Beispiel 6.2.3 Es werden die beiden 4×4–Pläne

$$A = \begin{pmatrix} 1 & 6 & 7 & 8 \\ 2 & 5 & 6 & 7 \\ 3 & 4 & 5 & 1 \\ 4 & 7 & 1 & 2 \end{pmatrix} \quad \text{und} \quad A' = \begin{pmatrix} 2 & 6 & 7 & 8 \\ 1 & 5 & 6 & 7 \\ 3 & 4 & 5 & 1 \\ 4 & 7 & 1 & 2 \end{pmatrix}$$

betrachtet. Die Pläne $A = (a_{ij})$ und $A' = (a'_{ij})$ unterscheiden sich nur in den Einträgen a_{11}, a_{21} bzw. a'_{11}, a'_{21}. Daher sind in den entsprechenden Ablaufgraphen $G(A)$ und $G(A')$ nur diejenigen dominanten Wege verschieden, die in a_{11} bzw. a'_{21} starten. Diese dominanten Wege besitzen jedoch jeweils die gleiche Menge von Knoten, also gilt $A \sim A'$.

Definition 6.2.4 Ein Plan B heißt *irreduzibel*, wenn es keinen Plan A mit $A \prec B$ gibt.

Die Relation '$\prec$' induziert eine Halbordnung auf der Menge aller Pläne eines gegebenen Formats $n \times m$. Die minimalen Elemente dieser Halbordnung sind gerade die irreduziblen Pläne. Man kann nachprüfen, daß die Pläne A und A' aus Beispiel 6.2.3 irreduzibel sind. Es existieren also ähnliche irreduzible Pläne.

Für einen Plan A ist der Plan, der sich durch Umkehrung aller seiner technologischen und organisatorischen Reihenfolgen ergibt, der Umkehrplan $\overline{A}$ von A. Offensichtlich ist stets $A \sim \overline{A}$, und es gelten für zwei Pläne A und B mit $A \preceq B$ die Beziehungen

$$\overline{A} \preceq B, \quad A \preceq \overline{B} \quad \text{und} \quad \overline{A} \preceq \overline{B}. \tag{6.4}$$

An dieser Stelle sei darauf hingewiesen, daß die Definition

$$A \prec B \;:=\; A \preceq B \;\wedge\; A \neq B \;\wedge\; A \neq \overline{B}$$

in [55] im Zusammenhang mit der Irreduzibilität nicht zweckmäßig ist, denn aus $A \sim B$ folgt nicht notwendig $A = B \;\vee\; A = \overline{B}$. Es existieren also ähnliche Pläne A, B mit $A \neq B$ und $A \neq \overline{B}$. Beispielsweise sind alle Pläne paarweise ähnlich, deren zugeordnete Ablaufgraphen einen Weg enthalten, der alle Operationen o_{ij}

($i = 1, \ldots, n$ und $j = 1, \ldots, m$) umfaßt. Weiterhin sind z. B. die beiden Pläne aus Beispiel 6.2.3 ähnlich. Aus diesem Grund wird stets

$$A \prec B := A \preceq B \ \wedge \ A \not\sim B, \tag{6.5}$$

gemäß Definition 6.2.2 gesetzt.

Offensichtlich handelt es sich bei der Ähnlichkeit („$\sim$“) von Plänen um eine Äquivalenzrelation. Zwischen der Ähnlichkeit und den Äquivalenzrelationen aus Abschnitt 5.1 (Isomorphie, Äquivalenz und Struktur-Äquivalenz) besteht ein grundlegender Unterschied: Die Ähnlichkeit zweier Pläne A und B beruht ausschließlich auf dem direkten Vergleich der Knotenmengen der gerichteten Wege in den zugehörigen Ablaufgraphen $G(A)$ und $G(B)$. Bei den anderen Äquivalenzrelationen gehören zwei Pläne einer Äquivalenzklasse an, wenn alle Wegstrukturen in den zugehörigen Ablaufgraphen bei entsprechender Vertauschung von Aufträgen und Maschinen bzw. Umkehrung der Orientierungen identisch bleiben.

Dieser Unterschied läßt sich anhand von Beispiel 6.2.3 darstellen: Die Pläne A und A' sind zwar ähnlich ($A \sim A'$), aber nicht-struktur-äquivalent ($A \not\equiv_S A'$). Das heißt, diese Pläne lassen sich durch Vertauschung von Zeilen und Spalten, durch Matrix-Transposition oder durch Umkehrung aller Reihenfolgen nicht ineinander überführen, obwohl sie in den entsprechenden Ablaufgraphen bezüglich der Mengen von Knoten der gerichteten Wege gleichartig sind.

Die Eigenschaft eines Plans, irreduzibel zu sein, ist offensichtlich invariant bezüglich Isomorphie, Äquivalenz und Struktur-Äquivalenz. Das bedeutet zum Beispiel: Wenn A und B zwei struktur-äquivalente Pläne sind ($A \equiv_S B$) und A irreduzibel ist, dann ist auch B irreduzibel. Dieser Zusammenhang wird später bei der Enumeration der irreduziblen Pläne ausgenutzt, bei der nur die nicht-struktur-äquivalenten Vertreter irreduzibler Pläne erzeugt werden.

Der anschließende Satz stellt die eigentlich Motivation für die Einführung des Konzepts der Irreduzibilität dar.

Satz 6.2.5 [15] *Es seien A und B zwei Pläne eines Open-Shop-Problems des Typs $O||C_{\max}$ mit n Aufträgen und m Maschinen. Weiterhin sei $A \preceq B$. Dann gilt*

$$C_{\max}(A) \leq C_{\max}(B). \tag{6.6}$$

Beweis: Die Aussage des Satzes folgt direkt aus der Definition von „$\preceq$“ zusammen mit den Beziehungen (6.2) und (6.3). □

Aus $A \prec B$ läßt sich nicht notwendig $C_{\max}(A) < C_{\max}(B)$ folgern. Nur wenn es im Fall gegebener Bearbeitungszeiten $p_{ij} > 0$ ($i = 1, \ldots, n$ und $j = 1, \ldots, m$) für

$A \prec B$ einen eindeutigen kritischen Weg[1] W_b in $G(B)$ gibt, für den kein dominanter Weg W_a in $G(A)$ mit $V_{W_a} = V_{W_b}$ existiert, gilt $C_{\max}(A) < C_{\max}(B)$.

Die anschließende Folgerung zeigt, daß es hilfreich ist, den Suchraum auf die Menge der irreduziblen Pläne einzuschränken, wenn man nach einem optimalen Plan für ein Open-Shop-Problem sucht.

Folgerung 6.2.6 *Es sei $\mathcal{P}^*_{n,m}$ die Menge aller irreduziblen $n \times m$–Pläne. Für jedes Open-Shop-Problem des Typs $O||C_{\max}$ mit n Aufträgen und m Maschinen enthält $\mathcal{P}^*_{n,m}$ einen optimalen Plan.*

Beweis: Es sei B ein optimaler Plan eines gegebenen Open-Shop-Problems. Falls kein Plan A mit $A \prec B$ existiert, ist B irreduzibel. Sei daher B streng reduzierbar. Dann existiert eine Menge $\{A_1, A_2, \ldots, A_k\}$ von Plänen mit $k \geq 1$ und $A_1 \prec A_2 \prec \cdots \prec A_k \prec B$, so daß A_1 irreduzibel ist. Wegen $A_1 \prec A_2 \prec \cdots \prec A_k \prec B$ ist auch $A_1 \preceq B$ und nach Satz 6.2.5 gilt dann $C_{\max}(A_1) \leq C_{\max}(B)$. Damit ist der irreduzible Plan A_1 ebenfalls optimal. □

Die Menge $\mathcal{P}^*_{n,m}$ der irreduziblen $n \times m$–Pläne ist eine Teilmenge von $\mathcal{P}_{n,m}$, der Menge aller $n \times m$–Pläne für $O||C_{\max}$ mit n Aufträgen und m Maschinen. Da sich in $\mathcal{P}^*_{n,m}$ jede beliebige Bearbeitungszeit-Matrix eine optimale Lösung befindet, ist $\mathcal{P}^*_{n,m}$ eine potentiell-optimale Menge von Plänen im Sinne von Definition 6.1.1.

Es stellt sich heraus, daß die Menge $\mathcal{P}^*_{n,m}$ keine minimale Menge von potentiell-optimalen Plänen ist: Die beiden irreduziblen Pläne A und A' aus Beispiel 6.2.3 sind ähnlich. Das heißt, immer wenn A für eine gegebene Menge von Bearbeitungszeiten p_{ij} optimal ist, so ist es auch A', und man kann sich bei der Suche nach einer minimalen Menge potentiell-optimaler Pläne stets auf einen der beiden Pläne beschränken.

Es existiert eine echte Teilmenge $\mathcal{M} \subset \mathcal{P}^*_{n,m}$, in der sich unabhängig von den gegebenen Bearbeitungszeiten stets ein optimaler Plan befindet. Eine minimale Menge $\mathcal{M}$ von potentiell-optimalen Plänen heißt *unvermeidbar*. Eine solche unvermeidbare Menge $\mathcal{M}$ von Plänen ist im Gegensatz zur Menge $\mathcal{P}^*_{n,m}$ jedoch im allgemeinen nicht eindeutig bestimmt (siehe [96, 97]).

6.3 Stabilität optimaler Pläne

Thema der ersten beiden Abschnitte dieses Kapitels war die Beschreibung potentiell-optimaler bzw. irreduzibler Pläne, deren potentielle Optimalität unabhängig von den gegebenen Bearbeitungszeiten p_{ij} ist.

[1] Ein *kritischer Weg* in einem Ablaufgraphen ist ein gerichteter Weg W, dessen Knotenmenge V_W ausschließlich aus Operationen o_{ij} besteht, die nicht später begonnen werden können, ohne die Gesamtbearbeitungszeit $C_{\max}$ zu verlängern (sogenannte kritische Operationen).

In diesem Abschnitt spielen bei der sogenannten Stabilität von Plänen die Bearbeitungszeiten p_{ij} eine größere Rolle. Im Falle einer gegebenen Bearbeitungszeit-Matrix $P = (p_{ij})$ wird für einen zugehörigen optimalen Plan A untersucht, für welche Abweichungen von p_{ij} der Plan A optimal bleibt.

Stabilitätsradius eines optimalen Plans

Arbeiten von SOTSKOV, STRUSEVICH und TANAEV [89, 94] dienen als Ausgangspunkt und Grundlage der Definitionen im Zusammenhang mit dem Stabilitätsradius optimaler Pläne. Zur Vereinfachung wird im folgenden für die Menge der Bearbeitungszeiten p_{ij} anstelle der Bearbeitungszeit-Matrix $P = (p_{ij})$ der Vektor $\boldsymbol{p} = (p_{11}, p_{12}, \ldots, p_{nm})$ geschrieben, bei dem die Elemente von P zeilenweise hintereinander stehen.

Eine optimale Lösung eines Shop-Scheduling-Problems ist im allgemeinen nicht eindeutig. Es sei $\mathcal{S}^*(\boldsymbol{p})$ die Menge aller Pläne, die bezüglich des Bearbeitungszeitvektors $\boldsymbol{p}$ optimal sind. Weiterhin sei $\mathbb{R}_+^{nm}$ der nm-dimensionale Raum nicht-negativer reeller Vektoren mit der *Maximum-* bzw. *Tschebyschev-Metrik*, d. h. zwischen zwei Vektoren $\boldsymbol{p}, \boldsymbol{p'} \in \mathbb{R}_+^{nm}$ mit

$$\boldsymbol{p} = (p_{11}, p_{12}, \ldots, p_{nm}) \text{ und } \boldsymbol{p'} = (p'_{11}, p'_{12}, \ldots, p'_{nm})$$

wird der *Abstand* durch

$$d(\boldsymbol{p}, \boldsymbol{p'}) := \max_{\substack{1 \le i \le n \\ 1 \le j \le m}} |p_{ij} - p'_{ij}|$$

definiert. Eine *abgeschlossene Kugel* mit dem *Mittelpunkt* $\boldsymbol{p}$ und dem *Radius* ε ist die Menge

$$K_\varepsilon(\boldsymbol{p}) := \{\boldsymbol{p'} \in \mathbb{R}_+^{nm} \mid d(\boldsymbol{p}, \boldsymbol{p'}) \le \varepsilon\}.$$

Eine abgeschlossene Kugel $K_\varepsilon(\boldsymbol{p})$ heißt *Stabilitätskugel* eines optimalen Plans $A \in \mathcal{S}^*(\boldsymbol{p})$, wenn der Plan A für jeden Vektor dieser Kugel optimal bleibt, d. h. wenn $A \in \mathcal{S}^*(\boldsymbol{p'})$ für alle $\boldsymbol{p'} \in K_\varepsilon(\boldsymbol{p})$ gilt. Der *Stabilitätsradius* $\varepsilon(A, \boldsymbol{p})$ eines optimalen Plans $A \in \mathcal{S}^*(\boldsymbol{p})$ ist der maximale Wert, den der Radius einer Stabilitätskugel von A um $\boldsymbol{p}$ annehmen kann, also

$$\varepsilon(A, \boldsymbol{p}) := \max\{r \in \mathbb{R}^+ \mid K_r(\boldsymbol{p}) \text{ ist Stabilitätskugel von } A \in \mathcal{S}^*(\boldsymbol{p})\}.$$

Ein optimaler Plan $A \in \mathcal{S}^*(\boldsymbol{p})$ heißt *stabil*, wenn $\varepsilon(A, \boldsymbol{p}) > 0$ gilt. Für die Lösung von Shop-Scheduling-Problemen sind optimale Pläne $A \in \mathcal{S}^*(\boldsymbol{p})$ mit einem möglichst großen Stabilitätsradius interessant, denn derartige Pläne bleiben auch bei relativ starken Abweichungen der Bearbeitungszeiten p_{ij} optimal im Gegensatz zu anderen Plänen mit vergleichsweise geringerem Stabilitätsradius. Es stellt sich nun die Frage, ob es stets stabile optimale Pläne gibt, und ob in bestimmten Fällen Pläne existieren, die für alle Bearbeitungszeitvektoren $\boldsymbol{p}$ optimal bleiben.

Zusammenhang zwischen Stabilität und Irreduzibilität

In diesem Unterabschnitt wird ein Zusammenhang zwischen dem Stabilitätsradius optimaler Pläne und ihrer Irreduzibilität hergestellt. Auf diese Weise ist es unter anderem möglich, wichtige Ergebnisse aus [89, 90] bezüglich der Stabilität optimaler Pläne mit Hilfe des Konzepts der Irreduzibilität zu formulieren.

Für ein Open-Shop-Problem mit n Aufträgen und m Maschinen ist $\mathcal{P}_{n,m}$ die Menge aller $n \times m$–Pläne. Bei den anderen Shop-Scheduling-Problemen (Job-Shop-, Flow-Shop- und General-Shop-Problem) sind die entsprechenden Mengen ($\mathcal{P}^J_{n,m}$, $\mathcal{P}^F_{n,m}$, $\mathcal{P}^G_{n,m}$) offensichtlich kleiner als $\mathcal{P}_{n,m}$, da nicht jeder Plan aus $\mathcal{P}_{n,m}$ die gegebenen Vorrangbedingungen zwischen den Operationen erfüllt. Bei einem General-Shop-Problem $G||C_{\max}$ werden die gegebenen Vorrangbedingungen zwischen den Operationen anhand der disjunktiven Kanten in der Menge C des disjunktiven Graphen $G^* = (V, C \cup D)$ repräsentiert (vgl. Abbildung 3.1). In diesem Fall ist ein Plan A genau dann in der zugehörigen Menge $\mathcal{P}^G_{n,m}$ enthalten, wenn für seinen Ablaufgraphen $G(A) = (V, E)$ die Beziehung $C \subset E$ erfüllt ist.

Das Konzept der Irreduzibilität kann auch für die Shop-Scheduling-Probleme mit Vorrangbedingungen übernommen werden. So ist z. B. bei einem gegebenen General-Shop-Problem mit zugrundeliegender Menge $\mathcal{P}^G_{n,m}$ von Plänen ein Plan B genau dann *irreduzibel*, wenn es keinen Plan $A \in \mathcal{P}^G_{n,m}$ mit $A \prec B$ gibt. Im restlichen Teil dieses Abschnitts wird bei der Benutzung der Begriffe „irreduzibel“, „reduzierbar“, usw. stets von dieser Beschränkung auf diejenigen Pläne ausgegangen, die für das jeweils gegebene General-Shop-Problem relevant sind.

Die folgenden Aussagen für General-Shop-Probleme sind offensichtlich genauso auf Open-Shop-, Job-Shop- und Flow-Shop-Probleme übertragbar, da diese Probleme Spezialfälle des General-Shop-Problems sind. Für das Problem $G||C_{\max}$ beschreibe der Ausdruck $\mathcal{S}^*(\boldsymbol{p})$ anschließend stets die Menge der optimalen Pläne zum gegebenen Bearbeitungszeitvektor $\boldsymbol{p}$.

Satz 6.3.1 *Es sei $A \in \mathcal{S}^*(\boldsymbol{p})$ ein Plan für das Problem $G||C_{\max}$ mit $\varepsilon(A, \boldsymbol{p}) > 0$. Ist A' ein Plan mit $A' \preceq A$, so gilt $A' \in \mathcal{S}^*(\boldsymbol{p})$ und $\varepsilon(A', \boldsymbol{p}) \geq \varepsilon(A, \boldsymbol{p})$.*

Beweis: Es sei A ein optimaler Plan für $G||C_{\max}$. Aus $A' \preceq A$ folgt wegen Satz 6.2.5 die Ungleichung $C_{\max}(A') \leq C_{\max}(A)$, also ist A' auch optimal. Der Stabilitätsradius eines Plans A zum Bearbeitungszeitvektor $\boldsymbol{p} = (p_{11}, \ldots, p_{nm})$ kann gemäß [89] anhand von

$$\varepsilon(A, \boldsymbol{p}) = \inf \left\{ d(\boldsymbol{p}, \boldsymbol{p'}) \;\middle|\; \boldsymbol{p'} \in \mathbb{R}^{nm}_+, \max_{W_a \in \mathcal{W}^*_A} \sum_{o_{ij} \in V_{W_a}} p'_{ij} > \min_{\substack{B \in \mathcal{P}^G_{n,m} \\ B \neq A}} \max_{W_b \in \mathcal{W}^*_B} \sum_{o_{ij} \in V_{W_b}} p'_{ij} \right\} \tag{6.7}$$

berechnet werden, wobei $\mathcal{W}_A^*$ die Menge der dominanten Wege im Ablaufgraphen $G(A)$ des Plans A ist. Wegen $A' \preceq A$ gibt es zu jedem $W_{A'} \in \mathcal{W}_{A'}^*$ ein $W_A \in \mathcal{W}_A^*$ mit $V_{W_{A'}} \subseteq V_{W_A}$. Die Gültigkeit der Ungleichung im entsprechenden Ausdruck (6.7) für $\varepsilon(A', \boldsymbol{p})$ hängt daher von höchstens sovielen Komponenten p'_{ij} ab wie im Fall $\varepsilon(A, \boldsymbol{p})$. Also gibt es für die Wahl der Bearbeitungszeitvektoren $\boldsymbol{p'} \in \mathbb{R}_+^{nm}$ beim Abstand $d(\boldsymbol{p}, \boldsymbol{p'})$ in $\varepsilon(A', \boldsymbol{p})$ mindestens soviele Freiheitsgrade wie in $\varepsilon(A, \boldsymbol{p})$. Damit ist $\varepsilon(A', \boldsymbol{p}) \geq \varepsilon(A, \boldsymbol{p})$. □

Der anschließende Satz gibt ein eineindeutiges Kriterium für die Existenz von stabilen Plänen.

Satz 6.3.2 *Es sei $A \in \mathcal{S}^*(\boldsymbol{p})$ ein Plan für das Problem $G||C_{\max}$. Es ist genau dann $\varepsilon(A, \boldsymbol{p}) > 0$, wenn für alle $B \in \mathcal{S}^*(\boldsymbol{p})$ die Beziehung $A \preceq B$ gilt.*

Beweis: Die in [89] angegebene indirekte Beweisführung kann offensichtlich ohne Schwierigkeiten an die Terminologie des Konzepts der Irreduzibilität angepaßt werden, da der Berechnung des Stabilitätsradius von A stets die Anzahl der Knoten der dominanten Wege im Ablaufgraphen $G(A)$ zugrundeliegt. □

Folgerung 6.3.3 *Jeder eindeutig optimale Plan ist stabil.*

Analog zu Satz 6.3.2 kann nun auch ein Kriterium für Pläne mit unbegrenzter Stabilität anhand des Konzepts der Irreduzibilität formuliert werden.

Satz 6.3.4 *Es sei $A \in \mathcal{S}^*(\boldsymbol{p})$ ein Plan für das Problem $G||C_{\max}$. Es ist genau dann $\varepsilon(A, \boldsymbol{p}) = \infty$, wenn für alle $B \in \mathcal{P}_{n,m}^G$ die Beziehung $A \preceq B$ gilt.*

Beweis: Analog zu Satz 6.3.2. □

Für ein General-Shop-Problem existiert also genau dann ein Plan mit unendlichem Stabilitätsradius, wenn es einen irreduziblen Plan gibt, auf den alle anderen Pläne dieses Problems reduzierbar sind. Offensichtlich kann die Bedingung

$$A \preceq B \quad \text{für alle} \quad B \in \mathcal{P}_{n,m}^G$$

in diesem Satz für General-Shop-Probleme nur dann erfüllt sein, wenn n und m klein sind oder schon einer Vielzahl von technologischen und organisatorischen Reihenfolgen vorgegeben sind, denn „$\preceq$“ ist im allgemeinen keine lineare Ordnung, d. h. im allgemeinen sind nicht alle Pläne bezüglich „$\preceq$“ vergleichbar.

Da eine optimale Lösung eines Shop-Scheduling-Problems im allgemeinen nicht eindeutig ist, erscheint die Darstellung der qualitativen Unterschiede zwischen verschiedenen optimalen Plänen für gegebene Bearbeitungszeiten p_{ij} mit Hilfe der Konzepte der Irreduzibilität und der Stabilität sinnvoll. Abbildung 6.2 zeigt unter allen

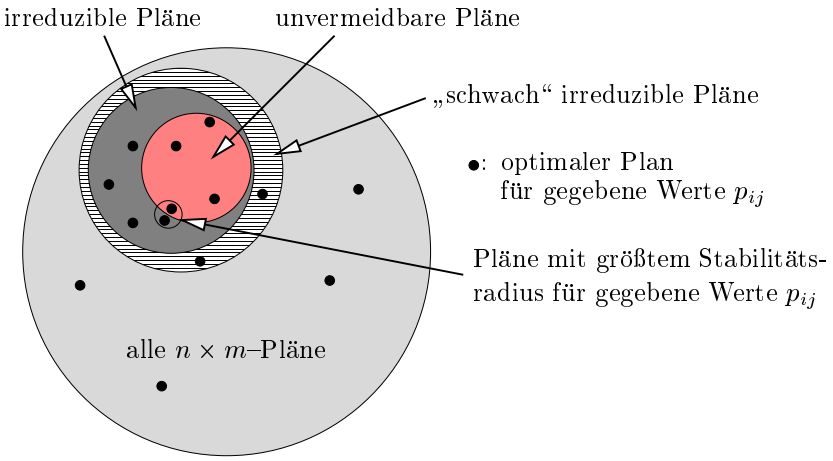

Abbildung 6.2: Qualitative Unterschiede zwischen verschiedenen optimalen Plänen.

Plänen eines gegebenen Formats $n \times m$ die verschiedenen Arten optimaler Pläne. Es ist klar, daß unter den optimalen Plänen diejenigen Pläne im Falle von möglichen Abweichungen der vorgegebenen Werte p_{ij} vorzuziehen sind, die maximalen Stabilitätsradius besitzen. Mit „schwach“ irreduziblen Plänen sind in Abbildung 6.2 diejenigen Pläne gemeint, die sich während eines Enumerationsalgorithmus durch die Anwendung polynomial nachprüfbarer hinreichender Bedingungen für die Reduzibilität als Kandidaten für irreduzible Pläne herausstellen (vgl. Abschnitt 7.3).

Die explizite Berechnung des Stabilitätsradius eines optimalen Plans gemäß [89] ist kompliziert und zeitintensiv. In [16] haben Bräsel, Sotskov und Werner erstmals auch Stabilitätsradien von Plänen für diejenigen Shop-Scheduling-Probleme betrachtet, die die Summe der Fertigstellungszeiten ($\sum C_i$) als Optimalitätskriterium besitzen. Beim Vergleich mit dem $C_{\max}$-Kriterium (siehe auch Sotskov, Tanaev und Werner [90]) stellt sich heraus, daß ein optimaler Plan im Falle der Gesamtbearbeitungszeit $C_{\max}$ gewöhnlich einen größeren Stabilitätsradius aufweist als bei $\sum C_i$. Weiterhin haben die Stabilitätsradien von optimalen Plänen bei dem $C_{\max}$-Kriterium eine größere Varianz. Beim $\sum C_i$-Kriterium besitzen alle optimalen Pläne sogar häufig den gleichen Stabilitätsradius.

Kapitel 7

Enumeration irreduzibler Pläne

Zur Bestimmung der Anzahl $P^*_{n,m}$ der irreduziblen $n \times m$ Pläne werden zwei verschiedene Enumerationsmethoden vorgestellt. Die Enumeration irreduzibler Pläne anhand von Vergleichbarkeitsgraphen ist Gegenstand der Abschnitte 7.1 und 7.2. Bei dieser Methode werden Vertreter der Ähnlichkeitsklassen (Äquivalenzklassen bezüglich der Relation „$\sim$“) erzeugt. Mit Hilfe der zweiten Methode, die in Abschnitt 7.3 und 7.4 beschrieben ist, können alle bzw. alle nicht-struktur-äquivalenten irreduziblen Pläne enumeriert werden. Eine effiziente Implementation dieser Methode, die auf der Enumeration in Kapitel 5 basiert, liefert eine Reihe von numerischen Resultaten, auf die abschließend in Abschnitt 7.5 eingegangen wird.

7.1 Reduzibilität zwischen zwei Plänen

Es sei $G = (V, E)$ ein azyklischer Digraph. Ein Weg $W = (v_0, v_1, \ldots, v_k)$ in G enthält die Knoten $v_0, v_1, \ldots, v_k$ und die Kanten $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$. Zu $G = (V, E)$ wird der Digraph $G^{tc} = (V, E^{tc})$ definiert, in dem für alle $v, w \in V$ genau dann $(v, w) \in E^{tc}$ ist, wenn $v \neq w$ ist und in G ein Weg von v nach w existiert. Der Digraph $G^{tc} = (V, E^{tc})$ heißt *transitive Hülle (transitive closure)* von $G = (V, E)$. Eine Kante (v, w) eines azyklischen Digraphen heißt *redundant*, wenn es einen Weg von v nach w gibt, der die Kante (v, w) nicht enthält. Als *transitive Reduktion (transitive reduction)* von $G = (V, E)$ wird der Digraph $G_{tr} = (V, E_{tr})$ bezeichnet, der keine redundante Kanten enthält, und dessen transitive Hülle gleich der transitiven Hülle von G ist. Abbildung 7.2 zeigt die transitive Hülle $G^{tc}(A)$ und die transitive Reduktion $G_{tr}(A)$ des Ablaufgraphen $G(A)$ aus Abbildung 7.1.

Mit $[G]$ wird der ungerichtete Graph bezeichnet, der einem Digraphen G zugrunde liegt. Weiterhin sei an dieser Stelle daran erinnert, daß es sich bei $n \times m$–Ablaufgraphen um indizierte Digraphen handelt, d. h. jeder Knoten eines Ablaufgraphen wird jeweils mit einer bestimmten Operation o_{ij} des betrachteten Shop-Schedu-

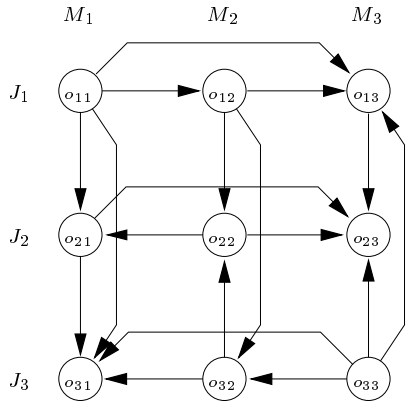

Abbildung 7.1: Ein Ablaufgraph $G(A)$.

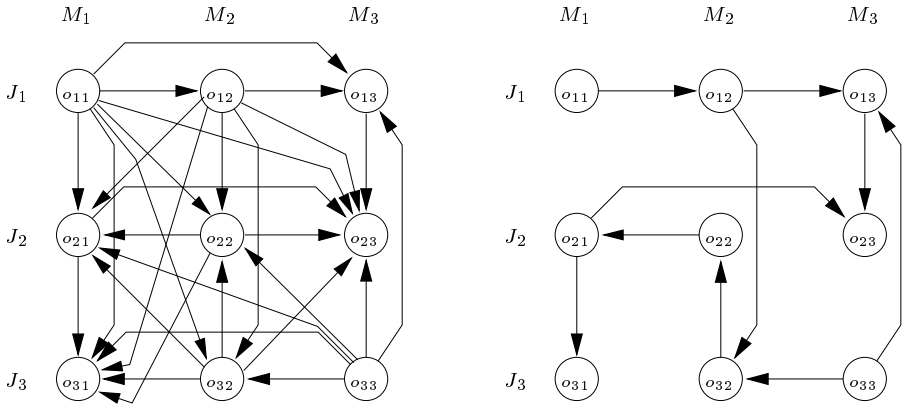

Abbildung 7.2: Die transitive Hülle $G^{tc}(A)$ und die transitive Reduktion $G_{tr}(A)$ von $G(A)$.

ling-Problems identifiziert, da sonst keine eindeutige Beziehung zwischen Plänen und Ablaufgraphen im Sinne von Satz 3.2.2 bestünde. Für zwei indizierte (Di-)Graphen $G_1 = (V, E_1)$ und $G_2 = (V, E_2)$ mit derselben Knotenmenge V wird $G_1 \subset G_2$, $G_1 \subseteq G_2$, bzw. $G_1 = G_2$ geschrieben, falls $E_1 \subset E_2$, $E_1 \subseteq E_2$ bzw. $E_1 = E_2$ gilt. Der folgende Satz gibt ein eineindeutiges Kriterium für die strenge Reduzibilität eines Plans B auf einen Plan A.

Satz 7.1.1 [10] *Es seien A und B zwei $n \times m$–Pläne und $G(A)$ bzw. $G(B)$ die zugehörigen $n \times m$–Ablaufgraphen. Es gilt genau dann $A \prec B$, wenn $[G^{tc}(A)] \subset [G^{tc}(B)]$ ist.*

Beweis: Angenommen, es ist $A \prec B$. Dann gibt es zu jedem Weg W_a in $G(A)$ einen Weg W_b in $G(B)$ mit $V_{W_a} \subseteq V_{W_b}$. Außerdem existiert in $G(B)$ mindestens ein dominanter Weg W_b^*, für den es in $G(A)$ keinen dominanten Weg gibt, der alle Knoten von W_b^* enthält. Gemäß der Definition der transitiven Hülle entspricht jeder dominante Weg eines Digraphen G einer maximalen Clique in $[G^{tc}]$. Daher gilt also $[G^{tc}(A)] \subset [G^{tc}(B)]$.

Ist umgekehrt $[G^{tc}(A)] \subset [G^{tc}(B)]$, so sind alle Cliquen aus $[G^{tc}(A)]$ in Cliquen aus $[G^{tc}(B)]$ enthalten, und es gibt in $[G^{tc}(B)]$ eine maximale Clique C_b, deren Knotenmenge echt größer ist als die einer entsprechenden maximalen Clique C_a in $[G^{tc}(A)]$. Jede orientierte Clique besitzt einen Hamiltonschen Weg (siehe RÉDEI [78]). Es kann schnell eingesehen werden, daß dieser Hamiltonsche Weg eindeutig ist, wenn die Orientierung der Clique transitiv ist. Die den Graphen $[G^{tc}(A)]$ und $[G^{tc}(B)]$ zugrunde liegenden Orientierungen sind transitiv. Seien also W_a^* bzw. W_b^* die eindeutigen dominanten Wege der Ablaufgraphen $G(A)$ bzw. $G(B)$, die den maximalen Cliquen C_a bzw. C_b in den Graphen $[G^{tc}(A)]$ bzw. $[G^{tc}(B)]$ entsprechen. Es ist $V_{W_a^*} \subset V_{W_b^*}$ und für alle anderen Wege W_a in $G(A)$ existiert ein Weg W_b in $G(B)$ mit $V_{W_a} \subseteq V_{W_b}$. Also gilt $A \prec B$. □

Mit Hilfe dieses Satzes läßt sich relativ leicht ein Algorithmus konstruieren, der für zwei gegebene $n \times m$–Pläne A und B anhand der zugehörigen Ablaufgraphen $G(A)$ und $G(B)$ testet, ob $A \prec B$, $A \preceq B$ oder $A \sim B$ gilt. Für diesen Test ist die Berechnung der transitiven Hülle eines Ablaufgraphen sowie das Überprüfen der Relation $[G^{tc}(A)] \subset [G^{tc}(B)]$ notwendig. Ein hierzu in [10] benutztes Verfahren hat die Zeitkomplexität $O(n^2m^2)$.

Das Problem der Berechnung der transitiven Hülle bzw. der transitiven Reduktion ist sehr häufig untersucht worden (siehe z. B. [43]). Es werden nun in diesem Zusammenhang zwei Arbeiten mit den besten, bis heute bekannten Laufzeiten zitiert.

Der in [40] vorgestellte Algorithmus von GORALČÍKOVÁ und KOUBEK berechnet die transitive Hülle $G^{tc} = (V, E^{tc})$ eines azyklischen Digraphen $G = (V, E)$ in der

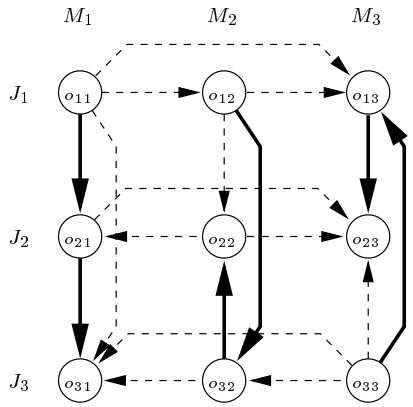

Abbildung 7.3: Eine Ketten-Zerlegung des Ablaufgraphen $G(A)$ aus Abbildung 7.1.

Zeit $O(|V| \cdot |E_{tr}| + |E^{tc}|)$, wobei E_{tr} bzw. E^{tc} die Menge der Kanten der zugehörigen transitiven Reduktion G_{tr} bzw. Hülle G^{tc} darstellt. Da für einen Ablaufgraphen $G = (V, E)$ stets die Beziehung $O(|E^{tc}|) = O(|V| \cdot |E_{tr}|)$ gilt, reduziert sich in diesem Fall die Laufzeit für die Berechnung der transitiven Hülle zu $O(|V| \cdot |E_{tr}|)$. Die transitive Reduktion $G_{tr}(A)$ eines $n \times m$–Ablaufgraphen $G(A)$ enthält keine redundante Kanten. Also sind für jeden Auftrag J_i bzw. für jede Maschine M_j höchstens die direkten Vorgänger-Nachfolger-Beziehungen der Technologie bzw. Organisation in $G_{tr}(A)$ enthalten. Das heißt, in der transitiven Reduktion $G_{tr}(A)$ existieren maximal $n(m-1) + m(n-1)$ gerichtete Kanten (vgl. Abbildung 7.2). Also folgt für die Laufzeit des Algorithmus zur Bestimmung der transitiven Hülle eines $n \times m$–Ablaufgraphen insgesamt die Schranke $O(n^2m^2)$. Mit Hilfe sogenannter Ketten-Zerlegungen konnte SIMON in [86] die Laufzeit für die Berechnung der transitiven Hülle eines Graphen verringern:

Definition 7.1.2 Es sei $G = (V, E)$ ein Digraph. Eine *Ketten-Zerlegung (chain decomposition)* von G ist eine Partition $P = \{P_1, \ldots, P_k\}$ von V in disjunkte nichtleere Mengen P_i mit $V = P_1 \cup \ldots \cup P_k$, bei der jede Menge P_i, $i = 1, \ldots, k$ einen gerichteten Weg (auch *Kette* genannt) in G aufspannt, wenn transitive Kanten ignoriert werden. Die Zahl k heißt *Weite (width)* der Ketten-Zerlegung P.

Als Beispiel ist in Abbildung 7.3 eine Ketten-Zerlegung der Weite 3 eines 3×3–Ablaufgraphen dargestellt.

In [86] wird gezeigt, daß sich eine Ketten-Zerlegung eines Digraphen $G = (V, E)$ in der Zeit $O(|V| + |E|)$ konstruieren läßt. Darauf aufbauend hat SIMON den Algorithmus in [40] verbessert, so daß die transitive Hülle von $G = (V, E)$ in $O(k \cdot |E_{tr}|)$ berechnet werden kann, wobei k die Weite einer Ketten-Zerlegung von G ist. Die

Laufzeit des Algorithmus in [40] wird dadurch verkürzt, daß anstelle der Bestimmung aller Knoten $w \in V$ die von einem gegebenen Knoten $v \in V$ über einen Weg aus erreichbar sind, nur die jeweiligen ersten Knoten der Mengen P_i der Ketten-Zerlegung bestimmt werden, die von v aus erreichbar sind. Für diesen Schritt ergibt sich also anstelle von $O(|V|)$ die Laufzeit $O(k)$, da die bereits bestehende Information der Ketten-Zerlegung auf diese Weise ausgenutzt wird.

Offensichtlich kann die Menge der Knoten eines $n \times m$–Ablaufgraphen stets in n bzw. m Ketten zerlegt werden (siehe Abbildung 7.3). Für die Bestimmung der transitiven Hülle eines $n \times m$–Ablaufgraphen anhand des Algorithmus von Simon [86] ergibt sich wegen $|E_{tr}| = O(nm)$ für $n \leq m$ also insgesamt ein Zeitaufwand der Größenordnung $O(n^2 m)$. Allerdings benötigt der Test, ob die Relationen $\subset$, $\subseteq$ oder $=$ zwischen den Graphen $[G^{tc}(A)]$ und $[G^{tc}(B)]$ bestehen, bereits $O(n^2 m^2)$ Zeit. Also verringert sich trotz des verbesserten Algorithmus von Simon [86] nicht die Laufzeit für den gesamten Algorithmus, der gemäß Satz 7.1.1 zwei Pläne A und B auf Reduzibilität testet.

7.2 Enumeration bezüglich Ähnlichkeitsklassen

Ein *Vergleichbarkeitsgraph (comparability graph)* G^* ist ein ungerichteter Graph, der sich transitiv orientieren läßt. Mit Hilfe bestimmter Vergleichbarkeitsgraphen wird in diesem Abschnitt eine Enumerationsmethode vorgestellt, die auf einer neuen Charakterisierung irreduzibler Pläne beruht.

Es sei $H_{n\times m} = (V_H, E_H)$ der in Abschnitt 3.1 eingeführte Hamming-Graph $K_n \times K_m$. Die Knotenmenge V_H korrespondiert in naheliegender Weise mit der Menge der Operationen o_{ij} des betrachteten Shop-Scheduling-Problems mit n Aufträgen und m Maschinen. Weiterhin sei $G(A)$ der $n \times m$–Ablaufgraph eines beliebigen $n \times m$–Plans A. Jede Kante e des Graphen $[G^{tc}(A)]$ mit $e \notin E_H$ heißt *Diagonalkante*. Aufgrund von Satz 7.1.1 kann die folgende Charakterisierung irreduzibler Pläne gegeben werden.

Folgerung 7.2.1 *Ein $n \times m$–Plan A ist genau dann irreduzibel, wenn kein Vergleichbarkeitsgraph G^* mit $H_{n\times m} \subseteq G^* \subset [G^{tc}(A)]$ existiert.*

In [10] haben Bräsel *et al.* mit Hilfe dieses Zusammenhangs einen Enumerationsalgorithmus für irreduzible Pläne konstruiert. Der Algorithmus beruht auf der Entwicklung ungerichteter Graphen G durch sukzessives Hinzufügen von Diagonalkanten zum Hamming-Graphen $H_{n\times m}$. Sobald unter diesen Graphen ein Vergleichbarkeitsgraph G^* gefunden wird, handelt es sich um einen inklusions-minimalen Vergleichbarkeitsgraphen bezüglich Kantenanzahl unter der Voraussetzung $H_{n\times m} \subseteq G^*$.

Die zum Erkennen des Vergleichbarkeitsgraphen notwendige transitive Orientierung von G^* liefert dann einen zugehörigen irreduziblen Plan A' mit $G^* = [G^{tc}(A')]$.

Für diesen Algorithmus zur Enumeration irreduzibler Pläne spielt die transitive Orientierung von Graphen eine zentrale Rolle. In [65, 66, 67, 91] haben McConnell und Spinrad das Problem der transitiven Orientierung von Graphen ausführlich untersucht und mit Hilfe der sogenannten modularen Dekomposition (Zerlegung eines Graphen in bestimmte Komponenten) schnelle Algorithmen zu dessen Lösung gefunden. Das aktuellste Resultat [67] beschreibt einen Algorithmus für die transitive Orientierung eines Graphen $G = (V, E)$ mit Zeitkomplexität $O(|V| + |E|)$. Da dieser Algorithmus jedoch nicht erkennt, ob es sich beim betrachteten Graphen G überhaupt um einen Vergleichbarkeitsgraphen handelt, d. h. ob sich G überhaupt transitiv orientieren läßt, muß zum Erkennen eines Vergleichbarkeitsgraphen zusätzlich der Algorithmus von Simon [86] angewandt werden. Es wird dabei getestet, ob sich die gefundene Orientierung von G von seiner transitiven Hülle unterscheidet. Ist dies nicht der Fall, liegt ein Vergleichbarkeitsgraph vor. Im Rahmen des Enumerationsalgorithmus ist für das Erkennen von Vergleichbarkeitsgraphen G^* mit $H_{n\times m} \subseteq G^* \subset [G^{tc}(A)]$ die Zeitkomplexität des Verfahrens aus [86] nicht dominierend. Der zugrunde liegende Graph $[G^{tc}(A)]$ der transitiven Hülle eines Ablaufgraphen $G(A)$ enthält maximal $O(n^2m^2)$ Kanten. Diese Schranke ist für die Laufzeit eines Verfahrens zum Erkennen entsprechender Vergleichbarkeitsgraphen entscheidend.

Neben den hier erwähnten Verfahren zur transitiven Orientierung und zum Erkennen von Vergleichbarkeitsgraphen bildet die Auswahl der Reihenfolge der hinzuzufügenden Diagonalkanten einen wichtigen Bestandteil des in [10] beschriebenen Enumerationsalgorithmus. Bei dieser Enumeration wird ein Vertretersystem M für die Ähnlichkeitsklassen in der Menge $\mathcal{P}^*_{n,m}$ aller irreduziblen $n \times m$–Pläne erzeugt.

7.3 Hinreichende Bedingungen

Aufbauend auf Ergebnissen von M. Kleinau [55] werden in diesem Abschnitt hinreichende Bedingungen für die Reduzibilität von Plänen hergeleitet. Offensichtlich korrespondieren hinreichende Bedingungen für die Reduzibilität eines Plans B mit entsprechenden notwendigen Bedingungen für die Irreduzibilität von B. Anhand dieser Bedingungen können im Verlaufe des Plan-Enumerationsalgorithmus aus Abschnitt 5.3 diejenigen Pläne in effizienter Weise eliminiert werden, deren zugeordnete Ablaufgraphen ungünstige Wegstrukturen besitzen. Durch diese Selektion ist die Menge der enumerierten Pläne im Vergleich zur Menge aller Pläne schon erheblich eingeschränkt.

Die folgenden hinreichende Bedingungen für die strenge Reduzibilität eines ge-

gebenen Plans B beruhen jeweils auf langen gerichteten Wegen im zugeordneten Ablaufgraphen $G(B)$. Die Beweise von Satz 7.3.1 – 7.3.4 erscheinen in [10, 11]. Exemplarisch wird hier nur Satz 7.3.1 bewiesen.

Satz 7.3.1 [11] *Es sei B ein Plan, der eine Operation o_{ij} mit folgenden Eigenschaften enthält: Die Operation o_{ij} besitzt mindestens einen Nachfolger, aber kein Nachfolger von o_{ij} in der Zeile i bzw. Spalte j hat einen direkten Vorgänger außerhalb der Zeile i bzw. Spalte j. Dann gibt es einen Plan A mit $A \prec B$.*

Beweis: Es wird ein Plan A konstruiert, indem die Operation o_{ij} gelöscht und als Senke in die technologische Reihenfolge des Auftrags J_i und die organisatorische Reihenfolge der Maschine M_j wiedereingefügt wird. Wenn auf diese Weise im zugehörigen Ablaufgraphen eine neue Menge von Operationen entstehen würde, die auf einem gemeinsamen Weg liegen, dann muß diese Menge o_{ij} enthalten, da der Rest des Plans unverändert bleibt. Angenommen, es gibt eine Operation o_{kl}, die auf einem Weg mit o_{ij} in $G(A)$ liegt, aber nicht in $G(B)$. Da o_{ij} Senke in $G(A)$ ist, muß dieser Weg von o_{kl} nach o_{ij} gerichtet sein, und an einer Stelle in Zeile i oder Spalte j eintreten. Da es keinen Weg von o_{kl} nach o_{ij} in $G(B)$ gibt, muß die Operation, bei der der neue Weg in $G(A)$ in Zeile i oder Spalte j eintritt, ein Nachfolger von o_{ij} in $G(B)$ sein. Dies ist ein Widerspruch zur im Satz gemachten Annahme. Daher gilt $A \preceq B$.

Es wird nun gezeigt, daß in $G(A)$ weniger Mengen von Operationen existieren, die jeweils auf einem gemeinsamen Weg liegen. Ohne Beschränkung der Allgemeinheit gelte für die Einträge von B die Beziehung

$$b_{\max} = \max\{b_{pj} | p = 1, \ldots, n\} \geq \{b_{iq} | q = 1, \ldots, m\}.$$

Es sei k so gewählt, daß $b_{kj} = b_{\max}$ ist. Also ist o_{kj} in $G(B)$ ein Nachfolger von o_{ij} und hat nach Annahme keinen Vorgänger außerhalb von Spalte j. Weiterhin ist jede Operation o_{kl} mit $l \neq j$ ein Nachfolger von o_{kj} und liegt daher in $G(B)$ auf einem gemeinsamen Weg mit o_{ij}. Wegen $b_{kl} > b_{\max}$ gibt es in $G(A)$ keinen Weg, der in o_{kl} startet und in Zeile i oder Spalte j eintritt. Andererseits gibt es in $G(A)$ keine Wege, die in o_{ij} starten. Wegen Satz 7.1.1 gilt also $A \prec B$. □

Satz 7.3.2 [10] *Es sei B ein $n \times m$–Plan, bei dem jeder Auftrag J_i, $i = 1, \ldots, n$ als erstes auf derselben Maschine M_j bearbeitet wird. Dann gibt es einen Plan A mit $A \prec B$.*

Satz 7.3.3 [11] *Es sei $n, m \geq 3$ und B ein $n \times m$–Plan mit maximalen Rang $r \geq nm - 2$. Dann gibt es einen Plan A mit $A \prec B$.*

Satz 7.3.4 [11] *Es sei $B = (b_{ij})$ ein Plan, der zwei Aufträge J_i und J_k sowie eine Maschine M_j enthält, so daß o_{ij} die letzte Operation von J_i, o_{kj} die erste Operation von J_k, und o_{kj} direkter Nachfolger von o_{ij} in der organisatorischen Reihenfolge von M_j ist. Wenn ein $l \neq j$ existiert mit $b_{kl} \leq b_{ij} + 3$ oder $b_{kj} \leq b_{il} + 3$, dann gibt es einen Plan A mit $A \prec B$.*

Dieser Satz ist eine alternative Formulierung eines ursprünglich in [55] erzielten Resultats. Er liefert eine hinreichende Bedingung dafür, daß der Plan B durch Umdrehen einer gerichteten Kante im zugehörigen Ablaufgraphen $G(B)$ zu einem Plan A streng reduzierbar ist. Ein irreduzibler Plan darf daher notwendiger Weise keine der im Satz 7.3.4 beschriebenen Eigenschaften haben.

7.4 Enumeration durch Ausschlußverfahren

Die Dissertation von M. Kleinau [55] enthält einen Algorithmus, der entscheidet, ob ein gegebener Plan B irreduzibel ist. Dieser Irreduzibilitätstest benötigt exponentiellen Zeitaufwand und ist daher zur direkten Anwendung auf jeden enumerierten Plan ungeeignet, weil die Anzahl aller $n \times m$–Pläne bereits für vergleichsweise kleine Werte von n und m sehr groß ist (siehe Tabelle 5.4).

Der hier vorgestellte neue Enumerationsalgorithmus basiert auf Algorithmus 5.3.7 zur Enumeration aller $n \times m$–Pläne für gegebene n und m. An geeigneten Stellen in Algorithmus 5.3.7 werden die Bedingungen aus dem vorangegangenen Abschnitt angewandt, um die streng reduzierbaren Pläne verwerfen zu können. Es ist leicht zu sehen, daß die notwendigen Bedingungen für die Irreduzibilität gemäß Satz 7.3.4–7.3.3 jeweils in polynomialer Zeit getestet werden können. Zum Teil ist es sogar möglich, diese Bedingungen bereits auf Teilpläne anzuwenden, so daß auf die vollständige Erzeugung bestimmter Pläne ganz verzichtet werden kann, falls ein Teilplan bereits eine der notwendigen Bedingungen für die Irreduzibilität verletzt.

Bei Anwendung der Tests aufgrund Satz 7.3.4–7.3.3 stellt sich heraus, daß diejenigen Verfahren sehr effektiv sind, die auf dem Wiedereinsetzen einer Operation als Quelle oder Senke beruhen. Das heißt, bei Anwendung dieser Verfahren werden die streng reduzierbaren Pläne am häufigsten als solche erkannt. Daher wird im folgenden die Zeitkomplexität dieses Verfahrens angegeben.

Satz 7.4.1 [11] *Es sei $n \leq m$. Das Problem, ob ein $n \times m$–Plan B durch Löschen und Wiedereinfügen einer Operation als Quelle oder Senke auf einen Plan A streng reduziert werden kann, ist in $O(n^2m^2)$ Zeit entscheidbar.*

Beweis: Die transitive Hülle $G^{tr}(B)$ des Ablaufgraphen $G(B)$ eines Plans B läßt sich wie schon gezeigt für $n \leq m$ in $O(n^2m)$ Zeit bestimmen. Der Beweis dieses Satzes in [11] zeigt, daß für eine feste Operation o_{ij} maximal $O(nm)$ gerichtete

Wege in $G^{tr}(B)$ getestet werden müssen, um über die strenge Reduzibilität von B entscheiden zu können. Da diese Prozedur für alle nm Operationen in $G(B)$ durchgeführt werden muß, ergibt sich insgesamt die Zeitkomplexität $O(n^2m^2)$. □

Falls ein Plan während des Enumerationsalgorithmus alle beschriebenen notwendigen Bedingungen für die Irreduzibilität erfüllt, wird ein abschließender Test auf Irreduzibilität angewandt, der allerdings exponentiellen Zeitaufwand benötigt. Dieser Test basiert auf sogenannten Implikationsklassen, in die die Menge der gerichteten Kanten des Ablaufgraphen $G(B)$ zum betrachteten Plan B partitioniert werden kann.

Die Einführung der Implikationsklassen ist folgendermaßen motiviert: Angenommen, für einen Plan B gibt es im Graphen $[G^{tc}(B)]$ keine ungerichtete Kante $\{o_{ij}, o_{kl}\}$ zwischen den Operationen o_{ij} und o_{kl}. Dann existiert diese Kante nach Satz 7.1.1 auch nicht in $[G^{tc}(A)]$ für jeden Plan A mit $A \prec B$. Also enthält $G(A)$ entweder die beiden gerichteten Kanten (o_{ij}, o_{kj}) und (o_{kl}, o_{kj}) oder (o_{kj}, o_{ij}) und (o_{kj}, o_{kl}). In dieser Weise bedingen sich die Orientierungen der beiden Kanten $\{o_{ij}, o_{kj}\}$ und $\{o_{kj}, o_{kl}\}$ gegenseitig.

Zur Definition der Implikationsklassen wird die Terminologie von GOLUMBIC [39] übernommen: Es wird von einem gegebenen ungerichteten Graphen $[G^{tc}(A)]$ ausgegangen. Eine gerichtete Kante (o_{ij}, o_{kj}) in $G(A)$ *erzwingt direkt* die Kante (o_{kl}, o_{kj}), geschrieben, $(o_{ij}, o_{kj})\Gamma(o_{kl}, o_{kj})$, wenn $\{o_{ij}, o_{kl}\}$ nicht in $[G^{tc}(A)]$ existiert. Diese binäre Relation wird analog für $(o_{kj}, o_{ij})\Gamma(o_{kj}, o_{kl})$ definiert. Die transitive Hülle Γ^{tc} von Γ ist offensichtlich eine Äquivalenzrelation. Die Äquivalenzklassen bezüglich Γ^{tc} heißen *Implikationsklassen*. Zwei Kanten e und f sind genau dann in derselben Implikationsklasse ($e\,\Gamma^{tc} f$), wenn es Kanten $e_1, \ldots, e_k$ mit

$$e\Gamma e_1\Gamma e_2\Gamma \cdots \Gamma e_k\Gamma f$$

gibt.

Wenn ein Plan B mit $G(B) = (V, E)$ durch Umkehrung der Orientierungen der Kanten einer bestimmten Menge $E' \subseteq E$ zu einem Plan A reduziert werden soll, so müssen offensichtlich jeweils die Orientierungen aller Kanten einer Implikationsklasse umgekehrt werden, da sonst zusätzliche Wege in $G(B)$ entstehen würden. Es ergibt sich die folgende Aussage.

Satz 7.4.2 [11] *Es sei B ein Plan mit Ablaufgraph $G(B) = (V, E)$. Wenn alle Kanten aus E zur gleichen Implikationsklasse gehören, dann ist B irreduzibel.*

Beweis: Da alle Kanten von $G(B)$ zu einer Implikationsklasse gehören, wird der Plan B nur dann reduziert, wenn die Orientierungen aller Kanten in $G(B)$ umgekehrt werden. Wegen $B \sim \overline{B}$ ist B irreduzibel. □

Beim abschließenden Test auf Irreduzibilität eines Plans B werden die Kanten aus $G(B)$ in ihre Implikationsklassen partitioniert, wobei k die Anzahl dieser Implikationsklassen sei. Danach wird für jede Teilmenge $\mathcal{T}$ der Menge aller Implikationsklassen der Plan A bzw. Ablaufgraph $G(A)$ konstruiert, der durch Umkehrung aller Orientierungen der Kanten entsteht, die zu den Implikationsklassen aus $\mathcal{T}$ gehören. Da alle 2^k Teilmengen der Menge der Implikationsklassen geprüft werden müssen, ist dieser Test auf Irreduzibilität nicht in polynomialer Zeit realisierbar.

Ablauf des Enumerationsalgorithmus

Es wird von der *Plan-Enumeration* (Algorithmus 5.3.7) ausgegangen. In Schritt 2 werden bereits bei der Technologie-Erzeugung jeweils Tests gemäß Satz 7.3.1 und 7.3.2 angewandt. Dabei werden nur diejenigen Pläne vollständig erzeugt, die keine der Bedingungen aus Satz 7.3.1 oder 7.3.2 erfüllen. Auf jeden vollständig erzeugten Plan wird zunächst der Test bezüglich seines maximalen Rangs (gemäß Satz 7.3.3) angewandt (im Algorithmus 5.3.7 zwischen Schritt 3 und 4). Nach dem Ermitteln der Automorphismen der Pläne bezüglich Isomorphie, Äquivalenz oder Struktur-Äquivalenz in Schritt 4 werden die übrigbleibenden Vertreter anhand von Satz 7.3.2 und 7.3.4 auf mögliche strenge Reduzibilität hin überprüft. Bevor der abschließende exponentielle Irreduzibilitätstest aufgrund der Betrachtung der Implikationsklassen für die verbleibenden Pläne gestartet wird, testet man für jede Operation o_{ij} eines Plans, ob der Plan durch Löschen und Wiedereinsetzen von o_{ij} als Quelle bzw. Senke streng reduziert werden kann (vgl. Satz 7.4.1). Am Ende des Algorithmus werden auf diese Weise ausschließlich die irreduziblen Vertreter der betrachteten Äquivalenzklassen als Pläne ausgegeben.

Der anschließende Unterabschnitt zeigt, daß ein polynomialer Irreduzibilitätstest bisher nur im Fall einer bestimmten Klasse von unvollständigen Mengen von Operationen bekannt ist.

Polynomialer Test auf Irreduzibilität bei bestimmten unvollständigen Mengen von Operationen

Es sei $\mathcal{J} = \{J_1, \ldots, J_n\}$ die Menge der Aufträge und $\mathcal{M} = \{M_1, \ldots, M_m\}$ die Menge der Maschinen eines Shop-Scheduling-Problems. Weiterhin sei $G_{n,m} = (\mathcal{J} \cup \mathcal{M}, \mathcal{O})$ der assoziierte bipartite Graph, bei dem die Kantenmenge $\mathcal{O} \subseteq \mathcal{J} \times \mathcal{M}$ die Menge der Operationen o_{ij} ($i = 1, \ldots, n$ und $j = 1, \ldots, m$) ist. Es werden nun Shop-Scheduling-Probleme mit $\mathcal{O} \subsetneq \mathcal{J} \times \mathcal{M}$ betrachtet. Die Lösungen von derartigen Problemen mit *unvollständigen Mengen von Operationen* werden anhand von Teilplänen $A = (a_{ij})$ repräsentiert, wobei für alle $i = 1, \ldots, n$ und $j = 1, \ldots, m$ der leere Eintrag $a_{ij} = \cdot$ gesetzt wird, wenn $o_{ij} \notin \mathcal{O}$ ist. Ansonsten geben die Einträge $a_{ij} \neq \cdot$

n	m	$P_{n,m}$	$P^*_{n,m}$	$S_{n,m}$	$S^*_{n,m}$
2	2	14	2	3	1
2	3	204	12	12	1
3	3	19 164	516	147	7
2	4	5 016	72	68	2
3	4	3 733 056	32 688	13 100	123
4	4	6 941 592 576	27 106 560	3 017 369	12 073
2	5	185 520	480	422	2
3	5	1 288 391 040	2 932 560	895 388	2 073
4	5	26 549 943 275 520	??	4 609 489 912	5 936 306
2	6	9 595 440	3 600	3 495	3
3	6	712 770 186 240	352 098 720	82 507 654	40 933
2	7	659 846 880	30 240	33 193	3
3	7	589 563 294 888 960	??	9 748 141 078	??

Tabelle 7.1: Anzahlen irreduzibler $n \times m$–Pläne und Gesamtanzahlen der $n \times m$–Pläne.

analog zu den Plänen die Reihenfolge der Bearbeitung der Operationen wieder (vgl. Definition 5.3.5).

Für Teilpläne eines Shop-Scheduling-Problems, dessen zugrundeliegende Menge von Operationen $\mathcal{O}$ als Kantenmenge im bipartiten Graphen $G_{n,m}$ einen Baum aufspannt, hat TAUTENHAHN in [96] einen polynomialen Irreduzibilitätstest entwickelt.

7.5 Numerische Auswertungen

In Tabelle 7.1 werden die Anzahlen der irreduziblen bzw. nicht-struktur-äquivalenten irreduziblen $n \times m$–Pläne ($P^*_{n,m}$ bzw. $S^*_{n,m}$) den entsprechenden Gesamtanzahlen für kleine Werte n und m gegenübergestellt. Es zeigt sich, daß jeweils nur ein kleiner Prozentsatz einer Menge von $n \times m$–Plänen irreduzibel ist (vgl. Tabelle 7.2 für $P^*_{n,m}/P_{n,m}$ in %). Die Algorithmen zur Lösung von Shop-Scheduling-Problemen, die ausschließlich in der Menge der irreduziblen Pläne nach einem Optimum suchen, sind folglich wesentlich effektiver als diejenigen, deren Suchraum durch die Menge aller zulässigen Lösungen gebildet wird.

Die Effektivität der verschiedenen Tests aufgrund der hinreichenden Bedingungen für die Reduzibilität von Plänen gemäß Satz 7.3.1 – 7.3.4 und Satz 7.4.1 kann z. B. an der Anzahl der 3×4–Pläne abgelesen werden, die diese Bedingungen erfüllen (siehe Tabelle 7.3). Alle bis auf den letzten Test (Irreduzibilitätstest gemäß der

$n \setminus m$	2	3	4	5	6	7
2	14.30	5.88	1.44	0.26	0.038	0.0046
3		2.69	0.88	0.23	0.049	??
4			0.39	??	??	??

Tabelle 7.2: Verhältnisse der Anzahlen irreduzibler $n \times m$–Pläne zu den jeweiligen Gesamtanzahlen (in %).

Test gemäß ...	# 3×4–Pläne
(ohne Test)	13100
Satz 7.3.1 und 7.3.2, ausschließlich auf die Technologien angewandt	6081
Satz 7.3.3	5640
Satz 7.3.2, ausschließlich auf die Organisationen angewandt	5050
Satz 7.3.4	4291
Satz 7.4.1	256
Umkehrung der Kanten aus Implikationsklassen	123

Tabelle 7.3: Anzahlen nicht-struktur-äquivalenter 3×4–Pläne, die nach Anwendung der verschiedenen Tests auf Irreduzibilität übrigbleiben.

Umkehrung der Kanten aus den Implikationsklassen) sind in polynomialer Zeit ausführbar. Außerdem zeigt sich, daß dieser exponentielle Irreduzibilitätstest auf nur ca. 2% aller nicht-struktur-äquivalenten 3×4–Pläne angewandt werden muß, und vergleichbare Werte gelten auch für $n, m \geq 3$.

Die Werte in Tabelle 7.4 legen die Vermutung nahe, daß die hinreichende Bedingung für die strenge Reduzibilität bezüglich des maximalen Rangs aus Satz 7.3.3 für $n, m \geq 3$ noch verschärft werden kann.

Während die maximale Anzahl von Implikationsklassen für $n \leq 3$ und $m \leq 4$ noch relativ klein ist (≤ 4), wächst dieser Wert mit steigenden n bzw. m stark an. Dies liefert eine Begründung dafür, daß bereits die Anzahl der irreduziblen 4×5–Pläne nicht mehr in angemessener Zeit auf die beschriebene Methode berechenbar ist, denn der abschließende Irreduzibilitätstest muß auf alle Teilmengen der Menge aller Implikationsklassen eines Plans angewandt werden.

Die in der vorliegenden Arbeit vorgestellten Algorithmen zur Enumeration der irreduziblen Pläne sind zu einem gesamten C++–Programm zusammengefügt worden, so daß für gegebene n und m alle Werte für die irreduziblen $n \times m$–Pläne in

$n \times m$	max. Rang r	# irred. Pläne
2×2	2	1
2×3	3	1
3×3	3	1
	4	3
	5	3
2×4	4	2
3×4	4	4
	5	40
	6	75
	7	4
4×4	4	4
	5	88
	6	1 847
	7	5 845
	8	3 932
	9	355
	10	2

$n \times m$	max. Rang r	# irred. Pläne
2×5	5	2
3×5	5	38
	6	541
	7	1 153
	8	334
	9	7
2×6	6	3
3×6	6	658
	7	8 428
	8	19 744
	9	10 844
	10	1 248
	11	11
2×7	7	3

Tabelle 7.4: Anzahlen der nicht-struktur-äquivalenten irreduziblen $n \times m$–Pläne mit maximalem Rang r.

$n \times m$	# Implikationsklassen	
	durchschnittlich	maximal
2×2	1.00	1
2×3	1.00	1
3×3	1.14	2
2×4	1.00	1
3×4	1.30	4
4×4	1.53	9
2×5	1.00	1
3×5	1.87	8
2×6	1.00	1
3×6	2.97	18
2×7	1.00	1

Tabelle 7.5: Durchschnittliche und maximale Anzahlen der Implikationsklassen unter den nicht-struktur-äquivalenten irreduziblen $n \times m$–Plänen.

den Tabellen dieses Abschnitts in einem Durchlauf errechnet werden können. Der gesamte Programmdurchlauf benötigt auf einem PC Pentium I (133 Mhz) im Fall $n = m = 4$ ca. 184 Minuten.

Abschließend wird nun ein Beispiel angeführt, das drei spezielle irreduzible 4×4–Pläne zeigt.

Beispiel 7.5.1 Es seien die Pläne A_1, A_2 und A_3 mit

$$A_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 5 & 10 \\ 5 & 8 & 4 & 9 \\ 6 & 7 & 8 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 3 & 8 & 9 \\ 5 & 6 & 4 & 10 \\ 6 & 1 & 7 & 2 \\ 7 & 2 & 3 & 4 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 2 & 6 & 7 & 8 \\ 4 & 5 & 6 & 7 \\ 1 & 2 & 9 & 3 \\ 3 & 4 & 8 & 1 \end{pmatrix}$$

gegeben. Die Pläne A_1 und A_2 sind die einzigen beiden nicht-struktur-äquivalenten irreduziblen 4×4–Pläne mit maximalem Rang 10. Der Plan A_3 besitzt 9 Implikationsklassen, welches der maximalen Anzahl von Implikationsklassen unter den irreduziblen 4×4–Plänen entspricht.

Die Pläne A_1 und A_2 sind also die einzigen beiden nicht-struktur-äquivalenten 4×4–Pläne, deren zugehörige Ablaufgraphen $G(A_1)$ und $G(A_2)$ einen gerichteten Weg der Länge 10 enthalten. Obwohl ein Weg der Länge 10 relativ lang ist für optimale Pläne des Formats 4×4, können A_1 und A_2 bei der Suche nach einem optimalen Plan nicht von vorneherein ausgeschlossen werden, da keine Pläne A' mit $A' \prec A_1$ bzw. $A' \prec A_2$ existieren.

Der Plan A_3 ist ein Beispiel eines 4×4-Plans, bei dem während der Enumeration der abschließende Test auf Irreduzibilität am meisten Zeit beansprucht, weil A_3 die maximale Anzahl von Implikationsklassen besitzt. In diesem Fall müssen also beim Irreduzibilitätstest alle Kombinationen der 9 Implikationsklassen bezüglich der Orientierung ihrer Kanten überprüft werden.

Kapitel 8

Schlußbemerkungen

Die Grundlage vieler Ergebnisse dieser Arbeit ist die Identifikation von zulässigen Lösungen eines Open-Shop-Problems mit Plänen und Ablaufgraphen.

Es werden zunächst neue theoretische Resultate über die Anzahl der Pläne eines Open-Shop-Problems erzielt. So ist es nun möglich, die Anzahl der $3 \times m$–Pläne anhand einer Summenformel direkt anzugeben. Dieses Ergebnis baut auf einer in [3] gezeigten allgemeinen Formel für die Anzahl lateinischer Rechtecke des Formats $3 \times m$ mit maximalem Eintrag $r \leq 3m$ auf. Da auch für die entsprechenden lateinischen Rechtecke des Formats $4 \times m$ in [3] eine Formel erscheint, liegt die gleiche Vorgehensweise für die $4 \times m$–Pläne nahe. Die Korrektheit der Formel aus [3] für die 4-zeiligen lateinischen Rechtecke ist jedoch fragwürdig, da sich schon beim Einsetzen kleiner m und r Abweichungen von den zu erwartenden Anzahlen ergeben. Die fragliche Formel ist sehr kompliziert und der Beweis in [3] ist nur kurz mit dem Hinweis auf den nächstkleineren Fall skizziert. Daher verspricht ein weiterer intensiver Kontakt mit den Autoren dieser Arbeit Aussicht auf Erfolg bezüglich einer allgemeinen Formel für die $4 \times m$–Pläne.

Für $n \times m$–Pläne mit $n \geq 4$ werden neue obere und untere Schranken bewiesen. Während die obere Schranke eine echte Verbesserung der bisher bekannten Ergebnisse liefert, erweist sich die neue untere Schranke nach langer Rechnung als etwas schlechter im Vergleich zu einer bereits bekannten unteren Schranke in [14].

Sowohl die Bestimmung einer allgemeinen Formel für die Anzahl der $3 \times m$–Pläne als auch die Methoden zur Gewinnung der neuen Schranken basieren auf der Enumeration der $n \times m$–Ablaufgraphen bzw. der azyklischen Orientierungen des entsprechenden Hamming-Graphen $K_n \times K_m$. Die auf diese Weise erzielten Schranken können durch geeignete Ausnutzung der Eigenschaften der Ablaufgraphen eventuell noch verbessert werden.

Die Enumeration der azyklischen Orientierungen eines Graphen G steht in direktem Zusammenhang mit dem chromatischen Polynom von G. Zur Bestimmung der Plan-Anzahlen ist also das chromatische Polynom des Hamming-Graphen $K_n \times K_m$

grundlegend. Obwohl für das chromatische Polynom der Graphen des Typs $P_n \times K_m$ eine geschlossene Formel bekannt ist, bleibt das Problem der allgemeinen Bestimmung von $\chi(K_n \times K_m)$ bis heute ungelöst. Wenn gezeigt werden kann, daß dieses Problem $\#\mathcal{P}$-vollständig ist, so ist es auch das Problem der Bestimmung der Anzahl aller $n \times m$–Pläne.

Es ist nicht bekannt, ob über die Isomorphie zweier beliebigen Graphen in polynomialer Zeit entschieden werden kann, oder ob dieses Problem $\mathcal{NP}$-vollständig ist. In dieser Arbeit wird gezeigt, daß die Ablaufgraphen eine Klasse von Digraphen bilden, in der das Isomorphie-Problem polynomial lösbar ist. Dieses Ergebnis wird ausgenutzt, um Pläne mit gleichartiger Struktur zu identifizieren. Unter anderem auf dieser Grundlage wird ein Algorithmus zur Enumeration der $n \times m$–Pläne entwickelt.

Das in [15, 55] eingeführte Konzept der Irreduzibilität wird anhand neuer Terminologie und neuer Charakterisierungen irreduzibler Pläne erweitert. Darauf aufbauend sowie mit Hilfe der beschriebenen Enumeration aller $n \times m$–Pläne, wird ein Algorithmus zur Enumeration der irreduziblen Pläne entwickelt. Es stellt sich heraus, daß die Menge der irreduziblen Pläne keine minimale Menge potentiell-optimaler Pläne ist. Die Menge der irreduziblen $n \times m$–Pläne ist für gegebene n, m jedoch eindeutig bestimmt, während dies im Fall einer minimalen Menge von potentiell-optimalen Plänen nicht der Fall sein muß. Im Bereich der Bestimmung einer minimalen Menge von potentiell-optimalen Plänen bestehen noch vielfältige Forschungsaufgaben. Weiterhin ist bis heute unbekannt, ob das Entscheidungsproblem „Ist ein gegebener Plan irreduzibel?“ polynomial lösbar oder $\mathcal{NP}$-vollständig ist.

Die Enumerationsalgorithmen können für Probleme mit Vorrangbedingungen angepaßt werden, so daß mit Hilfe der Verhältnisse aus der Anzahl irreduzibler Pläne zur Anzahl der jeweils zulässigen Pläne Aussagen über die Schwierigkeit der Job-Shop-Probleme mit bestimmten Technologien getroffen werden können. Auf diese Weise lassen sich die praktischen Erfahrungen bezüglich der unterschiedlichen Dauer für das Berechnen einer optimalen Lösung verschiedener Job-Shop-Probleme theoretisch begründen (siehe [10]).

Es werden in dieser Arbeit Zusammenhänge zwischen der Irreduzibilität und der Stabilität von Plänen hergestellt. Da die Stabilität von Plänen für Algorithmen zur Lösung von Shop-Scheduling-Problemen mit nicht genau vorgegebenen Bearbeitungszeiten p_{ij} eine wesentliche Rolle spielen, könnte hier auch der Einsatz der entwickelten notwendigen Bedingungen für die Irreduzibilität eines Plans angewandt werden.

Wenn bei einem Shop-Scheduling-Problem für alle Bearbeitungszeiten nur jeweils untere und obere Schranken ($a_{ij} \leq p_{ij} \leq b_{ij}$) bekannt sind, ist also denkbar, die durch einen Lösungsalgorithmus entwickelten Pläne so weit wie möglich zu redu-

zieren, da diese Pläne im allgemeinen bezüglich Optimalität stabiler sind. Weitere Untersuchungen in diesem Bereich versprechen Ergebnisse für Shop-Scheduling-Probleme, bei denen Abweichungen von den Bearbeitungszeiten zugelassen sind. Solche Shop-Scheduling-Probleme sind bei der praktischen Anwendung offensichtlich von großem Interesse (vgl. LAI *et. al* [58]).

Anhang A

Symbolverzeichnis

$A \prec B$ ..... Plan B ist streng reduzierbar auf Plan B

$A \preceq B$ ..... Plan B ist reduzierbar auf Plan B

$A \sim B$ ..... Pläne A und B sind ähnlich

$A \cong B$ ..... Pläne A und B sind isomorph

$A \equiv B$ ..... Pläne A und B sind äquivalent

$A \equiv_S B$ ..... Pläne A und B sind struktur-äquivalent

$A_{n,m}$ ..... Anzahl der Äquivalenzklassen von $n \times m$–Plänen

$\alpha(G)$ ..... Anzahl der azyklischen Orientierungen des Graphen G

$\chi(G,k)$ ..... chromatisches Polynom des Graphen G

D_{2n} ..... Diedergruppe der Ordnung $2n$

$\det(B)$ ..... Determinante einer Matrix B

$G = (V,E)$ .. Graph bzw. Digraph mit der Knotenmenge V und der Menge E von Kanten bzw. gerichteten Kanten

$[G]$ ..... zugrunde liegende Graph eines Digraphen G

$G_1 \cup G_2$ .... Vereinigung zweier disjunkter Graphen G_1 und G_2

$G_1 + G_2$ .... Verbindung zweier disjunkter Graphen G_1 und G_2

$I_{n,m}$ ..... Anzahl der Isomorphieklassen von $n \times m$–Plänen

$I_{TR}(n,m)$ ... Anzahl der Isomorphieklassen von $n \times m$–Technologien

$K(G)$ ..... Kirchhoff-Matrix eines Graphen G

$l(G)$ ..... Kantengraph eines Graphen G

$\mathcal{L}_{n,m,r}$ ..... lateinisches Rechteck des Formats $n \times m$ mit der Belegungsmenge $\{1, \ldots, r\}$ (für $n \leq m \leq r$)

$L(n, m, r)$ ... Anzahl der lateinischen Rechtecke $\mathcal{L}_{n,m,r}$, deren Einträge aus der Menge $\{1, \ldots, r\}$ stammen

$\mathbb{N}$ ..... Menge der natürlichen Zahlen $\{1, 2, 3, \ldots\}$

$\#\mathcal{P}$ ..... Klasse aller Enumerationsprobleme, für die ein nichtdeterministischer polynomialer Lösungsalgorithmus existiert

$\mathcal{NP}$ ..... Klasse aller Entscheidungsprobleme, für die ein nichtdeterministischer polynomialer Lösungsalgorithmus existiert

$O(g(n))$ ..... Klasse aller Funktionen $f(n)$, für die eine Konstante $C > 0$ existiert, so daß $|f(n)| \leq C|g(n)|$ für alle $n \geq n_0$ ist.

$o(g(n))$ ..... Klasse aller Funktionen $f(n)$, bei denen für jedes $\varepsilon > 0$ ein $n_0(\varepsilon)$ existiert mit $|f(n)| \leq \varepsilon|g(n)|$ für alle $n \geq n_0(\varepsilon)$.

$o_{ij} \prec o_{kl}$ .... Operation o_{ij} wird vor o_{kl} bearbeitet

$o_{ij} \rightarrow o_{kl}$ .... Vorrangbedingung zwischen Operationen o_{ij} und o_{kl}

$\mathcal{P}$ ..... Klasse aller Entscheidungsprobleme, für die ein deterministischer polynomialer Lösungsalgorithmus existiert

$P(n)$ ..... Anzahl der rangminimalen $n \times n$–Pläne (= Anzahl lateinischer Quadrate der Ordnung n)

$P(n, m)$ .... Anzahl rangminimaler $n \times m$–Pläne

$P(n, m, r)$ .. Anzahl der $n \times m$–Pläne mit maximalem Rang r, $m \leq r \leq nm$

P_n ..... Weg mit n Knoten

$P_{n,m}$ ..... Anzahl der $n \times m$–Pläne

$P^*_{n,m}$ ..... Anzahl irreduzibler $n \times m$ Pläne

$\mathcal{P}_{n,m}$ ..... Menge der $n \times m$–Pläne bzw. der zulässigen Lösungen eines Open-Shop-Problems mit n Aufträgen und m Maschinen

$\mathcal{P}^*_{n,m}$ ..... Menge der irreduziblen $n \times m$–Pläne

$\mathcal{P}^F_{n,m}$ ..... Menge der $n \times m$–Pläne bezüglich eines Flow-Shop-Problems mit n Aufträgen und m Maschinen

$\mathcal{P}^G_{n,m}$ ..... Menge der $n \times m$–Pläne bezüglich eines General-Shop-Problems mit n Aufträgen und m Maschinen

$\mathcal{P}^J_{n,m}$ ..... Menge der $n \times m$–Pläne bezüglich eines Job-Shop-Problems mit n Aufträgen und m Maschinen

$\mathrm{per}(B)$ ..... Permanente einer Matrix B

$\varrho(v)$ ..... Rang eines Knotens v in einem Digraphen

S_n ..... Permutationsgruppe, bestehend aus den Permutationen der Zahlen $\{1, 2, \ldots, n\}$

$S_{n,m}$ ..... Anzahl der Struktur-Äquivalenzklassen von $n \times m$–Plänen

$S_{TR}(n, m)$ ... Anzahl der Struktur-Isomorphieklassen von $n \times m$–Technologien

T_n ..... Baum mit n Knoten

$\tau(G)$ ..... Anzahl der aufspannenden Bäume eines Graphen G

V_W ..... Menge der Knoten eines Weges W in einem (Di-)Graphen

$\mathbb{Z}_n$ ..... zyklische Gruppe der Ordnung n

Literaturverzeichnis

[1] S. B. Akers und J. Friedman, A non-numerical approach to production scheduling problems, *Oper. Res.* **3** (1955), 429–442.

[2] S. Ashour, *Sequencing Theory*, vol. 69 of Lecture Notes in Economical and Mathematical Systems, Springer, 1972.

[3] K. B. Athreya, C. R. Pranesachar, und N. M. Singhi, On the number of Latin rectangles and chromatic polynomial of $L(K_{r,s})$, *European J. Combin.* **1** (1980), 9–17.

[4] G. D. Birkhoff, A determinant formula for the number of ways of coloring a map, *Ann. Math.* **14** (1912), 42–46.

[5] K. P. Bogart, An obvious proof of Burnside's lemma, *Amer. Math. Monthly* **98**, 10 (1991), 927–928.

[6] K. P. Bogart und J. Q. Longyear, Counting 3 by n Latin rectangles, *Proc. Amer. Math. Soc.* **54** (1976), 463–467.

[7] H. Bräsel, *Lateinische Rechtecke und Maschinenbelegung*, Habilitationsschrift, Technische Universität "Otto von Guericke" Magdeburg, 1990.

[8] H. Bräsel, *Schedulingtheorie: Mathematische Modelle und Methoden*, Universität Kaiserslautern, Fachbereich Mathematik, 1996. Vorlesungsskript, Wintersemester 1995/1996.

[9] H. Bräsel, L. Dornheim, M. Harborth, T. Tautenhahn, I. Wasmund, P. Willenius, und A. Winkler, LiSA – A Library of Scheduling Algorithms. Dynamic Survey, http://fma2.math.uni-magdeburg.de/~lisa/.

[10] H. Bräsel, M. Harborth, T. Tautenhahn, und P. Willenius, On the hardness of the classical job shop problem. To appear in *Ann. Oper. Res.*

[11] H. Bräsel, M. Harborth, T. Tautenhahn, und P. Willenius, On the set of solutions of an open shop problem. To appear in *Ann. Oper. Res.*

[12] H. Bräsel, M. Harborth, und P. Willenius, Isomorphism for digraphs and sequences of shop scheduling problems. To appear in *J. Combin. Math. Combin. Comput.*

[13] H. Bräsel und M. Kleinau, On number problems for the open shop problem, in *System Modelling and Optimization, Proc. 15th IFIP Conf.*, P. Kall, ed., vol. 180 of Lecture Notes in Control and Inform. Sci., Berlin, 1992, Springer, 145–154.

[14] H. Bräsel und M. Kleinau, On the number of feasible schedules of the open-shop-problem – an application of special Latin rectangles, *Optimization* **23** (1992), 251–260.

[15] H. Bräsel und M. Kleinau, New steps in the amazing world of sequences and schedules, *Math. Methods Oper. Res.* **43** (1996), 195–214.

[16] H. Bräsel, Y. N. Sotskov, und F. Werner, Stability of a schedule minimizing mean flow time, *Math. Comput. Modelling* **24**, 10 (1996), 39–53.

[17] J. W. Brown, Enumeration of Latin squares with application to order 8, *J. Combin. Theory Ser. B.* **5** (1968), 177–184.

[18] P. Brucker, *Scheduling Algorithms*, Springer, Berlin, 1995.

[19] P. Brucker, B. Jurisch, und M. Jurisch, Open shop problems with unit time operations, *Z. Oper. Res.* **37** (1993), 59–73.

[20] P. Brucker und S. Knust, Complexity results of scheduling problems. Dynamic Survey, http://www.mathematik.uni-osnabrueck.de/research/OR/class/.

[21] W. Burnside, *Theory of Groups of Finite Order*, University Press, Cambridge, 1897.

[22] J. Chen, A linear-time algorithm for isomorphism of graphs of bounded average genus, *SIAM J. Discrete Math.* **7**, 4 (1994), 614–631.

[23] G. L. Chia, Some problems on chromatic polynomials, *Discrete Math.* **172**, 1-3 (1997), 39–44. Chromatic polynomials and related topics (Shanghai, 1994).

[24] W. Clark, *The Gantt Chart*, Pitman and Sons, London, 3rd ed., 1952.

[25] C. J. Colbourn, The complexity of completing partial Latin squares, *Discrete Appl. Math.* **8**, 1 (1984), 25–30.

[26] R. W. Conway, W. L. Maxwell, und L. W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, MA, 1967.

[27] S. A. Cook, The complexity of theorem-proving procedures, in *Proc. 3rd Annual ACM Symp. Theory of Computing*, 1971, 151–158.

[28] N. G. de Bruijn, A note on the Cauchy-Frobenius lemma, *Indag. Math.* **41** (1979), 225–228.

[29] J. Dénes und A. D. Keedwell, *Latin Squares and their Applications*, Academic Press, New York and London, 1974.

[30] J. Dénes und A. D. Keedwell, *Latin Squares: New Developments in the Theory and Applications*, vol. 46 of Ann. Discrete Math., North-Holland, Amsterdam, 1991.

[31] K. Dohmen, Lower bounds and upper bounds for chromatic polynomials, *J. Graph Theory* **17**, 1 (1993), 75–80.

[32] K. Dohmen, Bounds to the chromatic polynomial of a graph, *Results Math.* **33**, 1-2 (1998), 87–88.

[33] P. Erdös und I. Kaplansky, The asymptotic number of Latin rectangles, *Amer. J. Math.* **68** (1946), 230–236.

[34] L. Euler, Recherches sur une nouvelle espèce de quarrés magiques, *Ges. Werke* **7** (1782), 291–392.

[35] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Horwood, Chichester, 1982.

[36] M. R. Garey und D. S. Johnson, *Computers and Intractability - a Guide to the Theory of NP-Completeness*, W. H. Freeman & Co, New York, 1979.

[37] W. Goddard, C. Kenyon, V. King, und L. J. Schulman, Optimal randomized algorithms for local sorting and set-maxima, *SIAM J. Comput.* **22**, 2 (1993), 272–283.

[38] C. D. Godsil und B. D. McKay, Asymptotic enumeration of Latin rectangles, *J. Combin. Theory Ser. B* **48** (1990), 19–44.

[39] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Comput. Sci. Appl. Math., Academic Press, New York, 1980.

[40] A. Goralčíková und V. Koubek, A reduct-and-closure algorithm for graphs, in *Mathematical Foundations of Computer Science. (Proc. Eighth Sympos., Olomouc, 1979)*, J. Becvar, ed., vol. 74 of Lect. Notes Comput. Sci., Berlin - New York, 1979, Springer, 301–307.

[41] R. L. Graham, E. L. Lawler, J. K. Lenstra, und A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Ann. Discrete Math.* **5** (1979), 287–326.

[42] T. A. Green, Asymptotic enumeration of generalized Latin rectangles, *J. Combin. Theory Ser. A* **51**, 2 (1989), 149–160.

[43] M. Habib, M. Morvan, und J.-X. Rampon, On the calculation of transitive reduction-closure of orders, *Discrete Math.* **111** (1993), 289–303.

[44] F. Harary, *Graphentheorie*, Oldenbourg, München, 1974. Germ. Transl. of "Graph Theory", Addison-Wesley, Reading, 1969.

[45] J. E. Hopcroft und R. M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, *SIAM J. Comput.* **2** (1973), 225–231.

[46] J. E. Hopcroft und J. K. Wong, Linear time algorithm for isomorphism of planar graphs, in *Proc. 6th ann. ACM Symp. Theory Comput.*, 1974, 172–184.

[47] W. Imrich und S. Klavžar, On the complexity of recognition Hamming graphs and related classes of graphs, *European J. Combin.* **17**, 2/3 (1996), 209–221.

[48] S. M. Jacob, The enumeration of the Latin rectangle of depth three, *Amer. J. Math.* **31** (1930), 329–354.

[49] N. Kahale und L. J. Schulman, Bounds on the chromatic polynomial and on the number of acyclic orientations of a graph, *Combinatorica* **16**, 3 (1996), 383–397.

[50] A. B. Kahn, Topological sorting of large networks, *Comm. ACM* **5** (1962), 558–562.

[51] R. M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, R. E. Miller und J. W. Thatcher, eds., New York, 1972, Plenum, 85–103.

[52] S. M. Kerawala, The enumeration of the Latin rectangle of depth three by means of difference equation, *Bull. Calcutta Math. Soc.* **33** (1941), 119–127.

[53] S. M. Kerawala, The asymptotic number of three-deep Latin rectangles, *Bull. Calcutta Math. Soc.* **39** (1947), 71–72.

[54] G. Kirchhoff, Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird, *Ann. Phys. Chem.* **72** (1847), 497–508.

[55] M. Kleinau, *Zur Struktur von Shop-Scheduling-Problemen: Anzahlprobleme, Reduzierbarkeit und Komplexität*, Dissertation, Technische Universität "Otto von Guericke" Magdeburg, 1993.

[56] U. Kleinau, *Zur Struktur und Lösung verallgemeinerter Shop-Scheduling-Probleme*, Dissertation, Technische Universität "Otto von Guericke" Magdeburg, 1993.

[57] R. E. Ladner, On the structure of polynomial time reducibility, *J. Assoc. Comput. Mach.* **22** (1975), 155–171.

[58] T.-C. Lai, Y. N. Sotskov, N. Y. Sotskova, und F. Werner, Optimal makespan scheduling with given bounds of processing times, *Math. Comput. Modelling* **26**, 3 (1997), 67–86.

[59] D. J. Lasser, Topological sorting of a list of randomly-numbered elements of a network, *Comm. ACM* **4** (1961), 12.

[60] C. F. Laywine und G. L. Mullen, *Discrete mathematics using Latin squares*, John Wiley & Sons Inc., New York, 1998. A Wiley-Interscience Publication.

[61] J. Light, F. W., A procedure for the enumeration of $4 \times n$ Latin rectangles, *Fibonacci Quart.* **11**, 3 (1973), 241–246.

[62] N. Linial, Hard enumeration problems in geometry and combinatorics, *SIAM J. Algebraic Discrete Methods* **7**, 2 (1986), 331–335.

[63] G. S. Lueker und K. S. Booth, A linear time algorithm for deciding interval graph isomorphism, *J. Assoc. Comput. Mach.* **26**, 2 (1979), 183–195.

[64] E. M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, *J. Comput. Syst. Sci.* **25**, 1 (1982), 42–65.

[65] R. M. McConnell und J. P. Spinrad, Linear-time modular decomposition and efficient transitive orientation of comparability graphs, in *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Arlington, VA, 1994)*, New York, 1994, ACM, 536–545.

[66] R. M. McConnell und J. P. Spinrad, Modular decomposition and transitive orientation, Preprint-Reihe Mathematik 475, Technische Universität Berlin, Fachbereich 3, 1995.

[67] R. M. McConnell und J. P. Spinrad, Linear-time transitive orientation, in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, LA, 1997)*, New York, 1997, ACM, 19–25.

[68] B. D. McKay und E. Rogoyski, Latin squares of order 10, *Electron. J. Combin.* **2** (1995), Note 3, approx. 4 pp. (electronic).

[69] B. D. McKay und I. M. Wanless, Maximising the permanent of $(0,1)$-matrices and the number of extensions of Latin rectangles, *Electron. J. Combin.* **5**, 1 (1998), Research Paper 11, 20 pp. (electronic).

[70] G. L. Miller, Graph isomorphism, general remarks, *J. Comput. System Sci.* **18** (1979), 128–142.

[71] G. L. Mullen und D. Purdy, Some data concerning the number of Latin rectangles, *J. Combin. Math. Combin. Comput.* **13** (1993), 161–165.

[72] J. R. Nechvatal, Asymptotic enumeration of generalized Latin rectangles, *Utilitas Math.* **20** (1981), 273–292.

[73] P. M. Neumann, A lemma that is not Burnside's, *Math. Sci.* **4** (1979), 133–141.

[74] I. N. Ponomarenko, Polynomial time algorithms for recognizing and isomorphism testing of cyclic tournaments, *Acta Appl. Math.* **29**, 1/2 (1992), 139–160.

[75] D. B. Porter, The Gantt chart as applied to production scheduling and control, *Naval Res. Logist. Quart.* **15** (1968), 311–317.

[76] C. R. Pranesachar, Enumeration of Latin rectangles via SDR's, in *Combinatorics and Graph Theory*, S. B. Rao, ed., vol. 885 of Lecture Notes in Math., Springer, Berlin, 1981, 380–390.

[77] R. C. Read und W. T. Tutte, Chromatic polynomials, in *Selected topics in graph theory, 3*, Academic Press, San Diego, CA, 1988, 15–42.

[78] L. Rédei, Ein kombinatorischer Satz, *Acta Litt. Sci. Szeged* **7** (1934), 39–43.

[79] M. Rezaie, Chromatic polynomial of Cartesian product of graphs, in *Proceedings of the 28th Annual Iranian Mathematics Conference, Part 1 (Tabriz, 1997)*, Tabriz Univ., Tabriz, 1997, 447–450.

[80] J. Riordan, Three-line Latin rectangles, *Amer. Math. Monthly* **51** (1944), 450–452.

[81] J. Riordan, Three-line Latin rectangles II, *Amer. Math. Monthly* **53** (1946), 18–20.

[82] J. Riordan, A recurrence relation for three-line Latin rectangles, *Amer. Math. Monthly* **59** (1952), 159–162.

[83] G.-C. Rota, On the foundations of combinatorial theory. I: Theory of Möbius functions, *Z. Wahrsch. Verw. Gebiete* **2** (1964), 340–368.

[84] B. Roy und B. Sussmann, Les problèmes d'ordonnancement avec contraintes disjonctives, Note DS No.9 bis, Montrouge, 1964.

[85] J. Y. Shao und W. D. Wei, A formula for the number of Latin squares, *Discrete Math.* **110**, 1-3 (1992), 293–296.

[86] K. Simon, An improved algorithm for transitive closure on acyclic digraphs, *Theoret. Comput. Sci.* **58** (1988), 325–346.

[87] K. Simon, *Effiziente Algorithmen für perfekte Graphen*, Teubner, Stuttgart, 1992.

[88] I. Skau, A note on the asymptotic number of Latin rectangles, *European J. Combin.* **19**, 5 (1998), 617–620.

[89] Y. N. Sotskov, Stability of an optimal schedule, *European J. Oper. Res.* **55** (1991), 91–102.

[90] Y. N. Sotskov, V. S. Tanaev, und F. Werner, Stability radius of an optimal schedule: a survey and recent developments, in *Industrial applications of combinatorial optimization*, Kluwer Acad. Publ., Dordrecht, 1998, 72–108.

[91] J. Spinrad, On comparability and permutation graphs, *SIAM J. Comput.* **14**, 3 (1985), 658–670.

[92] R. P. Stanley, Acyclic orientations of graphs, *Discrete Math.* **5** (1973), 171–178.

[93] C. M. Stein, Asymptotic evaluation of the number of Latin rectangles, *J. Combin. Theory Ser. A* **25** (1978), 38–49.

[94] V. S. Tanaev, Y. N. Sotskov, und V. A. Strusevich, *Scheduling theory. Multi-stage systems*, Kluwer Academic Publishers Group, Dordrecht, 1994. Translated and revised from the 1989 Russian original by the authors.

[95] T. Tautenhahn, *Open-Shop-Probleme mit Einheitsbearbeitungszeiten*, Dissertation, Otto-von-Guericke-Universität Magdeburg, 1993.

[96] T. Tautenhahn, Irreducible sequences - an approach to interval edge colouring trees, Preprint No. OR83, Faculty of Mathematical Studies, University of Southampton, 1996.

[97] T. Tautenhahn und P. Willenius, Irreducibility and unavoidability of sequences, Preprint, Otto-von-Guericke-Universität Magdeburg, 1999. To appear.

[98] J. Valdes, R. E. Tarjan, und E. L. Lawler, The recognition of series parallel digraphs, *SIAM J. Comput.* **11**, 2 (1982), 298–313.

[99] L. G. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.* **8**, 2 (1979), 189–201.

[100] L. G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput.* **8**, 3 (1979), 410–421.

[101] J. H. van Lint und R. M. Wilson, *A Course in Combinatorics*, University Press, Cambridge, 1992, ch. 17, 157–171.

[102] K. P. Vo, Graph colorings and acyclic orientations, *Linear and Multilinear Algebra* **22**, 2 (1987), 161–170.

[103] E. M. Wright, Burnside's lemma: A historical note, *J. Combin. Theory Ser. B.* **30** (1981), 89–90.

[104] K. Yamamoto, An asymptotic series for the number of three-line Latin rectangles, *J. Math. Soc. Japan* **1** (1949), 226–241.

[105] K. YAMAMOTO, On the asymptotic number of Latin rectangles, *Japan. J. Math.* **21** (1951), 113–119.

Index

Lebenslauf

Martin Harborth, geboren am 21. Oktober 1968 in Braunschweig (Deutschland).

Schulbesuch

1975-1981	Grundschule Schuntersiedlung und Orientierungsstufe Nibelungenschule, Braunschweig
1981-1988	Gymnasium Martino-Katharineum, Braunschweig
1988	Abiturprüfung bestanden am 18. Mai

Studium

WS 1988/89	Beginn des Studiums an der Technischen Universität Braunschweig, Studiengang Mathematik und Chemie für das höhere Lehramt
WS 1989/90	Wechsel zum Studiengang Mathematik-Diplom, Nebenfach Informatik
1994	Diplomprüfung mit der Gesamtnote „mit Auszeichnung“ abgelegt; TU Braunschweig, 24. Oktober

Studentische Nebentätigkeiten

1990-1992	Betreuung von Übungen als Hilfsassistent; TU Braunschweig
1992-1995	Entwicklung und Wartung von Datenbanken und Fachinformationssystemen in der Bibliothek der mathematischen Institute; TU Braunschweig
seit 1995	Verwaltung und Betreuung der WWW-Internet-Präsentation der Fakultät für Mathematik, Pflege der Datenbank der Fakultätsbibliothek; Otto-von-Guericke-Universität Magdeburg, seit dem 1. Juli

Promotion

1995-1997	Stipendium zur Graduiertenförderung des Landes Sachsen-Anhalt; Otto-von-Guericke-Universität Magdeburg, 1. April 1995 bis 30. September 1997
seit 1997	Wissenschaftlicher Mitarbeiter im Rahmen des vom Land Sachsen-Anhalt finanzierten Projekts „Lateinische Rechtecke in der Schedulingtheorie“; Otto-von-Guericke-Universität Magdeburg, seit dem 1. Oktober