Praktische Job-Shop Scheduling-Probleme

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Dipl.-Math. André Henning,
geboren am 12.5.1971 in Eisfeld

Jena, August 2002

Gutachter

1. Prof. Dr. Ingo Althöfer, Jena
2. Prof. Dr. Lorenz Hempel, Weimar
3. Prof. Dr. Heidemarie Bräsel, Magdeburg

Tag der letzten Prüfung des Rigorosums: 18. Dezember 2002

Tag der öffentlichen Verteidigung: 8. Januar 2003

Vorwort

Im September 1997 fand in Jena das Symposium über Operations Research statt. Prof. Jan Karel Lenstra von der Technischen Universität Eindhoven hielt den Hauptvortrag mit dem Titel *„Computing Near-Optimal Schedules“*. In diesem Vortrag kündigte er den *„Whizzkids“* Wettbewerb an, der im Oktober und November 1997 ausgetragen wurde. Die Aufgabe des Wettbewerbes war es, eine möglichst gute Lösung für eine Instanz des Mixed-Job Problems zu finden. Die Zielfunktionen waren der Makespan und die Summe der Durchlaufzeiten der Jobs. Diese Zielfunktionen sollten lexikographisch optimiert werden. Der Vortrag und die Aufgabe faszinierten mich und wurden die ersten Berührungspunkte mit dem Scheduling.

Zu dieser Zeit war ich Mitarbeiter am Lehrstuhl für mathematische Optimierung von Prof. Ingo Althöfer. Zu den dortigen Forschungs-Interessen gehören Mehrheitssysteme in der kombinatorischen Optimierung (siehe [49]) und das Design und die Analyse von Entscheidungs-Unterstützungs-Systemen. Prof. Ingo Althöfer möchte ich für die Motivation der Forschung auf diesem Gebiet und für die Betreuung der vorliegenden Arbeit herzlich danken.

Im April 1999 bot sich mir die Möglichkeit der Mitarbeit im Projekt „Effektive interaktive Entscheidungs-Unterstützung“ bei Prof. Lorenz Hempel an der Universität Weimar. Dies war der eigentliche Startpunkt für die Forschung auf dem Gebiet des Schedulings und der Produktionsplanung, welche schließlich zu der vorliegenden Arbeit führte.

Mein Dank gilt den Kollegen, die mit Diskussionen und wertvollen Anregungen meine Forschung beeinflusst und unterstützt haben. Neben Ingo Althöfer sind Lorenz Hempel, Claus Rose und Stefan Schwarz besonders hervorzuheben. Weiterhin möchte ich dem Rechenzentrum der Fakultät für Mathematik und Informatik und dem Institut für Angewandte Mathematik der Friedrich-Schiller-Universität Jena sowie dem Institut für Mathematik und Physik der Bauhaus-Universität Weimar danken. Die Rechentechnik an diesen Einrich-

tungen wurde von mir während des Entstehens dieser Arbeit intensiv genutzt. Ferner gebührt mein besonderer Dank Claus Rose und Hagen Held für das intensive Korrekturlesen der Arbeit.

Jena und Weimar, August 2002

Inhaltsverzeichnis

Einleitung

Scheduling hat sich in den letzten Jahren zu einem „hot spot“ in der Forschung auf dem Gebiet der Angewandten Mathematik und des Operations Research entwickelt. Gerade in das klassische Job-Shop Scheduling-Problem $J||C_{max}$ wurde viel Forschungsarbeit und auch Rechenzeit investiert. Dieses Scheduling-Problem bietet eine gute Basis für die Modellierung realistischer Produktionsvorgänge mit weiteren Nebenbedingungen und Verallgemeinerungen. Die Optimierung dieser verallgemeinerten Job-Shop Scheduling-Probleme auch bezüglich multikriterieller Anforderungen ist Gegenstand dieser Arbeit. Zur Klassifikation der Scheduling-Probleme wird in dieser Arbeit die $\alpha|\beta|\gamma$-Notation von Graham *et al.* verwendet, wie sie in [5] und [10] beschrieben wird.

Im ersten Kapitel werden das Modell und die Lösungsrepräsentationen der Scheduling-Probleme beschrieben. Des Weiteren werden Verallgemeinerungen für das klassische Job-Shop-Problem eingeführt. Zu den verallgemeinerten Reihenfolgerestriktionen gehören Reihenfolgebeziehungen zwischen Jobs und Mixed-Job-Probleme. Die Reihenfolgebeziehungen zwischen Jobs modellieren Montageaufträge in der Produktion. Bei den Mixed-Job-Problemen weicht die feste Ordnung der Vorgänge in den Jobs einer teilweisen Ordnung mit parallelen Vorgängen. Als weitere zeitliche Nebenbedingung werden Fälligkeitstermine für die Aufträge betrachtet. Auf den Maschinen werden deterministische Nichtverfügbarkeitsintervalle eingeführt, um ein Schichtmodell in der Produktion nachzubilden. Schließlich werden die Scheduling-Probleme mit erneuerbaren Ressourcen erweitert. Diese verallgemeinerten Scheduling-Probleme werden hier als **praktische Job-Shop Scheduling-Probleme** bezeichnet.

Die Zielfunktion für das klassische deterministische Job-Shop-Problem ist die Gesamtbearbeitungszeit, d.h. der Makespan C_{max}. In der Praxis sind weitere Zielfunktionen von Bedeutung. Zu diesen gehören Zielfunktionen wie

die Summe der Fertigstellungszeiten $\sum C_j$, die Summe der Durchlaufzeiten $\sum F_j$, die Summe der Verspätungen $\sum T_j$, die maximale absolute Terminabweichung $|L|_{max}$, die Anzahl der verspäteten Aufträge $\sum U_j$, die Anzahl der Werkzeugwechsel und die Auslastung der Maschinen. Von diesen Zielfunktionen sind meist mehrere gleichzeitig relevant. Deshalb sind multikriterielle Zielstellungen eine weitere Eigenschaft der praktischen Job-Shop Scheduling-Probleme.

Ein Ziel der Arbeit ist es, Algorithmen zu entwickeln, die für Instanzen der praktischen Job-Shop-Probleme mit mehreren Zielfunktionen die Menge der nichtdominierten Lösungen approximiert. Es werden Heuristiken benötigt, die Probleme mit allen der genannten Zielfunktionen und allen Verallgemeinerungen optimieren können. Im zweiten Kapitel werden hierfür Lokale-Suche-Heuristiken wie Tabu-Suche, Simulated Annealing und deterministische Threshold-Algorithmen entwickelt. Ein Hauptproblem bei der Entwicklung der Heuristiken waren die unterschiedlichen Strukturen der Scheduling-Probleme. So konnten keine Nachbarschaftsstrukturen basierend auf dem kritischen Pfad verwendet werden. Als Ausweg bot sich die Verwendung von Mehrheitsansätzen an, um die Struktur vorhandener Lösungen in den Algorithmen zu nutzen. Hier wurde die Mittelwertbildung als Rekombinationsoperator mit den Lokale-Suche-Heuristiken zu einem Hybrid Algorithmus — der **genetischen lokalen Suche** — kombiniert.

In diesem Kapitel werden die Algorithmen untersucht und zur Vergleichbarkeit mit anderen Heuristiken an aus der Literatur bekannten 241 Standard Job-Shop-Benchmark-Instanzen ($J||C_{max}$) getestet. Von diesen gut untersuchten Instanzen wurden bei 78 die oberen Schranken für den Makespan verbessert und bei weiteren 140 Instanzen Lösungen für die bekannten oberen Schranken gefunden. Bei den restlichen 23 Benchmark-Instanzen lag die Abweichung zur oberen Schranke bei unter 0.4 Prozent. Weiterhin wurden 50 Job-Shop-Benchmark-Instanzen mit maximaler Terminabweichung als Zielfunktion verwendet ($J||L_{max}$). Bei diesen wurden alle bekannten oberen Schranken verbessert. Für die praktischen Job-Shop-Probleme wurden eigene Benchmark-Instanzen generiert. Der hierzu entwickelte Generator wird beschrieben. Für die Benchmark-Instanzen wurden obere Schranken für den Makespan, die totale Durchlaufzeit der Aufträge, die Summe der Fertigstellungszeiten, die maximale absolute Terminabweichung und die totale Verspätung berechnet.

Für das Verständnis der Heuristiken wird weiterhin die Struktur des Lösungsraumes untersucht. Es werden Abstandsmaße zwischen Lösungen im Lösungs-

raum eingeführt und die Existenz einer Big-Valley-Struktur untersucht.

Im dritten Kapitel werden die Lokale-Suche-Heuristiken und die genetische lokale Suche für multikriterielle Probleme erweitert. Im Gegensatz zu Job-Shop-Problemen mit genau einer Zielfunktion erzeugen die Optimierungsheuristiken im multikriteriellen Fall nicht eine beste Lösung, sondern eine Menge von potentiellen Pareto-Lösungen. Dies wird für den Entscheider, der aus dieser Menge die praktikabelste Lösung auswählen muss, problematisch, da der Pool der Pareto-Lösungen sehr umfangreich werden kann und die Dimensionen bei realistischen Instanzen sehr groß sind. Visualisierungen sind im Sinne einer Entscheidungs-Unterstützung nur bei relativ kleinen Anzahlen von Lösungen hilfreich. Aus diesem Grund muss die Anzahl der Lösungen reduziert werden. Da die Diversität der ausgewählten Lösungen für die Entscheidungs-Unterstützung von Bedeutung ist, werden für diese Reduktion Clusterungsalgorithmen auf Basis der eingeführten Abstandsmaße im Lösungsraum verwendet und untersucht.

Kapitel 1

Grundlagen

Das Scheduling ist ein Teilgebiet des Operations Research und umfasst seinerseits viele Spezialisierungsrichtungen. Die Anwendungsgebiete reichen vom Crew-Scheduling in der Luftfahrtindustrie über die Erstellung von Stundenplänen in Schulen und Universitäten bis zum Projektscheduling in der Bau- oder Softwareindustrie. Dazu gehört ebenso das Scheduling von Prozessen auf Computern wie die Maschinenbelegungsplanung in der Industrie. Einen Überblick über die verschiedenen Scheduling-Probleme findet man in [5], [10] und [16].

Das in dieser Arbeit behandelte Job-Shop Scheduling-Problem und seine Verallgemeinerungen gehören zur Maschinenbelegungsplanung. Maschinenbelegungsprobleme befassen sich mit der Zuordnung von Aufträgen zu Maschinen und umgekehrt. Dabei werden vorgegebene Zielfunktionen und Nebenbedingungen beachtet. In dieser Arbeit wird von einem deterministischen Modell für die praktischen Job-Shop-Probleme ausgegangen. Das heißt, alle Eingabedaten wie Bearbeitungszeiten, Maschinenverfügbarkeiten oder Reihenfolgebeziehungen werden als bekannt und deterministisch vorausgesetzt.

1.1 Modell

1.1.1 Das „klassische“ Job-Shop Scheduling-Problem

Das Job-Shop Scheduling-Problem, im Folgenden als JSP bezeichnet, ist formal wie folgt definiert: Das JSP besteht aus einer Menge von Jobs $\mathscr{J} = \{J_j\}_{j=1}^{n}$, die auf einer Menge von Maschinen $\mathscr{M} = \{M_i\}_{i=1}^{m}$ bearbeitet werden müssen. Jeder Job J_j besteht aus einer Menge von m_j Vorgängen $\mathscr{T} = \{T_{j1}, T_{j2}, \ldots, T_{jm_j}\}$, die auch als Tasks oder Operationen bezeichnet werden. Die zu einem Job gehörenden Tasks müssen in einer vorgegebenen festen Reihenfolge auf den Maschinen bearbeitet werden. Es gibt insgesamt N Vorgänge, $|\mathscr{T}| = N = \sum_{j=1}^{n} m_j$. Der Vorgang T_{ji} gehört zu Job J_j und muss auf Maschine M_i für eine ununterbrochene Dauer p_{ji} bearbeitet werden. Das bedeutet, dass die Abarbeitung eines Tasks nicht unterbrochen und zu einem späteren Zeitpunkt wieder aufgenommen werden darf. Jeder Job hat seine eigene, von den anderen Jobs unabhängige Maschinenreihenfolge und durchläuft jede Maschine höchstens einmal. Jede Maschine kann nur einen Vorgang gleichzeitig bearbeiten. Zwei Vorgänge des gleichen Jobs können nicht simultan bearbeitet werden. Ein zulässiger **Maschinenbelegungsplan (engl. Schedule)** ist durch Startzeiten $s_{ji} \geq 0$ für alle Vorgänge T_{ji} definiert, so dass alle obigen Nebenbedingungen erfüllt sind. Gesucht ist ein Schedule, der eine gegebene Zielfunktion minimiert. Beim klassischen Job-Shop-Problem ist der Makespan $C_{max} = \max(s_{ji} + p_{ji})$ die Zielfunktion.

Für das JSP gelten des Weiteren die folgenden Annahmen und Voraussetzungen:

- Die Bearbeitungszeit jedes Vorgangs hängt nicht von der Bearbeitungsreihenfolge auf den Maschinen ab.
- Es werden keine Transportzeiten zwischen den Maschinen und keine Rüstzeiten betrachtet.
- Alle Maschinen sind vom Zeitpunkt 0 an immer verfügbar. Die Jobs können zum Zeitpunkt 0 beginnen.
- Es existieren keine parallelen Maschinen oder Maschinengruppen.
- Alle Jobs haben die gleiche Priorität.
- Es gibt keine Freigabe- oder Fälligkeitstermine für die Jobs. Leerzeiten auf den Maschinen und Wartezeiten in den Jobs sind zulässig.

Die Dimension einer JSP-Instanz wird mit $n \times m$ bezeichnet. Bei den aus der Literatur ([14],[57]) bekannten Benchmark-Instanzen wird die Anzahl der Tasks mit $N = nm$ angenommen. Hierbei ist $m_j = m$ für alle $J_j \in \mathscr{J}$.

1.1.2 Lösungsrepräsentationen

Als Beispiel wird im Weiteren die Benchmark-Instanz FT06 von Fisher und Thompson (1963) verwendet. Tabelle 1.1 zeigt die Daten für die Instanz FT06. Für jeden Job werden die Vorgangsnummern, die Maschinenreihenfolge und die Bearbeitungszeiten angegeben. Jeder Vorgang bekommt eine eindeutige Nummer als Index aus $\{1, \ldots, N\}$ zugeordnet. Der erste Vorgang im ersten Job bekommt die Nummer 1, der zweite Vorgang im ersten Job die Nummer 2, bis zum letzten Vorgang im ersten Job, dem die Nummer m_1 zugeordnet wird. Der erste Vorgang im zweiten Job bekommt die Nummer $m_1 + 1$ und so weiter. Bei den aus der Literatur bekannten Job-Shop-Benchmarks laufen alle Jobs **genau** einmal auf allen Maschinen. Der $i-te$ Vorgang im $j-ten$ Job hat demnach die Nummer $m(j-1)+i$. Die jedem Vorgang

	T_i, M_i, p_i	T_i, M_i, p_i	T_i, M_i, p_i	T_i, M_i, p_i	T_i, M_i, p_i	T_i, M_i, p_i
J_1	1, 3, 1	2, 1, 3	3, 2, 6	4, 4, 7	5, 6, 3	6, 5, 6
J_2	7, 2, 8	8, 3, 5	9, 5,10	10, 6,10	11, 1,10	12, 4, 4
J_3	13, 3, 5	14, 4, 4	15, 6, 8	16, 1, 9	17, 2, 1	18, 5, 7
J_4	19, 2, 5	20, 1, 5	21, 3, 5	22, 4, 3	23, 5, 8	24, 6, 9
J_5	25, 3, 9	26, 2, 3	27, 5, 5	28, 6, 4	29, 1, 3	30, 4, 1
J_6	31, 2, 3	32, 4, 3	33, 6, 9	34, 1,10	35, 5, 4	36, 3, 1

Tabelle 1.1: *Vorgangsnummern, Bearbeitungszeiten und -reihenfolgen der Benchmark-Instanz FT06 mit 6 Jobs und 6 Maschinen.*

eindeutig zugeordnete Nummer wird im Folgenden auch als Bezeichnung des Vorgangs verwendet.

Modellierung als gerichteter Graph

Im Operations Research und in Optimierungsalgorithmen findet das Konzept des **disjunktiven Graphen** $G = (V, A, E)$ die häufigste Verwendung in der Modellierung des Problems und in den Optimierungsalgorithmen. Die Menge V der Knoten des Graphen setzt sich aus der Menge $\mathscr{T}$ der Vorgänge des

JSP und zwei fiktiven Knoten, der Quelle 0 und der Senke $*$, zusammen. Es gilt also $V = \mathscr{T} \cup \{0, \star\}$ und $|V| = N + 2$. Der Index der Quelle ist 0 und der Index der Senke $N + 1$. Bei den klassischen Job-Shop-Problemen entspricht das $nm+1$. Der Graph ist knotengewichtet, wobei die Knoten mit der Bearbeitungsdauer p_j gewichtet sind. Das Gewicht der Quelle und der Senke ist 0 ($p_0 = p_{N+1} = 0$).

A ist die Menge der gerichteten Kanten, welche die Reihenfolgebeziehungen in den Jobs repräsentieren. $(i, j) \in A$ bedeutet, dass i der unmittelbare Vorgänger von j innerhalb eines Jobs ist. Diese Kanten werden auch als **konjunktive** Kanten bezeichnet. Von der Quelle werden gerichtete Kanten zu den jeweils ersten Vorgängen in den Jobs eingefügt. Weiterhin führt von jedem letzten Task eines Jobs eine Kante zur Senke. Damit lässt sich 0 als Start- und $*$ als Endvorgang interpretieren. Diese Kanten gehören ebenfalls zur Menge A.

Für je zwei Tasks, die auf der gleichen Maschine bearbeitet werden müssen, wird eine ungerichtete oder **disjunktive** Kante in den Graphen eingefügt. E ist die Menge dieser disjunktiven Kanten. Diese Kanten repräsentieren die Kapazitätsbeschränkungen der Maschinen. $E_k, k \in 1, \ldots, m$, ist die Menge der zur Maschine k gehörenden ungerichteten Kanten. Abbildung 1.1 zeigt den zum Beispiel FT06 gehörenden disjunktiven Graphen, wobei aus Übersichtsgründen nur die disjunktiven Kanten für Maschine 1 eingezeichnet wurden.

Auf der Basis des Konzeptes des disjunktiven Graphen kann das JSP wie folgt beschrieben werden:
Für jede ungerichtete Kante ist eine der beiden möglichen Orientierungen zu wählen, so dass ein kreisfreier gerichteter Graph entsteht. Unter allen solchen Graphen wird derjenige gesucht, bei dem der längste Weg von der Quelle zur Senke minimal ist.

Die Startzeitpunkte der Vorgänge werden mit $s_j, j \in V$, bezeichnet. Die Startzeitpunkte s_j der Vorgänge werden durch den längsten Weg von der Quelle zu T_j bestimmt. Der Startzeitpunkt s_0 der Quelle ist 0. Das diskrete

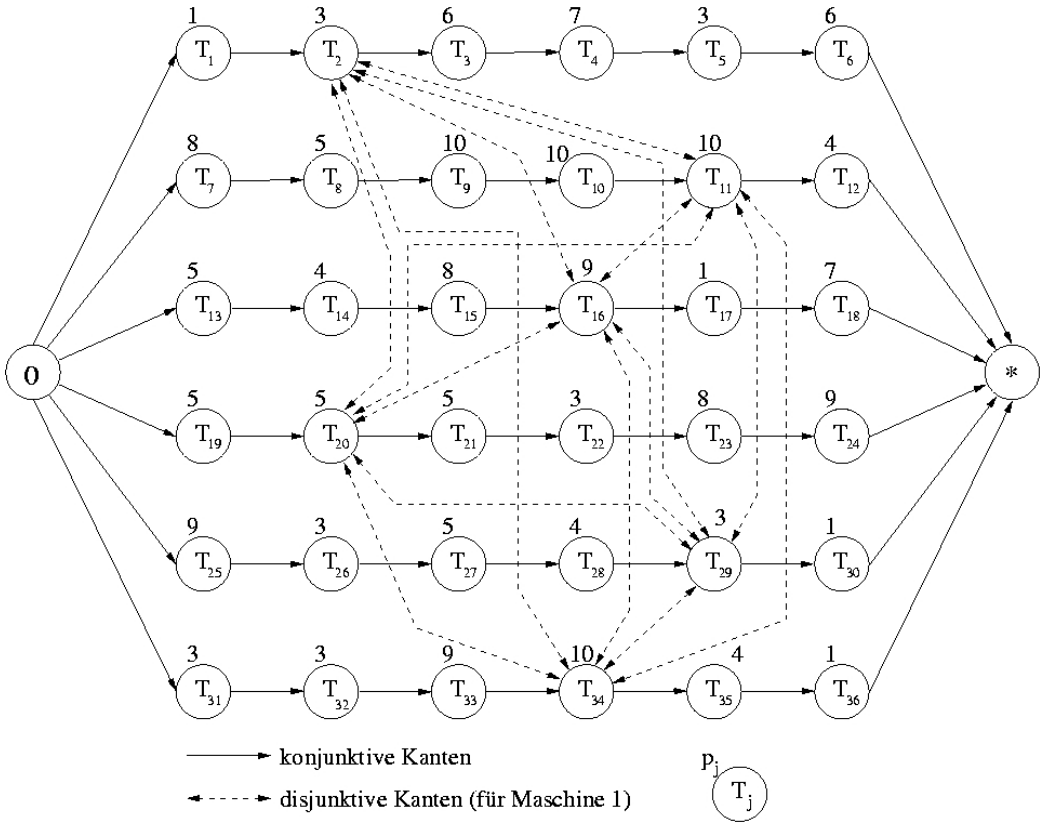

Abbildung 1.1: *Modellierung als Graph am Beispiel FT06*

Optimierungsproblem kann formal wie folgt formuliert werden:

$minimiere\ s_*$	$unter\ den\ Nebenbedingungen$
$s_j - s_i \geq p_i$	$für\ alle\ i,j \in V, (i,j) \in A$
	$(Reihenfolgebeziehungen)$
$s_j \geq 0$	$für\ alle\ j \in V$
	$(frühester\ Starttermin)$
$s_j - s_i \geq p_i \vee s_i - s_j \geq p_j$	$für\ alle\ i,j \in V, i \neq j, (i,j) \in E_k, k \in 1,\ldots,m$
	$(Kapazitätsbeschränkung)$

Ein längster Weg von der Quelle zur Senke im Graphen wird als **kritischer Pfad** bezeichnet. Die Tasks auf dem kritischen Pfad werden **kritische Vorgänge** genannt. Eine disjunktive Kante ist **fixiert**, wenn sie mit eine der beiden Orientierungen gerichtet ist. Eine Menge fixierter Kanten heißt **Selektion** S. Bei einer **vollständigen Selektion** sind alle disjunktiven Kanten gerichtet. Eine Selektion S_k der Kanten aus E_k gibt die Reihenfolge der Jobs auf einer Maschine an. Der zu einer vollständigen Selektion korrespondieren-

de Graph $G_S = (V, A \cup S)$ definiert eine zulässige Lösung, falls folgendes gilt:

- Alle disjunktiven Kanten sind gerichtet.
- Der resultierende Graph $G_S = (V, A \cup S)$ ist kreisfrei.

Eine Selektion S ist kreisfrei, falls der dazugehörige Graph G_S kreisfrei ist. Aufgrund einer Selektion lassen sich im disjunktiven Graphen **Bereitstellungszeitpunkte** oder **Vorlaufzeiten** a_j und **Nachlaufzeiten** n_j für alle Tasks T_j ermitteln. Der Bereitstellungszeitpunkt a_j für den Vorgang T_j ist die Länge des längsten Weges von der Quelle nach T_j im Graphen. Die Nachlaufzeit n_j ist die Länge des längsten Weges von T_j zur Senke. Abbildung 1.2 zeigt

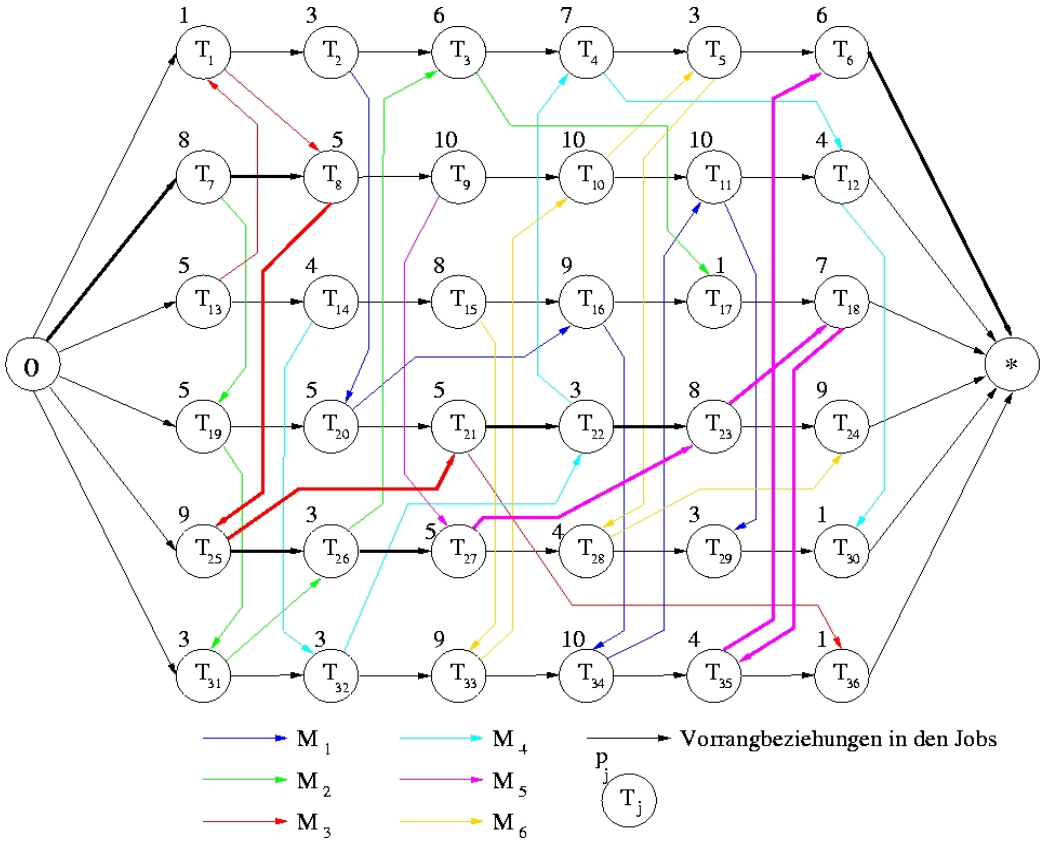

Abbildung 1.2: *Optimale Lösung des Benchmarks FT06 als Graph*

einen zulässigen Graphen für das Beispiel FT06. Auf redundante, transitive disjunktive Kanten wurde verzichtet. Jeder Vorgang hat damit höchstens

2 Vorgänger und 2 Nachfolger, jeweils einen Maschinen- und einen Jobvorgänger und -nachfolger. Der Graph zeigt eine optimale Lösung dieses Benchmarkproblems. Der kritische Pfad wurde mit breiteren Pfeilen markiert. Hier ist zu sehen, dass der kritische Pfad nicht eindeutig sein muss. In diesem Beispiel gibt es 2 kritische Pfade $(T_7, T_8, T_{25}, T_{26}, T_{27}, T_{23}, T_{18}, T_{35}, T_6)$ und $(T_7, T_8, T_{25}, T_{21}, T_{22}, T_{23}, T_{18}, T_{35}, T_6)$. Der Zielfunktionswert für C_{max} beträgt 55 Zeiteinheiten, falls alle Vorgänge an ihrem Bereitstellungszeitpunkt beginnen. Falls die Tasks nach ihrem Bereitstellungszeitpunkt eingeplant werden, spricht man vom passiven Scheduling. Der Ablaufplan wird dementsprechend als **passiver** Schedule bezeichnet. Eine zulässige Verringerung der Startzeitpunkte für einen oder mehrere Vorgänge ohne Änderung der Auftragsfolgen auf den Maschinen wird als **lokale Linksverschiebung** bezeichnet. Unter einer **globalen Linksverschiebung** versteht man eine Verringerung von Startzeitpunkten für einen oder mehrere Tasks, ohne dass ein Vorgang später beginnt. Ein Schedule heißt **semiaktiv**, falls es keine lokale Linksverschiebung gibt. Falls keine globale Linksverschiebung möglich ist, wird der Schedule als **aktiv** bezeichnet.

Gantt-Diagramm

Das **Gantt-Diagramm** ist die einfachste und in der Praxis am meisten verbreitete Lösungsrepräsentation. Auf der x-Achse wird die Zeit abgetragen und auf der y-Achse die Maschinen. Das Diagramm bildet die Matrix der Start- und Endzeiten der Vorgänge ab. Für jeden Vorgang wird in der dazugehörigen Maschinenzeile ein Rechteck eingetragen. Die Länge des Rechtecks entspricht der Produktionsdauer. Es werden die Nummer des Vorgangs sowie die Anfangs- und Endtermine eingetragen. Es ist natürlich auch möglich, auf der y-Achse die Jobs einzutragen. Da die Reihenfolge in den Jobs beim JSP fest steht, ist die Maschinenvariante zumindest bei klassischen Job-Shop Scheduling-Problemen gebräuchlicher. Abbildung 1.3 zeigt eine optimale Lösung der Benchmark-Instanz FT06. Die kritischen Vorgänge sind rot eingezeichnet.

1.1.3 Verallgemeinerungen und weitere Nebenbedingungen

Die Struktur des „klassischen“ Job-Shop-Problems ist sehr restriktiv. Im Gegensatz dazu werden reale Produktionsprozesse immer komplexer und benöti-

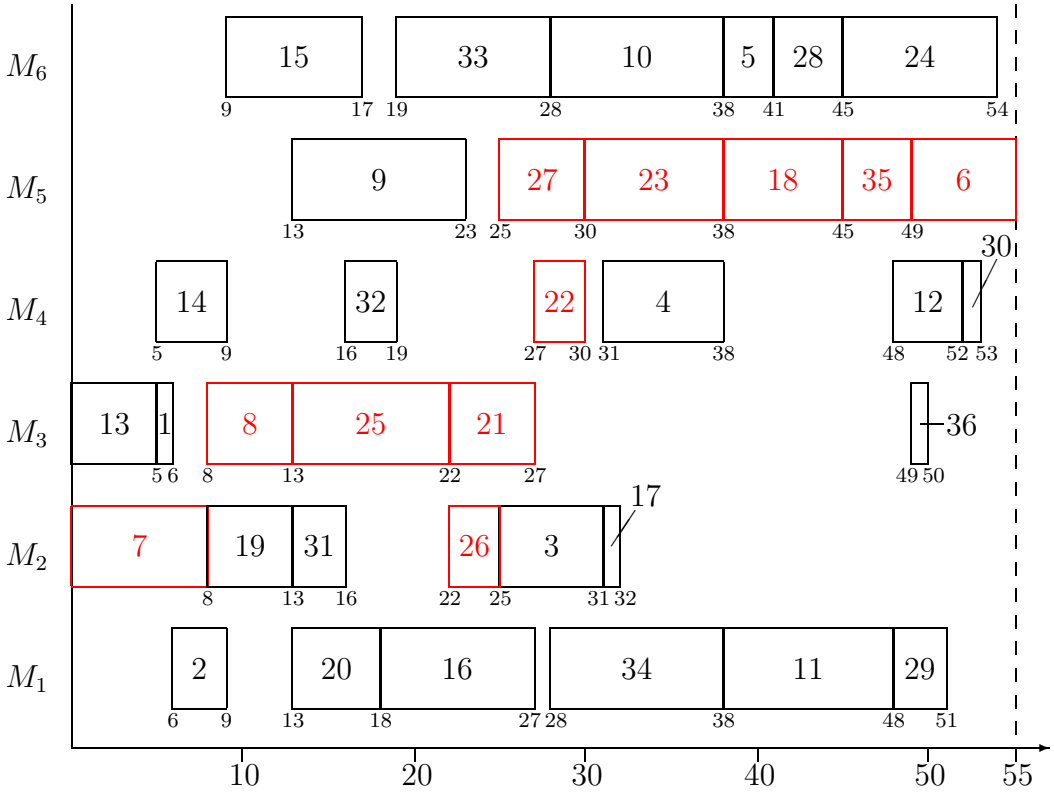

Abbildung 1.3: *Gantt-Diagramm der optimalen Lösung für FT06 mit kritischen Vorgängen (rot)*

gen flexiblere Ansätze. Im Folgenden werden verschiedene Verallgemeinerungen des Job-Shop-Problems eingeführt, um die praktischen Einsatzfähigkeiten des Job-Shop zu verbessern.

Verallgemeinerte Reihenfolgebeziehungen

Im Allgemeinen kann nicht davon ausgegangen werden, dass die technologische Reihenfolge aller Vorgänge bei der Herstellung eines Produkts fest vorgegeben und linear ist. Das Scheduling-Problem, bei dem keine Reihenfolgebeziehungen in den Jobs gegeben sind, wird als **Open-Shop Scheduling-Problem**[1] bezeichnet. Die Vorgänge eines Jobs können in beliebiger Reihenfolge bearbeitet werden, wobei weiterhin die Nebenbedingung gilt, dass die Vorgänge eines Jobs nicht simultan bearbeitet werden können. Dies wird modelliert, indem für alle Vorgänge eines Jobs, zwischen denen keine

[1] Klassifikation mit Zielfunktion Makespan ist $O||C_{max}$

Reihenfolgebeziehungen existieren, eine virtuelle Maschine eingeführt wird. Die Reihenfolge auf der virtuellen Maschine determiniert die Reihenfolge der Vorgänge des Jobs in einem Schedule. Im disjunktiven Graphenmodell werden dementsprechend ungerichtete Kanten für die virtuellen Maschinen eingefügt.

In der Praxis treten Job-Shop- und Open-Shop-Probleme in gemischter Form auf. Das bedeutet, die Vorgänge in einem Job sind nur teilweise geordnet. Diese Scheduling-Probleme werden als **Mixed-Job-** oder **DAG[2]-Job-Probleme** bezeichnet. Die Klassifikation für dieses Scheduling-Problem ist deshalb $D||C_{max}$. Vorgänge eines Jobs, zwischen denen keine Vorrangbeziehungen bestehen, werden im Weiteren als **parallele Tasks** bezeichnet. Die Menge der Maschinen $\mathscr{M}$ besteht demnach aus der Teilmenge der realen Maschinen $\mathscr{M}^r$ und der Teilmenge der virtuellen Maschinen $\mathscr{M}^v$.

$$\mathscr{M} = \{M_i\}_{i=1}^{m} = \{M_k^r\}_{k=1}^{m^r} \bigcup \{M_l^v\}_{l=m^r+1}^{m^r+m^v} \; und \; m = m^r + m^v$$

Dementsprechend setzt sich auch die Menge E der disjunktiven Kanten zusammen.

Die Paare (T_2, T_3), (T_{14}, T_{15}) und (T_{16}, T_{17}) in Abbildung 1.4 bestehen aus je zwei parallelen Vorgängen. In dem Beispiel sind nur die disjunktiven Kanten der virtuellen Maschinen eingezeichnet. Die Bearbeitungsreihenfolge der parallelen Tasks wird durch den Scheduling Algorithmus festgelegt. Da die redundanten Kanten in dem zu einer vollständigen Selektion gehörenden Graphen wieder weggelassen werden können, hat dann jeder Vorgang wieder zwei Vorgänger und Nachfolger. Als weitere Verallgemeinerung wird vereinbart, dass die Jobs auf jeder Maschine höchstens einmal bearbeitet werden müssen. Damit ist die Anzahl der Tasks in jedem Job kleiner gleich der Anzahl der Maschinen $m_j \leq m$.

Weitere Reihenfolgebeziehungen ergeben sich aus Montageaufträgen. Komplexe Endprodukte bestehen in der Praxis meist aus mehreren Baugruppen. Jede Baugruppe benötigt zur Produktion Ressourcen und hat eine vorgegebene technologische Reihenfolge. Die Produktion einer Baugruppe kann deshalb als Job im Sinne eines Mixed-Job-Problems betrachtet werden. Die gesamte Fertigung bis zum Endprodukt wird als **Produktionsauftrag** (engl. production order) bezeichnet. Ein Produktionsauftrag muss demnach mindestens aus einem, kann aber aus mehreren Jobs bestehen. Die aus dem Montageablauf resultierenden Reihenfolgebeziehungen werden durch konjunktive

[2]DAG für directed acyclic graph

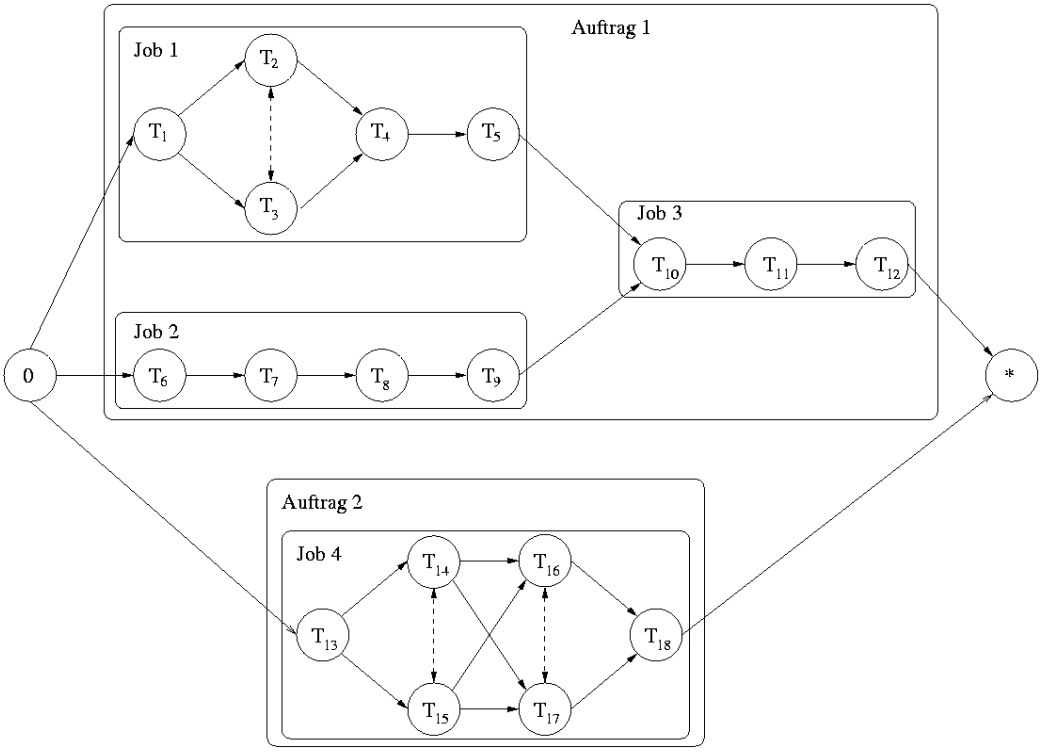

Abbildung 1.4: *Beispiel eines Mixed-Job Problems*

Kanten im gerichteten Graphenmodell eingefügt. Damit entsteht eine Baumstruktur. Im Gegensatz zu diesen konvergenten Produktionsaufträgen sind in der Praxis auch divergente Produktionsaufträge möglich. Hier werden aber nur Montageaufträge betrachtet. Die Bezeichnung hierfür ist $D|intree|C_{max}$. In der zu dieser Verallgemeinerung gehörenden Graphenrepräsentation kann der Indegree der Knoten größer als zwei sein.

Das Scheduling-Problem in Abbildung 1.4 besteht aus 2 Produktionsaufträgen und 4 Jobs. Die Bearbeitung von Job 3 kann erst beginnen, wenn Job 1 und Job 2 beendet sind. Der Vorgang T_{10} hat in einer vollständigen Selektion auch bei Nichtberücksichtigung der redundanten Kanten 3 Vorgänger.

Verfügbarkeitsintervalle auf Maschinen und Fälligkeitstermine

Die Maschinen sind im JSP immer verfügbar. Im Gegensatz dazu haben Maschinen in realen Produktionsumgebungen **deterministische Stillstands-**

zeiten. Diese Stillstandszeiten können zum Beispiel durch Schichtzeiten entstehen. So ist es möglich, dass eine Maschine im Ein-Schicht-System — und somit nur 8 Stunden pro Tag — verfügbar ist und eine Maschine im Zwei-Schicht-System dementsprechend 16 Stunden.

Jede Maschine M_i hat eine vorgegebene Schichtzeit $st_i \in \{1, \ldots, 24\}$. Eine Schichtzeit von 24 Stunden bedeutet, dass die Maschine wie im klassischen Job-Shop-Problem immer verfügbar ist. Die Schichtzeiten starten immer um 00:00 Uhr. Damit sind die Schichtintervalle von Maschinen mit kürzeren Schichtzeiten immer in denen mit längeren Schichtzeiten enthalten. Die Schichtzeiten sind somit periodisch. Die Zeiteinheit Stunde und die Periodenlänge von 24 Stunden wurden gewählt, um einen praktischen Bezug herzustellen. Damit ist auch ein Rahmen für die im Weiteren generierten Benchmarks und die Optimierungsalgorithmen gegeben. Die Wahl anderer Zeiteinheiten (Minuten, Sekunden) oder anderer Periodenlängen hat keinen Einfluss auf die Richtigkeit oder die Laufzeit der Algorithmen.

Die Bearbeitung der Tasks wird während der Stillstandszeiten gestoppt und zu Beginn des nächsten Verfügbarkeitsintervalls weitergeführt. Die Bearbeitungszeit des Vorgangs und damit das Gewicht des zugehörigen Knotens verlängert sich entsprechend um die Stillstandszeit. Weiterhin gilt aber, dass ein einmal begonnener Vorgang fertig bearbeitet werden muss. Es ist nicht möglich, einen Vorgang zu unterbrechen und mit einem anderen Vorgang auf dieser Maschine fortzufahren. Das Einführen von Verfügbarkeitsintervallen bedeutet, dass die Bearbeitungszeit eines Vorgangs im Scheduling-Algorithmus in Abhängigkeit vom Startzeitpunkt berechnet werden muss (Abschnitt 2.1.1). Die Bezeichnung für Mixed-Job Probleme mit Schichtzeiten und für Stillstandszeiten unterbrechbarer Vorgänge ist $D|resum|C_{max}$.

In realen Produktionsumgebungen müssen die Produkte an einem determinierten Zeitpunkt fertig produziert sein und an den Kunden ausgeliefert werden. Damit hat jeder Produktionsauftrag einen vorgegebenen **Fälligkeitstermin** d_j (engl. due date). Im Gegensatz zum klassischen Job-Shop-Problem, bei dem der Makespan das Zielkriterium ist, führt diese Annahme zu Zielfunktionen wie totale Verspätung und maximale Terminabweichung (Abschnitt 1.2.1).

Erneuerbare diskrete Ressourcen

Oft benötigen Vorgänge weitere Ressourcen zur Bearbeitung. Das können Werkzeuge, Paletten oder Arbeitskräfte sein. Diese Ressourcen heißen erneuerbar, da sie nach Beendigung eines Vorgangs wieder zur Verfügung stehen und nicht — wie z.B. Materialien — verbraucht werden. Der Menge $\mathscr{R} = \{R_k\}_{k=1}^{r}$ der r verschiedenen Ressourcenarten im Shop Scheduling-Problem wird der Vektor $ra = (ra_1, \ldots, ra_r), ra_k \in \mathbb{N}$ zugeordnet, der für jede Ressource die verfügbare Menge angibt.

Jedem Task wird ein Vektor $rd = (rd_1, \ldots, rd_r), rd_k \in \mathbb{N}_0, rd_k \leq ra_k$ zugeordnet. rd_k gibt die Menge der Ressource R_k an, die der Vorgang zur Bearbeitung benötigt. Die von einem Vorgang benötigten Ressourcen werden über die gesamte Dauer des Vorgangs belegt. Auch wenn die Bearbeitung eines Vorgangs durch Stillstandszeiten unterbrochen wird, bleiben die Ressourcen während dieser Zeiten belegt. Die Klassifikation für Shop Scheduling-Probleme mit erneuerbaren Ressourcen ist $D|res|C_{max}$.

Maschinen sind ebenfalls erneuerbare Ressourcen, wobei es in dem hier verwendeten Modell von jeder Maschine genau eine gibt und jeder Vorgang genau eine Maschine benötigt. Deshalb werden sie weiterhin gesondert betrachtet.

Abbildung 1.5 zeigt die Repräsentation des Ressourcenverbrauchs als Histogramm analog zum Gantt-Diagramm. Das Beispiel in Diagramm hat 4 Ressourcen, wobei der Vektor der Ressourcenverfügbarkeiten $ra = (2, 1, 2, 4)$ ist.

In einer Lösung für ein Scheduling-Problem mit diskreten erneuerbaren Ressourcen wird die Belegung der Ressourcen durch eine Histogrammfunktion $rf_k : \mathbb{R}^+ \to \{0, \ldots, ra_k\}$ mit $k = 1, \ldots, r$ beschrieben.

1.2 Zielsetzungen für Scheduling-Probleme

1.2.1 Zielfunktionen

Als Zielkriterien kommen bei praktischen Job-Shop-Problemen verschiedene Zeitgrößen und Auftragszahlen in Frage, deren Gesamtbetrag oder Maximalwert zu minimieren ist. Im Folgenden werden die für diese Arbeit relevanten

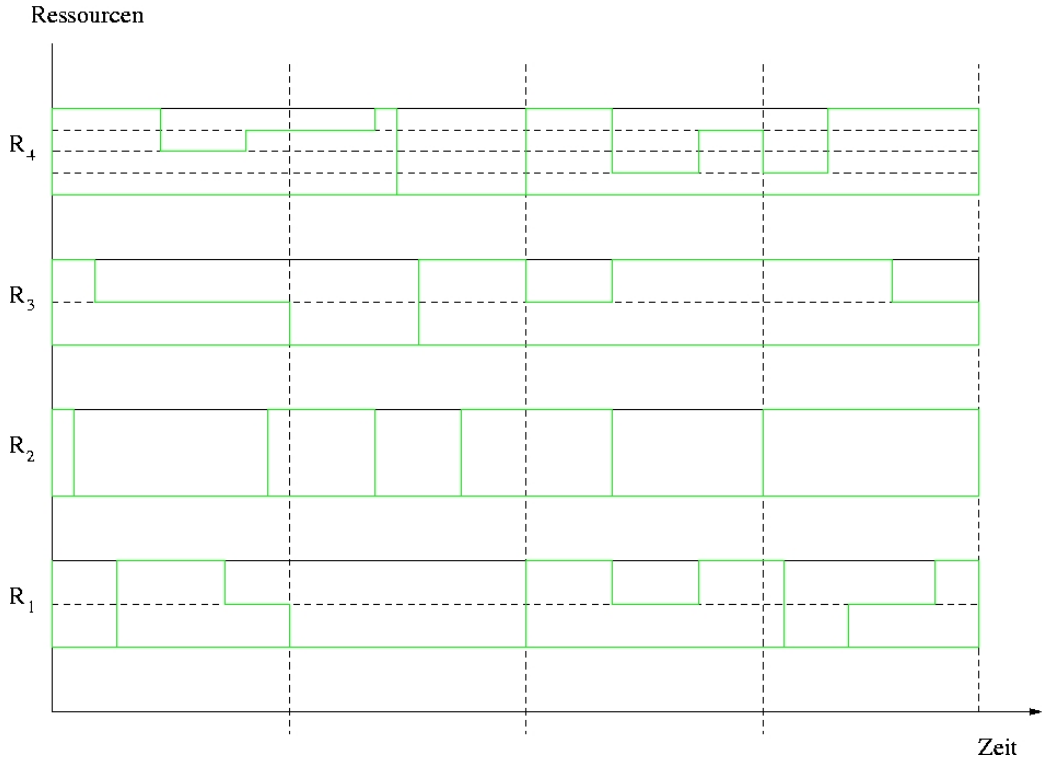

Abbildung 1.5: *Repräsentation der Ressourcenverbrauchs im Histogramm*

Begriffe und Zielfunktionen definiert.

Fertigstellungszeitpunkt
C_j ist der realisierte Fertigstellungszeitpunkt des Fertigungsauftrages j (engl. completion time). Der Startzeitpunkt des gesamten Scheduling-Problems ist 0.

Gesamtbearbeitungszeit
$C_{max} := \max_j C_j$ ist die klassische Zielfunktion der JSP. Die Gesamtbearbeitungszeit (engl. makespan) ist die Zeitspanne vom Beginn der Bearbeitung zum Zeitpunkt 0 bis zur Fertigstellung des letzten Auftrages.

Durchlaufzeit
$F_j := C_j - a_j$ ist die Durchlaufzeit (engl. flow time) des Auftrages j. Das ist die Zeitdifferenz vom Beginn a_j des ersten Vorgangs im Auftrag bis zum Fertigstellungszeitpunkt des letzten Vorgangs im Auftrag.
Als Zielkriterien sind die Summe der Fertigstellungszeiten $\sum C := \sum C_j$ und

die Summe der Durchlaufzeiten $\sum F := \sum F_j$ zu minimieren.

Terminabweichung
$L_j := C_j - d_j$ ist die Differenz zwischen realisiertem Fertigstellungszeitpunkt und due date eines Auftrages (engl. lateness). Positive L_j werden als Verspätung (engl. tardiness) bezeichnet und negative als Verfrühung (engl. earliness).
Die Verspätung ist dann $T_j := \max(0, L_j)$ und $|L_j|$ ist die absolute Terminabweichung der Auftrages j. Als zu minimierende Zielfunktionen werden die maximale Terminabweichung $L_{max} = \max_j L_j$, die maximale absolute Terminabweichung $|L|_{max} = \max_j |L_j|$ und die totale Verspätung $\sum T = \sum T_j$ betrachtet.

Verspätete Aufträge
Ein Auftrag j ist verspätet, falls $T_j > 0$ gilt. Mit Hilfe der Binärvariablen

$$u_j := \begin{cases} 1, & T_j > 0 \\ 0, & sonst \end{cases} \quad für\ j = 1, \ldots, n$$

lässt sich die Anzahl der verspäteten Aufträge als $\sum U := \sum u_j$ darstellen.

Diese Zielfunktionen lassen sich in zwei Gruppen, die regulären und die nichtregulären Zielfunktionen, einteilen. Eine Zielfunktion ist genau dann **regulär**, wenn sie minimiert werden soll und der Wert der Zielfunktion bei einer Rechtsverschiebung eines Vorgangs nicht abnehmen kann. C_{max}, L_{max}, $\sum C$ und $\sum T$ sind reguläre Zielfunktionen. $\sum F$, $|L|_{max}$ und auch die Earliness sind nichtreguläre Zielfunktionen. Durch eine Rechtsverschiebung und damit einem verzögerten Start von Vorgängen können die Zielfunktionswerte verbessert werden.

1.2.2 Ranking-Funktionen und multikriterielle Optimierung

Bei realen Shop Scheduling-Problemen wird meist die Optimierung hinsichtlich mehrerer Zielfunktionen gewünscht. Dies wirft Probleme inbesondere bei konkurrierenden Zielfunktionen wie dem Makespan C_{max} und der Durchlaufzeit $\sum F$ auf.

Ein Zugang zur multikriteriellen Optimierung besteht darin, Pareto-optimale Lösungen zu bestimmen. Der andere Zugang vergleicht Lösungen mittels Aggregation der Zielfunktionen.

Definition 1.1 *(Pareto-Dominanz)*
Sei $S = \{s_j\}_{j=1,\ldots,n}$ eine Menge von Lösungen eines Scheduling-Problems und $F = \{f_i\}_{i=1,\ldots,m}$ die Menge der relevanten Zielfunktionen. Die Lösung s_j ***dominiert*** *die Lösung s_k, $s_j \succ s_k$, falls $f_i(s_j) \leq f_i(s_k)$ für alle $i \in \{1,\ldots,m\}$ und $f_i(s_j) < f_i(s_k)$ für mindestens ein $i \in \{1,\ldots,m\}$. Falls eine Lösung von keiner anderen Lösung dominiert wird, ist sie* ***Pareto-optimal****.*

Die Menge der nichtdominierten Lösungen, die **Pareto-Menge**, wird gesucht. Da die betrachteten Scheduling-Probleme schon im Fall einer Zielfunktion NP-schwer sind, wird die Pareto-Menge durch die Optimierungsheuristiken approximiert. Dies wird als Menge der potentiellen Pareto-Lösungen bezeichnet.

Eine Möglichkeit eines Ranking von Lösungen in der multikriterielle Optimierung ist die **lexikographische Optimierung**. Sei $W = \{w_i\}_{i=1,\ldots,m}$ eine Menge von Gewichten für die Zielfunktionen aus F mit $\sum_{i=1}^{m} w_i = 1$. Die Zielfunktionen werden nach ihrem Gewicht und die Lösungen nach der Zielfunktion mit dem größten Gewicht geordnet. Falls zwei Funktionswerte gleich sind, wird nach der Zielfunktion mit dem nächstkleineren Gewicht geordnet und so weiter. Der Vorteil der lexikographischen Optimierung ist die Transitivität. Der Nachteil ist die strenge hierarchische Ordnung der Zielfunktionen. Speziell in Lokale-Suche-Heuristiken wird durch die Bevorzugung der Zielfunktion mit dem höchsten Gewicht die Suchrichtung starr vorgegeben.

Als Alternative bietet sich die Aggregation der Zielfunktionen auf einen multikriteriellen Zielfunktionswert an. Da die Zielfunktionen sehr unterschiedliche Größenordnungen haben, ist der erste Schritt eine Normierung der Zielfunktionswerte. Sei S die Menge der Lösungen, F die Menge der Zielfunktionen und W die Menge der Gewichte wie oben.

Ranking-Funktion 1: Sei π_j^i der Rang der Lösung s_j für die Zielfunktion f_i, dann ist die multikriterielle Zielfunktion F^1 wie folgt definiert:

$$\mathrm{F}^1(s_j) = \sum_{i=1}^{m} w_i \pi_j^i$$

Ranking-Funktion 2: Sei $min_{f_i} = \min_{j=1,\ldots,n} f_i(s_j)$ das Minimum aller Lösungen aus S und $max_{f_i} = \max_{j=1,\ldots,n} f_i(s_j)$ das Maximum, dann ist die multikriterielle Zielfunktion F^2 wie folgt definiert:

$$\mathrm{F}^2(s_j) = \sum_{i=1}^{m} w_i \frac{f_i(s_j) - min_{f_i}}{max_{f_i} - min_{f_i}}$$

für $max_{f_i} \neq min_{f_i}$. Bei Gleichheit wird der Quotient gleich 0 gesetzt.

Ranking-Funktion 3: Die multikriterielle Zielfunktion F^3 wird wie folgt definiert:

$$\mathrm{F}^3(s_j) = \sum_{i=1}^{m} w_i \frac{f_i(s_j) \cdot 100}{max_{f_i}}$$

für $max_{f_i} \neq 0$. Ansonsten wird der Quotient wieder 0 gesetzt.

Ranking-Funktion 4: Sei $av_{f_i} = \frac{1}{n} \sum_{j=1}^{n} f_i(s_j)$, dann ist die multikriterielle Zielfunktion F^4 wie folgt definiert:

$$\mathrm{F}^4(s_j) = \sum_{i=1}^{m} w_i \frac{f_i(s_j)}{\sqrt{\sum_{j=1}^{n} (f_i(s_j) - av_{f_i})^2}}$$

Die Ranking-Funktionswerte einer Lösung hängen sowohl von der Menge der Zielfunktionen und den Gewichten als auch von der Menge der Lösungen ab.

Die in diesem Kapitel eingeführten verallgemeinerten Job-Shop-Probleme werden im Weiteren als **praktische Job-Shop Scheduling-Probleme** $PJSP$ bezeichnet. Diese beinhalten eine oder mehrere Verallgemeinerungen sowie verschiedene oder mehrere Zielfunktionen.

Für die Notation multikriterieller Scheduling-Probleme wird die in [59] vorgeschlagene Schreibweise für das γ-Feld in der $\alpha|\beta|\gamma$-Notation genutzt. Die Optimierung nach der lexikographischen Ordnung von m Zielfunktionen wird mit $Lex(f_1, \ldots, f_m)$ bezeichnet. Die Suche nach der Menge der nichtdominierten Lösungen bei m Zielfunktionen wird mit $\#(f_1, \ldots, f_m)$ im γ-Feld angegeben.

Kapitel 2

Optimierung bei einer Zielfunktion

Das JSP ist mit der Zielfunktion C_{max} ein NP-schweres Optimierungsproblem. Bei einer Instanz mit 15 Jobs und 15 Maschinen gibt es $(15!)^{15} \approx 5.59 \cdot 10^{181}$ mögliche Reihenfolgen, wobei nicht zulässige hierbei eingeschlossen sind. Zur exakten Lösung des $J||C_{max}$ wurden verschiedene Branch & Bound-Algorithmen entwickelt (siehe [8, 9, 38]). Die Grenzen dieser Algorithmen liegen in etwa bei Problemdimensionen von 15 Jobs und 15 Maschinen (225 Vorgänge) bis maximal 20 Jobs und 15 Maschinen (300 Vorgänge). Das kleinste offene Benchmarkproblem hat 20 Jobs und 10 Maschinen. Bei Benchmark-Instanzen mit 20 Jobs und 15 Maschinen gibt es Abweichungen zwischen unterer und oberer Schranke von bis zu 16 %.

Um gute Lösungen für größere Problemdimensionen des $J||C_{max}$ zu erhalten, wurde eine große Anzahl verschiedener Optimierungsheuristiken implementiert. Einen Überblick enthalten [29], [30] und [60]. Die Algorithmen, die im Moment in der kürzesten Zeit die Lösungen mit den besten oberen Schranken erzeugen, sind die Tabu-Suche von Nowicki und Smutnicki [43, 44], der genetische Algorithmus von Yamada und Nakano [62] und die Hybrid-Algorithmen von Jain [29].

Im Gegensatz zum klassischen JSP gibt es für praktische und multikriterielle Job-Shop-Probleme relativ wenige Ansätze. Diese beschränken sich im Wesentlichen auf die Erzeugung von Schedules mittels Prioritätsregeln [27, 28] und Shifting-Bottleneck-Heuristiken [52]. Erst in letzter Zeit wurden reine genetische Algorithmen zur multikriteriellen Optimierung von Job-Shop-Pro-

blemen verwendet [19].

2.1 Heuristiken

2.1.1 Basisalgorithmen

In Abschnitt 1.1.2 wurde das disjunktive Graphenmodell eingeführt. Die auf dem Graphenmodell basierende **aufsteigende Nummerierung** wird als Lösungsrepräsentation in den folgenden Optimierungsheuristiken benutzt.

Definition 2.1 *(aufsteigende Nummerierung)*
Sei $G = (V, A)$ ein gerichteter Graph und $pre(v, G) = \{u \in V | (u, v) \in A\}$ für $v \in V$ die Menge der direkten Vorgänger von v in G. Weiter sei $suc(v, G) = \{w \in V | (v, w) \in A\}$ die Menge der direkten Nachfolger von v in G. Sei weiterhin $\pi : V \setminus \{0, *\} \rightarrow \{1, \ldots, N\}$ eine bijektive Abbildung, die jedem Knoten eine Zahl von 1 bis N zuordnet.
*π heißt **aufsteigende Nummerierung** der Knoten aus G, falls $\pi(u) < \pi(v)$ für alle Vorgänger $u \in pre(v, G)$ gilt.*

Ein gerichteter Graph ist genau dann kreisfrei, falls eine zugehörige aufsteigende Nummerierung existiert. Algorithmus 2.2 berechnet eine aufsteigende Nummerierung für einen gerichteten Graphen $G = (V, A)$.

Algorithmus 2.2 *Bestimmung einer aufsteigenden Nummerierung*
Es sei $\mathsf{G} = (\mathsf{V}, \mathsf{A})$ mit der Knotenmenge $\mathsf{V} = \{0, 1, 2, \ldots, \mathsf{N}, *\}$. Weiterhin sei $\mathsf{i} = 1$, $\mathsf{W} = \{\mathsf{u} \in \mathsf{V} | \mathsf{pre}(\mathsf{u}, \mathsf{G}) = \{0\}\}$ die Menge der vorläufig markierten Knoten und $\mathsf{U} = \mathsf{V} \setminus \mathsf{W}$ die Menge der noch nicht markierten Knoten.

1. *Wähle ein $\mathsf{v} \in \mathsf{W}$. Setze $\pi(\mathsf{v}) = \mathsf{i}$ und $\mathsf{W} = \mathsf{W} \setminus \{\mathsf{v}\}$.*
2. *$\mathsf{i} = \mathsf{i} + 1$. Sei $\mathsf{S_v} = \{\mathsf{w} \in \mathsf{suc}(\mathsf{v}, \mathsf{G}) | \mathsf{pre}(\mathsf{w}, \mathsf{G}) \cap (\mathsf{W} \cup \mathsf{U}) = \emptyset\}$.*
   Falls $\mathsf{W} \cup \mathsf{S_v} = \emptyset$, Abbruch. Andernfalls setze $\mathsf{W} = \mathsf{W} \cup \mathsf{S_v}$ und $\mathsf{U} = \mathsf{U} \setminus \mathsf{S_v}$.
3. *Falls $\mathsf{W} = \{*\}$, Stopp. Andernfalls gehe zu Schritt 1.*

Da die Menge W im Schritt 1 mehr als ein Element enthalten kann, sind verschiedene aufsteigende Nummerierungen für einen Graphen möglich.

Wie in Abschnitt 1.1.2 beschrieben, wird die Lösung einer JSP-Instanz durch den zu einer vollständigen Selektion gehörenden gerichteten Graphen $G_S = (V, A \cup S)$ definiert. Mit Algorithmus 2.2 kann die Kreisfreiheit verifiziert werden. Mit Hilfe der aufsteigenden Nummerierung werden die frühesten **Startzeitpunkte** s_v und die **Endzeitpunkte** e_v für jedes $v \in V$ berechnet.

Algorithmus 2.3 *Berechnung der frühesten Start- und Endzeitpunkte (klassische Job-Shop-Probleme)*
Es sei π *eine aufsteigende Nummerierung der* N *Knoten aus* $\mathsf{G} = (\mathsf{V}, \mathsf{A})$, $\mathsf{i} = 1$, $\mathsf{s}_0 = \mathsf{e}_0 = 0$.

1. $\mathsf{v} = \pi^{-1}(\mathsf{i})$.
2. $\mathsf{s}_\mathsf{v} = \max\{\mathsf{e}_\mathsf{w} | \mathsf{w} \in \mathsf{pre}(\mathsf{v}, \mathsf{G})\}$.
3. $\mathsf{e}_\mathsf{v} = \mathsf{s}_\mathsf{v} + \mathsf{p}_\mathsf{v}$
4. $\mathsf{i} = \mathsf{i} + 1$. *Falls* $\mathsf{i} \leq \mathsf{N}$, *gehe zu Schritt 1.*

Dieser Algorithmus berechnet die frühesten Start- und Endzeitpunkte für klassische Job-Shop-Probleme in $O(N)$, da jeder Knoten maximal 2 Vorgänger hat. Der zugehörige Schedule ist semiaktiv.

Für praktische Job-Shop-Probleme muss der Algorithmus erweitert werden, da die Verfügbarkeitsintervalle der Maschinen und der Ressourcenverbrauch beachtet werden müssen.

Algorithmus 2.4 *Berechnung der frühesten Start- und Endzeitpunkte (praktische Job-Shop-Probleme)*
Es sei π *eine aufsteigende Nummerierung der* N *Knoten aus* $\mathsf{G} = (\mathsf{V}, \mathsf{A})$, $\mathsf{i} = 1$, $\mathsf{s}_0 = \mathsf{e}_0 = 0$. *Initialisiere* $rf_k(t) = ra_k$ *für* $\mathsf{t} \in \mathbb{R}^+$.

1. $\mathsf{v} = \pi^{-1}(\mathsf{i})$. *Sei* M_v *die benötigte Maschine und* rd^v *der zum Vorgang gehörende Ressourcenvektor.*
2. $\mathsf{s}_\mathsf{v} = \max\{\mathsf{e}_\mathsf{w} | \mathsf{w} \in \mathsf{pre}(\mathsf{v}, \mathsf{G})\}$.
3. $\mathsf{t}_1 = \mathsf{s}_\mathsf{v}$ DIV 24, $\mathsf{t}_2 = \mathsf{s}_\mathsf{v}$ MOD 24.
4. *Falls* $\mathsf{t}_2 \geq \mathsf{st}_{\mathsf{M}_\mathsf{v}}$, *setze* $\mathsf{t}_1 = \mathsf{t}_1 + 1$, $\mathsf{t}_2 = 0$ *und* $\mathsf{s}_\mathsf{v} = \mathsf{t}_1 \cdot 24$.
5. $\mathsf{t}_1 = \mathsf{t}_1 \cdot \mathsf{st}_{\mathsf{M}_\mathsf{v}} + \mathsf{t}_2$, $\mathsf{t}_1 = \mathsf{t}_1 + \mathsf{p}_\mathsf{v}$.

6. $t_2 = t_1$ DIV st_{M_v}, $t_1 = t_1$ MOD st_{M_v}.

7. *Falls* $t_1 = 0$, *setze* $t_1 = st_{M_v}$ *und* $t_2 = t_2 - 1$.

8. $e_v = t_2 \cdot 24 + t_1$.

9. $j = 1$.

10. *Falls* $rf_j(t) < rd_j^v$ *für ein* $t \in [s_v, e_v)$, *suche nächstes Intervall* $[t_1, t_2)$ *von* rf_j *mit* $rf_j(t) \geq rd_j^v$. *Setze* $s_v = t_1$ *und gehe zu Schritt 3.*

11. $j = j + 1$. *Falls* $j \leq r$, *gehe zu Schritt 10.*

12. *Aktualisiere* rf_j *für alle* $j = 1, \ldots, r$.

13. $i = i + 1$. *Falls* $i \leq N$, *gehe zu Schritt 1.*

Auch die mit Algorithmus 2.4 erzeugten Schedules sind semiaktiv. Jeder Vorgang wird zum frühesten Zeitpunkt eingeplant, der ohne Änderung der Reihenfolge auf den Maschinen möglich ist. Mit Hilfe der durch den Algorithmus berechneten Start- und Endzeitpunkte der Vorgänge können die Zielfunktionswerte berechnet werden. Ein Problem hierbei ergibt sich bei nichtregulären Zielfunktionen wie $\sum F$ oder $|L|_{max}$. Die Zielfunktionswerte können durch verzögerte Schedules mit den gleichen Reihenfolgen auf den Maschinen verbessert werden.

Dass in dieser Arbeit trotzdem nur semiaktive Schedules betrachtet werden, hat mehrere Gründe. Die Bestimmung der Start- und Endzeitpunkte in den Lokale-Suche-Heuristiken ist rechenzeitkritisch. Die Einbeziehung einer Rechtsverschiebung der Vorgänge bei vorgegebenen Maschinenreihenfolgen wäre eine Möglichkeit zur Optimierung von nichtregulären Zielfunktionen. Dies würde aber eine nicht unbedeutende Verlangsamung der Optimierungsheuristiken bedeuten. Ein weiterer Grund liegt in den Praxisanforderungen. Bei der Verwendung semiaktiver Schedules startet jeder Vorgang am Beginn des Realisierungsintervalls zum frühesten Startzeitpunkt. Semiaktive Produktionspläne sind demnach in der Praxis robuster gegenüber Maschinenausfällen, da hier die Zeit zwischen frühesten und spätesten Startzeitpunkt als Puffer dient. Der dritte Grund sind die in Abschnitt 3.2 beschriebenen Abstandsmaße im Lösungsraum. Diese Abstände werden über die Permutationen der Tasks auf den Maschinen bestimmt. Wenn verzögerte Schedules betrachtet werden, können zwei Lösungen, die die gleiche Reihenfolge auf den Maschinen und Abstand 0 zueinander haben, verschiedene Schedules und

Zielfunktionswerte haben. Deshalb ist eine eindeutige Zuordnung von aufsteigender Nummerierung zum Schedule, wie sie Algorithmus 2.4 vornimmt, nötig.

Die Laufzeit von Algorithmus 2.4 hängt von der Anzahl der Vorgänger eines Tasks und der Anzahl sowie der Belegung der Ressourcen ab. Im Weiteren wird davon ausgegangen, dass der Verzweigungsgrad der Intree-Struktur in Montageaufträgen durch eine Konstante K_1 beschränkt ist. Ein Task hat demnach höchstens $K_1 + 1$ Vorgänger im gerichteten Graphen. Die Berechnung der tatsächlichen Bearbeitungszeiten bei einer Verlängerung durch Stillstandszeiten benötigt einen konstanten Rechenaufwand. In einem praktischen Job-Shop-Problem mit verallgemeinerten Reihenfolgebeziehungen und Nichtverfügbarkeitsintervallen auf den Maschinen ist die Laufzeit demnach $O(N)$.

Im Fall eines Scheduling-Problems mit Ressourcen muss für jeden Vorgang ein Intervall gefunden werden, in dem genügend Einheiten der benötigten Ressourcen vorhanden sind. Es wird angenommen, dass die Anzahl der Ressourcen, die von einem Vorgang benötigt werden, durch eine Konstante K_2 beschränkt ist. Das bedeutet, höchstens K_2 Einträge im Vektor rd sind größer als 0. Im schlechtesten Fall wird eine Ressource k von allen Tasks benötigt. Damit sind höchstens $2N + 1$ Intervalle in der Liste der Intervalle, die die Funktion rf_k beschreibt. Das Suchen des Intervalls in der Liste, in dem der früheste Startzeitpunkt des Vorgangs liegt, benötigt $O(\log N)$ Schritte. Von da beginnend wird ein Intervall gesucht, in dem genügend Ressourcen vorhanden sind. Dies benötigt $O(N)$ Schritte und muss für höchstens K_2 Ressourcen pro Vorgang durchlaufen werden. Algorithmus 2.4 benötigt in diesem Fall insgesamt $O(N(\log N + N)) = O(N^2)$ Schritte im worst case.

2.1.2 Startlösungen und Nachbarschaften

Startlösungen

Für die im Weiteren untersuchten Verbesserungsverfahren werden Startlösungen als Ausgangsdaten benötigt. Hierzu kann Algorithmus 2.2 verwendet werden. Gegeben ist der disjunktive Graph $G = (V, A, E)$. Gesucht ist eine aufsteigende Nummerierung π. Als Eingabe für den Algorithmus wird der gerichtete Graph $G = (V, A)$ mit der Menge A der aus den Reihenfolgebeziehungen resultierenden konjunktiven Kanten verwendet.

Die vom Algorithmus berechnete aufsteigende Nummerierung π induziert eine Reihenfolge der Vorgänge auf den Maschinen. Falls in der aufsteigenden Nummerierung $\pi(v) < \pi(w)$ gilt und die Vorgänge v und w auf der gleichen Maschine M_k mit $k = 1, \ldots, m$ bearbeitet werden müssen, so ist die gerichtete Kante (v, w) in der zur Maschine M_k gehörenden Selektion S_k enthalten.

Somit erhält man eine vollständige Selektion S und den dazu gehörenden Graphen $G_S = (V, A \cup S)$. Da für diesen Graphen eine aufsteigende Nummerierung π existiert, ist die Lösung zulässig. Mit Algorithmus 2.4 können dann Start- und Endtermine der Vorgänge im zugehörigen semiaktiven Schedule berechnet werden.

Dieser Algorithmus zur Erzeugung von Startlösungen entspricht Prioritätsregelverfahren (siehe z.B. [16]). Die Prioritätsregeln beziehen sich hierbei auf die in Schritt 1 des Algorithmus auszuführende Auswahl aus der Menge W der vorläufig markierten Knoten. Da für die iterierte lokale Suche oder zum Füllen des Ausgangspools bei genetischen Algorithmen viele verschiedene Startlösungen nötig sind, werden die Knoten aus W zufällig gemäß Gleichverteilung ausgewählt.

Nachbarschaften

Die aufsteigende Nummerierung π wurde als Lösungsrepräsentation innerhalb der Heuristiken verwendet. Damit ist eine zulässige vollständige Selektion determiniert. Die Zuordnung zu einem semiaktiven Schedule ist eindeutig und wird durch Algorithmus 2.4 bestimmt. Die Menge aller möglichen aufsteigenden Nummerierungen π und damit die Menge der zulässigen Lösungen bildet den Lösungsraum $\mathscr{X}$ für eine Instanz des $PJSP$.

Für die im Weiteren vorgestellten Lokale-Suche-Heuristiken wird das Konzept der **Nachbarschaft** innerhalb des Lösungsraumes $\mathscr{X}$ benötigt. Um die Nachbarn einer Lösung $x \in \mathscr{X}$ zu beschreiben, wird eine **Nachbarschaftsfunktion** $N : \mathscr{X} \to \mathfrak{P}(\mathscr{X})$ definiert. Die Menge der Nachbarn von x ist durch $N(x) \subseteq \mathscr{X}$ gegeben. Jede Lösung in $N(x)$ wird als **Nachbar** von x bezeichnet. Der durch die Nachbarschaftsfunktion definierte Graph wird als **Nachbarschaftsgraph** bezeichnet. Eine Nachbarschaft ist zusammenhängend, falls der dazugehörige Nachbarschaftsgraph zusammenhängend ist. Die Nachbarschaft ist symmetrisch, falls aus $x' \in N(x)$ $x \in N(x')$ folgt.

Nachbarschaftsfunktionen sind normalerweise durch einfache Änderungen an einer zulässigen Lösung definiert. In der Literatur werden verschiedene Nach-

barschaftsfunktionen für das JSP vorgeschlagen. Die meisten dieser Nachbarschaftsfunktionen sind auf dem zu einem Schedule korrespondierenden gerichteten Graphen G_S definiert. Der Schritt zu einer benachbarten Lösung hängt demnach von der Orientierung der disjunktiven Kanten in der vollständigen Selektion S ab.

Die einfachste Nachbarschaftsfunktion hierbei ist die Umkehrung der Orientierung genau einer Kante in einem Block auf einen kritischen Pfad in G_S. Ein **Block** ist eine maximale Folge von adjazenten Vorgängen, die auf der gleichen Maschine bearbeitet werden müssen. Ein Block muss mindestens aus zwei Vorgängen bestehen, um eine Vertauschung der Reihenfolge der Vorgänge zu ermöglichen. Das Beispiel aus Abbildung 1.2 hat den kritischen Pfad $(T_7, T_8, T_{25}, T_{26}, T_{27}, T_{23}, T_{18}, T_{35}, T_6)$. Hier sind zwei Blöcke enthalten, nämlich (T_8, T_{25}) und $(T_{27}, T_{23}, T_{18}, T_{35}, T_6)$. Diese Nachbarschaft wurde durch Van Laarhoven [34] eingeführt. Sie ist zusammenhängend, und die Nachbarlösungen sind zulässig.

Da die Berechnung der aufsteigenden Nummerierung und des kritischen Pfades der Nachbarlösungen jeweils $O(N)$ Zeit benötigt, wurde versucht, die Nachbarschaft weiter einzuschränken. Nowicki und Smutnicki [43] benutzen in dem von ihnen entwickelten Tabu-Suche-Algorithmus nur die Vertauschung der ersten und letzten beiden Vorgänge in einem Block als Nachbarschaft. Im ersten Block auf dem kritischen Pfad werden nur die letzten beiden Vorgänge und im letzten Block auf dem kritischen Pfad werden nur die ersten beiden Vorgänge vertauscht. Diese Nachbarschaft ist nicht mehr zusammenhängend. Deshalb muss mehr Aufwand für die Erzeugung einer Startlösung betrieben werden [44].

Die Nachbarschaften für das klassische Job-Shop Scheduling-Problem, die auf dem kritischen Pfad definiert sind, haben für praktische Job-Shop Scheduling-Probleme mehrere Nachteile. Die Nachbarschaften sind auf den Makespan C_{max} als Zielfunktion zugeschnitten. Für die totale Verspätung $\sum T$ muss für jeden verspäteten Auftrag der kritische Pfad berechnet werden [3]. Für nichtreguläre Zielfunktionen sind diese Nachbarschaften nicht Erfolg versprechend.

Ein weiteres Problem sind die verallgemeinerten Reihenfolgebeziehungen in $PJSP$. So müssen Blöcke auf virtuellen Maschinen berücksichtigt werden. Hierbei können Vertauschungen auch innerhalb von Jobs auftreten. Des Weiteren müssen nicht alle Vertauschungen innerhalb von Blöcken zulässig sein, da Vorgänge in einem Block zu verschiedenen aufeinanderfolgenden Jobs innerhalb eines Montageauftrages gehören können. In den hier implementierten

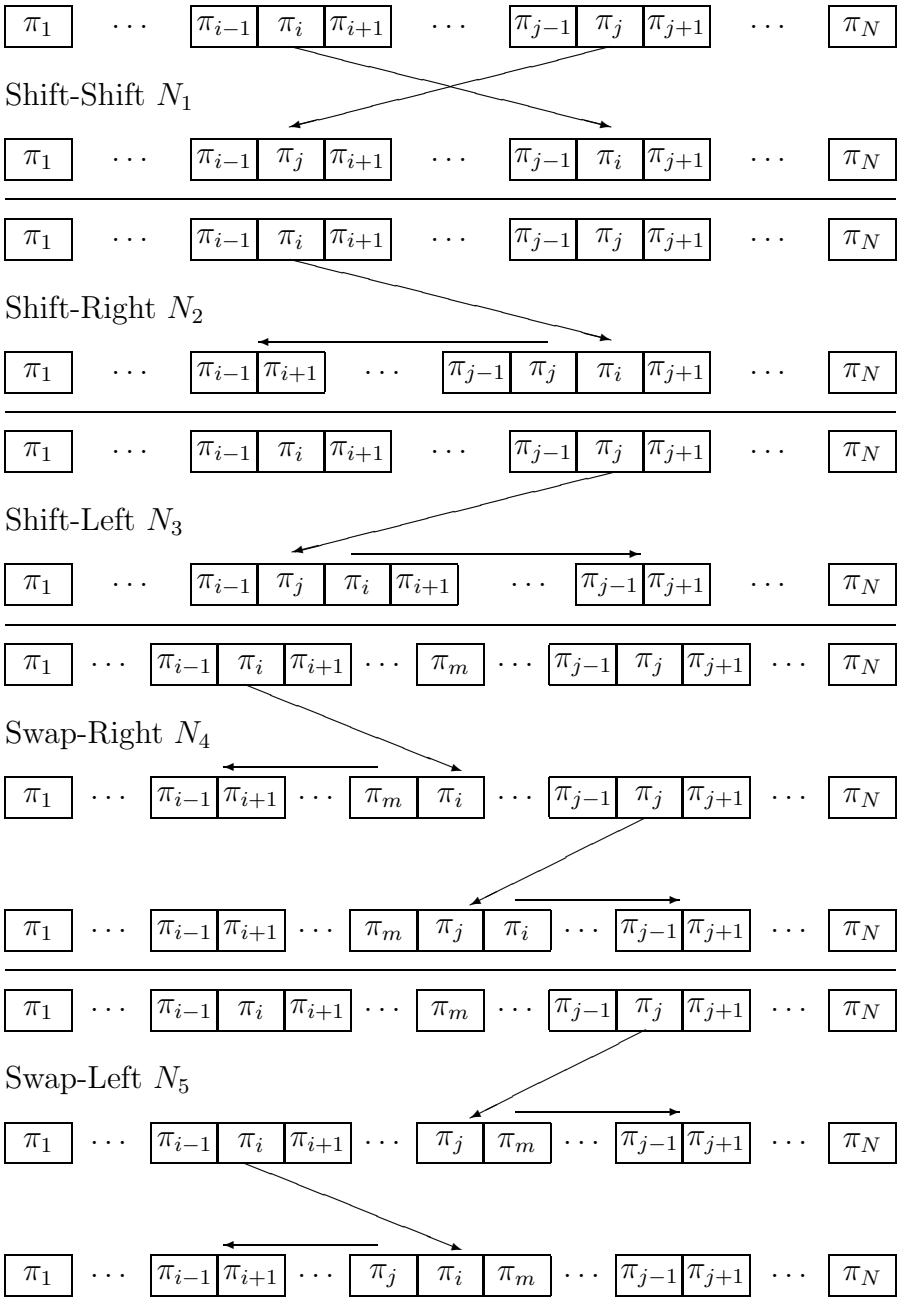

Abbildung 2.1: *Nachbarschaften*

Heuristiken wurden die folgenden Nachbarschaften verwendet:

Definition 2.5 *(Nachbarschaften)*
Sei x eine zulässige Lösung, und sei π die zugehörige aufsteigende Nummerierung der N Vorgänge.

1. ***Shift-Shift Nachbarschaft***
   *Die Nachbarschaft $N_1(x)$ wird durch alle zulässigen Vertauschungen zweier Vorgänge π_i und π_j in π mit $\pi_i < \pi_j$ definiert. Ein Nachbar ist nur zulässig, falls π_i mit allen Vorgängen π_k, $k = i+1, \ldots, j$, und π_j mit allen Vorgängen π_k, $k = i, \ldots, j-1$, vertauscht werden können. Zwei Vorgänge sind **vertauschbar**, falls es keinen gerichteten Pfad zwischen ihnen in $G = (V, A, E)$ gibt.*

2. ***Shift-Right Nachbarschaft***
   Bei der Nachbarschaft $N_2(x)$ wird ein Vorgang π_i nach einem Vorgang π_j mit $\pi_i < \pi_j$ eingefügt. Der Vorgang π_i wird nach rechts geshiftet und alle Vorgänge π_k, $k = i+1, \ldots, j$, um eine Stelle nach links. Um die Zulässigkeit zu garantieren, muss π_i mit allen Vorgängen π_k, $k = i+1, \ldots, j$, vertauschbar sein.

3. ***Shift-Left Nachbarschaft***
   Die Nachbarschaft $N_3(x)$ wird durch Einfügen eines Vorgangs π_j vor einem Vorgang π_i mit $\pi_i < \pi_j$ definiert. π_j muss mit allen Vorgängen π_k, $k = i, \ldots, j-1$, vertauschbar sein.

4. ***Swap-Right Nachbarschaft***
   Die Nachbarschaft $N_4(x)$ ist durch die Vertauschung der Reihenfolge zweier Vorgänge π_i und π_j in π mit $\pi_i < \pi_j$ definiert. Hierbei wird zuerst Vorgang π_i so weit wie möglich nach rechts bis maximal nach π_j bewegt. Falls ein Vorgang π_{m+1} mit $\pi_{m+1} < \pi_j$ existiert, mit dem π_i nicht vertauschbar ist, wird π_i nach π_m in π eingefügt. Falls π_j mit allen Vorgängen π_k, $k = i, m+1, \ldots, j-1$, vertauschbar ist, wird π_j vor π_i eingefügt.

5. ***Swap-Left Nachbarschaft***
   Die Nachbarschaft $N_5(x)$ ist das Analogon zu $N_4(x)$. Hierbei wird Task π_j so weit wie möglich nach links bewegt. Ist π_{m-1} der Vorgang mit dem größten Index, mit dem π_j nicht vertauschbar ist, so wird π_j vor π_m in π eingefügt. Falls π_i mit allen Vorgängen π_k, $k = i+1, \ldots, m-1, j$, vertauschbar ist, wird π_i vor π_j eingefügt.

Seien $i, j \in \{1, \ldots, N\}$ mit $i < j$. Sei $k \in \{1, 2, 3, 4, 5\}$. Dann bezeichnet $N_k^{ij}(x)$ denjenigen Nachbarn von x, der durch Vertauschung der Vorgänge π_i und π_j bei Verwendung der Nachbarschaft N_k entsteht.

Abbildung 2.1 zeigt die fünf benutzten Nachbarschaftsfunktionen. Für jede Lösung x existieren $\binom{N}{2}$ potentielle Nachbarn. Der Test auf Zulässigkeit des Nachbarn benötigt für alle Nachbarschaften im schlechtesten Fall $O(N)$ Zeit. Das entspricht dem Aufwand der Berechnung einer aufsteigenden Nummerierung aus dem gerichteten Graphen und dem Aufwand zur Erzeugung des kritischen Pfades.

Mit Hilfe der Nachbarschaftsfunktionen kann der im Weiteren benötigte Begriff einer **lokal optimalen Lösung** definiert werden.

Definition 2.6 *(lokal optimale Lösung)*
*Sei I eine Instanz eines praktischen Job-Shop-Problems $PJSP$ mit dem Lösungsraum $\mathscr{X}$ und einer Nachbarschaftsfunktion $N : \mathscr{X} \to \mathfrak{P}(\mathscr{X})$. Dann heißt eine Lösung $x \in \mathscr{X}$ zu einer gegebenen Zielfunktion $f : \mathscr{X} \to \mathbb{R}$ **lokal minimal**, wenn für alle $x' \in N(x)$ gilt:*

$$f(x) \leq f(x').$$

Analog wird für Maximierungsaufgaben eine **lokal maximale Lösung** definiert. Da hier nur Minimierungsaufgaben behandelt werden, wird im Folgenden häufig der Begriff „lokal optimale Lösung“ statt „lokal minimale Lösung“ verwendet.

2.1.3 Genetische lokale Suche

Die in dieser Arbeit entwickelten Optimierungsheuristiken sind Algorithmen, die Eigenschaften von genetischen Algorithmen und Lokale-Suche-Metaheuristiken kombinieren. In der Literatur werden hierfür die Bezeichnungen **genetische lokale Suche (GLS)**, Hybrid-Algorithmus oder Shell-Algorithmus[1] verwendet. Der Pool von lokalen Optima des genetischen Algorithmus ist das Langzeitgedächtnis des Hybrid-Algorithmus und verantwortlich für die Diversifikation der Suche. Die Lokale-Suche-Metaheuristiken verwalten das

[1] Shell-Algorithmus steht hierbei für den Aufbau der Heuristik in Schalen. Im Kern steht die lokale Suche, und als äußere Schale dient der genetische Algorithmus mit Rekombination und Selektion.

Kurzzeitgedächtnis des Hybrid-Algorithmus und sind verantwortlich für die Intensifikation der Suche. Die in diesem Abschnitt entwickelten Heuristiken können verschiedene Zielfunktionen optimieren, aber davon jeweils nur eine. Sie dienen als Basis für die im Weiteren vorgestellten multikriteriellen Heuristiken.

Allgemeine Struktur

Die in den letzten Jahren entwickelte Idee, mittels Path Relinking, Struktursuche und Crossover-Operatoren aus schon gegebenen guten Lösungen eines kombinatorischen Optimierungsproblems neue Startlösungen zu erzeugen, wurde auf verschiedene Optimierungsprobleme mit Erfolg angewendet.

Rose [49] wendete Mehrheits- und Mittelwertbildung mit iterierter lokaler Suche auf mehrere kombinatorische Optimierungsprobleme wie zum Beispiel MAX-3-SAT, Rucksack Probleme, Ising-Spinglas-Probleme und ein Scheduling-Problem an. Der Grundgedanke hierbei ist die Übernahme gewisser Teilstrukturen aus den vorhandenen lokalen Optima in die Nachkommen.

Der folgende Algorithmus gibt das Grundmodell der Hybrid-Algorithmen an:

Algorithmus 2.7 *Genetische lokale Suche*
Sei n die maximale Anzahl der Lösungen im Pool P.

1. *Für $i = 1, \ldots, n$*
   (a) *Generiere Startlösung y mittels Algorithmus 2.2.*
   (b) *Finde mit lokaler Suche beginnend mit y lokales Optimum x und füge x in P ein.*
2. *Wähle k Lösungen $\{x_1, \ldots, x_k\}$ aus P mit $k \leq n$.*
3. *Bilde Nachkommen x aus den k Lösungen.*
4. *Mutiere $x \rightarrow y$.*
5. *Finde mit lokaler Suche beginnend mit y lokales Optimum x und füge x in P ein.*
6. *Entferne die gemäß Zielfunktion schlechteste Lösung aus P.*
7. *Falls Abbruchkriterium erfüllt, STOPP. Sonst gehe zu 2.*

Diese allgemeine Strategie kann auf verschiedenste kombinatorische Optimierungsprobleme angewendet werden. Für die zu optimierenden praktischen Job-Shop Scheduling-Probleme werden im Folgenden die Rekombination der Lösungen, die Mutation, die Lokale-Suche-Algorithmen und weitere Parameter der Hybrid-Algorithmen konkretisiert.

Rekombination

Die Idee der Rekombination ist, aus einer Anzahl von Elternlösungen neue Lösungen, die Nachkommen, zu erzeugen. Dabei stammen die Elternlösungen aus einem Pool von Elitelösungen. Elitelösungen sind lokal optimale Lösungen (im Fall mit einer Zielfunktion) oder Pareto-Lösungen (bei mehreren Zielfunktionen). Durch die Rekombination sollten die Strukturinformationen der Elternlösungen in die Nachkommen übernommen werden, um Startlösungen für einen neuen Lauf mit lokaler Suche zu erzeugen.

Der hier verwendete Rekombinationsoperator ist die in [49] eingeführte **Mittelwertbildung** der Anfangszeiten.

Definition 2.8 *(Mittelwertbildung)*
Sei $\Pi = \{\pi^1, \pi^2, \ldots, \pi^k\},\ k \geq 2$ *die Menge der zulässigen Elternlösungen in ihrer Darstellung als aufsteigende Nummerierung und* N *die Anzahl der Vorgänge der Lösungen.* $s_i^j,\ i = 1, \ldots, N,\ j = 1, \ldots, k$ *sei die Startzeit des Vorgangs* i *in der Lösung* j*. Setze*

$$\bar{s}_i = \sum_{j=1}^{k} s_i^j$$

für jedes $i = 1, \ldots, N$ *und sortiere die Vorgänge aufsteigend nach* $\bar{s}_i$*. Die resultierende Permutation* $\bar{\pi}$ *der Vorgänge ist die durch* ***Mittelwertbildung*** *erzeugte aufsteigende Nummerierung der Rekombinationslösung.*

Die durch die Mittelwertbildung gewonnene Lösung $\bar{\pi}$ ist zulässig und damit der korrespondierende gerichtete Graph kreisfrei. Aus $\bar{\pi}$ lassen sich unmittelbar die frühesten Startzeiten der Rekombinationslösung berechnen und damit auch die Werte der Zielfunktionen.

Mutation

Die Mutation von Lösungen wird eingesetzt, um die Diversität im Lösungspool zu erhöhen, aus den Einzugsgebieten starker lokaler Optima zu entkommen und das Phänomen der Inzucht in der Generationenfolge zu verhindern. Als Mutationsoperator werden zufällige zulässige Vertauschungen in der aufsteigenden Nummerierung der zu mutierenden Lösung verwendet. Bei den Vertauschungen werden die in Definition 2.5 eingeführten Nachbarschaften benutzt. Durch diese Vertauschungen können sich die Werte der Zielfunktionen verschlechtern und die mutierten Lösungen aus dem Bereich der guten lokalen Optima herauswandern. Aus diesem Grund wurden als Parameter für die Mutation eine maximale Anzahl von Nachbarschaftsdurchläufen und eine obere Schranke für die Zielfunktionswerte eingeführt.

Algorithmus 2.9 *Mutation von Lösungen*
Sei $x \in \mathscr{X}$ die zu mutierende Lösung mit der aufsteigenden Nummerierung π und N Vorgängen. Seien weiter $n \in \mathbb{N}$ die Anzahl der Nachbarschaftsdurchläufe, $p \in \mathbb{R},\ p \geq 1$ der Parameter für die obere Schranke, $f : \mathscr{X} \to \mathbb{R}^+$ die Zielfunktion und $N_\alpha, \alpha \in \{1,2,3,4,5\}$ die verwendete Nachbarschaft.

1. $B = p \cdot f(x)$
2. *Für* $k = 1, \ldots, n$
   (a) *Für* $i = N-1, \ldots, 1$
      i. *Wähle zufällig eine Permutation σ aus S_{N-i}.*
      ii. *Für* $j = 1, \ldots, N-i$
         A. *Falls* $x' = N_\alpha^{\sigma(j),\sigma(j)+i}(x)$ *zulässige Lösung und* $f(x') \leq B$, *dann setze* $x = x'$

Es wird in dem Algorithmus versucht, die Tasks zuerst über eine große Distanz und dann über die kleineren Distanzen zu tauschen. In der inneren Schleife werden die Vorgänge in der Reihenfolge gemäß der zufällig gewählten Permutation durchlaufen. Der Parameter für die Nachbarschaftsdurchläufe wurde aus $n \in \{0, \ldots, 10\}$ und der Parameter für die obere Schranke aus $p \in (1.0, 1.1]$ gewählt. Falls mehrere Zielfunktionen relevant sind, wird für jede Zielfunktion $f_i : \mathscr{X} \to \mathbb{R}^+$ die Schranke $B_i = p \cdot f_i(x)$ berechnet. Ein zulässiger Nachbar x' wird nur akzeptiert, falls $f_i(x') \leq B_i$ für alle i gilt. Der Parameter 0 für die Anzahl der Nachbarschaftsdurchläufe bedeutet, dass die durch Mittelwertbildung entstandene Lösung auch ohne Mutation als Startlösung für eine Lokale-Suche-Heuristik verwendet werden kann.

Selektion

Die Mittelwertbildung als Rekombinationsoperator kann aus beliebig vielen Elternlösungen eine Nachkommenlösung erzeugen. Die Elternlösungen werden zufällig gemäß Gleichverteilung aus dem Pool der momentan vorhandenen Elitelösungen ausgewählt. Die Nachkommenlösung wird gegebenenfalls nach einer Mutation als Startlösung für eine Lokale-Suche-Heuristik verwendet. Die dadurch erzeugte lokal optimale Lösung wird nur dann in den Pool der Elitelösungen eingefügt, falls in dem Pool keine Lösung mit dem gleichen kritischen Pfad existiert.

Falls eine Lösung in den Pool eingefügt wurde, werden die Lösungen aufsteigend nach dem Zielfunktionswert geordnet. Eine Lösung mit dem schlechtesten Zielfunktionswert wird aus dem Pool entfernt.

2.1.4 Lokale Suche

Mit der Definition von Nachbarschaften auf dem Raum der zulässigen Schedules und dem Begriff des lokalen Minimums können Algorithmen zur Ermittlung eines lokalen Minimums angegeben werden. In diesem Abschnitt werden zwei einfache Varianten der lokalen Suche angegeben. Diese sind in Hinsicht auf die Qualität der lokalen Optima gegenüber den komplexeren Varianten nicht konkurrenzfähig. Da sie im Gegensatz zur Tabu-Suche oder zum Sidestep-Algorithmus keine Eingabeparameter wie Länge der Tabuliste oder Anzahl der Iterationen haben, kann die Effizienz von Nachbarschaften bei verschiedenen Zielfunktionen oder die Parameter des übergeordneten Hybrid-Algorithmus ohne Nebeneffekte getestet werden.

Volle lokale Suche

Bei der vollen lokalen Suche wird die gesamte Nachbarschaft der aktuellen Lösung durchsucht und die Zielfunktionswerte bestimmt. Die Lösung mit dem kleinsten Zielfunktionswert wird mit der aktuellen Lösung verglichen.

Algorithmus 2.10 *Volle lokale Suche*
Sei N_α die verwendete Nachbarschaft und $f : \mathscr{X} \to \mathbb{R}^+$ die zu minimierende Zielfunktion.

1: Ermittle eine Startlösung $x \in \mathscr{X}$ und setze $f_{min} = f(x)$.

2: Für $i := N-1, \ldots, 1$

3: Für $j := 1, \ldots, N-i$

4: Falls $x' := N_\alpha^{j,j+i}(x)$ zulässige Lösung und $f_{min} > f(x')$, dann setze $f_{min} = f(x')$ und $i_0 = i,\ j_0 = j$.

5: Falls $f_{min} < f(x)$, setze $x = N_\alpha^{j_0,j_0+i_0}(x)$ und gehe zu Schritt 2. Sonst: stopp, x ist lokal minimal.

Bei der vollen lokalen Suche wird nach dem größtmöglichen Verbesserungsschritt in der Nachbarschaft gesucht. Der beste Funktionswert kann in der Nachbarschaft mehrfach auftreten. Dann wird zu der in der Reihenfolge des Nachbarschaftsdurchlaufs ersten Lösung übergegangen.

Die Rechenzeit hängt hierbei von dem Test auf Zulässigkeit eines Nachbarn und von der Berechnung der Zielfunktion ab.

Schnelle lokale Suche

Die hier eingesetzte Variante der lokalen Suche wird als schnelle lokale Suche bezeichnet. Beim Durchlaufen der Nachbarschaft wird zuerst versucht, Tasks über große Distanzen und danach über geringere Distanzen in der aufsteigenden Nummerierung zu tauschen. In einer aufsteigenden Nummerierung der Länge N gibt es bei einer Distanz d, $d \in \{1, \ldots, N-1\}$, genau $(N-d)$ Paare von vertauschbaren Indizes. Diese Paare werden in einer zufälligen Permutation durchlaufen. Im Gegensatz zur vollen lokalen Suche geht die schnelle lokale Suche sofort zu einem besseren Nachbar über, ohne die gesamte Nachbarschaft zu durchlaufen.

Algorithmus 2.11 *Schnelle lokale Suche*
Sei N_α die verwendete Nachbarschaft, und sei $f : \mathscr{X} \to \mathbb{R}^+$ die zu minimierende Zielfunktion.

1: Ermittle eine Startlösung $x \in \mathscr{X}$.

2: Setze $i := N-1$ und $i^\star := i$.

3: Wähle zufällig eine Permutation σ aus S_{N-i}.

4: *Für* $j := 1, \ldots, N - i$

5: *Falls* $x' := N_\alpha^{\sigma(j),\sigma(j)+i}(x)$ *zulässige Lösung und* $f(x') < f(x)$, *dann setze* $x := x'$ *und* $i^\star := i$.

6: *Setze* $i := i - 1$. *Falls* $i = 0$, *setze* $i := N - 1$.

7: *Falls* $i = i^\star$, *stopp.* x *ist lokal optimal.*

8: *Gehe zu Schritt 3.*

Die Permutation der Nachbarschaftspaare wird bei jedem Durchlauf neu bestimmt. Aus diesem Grund müssen alle Paare einer Distanz noch einmal durchlaufen werden, um die lokale Optimalität zu garantieren.

Sowohl die schnelle als auch die volle lokale Suche hat den Nachteil, ein einmal erreichtes lokales Optimum nicht mehr verlassen zu können. Die hierbei erreichten Zielfunktionswerte können noch weit vom globalen Optimum entfernt sein. Um dieses Problem zu umgehen, wurden Metaheuristiken wie Schwellwertalgorithmen oder Tabu-Suche entwickelt.

2.1.5 Sidestep-Algorithmus

Der Sidestep-Algorithmus ist ein deterministischer Schwellwertalgorithmus. Hierbei ist es der Nachbarschaftssuche erlaubt, in einen Nachbarn mit gleichem Funktionswert zu wechseln. Dieser Nachbarschaftsschritt ohne Verbesserung des Zielfunktionswertes wird als **Sidestep** bezeichnet. Da der Algorithmus beim Erreichen eines lokalen Optimums nicht mehr stoppt, wird ein Abbruchkriterium benötigt. Aus diesem Grund wird nur eine bestimmte Anzahl von Sidesteps zugelassen. Diese maximale Anzahl von Sidesteps wird dem Algorithmus als Parameter übergeben.

Algorithmus 2.12 *Sidestep-Algorithmus*
Seien N_α *die verwendete Nachbarschaft,* $f : \mathscr{X} \to \mathbb{R}^+$ *die zu minimierende Zielfunktion und* IS *die maximale Anzahl von Sidesteps.*

1: *Ermittle eine Startlösung* $x \in \mathscr{X}$. $f^\star := f(x)$

2: $i := N - 1$ *und* $k := 0$.

3: *Wähle zufällig eine Permutation* σ *aus* S_{N-i}.

4: *Für* $j := 1, \ldots, N - i$

5: *Sei* $x' := N_\alpha^{\sigma(j),\sigma(j)+i}(x)$ *zulässige Lösung.*
Falls $f(x') < f^\star$, *dann setze* $x := x'$, $k := 0$ *und* $f^\star := f(x')$.
Falls $f(x') = f^\star$, *dann setze* $x := x'$ *und* $k := k + 1$.

6: *Setze* $i := i - 1$. *Falls* $i = 0$, *setze* $i := N - 1$.

7: *Falls* $k = IS$, *stopp. Gib* x *aus.*

8: *Gehe zu Schritt 3.*

Wie aus Algorithmus 2.12 ersichtlich ist, sind die Sidesteps sofort und nicht erst nach dem Erreichen eines lokalen Optimums zugelassen. Des Weiteren kann der Algorithmus durch das Abbruchkriterium in Schritt 7 terminieren, ohne dass alle Nachbarn der aktuellen Lösung untersucht worden sind. Die ausgegebene Lösung x muss demnach kein lokales Minimum sein.

2.1.6 Threshold-Accepting

Der hier implementierte Threshold-Accepting-Algorithmus ist eine Erweiterung des Sidestep-Algorithmus. Während der Nachbarschaftssuche wird jeder Nachbar als neue aktuelle Lösung akzeptiert, deren Funktionswert einen vorgegebenen Schwellwert S unterbietet. Da sich dadurch die aktuelle Lösung verschlechtern kann, wird der bisherige Rekordhalter gespeichert und bei einem Verbesserungsschritt aktualisiert. Der Algorithmus hat zwei Parameter: Zum einen den Faktor T, $T \in \mathbb{R}, T > 1$ für den Startschwellwert und zum anderen die Abnahmerate für den Schwellwert λ, $\lambda \in \mathbb{R}, \lambda \in (0, 1)$. Der Startschwellwert ist das Produkt von T und dem Zielfunktionswert der Startlösung. Da die Zielfunktionen ganzzahlig sind, wird für den Schwellwert der ganzzahlige Anteil des Produkts verwendet.

Algorithmus 2.13 *Threshold-Accepting*
Seien N_α *die verwendete Nachbarschaft,* $f : \mathscr{X} \to \mathbb{R}^+$ *die zu minimierende Zielfunktion,* T *der Faktor für den Schwellwert und* λ *die Abnahmerate.*

1: *Ermittle eine Startlösung* $x \in \mathscr{X}$.

2: $S := \lfloor T \cdot f(x) \rfloor$ *und* $x^* := x$.

3: $i := N - 1$ *und* $i^\star := i$.

4: *Wähle zufällig eine Permutation σ aus S_{N-i}.*

5: *Für $j := 1, \ldots, N - i$*

6: *Sei $x' := N_\alpha^{\sigma(j),\sigma(j)+i}(x)$ zulässige Lösung.*
Falls $f(x') < S$, dann setze $x := x'$.
Falls $f(x') < f(x^*)$, dann setze $x^* := x'$ und $i^\star := i$.

7: *Setze $i := i - 1$. Falls $i = 0$ setze $i := N - 1$.*

8: *Falls $i = i^\star$, stopp. Gib x^* aus.*

9: *Falls $\lfloor T \cdot f(x) \rfloor < S$, setze $S := \lfloor T \cdot f(x) \rfloor$.*
Sonst $\delta := \lfloor \lambda(S - f(x)) \rfloor$ und $S := S - \max\{1, \delta\}$.

10: *Gehe zu Schritt 4.*

Wie in Algorithmus 2.13 zu sehen, wird die Nachbarschaft in zwei Schleifen durchlaufen. Die äußere Schleife kontrolliert die Distanz, über die Vorgänge in der aufsteigenden Nummerierung vertauscht werden. Die innere Schleife durchläuft eine Permutation der Vorgänge, die über die jeweilige Distanz vertauscht werden können. Nach jedem Durchlauf der inneren Schleife wird der Schwellwert neu berechnet (Schritt 9). Auch hier muss die zurückgegebene Lösung x^* wie im Sidestep-Algorithmus kein lokales Optimum sein.

2.1.7 Simulated Annealing

Das Simulated Annealing ist eine probabilistische Variante eines Threshold-Accepting Algorithmus. Hierbei werden Nachbarn mit schlechterem Funktionswert mit einer gewissen Wahrscheinlichkeit akzeptiert. Die Heuristik hat die folgenden vier Parameter:

1. T_0: die Starttemperatur;
2. α: der Abkühlungsfaktor;
3. IA: die Anzahl der Iterationen zwischen zwei Abkühlungsschritten;
4. T_{stop}: die Endtemperatur.

Algorithmus 2.14 *Simulated Annealing*
Seien N_β die verwendete Nachbarschaft, $f : \mathscr{X} \to \mathbb{R}^+$ die zu minimierende Zielfunktion, T_0 die Starttemperatur, IA die Anzahl von Iterationen zwischen zwei Abkühlungsschritten, α der Abkühlungsfaktor und T_{stop} die Endtemperatur.

1. *Ermittle eine Startlösung $x \in \mathscr{X}$.*
2. *$x^* := x$ und $T := T_0$.*
3. *$i := N - 1$ und $k := 0$.*
4. *Wähle zufällig eine Permutation σ aus S_{N-i}*
5. *Für $j := 1, \ldots, N - i$*
   (a) *Sei $x' := N_\beta^{\sigma(j),\sigma(j)+i}(x)$ eine zulässige Lösung.*
   (b) *Falls $f(x') < f(x)$, dann setze $x := x'$.*
   Sonst akzeptiere Nachbarn x' mit Wahrscheinlichkeit $p = e^{\frac{f(x)-f(x')}{T}}$.
   (c) *Falls $f(x') < f(x^*)$, dann setze $x^* := x'$.*
   (d) *$k := k + 1$. Falls $k(mod\ IA) = 0$, setze $T := \alpha T$.*
6. *Falls $T < T_{stop}$, stopp. Gib x^* aus.*
7. *Setze $i := i - 1$. Falls $i = 0$, setze $i := N - 1$.*
8. *Gehe zu Schritt 4.*

Für diesen Algorithmus gilt die gleiche Aussage wie für die deterministischen Threshold-Algorithmen 2.12 und 2.13. Die gefundenen Lösung x^* muss kein lokales Optimum sein.

2.1.8 Tabu-Suche

Die Tabu-Suche ist eine auf Nachbarschaftssuche basierende Metaheuristik, die Ende der 80er Jahre von Glover et al. vorgeschlagen und entwickelt wurde. Der von Nowicki und Smutnicki [43, 44] entwickelte Tabu-Suche-Algorithmus ist der im Moment beste Algorithmus für das Problem $J||C_{max}$. Taillard [58] implementierte eine parallele Variante einer Tabu-Suche für $J||C_{max}$. Der von Armentano und Scrich [3] vorgestellte Tabu-Suche-Algorithmus ist eine Optimierungsheuristik für $J||\sum T$.

Die hier entwickelte Tabu-Suche optimiert praktische Job-Shop-Probleme mit den Zielfunktionen C_{max}, $\sum C$, $\sum T$ und $|L|_{max}$. Der Algorithmus hat zwei Tabulisten. In der ersten Liste TL_N werden die verbotenen Nachbarschaftsschritte gespeichert. Wenn ein Nachbarschaftsschritt mit der Vertauschung der Vorgänge T_i und T_j durchgeführt wird, werden die Nummern der Vorgänge in umgekehrter Reihenfolge in die Liste TL_N eingefügt. Somit ist der Rücktausch dieser Vorgänge tabu. Die Liste ist am Start des Algorithmus leer. Die Liste wird bis zu einer maximalen Länge L_N aufgefüllt. Mit jedem weiteren Eintrag wird der älteste verbotene Nachbarschaftsschritt zyklisch überschrieben. Die zweite Tabuliste TL_L enthält alle kritischen Pfade verbotener Lösungen. Auch diese Liste ist zu Beginn leer und wird bis zu einer maximalen Länge L_L aufgefüllt. Danach wird auch hier zyklisch überschrieben. Ein Nachbar gilt ebenfalls als tabu, wenn er einen kritischen Pfad mit einer der Lösungen in der Lösungstabuliste gemeinsam hat. Ein weiterer Parameter IT gibt die maximale Anzahl von Nachbarschaftsschritten an, die ohne Verbesserung der bisher besten gefundenen Lösung durchgeführt werden können.

Die Heuristik 2.15 durchsucht die gesamte Nachbarschaft einer Lösung. Dabei können die Nachbarn von x drei Mengen zugeordnet werden:

1. der Menge $NT(x) = (N_\alpha(x) \setminus TL_N) \setminus TL_L$ der Lösungen, die nicht tabu sind;
2. der Menge
$$TV(x) = \{x' \in (N_\alpha(x) \cap TL_N) \cup (N_\alpha(x) \cap TL_L) | f(x') < f(x^*)\}$$
der Lösungen, die tabu sind, aber die beste bisher gefundenen Lösung verbessern;
3. der Menge $TN(x) = N_\alpha(x) \setminus (NT(x) \cup TV(x))$ der Lösungen, die tabu sind und keine Verbesserung der besten bisher gefundenen Lösung erreichen.

Falls in der Nachbarschaft Lösungen existieren, die nicht tabu sind oder die tabu, aber besser als die bisher beste gefundene Lösung sind, wird die beste davon zur neuen aktuellen Lösung. Ansonsten werden die ältesten Nachbarschaftsschritte in der Nachbarschaftstabuliste TL_N gelöscht, bis eine Lösung in der Nachbarschaft der aktuellen Lösung existiert, deren Vertauschung nicht in TL_N ist. Danach werden die ältesten Lösungen in der Lösungstabuliste TL_L gelöscht, bis eine Nachbarlösung existiert, die nicht tabu ist. Diese wird ausgewählt.

Algorithmus 2.15 *Tabu-Suche*
Sei N_α die verwendete Nachbarschaft, $f : \mathscr{X} \to \mathbb{R}^+$ die zu minimierende Zielfunktion, IT die maximale Anzahl von Iterationen, L_N die Länge der Nachbarschaftstabuliste und L_L die Länge der Lösungstabuliste.

1. *Ermittle Startlösung $x \in \mathscr{X}$.*
2. *Setze $x^* := x$, $k := 0$ und initialisiere die Listen TL_N und TL_L.*
3. *Durchsuche die gesamte Nachbarschaft von $N_\alpha(x)$ und finde die Mengen $NT(x)$, $TV(x)$ und $TN(x)$.*
4. *Falls $NT(x) \cup TV(x) \neq \emptyset$, wähle denjenigen Nachbarn x' mit $f(x') = \min\{f(y)|y \in NT(x) \cup TV(x)\}$ und gehe zu Schritt 8.*
5. *Falls $N_\alpha(x) \cup TL_N \neq \emptyset$, lösche den ältesten Nachbarschaftsschritt in TL_N solange, bis $N_\alpha(x) \setminus TL_N \neq \emptyset$.*
6. *Falls $(N_\alpha(x) \setminus TL_N) \cup TL_L \neq \emptyset$, lösche die älteste Lösung in TL_L solange, bis $(N_\alpha(x) \setminus TL_N) \setminus TL_L \neq \emptyset$.*
7. *Wähle den Nachbarn x' mit $f(x') = \min\{f(y)|y \in (N_\alpha(x) \setminus TL_N) \setminus TL_L\}$*
8. *Setze $x := x'$.*
9. *Falls $f(x) < f(x^*)$, setze $x^* := x$ und $k := 0$, sonst $k := k + 1$.*
10. *Falls $k = IT$, stopp. Gib x^* aus.*
11. *Gehe zu Schritt 3.*

Diese Tabu-Strategie mit zwei Tabulisten wurde gewählt, da die Nachbarschaften N_α, $\alpha = 1, \ldots, 5$, die Eigenschaft haben, dass verschiedene Nachbarschaftsschritte zu einer Lösung mit einem gleichen kritischen Pfad führen können. Die kritischen Pfade müssen nur bei Lösungen mit dem gleichen Funktionswert für C_{max} verglichen werden. Eine Lösung ist im Sinne der Lösungsliste genau dann tabu, wenn ein kritischer Pfad der Lösung mit einem kritischen Pfad einer Lösung in der Liste übereinstimmt. Dabei wird zwischen unterschiedlichen Reihenfolgen von Vorgängen innerhalb eines Blocks auf dem kritischen Pfad nicht unterschieden.

Der Nachteil der Tabu-Suche ist die große Anzahl der Nachbarn. Da in jedem Nachbarschaftsschritt alle Nachbarn erzeugt werden müssen, wird die

Tabu-Suche bei vielen Nachbarn stark verlangsamt. Dies gilt ebenso für die volle lokale Suche. Die schnelle lokale Suche, der Sidestep-Algorithmus, der Threshold-Accepting-Algorithmus und das Simulated Annealing akzeptieren den ersten Nachbarn mit besserem Funktionswert. Für einen Nachbarschaftsschritt müssen weniger Zielfunktionswerte berechnet werden. Die Auswirkungen dieser Strategien werden im nächsten Abschnitt untersucht.

2.2 Analyse der Heuristiken

Bei Optimierungsheuristiken hängen sowohl Laufzeit als auch Qualität der gefundenen Lösungen von Parametern der Algorithmen ab. Diese Parameter können die Anzahl von Iterationen, Länge von Tabulisten, verwendete Nachbarschaften, die Anzahl der Elternlösungen usw. sein. Die Analysen und Tests werden durchgeführt, um gute Parameter für verschiedene Restriktionen der praktischen Job-Shop-Probleme und für verschiedene Zielfunktionen zu bestimmen.

Die Algorithmen wurden nicht in einer Wettbewerbssituation mit aus der Literatur bekannten Heuristiken getestet. Zum einen ist es schwierig, die Algorithmen fair zu testen, da der Aufwand für eine konkurrenzfähige Implementation der Algorithmen und zum Tuning der Parameter sehr hoch ist [26]. Des Weiteren existieren für einige hier behandelten Problemklassen keine Referenzalgorithmen in der Literatur. Aus diesem Grund werden im Folgenden nur die in Abschnitt 2.1.3 vorgestellten Algorithmen getestet. Diese wurden alle mit den gleichen Mitteln — Datenstrukturen und Programmiersprache — implementiert.

Die Heuristiken wurden auf einer Menge von Benchmark-Instanzen getestet. Hierzu wurden 242 Standardprobleme für $J||C_{max}$ und 50 für $J||L_{max}$ verwendet. Diese Instanzen wurden aus verschiedenen Quellen gewählt, um eine Anpassung der Algorithmen an die Instanzen zu verhindern. Für praktische Job-Shop-Probleme standen keine Benchmark-Instanzen aus der Literatur zur Verfügung. Hierfür wurde ein Benchmarkgenerator entwickelt. Damit wurden 70 Probleminstanzen für praktische Job-Shop-Probleme mit verschiedenen Restriktionen und Dimensionen generiert.

2.2.1 Benchmark-Instanzen für klassische Job-Shop--Probleme

Die folgende Liste enthält Informationen zu den verwendeten Benchmark-Instanzen. Es wird kurz auf die Generierung der Bearbeitungszeiten, der Reihenfolgebeziehungen und der Fälligkeitstermine eingegangen. Die Bearbeitungszeiten wurden bei allen Instanzen (bis auf die ORB-Instanzen) gleichverteilt aus einem vorgegebenen Intervall gezogen. Die Dimensionen der Instanzen und Daten zu den Werten der Zielfunktionen C_{max}, $\sum T$, $\sum C$, $\sum F$ und $|L|_{max}$ können den angegebenen Tabellen entnommen werden. Detailliertere Informationen sind in [29] und [30] zu finden.
Zu den Bezeichnungen:

FT - 3 Probleme mit verschiedenen Dimensionen von Fisher und Thompson (1963): 6x6, 10x10, 20x5 (Tabelle A.7). Die Bearbeitungszeiten wurden für die Instanz FT06 aus dem Intervall [1,10] und für die Instanzen FT10 und FT20 aus dem Intervall [1,99] generiert. Um praktische Shop Scheduling-Probleme zu simulieren, wurden bei den Instanzen FT10 und FT20 Maschinen mit niedrigen Nummern frühen Operationen und Maschinen mit höheren Nummern späteren Operationen zugeordnet.

SWV - 20 Probleme mit 4 verschiedenen Größenordnungen von Storer, Wu und Vaccari (1992) (Tabelle A.7). Die Bearbeitungszeiten wurden aus dem Intervall [1,100] generiert. Die Menge der Maschinen wurde in k gleich große Teilmengen zerlegt. Die Reihenfolgebeziehungen wurden so festgelegt, dass die Jobs erst auf einer gleichverteilt gewählten Permutation der Maschinen in der ersten Teilmenge bearbeitet werden müssen und dann zu einer Permutation der nächsten Teilmenge übergehen. Für SWV1 bis SWV15 ist $k = 2$ und für SWV16 bis SWV20 ist $k = 1$.

YN - 4 Probleme von Yamada und Nakano (1992) mit 20 Jobs und 20 Maschinen (Tabelle A.8). Die Bearbeitungszeiten wurden aus dem Intervall [10,50] gewählt.

ABZ - 5 Probleme von Adams, Balas und Zawack (1988) (Tabelle A.8). Die Bearbeitungszeiten wurden aus [50,100] für ABZ5, aus [25,100] für ABZ6 und aus [11,40] für ABZ7 bis ABZ9 gewählt.

ORB - 10 Probleme, die von Applegate und Cook genutzt wurden (1986) (Tabelle A.8). Sie wurden als speziell erzeugte schwierige Probleme bezeichnet.

LA - 40 Probleme von Lawrence (1984) mit 8 verschiedenen Dimensionen (Tabellen A.9 und A.10). Die Bearbeitungszeiten wurden aus dem Intervall [5,99] generiert.

TA - 80 Probleme mit 8 verschiedenen Dimensionen von Taillard (1993) [57] (Tabellen A.11 bis A.14). Das Intervall für die Erzeugung der Bearbeitungszeiten war [1,99].

DMU - 80 Probleme mit 8 verschiedenen Dimensionen von Demirkol, Mehta und Uzsoy (1998) [14] (Tabellen A.15 bis A.18). Die Bearbeitungszeiten wurden aus [1,200] erzeugt. Die Reihenfolgebeziehungen für die ersten 40 Probleme (DMU1-DMU40 oder rcmax) wurden mit $k = 1$ ($J||C_{max}$) generiert. Für die Probleme DMU41-DMU80 oder cscmax wurde $k = 2$ ($J|2SETS|C_{max}$) verwendet (siehe SWV1-SWV15).

Alle hier verwendeten Instanzen sind frei verfügbar. Die FT, LA, ABZ, ORB, SWV und YN Benchmarks sind unter der URL `http://www.ms.ic.ac.uk/jeb/pub/jobshop1.txt` zu finden. Diese Seite ist Teil der Operations Research Library der Management School des Imperial College in London, UK. Die TA-Instanzen sind unter der URL `http://www.eivd.ch/ina/Collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html` zu finden. Dies ist eine Seite von Prof. Taillard an der EIVD Hochschule für Technik Waadt in der Schweiz. Die DMU-Instanzen sind verfügbar von Prof. Uzsoy an der Purdue Electronics Manufacturing Research Group, Purdue University, Indiana, USA. Die URL lautet: `http://gilbreth.ecn.purdue.edu/~uzsoy2/benchmark/problems.html`.

Alle diese Instanzen wurden für die Zielfunktion C_{max} generiert. Um diese auch für die Zielfunktionen $\sum T$ und $|L|_{max}$ verwenden zu können, mussten Fälligkeitstermine d_j für alle $j = 1, \ldots, n$ festgelegt werden. Die d_j wurden wie folgt berechnet:

$$d_j = \left\lfloor \frac{3}{2} \sum_{i=1}^{m_j} p_{ij} \right\rfloor .$$

Der Faktor $\frac{3}{2}$ ergibt für Instanzen mit $|\mathscr{J}| > 2|\mathscr{M}|$ knappe Fälligkeitsdaten. Er wurde aber für alle Benchmarkprobleme gleich gewählt, um die Werte einfach berechnen und nachvollziehen zu können. Von diesen 242 Benchmark-Instanzen wurden 241 verwendet. Das Problem FT06 mit 36 Operationen ist akademischer Natur und wurde in der Arbeit nur als Beispiel für den disjunktiven Graphen (Abbildungen 1.1, 1.2) und das Gantt-Diagramm (Abbildung 1.3) genutzt.

Demirkol *et al.* generierte ebenfalls 320 Benchmark-Instanzen für das Job-Shop-Problem mit Zielfunktion L_{max}. Von diesen wurden hier 50 Instanzen verwendet (Tabellen A.19, A.20 und A.21). Davon haben jeweils 20 Instanzen die Dimensionen 20×15 und 20×20, sowie zehn Instanzen die Dimension 50×20. Bei 25 Instanzen ist $k = 1$ ($J||L_{max}$). Bei den restlichen 25 ist $k = 2$ ($J|2SETS|L_{max}$). Die Bearbeitungszeiten wurden gemäß Gleichverteilung aus [1,200] gezogen. Für die Fälligkeitstermine d_j wurden zwei Parameter τ und R verwendet. Damit wurden eine untere Schranke d_{min} und eine obere Schranke d_{max} für die d_j berechnet:

$$d_{min} = (1 - \tau)|\mathscr{J}|\frac{200 + 1}{2}\left(1 - \frac{R}{2}\right)$$

und

$$d_{max} = (1 - \tau)|\mathscr{J}|\frac{200 + 1}{2}\left(1 + \frac{R}{2}\right).$$

Für jeden Job J_j wurde eine Zufallszahl $p \in [0, 1]$ gemäß Gleichverteilung erzeugt. Die d_j wurden wie folgt berechnet:

$$d_j = \lfloor d_{min} + p(d_{max} - d_{min} + 1)\rfloor.$$

Von den 241 C_{max} Instanzen sind 134 gelöst und für die restlichen stehen gute untere und obere Schranken zur Verfügung. Auf die L_{max} Instanzen wurden bisher nur Algorithmen von Demirkol *et al.* angewendet. Dementsprechend stammen die oberen und unteren Schranken für L_{max} von dort. Die angegebenen unteren Schranken für C_{max} bei den 50 L_{max} Instanzen wurden mit Hilfe der Ein-Maschinen-Schranke von Carlier [11] berechnet.

2.2.2 Generierung von Benchmark-Instanzen für praktische Job-Shop-Probleme

Für praktische Job-Shop-Probleme wurden 70 Instanzen mit verschiedenen Dimensionen, Nebenbedingungen und Verallgemeinerungen generiert. Der hierfür entworfene Generator wurde mit dem Ziel entwickelt, mit möglichst wenigen Parametern realistische Instanzen zu erzeugen.

Generierung der Maschinendaten

Die Anzahl der Maschinen wurde zufällig gemäß Gleichverteilung aus einem Intervall $[M_{min}, M_{max}]$ bestimmt. Für die Generierung der Schichtzei-

ten wurde eine Wahrscheinlichkeitsverteilung $\Lambda = (\lambda_1, \ldots, \lambda_n)$ mit $\lambda_j \geq 0$ und $\sum_{j=1}^{n} \lambda_j = 1$ vorgegeben. n ist die Periodenlänge und bei der Erzeugung der Benchmark-Instanzen immer 24. Zu jeder Maschine M_i wird eine gleichverteilte Zufallszahl $p \in [0, 1]$ erzeugt und $st_i := k$ gesetzt, falls $p \in (\sum_{j=1}^{k-1} \lambda_j, \sum_{j=1}^{k} \lambda_j]$ liegt.

Generierung der Ressourcendaten

Die Anzahl der Ressourcen wurde ebenfalls zufällig gleichverteilt aus einem Intervall $[R_{min}, R_{max}]$ generiert. Die Verfügbarkeit ra_i der Ressource R_i wurde aus dem Intervall $[1, R^S]$ generiert, wobei die obere Schranke R^S als Parameter gewählt war.

Generierung der Auftragsdaten

Die Auftragsanzahl wurde aus einem vorgegebenen Intervall $[A_{min}, A_{max}]$ gewählt. Die Aufträge sollten eine Baumstruktur haben. Aus diesem Grund wurden zur Generierung der Aufträge eine maximale Tiefe des Baumes T_{max} und ein maximaler Verzweigungsgrad V_{max} als Parameter eingeführt. In jedem Auftrag muss mindestens ein Job in der Wurzel enthalten sein. Dieser wurde zu Beginn erzeugt. Falls die maximale Tiefe des Auftrages noch nicht erreicht war, wurden für diesen Job k Vorgängerjobs erzeugt, wobei $k \in [0, V_{max}]$ gilt. Dies wurde in den nächsten Ebenen des Auftragsbaumes fortgeführt, bis die maximale Tiefe erreicht wurde oder keine Vorgänger mehr erzeugt wurden ($k = 0$). $T_{max} = 1$ bedeutet, dass jeder Auftrag nur einen Job in der Wurzel hat.

Nachdem die Struktur der Aufträge und damit die Anzahl der Jobs feststand, wurde für jeden Job die Anzahl der Operationen aus dem Intervall $[O_{min}, O_{max}]$ generiert. Hier ist O_{max} kleiner gleich der Anzahl der für diese Instanz erzeugten Maschinen, da jeder Job auf jeder Maschine höchstens einmal bearbeitet werden muss. Für jede Operation in diesem Job wird eine Maschine ohne Zurücklegen gezogen. Die Bearbeitungsdauer p_{ji} eines Vorgangs wurde in Abhängigkeit der Schichtzeit st_i der zugehörigen Maschine und des Intervalls $[Z_{min}, Z_{max}]$ bestimmt. Die Bearbeitungsdauer eines Vorgangs wurde zufällig gleichverteilt aus $[\lceil \frac{st_i}{24} Z_{min} \rceil, \lceil \frac{st_i}{24} Z_{max} \rceil]$ generiert. Um die Reihenfolgebeziehungen innerhalb von Jobs zu generieren, wurden zwei Parameter vorgegeben. P^O gibt die maximale Anzahl zueinander paralleler Operationen innerhalb eines Jobs an. $p^p \in [0, 1]$ gibt die Wahrscheinlichkeit

	Instanzen			
Parameter	1-5	6-10	11-15	16-20
$[M_{min}, M_{max}]$	$[15, 15]$	$[20, 20]$	$[20, 20]$	$[30, 30]$
Λ	$\lambda_{24} = 1$	$\lambda_{24} = 1$	$\lambda_{24} = 1$	$\lambda_{24} = 1$
$[R_{min}, R_{max}]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$
R^S	-	-	-	-
$[A_{min}, A_{max}]$	$[20, 20]$	$[30, 30]$	$[40, 40]$	$[50, 50]$
T_{max}	1	1	1	1
V_{max}	-	-	-	-
$[O_{min}, O_{max}]$	$[15, 15]$	$[20, 20]$	$[20, 20]$	$[30, 30]$
$[Z_{min}, Z_{max}]$	$[1, 100]$	$[1, 100]$	$[1, 100]$	$[1, 100]$
P^O	3	3	3	3
p^p	0.5	0.5	0.5	0.5
RD_{max}	-	-	-	-
p^{RD}	-	-	-	-
RA_{max}	-	-	-	-
p^{RA}	-	-	-	-
F	1.5	1.5	2	2

Tabelle 2.1: *Generatorparameter der erzeugten Instanzen 1 bis 20 D||multi*

an, für die der nächste generierte Task parallel oder Nachfolger ist. Nachdem die erste Operation im Job erzeugt wurde, wird für die nächste Operation eine Zufallszahl $p \in [0, 1]$ generiert. Falls $p \leq p^p$ gilt und die maximale Anzahl paralleler Vorgänge nicht überschritten wird, ist der Vorgang parallel, ansonsten Nachfolger.

Falls die Anzahl der Ressourcen für die Instanz größer als 0 ist, muss für jede Operation der Vektor $rd = (rd_1, \ldots, rd_r)$ der benötigten Ressourcen generiert werden. Der Generatorparameter RD_{max} gibt hierbei an, wie viele der Werte rd_i maximal größer als 0 sein können. Der Parameter RA_{max} gibt eine oberer Schranke für diese rd_i an. Die rd_i müssen natürlich weiterhin kleiner gleich der vorhandenen Anzahl ra_i der Ressource i sein.

Für die Erzeugung der benötigten Ressourcen sind noch die Parameter $p^{RD} \in [0, 1]$ und $p^{RA} \in [0, 1]$ nötig. Zuerst wird eine Zufallszahl $p \in [0, 1]$ erzeugt. Falls $p \leq p^{RD}$ ist und die Anzahl der Ressourcen für diese Operation kleiner RD_{max} ist, wird dem Vorgang eine weitere Ressource zugeordnet. Ansonsten wird zur nächsten Operation übergegangen. Nachdem für alle Operationen die benötigten Ressourcen generiert wurden, wird die benötigte Menge pro

Parameter	Instanzen			
	21-25	26-30	31-35	36-40
$[M_{min}, M_{max}]$	$[15, 15]$	$[20, 20]$	$[20, 20]$	$[30, 30]$
Λ	$\lambda_{8,16,24} = \frac{1}{3}$	$\lambda_i = \frac{1}{3}, i = 8, 16, 24$	$\lambda_{8,16,24} = \frac{1}{3}$	$\lambda_{8,16,24} = \frac{1}{3}$
$[R_{min}, R_{max}]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$
R^S	-	-	-	-
$[A_{min}, A_{max}]$	$[20, 20]$	$[30, 30]$	$[40, 40]$	$[50, 50]$
T_{max}	1	1	1	1
V_{max}	-	-	-	-
$[O_{min}, O_{max}]$	$[15, 15]$	$[20, 20]$	$[20, 20]$	$[30, 30]$
$[Z_{min}, Z_{max}]$	$[1, 100]$	$[1, 100]$	$[1, 100]$	$[1, 100]$
P^O	3	3	3	3
p^p	0.5	0.5	0.5	0.5
RD_{max}	-	-	-	-
p^{RD}	-	-	-	-
RA_{max}	-	-	-	-
p^{RA}	-	-	-	-
F	1.5	1.5	2	2

Tabelle 2.2: *Generatorparameter der erzeugten Instanzen 21 bis 40 D|resum|multi*

Ressource nach dem gleichen Schema mit den Parametern RA_{max} und p^{RA} zugeordnet. Von jeder zugeordneten Ressource wird aber mindestens die Menge 1 benötigt.

Die Fälligkeitstermine wurden wie bei den klassischen Job-Shop-Problemen erzeugt. Es wurde der längste Weg in jedem Auftragsbaum von den Blättern zur Wurzel ermittelt. Dabei wurde die Bearbeitungszeit der Vorgänge um die minimale Dauer der Stillstandszeiten verlängert. Die Länge des längsten Weges, multipliziert mit einem Generatorparameter $F \in \mathbb{R}^+$, ergab den Fälligkeitstermin d_j für jeden Auftrag. Die Tabellen 2.1 bis 2.4 geben die Generatorparameter für die hier verwendeten Benchmark-Instanzen der praktischen Job-Shop-Probleme an. Diese Instanzen lassen sich in vier Gruppen unterteilen. Die erste Gruppe D1-D20 besteht aus Mixed-Shop-Probleme mit parallelen Tasks. Die zweite Gruppe DR21-DR40 enthält Mixed-Shop-Probleme mit parallelen Tasks und Stillstandszeiten der Maschinen. Die Maschinen sind hierbei 8, 16 oder 24 Stunden pro Tag verfügbar. Dies simuliert ein Schichtsystem. DRI41-DRI55 sind Instanzen mit Montageaufträgen. Es exi-

Parameter	Instanzen		
	41-45	46-50	51-55
$[M_{min}, M_{max}]$	$[15, 15]$	$[15, 15]$	$[20, 20]$
Λ	$\lambda_{8,16,24} = \frac{1}{3}$	$\lambda_i = \frac{1}{3}, i = 8, 16, 24$	$\lambda_{8,16,24} = \frac{1}{3}$
$[R_{min}, R_{max}]$	$[0, 0]$	$[0, 0]$	$[0, 0]$
R^S	-	-	-
$[A_{min}, A_{max}]$	$[15, 15]$	$[20, 20]$	$[20, 20]$
T_{max}	3	3	3
V_{max}	3	3	3
$[O_{min}, O_{max}]$	$[8, 15]$	$[8, 15]$	$[10, 20]$
$[Z_{min}, Z_{max}]$	$[1, 100]$	$[1, 100]$	$[1, 100]$
P^O	3	3	3
p^p	0.5	0.5	0.5
RD_{max}	-	-	-
p^{RD}	-	-	-
RA_{max}	-	-	-
p^{RA}	-	-	-
F	2	2	2

Tabelle 2.3: *Generatorparameter der erzeugten Instanzen 41 bis 55* $D|resum, intree|multi$

stieren Reihenfolgebeziehungen zwischen Jobs, die Montageaufträge simulieren. Die letzte Gruppe DRIR56-DRIR70 hat als zusätzliche Nebenbedingung erneuerbare diskrete Ressourcen. Eine Instanz einer höheren Gruppe beinhaltet die Verallgemeinerungen der vorhergehenden Instanzen. Die Gruppen sind in Untergruppen von je 5 Instanzen mit verschiedenen Anzahlen von Aufträgen, Maschinen und Ressourcen unterteilt. Die besten erreichten Zielfunktionswerte für diese Benchmarks sind im Anhang in den Tabellen A.22 bis A.25 zu finden.

2.2.3 Analyse der Nachbarschaften und der Rekombination

Die Wahl der Nachbarschaft in der lokalen Suche und die Anzahl der Elternlösungen bei der Rekombination determinieren grundlegende Eigenschaften der genetischen lokalen Suche. In diesem Abschnitt wird getestet, welche Nachbarschaft und welche Anzahl von Elternlösungen bei einer gegebenen

Parameter	Instanzen		
	41-45	46-50	51-55
$[M_{min}, M_{max}]$	$[15, 15]$	$[15, 15]$	$[20, 20]$
Λ	$\lambda_{8,16,24} = \frac{1}{3}$	$\lambda_i = \frac{1}{3}, i = 8, 16, 24$	$\lambda_{8,16,24} = \frac{1}{3}$
$[R_{min}, R_{max}]$	$[15, 15]$	$[15, 15]$	$[20, 20]$
R^S	3	3	3
$[A_{min}, A_{max}]$	$[15, 15]$	$[20, 20]$	$[20, 20]$
T_{max}	3	3	3
V_{max}	3	3	3
$[O_{min}, O_{max}]$	$[8, 15]$	$[8, 15]$	$[10, 20]$
$[Z_{min}, Z_{max}]$	$[1, 100]$	$[1, 100]$	$[1, 100]$
P^O	3	3	3
p^p	0.5	0.5	0.5
RD_{max}	2	2	2
p^{RD}	0.6	0.6	0.75
RA_{max}	3	3	3
p^{RA}	0.6	0.6	0.75
F	2	2	2

Tabelle 2.4: *Generatorparameter der erzeugten Instanzen 56 bis 70* $D|resum, intree, res|multi$

Zielfunktion am günstigsten ist. Um den Einfluss anderer Parameter auszuschließen, wurde auf die einfachen Varianten der lokalen Suche (Algorithmen 2.10 und 2.11) zurückgegriffen. Da im Folgenden Rechenzeitschranken angegeben werden, muss der Prozessortyp angegeben werden. Die Testläufe zum Vergleich der Algorithmen wurden auf einem Pentium III mit 1GHz Taktrate durchgeführt. Auch alle anderen Rechenzeitangaben beziehen sich auf diesen Prozessortyp und diese Taktfrequenz.

Testmenge

Da ein Test der Algorithmen auf allen 371 Benchmark-Instanzen zu zeitaufwändig ist, wurde eine Teilmenge von 40 Instanzen als Referenzmenge bestimmt. Diese Teilmenge wurde so gewählt, dass alle Dimensionen und alle Nebenbedingungen abgedeckt werden. Es wurden nur Instanzen ab 300 Operationen berücksichtigt. Die klassischen Job-Shop-Instanzen wurden aus verschiedenen Quellen gewählt, um unterschiedliche Parameter für die Be-

SWV07	SWV10	ABZ8	TA13	TA16	DMU1	DMU41	YN3
YN4	TA21	TA30	DMU6	DMU46	TA40	DMU11	DMU51
SWV12	TA41	DMU16	DMU56	DMU61	DMU62	DMU26	DMU67
DMU71	DMU76	D1	D6	D11	D16	DR21	DR26
DR31	DR36	DRI41	DRI46	DRI51	DRIR56	DRIR61	DRIR66

Tabelle 2.5: *Referenzmenge der Benchmark-Instanzen*

arbeitungsdauern und die Reihenfolgebeziehungen zu berücksichtigen. Von den praktischen Job-Shop-Problemen wurde aus jeder Untergruppe eine Instanz gewählt. Die Bezeichnungen der Instanzen stimmen mit den Tabellen im Anhang überein.

Nachbarschaften

Durch die in Abschnitt 2.1.2 definierten Nachbarschaften hat jede Lösung mit N Vorgängen in der aufsteigenden Nummerierung $\binom{N}{2}$ potentielle Nachbarn. Die Nachbarschaften N_4 und N_5 sind durch ihre Struktur diejenigen mit der größten Anzahl von zulässigen Nachbarn. Um diese einzuschränken, wurden bei N_4 und N_5 nur die Nachbarn einer Lösung x zugelassen, bei der die zu vertauschenden Vorgänge T_i und T_j einer der folgenden Bedingungen genügen:

1. T_i und T_j müssen auf der gleichen Maschine bearbeitet werden.
2. Die Vorgänge gehören zum gleichen Job und sind zueinander parallel.
3. Mindestens eine Ressource wird von T_i und T_j benötigt.

Bei den Nachbarschaften N_1, N_2 und N_3 werden von allen zulässigen Nachbarn die Zielfunktionswerte berechnet.

Vergleich der Nachbarschaften

In diesem Abschnitt werden die Nachbarschaften anhand der Güte der erreichten Zielfunktionswerte auf der Testmenge der Instanzen mit den Zielfunktionen C_{max}, $\sum C$, $\sum T$ und $|L|_{max}$ verglichen. Diese vier Zielfunktionen wurden nach folgenden Kriterien ausgewählt:

- je zwei Zielfunktionen minimieren ein Maximum und zwei sind vom Summentyp;
- zwei Zielfunktionen hängen vom Fälligkeitstermin d_j ab;
- es ist mit $|L|_{max}$ eine nichtreguläre Zielfunktion vertreten.

Zur Erzeugung der Lösungen wurde die schnelle lokale Suche (Algorithmus 2.11) verwendet. Für jede Instanz wurden 25 unabhängige Läufe der schnellen lokalen Suche beginnend mit zufällig erzeugten Startlösungen und allen 5 Nachbarschaften durchgeführt.

Für jede Instanz i, $i = 1, \ldots, 40$, und jede Nachbarschaft N_α, $\alpha = 1, \ldots, 5$, wurde aus dem Pool der 25 Lösungen diejenige Lösung $x_i^{N_\alpha}$ mit dem besten Funktionswert bestimmt. Jetzt wurden die, mit den verschiedenen Nachbarschaften erreichten besten Lösungen einer Instanz i verglichen und Rangzahlen $R(x_i^{N_\alpha})$, $\alpha = 1, \ldots, 5$, vergeben. Die Lösung mit dem kleinsten Funktionswert bekam den Rang 1, die zweitbeste den Rang 2 usw. Falls mehrere Lösungen den gleichen Funktionswert hatten, wurde der gleiche Rang vergeben. Die nächstschlechtere Lösung bekam eine entsprechend größere Rangzahl. Zu jeder Nachbarschaft N_α, $\alpha = 1, \ldots, 5$, wurde die Anzahl der Rangzahlen $R_j^{N_\alpha}$, $j = 1, \ldots, 5$, für jeden Rang bestimmt:

$$R_j^{N_\alpha} = |\{R(x_i^{N_\alpha}) | R(x_i^{N_\alpha}) = j, i = 1, \ldots, 40, \alpha = 1, \ldots, 5\}|$$

Die Tabelle 2.6 listet die Daten für die 4 untersuchten Zielfunktionen auf. Die Nachbarschaft N_4 stellt sich hierbei als die beste Alternative heraus. In etwa der Hälfte der Fälle wird unabhängig von der Zielfunktion mit dieser Nachbarschaft der beste Zielfunktionswert erreicht. Die Rangzahl 1 erreichen auch die Nachbarschaften N_1 und N_5. Die Tabelle 2.7 zeigt die durchschnittlichen Anzahlen zulässiger Nachbarn bei den verschiedenen Nachbarschaften und unterschiedlichen Problemdimensionen. Hierzu wurde für einer Teilmenge der klassischen Job-Shop-Benchmark-Instanzen zufällige Startlösungen erzeugt und die zulässigen Nachbarn gezählt. Die Teilmenge der Instanzen besteht aus 11 Gruppen verschiedener Dimensionen (Job-, Maschinen- und Tasksanzahlen). Für jede Gruppe wurden jeweils 10 Instanzen ausgewählt. Die Bezeichnungen der Instanzen sind in Tabelle A.1 im Anhang aufgelistet. Zu jeder Instanz wurden 10 zufällige Startlösungen erzeugt und die Anzahl der zulässigen Nachbarn für jeder Lösung ermittelt. Die so ermittelte durchschnittliche Anzahl von zulässigen Nachbarn pro Instanz wurde wiederum über die 10 Instanzen einer Problemdimension gemittelt. Dies ergab

	C_{max}					L_{max}				
	N_1	N_2	N_3	N_4	N_5	N_1	N_2	N_3	N_4	N_5
$R_1^{N_\alpha}$	10	0	0	19	11	7	0	0	20	13
$R_2^{N_\alpha}$	9	0	0	13	18	5	0	0	14	21
$R_3^{N_\alpha}$	19	2	2	7	11	23	4	1	6	6
$R_4^{N_\alpha}$	2	27	9	1	0	5	24	11	0	0
$R_5^{N_\alpha}$	0	11	29	0	0	0	12	28	0	0
	$\sum C$					$\sum T$				
	N_1	N_2	N_3	N_4	N_5	N_1	N_2	N_3	N_4	N_5
$R_1^{N_\alpha}$	10	0	0	20	10	12	0	0	24	4
$R_2^{N_\alpha}$	11	0	0	14	15	10	0	0	13	17
$R_3^{N_\alpha}$	18	0	1	6	15	17	0	1	3	19
$R_4^{N_\alpha}$	1	22	17	0	0	1	20	19	0	0
$R_5^{N_\alpha}$	0	18	22	0	0	0	20	20	0	0

Tabelle 2.6: *Rangzahlen der besten Lösungen für alle Nachbarschaften und 4 Zielfunktionen auf der Testmenge (schnelle lokale Suche mit zufälligen Startlösungen)*

$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	N_1	N_2	N_3	N_4	N_5	Tasks
10	10	392.25	805.69	741.74	1457.52	1149.22	100
15	15	1405.08	2943.49	2701.62	5435.11	4237.33	225
20	15	2534.86	5317.61	4870.82	9808.43	7607.69	300
20	20	3434.04	7224.12	6649.14	13493.26	10459.99	400
30	15	5821.23	12188.87	11167.91	22486.80	17347.81	450
30	20	7832.04	16530.92	15247.24	30908.16	23772.97	600
40	15	10385.96	21839.53	19995.60	40275.17	30945.66	600
40	20	14026.20	29625.66	27285.38	55351.04	42468.31	800
50	10	10669.62	22052.90	20139.91	39745.20	30505.66	500
50	15	16351.22	34322.48	31411.02	63283.56	48575.64	750
50	20	22006.86	46539.72	42870.64	86960.36	66571.00	1000

Tabelle 2.7: *Durchschnittliche Anzahl von zulässigen Nachbarn bei den 5 Nachbarschaftsfunktionen*

die durchschnittlichen Nachbarschaftszahlen für die verschiedenen Problemdimensionen und Nachbarschaften. Tabelle A.2 im Anhang enthält die durchschnittlichen Nachbarschaftszahlen der praktischen Job-Shop-Benchmark-Instanzen zusammengefasst zu den Untergruppen von je 5 Instanzen.

N_4 ist die Nachbarschaft mit der größten Anzahl zulässiger Nachbarn. Dies ist der Grund für die guten Ergebnisse, die die schnelle lokale Suche mit dieser Nachbarschaft erreicht. Diese Ergebnisse werden mit einer entsprechend höheren Laufzeit der Algorithmen erkauft.

Die Lokale-Suche-Heuristiken werden im Weiteren in der genetischen lokalen Suche (Algorithmus 2.7) verwendet. Da in praktischen Anwendungen die Antwortzeit von Bedeutung ist, wurde die genetische lokale Suche nach einer festen Rechenzeitschranke abgebrochen. Es wurde im Gegensatz zum Vergleich mit der schnellen lokalen Suche ohne Rekombination, keine feste Anzahl von Läufen durchgeführt. Im Kern des Hybrid-Algorithmus wurde die schnelle lokale Suche genutzt. Der Pool des genetischen Algorithmus enthielt 25 lokale Optima. Für die ersten 25 lokalen Optima wurde die schnelle lokale Suche von zufälligen Startlösungen aus gestartet. Danach wurden Mittelwertlösungen ohne Mutation als Startlösungen verwendet. Für die Mittelwertbildung wurden jeweils 2 Lösungen aus dem Pool zufällig gemäß Gleichverteilung gewählt. Die durch die schnelle lokale Suche entstandene Lösung wurde nur in den Pool eingefügt, falls noch keine Lösung mit dem gleichen kritischen Pfad im Pool existierte. Falls die Lösung in den Pool eingefügt wurde, wurde die Lösung mit dem schlechtesten Zielfunktionswert aus dem Pool entfernt. Der Algorithmus wurde nach 300 Sekunden Rechenzeit gestoppt, wobei der letzte Lauf der schnellen lokalen Suche vollständig durchgeführt wurde.

Es wurde jeweils ein Lauf dieser Variante der genetischen lokalen Suche mit allen 40 Testinstanzen und allen 5 Nachbarschaften durchgeführt. Für jede Instanz i, $i = 1, \ldots, 40$, und jede Nachbarschaft N_α, $\alpha = 1, \ldots, 5$, wurde aus dem Pool der 25 Lösungen diejenige Lösung $x_i^{N_\alpha}$ mit dem besten Funktionswert bestimmt. Die Rangzahlen $R_j^{N_\alpha}$ wurden wie oben definiert ermittelt. Tabelle 2.8 zeigt die $R_j^{N_\alpha}$ für die 4 untersuchten Zielfunktionen. Die Nachbarschaft N_1 stellt sich hierbei als die beste Alternative heraus, gefolgt von den Nachbarschaften N_4 und N_5. Die Nachbarschaften N_2 und N_3 sowie N_4 und N_5 liefern strukturbedingt ähnliche Werte. Die qualitativ gleichen Ergebnisse ergeben sich auch, wenn statt der jeweils besten Zielfunktionswerte im Pool die durchschnittlichen Zielfunktionswerte verglichen werden.

Neben den Rangzahlen sind auch die relativen Abweichungen zum besten lokalen Minimum interessant. Aus diesem Grund wurde für jede Instanz der beste der 5 Zielfunktionswerte bestimmt und die relativen Differenzen Δ berechnet. Diese Differenzen wurden für jede Nachbarschaft über alle 40 Testinstanzen summiert und gemittelt. In Tabelle 2.9 finden sich die durchschnittlichen Abweichungen $\overline{\Delta}$ für die Zielfunktionen C_{max}, $\sum C$ und L_{max}.

	C_{max}					L_{max}				
	N_1	N_2	N_3	N_4	N_5	N_1	N_2	N_3	N_4	N_5
$R_1^{N_\alpha}$	30	1	1	7	2	26	2	0	5	7
$R_2^{N_\alpha}$	8	6	1	7	17	7	3	2	13	16
$R_3^{N_\alpha}$	1	5	3	16	15	4	8	1	12	14
$R_4^{N_\alpha}$	1	17	11	5	6	3	17	11	7	2
$R_5^{N_\alpha}$	0	11	24	5	0	0	10	26	3	1
	$\sum C$					$\sum T$				
	N_1	N_2	N_3	N_4	N_5	N_1	N_2	N_3	N_4	N_5
$R_1^{N_\alpha}$	26	0	0	5	9	24	0	0	9	9
$R_2^{N_\alpha}$	11	1	1	17	10	11	0	0	14	14
$R_3^{N_\alpha}$	3	1	0	17	19	5	1	1	16	16
$R_4^{N_\alpha}$	0	22	16	1	1	0	20	19	0	1
$R_5^{N_\alpha}$	0	16	23	0	1	0	19	20	1	0

Tabelle 2.8: Rangzahlen der besten Lösungen für alle Nachbarschaften und 4 Zielfunktionen auf der Testmenge (genetische lokale Suche)

	N_1	N_2	N_3	N_4	N_5
$\overline{\Delta}_{C_{max}}$	0.437	7.001	8.317	3.687	3.130
$\overline{\Delta}_{L_{max}}$	2.013	27.934	32.483	18.626	13.005
$\overline{\Delta}_{\sum C}$	0.834	8.976	10.515	2.261	2.517

Tabelle 2.9: *Durchschnittliche Abweichung zur besten Lösung bei allen Nachbarschaften auf der Testmenge*

Die Zielfunktion $\sum T$ wurde nicht berücksichtigt, da einige Minima den Zielfunktionswert 0 erreichten. Im Vergleich mit den Rangzahlen ergeben sich die gleichen Beobachtungen. Die Nachbarschaft N_1 liefert die besten Ergebnisse, gefolgt von N_4 und N_5.

Die Reihenfolge der Nachbarschaften N_1 und N_4 ist bei der genetischen lokalen Suche im Gegensatz zur reinen schnellen lokalen Suche vertauscht. Dies liegt an dem Laufzeitvorteil durch die geringere Anzahl von Nachbarn bei der Nachbarschaft N_1. Dadurch werden in dem Hybrid-Algorithmus mehr Läufe beginnend mit Rekombinationslösungen durchgeführt. Dies führt im begrenztem Zeithorizont unter Verwendung der Nachbarschaft N_1 zu besseren Zielfunktionswerten.

Rekombination

Die Anzahl der Elternlösungen im genetischen Algorithmus ist der zweite grundlegende Parameter für die genetische lokale Suche. In der Literatur wird häufig die Anzahl von zwei Eltern als optimal angegeben. So verwenden YAMADA und NAKANO [62] in ihrem genetischen Algorithmus für $J||C_{max}$ immer zwei Lösungen aus dem Pool als Elternlösungen. ROSE [49] beobachtet ebenfalls in seinem Algorithmus für $D||C_{max}$, dass die Elternzahl zwei optimal ist.

Hier wird die Frage untersucht, welche Elternzahl in der genetischen lokalen Suche für die 4 verschiedenen Zielfunktionen optimal ist. Es wurde wieder die genetische lokale Suche mit der schnellen lokalen Suche und Nachbarschaft N_1 verwendet. Der Pool des genetischen Algorithmus enthielt 25 lokale Optima. Für die ersten 25 lokalen Optima wurde die schnelle lokale Suche von zufälligen Startlösungen aus gestartet. Danach wurden Mittelwertlösungen ohne Mutation als Startlösungen verwendet. Für die Mittelwertbildung wurden jeweils 2, 3, 4 oder 5 Lösungen aus dem Pool zufällig gemäß Gleichverteilung gewählt. Die durch die schnelle lokale Suche entstandenen Lösung wurde nur in den Pool eingefügt, falls noch keine Lösung mit dem gleichen kritischen Pfad im Pool existierte. Falls die Lösung in den Pool eingefügt wurde, wurde die Lösung mit dem schlechtesten Zielfunktionswert aus dem Pool entfernt. Um den Einfluss der Anzahl der Generationen zu vergleichen, wurde der Algorithmus mit zwei Zeitschranken, 100 und 500 Sekunden, auf der Menge der 40 Instanzen getestet.

Die Rangzahlen R_j, $j = 1, \ldots, 4$, für die verschiedenen Elternzahlen, wurden wie oben ermittelt. Dazu wurden die jeweils besten Lösungen im Pool verglichen. Dies wurde für alle 4 Zielfunktionen durchgeführt. Tabelle 2.10 zeigt die erreichten Rangzahlen.

Man kann zwei Beobachtungen machen: Der Algorithmus konvergiert mit größeren Elternzahlen schneller, und das ist unabhängig von der Zielfunktion. Falls der Anwender wenig Laufzeit für den Algorithmus zur Verfügung hat, sollte die Elternzahl 4 oder 5 verwendet werden. Falls mehr Rechenzeit zur Verfügung steht, sollte die Elternzahl 2 Verwendung finden.

C_{max}								
	100 Sekunden				500 Sekunden			
Eltern	2	3	4	5	2	3	4	5
R_1	9	5	15	12	15	9	9	7
R_2	10	13	8	8	13	9	10	8
R_3	11	11	11	7	5	12	13	10
R_4	10	11	6	13	7	10	8	15
L_{max}								
	100 Sekunden				500 Sekunden			
Eltern	2	3	4	5	2	3	4	5
R_1	9	10	11	10	11	16	7	7
R_2	9	8	10	13	7	11	14	7
R_3	11	9	12	9	8	8	12	12
R_4	11	13	7	8	14	5	7	14
$\sum C$								
	100 Sekunden				500 Sekunden			
Eltern	2	3	4	5	2	3	4	5
R_1	11	11	6	12	11	16	7	6
R_2	12	14	9	5	9	10	15	6
R_3	10	7	16	7	11	8	12	9
R_4	7	8	9	16	9	6	6	19
$\sum T$								
	100 Sekunden				500 Sekunden			
Eltern	2	3	4	5	2	3	4	5
R_1	4	14	10	13	12	14	7	9
R_2	12	11	5	11	14	16	6	3
R_3	9	8	13	10	8	7	13	11
R_4	15	7	12	6	6	3	14	17

Tabelle 2.10: *Rangzahlen der besten Lösungen für verschiedenen Elternanzahlen und 4 Zielfunktionen auf der Testmenge (genetische lokale Suche)*

2.2.4 Vergleich der Heuristiken

Nachdem die grundlegenden Parameter der genetischen lokalen Suche verglichen wurden, werden in diesem Abschnitt die Lokale-Suche-Heuristiken im Inneren der GLS untersucht. Da die Nachbarschaft N_1 für alle Zielfunktionen die besten Ergebnisse erzielt hat, wird sie im Weiteren verwendet. Für die Rekombination werden zwei Elternlösungen aus dem Pool der vorhande-

nen Lösungen zufällig gemäß Gleichverteilung ausgewählt. Abweichungen zu dieser Parameterwahl werden in Ausnahmefällen angegeben.

Schnelle lokale Suche versus volle lokale Suche

Zum Vergleich der schnellen lokalen Suche (SLS) mit der vollen lokalen Suche (VLS) als Kern des Hybrid-Algorithmus wird die Testmenge aus Tabelle 2.5 herangezogen. Der Pool enthielt 25 Lösungen. Für die ersten 25 lokalen Optima wurden zufällig generierte Startlösungen verwendet. Danach wurden Mehrheitslösungen gebildet. Der Hybrid-Algorithmus wurde nach 300 Sekunden Laufzeit abgebrochen. Der letzte Lauf der lokalen Suche wurde bis zum Ende durchgeführt. Die beiden Lokale-Suche-Heuristiken wurden anhand des

	C_{max}	$\sum C$	$\sum T$	L_{max}
beste Lösung	39	39	39	39
Durchschnitt	40	40	30	38

Tabelle 2.11: *Vergleich von schneller und voller lokaler Suche, Anzahl der besseren Zielfunktionswerte bei schneller lokaler Suche*

jeweils besten Zielfunktionswertes im Pool und dem Durchschnitt der Zielfunktionswerte im Pool verglichen. Die Daten in Tabelle 2.11 zeigen, dass die SLS bedeutend bessere Ergebnisse erzielt als die VLS. Bei allen Zielfunktionen liefert die SLS in jeweils 39 der 40 Instanzen die beste Lösung. Die Instanzen bei denen die VLS den besseren Zielfunktionswert erreicht, sind bei allen Zielfunktionen verschieden.

Dieses Ergebnis ist nicht überraschend. Um eine Verbesserungsschritt durchzuführen, muss die VLS die gesamte Nachbarschaft durchsuchen. Da die Nachbarschaften sehr groß sind und die Berechnung einer Zielfunktion aufwändig ist, ist die VLS bei realistischen Zeitschranken unterlegen. Da die hier verwendete Nachbarschaft N_1 die kleinste Anzahl zulässiger Nachbarn liefert, ist bei Verwendung der anderen Nachbarschaften keine Änderung der Beobachtung zu erwarten. Das Ergebnis ist der Grund, warum die SLS beim Vergleich der Nachbarschaften und der Elternzahlen genutzt wurde.

Vergleich der Schwellwertalgorithmen und der Tabu-Suche

Die drei Schwellwertalgorithmen Sidestep-Algorithmus, Threshold-Accepting und Simulated Annealing sowie die Tabu-Suche wurden als Lokale-Suche-Heuristiken im Kern der genetischen lokalen Suche verwendet. Die Parameter der GLS waren bei allen 4 Algorithmen identisch. Der Pool enthielt 15 Lösungen. Die ersten 15 Lösungen wurden mit der jeweiligen Lokale-Suche-Heuristik, beginnend mit zufälligen Startlösungen, erzeugt. Danach wurden Mittelwertlösungen als Startlösungen verwendet. Die Mittelwertlösungen hatten 2 Elternlösungen, die zufällig gemäß Gleichverteilung aus dem Pool gezogen wurden. Die genetische lokale Suche wurden nach 300 Sekunden abgebrochen. Der letzte Lauf der lokalen Suche wurde bis zum Ende ausgeführt. Nachbarschaft N_1 fand in allen Heuristiken Verwendung. Als Testmenge dienten wiederum die Instanzen aus Tabelle 2.5.

Bei den Lokale-Suche-Heuristiken wurden folgende Parameter eingesetzt:

- Sidestep-Algorithmus: Anzahl der Sidesteps $IS := 4000$;
- Threshold-Accepting: Faktor für den Schwellwert $T := 1.01$ und Abnahmerate $\lambda := 0.07$;
- Simulated Annealing: Starttemperatur $T_0 := 1.0$, Abkühlungsfaktor $\alpha := 0.985$, Anzahl der Iterationen zwischen zwei Abkühlungsschritten $IA := 200$ und Endtemperatur $T_{stop} := 0.001$;
- Tabu-Suche: Anzahl der Iterationen $IT := 10$, Länge der Nachbartabuliste $L_N := 100$ und Länge der Lösungstabuliste $L_L := 40$.

Tabelle 2.12 zeigt die Rangzahlen R_j, $j = 1, \ldots, 4$, die durch den Vergleich der besten Lösungen im Pool ermittelt wurden. Die letzte Zeile Δ gibt die durchschnittliche relative Abweichung zur besten Lösung über die 40 Instanzen an. Die Zielfunktion $\sum T$ wurde ausgeklammert, da bei manchen Instanzen der Zielfunktionswert 0 angenommen wurde.

Die erste Beobachtung ist das schlechte Abschneiden der GLS mit Tabu-Suche. Dieser Algorithmus liefert bei allen 4 Zielfunktionen und allen Instanzen die schlechtesten Lösungen. Die Ursache ist wie bei der vollen lokalen Suche die große Anzahl zulässiger Nachbarn. Die Konvergenz der GLS mit Tabu-Suche ist gegenüber der GLS mit Schwellwertalgorithmen zu langsam.

	C_{max}				L_{max}			
	Side	TA	SA	Tabu	Side	TA	SA	Tabu
R_1	35	1	4	0	39	0	2	0
R_2	3	2	35	0	1	6	32	0
R_3	2	37	1	0	0	34	6	0
R_4	0	0	0	40	0	0	0	40
Δ	9.7	14.2	7.9	62.9	28.4	395.0	200.0	2310.0
	$\sum C$				$\sum T$			
	Side	TA	SA	Tabu	Side	TA	SA	Tabu
R_1	13	27	0	0	18	23	2	0
R_2	26	11	3	0	22	13	3	0
R_3	1	2	37	0	0	4	35	0
R_4	0	0	0	40	0	0	0	40
Δ	25.7	4.6	32.8	64.5	-	-	-	-

Tabelle 2.12: *Rangzahlen der besten Lösungen der 4 Heuristiken und für 4 Zielfunktionen auf der Testmenge*

Bei den Schwellwertalgorithmen zeigt sich ein differenzierteres Bild. Der Sidestep-Algorithmus liefert bei den Zielfunktionen vom min-max-Typ die besten Ergebnisse. Bei den Zielfunktionen vom Summentyp erreicht der Threshold-Accepting-Algorithmus die besseren Ergebnisse. Das ist insofern überraschend, als dass der Sidestep-Algorithmus der einfachste Algorithmus ist. Er hat keine Möglichkeit, in einen Nachbarn mit schlechterem Zielfunktionswert zu wechseln. Die einzige Möglichkeit, aus lokalen Optima zu entkommen, ist der Wechsel in gleich gute Nachbarlösungen.

Hier stellt sich die Frage, wie groß der Anteil der Nachbarlösungen mit gleichem Zielfunktionswert an der Menge aller zulässigen Nachbarlösungen ist. Um diese zu ermitteln, wurde die Benchmarkmenge aus Tabelle A.1 im Anhang benutzt. Zu jeder Instanz wurden 10 Lösungen ermittelt. Für jede Lösung wurde die Anzahl der zulässigen Nachbarn und die Anzahl der zulässigen Nachbarn mit gleichem Zielfunktionswert bestimmt. Mit diesen Werten wurde der Quotient aus der Anzahl gleich guter Nachbarlösungen und aller Nachbarn gebildet.

Die Diagramme in Abbildung 2.2 zeigen die durchschnittlichen Quotienten der Nachbarn mit gleichem Zielfunktionswert und aller Nachbarn über die 10 Lösungen zu einer Instanz und der 10 Instanzen pro Gruppe. Für jede Gruppe der Instanzen sind die Quotienten für die 4 Zielfunktionen C_{max},

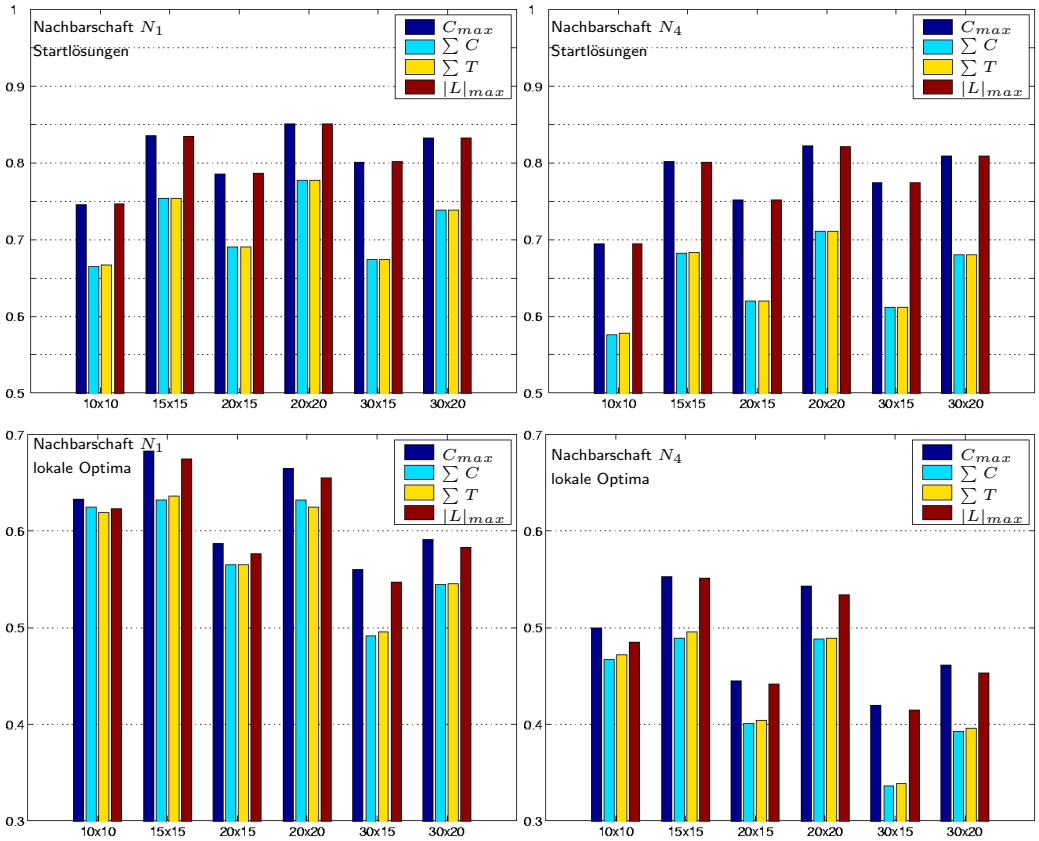

Abbildung 2.2: *Anteil von Nachbarlösungen mit gleichem Zielfunktionswert an allen zulässigen Nachbarlösungen*

$\sum C$, $\sum T$ und $|L|_{max}$ angegeben. Die Diagramme auf der linken Seite zeigen die Werte für die Nachbarschaft N_1 und auf der rechten Seite die Werte für N_4. In den oberen Diagrammen wurden zufällig erzeugte Startlösungen und in den unteren Diagrammen lokale Optima verwendet. Die lokalen Optima wurden mit dem Sidestep-Algorithmus mit anschließender schneller lokaler Suche erzeugt. Dabei wurde die jeweilige Nachbarschaft N_1 und N_4 eingesetzt. Beim Sidestep-Algorithmus waren 1000 Sidestep-Schritte zulässig. Die Ergebnisse für praktische Job-Shop-Probleme sind vergleichbar. Die Daten für die Instanzen der $PJSP$ sind in Tabelle A.2 im Anhang verzeichnet.

Bei allen Benchmarkgruppen haben über ein Drittel der Nachbarn einer Lösung einen gleichen oder besseren Zielfunktionswert. Bei Startlösungen liegt der Prozentsatz sogar über 50 % unabhängig von der Nachbarschaft und der Zielfunktion. Durch diese große Anzahl von gleich guten Nachbarn hat der

Sidestep-Algorithmus in der lokalen Suche genügend Möglichkeiten, schwachen lokalen Optima zu entkommen. Beim Threshold-Accepting Algorithmus und dem Simulated Annealing wird die Wahrscheinlichkeit, einen schlechteren Nachbarn zu akzeptieren, mit wachsender Laufzeit immer geringer. Falls der Schwellwert S im Threshold-Accepting Algorithmus den aktuellen Zielfunktionswert erreicht, hat der Algorithmus noch während eines Nachbarschaftsdurchlaufs die Möglichkeit in gleich gute Nachbarn zu wechseln. Das Simulated Annealing verhält sich ähnlich. Auch bei einer sehr kleinen aktuellen Temperatur wird noch mit Wahrscheinlichkeit Eins in Nachbarn mit identischen Zielfunktionswert gewechselt. Die Möglichkeit, in Nachbarlösungen mit größerem Zielfunktionswert zu wechseln, haben beide Algorithmen nur zu Beginn der lokalen Suche. Bei zufälligen Startlösungen ist dies kein Nachteil für den Sidestep-Algorithmus, da hier leicht Verbesserungen gefunden werden können.

2.2.5 Obere Schranken

Der Vergleich der hier entwickelten Algorithmen mit aus der Literatur bekannten Heuristiken wirft Probleme auf. Die erreichbaren Ergebnisse hängen im starken Maße von der Implementation der Algorithmen und dem Tuning der Algorithmenparameter ab. Um trotzdem Testaussagen über die Qualtität der durch die hier vorgestellten Algorithmen erreichten Lösungen treffen zu können, werden die Lösungen mit den Zielfunktionswerten der besten bekannten Lösungen verglichen.

Gute obere Schranken für Job-Shop Scheduling-Instanzen sind nur für die Zielfunktionen C_{max} und L_{max} verfügbar. Die nächsten Abschnitte konzentrieren sich aus diesem Grund auf diese zwei Zielfunktionen.

Das klassische Job-Shop-Problem $J||C_{max}$

Das Job-Shop-Problem mit der Zielfunktion C_{max} ist das am besten untersuchte Problem. Hierfür existiert der umfangreichste Satz von Probleminstanzen. Für dieses Problem steht nicht nur eine umfangreiche Anzahl von Heuristiken zur Verfügung [1, 4, 29, 30, 32, 34, 43, 44, 46, 54, 58, 60, 62], sondern auch Branch & Bound-Algorithmen [8, 9, 10, 38] und untere Schranken [11, 12]. Es wurde nicht nur viel Forschungsarbeit in das Problem investiert, sondern auch viel Rechenzeit in die Benchmark-Instanzen. 136 der

241 Benchmark-Instanzen wurden bisher gelöst. Für die restlichen Instanzen stehen sehr gute obere Schranken zu Verfügung.

Die genetische lokale Suche soll im multikriteriellen Fall in einer interaktiven Umgebung Verwendung finden. Aus diesem Grund ist die Frage zu beantworten: Welche Abweichung zu der besten bekannten oberen Schranke ist nach einer festen Laufzeit des Algorithmus zu erwarten? In dem folgenden Test wurden die erreichten Zielfunktionswerte bis zu einer Laufzeit von maximal 500 Sekunden ausgewertet.

$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	PG	IS	L_N	L_L	IT
10	5	20	2500	8	23	1500
15	5	20	2500	8	23	1500
20	5	20	2500	8	23	1500
10	10	20	2000	8	23	1500
15	10	20	2000	8	13	1000
20	10	20	2000	8	17	1000
15	15	20	1500	8	23	1500
20	15	15	1500	8	23	1500
30	10	15	2500	8	19	500
20	20	15	2000	8	19	1000
30	15	15	2000	8	17	1000
50	10	10	2000	8	11	200
30	20	10	2500	8	11	200
40	15	10	3000	8	11	200
50	15	5	2000	8	11	100
40	20	7	2000	8	11	100
50	20	5	2000	8	11	100
100	20	5	2000	8	7	75

Tabelle 2.13: *Startparameter für GLS mit Sidesteps und GLS mit Tabu-Suche bei verschiedenen Dimensionen der Probleminstanzen*

Für den Vergleich mit den besten bekannten oberen Schranken wurden zwei Lokale-Suche-Heuristiken in der genetischen lokalen Suche verwendet. Die erste Heuristik ist der Sidestep-Algorithmus 2.12. Dieser hatte sich in den oben angegebenen Versuchen als der beste Algorithmus für die Zielfunktion C_{max} herausgestellt. $N_1(x)$ war die verwendete Nachbarschaft. Der zweite Algorithmus ist ein für diese Zielfunktion speziell angepasster Tabu-Suche-Algorithmus. Da ein Grund für das schlechte Abschneiden der Tabu-Suche in der großen Anzahl zulässiger Nachbarn bei den Nachbarschaften $N_\alpha(x)$,

$\alpha = 1, \ldots, 5$, liegt, wurde eine andere Nachbarschaft verwendet. Als die beste Optimierungsheuristik für das Problem $J||C_{max}$ wird im Moment die Tabu-Suche von NOWICKI und SMUTNICKI [43, 44] angesehen. In der modifizierten Tabu-Suche wird aus diesem Grund die bei NOWICKI und SMUTNICKI angegebene Nachbarschaft verwendet. Diese Nachbarschaft wurde in Abschnitt 2.1.2 vorgestellt und wird im Weiteren mit $N_{NS}(x)$ bezeichnet. Ansonsten ist der modifizierte Tabu-Suche-Algorithmus identisch mit Algorithmus 2.15.

Die Startparameter der genetischen lokalen Suche waren abhängig von der Anzahl der Jobs und der Maschinen in den jeweiligen Instanzen und sind in Tabelle 2.13 zu finden. PG steht für die Poolgröße der GLS, IS bezeichnet die maximale Anzahl der Sidesteps. L_N und L_L sind die Längen der Nachbarschafts- und der Lösungstabuliste. IT ist die Bezeichnung für die maximale Anzahl von Iterationen in der Tabu-Suche.

Algorithmus 2.16 *Genetische lokale Suche*

1. *Für* $i := 1, \ldots, PG$
   (a) *Generiere Startlösung* y *mittels Algorithmus 2.2.*
   (b) *Finde mit Sidestep-Algorithmus oder Tabu-Suche beginnend mit* y *lokales Optimum* x *und füge* x *in* P *ein.*
2. *Für* $i := 1, \ldots, PG$
   (a) *Wähle zwei Lösungen* x_1 *und* x_2 *zufällig, gemäß Gleichverteilung aus* P*.*
   (b) *Bilde Nachkommen* y *aus den zwei Lösungen* x_1 *und* x_2*.*
   (c) *Finde mit Sidestep-Algorithmus oder Tabu-Suche beginnend mit* y *lokales Optimum* x*.*
   (d) *Falls in* P *keine Lösung mit einem gleichen kritischen Pfad existiert, füge* x *in* P *ein.*
   (e) *Falls* $|P| > PG$*, entferne die gemäß Zielfunktion schlechteste Lösung aus* P*.*
3. *Setze* $IS := 2 \cdot IS$ *(im Sidestep Fall).*
   Setze $L_N := L_N + 2$, $IT := 2 \cdot IT$ *und aktualisiere* L_L *(im Tabu-Suche Fall).*
4. *Falls Zeitschranke überschritten, STOPP, sonst gehe zu 2.*

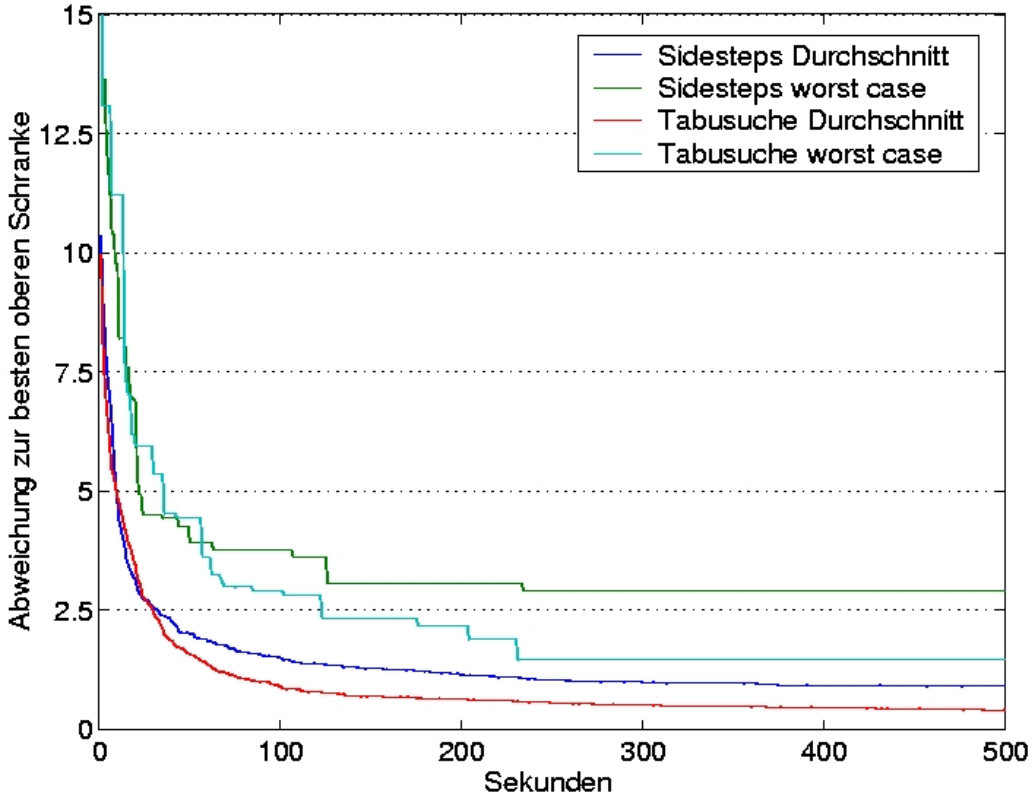

Abbildung 2.3: *Durchschnittliche und größte Abweichung zur besten bekannten oberen Schranke der Job-Shop-Benchmarks der Dimension 15×15 der genetischen lokalen Suche mit Sidestep-Algorithmus und Tabu-Suche*

Der Algorithmus 2.16 gibt die genaue Arbeitsweise der genetischen lokalen Suche an. Zu Beginn wird der Lösungspool mit lokalen Optima gefüllt, bei denen die lokale Suche mit zufällig erzeugten Lösungen startet. Danach wurde der Hybrid-Algorithmus mit Rekombinationsläufen fortgesetzt. Zur Mittelwertbildungen werden jeweils zwei Lösungen aus dem Pool verwendet. Nach PG Rekombinationsläufen wurden die Parameter des Sidestep-Algorithmus oder der Tabu-Suche aktualisiert. Dem Parameter L_L wurde bei der Aktualisierung die nächstgrößere Primzahl zugeordnet.

Die 241 Job-Shop-Benchmark-Instanzen wurden nach ihren Job- und Maschinenzahlen zu Gruppen zusammengefasst. Für jede Instanz wurden 3 Läufe der genetischen lokalen Suche mit dem Sidestep-Algorithmus und mit der Tabu-Suche durchgeführt. Zu jedem Zeitpunkt $t \in \{1, \ldots, 500\}$ wurde die

durchschnittliche und die größte relative Abweichung zur besten bekannten oberen Schranke über alle Instanzen einer Gruppe und alle Läufe ermittelt. So sind in den 241 Instanzen 24 Instanzen mit 20 Jobs und 20 Maschinen enthalten. Die sind die Instanzen YN1-4, TA21-30, DMU6-10 und DMU46-50. Auf diesen Instanzen wurden 72 Läufe der genetischen lokalen Suche mit Sidestep-Algorithmus und 72 mit Tabu-Suche mit jeweils 500 Sekunden Laufzeit durchgeführt. Dabei wurden die Werte für die durchschnittliche relative Abweichung in Prozent über alle 72 Läufe zu jedem Zeitpunkt ermittelt. Die größte relative Abweichung gibt die maximale Abweichung zur oberen Schranke zu jedem Zeitpunkt in allen 72 Läufen an. Das Liniendiagramm

			Sekunden							
$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	$\|\mathcal{T}\|$	5	10	30	60	100	200	300	500
10	5	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15	5	75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	5	100	0.11	0.07	0.00	0.00	0.00	0.00	0.00	0.00
10	10	100	2.29	1.50	0.84	0.62	0.52	0.47	0.43	0.39
15	10	150	2.27	1.62	0.75	0.64	0.52	0.39	0.34	0.25
20	10	200	7.17	5.10	3.12	2.33	1.93	1.61	1.53	1.34
15	15	225	7.21	4.79	2.53	1.84	1.50	1.14	0.98	0.90
20	15	300	13.07	11.68	6.76	4.82	3.79	3.10	2.77	2.40
30	10	300	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	20	400	14.19	12.43	8.36	5.63	4.30	3.28	2.82	2.41
30	15	450	16.80	15.25	11.96	8.10	6.35	4.12	3.24	2.34
50	10	500	18.88	13.37	8.41	4.47	2.85	1.78	1.62	1.27
30	20	600	25.27	22.22	18.95	13.96	10.14	6.56	5.43	4.45
40	15	600	29.39	24.51	15.94	9.99	6.98	3.42	3.06	2.51
50	15	750	24.55	21.50	12.53	6.28	3.45	2.43	1.98	1.33
40	20	800	33.38	29.88	21.46	17.28	11.95	7.50	6.07	4.57
50	20	1000	40.92	34.72	17.20	13.58	11.75	6.45	4.63	3.43
100	20	2000	45.98	38.25	28.05	17.65	8.19	2.57	1.34	0.44

Tabelle 2.14: *Durchschnittliche Abweichung zur besten oberen Schranke zu festen Zeitpunkten*

in Abbildung 2.3 zeigt die durchschnittliche und maximale Abweichung zur oberen Schranke bei Verwendung der genetischen lokalen Suche mit Sidestep-Algorithmus und Tabu-Suche unter Verwendung der 15 Instanzen mit 15 Jobs und 15 Maschinen. Die Liniendiagramme in Abbildung 2.4 zeigen die

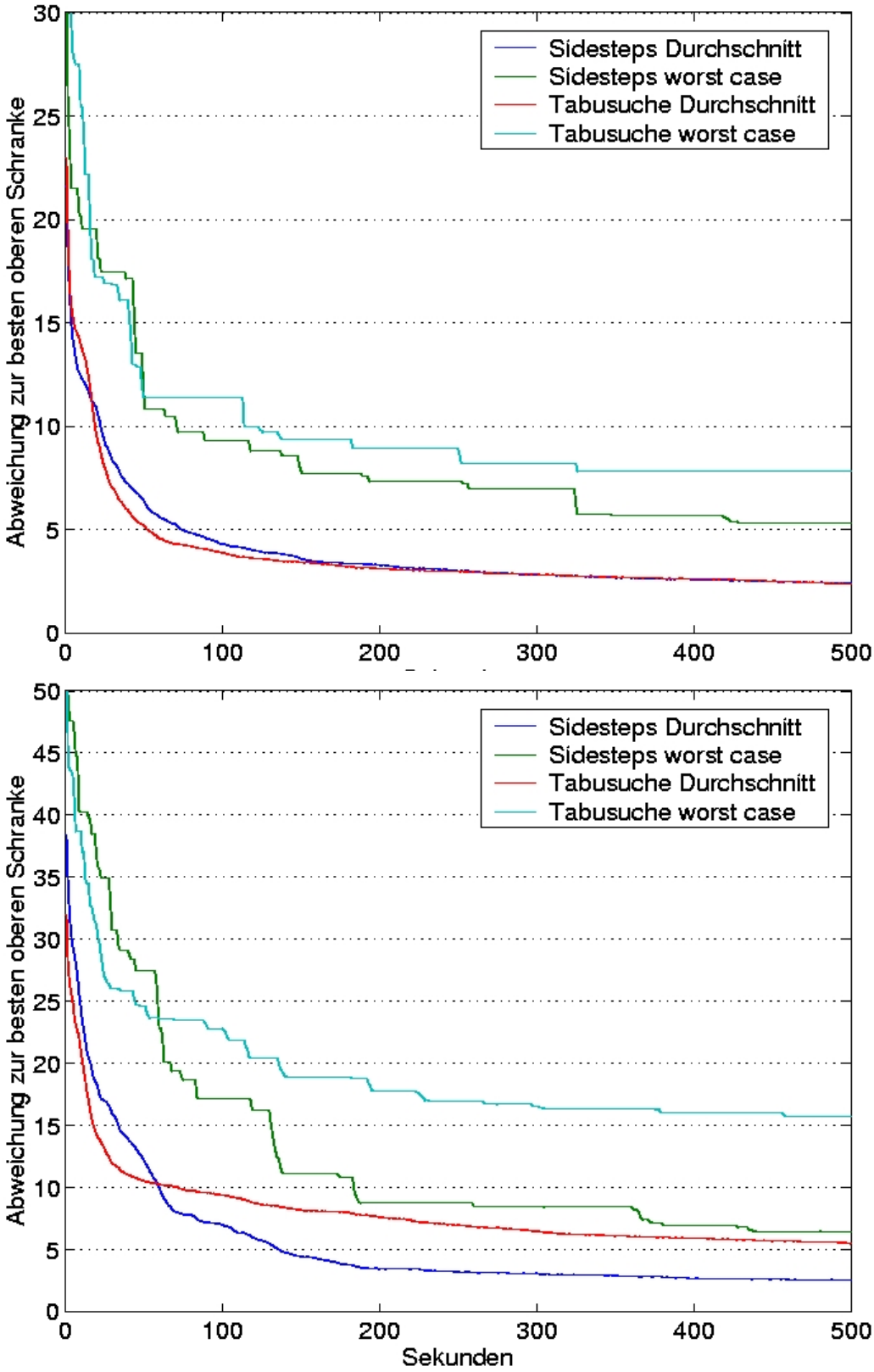

Abbildung 2.4: *Durchschnittliche und größte Abweichung zur besten bekannten oberen Schranke der Job-Shop-Benchmarks der Dimension 20×20 (oben) und 40×15 (unten) der genetischen lokalen Suche mit Sidestep-Algorithmus und Tabu-Suche*

Daten für die 24 Instanzen mit 20 Jobs und 20 Maschinen sowie die Daten für die 10 Instanzen mit 40 Jobs und 15 Maschinen. Aus Tabelle 2.14 ist zu entnehmen, dass die genetische lokale Suche mit Sidestep-Algorithmus nach 500 Sekunden bei allen Instanzen eine durchschnittliche relative Abweichung von weniger als 5 Prozent erreicht. Die maximale Abweichung liegt bei Instanzen mit bis zu 400 Vorgängen ebenfalls unter 5 Prozent. Die Instanzengruppen mit den Dimensionen 50×10 und 100×20 fallen ebenfalls in diese Kategorie.

Die Daten führen zu zwei Beobachtungen:

1. Der Hybrid-Algorithmus mit Tabu-Suche und der Zielfunktion C_{max} liefert bei Instanzen mit $\frac{|\mathscr{J}|}{|\mathscr{M}|} < 2$ bessere Ergebnisse als bei Instanzen mit einem Verhältnis $\frac{|\mathscr{J}|}{|\mathscr{M}|} \geq 2$.

2. Der Hybrid-Algorithmus mit dem Sidestep-Algorithmus liefert bessere Ergebnisse bei Instanzen mit dem Verhältnis $\frac{|\mathscr{J}|}{|\mathscr{M}|} \geq 2$. Besonders große Unterschiede traten bei den $J|2SETS|C_{max}$-Instanzen auf. Hier schnitt die GLS mit Sidestep-Algorithmus vor allem im „worst case“ der maximalen Abweichung bedeutend besser als die GLS mit Tabu-Suche ab.

3. Bei den Benchmarkgruppen, in denen $J|2SETS|C_{max}$-Instanzen vorhanden sind, wurden die Werte mit der größten Abweichung zur besten bekannten oberen Schranke immer von den $J|2SETS|C_{max}$-Instanzen angenommen. Diese Instanzen sind demnach mit der GLS schwieriger zu optimieren. Diese Instanzen haben ebenfalls die größten Differenzen zwischen Ein-Maschinen-Schranke und der jeweils besten bekannten oberen Schranke. Da die GLS aber gerade bei den $J|2SETS|C_{max}$-Instanzen die größten Verbesserungen erreicht hat (Tabelle 2.15), sind diese Instanzen allgemein schwieriger zu optimieren.

Von besonderem Interesse ist auch die Frage, ob die genetische lokale Suche in der Lage ist, bessere als die bisher bekannten oberen Schranken zu liefern. Tabelle 2.15 listet die 78 Benchmark-Instanzen auf, bei denen bessere obere Schranken gefunden wurden. Sowohl mit der GLS mit Sidestep-Algorithmus als auch mit der GLS mit Tabu-Suche wurden viele Läufe auf der gesamten Menge der Benchmark-Instanzen durchgeführt, um gute Bereiche für die Algorithmenparameter zu finden. Im Gegensatz zu Algorithmus 2.16 wurden in der genetischen lokalen Suche die Mittelwertlösungen nicht direkt als Startlösungen für die Optimierung verwendet. Die Lösungen wurden zwischen Schritt 2(b) und 2(c) mit Algorithmus 2.9 mutiert. Bei allen Läufen

Instanz	US-OS	neue OS	Δ
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=200×20×10			
SWV 4	1450-1483	1482	1
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=300×20×15			
SWV 7	1446-1620	1613	7
SWV 8	1641-1763	1759	4
SWV 9	1604-1663	1662	1
SWV10	1631-1767	1761	6
TA13	1282-1345	1342	3
TA15	1304-1342	1340	2
TA16	1302-1362	1360	2
DMU 1	2501-2579	2563	16
DMU 2	2651-2716	2706	10
DMU41	2839-3376	3312	64
DMU42	3066-3574	3416	158
DMU43	3121-3535	3455	80
DMU44	3112-3599	3501	98
DMU45	2930-3355	3273	82
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=400×20×20			
TA26	1539-1651	1647	4
DMU 6	2834-3269	3252	17
DMU 8	2901-3210	3199	11
DMU 9	2739-3126	3092	34
DMU10	2716-3001	2985	16
DMU46	3425-4192	4120	72
DMU47	3353-4060	3999	61
DMU48	3317-3918	3834	84
DMU49	3369-3810	3765	45
DMU50	3379-3866	3772	94
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=450×30×15			
TA32	1774-1798	1796	2
TA38	1673-1677	1673	4
DMU11	3395-3491	3481	10
DMU12	3481-3578	3540	38
DMU13	3681-3715	3699	16
DMU14	3394-3396	3394	2
DMU51	3839-4320	4264	56
DMU52	4012-4586	4401	185
DMU53	4108-4635	4478	157
DMU54	4165-4525	4465	60
DMU55	4099-4500	4402	98
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=500×50×10			
SWV11	2983-2991	2988	3
SWV15	2885-2904	2903	1

Instanz	US-OS	neue OS	Δ
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=600×30×20			
TA41	1859-2021	2018	3
TA44	1927-1992	1989	3
TA46	1940-2025	2022	3
TA48	1912-1962	1956	6
TA49	1915-1971	1968	3
DMU16	3726-3802	3787	15
DMU17	3697-3944	3876	68
DMU18	3844-3894	3852	42
DMU19	3650-3891	3824	67
DMU20	3604-3788	3746	42
DMU56	4366-5234	5163	71
DMU57	4182-4962	4781	181
DMU58	4214-5038	4892	146
DMU59	4199-5004	4864	140
DMU60	4259-5088	4890	198
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=600×40×15			
DMU61	4886-5448	5349	99
DMU62	5004-5482	5342	140
DMU63	5049-5597	5437	160
DMU64	5130-5580	5367	213
DMU65	5072-5398	5311	87
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=750×50×15			
DMU71	6050-6610	6373	237
DMU72	6223-6790	6647	143
DMU73	5935-6536	6345	191
DMU74	6015-6623	6376	247
DMU75	6010-6552	6384	168
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=800×40×20			
DMU26	4647-4712	4684	28
DMU27	4848-4883	4848	35
DMU29	4691-4700	4691	9
DMU30	4732-4779	4732	47
DMU66	5357-6131	5910	221
DMU67	5484-6275	6117	158
DMU68	5423-6152	5949	203
DMU69	5419-6093	5915	178
DMU70	5492-6368	6115	253
$\|\mathcal{T}\| \times \|\mathcal{J}\| \times \|\mathcal{M}\|$=1000×50×20			
DMU37	5851-5852	5851	1
DMU76	6329-7454	6975	479
DMU77	6399-7329	6969	360
DMU78	6508-7301	6962	339
DMU79	6593-7590	7164	426
DMU80	6435-7173	6824	349

Tabelle 2.15: *Verbesserte Benchmark-Instanzen für* $J||C_{max}$

wurden die erreichten Optima mitprotokolliert. Aus diesem Grund wurden für die erreichten oberen Schranken keine Zeitschranken angegeben.

Es wurden bessere obere Schranken für die Instanzen von STORER *et al.*, TAILLARD und DEMIRKOL *et al.* gefunden. Die vorher bekannten oberen Schranken wurden durch verschiedene Optimierungsheuristiken ermittelt. Insbesondere der Tabu-Suche-Algorithmus von NOWICKI und SMUTNICKI wurde auf alle 241 Instanzen angewendet [29, 43]. Auffällig ist, dass die Verbesserungen für $J|2SETS|C_{max}$-Instanzen (SWV4,7,8,9,10,11 und DMU41-80) durch die GLS mit Sidestep-Algorithmus erreicht wurden. Die Abweichung Δ zwischen der alten und der neuen oberen Schranke ist bei den Instanzen DMU41-80 am größten. Damit bestätigt sich die Beobachtung, dass die GLS mit Sidestep-Algorithmus bei $J|2SETS|C_{max}$-Instanzen bessere Ergebnisse in den Zielfunktionswerten liefert als mit der Tabu-Suche. Dies gilt nicht nur für die GLS mit Tabu-Suche, sondern ebenfalls im Vergleich mit der Tabu-Suche von NOWICKI und SMUTNICKI.

Von den 241 Benchmark-Instanzen wurde bei weiteren 140 Instanzen Lösungen gefunden, die die bekannten oberen Schranken erreichten. Bei den restlichen 23 Instanzen wurde die obere Schranke verfehlt. Die maximale Abweichung lag bei allen 23 Instanzen unter 0.38 %. Die oberen Schranken dieser Instanzen wurden entweder mit Branch & Bound-Algorithmen [8, 38] oder mit der Tabu-Suche von NOWICKI und SMUTNICKI erreicht.

Das klassische Job-Shop-Problem $J||L_{max}$

Aus der Benchmarkmenge von DEMIRKOL *et al.* zum $J||L_{max}$-Problem wurden für die Untersuchung 50 Instanzen ausgewählt. Dabei wurden Instanzen mittlerer Größe mit 300 und 400 Vorgängen sowie große Instanzen mit 1000 Vorgängen genutzt. Von den 50 Instanzen gehören 25 zum Problem $J||L_{max}$ und 25 zum Problem $J|2SETS|L_{max}$.

Die bisher besten bekannten oberen Schranken wurden von Demirkol *et al.* ermittelt [13]. Um diese zu berechnen, wurden 11 Prioritätsregeln und 3 verschiedene Versionen der Shifting-Bottleneck-Prozedur genutzt. Die hierfür aufgewendeten Zeiten lagen bei ≈160 Sekunden für alle 20×15 Instanzen, bei ≈400 Sekunden für alle 20×20 Instanzen, bei ≈5000 Sekunden für 50×20 Instanzen aus $J||L_{max}$ und bei ≈10000 Sekunden für 50×20 Instanzen aus $J|2SETS|L_{max}$. Die unteren Schranken wurden mit der Ein-Maschinen-Relaxation ermittelt. Zur Optimierung dieser Instanzen wurde Algorithmus 2.16

Instanz	US-OS	neue OS	Δ
r_50_20_1_1_5	1591-2181	1785	396
r_50_20_1_1_3	1746-2390	1970	420
r_50_20_1_1_2	1794-2355	2045	310
r_50_20_1_1_6	1845-2219	1918	301
r_50_20_1_1_4	1786-2142	1822	320
cr_50_20_1_1_1	2140-3471	3146	325
cr_50_20_1_1_7	2424-3721	3340	381
cr_50_20_1_1_5	2482-3574	3366	208
cr_50_20_1_1_3	2802-4011	3895	116
cr_50_20_1_1_10	2417-3448	3331	117

Tabelle 2.16: *Verbesserte Benchmark-Instanzen für* $J||L_{max}$

mit dem Sidestep-Algorithmus eingesetzt. Im Pool wurden für die Instanzen mit 300 und 400 Vorgängen 50 Lösungen eingefügt $PG = 50$, der Startwert für Anzahl der Sidesteps wurde auf $IS = 1000$ gesetzt, und die Zeitschranke betrug 500 Sekunden. Für die großen Instanzen galten die Parameter $PG = 30$ und $IS = 3000$. Die Zeitschranken wurden analog zu den Vorgaben von Demirkol auf 5000 Sekunden für die fünf $J||L_{max}$ Instanzen und auf 10000 Sekunden für die fünf $J|2SETS|L_{max}$ Instanzen gesetzt.

Mit diesen Parametern und Zeitschranken wurden bei allen 50 Instanzen die oberen Schranken zwischen 2.49% und 18.16% verbessert. Tabellen 2.16 und 2.17 listen in der ersten Spalte die Bezeichnungen der Instanzen auf. Die zweite Spalte enthält die untere Schranke und die bisher beste bekannte obere Schranke. Die Spalte drei listet die durch die GLS mit Sidestep-Algorithmus erzielten oberen Schranken auf; und die vierte Spalte enthält die Differenz.

2.3 Landschaften im Lösungsraum und das „Big Valley“

Der Begriff **Landschaft** bezeichnet die Verteilung der Zielfunktionswerte in Lösungsraum. Die Untersuchung der Landschaft hat zum Ziel, Strukturen im Lösungsraum zu erkennen. Die Informationen über Strukturen oder Regularitäten im Suchraum können Erklärungen für das Funktionieren von Optimierungsheuristiken liefern. Sie können ebenfalls genutzt werden, um die

Instanz	US-OS	neue OS	Δ
r_20_15_1_1_6	1027-1448	1245	203
r_20_15_1_1_8	1127-1552	1444	108
r_20_15_1_1_4	1160-1492	1351	141
r_20_15_1_1_2	1140-1464	1244	220
r_20_15_1_1_3	1182-1501	1294	207
r_20_15_2_1_7	1575-1957	1892	65
r_20_15_2_1_3	1727-2100	1941	159
r_20_15_2_1_1	1785-2165	1973	192
r_20_15_2_1_5	1521-1839	1718	121
r_20_15_2_1_9	1858-2143	1942	201
r_20_20_1_1_7	1391-2013	1893	120
r_20_20_1_1_10	1182-1708	1560	148
r_20_20_1_1_6	1366-1962	1748	214
r_20_20_1_1_3	1569-2248	1993	255
r_20_20_1_1_4	1226-1753	1531	222
r_20_20_2_1_2	1776-2638	2404	234
r_20_20_2_1_6	1868-2647	2242	405
r_20_20_2_1_4	1845-2535	2373	162
r_20_20_2_1_8	1927-2627	2406	221
r_20_20_2_1_7	1947-2640	2392	248
cr_20_15_1_1_8	1434-2159	1932	227
cr_20_15_1_1_3	1619-2293	2139	154
cr_20_15_1_1_1	1558-2126	2005	121
cr_20_15_1_1_5	1522-2049	1998	51
cr_20_15_1_1_9	1693-2224	2162	62
cr_20_15_2_1_6	2075-2761	2574	187
cr_20_15_2_1_7	2397-3098	2871	227
cr_20_15_2_1_3	2033-2612	2429	183
cr_20_15_2_1_4	2298-2914	2742	172
cr_20_15_2_1_10	1989-2770	2641	104
cr_20_20_1_1_10	1759-2797	2572	225
cr_20_20_1_1_5	1895-2879	2771	108
cr_20_20_1_1_3	1911-2894	2703	191
cr_20_20_1_1_9	1770-2643	2582	61
cr_20_20_1_1_6	2088-2948	2858	90
cr_20_20_2_1_8	2758-3950	3662	288
cr_20_20_2_1_9	2504-3489	3126	363
cr_20_20_2_1_1	2550-3538	3267	271
cr_20_20_2_1_7	2426-3304	3110	194
cr_20_20_2_1_6	2431-3244	3120	124

Tabelle 2.17: *Verbesserte Benchmark-Instanzen für* $J||L_{max}$

Heuristiken zu verbessern.

Speziell wird die Struktur des **„Big Valley“** gesucht. Das Vorhandensein des „Big Valley“ bedeutet, dass die Abstände lokaler Minima im Lösungsraum und die Zielfunktionswerte dieser Lösungen signifikant positiv korreliert sind. Weiterhin ist die Mehrheit der sehr guten lokalen Optima in einem relativ kleinen Gebiet des Lösungsraumes konzentriert. Struktur, Regularität und „Big Valley“ sind sehr starke Begriffe und sollten nicht zu 2-dimensionalen oder 3-dimensionalen Vorstellungen führen. Auch im Lösungsraum nah beieinander liegende Lösungen können große Unterschiede im Zielfunktionswert aufweisen.

Die Existenz des „Big Valley“ wurde experimentell für das symmetrische TSP und für das Permutation-Flow-Shop-Problem mit der Zielfunktion C_{max} bestätigt [48]. Nowicki und Smutnicki [44] untersuchten die Struktur des „Big Valley“ für das Problem $J||C_{max}$. Sie verwendeten hierfür die Instanzen TA 1 bis TA50. In dem folgenden Abschnitt wird die Struktur des Lösungsraumes für praktische Job-Shop-Probleme mit den Zielfunktionen C_{max}, $\sum C$, $\sum T$ und $|L|_{max}$ untersucht.

2.3.1 Abstandsmaße im Lösungsraum

Die Landschaft im Lösungsraum wird durch die verwendete Nachbarschaft in den Heuristiken induziert. Die Distanz zwischen Lösungen würde in diesem Fall durch die minimale Anzahl von Nachbarschaftsschritten definiert, die nötig ist, um von einer Lösung zur anderen zu gelangen. Dazu muss der Nachbarschaftsgraph zusammenhängend und die Nachbarschaft symmetrisch sein. Die Ermittlung der Abstände ist hierbei kompliziert und algorithmisch sehr aufwändig. Aus diesem Grund wird ein anderes Konzept für Abstandsmaße verwendet.

Die hier entwickelten Algorithmen sind Heuristiken zur Reihenfolgeoptimierung der Vorgänge auf den Maschinen. Die Menge der Vorgänge $\{T_{ji}\}$, die auf einer Maschine M_i, $M_i \in \mathscr{M}$ bearbeitet werden müssen, ist für alle Lösungen einer Instanz gleich. Der Abstand $A(\pi, \sigma)$ zwischen zwei Lösungen π und σ kann aus diesem Grund wie folgt definiert werden:

$$A(\pi, \sigma) = \sum_{i=1}^{m} A_i(\pi_i, \sigma_i).$$

π_i und σ_i sind hierbei Permutationen der Tasks aus $\{T_{ji}\}$, die auf der Maschi-

Abstand	Berechnung	Zeitkomplexität
$D(\alpha,\beta)$	$\sum \lvert\alpha(i)-\beta(i)\rvert$	$O(n)$
$S^2(\alpha,\beta)$	$\sum\{\alpha(i)-\beta(i)\}^2$	$O(n)$
$I(\alpha,\beta)$	Anzahl der Fehlstellungen in $\alpha^{-1}\beta$	$O(n^2)$
$T(\alpha,\beta)$	n − Anzahl der Zyklen in $\alpha^{-1}\beta$	$O(n)$
$L(\alpha,\beta)$	n − Länge der längsten gemeinsamen Unterfolge in α^{-1} und β^{-1}	$O(n\log n)$

Tabelle 2.18: *Berechnungsvorschriften und Zeitkomplexität der Abstandsmaße*

ne M_i, $M_i \in \mathscr{M}$ bearbeitet werden müssen. Weiterhin gilt durch die Definition von Abstandsmaßen das Folgende. Sei f eine Abbildung der Menge $\{T_{ji}\}$ auf die Menge $\{1,\ldots,n\}$, $n = |\{T_{ji}\}|$, dann gilt $A_i(\pi_i,\sigma_i) = A_i(\alpha,\beta)$, wobei $\alpha = f\circ\pi_i$ und $\beta = f\circ\sigma_i$ Permutationen auf der Menge $\{1,\ldots,n\}$ sind. Wir können uns somit auf die Untersuchung von Abstandsmaßen A, die auf der Menge $\{1,\ldots,n\}$ definiert sind, beschränken. Aus Vereinfachungsgründen wird im Weiteren der Index i weggelassen.

Abstandsmaße auf Permutationen sind aus der Literatur bekannt. Eine Übersicht findet man in [15]. In dieser Arbeit werden die folgenden fünf Distanzmaße verwendet.

$$D(\alpha,\beta) = \sum|\alpha(i)-\beta(i)| \text{ (Footrule)}$$

$$S^2(\alpha,\beta) = \sum\{\alpha(i)-\beta(i)\}^2 \text{ (Spearmans Rangkorrelation)}$$

$$I(\alpha,\beta) = \text{minimale Anzahl von paarweise benachbarten Transpositionen, um } \alpha^{-1} \text{ in } \beta^{-1} \text{ zu überführen (Kendalls tau)}$$

$$T(\alpha,\beta) = \text{minimale Anzahl von Transpositionen, um } \alpha \text{ in } \beta \text{ zu überführen (Cayley-Abstand)}$$

$$L(\alpha,\beta) = n - \text{Länge der längsten wachsenden Unterfolge in } \beta\alpha^{-1} \text{ (Ulams Abstandsmaß)}$$

In der Tabelle 2.18 sind die Berechnungsvorschriften und die Zeitkomplexität der Berechnung der Abstandsmaße verzeichnet. $I(\alpha,\beta)$ und $T(\alpha,\beta)$ haben eine eindeutige Beziehung zu den Nachbarschaften N_k, $k \in \{1,4,5\}$, die durch Vertauschungen der Vorgänge in der aufsteigenden Nummerierung definiert sind. Das Abstandsmaß von Ulam hat eine Beziehung zu den Nachbarschaften

N_k, $k \in \{2,3,4,5\}$. Es kann gezeigt werden, dass $L(\alpha^{-1},\beta^{-1}) = n - \Delta(\alpha,\beta)$ gilt [44]. $\Delta(\alpha,\beta)$ ist die Länge der längsten gemeinsamen Unterfolge in α und β. $L(\alpha,\beta)$ kann effizient in $O(n \log n)$ mittels Floyd's Game [15] berechnet werden. Die Abstandsmaße $D(\alpha,\beta)$ und $S^2(\alpha,\beta)$ wurden verwendet, um die Abhängigkeit der im Weiteren untersuchten Korrelationen von dem Distanzmaß zu untersuchen. Diese beiden Maße haben die gleiche Zeitkomplexität wie der Cayley-Abstand. Der Vorteil liegt in der kleineren Konstante vor dem n. Außerdem ist die Distanz $I(\alpha,\beta)$ nah bei $D(\alpha,\beta)$ in dem Sinne, dass $I \leq D \leq 2I$ gilt.

2.3.2 Korrelation der Distanzen im Lösungs- und Zielfunktionsraum

Die Existenz eines „Big Valley“ wurde experimentell mit einer Teilmenge der oben vorgestellten Benchmark-Instanzen untersucht. Für jede Instanz wurden $c = 100$ lokale Optima erzeugt. Die Instanzen wurden ausgehend von zufällig erzeugten Startlösungen mit dem Sidestep-Algorithmus optimiert. Die erhaltene Lösung muss kein lokales Optimum sein. Dies wurde mit der schnellen lokalen Suche überprüft. Die Lösung mit dem besten Zielfunktionswert wurde als Referenzlösung fixiert. O.B.d.A. wurde dieser Lösung der Index 1 zugeordnet. Mit den lokalen Optima wurden drei experimentelle Merkmale berechnet:

(1) der Abstand zur Referenzlösung $X_1(i) = A(\pi^i,\pi^1)$, $i = 2,\ldots,100$, $A \in \{D, S^2, I, T, L\}$

(2) der Durchschnitt der Abstände zu den anderen lokalen Optima $X_2(i) = \frac{1}{c-1}\sum_{j=1;j\neq i}^{c} A(\pi^i,\pi^j)$, $i = 1,\ldots,100$, $A \in \{D, S^2, I, T, L\}$

(3) die Differenz zum Referenzfunktionswert $Y(i) = f(\pi^i) - f(\pi^1)$ $i = 1,\ldots,100$.

Für die Kombinationen $X_1(i), Y(i)$, $i = 2,\ldots,100$, und $X_2(i), Y(i)$, $i = 1,\ldots,100$, wurden die Pearsonschen Korrelationskoeffizienten ρ^P_{max} und ρ^P_{av} berechnet [25]. Für diese Korrelationskoeffizienten wurde die Signifikanz getestet. Da die X und Y keine unabhängigen zufälligen Stichproben sind (falls die Zielfunktionswerte der lokalen Optima A und B und die Zielfunktionswerte der lokalen Optima B und C nah beieinander liegen, so ist A auch in der Nähe von C), können keine Standardtests verwendet werden. Aus diesem

Grund wurde ein Randomisierungstest verwendet [37]. Dabei werden die Indizes einer Distanzmatrix permutiert und der Korrelationskoeffizient neu berechnet. Diese Prozedur wird viele Male wiederholt (10000 Wiederholungen). Die Anzahl der Wiederholungen, bei denen ein extremerer Korrelationskoeffizient auftritt als der ursprünglich berechnete Wert, kann als geschätzter Signifikanzlevel aufgefasst werden. Da nicht automatisch von einer Normalverteilung der Merkmale ausgegangen werden kann, wurde auch der Rangkorrelationskoeffizient von Spearman ρ^S_{max} und ρ^S_{av} berechnet. Tabelle 2.19 enthält die Resultate für die Zielfunktion C_{max} und die Distanzmaße $D(\alpha, \beta)$ und $T(\alpha, \beta)$. Zu jeder der 11 Gruppen zu je 10 Instanzen wurde der durchschnittliche Wert für die Korrelationskoeffizienten von Pearson und Spearman angegeben. Die Werte k geben für die Korrelationskoeffizienten ρ^P_{max} und ρ^P_{av} die Anzahl der Instanzen an, bei denen die Korrelationskoeffizienten statistisch signifikant zu einem Level von 0.1% sind. Die letzte Zeile enthält die Werte über alle 110 Instanzen.

Aus der zweiten und sechsten Spalte der Tabelle 2.19 folgt, dass eine starke Korrelation zwischen dem Abstand zur besten Lösung im Pool und dem Wert der Zielfunktion C_{max} bei beiden Korrelationskoeffizienten existiert. Die Korrelationskoeffizienten sind bei 96 ($D(\alpha, \beta)$) beziehungsweise 95 ($T(\alpha, \beta)$) der 110 getesteten Instanzen statistisch signifikant zu einem Level von 0.1%. Dies bestätigt die Vermutung der Existenz eines Big Valley bei dem Problem $J||C_{max}$. Ähnliche Beobachtungen können für die Werte der Korrelationskoeffizienten ρ^P_{av} und ρ^S_{av} sowie deren Signifikanz gemacht werden. Aus diesen Korrelationskoeffizienten folgt, dass Lösungen mit besseren Zielfunktionswerten näher am Zentrum des Clusters der Lösungen im Lösungsraum liegen. Tabellen 2.20 und 2.21 enthalten die Werte der Korrelationskoeffizienten und die Anzahl der zu einem Level von 0.1% signifikanten Koeffizienten für die Zielfunkionen C_{max} und $|L|_{max}$ sowie für alle fünf Abstandsmaße. Aus diesen Werten folgt die Vermutung, dass bei dem Problem $J||\,|L|_{max}$ ebenfalls eine Big-Valley-Struktur im Lösungsraum existiert. Die Wahl der Abstandsmaße hat bei keiner der beiden Zielfunktionen Einfluss auf die Folgerungen. Die Korrelationskoeffizienten wurden ebenfalls für die Benchmark-Instanzen der praktischen Job-Shop-Probleme ermittelt. Die dazugehörigen Daten bezüglich der Zielfunktion C_{max} befinden sich in Tabelle A.3 im Anhang. Die Werte der Korrelationskoeffizienten und die Anzahl der signifikanten Koeffizienten bestätigen die Vermutung auch für praktische Job-Shop-Probleme.

Um die Vermutung für Zielfunktionen vom Summentyp zu überprüfen, wurden die Korrelationskoeffizienten für die Zielfunktionen $\sum C$ und $\sum T$ ermittelt. Dazu wurden die ersten vier Gruppen der Benchmark-Instanzen aus

C_{max}	$D(\alpha,\beta)$					
	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
$\lvert\mathcal{J}\rvert/\lvert\mathcal{M}\rvert$	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
10/10	0.4207	0.4685	9	10	0.4194	0.4588
15/15	0.4913	0.6407	10	10	0.4633	0.5735
20/15	0.3876	0.5817	8	10	0.4006	0.5161
20/20	0.4924	0.6302	9	9	0.4827	0.6159
30/15	0.5531	0.6697	8	10	0.5488	0.6179
30/20	0.5705	0.6348	9	8	0.5828	0.6323
40/15	0.3927	0.3387	6	5	0.3795	0.2966
40/20	0.5064	0.5371	10	9	0.5116	0.5209
50/10	0.5475	0.5884	9	9	0.5448	0.5545
50/15	0.6243	0.7162	10	10	0.6549	0.7220
50/20	0.5679	0.6386	8	9	0.5756	0.6439
alle	0.5049	0.5858	96	99	0.5058	0.5593
	$T(\alpha,\beta)$					
$\lvert\mathcal{J}\rvert/\lvert\mathcal{M}\rvert$	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
10/10	0.3736	0.4579	8	9	0.3725	0.4513
15/15	0.4197	0.6069	8	10	0.4061	0.5421
20/15	0.3824	0.7007	9	10	0.3955	0.6608
20/20	0.4594	0.6667	9	10	0.4573	0.6500
30/15	0.5182	0.7892	8	10	0.5217	0.7219
30/20	0.6289	0.8211	10	10	0.6391	0.8241
40/15	0.3548	0.5884	7	9	0.3602	0.5519
40/20	0.6029	0.8279	10	10	0.6025	0.8285
50/10	0.3190	0.7571	7	10	0.3276	0.7169
50/15	0.6011	0.8704	10	10	0.6216	0.8682
50/20	0.5923	0.8657	9	10	0.6125	0.8695
alle	0.4774	0.7229	95	108	0.4833	0.6986

Tabelle 2.19: *Korrelationskoeffizienten und Signifikanzen bei klassischen Job-Shop-Problemen mit der Zielfunktion C_{max} und den Distanzmaßen $D(\alpha,\beta)$ und $T(\alpha,\beta)$*

Tabelle A.1 im Anhang verwendet. Es zeigt sich in den Tabellen 2.22 und 2.23, dass auch bei diesen Zielfunktionen die Struktur des Big Valley existiert. Im Vergleich mit den Zielfunktionen vom min-max-Typ ist sowohl der Durchschnitt der Korrelationskoeffizienten als auch die Anzahl der signifikan-

C_{max}	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Distanz	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
$D(\alpha,\beta)$	0.5049	0.5858	96	99	0.5058	0.5593
$S^2(\alpha,\beta)$	0.4794	0.5370	89	85	0.4780	0.5094
$I(\alpha,\beta)$	0.5004	0.5732	94	96	0.5009	0.5459
$T(\alpha,\beta)$	0.4774	0.7229	95	108	0.4833	0.6986
$L(\alpha,\beta)$	0.4100	0.6706	83	108	0.4116	0.6474

Tabelle 2.20: *Durchschnitt der Korrelationskoeffizienten und Signifikanzen bei klassischen Job-Shop-Problemen mit der Zielfunktion C_{max}*

$\|L\|_{max}$	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Distanz	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
$D(\alpha,\beta)$	0.5928	0.6395	103	102	0.5917	0.6000
$S^2(\alpha,\beta)$	0.5369	0.5568	92	89	0.5300	0.5168
$I(\alpha,\beta)$	0.5757	0.6092	100	99	0.5720	0.5691
$T(\alpha,\beta)$	0.6260	0.7962	110	110	0.6410	0.7637
$L(\alpha,\beta)$	0.5722	0.7342	106	110	0.5844	0.6937

Tabelle 2.21: *Durchschnitt der Korrelationskoeffizienten und Signifikanzen bei klassischen Job-Shop-Problemen mit der Zielfunktion $|L|_{max}$*

ten Koeffizienten bei den Zielfunktionen $\sum C$ und $\sum T$ geringer. Die Big-Valley-Struktur ist bei den Zielfunktionen vom Summentyp nicht so stark ausgeprägt wie bei den Zielfunktionen C_{max} und $|L|_{max}$. Ähnliche Ergebnisse lieferten die Versuche für die praktischen Job-Shop-Probleme (Tabellen A.4 und A.5).

Eine nähere Betrachtung der einzelnen Instanzen zeigte, dass beim Vergleich der Instanzen aus der Menge der $J||f$ mit den Instanzen aus der Menge $J|2SETS|f$, $f \in \{C_{max}, |L|_{max}, \sum C, \sum T\}$ die letzteren geringere Korrelationskoeffizienten aufweisen. Bei den $J|2SETS|f$-Instanzen ist die Big-Valley-Struktur demnach weniger ausgeprägt. Die Unterschiede bei den Zielfunktionen und den Instanzengruppen zeigen sich auch in den Abständen zwischen den lokalen Optima. Tabelle 2.24 enthält die Abstände der lokalen Minima bei den 20×15-Instanzen. Zu jeder der 10 Instanzen wurden beginnend mit zufällig erzeugten Startlösungen 100 lokale Optima für die Zielfunktionen C_{max} und $\sum C$ berechnet. Dazu wurde der Sidestep-Algorithmus verwendet.

$\sum C$	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Distanz	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
$D(\alpha,\beta)$	0.2782	0.2204	30	29	0.2687	0.2204
$S^2(\alpha,\beta)$	0.2779	0.2147	30	28	0.2642	0.2014
$I(\alpha,\beta)$	0.2806	0.2240	31	29	0.2691	0.2128
$T(\alpha,\beta)$	0.2287	0.2748	24	30	0.2237	0.2686
$L(\alpha,\beta)$	0.2297	0.2882	26	30	0.2250	0.2780

Tabelle 2.22: *Durchschnitt der Korrelationskoeffizienten und Signifikanzen bei klassischen Job-Shop-Problemen bis 400 Tasks mit der Zielfunktion* $\sum C$

$\sum T$	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Distanz	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
$D(\alpha,\beta)$	0.4077	0.3772	33	31	0.4018	0.3692
$S^2(\alpha,\beta)$	0.4056	0.3600	33	31	0.3977	0.3550
$I(\alpha,\beta)$	0.4074	0.3731	33	32	0.4006	0.3669
$T(\alpha,\beta)$	0.3490	0.4223	31	32	0.3434	0.4115
$L(\alpha,\beta)$	0.3351	0.4031	32	32	0.3373	0.3955

Tabelle 2.23: *Durchschnitt der Korrelationskoeffizienten und Signifikanzen bei klassischen Job-Shop-Problemen bis 400 Tasks mit der Zielfunktion* $\sum T$

Mit der schnellen lokalen Suche wurde überprüft, dass die Lösungen lokal minimal sind. $\overline{D}(\alpha,\beta)$ ist der mittlere Abstand und $D_{max}(\alpha,\beta)$ ist der maximale Abstand zwischen zwei Lösungen in der Menge der 100 lokalen Minima. Es ist zu beobachten, dass die lokalen Optima bei $J||f$-Instanzen stärker konzentriert sind als bei $J|2SETS|f$-Instanzen. Die $J|2SETS|f$-Instanzen sind in der Tabelle 2.24 mit * gekennzeichnet. Dies und die schwächere Korrelation erklärt die in Abschnitt 2.2.5 gemachte Beobachtung, dass $J|2SETS|C_{max}$-Instanzen schwieriger zu optimieren sind. Weiterhin ist der Tabelle zu entnehmen, dass lokale Optima für die Zielfunktion C_{max} stärker konzentriert sind als lokale Optima für die Zielfunktion $\sum C$. Analoge Beziehungen treten auch bei Instanzen anderer Dimensionen $|\mathcal{J}| \times |\mathcal{M}|$ auf. Die Beobachtungen sind wieder unabhängig von dem verwendeten Abstandsmaß. Des Weiteren wurden auch die Zielfunktionen $|L|_{max}$ und $\sum T$ untersucht. Die Abstände der lokalen Optima für $|L|_{max}$ liegen im Bereich der Abstände für C_{max}. Gleiches ist bei dem Vergleich von $\sum T$ und $\sum C$ zu erkennen. Die größeren

	C_{max}		$\sum C$	
Instanz	$\overline{D}(\alpha,\beta)$	$D_{max}(\alpha,\beta)$	$\overline{D}(\alpha,\beta)$	$D_{max}(\alpha,\beta)$
ABZ 7	619.92	902	912.20	1366
SWV 6*	1257.66	1806	1568.36	2214
SWV 7*	1204.45	1704	1519.68	2312
TA11	682.32	1076	948.06	1354
TA13	629.41	926	968.20	1484
DMU 1	669.66	1016	939.54	1350
DMU 2	614.55	924	943.94	1460
DMU41*	1251.87	1886	1544.10	2102
DMU44*	1291.94	1854	1572.60	2226
DMU45*	1273.78	1866	1590.96	2360

Tabelle 2.24: *Durchschnittliche und maximale Abstände zwischen lokalen Minima*

Abstände zwischen lokalen Minima und die schwächeren Korrelationskoeffizienten führen zu der Vermutung, dass die Zielfunktionen vom Summentyp schwieriger als die Zielfunktionen vom min-max-Typ zu optimieren sind.

2.3.3 Abstände und Mittelwertbildung

Im letzten Abschnitt wurde ein Eindruck der Landschaft der Job-Shop-Probleme mit den Zielfunktionen C_{max} $|L|_{max}$, $\sum T$ und $\sum C$ vermittelt. Die Korrelationen begründen die Vermutung, dass bei allen vier Zielfunktionen ein Big Valley existiert. Das Ziel der genetischen lokalen Suche ist es, Startlösungen für die Lokale-Suche-Heuristiken zu erzeugen, die in einer viel versprechenden Region des Suchraumes liegen. Eine Rekombinationslösung sollte idealerweise den Abstand zum globalen Optimum im Suchraum im Vergleich mit den Elternlösungen verringern. Es bleibt demnach zu untersuchen, ob die Mittelwertbildung Lösungen erzeugt, die die Struktur des Big Valley nutzen. Die Mittelwertbildung legt die Reihenfolgen der Vorgänge auf den Maschinen anhand der Startzeiten der Tasks s_i, $i = 1, \ldots, |\mathcal{T}|$, in den Elternlösungen π^j, $j = 1, \ldots, k$, fest. Damit haben die Startzeiten direkten Einfluss auf die Permutationen der Vorgänge auf den Maschinen in der Rekombinationslösung. Im letzten Abschnitt wurde die Big-Valley-Struktur mittels der fünf Abstandsmaße auf diesen Permutationen experimentell verifiziert. Hier stellt sich die Frage, ob die Big-Valley-Struktur auch bei einem

über die Startzeiten definierten Abstandsmaß existiert. Dazu wird das folgende Abstandsmaß verwendet:

Definition 2.17 *Seien x^1 und x^2 zwei Lösungen einer Instanz eines Job-Shop Scheduling-Problems und s_i^j, $i = 1, \ldots, |\mathcal{T}|$, $j = 1, 2$, die dazugehörigen Startzeiten. Dann ist das Zeitabstandsmaß Z wie folgt definiert:*

$$Z(x^1, x^2) = \sum_{i=1}^{|\mathcal{T}|} |s_i^1 - s_i^2|.$$

Mit diesem Distanzmaß wurden wie in Abschnitt 2.3.2 die Korrelationskoeffizienten ermittelt. Tabelle 2.25 zeigt die durchschnittlichen Korrelationskoeffi-

$Z(x^1, x^2)$	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Zielfunktion	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
C_{max}	0.8613	0.8052	60	60	0.8259	0.6552
$\sum C$	0.4979	0.3460	54	42	0.4594	0.2753
$\sum T$	0.5552	0.4092	55	42	0.5270	0.3557
$\|L\|_{max}$	0.8768	0.7985	60	60	0.8476	0.6571

Tabelle 2.25: *Durchschnitt der Korrelationskoeffizienten und Signifikanzen bei klassischen Job-Shop-Problemen bis 600 Tasks (60 Instanzen) mit dem Abstandsmaß $Z(x^1, x^2)$*

zienten für die vier verwendeten Zielfunktionen und Distanzmaß Z. Es wurde der Durchschnitt über die ersten 60 Instanzen (aus Tabelle A.1 im Anhang) gebildet. Im Vergleich mit den Daten zu den anderen Distanzmaßen zeigen sich bei dem Distanzmaß Z höhere Korrelationskoeffizienten. Dies ist nicht überraschend, da der Einfluss der Startzeiten auf die Zielfunktionen größer ist als der Einfluss der Reihenfolgen der Vorgänge. Die gleichen Ergebnisse lieferte die Untersuchung der praktischen Job-Shop Scheduling-Probleme. Die Korrelationskoeffizienten für die 70 Instanzen der $PJSP$ sind in Tabelle A.6 im Anhang verzeichnet.

Die Frage ist nun, ob die durch Mittelwertbildung erzeugten Lösungen näher am globalen Optimum liegen als die zur Rekombination verwendeten lokalen Optima. Hierzu wurden bei den untersuchten Instanzen 100 lokale Optima ermittelt. Die Startlösungen für die Läufe der lokalen Suche wurden zufällig erzeugt. Es wurde mit allen Distanzmaßen der durchschnittliche Abstand

Distanzmaß	$\overline{A}_g^l$	$\overline{A}_g^2$	$\overline{A}_g^3$	$\overline{A}_g^4$
$D(\alpha,\beta)$	1316.56	1215.52	1184.12	1167.46
$S^2(\alpha,\beta)$	7537.80	6490.06	6037.20	5978.48
$I(\alpha,\beta)$	865.68	790.76	767.83	746.75
$T(\alpha,\beta)$	304.70	299.34	295.87	293.07
$L(\alpha,\beta)$	193.44	191.10	188.55	188.47
$Z(x^1,x^2)$	83336.57	197748.49	181935.14	173871.88

Tabelle 2.26: *Durchschnittlicher Abstand zwischen lokalen Optima, Mehrheitslösungen und dem globalen Optimum (Instanz: TA38, Zielfunktion: C_{max})*

Distanzmaß	$\overline{A}_g^l$	$\overline{A}_g^2$	$\overline{A}_g^3$	$\overline{A}_g^4$
$D(\alpha,\beta)$	2655.60	2328.90	2169.62	2080.22
$S^2(\alpha,\beta)$	16376.14	12627.80	10755.18	10125.56
$I(\alpha,\beta)$	1736.12	1502.57	1389.57	1332.14
$T(\alpha,\beta)$	574.10	551.00	543.97	533.93
$L(\alpha,\beta)$	356.69	343.03	338.25	336.27
$Z(x^1,x^2)$	210063.00	552187.06	549231.43	549506.14

Tabelle 2.27: *Durchschnittlicher Abstand zwischen lokalen Optima und Mehrheitslösungen zum globalen Optimum (Instanz: DRI55, Zielfunktion: $\sum T$)*

$\overline{A}_g^l = \frac{1}{100}\sum_{i=1}^{100} A(x_{global}, x_i^{lokal})$ zwischen den 100 lokalen Optima und dem globalen Optimum berechnet. Weiterhin wurden aus den lokalen Optima jeweils 100 Mittelwertlösungen mit 2, 3 und 4 Eltern erzeugt. Für jede der 100 Mittelwertbildungen wurden die Elternlösungen zufällig gemäß Gleichverteilung aus dem Pool gewählt. Es wurden damit die durchschnittlichen Abstände $\overline{A}_g^\alpha, \alpha \in \{2,3,4\}$ zwischen dem globalen Optimum und den Mittelwertlösungen mit den verschiedenen Elternanzahlen berechnet.

Tabelle 2.26 zeigt die so erhaltenen Daten für die Instanz TA38 mit der Zielfunktion C_{max}, und Tabelle 2.27 enthält die Werte für die Instanz DRI55 mit der Zielfunktion $\sum T$. Es ist zu erkennen, dass die Mittelwertlösungen unter Berücksichtigung der auf den Permutationen definierten Abstandsmaße im Durchschnitt näher am globalen Optimum liegen als die unabhängig erzeugten lokalen Optima. Die Werte für den Zeitabstand zeigen hingegen größere Distanzen. Das liegt darin begründet, dass die Mittelwertlösungen keine lokalen Optima sind und schlechtere Zielfunktionswerte aufweisen. Weiterhin ist

zu erkennen, dass die durchschnittlichen Abstände mit wachsender Elternzahl abnehmen. Das erklärt die in Abschnitt 2.2.3 getroffene Beobachtung, dass die genetische lokale Suche mit wachsender Elternzahl für die Rekombination schneller konvergiert.

Analoge Untersuchungen wurden auch für andere Instanzen durchgeführt, bei denen die globalen Optima bekannt waren. Dabei wurden vergleichbare Ergebnisse erzielt.

Kapitel 3

Multikriterielle Optimierung und Entscheidungs-Unterstützung

3.1 Multikriterielle Heuristiken

Mit der im letzten Kapitel vorgestellten genetischen lokalen Suche steht ein robuster Algorithmus zur Verfügung, der gute Ergebnisse für die verschiedenen Zielfunktionen und Verallgemeinerungen liefert. Auf dieser Basis werden im Weiteren Heuristiken für die multikriterielle Optimierung entwickelt. Die untersuchten Zielfunktionen sind wie im letzten Kapitel C_{max}, $|L|_{max}$, $\sum C$ und $\sum T$. Das Hauptaugenmerk liegt hierbei auf der Approximation der Menge der Pareto-optimalen Lösungen. Dazu müssen die Selektionsstrategie und die Lokale-Suche-Heuristiken in der genetischen lokalen Suche erweitert werden. Es werden nur Erweiterungen des Sidestep-Algorithmus und des Threshold-Accepting verwendet, da sich diese Heuristiken im Fall einer Zielfunktion bei den Experimenten in Abschnitt 2.2.4 als die besten Algorithmen herausgestellt haben.

3.1.1 Multikriterieller Sidestep-Algorithmus

Für die Erweiterung der Sidestep-Heuristik zur Approximation der Pareto-Menge muss eine geeignete aggregierte Zielfunktion für die Suchrichtung in

der lokalen Suche definiert werden. Hierfür bieten sich die Ranking-Funktionen aus Abschnitt 1.2.2 an. Diese können aber nicht direkt zum Vergleich der aktuellen Lösung mit einer Nachbarlösung verwendet werden, da die Ranking-Funktionen beim Vergleich von je zwei Lösungen nicht transitiv sind.

Beispiel
In dem Beispiel mit drei Lösungen x_j, $j = 1, 2, 3$, und drei Zielfunktionen f_i, $i = 1, 2, 3$, werden jeweils zwei Lösungen anhand des Wertes der Ranking-Funktion F^2 verglichen. Die Werte der Ranking-Funktion F^2 werden bei dem Vergleich auf Basis der zwei Lösungen berechnet. Damit ändern sich das Minimum min_{f_i} und Maximum max_{f_i} der Zielfunktionen bei jedem Vergleich. Die Werte für die einzelnen Lösungen bei den drei Zielfunktionen sind in der folgenden Tabelle verzeichnet:

	$f_1(x_j)$	$f_2(x_j)$	$f_3(x_j)$
x_1	5	6	7
x_2	6	7	5
x_3	7	5	6

Die Zielfunktionen f_i, $i = 1, 2, 3$, sind in dem Beispiel alle gleich gewichtet, $w_i = \frac{1}{3}$ für alle $i \in \{1, 2, 3\}$. Die Werte der Ranking-Funktion bei den drei Vergleichen berechnen sich wie folgt:

$$\begin{aligned}
\mathrm{F}^2(x_1) &= \tfrac{1}{3}\left(\tfrac{5-5}{6-5} + \tfrac{6-6}{7-6} + \tfrac{7-5}{7-5}\right) = \tfrac{1}{3}\\
\mathrm{F}^2(x_2) &= \tfrac{1}{3}\left(\tfrac{6-5}{6-5} + \tfrac{7-6}{7-6} + \tfrac{5-5}{7-5}\right) = \tfrac{2}{3}\\
\mathrm{F}^2(x_1) &= \tfrac{1}{3}\left(\tfrac{5-5}{7-5} + \tfrac{6-5}{6-5} + \tfrac{7-6}{7-6}\right) = \tfrac{2}{3}\\
\mathrm{F}^2(x_3) &= \tfrac{1}{3}\left(\tfrac{7-5}{7-5} + \tfrac{5-5}{6-5} + \tfrac{6-6}{7-5}\right) = \tfrac{1}{3}\\
\mathrm{F}^2(x_2) &= \tfrac{1}{3}\left(\tfrac{6-6}{7-6} + \tfrac{7-5}{7-5} + \tfrac{5-5}{6-5}\right) = \tfrac{1}{3}\\
\mathrm{F}^2(x_3) &= \tfrac{1}{3}\left(\tfrac{7-6}{7-6} + \tfrac{5-5}{7-5} + \tfrac{6-5}{6-5}\right) = \tfrac{2}{3}
\end{aligned}$$

Falls die drei Lösungen benachbart sind, kann eine Lokale-Suche-Heuristik und damit auch der Sidestep-Algorithmus in einen Zyklus geraten. Beim Übergang von Lösung x_1 nach x_3, von x_3 nach x_2 und schließlich von x_2 nach x_1 verbessert sich der Wert von F^2 in jedem Schritt.

Um den Zyklen zu entgehen, wird in dem multikriteriellen Sidestep-Algorithmus eine Menge R von Referenzlösungen verwendet. Als Ranking-Funktion wird F^2 genutzt. Zu Beginn des Sidestep-Algorithmus werden die Werte von $\mathrm{F}^2(x)$ für alle $x \in R$ berechnet und die Lösungen in R danach sortiert. Die Werte für das Minimum min_{f_i} und das Maximum max_{f_i} beziehen sich

im Algorithmus nur auf die Lösungen in der Referenzmenge R und bleiben während der Abarbeitung des Algorithmus 3.1 konstant. Für die aktuelle Lösung x und eine Nachbarlösung x' werden der Wert der Ranking-Funktion und der Rang der Lösung bezüglich der Menge der Referenzlösungen ermittelt. Im Algorithmus wird von einer aktuellen Lösung x zur Nachbarlösung x' übergegangen, falls die Nachbarlösung den gleichen oder einen kleineren Rang hat. Ein Übergang zu einer Nachbarlösung mit gleichem Rang wird als Sidestep bezeichnet.

Algorithmus 3.1 *Multikriterieller Sidestep-Algorithmus*
Seien N_α die verwendete Nachbarschaft, $F = \{f_i\}_{i=1,\ldots,m}$ die Menge der Zielfunktionen, $W = \{w_i\}_{i=1,\ldots,m}$ eine Menge von Gewichten, IS die maximale Anzahl von Sidesteps, R die Menge der Referenzlösungen und P die Menge der gefundenen Pareto-Lösungen.

1: Berechne $\mathrm{F}^2(x)$ für alle $x \in R$ und ordne die Lösungen $x \in R$ aufsteigend nach den $\mathrm{F}^2(x)$.

2: Ermittle eine Startlösung $x \in \mathscr{X}$ und setze $P := \{x\}$.

3: Ermittle die Zielfunktion $\mathrm{F}^2(x)$ und den Rang π von x in R.

4: $i := N - 1$ und $k := 0$.

5: Wähle zufällig eine Permutation σ aus S_{N-i}.

6: Für $j := 1, \ldots, N - i$

7: Sei $x' := N_\alpha^{\sigma(j),\sigma(j)+i}(x)$ eine zulässige Lösung.
Berechne $\mathrm{F}^2(x')$ und ermittle den Rang π' von x' in R.
Falls $\pi' = \pi = 1$ oder $\pi' = \pi = |R| + 1$
Falls $\mathrm{F}^2(x') < \mathrm{F}^2(x)$, dann setze $x := x'$, $\pi := \pi'$ und $k := k + 1$.
Sonst
Falls $\pi' \leq \pi$, dann setze $x := x'$, $\pi := \pi'$ und
falls $\pi' < \pi$, setze $k := 0$, sonst $k := k + 1$.
Falls x' Pareto-optimal in P, dann setze $P := P \cup \{x'\}$, $k := 0$ und entferne alle dominierten Lösungen aus P.

8: Setze $i := i - 1$. Falls $i = 0$, setze $i := N - 1$.

9: Falls $k = IS$, stopp. Gib P aus.

10: Gehe zu Schritt 5.

Eine Ausnahme bildet der Fall, bei dem beide Lösungen den Rang 1 oder den Rang $|R| + 1$ haben. Das heißt, beide Lösungen sind bezüglich F^2 besser oder schlechter als alle Referenzlösungen. In diesem Fall wird nur zur Nachbarlösung x' übergegangen, falls $\mathrm{F}^2(x') < \mathrm{F}^2(x)$ gilt.

Der Algorithmus 3.1 sucht nach einer Menge P von Pareto-optimalen Lösungen. Diese Menge wird zu Beginn der Heuristik mit der Startlösung initialisiert. Bei jeder im Laufe der Nachbarschaftssuche erzeugten Lösung wird überprüft, ob sie bezüglich der Menge P nichtdominiert ist. Ist dies der Fall, wird sie in P eingefügt. Die eingefügte Lösung kann andere Lösungen in P dominieren. Diese werden aus P entfernt.

Die Richtung, in der sich die Nachbarschaftssuche bewegt, wird durch die Ranking-Funktion und die Menge R der Referenzlösungen beeinflusst. Um eine möglichst große Diversität der Lösungen im Pool der genetischen lokalen Suche zu erreichen, werden die Gewichte w_i der Zielfunktionen f_i, $i = 1, \ldots, m$, zu Beginn des multikriteriellen Sidestep-Algorithmus zufällig gemäß Gleichverteilung ermittelt.

Als Menge der Referenzlösungen werden die Lösungen aus dem Pool der genetischen lokalen Suche verwendet. Dies wirft Probleme zu Beginn der genetischen lokalen Suche auf, da hier der Pool noch keine Lösungen enthält. Aus diesem Grund wird der Pool bis zur vorgegebenen maximalen Poolgröße PG entweder mit zufälligen Startlösungen oder mit lokalen Optima bezüglich der einzelnen Zielfunktionen f_i, $i = 1, \ldots, m$, gefüllt.

Der Aufwand zur Erweiterung des Sidestep-Algorithmus für die lexikographische Optimierung ist geringer als für die Approximation der Pareto-Menge. Im Gegensatz zu Algorithmus 2.12 müssen die Zielfunktionswerte aller Zielfunktionen f_i, $i = 1, \ldots, m$, der aktuellen Lösung gespeichert werden. Es wird von der aktuellen Lösung zu einer Nachbarlösung übergegangen, falls die Nachbarlösung im Sinne der lexikographischen Optimierung besser als die aktuelle Lösung ist oder alle Zielfunktionen beider Lösungen die gleichen Werte haben. Der letztere Fall wird als Sidestep bezeichnet.

3.1.2 Multikriterielles Threshold-Accepting

Für die Erweiterung des Threshold-Accepting für die Approximation der Pareto-Menge werden zwei Änderungen zu Algorithmus 2.13 eingeführt. Wie im multikriteriellen Sidestep-Algorithmus wird eine Menge P der im Algorithmus gefundenen Pareto-Lösungen gespeichert. Die Menge P enthält zu Beginn des Algorithmus die übergebene Startlösung. Bei jeder in der Nachbarschaftssuche erzeugten Lösung wird die Pareto-Optimalität überprüft und die Menge P aktualisiert.

Algorithmus 3.2 *Multikriterielles Threshold-Accepting*
Seien N_α die verwendete Nachbarschaft, $F = \{f_k\}_{k=1,\dots,m}$ die Menge der Zielfunktionen, T der Faktor für den Schwellwert, λ die Abnahmerate und P die Menge der gefundenen Pareto-Lösungen.

1: Ermittle eine Startlösung $x \in \mathscr{X}$ und setze $P := \{x\}$.

2: $S_k := \lfloor T \cdot f_k(x) \rfloor$ für alle $k = 1, \dots, m$.

3: $i := N - 1$ und $i^\star := i$.

4: Wähle eine zufällige Permutation σ aus S_{N-i}.

5: Für $j := 1, \dots, N - i$

6: Sei $x' := N_\alpha^{\sigma(j),\sigma(j)+i}(x)$ eine zulässige Lösung.
Falls $f_k(x') < S_k$ für alle $k = 1, \dots, m$, dann setze $x := x'$,
$i^\star := i$, und falls $\lfloor T \cdot f_k(x) \rfloor < S_k$, setze $S_k := \lfloor T \cdot f_k(x) \rfloor$
für alle $k = 1, \dots, m$.
Falls x' Pareto-optimal in P, dann setze $P := P \cup \{x'\}$, $i^\star := i$
und entferne alle dominierten Lösungen aus P.

7: Setze $i := i - 1$. Falls $i = 0$, setze $i := N - 1$.

8: Falls $i = i^\star$, stopp. Gib P aus.

9: Für $k := 1, \dots, m$

10: Falls $\lfloor T \cdot f_k(x) \rfloor < S_k$, setze $S_k := \lfloor T \cdot f_k(x) \rfloor$.
Sonst $\delta_k := \lfloor \lambda(S_k - f_k(x)) \rfloor$ und $S_k := S_k - \max\{1, \delta_k\}$.

11: Gehe zu Schritt 4.

Weiterhin müssen im multikriteriellen Fall die Schwellwerte S_k, $k = 1, \ldots, m$, für alle Zielfunktionen ermittelt und im Laufe der Nachbarschaftssuche aktualisiert werden. Die Parameter T und λ sind für alle Zielfunktionen gleich. In der Heuristik wird die Nachbarlösung akzeptiert, falls die Zielfunktionswerte aller f_k, $k = 1, \ldots, m$, kleiner als der aktuelle Schwellwert S_k sind. Im Gegensatz zum Threshold-Accepting bei einer Zielfunktion wird bei jeder akzeptierten Nachbarlösung der Schwellwert S_k, $k = 1, \ldots, m$, aktualisiert. Der Algorithmus stoppt, falls im letzten Nachbarschaftsdurchlauf keine Nachbarlösung akzeptiert oder keine neue Pareto-optimale Lösung gefunden wurde.

Die Erweiterung für die lexikographische Optimierung erfolgt in gleicher Weise. Eine Nachbarlösung x' wird akzeptiert, falls $f_k(x') < S_k$ für alle $k = 1, \ldots, m$ gilt. Die bezüglich der lexikographischen Optimierung beste Lösung wird gespeichert und am Ende der Heuristik ausgegeben.

3.1.3 Multikriterielle genetische lokale Suche

Die Mittelwertbildung kann für multikriterielle Job-Shop Scheduling-Probleme ohne Änderung verwendet werden. Es werden weiterhin $k \in \{2, \ldots, |P|\}$ Elternlösungen zufällig gemäß Gleichverteilung aus dem Pool P der vorhandenen Lösungen ausgewählt. Die Rekombination funktioniert in gleicher Weise wie im Fall mit einer Zielfunktion.

Die Methode der Selektion der Lösungen, die aus dem Pool entfernt werden, muss für den Fall der multikriteriellen Optimierung angepasst werden. Bei einer Zielfunktion wird nach dem Einfügen einer Lösung in den Pool eine Lösung mit dem größten Zielfunktionswert entfernt. Mit der Dominanzrelation der Pareto-Dominanz ist eine lineare Ordnung der Lösungen im Pool nicht möglich. Weiterhin kann der Pool in der genetischen lokalen Suche je nach Größe des Pools und Anzahl der Pareto-optimalen Lösungen sowohl nur nichtdominierte als auch nichtdominierte und dominierte Lösungen enthalten. Aus diesem Grund werden zur Ordnung der Lösungen die Ranking-Funktionen F^α, $\alpha = 1, \ldots, 4$, aus Abschnitt 1.2.2 verwendet. Die in diesen Ranking-Funktionen verwendeten Gewichte $W = \{w_i\}_{i=1,\ldots,m}$ modellieren die Präferenzen des Entscheidungsträgers. Da diese *a priori* nicht bekannt sind, werden im Weiteren bei der Selektion alle Zielfunktionen $F = \{f_i\}_{i=1,\ldots,m}$ gleich gewichtet. Das heißt, es sei $w_i = \frac{1}{m}$ für alle $i = 1, \ldots, m$.

Die Algorithmen 3.1 und 3.2 haben die Möglichkeit, mehr als eine Lösung

zu erzeugen. Jede dieser Lösungen wird in den Pool eingefügt, falls noch keine Lösung mit demselben kritischen Pfad im Pool existiert. Danach werden die Lösungen mit Hilfe einer Ranking-Funktion sortiert und eine dominierte Lösung mit dem größten Ranking-Funktionswert aus dem Pool entfernt. Falls im Pool nur Pareto-optimale Lösungen vorhanden sind, wird eine nichtdominierte Lösung mit dem größten Ranking-Funktionswert entfernt. Danach wird die Ranking-Funktion neu berechnet, und die Lösungen werden wieder sortiert. Falls die Anzahl der Lösungen im Pool größer als der vorgegebene Parameter PG für die Poolgröße ist, wird eine weitere Lösung entfernt. Dies wird so lange iteriert, bis die vorgegebene Poolgröße erreicht ist.

3.1.4 Vergleich der multikriteriellen Heuristiken

Um multikriterielle Heuristiken zu vergleichen, wird eine Methode zum Vergleich der Qualität von Mengen potentieller Pareto-Lösungen benötigt. Für quantitative Vergleichsmethoden sind Referenzpunkte oder -mengen erforderlich. Weiterhin wird ein Gewichtsvektor für die Zielfunktionen benötigt, um die Präferenzen des Entscheidungsträgers nachzubilden [24]. Sowohl der Gewichtsvektor als auch die Referenzmenge sind von den Problemparametern und der Anwendung abhängig. Als Referenzmenge wäre die tatsächliche Pareto-Menge oder der ideale Punkt möglich. Beides steht bei den hier behandelten NP-schweren Problemen nicht zur Verfügung. Auch die Gewichtung der Zielfunktionen ist ohne Entscheidungsträger *a priori* nicht bekannt.

Aus diesen Gründen werden die Approximationen der Menge der Pareto-Lösungen durch die verschiedenen Heuristiken mittels **Outperformance-Relationen** verglichen. Gegeben sind zwei Approximationen A und B. Aus diesen muss die schließlich anzuwendende Lösung ausgewählt werden. Der Entscheidungsträger kann sich auf die Menge $ND(A \cup B)$ der nichtdominierten Lösungen aus der Vereinigung von A und B beschränken. Damit können Outperformance-Relationen basierend auf den Dominanzbeziehungen definiert werden.

Definition 3.3 *(Outperformance-Relationen)*

1. *Die Approximation A überbietet B schwach, bezeichnet mit $A\ O_W B$, falls $A \neq B$ und $ND(A \cup B) = A$. Das heißt, falls für jeden Punkt $z_2 \in B$ ein Punkt $z_1 \in A$ existiert, der gleiche Zielfunktionswerte wie z_2 hat oder z_2 dominiert und mindestens einen Punkt $z_1 \in A$, der nicht in B ist.*

2. *Die Approximation A überbietet B stark, bezeichnet mit $A\ O_S B$, falls $ND(A \cup B) = A$ und $B \setminus ND(A \cup B) \neq \emptyset$. Das heißt, falls für jeden Punkt $z_2 \in B$ ein Punkt $z_1 \in A$ existiert, der gleiche Zielfunktionswerte wie z_2 hat oder z_2 dominiert und mindestens ein Punkt $z \in B$ existiert, der von einem Punkt $z_1 \in A$ dominiert wird.*

3. *Die Approximation A überbietet B komplett, bezeichnet mit $A\ O_C B$, falls $ND(A \cup B) = A$ und $B \cap ND(A \cup B) = \emptyset$. Das heißt, falls jeder Punkt $z \in B$ durch einen Punkt $z_1 \in A$ dominiert wird.*

Es gilt $O_C \subset O_S \subset O_W$, d.h. die komplette Outperformance-Relation ist die stärkste und die schwache Outperformance-Relation ist die schwächste der Relationen. Diese Relationen definieren eine Halbordnung auf der Menge der Approximationen. Zum Vergleich der multikriteriellen Heuristiken wird im Weiteren die schwache Outperformance-Relation verwendet.

Vergleich der multikriteriellen genetischen lokalen Suche mit Threshold-Accepting und Sidestep-Algorithmus

Bei diesem Vergleich wurden die ersten 60 Benchmark-Instanzen aus Tabelle A.1 im Anhang und 20 Instanzen — je 5 aus den 4 Gruppen — der praktischen Job-Shop-Probleme verwendet.

Der Pool der multikriteriellen genetischen lokalen Suche bestand aus 50 Lösungen. F^2 wurde als Ranking-Funktion der Lösungen im Pool verwendet. Alle Zielfunktionen wurden gleich gewichtet. Bei beiden Lokale-Suche-Heuristiken wurde die Nachbarschaft N_1 verwendet. Im Sidestep-Algorithmus wurden 8000 Sidesteps zugelassen. Die Parameter für das Threshold-Accepting waren $T := 1.01$ und $\lambda := 0.07$. Die ersten 50 Läufe der lokalen Suchen wurden mit zufällig erzeugten Lösungen gestartet. Danach wurden je zwei Lösungen zufällig gemäß Gleichverteilung aus dem Pool der GLS ausgewählt und die Mittelwertlösung gebildet. Diese wurde als Startlösung verwendet. Da der multikriterielle Sidestep-Algorithmus einen Pool von Referenzlösungen benötigt, wurde der Pool der GLS mit zufälligen Startlösungen initialisiert. Mit jeder Benchmark-Instanz wurde je ein Lauf der GLS mit Sidestep-Algorithmus und ein Lauf mit Threshold-Accepting durchgeführt. Die multikriterielle genetische lokale Suche wurde nach 1800 Sekunden gestoppt, wobei der letzte Lauf der Lokale-Suche-Heuristiken nicht abgebrochen wurde. Danach wurden die dominierten Lösungen aus den beiden so erzeugten Mengen von Lösungen entfernt. Die so erhaltenen potentiellen Pareto-Lösungen

bildeten die Approximationen A und B. Diese wurden mittels der schwachen Outperformance-Relation verglichen.

Instanzen	$\#(C_{max}, \lvert L\rvert_{max})$	$\#(\sum C, \sum T)$	$\#(C_{max}, \sum T)$
10×10	9/1/0	8/2/0	4/6/0
15×15	8/2/0	7/2/1	3/7/0
20×15	9/1/0	5/0/5	6/4/0
20×20	9/1/0	8/0/2	7/2/1
30×15	10/0/0	0/0/10	10/0/0
30×20	10/0/0	0/0/10	8/1/1
D1-D5	5/0/0	1/3/1	5/0/0
DR21-DR25	5/0/0	3/1/1	5/0/0
DRI41-DRI45	0/5/0	5/0/0	5/0/0
DRIR56-DRIR60	0/5/0	4/1/0	4/1/0

Tabelle 3.1: *Vergleich der GLS mit Sidestep-Algorithmus und Threshold-Accepting*

Tabelle 3.1 enthält die Vergleichsdaten für die multikriterielle genetische lokale Suche mit Sidestep-Algorithmus und Threshold-Accepting. Sei A die durch die GLS mit Sidestep-Algorithmus und B die durch die GLS mit Threshold-Accepting erhaltene Approximation der Menge der Pareto-optimalen Lösungen einer Instanz I und der betrachteten Zielfunktionen $\#(f_1, f_2)$. Die Einträge $i/j/k$ in der Tabelle 3.1 geben die Resultate der Outperformance-Relation an. Das heißt, in i Fällen galt $A \; O_W B$ und in k Fällen $B \; O_W A$. j gibt die Anzahl der Instanzen an, bei denen die erzeugten Approximationen mit der schwachen Outperformance-Relation nicht vergleichbar waren.

Es wurden wie im letzten Kapitel die Zielfunktionen C_{max}, $\sum C$, $\sum T$ und $\lvert L\rvert_{max}$ verwendet. Von den sechs möglichen Kombinationen mit zwei Zielfunktionen wurden drei ausgewählt und die Vergleiche durchgeführt. Aus den Daten geht hervor, dass die multikriterielle genetische lokale Suche mit Sidestep-Algorithmus die besseren Ergebnisse liefert. Die Ausnahme hierbei ist die Kombination der Zielfunktionen $\#(\sum C, \sum T)$. Die GLS mit Threshold-Accepting erzeugt bei diesen Zielfunktionen und Instanzen ab 400 Vorgängen die besseren Approximationen. Um dies zu überprüfen, wurden die Algorithmen bei diesen Zielfunktionen auch auf der Grundlage größerer Instanzen verglichen. Auch bei diesen Instanzen lieferte die GLS mit Threshold-Accepting die besseren Ergebnisse.

Bei allen anderen Kombinationen von Zielfunktionen erreicht die GLS mit

Instanzen	$\#(C_{max}, \sum T, \lvert L\rvert_{max})$	$\#(C_{max}, \sum C, \sum T)$	$\#(C_{max}, \sum C, \sum T, \lvert L\rvert_{max})$
10×10	3/6/1	2/8/0	2/8/0
15×15	5/5/0	5/4/1	4/6/0
20×15	6/4/0	6/4/0	5/5/0
20×20	5/5/0	6/4/0	5/5/0
30×15	10/0/0	8/2/0	9/1/0
30×20	8/2/0	6/4/0	6/4/0

Tabelle 3.2: *Vergleich der GLS mit Sidestep-Algorithmus und Threshold-Accepting*

Sidestep-Algorithmus bessere Approximationen. In Tabelle 3.2 ist zu sehen, dass auch unter Verwendung von zwei Zielfunktionen vom Summentyp und einer Zielfunktion vom min-max-Typ die GLS mit Sidesteps die besseren Pareto-Optima erreicht. Die gleichen Resultate ergeben sich, wenn alle vier Zielfunktionen optimiert werden.

Vergleich weiterer Varianten der multikriteriellen Optimierung

Die Lösungen, die aus dem Pool der GLS zu entfernen sind, werden mittels Ranking-Funktionen selektiert. Im letzten Abschnitt wurde beim Vergleich der multikriteriellen Lokale-Suche-Heuristiken nur die Ranking-Funktion F^2 verwendet. Hier stellt sich die Frage, welchen Einfluss die Wahl der Ranking-Funktion auf die Qualität der Approximation der Menge der Pareto-optimalen Lösungen hat.

Für den Vergleich der Ranking-Funktionen wurde die GLS mit Sidestep-Algorithmus verwendet. Der Pool bestand aus 20 Lösungen und wurde als Referenzmenge mit zufälligen Startlösungen vorbelegt. Die ersten 20 Läufe des Sidestep-Algorithmus wurden ebenfalls von zufälligen Startlösungen aus gestartet. Danach wurden Mittelwertlösungen aus zwei zufällig ausgewählten Elternlösungen gebildet. In der Lokale-Suche-Heuristik waren 1000 Sidesteps zulässig. Die verwendete Nachbarschaft war N_1. Zu jeder Benchmark-Instanz und jeder Ranking-Funktion wurde ein Lauf der GLS durchgeführt. Das Zeitlimit betrug hierbei 500 Sekunden. Der letzte Lauf des Sidestep-Algorithmus wurde vollständig durchgeführt.

Tabelle 3.3 zeigt die Ergebnisse für 50 Instanzen des Problems $J||\#(C_{max}, \sum T)$.

Instanzen	F^1/F^2	F^3/F^4	F^1/F^3	F^2/F^4	F^1/F^4	F^2/F^3
10×10	1/7/2	1/6/3	1/8/1	1/6/3	2/8/0	0/8/2
15×15	4/5/1	0/8/2	5/5/0	1/8/1	3/4/3	1/8/1
20×15	5/4/1	2/6/2	2/3/5	3/4/3	2/5/3	5/3/2
20×20	2/3/5	2/6/2	6/3/1	4/3/3	1/3/6	6/4/0
30×15	4/3/3	1/4/5	2/4/4	2/5/3	2/4/4	6/3/1

Tabelle 3.3: *Vergleich der Ranking-Funktionen*

Beide Zielfunktionen waren gleich gewichtet. Den Daten ist keine Dominanz einer Ranking-Funktion zu entnehmen. Die Ranking-Funktionen wurden auch anhand weiterer Gruppen von Benchmark-Instanzen und Kombinationen von Zielfunktionen verglichen. Dabei war ebenfalls keine Dominanz einer Ranking-Funktion zu beobachten. Dies lässt vermuten, dass die Ranking-Funktionen nur einen geringen Einfluss auf die multikriterielle genetische lokale Suche haben.

Eine weitere Variante der multikriteriellen genetischen lokalen Suche betrifft die Erzeugung des Startpools. Bei einer Poolgröße PG werden die ersten PG Läufe mit zufälligen Startlösungen begonnen. Bisher wurden dabei in der multikriteriellen genetischen lokalen Suche die multikriteriellen Lokale-Suche-Heuristiken angewendet, und es wurde sofort nach Pareto-optimalen Lösungen gesucht. Eine andere Möglichkeit, eine Erstbelegung des Pools zu erzeugen, besteht darin, lokale Optima für jede der einzelnen Zielfunktionen $\{f_i\}$, $i = 1, \ldots, m$, zu verwenden. Das heißt, bei einer Poolgröße PG und m Zielfunktionen werden zu jeder Zielfunktion $\lceil \frac{PG}{m} \rceil$ lokale optimale Lösungen erzeugt. Dazu werden die Heuristiken für eine Zielfunktion verwendet, und es wird mit zufälligen Startlösungen begonnen. Für die Analyse, welche der beiden Möglichkeiten zur Startpoolerzeugung bessere Ergebnisse erzielt, wurden verschiedene Zielfunktionen und die beiden multikriteriellen Lokale-Suche-Heuristiken verwendet.

Die Instanzen der Probleme $J||\#(C_{max}, \sum T)$ und $J||\#(C_{max}, |L|_{max})$ wurden mit der GLS mit Sidestep-Algorithmus optimiert. Es waren dabei 5000 Sidesteps zulässig. Die verwendete Nachbarschaft war N_1. Im Pool der GLS wurden 20 Lösungen gespeichert. Für jede Instanz wurden zwei Läufe der GLS durchgeführt. Beim ersten Lauf der GLS wurde der Pool mit zufälligen Startlösungen gefüllt, um eine Referenzmenge für den multikriteriellen Sidestep-Algorithmus zu erhalten. Danach wurden 20 Läufe des multikriteriellen Sidestep-Algorithmus beginnend mit zufälligen Startlösungen durchgeführt.

Nach diesen 20 Läufen wurden Mittelwertlösungen aus zwei zufällig aus dem Pool gewählten Elternlösungen gebildet und als Startlösungen für den Sidestep-Algorithmus genutzt. Dies wurde bis zum Erreichen der Zeitschranke von 1000 Sekunden wiederholt, wobei der letzte Lauf des Sidestep-Algorithmus vollständig durchgeführt wurde. Beim zweiten Lauf der GLS für eine Instanz wurden jeweils 10 Läufe des Sidestep-Algorithmus zur Optimierung einer Zielfunktion durchgeführt. Dabei wurde von zufälligen Startlösungen aus begonnen. Danach wurde mit dem multikriteriellen Sidestep-Algorithmus beginnend mit Mittelwertlösungen fortgefahren. Die Mittelwertlösungen wurden aus zwei zufällig aus dem Pool gewählten Elternlösungen erzeugt. Nach dem Erreichen der Zeitschranke von 1000 Sekunden wurde die GLS abgebrochen. Bei den Instanzen des Problems $J||\#(C_{max}, \sum C, \sum T, |L|_{max})$ wurde ebenfalls die GLS mit Sidestep-Algorithmus angewandt. Die Unterschiede bestanden in einer Poolgröße von 40 Lösungen und einer Zeitschranke von 1800 Sekunden. Bei den Instanzen für das Problem $J||\#(\sum C, \sum T)$ wurde die GLS mit dem Threshold-Accepting verwendet. Die Parameter für das Threshold-Accepting waren $T := 1.001$ und $\lambda := 0.75$. Zeitschranke, Poolgröße und Nachbarschaft waren wie im obigen Fall mit zwei Zielfunktionen gewählt. Im ersten Lauf der GLS für jede Instanz wurden die ersten 20 Lösungen mit dem multikriteriellen Threshold-Accepting beginnend von zufälligen Startlösungen erzeugt. Im zweiten Lauf wurden je 10 Lösungen mit dem Threshold-Accepting und einer Zielfunktion erzeugt. Danach wurden bei beiden Varianten Mittelwertlösungen aus zwei Elternlösungen gebildet und mit dem multikriteriellen Threshold-Accepting optimiert. Nach Ablauf der Zeitschranke wurde die GLS abgebrochen. Bei allen Varianten wurde die Ranking-Funktion F^2 genutzt.

Instanzen	$\#(C_{max}, \sum T)$	$\#(\sum C, \sum T)$	$\#(C_{max}, \|L\|_{max})$	$\#(C_{max}, \sum C, \sum T, \|L\|_{max})$
10×10	1/8/1	3/4/3	0/9/1	0/10/0
15×15	0/8/2	2/3/5	0/8/2	0/9/1
20×15	3/4/3	3/2/5	2/6/2	0/9/1
20×20	2/4/4	3/1/6	1/7/2	1/7/2
30×15	1/1/8	4/0/6	0/6/4	0/7/3
30×20	0/2/8	5/0/5	0/4/6	1/5/4

Tabelle 3.4: *Vergleich der multikriteriellen GLS unter Verwendung verschiedener Heuristiken für die Erstbelegung der Pools*

Tabelle 3.4 enthält die Resultate für 60 Instanzen. Die Einträge $i/j/k$ ge-

ben wiederum die Ergebnisse der Outperformance-Relation an. Der Eintrag i gibt die Anzahl der Fälle an, bei denen die GLS bessere Lösungen lieferte, bei der sofort mit den multikriteriellen Lokale-Suche-Heuristiken begonnen wurde. Der Eintrag k gibt an, wie oft die Variante der GLS, die mit lokalen Optima für die einzelnen Zielfunktionen begann, besser war. Es ist zu beobachten, dass bei der GLS mit Sidestep-Algorithmus die Variante, die mit lokalen Optima beginnt, bei allen Instanzengruppen bessere oder zumindest vergleichbare Ergebnisse liefert. Je größer die Dimensionen der Instanzen, desto deutlicher wird dies. Bei kleineren Instanzen ist die Anzahl der Einträge j der Instanzen, die mit der Outperformance-Relation nicht vergleichbar sind, bedeutend höher. Bei diesen Instanzen führte die GLS innerhalb der Zeitbeschränkung mehr Mittelwertläufe aus. Je mehr Mittelwertläufe durchgeführt werden, desto geringer ist der Einfluss des Startpools, und beide Varianten erreichen vergleichbare Approximationen der Pareto-Menge. Auch bei der GLS mit Threshold-Accepting liefert die Variante mit lokalen Optima vergleichbare oder bessere Ergebnisse. Die Unterschiede zwischen den beiden Varianten sind im Vergleich mit der GLS mit Sidesteps geringer. Dies liegt zum einen daran, dass zwei Zielfunktionen vom Summentyp verwendet wurden. Zum anderen sind im Vergleich zum Sidestep-Algorithmus die Unterschiede zwischen multikriteriellem Threshold-Accepting und Threshold-Accepting bei einer Zielfunktion nicht so stark ausgeprägt. Es wird in Nachbarn gewechselt, deren Zielfunktionswerte alle unter den Schranken liegen. Die Suchrichtung hat aus diesem Grund beim Threshold-Accepting einen geringeren Einfluss.

Bei allen Zielfunktionskombinationen, Instanzengruppen und Lokale-Suche-Heuristiken liefert die Variante der GLS, die den Startpool mit lokalen Optima für alle Zielfunktionen füllt, bessere oder mindestens vergleichbare Approximationen. Je geringer die dem Algorithmus zur Verfügung stehende Zeit und je besser die Qualität der lokalen Optima ist, desto stärker ist diese Beobachtung.

3.2 Entscheidungs-Unterstützung

Der Abschnitt 3.1 beschäftigte sich mit Algorithmen zur Erzeugung von Approximationen für die Menge der Pareto-optimalen Lösungen. Die Aufgabe des Entscheidungsträgers besteht darin, aus dieser Menge von Lösungen diejenige auszuwählen, die für die Anwendung übernommen werden soll. Die Ranking-Funktionen können hierbei als multikriterielle Bewertungsmethoden

verwendet werden. Eine Schwierigkeit besteht in der subjektiven Gewichtung der Zielfunktionen. Aber selbst wenn die Präferenzen des Entscheiders quantifizierbar sind, können durch verschiedene Ranking-Funktionen auch bei gleicher Gewichtung der Zielfunktionen verschiedene Lösungen ausgewählt werden.

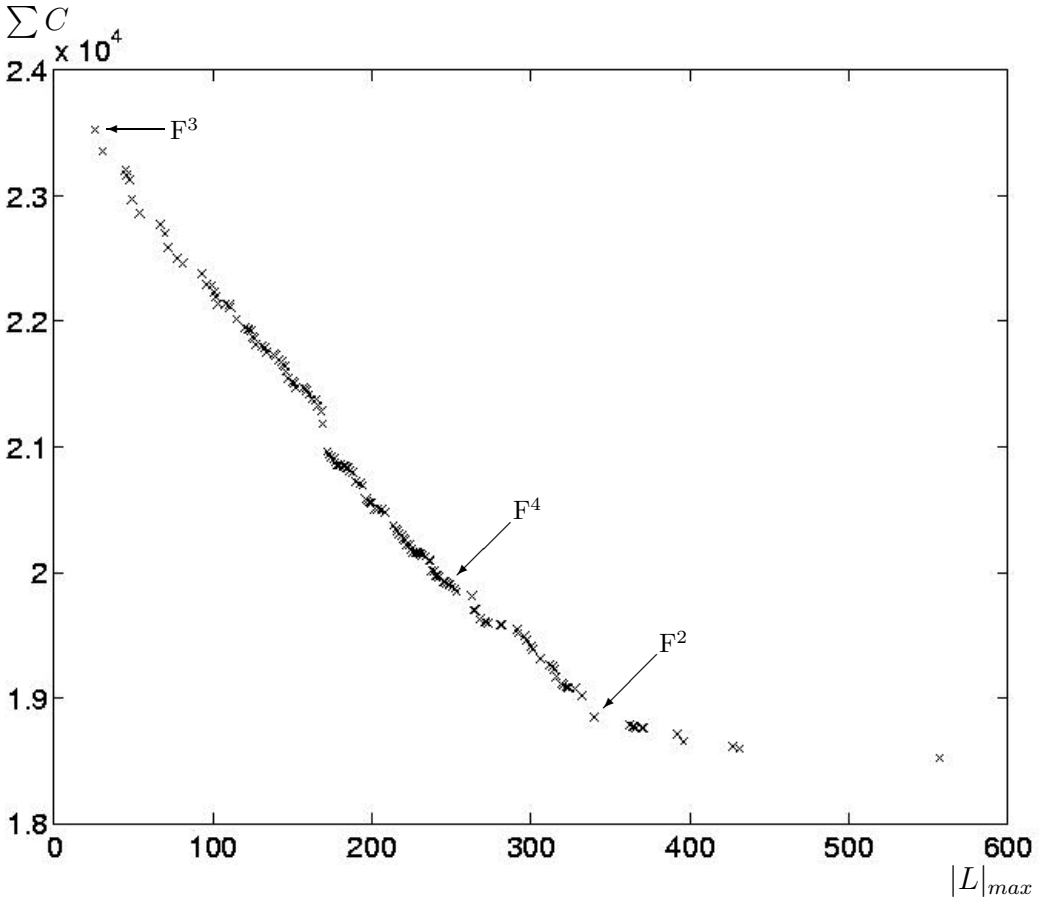

Abbildung 3.1: *Potentielle Pareto-optimale Lösungen der Instanz DRI41*

Beispiel:
Abbildung 3.1 zeigt eine Approximation der Menge der Pareto-optimalen Lösungen der Instanz DRI41 des Problems $D|resum, intree|\#(\sum C, |L|_{max})$. In dem Diagramm sind 150 verschiedene Pareto-optimale Lösungen eingezeichnet. Die durch Pfeile markierten Lösungen wurden durch die Ranking-Funktionen als beste Kompromisslösungen bestimmt. Die beiden Zielfunktionen waren in allen Fällen gleich gewichtet, d.h. es galt $w_1 = w_2 = \frac{1}{2}$. Da der Lösungspool in diesem Beispiel nur Pareto-optimale Lösungen enthielt, war die Ranking-Funktion F^1 nutzlos. Bei zwei gleich gewichteten Zielfunktionen hatten alle Lösungen den gleichen Ranking-Funktionswert.

Die große Anzahl von erzeugten Lösungen ist ein weiteres Problem für den Entscheider. Visualisierungen der Lösungen sind nur bedingt ein Ausweg. Die Daten der Lösungen einer Instanz mit mehreren hundert Vorgängen sind sehr umfangreich. Die Darstellung der Zielfunktionen oder der Struktur vieler Lösungen ist deshalb problematisch und wird dadurch erschwert, dass weitere Informationen für die Entscheidung benötigt werden. Dies können zusätzliche Zielfunktionen oder auch Soft Constraints sein.

Aus diesen Gründen wird eine Methode benötigt, um die große Anzahl der Lösungen auf wenige — 3 bis 6 Lösungen — zu reduzieren. Diese Vorauswahl sollte eine möglichst große Diversität im Lösungsraum gewährleisten, um dem Entscheider keine Lösungen zur Auswahl vorzulegen, die sich mit nur wenigen Nachbarschaftsschritten ineinander überführen lassen. Diese Lösungen würden keine echten Alternativen bieten.

Die multikriteriellen Lokale-Suche-Heuristiken in der GLS haben die Möglichkeit, eine Liste von potentiellen Pareto-Lösungen zurückzugeben. Diese Lösungen liegen im Regelfall sowohl im Raum der Lösungen als auch in Raum der Zielfunktionen dicht beieinander. Deshalb ist es ungünstig, die bezüglich der Ranking-Funktion ersten drei bis sechs Lösungen direkt als Vorauswahl zu verwenden.

Aussagen über die strukturelle Verschiedenheit von Lösungen können durch die in Abschnitt 2.3.1 eingeführten Abstandsmaße im Lösungsraum getroffen werden. Als Methode zur Reduktion der Anzahl der Lösungen bietet sich die Clusterung der Lösungen nach diesen Abstandsmaßen an. Aus den Clustern werden Lösungen ausgewählt, und die so entstandene Liste überschaubar weniger Lösungen wird dem Entscheider zur Auswahl vorgelegt. Der Vorgang der Reduktion vieler Vorschläge auf wenige Kandidaten wird auch als **Shortlisting** bezeichnet.

3.2.1 Clusteralgorithmen

Ziel der Clusterung ist es, eine Menge von n Lösungen $x_1, \ldots, x_n$ in c **Cluster** $C_1, \ldots, C_\mathrm{c}$ einzuteilen. Die Menge $\mathscr{C} = \{C_1, \ldots, C_\mathrm{c}\}$ wird als **Klassifikation** bezeichnet. Die Anzahl c der Cluster betrachten wir im Weiteren als vorgegeben. Dies ist die vom Entscheider gewünschte überschaubare Anzahl von Lösungen. Bei den hier vorgestellten Clusteralgorithmen handelt es sich um Algorithmen zur Erzeugung einer exhaustiven Partition. Das heißt, jede Lösung gehört genau einem Cluster an.

Für die Verfahren zur Konstruktion einer Partition werden Maße für die **Heterogenität** zweier disjunkter Cluster C_i und C_j benötigt. Die hier verwendeten **Heterogenitätsmaße** basieren auf den in Abschnitt 2.3.1 definierten Abstandsmaßen zwischen Lösungen. Die folgenden Heterogenitätsmaße werden verwendet:

$$
\begin{aligned}
v_c(C_i, C_j) &= \max_{x_k \in C_i, x_m \in C_j} A(x_k, x_m) && (\textit{complete linkage}),\\
v_s(C_i, C_j) &= \min_{x_k \in C_i, x_m \in C_j} A(x_k, x_m) && (\textit{single linkage}),\\
v_a(C_i, C_j) &= \frac{1}{|C_i| \cdot |C_j|} \sum_{x_k \in C_i} \sum_{x_m \in C_j} A(x_k, x_m) && (\textit{average linkage}).
\end{aligned}
$$

Für das Distanzmaß A gilt $A \in \{D, S^2, I, T, L\}$. Im ersten Fall wird die Verschiedenheit aufgrund des Lösungspaares mit dem größten Abstand und im zweiten Fall aufgrund des Lösungspaares mit dem kleinsten Abstand gemessen. Die Bezeichnungen **complete linkage** und **single linkage** sind hierbei üblich. Das dritte Maß für die Heterogenität zweier Cluster misst den durchschnittlichen Abstand der Lösungen aus den Clustern C_i und C_j. Die Bezeichnung hierfür ist **average linkage**.

Hierarchische Clusterung

Die hierarchischen Clusteralgorithmen werden in agglomerative und divisive Verfahren unterteilt. Im ersten Fall werden kleinere Klassen zu größeren generalisiert (bottom-up). Im zweiten Fall werden größere Klassen in kleinere unterteilt (top-down). Hier wurde das erste Verfahren implementiert.

Algorithmus 3.4 *Hierarchische Clusterung*
Seien c *die gewünschte Anzahl von Clustern,* $\mathscr{L} = \{x_1, \ldots, x_n\}$ *die Menge der Lösungen,* $A \in \{D, S^2, I, T, L\}$ *das verwendete Abstandsmaß und* v_α, $\alpha \in \{c, s, a\}$, *das verwendete Heterogenitätsmaß.*

1. *Initialisiere* n *Cluster;* $C_i := \{x_i\}$, $i = 1, \ldots, n$.
2. *Setze* $\mathrm{c}' := n$.
3. *Falls* $\mathrm{c}' > \mathrm{c}$
4. *Finde Cluster* C_i *und* C_j *mit dem kleinsten* $v_\alpha(C_i, C_j)$.
5. *Bilde die Vereinigung* $C_i \cup C_j$.

6. $c' := c' - 1$

7. *Gehe zu Schritt 3.*

8. *sonst gib* $\mathscr{C} = \{C_1, \ldots, C_c\}$ *aus. Stopp.*

Algorithmus 3.4 beginnt mit einer Anfangsklassifikation aus einelementigen Clustern bestehend aus den n Lösungen. Danach wird in der Menge der Cluster das Paar (C_i, C_j) gesucht, das bezüglich des verwendeten Abstands- und Heterogenitätsmaßes den geringsten Abstand hat. Diese beiden Cluster werden zusammengefasst. Der Algorithmus bricht ab, wenn $|\mathscr{C}| = c$ gilt. Die Laufzeit des Algorithmus ist $O(n^2)$.

Nicht-hierarchische Clusterung

Da die Anzahl c der Cluster vorgegeben ist, können nicht-hierarchische Clusterungsverfahren verwendet werden. Das hier implementierte Verfahren ist eine lokale Suche.

Algorithmus 3.5 *Nicht-hierarchische Clusterung*
Seien c *die Anzahl der Cluster,* $\mathscr{L} = \{x_1, \ldots, x_n\}$ *die Menge der Lösungen,* $A \in \{D, S^2, I, T, L\}$ *das verwendete Abstandsmaß und* v_α, $\alpha \in \{c, s, a\}$, *das verwendete Heterogenitätsmaß.*

1. *Initialisiere* c *Cluster;* $C_i := \emptyset$, $i = 1, \ldots, c$.

2. *Für* $i = 1, \ldots, n$

3. $\mathscr{L} = \mathscr{L} \setminus \{x_i\}$.

4. *Wähle zufällig gemäß Gleichverteilung einen Cluster* C' *aus* $\mathscr{C} = \{C_1, \ldots, C_c\}$.

5. *Setze* $C' := C' \cup \{x_i\}$.

6. *Wähle zufällig eine Permutation* σ *aus* S_n.

7. *Für* $i = 1, \ldots, n$

8. *Entferne die Lösung* $x_{\sigma(i)}$ *aus zugehörigem Cluster.*

9. *Berechne* $v_\alpha(\{x_{\sigma(i)}\}, C_j)$ *für* $j = 1, \ldots, c$.

10. *Füge $x_{\sigma(i)}$ in den Cluster mit minimalem v_α ein.*
11. *Falls mindestens eine Lösung einem anderen Cluster zugeordnet wurde, gehe zu Schritt 6.*
12. *Sonst gib $\{C_1, \ldots, C_c\}$ aus. Stopp.*

Der Algorithmus 3.5 beginnt mit einer zufällig erzeugten Partition mit c Clustern. Dann werden die n Lösungen in zufälliger Reihenfolge durchlaufen. Jede Lösung x_i wird aus dem zugehörigen Cluster entfernt und bildet einen eigenen Cluster $\{x_i\}$. Es werden die Heterogenitätsmaße zwischen $\{x_i\}$ und allen Clustern berechnet und x_i dem zum minimalen Abstand gehörenden Cluster zugeordnet. Der Algorithmus bricht ab, falls in einem Durchlauf keine Lösung den Cluster wechselte.

Um Klassifikationen vergleichen zu können, werden **Gütemaße** für Klassifikationen benötigt. In das Gütemaß sollen dabei nicht nur die Heterogenität zwischen Clustern, sondern auch die **Homogenität** innerhalb der Cluster eingehen. Die **Homogenitätsmaße** h werden analog zu den Heterogenitätsmaßen definiert:

$$
\begin{aligned}
h_c(C_i) &= \max_{x_k, x_m \in C_i} A(x_k, x_m), \\
h_s(C_i) &= \min_{\substack{x_k, x_m \in C_i \\ x_k \neq x_m}} A(x_k, x_m), \\
h_a(C_i) &= \begin{cases} 0, & für\ |C_i| = 1 \\ \frac{1}{|C_i| \cdot (|C_i|-1)} \displaystyle\sum_{\substack{x_k, x_m \in C_i \\ x_k \neq x_m}} A(x_k, x_m), & für\ |C_i| > 1 \end{cases}
\end{aligned}
$$

Mit den definierten Heterogenitäts- und Homogenitätsmaßen können mehrere Gütemaße definiert werden. Ein Vergleich der Güte von Klassifikationen ist natürlich nur sinnvoll, wenn für alle Klassifikationen das gleiche Gütemaß ermittelt wird. Das hier verwendete Gütemaß $g(\mathscr{C})$ wird folgendermaßen definiert:

$$
g(\mathscr{C}) = \frac{(|\mathscr{C}| - 1) \displaystyle\sum_{C_i \in \mathscr{C}} h_a(C_i)}{\displaystyle\sum_{\substack{C_i, C_j \in \mathscr{C} \\ C_i \neq C_j}} v_a(C_i, C_j)}
$$

Um die Güte der durch die beiden Algorithmen erzeugten Klassifikationen zu vergleichen, wurden wieder die ersten 60 Benchmark-Instanzen aus Tabelle A.1 verwendet. Für jede Instanz wurde mittels der multikriteriellen geneti-

schen lokalen Suche mit Sidestep-Algorithmus die Menge der nichtdominierten Lösungen für vier Zielfunktionskombinationen approximiert. Die Kombinationen waren $\#(C_{max}, \sum C)$, $\#(\sum C, \sum T, |L|_{max})$, $\#(C_{max}, \sum C, |L|_{max})$ und $\#(C_{max}, \sum C, \sum T, |L|_{max})$. Die Poolgröße war dabei auf 50 Lösungen festgelegt. Zur Ermittlung der Klassifikation wurden alle 50 Lösungen verwendet. Da die GLS nicht für jede Instanz und Zielfunktionskombination 50 Pareto-Lösungen erzeugte, wurden auch die im Pool enthaltenen dominierten Lösungen bei der Klassifikation berücksichtigt. In der Tabelle 3.5 sind die

	Hierarchisch					
	$D(\alpha,\beta)$			$T(\alpha,\beta)$		
Zielfunktionen	v_s	v_c	v_a	v_s	v_c	v_a
$\#(C_{max}, \sum C)$	0.66	1.06	0.82	0.69	1.18	0.93
$\#(\sum C, \sum T, \|L\|_{max})$	0.58	0.87	0.75	0.67	0.99	0.85
$\#(C_{max}, \sum C, \|L\|_{max})$	0.66	0.91	0.77	0.72	1.04	0.86
$\#(C_{max}, \sum C, \sum T, \|L\|_{max})$	0.60	0.87	0.67	0.67	1.00	0.77
	Nicht-hierarchisch					
	$D(\alpha,\beta)$			$T(\alpha,\beta)$		
Zielfunktionen	v_s	v_c	v_a	v_s	v_c	v_a
$\#(C_{max}, \sum C)$	1.76	1.26	1.26	1.78	1.48	1.40
$\#(\sum C, \sum T, \|L\|_{max})$	1.75	1.10	1.09	1.77	1.29	1.28
$\#(C_{max}, \sum C, \|L\|_{max})$	1.75	1.12	1.07	1.75	1.31	1.25
$\#(C_{max}, \sum C, \sum T, \|L\|_{max})$	1.74	1.07	1.08	1.73	1.27	1.22

Tabelle 3.5: *Gütemaße bei verschiedenen Heterogenitäts- und Abstandsmaßen für* c = 3

durchschnittlichen Werte des Gütemaßes der durch die beiden Clusterungsalgorithmen ermittelten Klassifikationen verzeichnet. Die Anzahl der Cluster war hierbei in allen Fällen auf c = 3 festgelegt. In den Clusterungsalgorithmen können verschiedenen Heterogenitäts- und Abstandsmaße verwendet werden. Um deren Einfluss zu untersuchen, wurden alle drei Heterogenitätsmaße und die zwei Abstandsmaße $D(\alpha,\beta)$ und $T(\alpha,\beta)$ verwendet. Somit wurden zu jedem Pool mit 50 Lösungen 12 Klassifikationen konstruiert. Die in der Tabelle eingetragenen Werte geben den Durchschnitt des Gütemaßes der Klassifikationen über die 60 Instanzen zu jeder Zielfunktionskombinationen an.

Die durch das hierarchische Clusterverfahren ermittelten Klassifikationen erreichen bei allen vier Zielfunktionskombinationen, zwei Abstandsmaßen und drei Heterogenitätsmaßen bessere Werte für das Gütemaß als das nicht-hier-

archische Verfahren. Die Wahl des Abstandsmaßes hat nur einen geringen Einfluss auf die Güte der Klassifikation. Bei den drei Heterogenitätsmaßen sind die Unterschiede in der Güte der Klassifikationen deutlicher. Das Maß v_s erreicht beim hierarchischen Clusterverfahren die beste Güte und beim nicht-hierarchischen die schlechteste. Beim Vergleich der Maße v_c und v_a erreichen die Verfahren mit v_a die etwas besseren Ergebnisse. Dieses Verhältnis ändert sich nicht, falls in der Berechnung des Gütemaßes statt h_a und v_a h_c und v_c verwendet wird. Bei Versuchen mit den restlichen Abstandsmaßen und anderen Clusteranzahlen c konnten die Beobachtungen bestätigt werden.

Neben der Güte der Klassifikation sind weitere Eigenschaften der Verfahren von Bedeutung. Zum einen die Laufzeit der Algorithmen, da die Auswahl der Lösung aus dem vorhandenen Pool durch den Entscheider interaktiv erfolgt. Hier hat das nicht-hierarchische Verfahren Vorteile, da die Antwortzeiten deutlich geringer sind. Bei einer Anzahl von 600 Vorgängen in der Instanz und einer Poolgröße von 50 Lösungen, ist das nicht-hierarchische Verfahren um etwa den Faktor 4 schneller. Zum anderen ist die Größe der Cluster von Bedeutung, falls der Entscheider die präferierte Lösung mit anderen ähnlich strukturierten Lösungen vergleichen möchte. In dem Fall sind Verfahren, die mehrere Cluster mit einer Lösung und einen Cluster mit den restlichen Lösungen liefern, ungünstig. Aus diesem Grund wurden die Clustergrößen der durch die 12 Varianten erstellten Klassifikationen ermittelt. Die Ergebnisse sind in Abbildung A.1 im Anhang zu finden. Dort ist ersichtlich, dass das hierarchische Clusterverfahren zur Bildung kleiner Cluster neigt. Vor allem bei der Verwendung des Homogenitätsmaßes v_s treten häufig Cluster mit nur einer Lösung auf. Dies erklärt auch die guten Werte für das Gütemaß, da bei Clustern C mit nur einer Lösung $h_a(C) = 0$ gilt.

Ziel der Clusterung der Lösungsmenge ist die Reduktion der zur Auswahl stehenden Lösungen. Dazu wurde aus jedem Cluster eine Lösung ausgewählt. Es wurden die folgenden zwei Methoden der Auswahl untersucht:

1. In jedem Cluster wurden die Lösungen nach der Ranking-Funktion sortiert und jeweils die Lösung auf dem ersten Rang selektiert. Dabei wurde die Ranking-Funktion F^2 verwendet und alle Zielfunktionen hatten die gleiche Gewichtung.

2. Aus jedem Cluster wird die Medianlösung selektiert, d.h. die Lösung die den minimalen durchschnittlichen Abstand zu den anderen Lösungen im Cluster hat.

Zur Untersuchung dieser beiden Varianten wurden die oben ermittelten Klassifikationen verwendet. Damit wurde aus jeder Klassifikation drei Lösungen ausgewählt. Es wurde der durchschnittliche Abstand zwischen den Lösungen und der minimale Abstand zwischen je zwei der drei Lösungen ermittelt. Diese beiden Abstände wurden mit dem durchschnittlichen Abstand zwischen allen Lösungen im Pool normiert, um die Vergleichbarkeit bei verschiedenen Instanzen zu gewährleisten. Tabelle 3.6 gibt die so ermittelten

		Hierarchisch					
		$D(\alpha,\beta)$			$T(\alpha,\beta)$		
Auswahl		v_s	v_c	v_a	v_s	v_c	v_a
Lösung mit	ave	1.43	1.39	1.44	1.31	1.27	1.31
Rangzahl 1	min	1.18	1.11	1.17	1.14	1.07	1.13
Median-	ave	1.38	1.32	1.38	1.28	1.23	1.28
lösung	min	1.14	1.02	1.14	1.10	1.01	1.11
		Nicht-hierarchisch					
		$D(\alpha,\beta)$			$T(\alpha,\beta)$		
Auswahl		v_s	v_c	v_a	v_s	v_c	v_a
Lösung mit	ave	1.01	1.29	1.26	1.02	1.15	1.16
Rangzahl 1	min	0.79	0.94	1.02	0.81	0.85	0.97
Median-	ave	0.87	1.26	1.22	0.88	1.13	1.11
lösung	min	0.71	0.90	0.97	0.74	0.86	0.90

Tabelle 3.6: *Abstände zwischen den ausgewählten Lösungen bei der Zielfunktionskombination* $\#(C_{max}, \sum C)$ *und drei Clustern*

durchschnittlichen relativen Abstände der Lösungen bei den Klassifikationen für die Zielfunktionskombination $\#(C_{max}, \sum C)$ und den 60 Instanzen an. Zum Vergleich wurden auch die Abstände zwischen den nach der Ranking-Funktion ersten drei Lösungen im gesamten Pool ermittelt. Die Werte für den durchschnittlichen Abstand und für den minimalen Abstand unter Verwendung von $D(\alpha,\beta)$ waren 0.67 und 0.23. Die entsprechenden Werte unter Verwendung von $T(\alpha,\beta)$ waren 0.66 und 0.25. Die Daten in Tabelle 3.6 lassen folgende Beobachtungen zu:

1. Die durch das hierarchische Clusterverfahren erzeugten Klassifikationen liefern größere Abstände zwischen den ausgewählten Lösungen als die durch das nicht-hierarchische Verfahren erzeugten Klassifikationen. Dies gilt für beide Auswahlverfahren.

2. Die Abstände zwischen den Lösungen sind bei Auswahl der jeweils ersten Lösung im Cluster größer als bei der Auswahl der Medianlösung. Dies gilt wiederum für beide Klassifikationsalgorithmen.

3. Die Wahl des Abstandsmaßes hat nur einen geringen Einfluss auf die relativen Abstände zwischen den ausgewählten Lösungen.

4. Das Heterogenitätsmaß v_s liefert unter Verwendung des nicht-hierarchischen Verfahrens die schlechtesten Ergebnisse.

Punkt 1 deckt sich mit der Beobachtung, dass die durch das hierarchische Verfahren erzeugten Klassifikationen bessere Werte für das Gütemaß erreichen. Da die Wahl des Abstandsmaße sowohl bei den Korrelationskoeffizieten als auch bei den Gütemaßen nur einen geringen Einfluss hatte, war auch Punkt 3 zu erwarten. Der Punkt 4 wiederum deckt sich mit der Beobachtung, dass das Heterogenitätsmaß v_s in Kombination mit Algorithmus 3.5 die schlechtesten Werte für das Gütemaß ausweist.

Nach diesen Daten ist es bei beiden Clusteralgorithmen am günstigsten, die Lösungen auf dem ersten Rang aus jedem Cluster zu wählen. Dabei wird im Vergleich mit der zweiten Auswahlmethode die größte Diversität zwischen den ausgewählten Lösungen erreicht. Weiterhin werden dadurch die Präferenzen des Entscheider berücksichtigt. Um dies näher zu untersuchen, wurden die durchschnittlichen Rangzahlen der ausgewählten Lösungen im

	Hierarchisch					
	$D(\alpha,\beta)$			$T(\alpha,\beta)$		
Auswahl	v_s	v_c	v_a	v_s	v_c	v_a
Beste Lösung	17.58	11.04	14.72	18.20	11.20	14.78
Medianlösung	26.32	24.78	25.21	26.17	24.11	24.67
	Nicht-hierarchisch					
	$D(\alpha,\beta)$			$T(\alpha,\beta)$		
Auswahl	v_s	v_c	v_a	v_s	v_c	v_a
Beste Lösung	6.11	8.71	8.17	5.97	7.41	8.36
Medianlösung	21.81	24.41	24.41	23.74	23.59	23.71

Tabelle 3.7: *Durchschnittliche Rangzahl der ausgewählten Lösungen bei der Zielfunktionskombination* $\#(C_{max}, \sum C)$ *und Ranking-Funktion* F^2

Ausgangspool über alle 60 Instanzen ermittelt. Tabelle 3.7 enthält die zugehörigen Daten. Die mit Methode 1 gewählten Lösungen erreichen die kleineren Rangzahlen. Besonders gute Werte erreicht dabei Algorithmus 3.5.

Diese Versuche wurden mit den anderen drei Zielfunktionskombinationen, Abstandsmaßen, Ranking-Funktionen und mit den Clusteranzahlen c = 4 und c = 5 wiederholt. Die dabei erhaltenen Daten bestätigten die Beobachtungen.

Es bleibt festzuhalten, dass die Vorteile des hierarchischen Verfahrens in der besseren Güte der Klassifikationen und in der größeren Diversität der ausgewählten Lösungen liegen. Die Vorteile des nicht-hierarchischen Algorithmus liegen in der kürzeren Laufzeit, der ungefähr gleichen Größe der Cluster und der besseren Berücksichtigung der Entscheiderpräferenzen. Bei beiden Algorithmen erreichten die Heterogenitätsmaße v_c und v_a sowie die Auswahlmethode 1 die besten Ergebnisse.

Fazit, Ausblick, offene Fragen

Dieser Abschnitt fasst die Beobachtungen und Ergebnisse der Arbeit zusammen. Außerdem werden offene Fragen und Ideen für die zukünftige Forschung erörtert.

Summarium

Die in der Arbeit entwickelte genetische lokale Suche liefert für Job-Shop Scheduling-Probleme mit verschiedenen Verallgemeinerungen und Zielfunktionen gute Ergebnisse in relativ kurzer Zeit. Dies wurde durch die Verwendung großer Nachbarschaften und der Mittelwertbildung als Rekombinationsoperator erreicht. Die großen Nachbarschaften mit einem hohen Anteil von Nachbarlösungen mit gleichem Funktionswert ermöglichen es den Lokale-Suche-Heuristiken, dem Einzugsgebiet schwacher lokaler Optima zu entkommen. Diese Struktur der Nachbarschaften war unabhängig von den vier verwendeten Zielfunktionen C_{max}, $\sum C$, $\sum T$ und $|L|_{max}$ sowie den Verallgemeinerungen in den praktischen Job-Shop Scheduling-Problemen. Die als Rekombinationsoperator verwendete Mittelwertbildung nutzt die Big-Valley-Struktur des Lösungsraumes aus. Durch Mittelwertbildung aus lokalen Optima erzeugte Nachkommen liegen im Durchschnitt näher am globalen Optimum als die Ausgangslösungen. Dies führt zur Konvergenz der Zielfunktionswerte in der genetischen lokalen Suche.

Im Rahmen der Dissertation wurden Versuche zur Untersuchung der Landschaft im Lösungsraum durchgeführt. Bei allen vier genannten Zielfunktionen wurde die Big-Valley-Struktur im Lösungsraum experimentell verifiziert. Dabei stellte sich heraus, dass die Zielfunktionswerte und die Abstände zum besten lokalen Optimum bei den Zielfunktionen C_{max} und $|L|_{max}$ stärker positiv korreliert sind als bei den Zielfunktionen $\sum C$ und $\sum T$. Gleiches gilt für den Vergleich der $J||f$-Instanzen mit den $J|2SETS|f$-Instanzen. Letz-

tere erreichten bei allen Zielfunktionen geringere Korrelationskoeffizienten. Bemerkenswert ist, dass die beobachteten Strukturen unabhängig vom verwendeten Abstandsmaß sind.

Diese Beobachtungen und Vermutungen ergeben sich auf den durch umfangreichen Computereinsatz erzeugten Fakten. Dabei wurde eine große Anzahl von Benchmark-Instanzen verwendet. 301 Instanzen stammen aus verschiedenen Literaturquellen. 70 Instanzen wurden für die PJSP generiert. Hervorzuheben sind die durch die Optimierung mit der GLS verbesserten oberen Schranken für die Instanzen der Probleme $J||C_{max}$ und $J|| \, |L|_{max}$. Bei 78 von 241 $J||C_{max}$-Instanzen wurden die oberen Schranken verbessert und bei weiteren 140 Instanzen Lösungen für die bekannten oberen Schranken gefunden. Dabei wurde mittels der GLS bei 6 Instanzen obere Schranken gefunden, die mit den unteren übereinstimmen. Somit sind nunmehr 135 der 241 Instanzen gelöst. Für alle der 50 $J|| \, |L|_{max}$-Instanzen wurden bessere obere Schranken gefunden.

Ein Vorteil der GLS ist die Unabhängigkeit vom konkreten Scheduling-Problemtyp. Die für Heuristiken geringe Anzahl von Algorithmenparametern ist ein weiterer Vorteil speziell der GLS mit Sidestep-Algorithmus. Die Versuche führten zu dem Resultat, dass die Nachbarschaft N_1 und die Elternzahl 2 im Vergleich die besten Ergebnisse lieferten. Die beiden anderen Parameter für den Algorithmus sind die Anzahl der Lösungen im Pool PG und die Anzahl der zulässigen Sidesteps IS. Die Poolgröße PG hat Einfluss auf die Diversität im Pool und die Konvergenzgeschwindigkeit. Je größer der Pool, desto langsamer die Konvergenz und desto besser sind die erreichten lokalen Optima. Die Poolgröße lag bei den Läufen zur Suche verbesserter oberer Schranken zwischen 50 und 100 Lösungen. Ähnliche Aussagen gelten für die Anzahl der Sidesteps. Größere Anzahlen führen zu besseren Zielfunktionswerten und längeren Laufzeiten. Als Basis für die Anwendung der Heuristiken kann Algorithmus 2.16 dienen. Im Vergleich mit anderen Heuristiken ist der Aufwand für das Parametertuning der GLS gering.

Die Flexibilität der GLS erzeugt Nachteile im Laufzeitverhalten. So erreicht die speziell für das Problem $J||C_{max}$ entwickelte Tabu-Suche von Nowicki und Smutnicki Lösungen mit sehr guten oberen Schranken in kürzerer Zeit. Auch die Shifting-Bottleneck-Prozedur für das $J|| \, |L|_{max}$ benötigt weniger Rechenzeit, erreicht aber vergleichsweise schlechtere lokale Optima.

Des Weiteren wird die genetische lokale Suche durch die Einführung von erneuerbaren Ressourcen verlangsamt, da Algorithmus 2.4 zur Ermittlung der frühesten Start- und Endtermine hier $O(N^2)$ Schritte im worst case benötigt.

Das Hauptproblem ist, dass die Vorgänge mehr als eine Mengeneinheit an Ressourcen benötigen können. Ein Ausweg besteht darin, sicherzustellen, dass jeder Vorgang von jeder Ressource höchstens eine Mengeneinheit belegen kann. Dies würde den Aufwand zur Berechnung der frühesten Start- und Endtermine auf $O(N)$ reduzieren und in der Praxis keine zu große Einschränkung bedeuten.

Durch ihre Flexibilität kann die GLS auf multikriterielle Scheduling-Probleme erweitert werden. Für die fitnessabhängige Selektion und die Suchrichtung in den multikriteriellen Lokale-Suche-Heuristiken wurden Ranking-Funktionen verwendet. Die in der Arbeit vorgestellen Schwellwert-Algorithmen wurden für die multikriterielle Optimierung erweitert. Damit ist es möglich, Zielfunktionen lexikographisch zu optimieren oder die Menge der Pareto-optimalen Lösungen zu approximieren. In den Versuchen erwies sich die GLS mit multikriteriellem Sidestep-Algorithmus im Vergleich als die beste Variante der multikriteriellen Hybrid-Algorithmen.

Ausblick und offene Fragen

Durch die Computer-Experimente bedingt, können nicht alle Fragestellungen allgemein erfasst werden. Mit jeder Beobachtung entstehen neue Fragen. Im Folgenden wird eine Auswahl von offenen Fragen, Ideen und Ansätze für die weitere Forschung angegeben:

1. Nicht alle praxisrelevanten Verallgemeinerungen konnten betrachtet werden. Als weitere Verallgemeinerungen können parallele Maschinen, Rüstzeiten, Multi-Mode Job-Shops und allgemeinere Reihenfolgebeziehungen eingeführt werden. Wie gut ist die genetische lokale Suche bei diesen Erweiterungen?

2. Die Big-Valley-Struktur wurde bei vier Zielfunktionen untersucht. Wie ist der Lösungsraum bei weiteren Zielfunktionen, wie der Summe der Durchlaufzeiten oder der Earliness strukturiert? Welche Resultate erzielt die GLS dort? Die untersuchten Korrelationen und die Big-Valley-Struktur geben nur ein grobes Bild der Lösungsraumstruktur an. Wie können die Strukturerkenntnisse präzisiert werden?

3. Liefert die genetische lokale Suche auch bei Problemen gute Resutate, bei denen keine Big-Valley-Struktur nachweisbar ist?

4. Die einzelnen Läufe der Lokale-Suche-Algorithmen sind unabhängig voneinander. Wie kann die GLS parallelisiert werden?

5. Die GLS wird nach dem Erreichen einer Zeitschranke oder einer maximalen Anzahl von Läufen der Lokale-Suche-Heuristiken abgebrochen. Wie können die Abstandsmaße und die Diversität im Lösungspool für Abbruchkriterien verwendet werden?

6. Wie wirkt sich der Einsatz anderer Algorithmen zur Startlösungserzeugung — z.B. Prioritätsregeln, Ameisensysteme oder Greedy Algorithmen — auf die GLS aus?

7. Für die Probleme $J||C_{max}$ und $J||L_{max}$ existiert eine große Anzahl von Optimierungsalgorithmen und Benchmark-Instanzen mit ihren unteren und oberen Schranken. Bei anderen Zielfunktionen und verallgemeinerten Problemen ist dies nicht der Fall. Hierfür ist eine größere Anzahl von Optimierungsalgorithmen und Benchmark-Instanzen wünschenswert.

8. Die Minimierung der Anzahl der verspäteten Aufträge $\sum U$ ist eine weitere praxisrelevante Zielfunktion. Da die Änderungen des Zielfunktionswertes innerhalb der Nachbarschaft problemgemäß sehr gering sind, stoßen Lokale-Suche-Heuristiken auf Probleme. Ein Ausweg ist die lexikographische Optimierung mit einer zweiten Zielfunktion wie zum Beispiel L_{max} oder $\sum T$. Welche Kombination ist hier am günstigsten?

9. Wie sind die Lösungsräume bei der lexikographischen Optimierung strukturiert? Die Korrelationen können mit dem Rangkorrelationskoeffizienten von Spearman ermittelt werden.

10. Approximationen der Menge der Pareto-optimalen Lösungen wurden in der Arbeit mittels Outperformance-Relationen verglichen. Welche Beobachtungen lassen sich bei quantitativen Vergleichsmethoden machen?

11. Die Dissertation stellt mehrere Algorithmen zur multikriteriellen Optimierung, mehrere Ranking-Funktionen und Auswahlverfahren zur Verfügung. Wie lassen sich diese in einem 3-Hirn-System einsetzen [2]?

Anhang A

Tabellen und Abbildungen

A.1 Auswahl von klassischen Job-Shop-Benchmark-Instanzen

10x10	FT10	ABZ 5	ABZ 6	LA 16	LA 17	LA 18	ORB 1	ORB 2	ORB 3	ORB 4
15x15	TA 1	TA 2	TA 3	TA 4	TA 5	TA 6	TA 7	TA 8	TA 9	TA10
20x15	ABZ 7	SWV 6	SWV 7	TA11	TA13	DMU 1	DMU 2	DMU41	DMU44	DMU45
20x20	YN 1	YN 2	TA21	TA22	TA23	DMU 7	DMU 6	DMU10	DMU48	DMU49
30x15	TA31	TA32	TA33	TA34	DMU12	DMU14	DMU15	DMU51	DMU52	DMU53
30x20	TA41	TA42	TA43	TA44	DMU16	DMU17	DMU20	DMU56	DMU57	DMU58
40x15	DMU21	DMU22	DMU23	DMU24	DMU25	DMU61	DMU62	DMU63	DMU64	DMU65
40x20	DMU26	DMU27	DMU28	DMU29	DMU30	DMU66	DMU67	DMU68	DMU69	DMU70
50x10	SWV11	SWV12	SWV13	SWV14	SWV15	SWV16	SWV17	SWV18	SWV19	SWV20
50x15	TA51	TA52	TA53	TA54	DMU31	DMU32	DMU33	DMU71	DMU72	DMU73
50x20	TA61	TA62	TA63	TA64	DMU36	DMU37	DMU38	DMU76	DMU77	DMU78

Tabelle A.1: *Testmenge der Benchmark-Instanzen für klassische Job-Shop-Probleme*

A.2 Nachbarschaftszahlen und Plateaus

Die Tabelle A.2 zeigt die Anzahlen der Nachbarn für alle 70 Instanzen der praktischen Job-Shop-Probleme. Die erste Spalte enthält die Nummern der Instanzengruppen zu je 5 Benchmarks. Diese Untergruppen sind nach ihren Generierungsparametern für die Anzahl der Aufträge und Maschinen zusammengefasst. Die Spalten zwei bis sechs enthalten die durchschnittlichen Anzahlen der Nachbarn. Für jede Untergruppe sind in der Tabelle zwei Zeilen enthalten. In der ersten Zeile pro Untergruppe stehen die durchschnittlichen Anzahlen zulässiger Nachbarn. In der zweiten Zeile stehen die durchschnittlichen Anzahlen zulässiger Nachbarn mit gleichem Zielfunktionswert für die Zielfunktion C_{max}. Als Lösungen wurden lokale Optima bezüglich C_{max} verwendet. Zu jeder Instanz wurden 10 lokale Optima erzeugt. Die letzte Spalte gibt die Anzahl der Vorgänge in den Instanzen an. Bei den Instanzen DRI41-55 und DRIR56-70 müssen die Jobs nicht auf allen Maschinen bearbeitet werden. Aus diesem Grund wurden bei gleichen Auftrags- und Maschinenanzahlen verschiedene Anzahlen von Vorgängen erzeugt. Die in der Tabelle angegebenen Anzahlen von Tasks sind somit als durchschnittliche Anzahlen aufzufassen.

Nr.	N_1	N_2	N_3	N_4	N_5	Tasks
1-5	3872.90	8474.60	6632.84	14036.78	11397.08	300
	2829.00	6898.62	5431.02	10326.94	7927.76	
6-10	11339.18	25711.84	19948.04	43446.34	35089.16	600
	8922.26	21922.04	17244.98	34166.88	26217.46	
11-15	20445.02	45997.88	35937.80	77989.62	62839.54	800
	16325.70	39678.66	31373.92	62264.28	47793.42	
16-20	49367.90	113443.38	87694.78	194373.16	156698.98	1500
	41629.28	101877.52	79122.18	164063.98	127537.46	
21-25	3888.96	8465.80	6673.78	14078.50	11415.90	300
	2976.82	7079.68	5639.84	10768.06	8321.18	
26-30	11666.56	26189.08	20394.16	44281.68	35804.66	600
	9484.56	22809.56	17956.78	35930.20	27832.20	
31-35	20635.14	46966.78	36169.72	79011.18	63816.26	800
	16856.42	41083.72	32066.76	64491.78	49723.00	
36-40	48957.62	113359.24	87163.82	193888.12	156328.08	1500
	42429.32	103556.56	80044.92	168229.58	131228.48	
41-45	4035.84	9900.10	6798.82	15514.38	12927.74	328.00
	3107.64	8361.90	5762.04	12118.74	9428.64	
46-50	13406.44	33036.78	23139.80	52806.86	43434.00	659.80
	10520.38	28126.42	19926.96	41603.08	32243.54	
51-55	11788.92	30957.10	20348.98	48735.82	40525.80	704.00
	9652.18	27136.62	18039.64	40274.50	31532.02	
56-60	4286.96	10470.50	7277.20	2207.12	1868.50	364.20
	2962.30	8112.96	5750.48	1353.94	1058.74	
61-65	10627.32	26194.94	18192.86	5289.18	4424.78	588.60
	7104.92	20186.14	14119.06	3186.92	2432.06	
66-70	12481.26	32020.70	21611.26	6942.72	5796.62	709.20
	7924.76	23769.86	16065.00	3961.02	2998.06	

Tabelle A.2: *Durchschnittliche Anzahl von zulässigen Nachbarn und Nachbarn mit gleichem Zielfunktionswert für C_{max} bei den 5 Nachbarschaftsfunktionen und allen praktischen Job-Shop-Benchmark-Instanzen*

A.3 Korrelationskoeffizienten der Instanzen für praktischen Job-Shop-Probleme

C_{max}	$D(\alpha,\beta)$					
	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Instanzen	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
D1-20	0.7110	0.8001	20	20	0.7098	0.7844
DR21-40	0.4570	0.6222	16	19	0.4403	0.5986
DRI41-55	0.6238	0.4898	12	11	0.6208	0.4754
DRIR56-70	0.4506	0.2580	11	6	0.4525	0.2404
	$T(\alpha,\beta)$					
$\lvert\mathcal{J}\rvert/\lvert\mathcal{M}\rvert$	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
D1-20	0.6997	0.8754	20	20	0.7091	0.8516
DR21-40	0.3965	0.6390	14	20	0.3811	0.6109
DRI41-55	0.5362	0.4845	10	10	0.5282	0.4704
DRIR56-70	0.4085	0.2575	10	5	0.4110	0.2388

Tabelle A.3: *Korrelationskoeffizienten und Signifikanzen bei praktischen Job-Shop-Problemen und C_{max}*

$\sum T$	$D(\alpha,\beta)$					
	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Instanzen	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
D1-5	0.3722	0.4508	4	5	0.3773	0.4466
DR21-25	0.3718	0.3246	5	4	0.3467	0.3124
DRI41-45	0.4723	0.5958	4	5	0.4729	0.5945
DRIR56-60	0.4754	0.5002	4	5	0.4856	0.4540
	$T(\alpha,\beta)$					
$\lvert\mathcal{J}\rvert/\lvert\mathcal{M}\rvert$	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
D1-5	0.2678	0.4988	3	5	0.2690	0.4753
DR21-25	0.2949	0.3831	3	4	0.2684	0.3675
DRI41-45	0.2908	0.5401	2	5	0.3462	0.5873
DRIR56-60	0.4043	0.5355	3	5	0.4161	0.5043

Tabelle A.4: *Korrelationskoeffizienten und Signifikanzen bei praktischen Job-Shop-Problemen und $\sum T$*

$\sum C$	$D(\alpha,\beta)$					
	Pearson				Spearman	
	Durchschnitt		k		Durchschnitt	
Instanzen	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
D1	0.4291	0.2836	0.0000	0.0024	0.4455	0.2928
DR21	0.5508	0.3493	0.0000	0.0001	0.3863	0.2306
DRI41	0.6021	0.5073	0.0000	0.0000	0.5053	0.4045
DRIR56	0.3087	0.3278	0.0012	0.0000	0.3264	0.3676
	$T(\alpha,\beta)$					
$\lvert\mathcal{J}\rvert/\lvert\mathcal{M}\rvert$	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
D1	0.3219	0.2652	0.0006	0.0040	0.3438	0.2652
DR21	0.3774	0.4764	0.0001	0.0000	0.3206	0.3869
DRI41	0.4020	0.4659	0.0000	0.0000	0.3092	0.3987
DRIR56	0.2720	0.3466	0.0046	0.0004	0.2511	0.3771

Tabelle A.5: *Korrelationskoeffizienten und Signifikanzen bei praktischen Job-Shop-Problemen und $\sum C$*

A.4 Benchmark-Instanzen für $J||C_{max}$

Die besten bekannten oberen und unteren Schranken bezüglich der Zielfunktion C_{max} sind in der Literatur über viele Veröffentlichungen gestreut. Für die anderen Zielfunktionen existieren wenige bis gar keine verfügbaren Daten. Um Vergleiche zu erleichtern, werden die Schranken aus der Literatur und die während der Arbeit ermittelten Schranken hier im Anhang angegeben.

In den folgenden Tabellen stehen jeweils in der ersten Spalte die Bezeichnungen der 241 verwendeten Instanzen. In den weiteren Spalten sind folgende Daten verzeichnet: die Anzahl der Jobs $|\mathcal{J}|$, die Anzahl der Maschinen $|\mathcal{M}|$, die beste bekannte untere Schranke für den Makespan C^{LB}_{max}, die Differenz zwischen der unteren und der besten bekannten oberen Schranke $\Delta^{(UB,LB)}_{C_{max}}$ in Prozent, die bisher beste bekannte obere Schranke C^{UB}_{max}, die Differenz zwischen der bisher besten bekannten oberen Schranke und der mit dem genetischen Algorithmus erzielten oberen Schranke $\Delta^{(GA,UB)}_{C_{max}}$ in Prozent, die mit dem genetischen Algorithmus erreichten oberen Schranken für den Makespan C^{GA}_{max}, für die totale Verspätung $\sum T^{GA}_j$, für die Summe der Fertigstellungszeiten $\sum C^{GA}_j$, für die Summe der Durchlaufzeiten $\sum F^{GA}_j$ und für die maximale absolute Terminabweichung $|L|^{GA}_{max}$. Zur Berechnung der oberen Schranken der Zielfunktionen $\sum T_j$, $\sum C_j$, $\sum F_j$ und $|L|_{max}$ wurde die

$Z(x^1, x^2)$		Pearson				Spearman	
		Durchschnitt		k		Durchschnitt	
Zielfunktion	Instanzen	ρ^P_{max}	ρ^P_{av}	ρ^P_{max}	ρ^P_{av}	ρ^S_{max}	ρ^S_{av}
C_{max}	D1-20	0.9679	0.8903	20	20	0.9583	0.7556
	D21-40	0.9145	0.7801	20	20	0.9040	0.7431
	D41-55	0.8578	0.6617	15	15	0.8558	0.5960
	D56-70	0.7749	0.5015	15	15	0.7762	0.4310
$\sum C$	D1-20	0.5953	0.5003	19	18	0.5228	0.3329
	D21-40	0.4268	0.3203	16	10	0.4015	0.2652
	D41-55	0.5737	0.4243	15	14	0.5511	0.3787
	D56-70	0.6666	0.4891	14	14	0.6440	0.4367
$\sum T$	D1-20	0.7025	0.6489	20	20	0.6467	0.4872
	D21-40	0.5251	0.4473	20	18	0.5053	0.3973
	D41-55	0.5031	0.5100	14	14	0.4950	0.4978
	D56-70	0.6942	0.5322	15	15	0.6858	0.4787
$\|L\|_{max}$	D1-20	0.9637	0.8868	20	20	0.9571	0.7705
	D21-40	0.8994	0.7356	20	20	0.8931	0.6979
	D41-55	0.8079	0.6636	15	15	0.8266	0.6509
	D56-70	0.7279	0.5313	15	15	0.7369	0.5131

Tabelle A.6: *Durchschnitt der Korrelationskoeffizienten und Signifikanzen bei praktischen Job-Shop-Problemen mit dem Abstandsmaß $Z(x^1, x^2)$*

genetische lokale Suche mit Sidestep-Algorithmus verwendet. Der Algorithmus wurde bei Instanzen bis 500 Tasks nach 1000 Sekunden und bei größeren Instanzen nach 2000 Sekunden abgebrochen.

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
FT 10	10	10	930	0.00	930	0.00	930	281	7501	5453	135
FT 20	20	5	1165	0.00	1165	0.00	1165	6546	13842	5331	683
SWV 1	20	10	1407	0.00	1407	+0.36	1412	5543	19750	10680	538
SWV 2	20	10	1475	0.00	1475	0.00	1475	6313	21066	10882	633
SWV 3	20	10	1398	0.00	1398	0.00	1398	6583	21507	10802	680
SWV 4	20	10	1450	+2.21	1483	**-0.07**	1482	7429	21809	11141	697
SWV 5	20	10	1424	0.00	1424	+0.14	1426	6761	21037	10807	662
SWV 6	20	15	1591	+5.47	1678	+0.18	1681	7174	28346	20779	569
SWV 7	20	15	1447	+11.47	1620	**-0.43**	1613	7117	26345	22182	566
SWV 8	20	15	1641	+7.19	1763	**-0.23**	1759	7102	28762	22906	567
SWV 9	20	15	1605	+3.55	1663	**-0.06**	1662	7041	28309	21829	625
SWV 10	20	15	1632	+7.90	1767	**-0.34**	1761	7639	28763	20778	647
SWV 11	50	10	2983	+0.17	2991	**-0.10**	2988	67114	101922	79040	2354
SWV 12	50	10	2972	+1.04	3003	+0.16	3008	66752	103071	83128	2560
SWV 13	50	10	3104	0.00	3104	0.00	3104	70064	107506	82414	2492
SWV 14	50	10	2968	0.00	2968	0.00	2968	66855	100184	80007	2225
SWV 15	50	10	2885	+0.62	2904	**-0.06**	2903	68061	98323	76196	2344
SWV 16	50	10	2924	0.00	2924	0.00	2924	63152	98506	69527	2005
SWV 17	50	10	2794	0.00	2794	0.00	2794	61616	92065	69687	1844
SWV 18	50	10	2852	0.00	2852	0.00	2852	61895	92508	72198	1952
SWV 19	50	10	2843	0.00	2843	0.00	2843	64798	97050	74031	1942
SWV 20	50	10	2823	0.00	2823	0.00	2823	62519	92122	67583	1879

Tabelle A.7: *Zielfunktionswerte der Benchmark-Instanzen von Fisher und Thompson und von Storer, Wu und Vaccari*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
YN 1	20	20	846	+4.73	886	0.00	886	46	16543	16424	34
YN 2	20	20	870	+4.48	909	0.00	909	418	17439	17029	49
YN 3	20	20	840	+6.31	893	0.00	893	64	16601	16652	26
YN 4	20	20	920	+5.22	968	0.00	968	500	17949	17323	77
ABZ 5	10	10	1234	0.00	1234	0.00	1234	58	10563	8545	83
ABZ 6	10	10	943	0.00	943	0.00	943	0	7808	6818	45
ABZ 7	20	15	656	0.00	656	+0.15	657	1202	11756	10845	134
ABZ 8	20	15	646	+2.94	665	+0.30	667	1115	11899	11288	109
ABZ 9	20	15	662	+2.57	679	0.00	679	1151	11919	10831	104
ORB 1	10	10	1059	0.00	1059	0.00	1059	742	8297	5663	187
ORB 2	10	10	888	0.00	888	0.00	888	141	7353	5778	73
ORB 3	10	10	1005	0.00	1005	0.00	1005	660	8136	5592	158
ORB 4	10	10	1005	0.00	1005	0.00	1005	291	8097	6696	107
ORB 5	10	10	887	0.00	887	0.00	887	238	6978	5282	122
ORB 6	10	10	1010	0.00	1010	0.00	1010	321	8227	6154	82
ORB 7	10	10	397	0.00	397	0.00	397	25	3350	2735	18
ORB 8	10	10	899	0.00	899	0.00	899	600	6983	4778	185
ORB 9	10	10	934	0.00	934	0.00	934	279	7479	6078	93
ORB 10	10	10	944	0.00	944	0.00	944	236	7896	6277	82

Tabelle A.8: *Zielfunktionswerte der Benchmark-Instanzen von Yamada und Nakano, von Adams, Balas und Zawack, sowie von Applegate und Cook*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
LA 1	10	5	666	0.00	666	0.00	666	803	4832	2857	278
LA 2	10	5	655	0.00	655	0.00	655	612	4459	2767	152
LA 3	10	5	597	0.00	597	0.00	597	700	4151	2431	218
LA 4	10	5	590	0.00	590	0.00	590	754	4318	2521	207
LA 5	10	5	593	0.00	593	0.00	593	871	4094	2292	244
LA 6	15	5	926	0.00	926	0.00	926	2782	8773	4070	470
LA 7	15	5	890	0.00	890	0.00	890	2708	8251	3776	500
LA 8	15	5	863	0.00	863	0.00	863	2528	8025	3839	401
LA 9	15	5	951	0.00	951	0.00	951	2935	9235	4359	445
LA 10	15	5	958	0.00	958	0.00	958	3147	9026	4047	493
LA 11	20	5	1222	0.00	1222	0.00	1222	6521	14626	5547	681
LA 12	20	5	1039	0.00	1039	0.00	1039	5210	12262	4891	589
LA 13	20	5	1150	0.00	1150	0.00	1150	5988	13617	5382	624
LA 14	20	5	1292	0.00	1292	0.00	1292	7176	15319	5551	855
LA 15	20	5	1207	0.00	1207	0.00	1207	6475	14625	5663	738
LA 16	10	10	945	0.00	945	0.00	945	118	7508	5935	46
LA 17	10	10	784	0.00	784	0.00	784	217	6537	5114	84
LA 18	10	10	848	0.00	848	0.00	848	61	7052	5749	44
LA 19	10	10	842	0.00	842	0.00	842	15	7260	5762	31
LA 20	10	10	902	0.00	902	0.00	902	0	7499	6118	25

Tabelle A.9: *Zielfunktionswerte der Benchmark-Instanzen von Lawrence*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
LA 21	15	10	1046	0.00	1046	0.00	1046	1112	12570	10589	195
LA 22	15	10	927	0.00	927	0.00	927	1414	11947	9092	261
LA 23	15	10	1032	0.00	1032	0.00	1032	1252	12679	10298	195
LA 24	15	10	935	0.00	935	0.00	935	976	12161	10074	161
LA 25	15	10	977	0.00	977	0.00	977	1102	11786	9755	203
LA 26	20	10	1218	0.00	1218	0.00	1218	4307	19787	15835	393
LA 27	20	10	1235	0.00	1235	0.00	1235	4820	20836	15976	380
LA 28	20	10	1216	0.00	1216	0.00	1216	4088	19732	14627	385
LA 29	20	10	1152	0.00	1152	+0.09	1153	3722	18343	12541	362
LA 30	20	10	1355	0.00	1355	0.00	1355	4268	20545	14014	517
LA 31	30	10	1784	0.00	1784	0.00	1784	15732	36889	28689	961
LA 32	30	10	1850	0.00	1850	0.00	1850	17350	40213	30390	929
LA 33	30	10	1719	0.00	1719	0.00	1719	16212	36146	27244	927
LA 34	30	10	1721	0.00	1721	0.00	1721	17079	39027	30005	827
LA 35	30	10	1888	0.00	1888	0.00	1888	16902	38498	27319	1056
LA 36	15	15	1268	0.00	1268	0.00	1268	493	16978	15150	114
LA 37	15	15	1397	0.00	1397	0.00	1397	240	17897	16322	76
LA 38	15	15	1196	0.00	1196	0.00	1196	282	15657	14343	74
LA 39	15	15	1233	0.00	1233	0.00	1233	42	15834	15067	40
LA 40	15	15	1222	0.00	1222	0.00	1222	69	15972	14977	36

Tabelle A.10: *Zielfunktionswerte der Benchmark-Instanzen von Lawrence*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
TA 1	15	15	1231	0.00	1231	0.00	1231	131	16216	15147	66
TA 2	15	15	1244	0.00	1244	0.00	1244	470	16029	14824	79
TA 3	15	15	1218	0.00	1218	0.00	1218	312	15813	14227	91
TA 4	15	15	1175	0.00	1175	0.00	1175	422	15444	13679	122
TA 5	15	15	1224	0.00	1224	0.00	1224	403	16287	14806	127
TA 6	15	15	1238	0.00	1238	0.00	1238	406	16398	15065	82
TA 7	15	15	1227	0.00	1227	+0.08	1228	252	16258	15756	80
TA 8	15	15	1217	0.00	1217	0.00	1217	83	15815	15486	42
TA 9	15	15	1274	0.00	1274	0.00	1274	393	16794	15158	78
TA 10	15	15	1241	0.00	1241	0.00	1241	422	16312	14757	89
TA 11	20	15	1323	+2.87	1361	0.00	1361	2888	23463	22111	298
TA 12	20	15	1351	+1.18	1367	0.00	1367	2264	24802	23819	237
TA 13	20	15	1282	+4.68	1345	**-0.23**	1342	2279	23844	22247	225
TA 14	20	15	1345	0.00	1345	0.00	1345	2624	22894	22075	257
TA 15	20	15	1304	+2.76	1342	**-0.15**	1340	2461	23297	21957	240
TA 16	20	15	1302	+4.45	1362	**-0.14**	1360	2176	24796	23310	224
TA 17	20	15	1462	0.00	1462	+0.14	1464	2785	24835	23295	280
TA 18	20	15	1369	+1.97	1396	0.00	1396	3852	24809	23034	303
TA 19	20	15	1297	+2.93	1335	0.00	1335	3000	24406	22848	320
TA 20	20	15	1318	+2.50	1351	0.00	1351	2237	24633	23326	251

Tabelle A.11: *Zielfunktionswerte der Benchmark-Instanzen von Taillard*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
TA 21	20	20	1539	+6.82	1644	+0.06	1645	1952	30526	29279	237
TA 22	20	20	1511	+5.89	1600	0.00	1600	1198	30059	28884	107
TA 23	20	20	1472	+5.77	1557	+0.13	1559	690	29062	28727	80
TA 24	20	20	1602	+2.87	1648	0.00	1648	1254	29936	29840	126
TA 25	20	20	1504	+6.12	1596	+0.06	1597	2213	29882	28360	197
TA 26	20	20	1539	+7.02	1651	**-0.24**	1647	1346	30768	29521	191
TA 27	20	20	1616	+3.96	1680	+0.18	1683	1439	31036	30570	155
TA 28	20	20	1591	+1.44	1614	0.00	1614	725	29754	28694	155
TA 29	20	20	1514	+7.33	1625	0.00	1625	920	30518	30007	141
TA 30	20	20	1472	+7.61	1584	+0.32	1589	1469	29558	29189	157
TA 31	30	15	1764	0.00	1764	0.00	1764	12365	45692	43024	639
TA 32	30	15	1774	+1.24	1798	**-0.11**	1796	14272	48115	45166	681
TA 33	30	15	1778	+0.84	1793	+0.33	1799	15022	47988	46423	670
TA 34	30	15	1828	+0.05	1829	+0.16	1832	16071	49229	46266	726
TA 35	30	15	2007	0.00	2007	0.00	2007	13820	45438	43290	816
TA 36	30	15	1819	0.00	1819	0.00	1819	14665	48126	44931	644
TA 37	30	15	1771	+0.45	1779	+0.06	1780	13523	48653	45555	652
TA 38	30	15	1673	0.00	1677	**-0.24**	1673	10761	43931	40619	579
TA 39	30	15	1795	0.00	1795	0.00	1795	14270	45666	42641	741
TA 40	30	15	1631	+2.64	1674	+0.30	1679	12487	45925	43572	629

Tabelle A.12: *Zielfunktionswerte der Benchmark-Instanzen von Taillard*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
TA 41	30	20	1859	+8.55	2021	**-0.15**	2018	11202	55485	56891	597
TA 42	30	20	1867	+4.77	1956	0.00	1956	12002	56657	52447	633
TA 43	30	20	1809	+3.37	1870	+0.11	1872	9248	55161	52720	566
TA 44	30	20	1927	+3.22	1992	**-0.15**	1989	11297	57503	56080	640
TA 45	30	20	1997	+0.15	2000	0.00	2000	12718	56546	54741	596
TA 46	30	20	1940	+4.38	2025	**-0.15**	2022	15681	60221	57710	785
TA 47	30	20	1789	+6.76	1910	+0.05	1911	11413	56990	51042	641
TA 48	30	20	1912	+2.67	1962	**-0.31**	1956	13304	58444	52777	566
TA 49	30	20	1915	+2.77	1971	**-0.15**	1968	11786	57586	51746	670
TA 50	30	20	1807	+6.70	1928	+0.16	1931	8992	55846	53143	580
TA 51	50	15	2760	0.00	2760	0.00	2760	63541	113690	104368	1535
TA 52	50	15	2756	0.00	2756	0.00	2756	60638	110737	102337	1612
TA 53	50	15	2717	0.00	2717	0.00	2717	58958	110435	100485	1568
TA 54	50	15	2839	0.00	2839	0.00	2839	60270	111577	100811	1694
TA 55	50	15	2679	0.00	2679	0.00	2679	62128	107379	96387	1597
TA 56	50	15	2781	0.00	2781	0.00	2781	65306	113955	103370	1674
TA 57	50	15	2943	0.00	2943	0.00	2943	67479	117006	104968	1654
TA 58	50	15	2885	0.00	2885	0.00	2885	65051	116321	107649	1650
TA 59	50	15	2655	0.00	2655	0.00	2655	59986	110779	100422	1502
TA 60	50	15	2723	0.00	2723	0.00	2723	59660	112154	102676	1529

Tabelle A.13: *Zielfunktionswerte der Benchmark-Instanzen von Taillard*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
TA 61	50	20	2868	0.00	2868	0.00	2868	62085	137604	124505	1408
TA 62	50	20	2869	+0.07	2872	0.00	2872	62729	140484	131748	1549
TA 63	50	20	2755	0.00	2755	0.00	2755	56835	133070	120261	1489
TA 64	50	20	2702	0.00	2702	0.00	2702	58096	129764	116915	1409
TA 65	50	20	2725	0.00	2725	0.00	2725	57288	134824	124973	1411
TA 66	50	20	2845	0.00	2845	0.00	2845	54852	137945	125696	1412
TA 67	50	20	2825	0.00	2825	0.00	2825	62130	132182	124551	1520
TA 68	50	20	2784	0.00	2784	0.00	2784	58020	136635	119691	1505
TA 69	50	20	3071	0.00	3071	0.00	3071	64950	143958	126350	1683
TA 70	50	20	2995	0.00	2995	0.00	2995	64413	137654	125231	1716
TA 71	100	20	5464	0.00	5464	0.00	5464	356472	503851	458847	5179
TA 72	100	20	5181	0.00	5181	0.00	5181	345597	465273	444239	4600
TA 73	100	20	5568	0.00	5568	0.00	5568	368565	506836	468846	5406
TA 74	100	20	5339	0.00	5339	0.00	5339	343827	469719	442951	4712
TA 75	100	20	5392	0.00	5392	0.00	5392	358332	481152	455698	5068
TA 76	100	20	5342	0.00	5342	0.00	5342	329912	475094	454905	5019
TA 77	100	20	5436	0.00	5436	0.00	5436	347115	494635	459185	4874
TA 78	100	20	5394	0.00	5394	0.00	5394	353605	486059	452346	4959
TA 79	100	20	5358	0.00	5358	0.00	5358	337194	471163	446597	4838
TA 80	100	20	5183	0.00	5183	0.00	5183	355536	478296	448463	4920

Tabelle A.14: *Zielfunktionswerte der Benchmark-Instanzen von Taillard*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
DMU 1 rcmax_20_15_4	20	15	2501	+2.48	2579	**-0.62**	2563	3161	45471	39669	352
DMU 2 rcmax_20_15_10	20	15	2651	+2.07	2716	**-0.37**	2706	4370	48224	43748	448
DMU 3 rcmax_20_15_5	20	15	2731	0.00	2731	+0.37	2741	3806	46483	43725	386
DMU 4 rcmax_20_15_8	20	15	2669	0.00	2669	0.00	2669	2707	47885	43397	336
DMU 5 rcmax_20_15_1	20	15	2749	0.00	2749	+0.29	2757	4851	49277	42753	507
DMU 6 rcmax_20_20_6	20	20	2834	+14.75	3269	**-0.52**	3252	241	60620	59234	91
DMU 7 rcmax_20_20_4	20	20	2677	+14.46	3064	0.00	3064	869	55706	54914	151
DMU 8 rcmax_20_20_7	20	20	2901	+10.27	3210	**-0.34**	3199	747	58377	56298	207
DMU 9 rcmax_20_20_8	20	20	2739	+12.89	3126	**-1.09**	3092	2283	58073	56091	344
DMU 10 rcmax_20_20_5	20	20	2716	+9.90	3001	**-0.53**	2985	578	56620	55906	122
DMU 11 rcmax_30_15_9	30	15	3395	+2.53	3491	**-0.29**	3481	24390	89611	84401	1221
DMU 12 rcmax_30_15_10	30	15	3481	+1.69	3578	**-1.06**	3540	29126	95326	88642	1470
DMU 13 rcmax_30_15_5	30	15	3681	+0.49	3715	**-0.43**	3699	28698	95890	85646	1590
DMU 14 rcmax_30_15_4	30	15	3394	0.00	3396	**-0.06**	3394	27377	89748	80215	1291
DMU 15 rcmax_30_15_1	30	15	3332	+0.33	3343	0.00	3343	23583	88027	80405	1209
DMU 16 rcmax_30_20_7	30	20	3726	+1.64	3802	**-0.40**	3787	19983	106125	104326	941
DMU 17 rcmax_30_20_10	30	20	3697	+4.84	3944	**-1.72**	3876	22565	109657	104564	993
DMU 18 rcmax_30_20_9	30	20	3844	+0.21	3894	**-1.08**	3852	18766	109401	105336	1049
DMU 19 rcmax_30_20_8	30	20	3650	+4.76	3891	**-1.72**	3824	20002	104633	102756	969
DMU 20 rcmax_30_20_2	30	20	3604	+3.94	3788	**-1.11**	3746	20272	105397	104079	828

Tabelle A.15: *Zielfunktionswerte der Benchmark-Instanzen von Demirkol, Mehta und Uzsoy*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
DMU 21 rcmax_40_15_5	40	15	4380	0.00	4380	0.00	4380	56944	149560	124241	2053
DMU 22 rcmax_40_15_9	40	15	4725	0.00	4725	0.00	4725	63012	156186	141354	2329
DMU 23 rcmax_40_15_10	40	15	4668	0.00	4668	0.00	4668	62009	155085	135937	2117
DMU 24 rcmax_40_15_8	40	15	4648	0.00	4648	0.00	4648	62647	154306	141166	2109
DMU 25 rcmax_40_15_2	40	15	4164	0.00	4164	0.00	4164	54842	139567	125696	1915
DMU 26 rcmax_40_20_1	40	20	4647	+0.80	4712	**-0.60**	4684	51298	171993	162965	1908
DMU 27 rcmax_40_20_3	40	20	4848	0.00	4883	**-0.72**	4848	55270	179145	167658	1919
DMU 28 rcmax_40_20_6	40	20	4692	0.00	4692	0.00	4692	54666	171300	158885	1950
DMU 29 rcmax_40_20_2	40	20	4691	0.00	4700	**-0.19**	4691	59031	179525	163046	1978
DMU 30 rcmax_40_20_7	40	20	4732	0.00	4779	**-0.98**	4732	52918	173040	166440	1889
DMU 31 rcmax_50_15_3	50	15	5640	0.00	5640	0.00	5640	121910	240177	203626	3191
DMU 32 rcmax_50_15_1	50	15	5927	0.00	5927	0.00	5927	114316	225465	180323	3505
DMU 33 rcmax_50_15_2	50	15	5728	0.00	5728	0.00	5728	118127	231708	196800	3429
DMU 34 rcmax_50_15_4	50	15	5385	0.00	5385	0.00	5385	119204	226177	189546	2950
DMU 35 rcmax_50_15_5	50	15	5635	0.00	5635	0.00	5635	120175	234313	196383	3080
DMU 36 rcmax_50_20_2	50	20	5621	0.00	5621	0.00	5621	116021	264748	243284	2831
DMU 37 rcmax_50_20_7	50	20	5851	0.00	5852	**-0.02**	5851	121430	270008	241673	3074
DMU 38 rcmax_50_20_6	50	20	5713	0.00	5713	0.00	5713	114271	271162	244865	3093
DMU 39 rcmax_50_20_9	50	20	5747	0.00	5747	0.00	5747	121298	265971	238520	2719
DMU 40 rcmax_50_20_3	50	20	5577	0.00	5577	0.00	5577	110711	252930	234941	2763

Tabelle A.16: *Zielfunktionswerte der Benchmark-Instanzen von Demirkol, Mehta und Uzsoy*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
DMU 41 cscmax_20_15_10	20	15	2839	+16.66	3376	**-1.90**	3312	10336	53192	32907	995
DMU 42 cscmax_20_15_5	20	15	3066	+11.42	3574	**-4.42**	3416	10372	57258	37785	1097
DMU 43 cscmax_20_15_8	20	15	3121	+10.70	3535	**-2.26**	3455	11364	55530	34009	1161
DMU 44 cscmax_20_15_7	20	15	3112	+12.50	3599	**-2.72**	3501	13093	59003	36291	1192
DMU 45 cscmax_20_15_1	20	15	2930	+11.71	3355	**-2.44**	3273	11615	54821	34426	903
DMU 46 cscmax_20_20_6	20	20	3425	+20.29	4192	**-1.72**	4120	10007	69826	59710	871
DMU 47 cscmax_20_20_4	20	20	3353	+19.27	4060	**-1.50**	3999	10806	72826	60239	792
DMU 48 cscmax_20_20_3	20	20	3317	+15.76	3918	**-2.24**	3834	10049	68875	57208	963
DMU 49 cscmax_20_20_2	20	20	3369	+11.75	3810	**-1.18**	3765	9338	65829	48725	898
DMU 50 cscmax_20_20_9	20	20	3379	+11.63	3866	**-2.43**	3772	9441	70212	55145	894
DMU 51 cscmax_30_15_2	30	15	3839	+11.07	4320	**-1.30**	4264	41728	106553	69383	2188
DMU 52 cscmax_30_15_9	30	15	4012	+9.70	4586	**-4.03**	4401	39826	104568	65100	2131
DMU 53 cscmax_30_15_10	30	15	4108	+9.01	4635	**-3.39**	4478	41304	116237	61643	2196
DMU 54 cscmax_30_15_5	30	15	4165	+7.20	4525	**-1.33**	4465	41041	110903	63368	2221
DMU 55 cscmax_30_15_6	30	15	4099	+7.39	4500	**-2.18**	4402	39444	103482	59721	2162
DMU 56 cscmax_30_20_9	30	20	4366	+18.25	5234	**-1.36**	5163	49460	143067	125120	2312
DMU 57 cscmax_30_20_7	30	20	4182	+14.32	4962	**-3.65**	4781	47980	131158	117067	2102
DMU 58 cscmax_30_20_3	30	20	4214	+16.09	5038	**-2.90**	4892	39193	129600	118364	2207
DMU 59 cscmax_30_20_6	30	20	4199	+15.84	5004	**-2.80**	4864	42061	129587	111341	2032
DMU 60 cscmax_30_20_4	30	20	4259	+14.82	5088	**-3.89**	4890	45629	138260	115618	2188

Tabelle A.17: *Zielfunktionswerte der Benchmark-Instanzen von Demirkol, Mehta und Uzsoy*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	$\Delta_{C_{max}}^{(UB,LB)}$	C_{max}^{UB}	$\Delta_{C_{max}}^{(GA,UB)}$	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$	$\|L\|_{max}^{GA}$
DMU 61 cscmax_40_15_3	40	15	4886	+9.48	5448	**-1.82**	5349	87329	177511	104457	3855
DMU 62 cscmax_40_15_6	40	15	5004	+6.75	5482	**-2.55**	5342	91076	181771	139858	3660
DMU 63 cscmax_40_15_8	40	15	5049	+7.68	5597	**-2.86**	5437	90027	169972	116085	4054
DMU 64 cscmax_40_15_4	40	15	5130	+4.62	5580	**-3.82**	5367	88966	175539	118671	3655
DMU 65 cscmax_40_15_7	40	15	5072	+4.71	5398	**-1.61**	5311	85842	171982	115218	3862
DMU 66 cscmax_40_20_10	40	20	5357	+10.32	6131	**-3.60**	5910	102315	219698	164111	4007
DMU 67 cscmax_40_20_6	40	20	5484	+11.54	6275	**-2.52**	6117	108337	230471	183322	4193
DMU 68 cscmax_40_20_8	40	20	5423	+9.70	6152	**-3.30**	5949	106940	217939	184543	4075
DMU 69 cscmax_40_20_5	40	20	5419	+9.15	6093	**-2.92**	5915	110245	225283	177391	4016
DMU 70 cscmax_40_20_9	40	20	5492	+11.34	6368	**-3.97**	6115	107251	227186	182673	4092
DMU 71 cscmax_50_15_8	50	15	6050	+5.34	6610	**-3.59**	6373	136899	257436	191796	5477
DMU 72 cscmax_50_15_6	50	15	6223	+6.81	6790	**-2.11**	6647	144730	262486	188275	5554
DMU 73 cscmax_50_15_10	50	15	5935	+6.91	6536	**-2.92**	6345	140010	253156	183769	5315
DMU 74 cscmax_50_15_4	50	15	6015	+6.00	6623	**-3.73**	6376	144269	261706	194186	5578
DMU 75 cscmax_50_15_3	50	15	6010	+6.22	6552	**-2.56**	6384	144772	256721	184488	5368
DMU 76 cscmax_50_20_1	50	20	6329	+10.20	7454	**-6.43**	6975	186957	341232	321229	5795
DMU 77 cscmax_50_20_4	50	20	6399	+8.91	7329	**-4.91**	6969	192969	343021	310282	5794
DMU 78 cscmax_50_20_3	50	20	6508	+6.98	7301	**-4.63**	6962	188046	339296	305195	5623
DMU 79 cscmax_50_20_7	50	20	6593	+8.66	7590	**-5.61**	7164	203874	346149	324921	5850
DMU 80 cscmax_50_20_9	50	20	6435	+6.04	7173	**-4.87**	6824	176561	324234	302460	5619

Tabelle A.18: *Zielfunktionswerte der Benchmark-Instanzen von Demirkol, Mehta und Uzsoy*

A.5 Benchmark-Instanzen für $J||L_{max}$

In den folgenden Tabellen sind in der ersten Spalte die Bezeichnungen der Instanzen, dann die Anzahlen der Jobs $|\mathcal{J}|$ und der Maschinen $|\mathcal{M}|$ zu finden. Die Spalten vier und fünf enthalten die Parameter τ und R, die zur Generierung der Instanzen verwendet wurden. Ihre Bedeutung ist in Abschnitt 2.2.1 beschrieben. In den nächsten Spalten sind die unteren Schranken für die Terminabweichung L_{max}^{LB}, die bisher besten bekannten oberen Schranken für die Terminabweichung L_{max}^{UB} und die mit dem genetischen Algorithmus berechneten oberen Schranken für die maximale absolute Terminabweichung $|L|_{max}^{GA}$ angegeben. Weiterhin sind in der Spalte neun die Ein-Maschinen-Schranken für den Makespan C_{max}^{LB} und in der Spalte zehn die vom genetischen Algorithmus ermittelten oberen Schranken für den Makespan C_{max}^{GA} notiert. Für die Instanzen mit 300 und 400 Vorgängen wurden desweiteren obere Schranken für die totale Terminabweichung $\sum T_j^{GA}$, die Summe der Fertigstellungszeiten $\sum C_j^{GA}$ und die Summe der Durchlaufzeiten $\sum F_j^{GA}$ ermittelt. Diese sind in den Tabellen A.20 und A.21 zu finden. Die für die Berechnungen aufgewendeten Zeiten sind in Abschnitt 2.2.5 vermerkt.

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	τ	R	L_{max}^{LB}	L_{max}^{UB}	$\|L\|_{max}^{GA}$	C_{max}^{LB}	C_{max}^{GA}
r_50_20_1_1_5	50	20	0.3	0.5	1591	2181	1883	5480	5511
r_50_20_1_1_3	50	20	0.3	0.5	1746	2390	2295	5818	5821
r_50_20_1_1_2	50	20	0.3	0.5	1794	2355	2070	5708	5722
r_50_20_1_1_6	50	20	0.3	0.5	1845	2219	2136	5595	5626
r_50_20_1_1_4	50	20	0.3	0.5	1786	2142	1959	5750	5766
cr_50_20_1_1_1	50	20	0.3	0.5	2140	3471	3146	6146	7277
cr_50_20_1_1_7	50	20	0.3	0.5	2424	3721	3340	6748	7463
cr_50_20_1_1_5	50	20	0.3	0.5	2482	3574	3366	6571	7269
cr_50_20_1_1_3	50	20	0.3	0.5	2802	4011	3895	6985	7693
cr_50_20_1_1_10	50	20	0.3	0.5	2417	3448	3331	6416	7318

Tabelle A.19: *Zielfunktionswerte der Benchmark-Instanzen von Demirkol, Mehta und Uzsoy*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	τ	R	L_{max}^{LB}	L_{max}^{UB}	$\|L\|_{max}^{GA}$	C_{max}^{LB}	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$
r_20_15_1_1_6	20	15	0.3	0.5	1027	1448	1245	2427	2719	19632	48221	40681
r_20_15_1_1_8	20	15	0.3	0.5	1127	1552	1444	2532	2895	21800	48637	41942
r_20_15_1_1_4	20	15	0.3	0.5	1160	1492	1351	2520	2757	20850	47938	41572
r_20_15_1_1_2	20	15	0.3	0.5	1140	1464	1302	2646	2875	18273	48765	41635
r_20_15_1_1_3	20	15	0.3	0.5	1182	1501	1294	2689	2828	20819	48892	41659
r_20_15_2_1_7	20	15	0.6	0.5	1575	1957	1892	2467	2785	30547	46679	42134
r_20_15_2_1_3	20	15	0.6	0.5	1727	2100	1941	2583	2821	31962	47551	41607
r_20_15_2_1_1	20	15	0.6	0.5	1785	2165	1973	2617	2861	33850	48944	42692
r_20_15_2_1_5	20	15	0.6	0.5	1521	1839	1718	2473	2594	29792	46300	41142
r_20_15_2_1_9	20	15	0.6	0.5	1858	2143	1942	2749	2867	32721	49605	42744
r_20_20_1_1_7	20	20	0.3	0.5	1391	2013	1893	2758	3413	33313	61484	57802
r_20_20_1_1_10	20	20	0.3	0.5	1182	1708	1651	2704	3223	27461	57075	52274
r_20_20_1_1_6	20	20	0.3	0.5	1366	1962	1748	2694	3243	30371	58048	53424
r_20_20_1_1_3	20	20	0.3	0.5	1569	2248	2055	2905	3438	35259	63167	55362
r_20_20_1_1_4	20	20	0.3	0.5	1226	1753	1531	2618	2967	26666	54872	52101
r_20_20_2_1_2	20	20	0.6	0.5	1776	2638	2404	2634	3298	40642	57506	53245
r_20_20_2_1_6	20	20	0.6	0.5	1868	2647	2340	2751	3210	42019	57846	52725
r_20_20_2_1_4	20	20	0.6	0.5	1845	2535	2373	2698	3270	42833	58274	54626
r_20_20_2_1_8	20	20	0.6	0.5	1927	2627	2444	2707	3301	41788	57393	54025
r_20_20_2_1_7	20	20	0.6	0.5	1947	2640	2448	2828	3275	41267	56194	53234

Tabelle A.20: *Zielfunktionswerte der Benchmark-Instanzen von Demirkol, Mehta und Uzsoy*

Benchmark	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	τ	R	L_{max}^{LB}	L_{max}^{UB}	$\|L\|_{max}^{GA}$	C_{max}^{LB}	C_{max}^{GA}	$\sum T_j^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$
cr_20_15_1_1_8	20	15	0.3	0.5	1434	2159	1932	3081	3624	27804	57331	35611
cr_20_15_1_1_3	20	15	0.3	0.5	1619	2293	2139	3174	3726	31630	59278	36146
cr_20_15_1_1_1	20	15	0.3	0.5	1558	2126	2005	3075	3578	28968	56786	33023
cr_20_15_1_1_5	20	15	0.3	0.5	1522	2049	1998	3121	3593	25812	56589	33959
cr_20_15_1_1_9	20	15	0.3	0.5	1693	2224	2162	3263	3680	29600	56470	34056
cr_20_15_2_1_6	20	15	0.6	0.5	2075	2761	2547	2865	3438	39255	55941	32246
cr_20_15_2_1_7	20	15	0.6	0.5	2397	3098	2871	3215	3807	42671	57340	35697
cr_20_15_2_1_3	20	15	0.6	0.5	2033	2612	2429	3003	3381	37656	53294	33831
cr_20_15_2_1_4	20	15	0.6	0.5	2298	2914	2742	3260	3798	42664	59722	38381
cr_20_15_2_1_10	20	15	0.6	0.5	1989	2770	2641	3072	3589	40695	56995	34944
cr_20_20_1_1_10	20	20	0.3	0.5	1759	2797	2572	3323	4085	41904	68658	46832
cr_20_20_1_1_5	20	20	0.3	0.5	1895	2879	2771	3387	4205	42160	71224	49709
cr_20_20_1_1_3	20	20	0.3	0.5	1911	2894	2703	3505	4192	41891	69778	48735
cr_20_20_1_1_9	20	20	0.3	0.5	1770	2643	2582	3312	4140	38220	64390	48295
cr_20_20_1_1_6	20	20	0.3	0.5	2088	2948	2858	3636	4440	44486	75373	58036
cr_20_20_2_1_8	20	20	0.6	0.5	2758	3950	3662	3527	4501	62552	74266	59374
cr_20_20_2_1_9	20	20	0.6	0.5	2504	3489	3126	3469	4100	52311	68038	48043
cr_20_20_2_1_1	20	20	0.6	0.5	2550	3538	3267	3329	4224	52734	69217	51414
cr_20_20_2_1_7	20	20	0.6	0.5	2426	3304	3110	3362	4106	50222	67486	47784
cr_20_20_2_1_6	20	20	0.6	0.5	2431	3244	3120	3422	4078	51697	67833	47570

Tabelle A.21: *Zielfunktionswerte der Benchmark-Instanzen von Demirkol, Mehta und Uzsoy*

A.6 Benchmark-Instanzen für praktische Job-Shop-Probleme

Die folgenden Tabellen A.22 bis A.25 enthalten die Daten für die Instanzen der praktischen Job-Shop Scheduling-Probleme. Die bisherigen Bezeichnungen der Spalten wurde beibehalten. In den Tabellen A.24 und A.25 werden zusätzlich die Anzahlen der Jobs und Tasks angegeben, da diese durch die Generierung bedingt nicht alle gleich sind. Für die Instanzen DRIR56-70 wird auch die Anzahl der Ressourcen angegeben.

Instanz	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	C_{max}^{GA}	$\sum T_j^{GA}$	$\|L\|_{max}^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$
D 1	20	15	1273	1383	1592	250	23985	17819
D 2	20	15	1301	1344	1344	181	23913	19083
D 3	20	15	1353	1371	1448	163	23856	19300
D 4	20	15	1253	1293	1518	209	22827	16237
D 5	20	15	1308	1397	2316	211	23839	18925
D 6	30	20	1938	1999	6565	765	56563	50555
D 7	30	20	1789	1875	4758	606	56115	50841
D 8	30	20	1834	1922	7511	632	56103	51597
D 9	30	20	1863	1995	5895	708	59504	53632
D10	30	20	1980	2003	6550	611	58846	52789
D11	40	20	2436	2450	5440	878	96948	84054
D12	40	20	2345	2386	6027	817	89989	81856
D13	40	20	2285	2362	7010	904	95293	85599
D14	40	20	2230	2330	4623	781	89875	81810
D15	40	20	2525	2525	3701	802	92604	83965
D16	50	30	3122	3552	2127	1382	174185	160268
D17	50	30	3034	3044	1074	1324	168640	155611
D18	50	30	3004	3384	427	1467	175185	157736
D19	50	30	2873	3202	3580	1580	165686	154476
D20	50	30	2970	3260	824	1441	170055	154562

Tabelle A.22: *Zielfunktionswerte der Benchmark-Instanzen für D||multi*

Instanz	$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	C_{max}^{LB}	C_{max}^{GA}	$\sum T_j^{GA}$	$\|L\|_{max}^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$
DR21	20	15	1267	1354	2635	311	23572	18939
DR22	20	15	1133	1255	1828	297	22762	18973
DR23	20	15	1170	1340	2597	322	22967	18827
DR24	20	15	1197	1233	2194	256	22672	17426
DR25	20	15	1239	1293	1243	229	23311	19697
DR26	30	20	1750	1879	9713	877	57077	52196
DR27	30	20	1795	1924	8304	939	56158	51773
DR28	30	20	2066	2113	8339	907	59854	54408
DR29	30	20	1941	2025	8455	813	58734	53385
DR30	30	20	1801	1996	11494	916	57312	52393
DR31	40	20	2287	2291	9246	1045	94171	84804
DR32	40	20	2442	2482	8321	949	96605	84405
DR33	40	20	2200	2335	9033	915	94680	84167
DR34	40	20	2338	2424	8743	1045	96607	85795
DR35	40	20	2434	2500	6215	1030	95897	87267
DR36	50	30	2917	3319	6423	1897	172942	165050
DR37	50	30	3138	3329	8375	2056	179388	164849
DR38	50	30	2871	3214	6374	1925	171186	159990
DR39	50	30	2835	3233	10936	2081	175510	157731
DR40	50	30	3042	3218	7511	2178	172938	158788

Tabelle A.23: *Zielfunktionswerte der Benchmark-Instanzen für* $D|resum|multi$

Instanz	$\|A\|$	$\|\mathcal{M}\|$	$\|\mathcal{J}\|$	$\|\mathcal{T}\|$	C_{max}^{LB}	C_{max}^{GA}	$\sum T_j^{GA}$	$\|L\|_{max}^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$
DRI41	15	15	32	343	1588	1781	0	8	16752	13384
DRI42	15	15	28	306	2168	2311	0	5	11960	10223
DRI43	15	15	28	330	1872	2032	0	8	16495	13766
DRI44	15	15	29	328	2067	2193	0	25	16033	12801
DRI45	15	15	33	379	2027	2214	0	8	19445	16756
DRI46	20	15	64	725	2791	3052	111	117	39310	38477
DRI47	20	15	55	672	2571	2833	989	284	37804	35929
DRI48	20	15	73	854	3260	3629	901	1270	52478	50722
DRI49	20	15	35	385	1681	1851	0	155	22839	17371
DRI50	20	15	56	663	2695	2817	0	544	37678	31917
DRI51	20	20	44	670	2279	2761	0	424	39416	36541
DRI52	20	20	45	688	2455	2908	0	452	36788	35317
DRI53	20	20	49	746	2494	3111	0	688	40229	38355
DRI54	20	20	42	624	2025	2333	0	241	33849	30757
DRI55	20	20	52	792	2362	3009	0	746	44278	42216

Tabelle A.24: *Zielfunktionswerte der Benchmark-Instanzen für* $D|resum, intree|multi$

Instanz	$\|A\|$	$\|\mathcal{M}\|$	$\|R\|$	$\|\mathcal{J}\|$	$\|\mathcal{T}\|$	C_{max}^{LB}	C_{max}^{GA}	$\sum T_j^{GA}$	$\|L\|_{max}^{GA}$	$\sum C_j^{GA}$	$\sum F_j^{GA}$
DRJR56	15	15	15	24	301	2001	2194	0	191	24149	22152
DRJR57	15	15	15	31	367	1846	2177	0	440	25198	22555
DRJR58	15	15	15	41	459	1909	2819	310	748	32583	29405
DRJR59	15	15	15	32	354	2355	3006	0	373	26918	25318
DRJR60	15	15	15	29	340	1592	2141	201	341	22845	20738
DRJR61	20	15	15	55	657	2740	4033	2480	1819	56920	48617
DRJR62	20	15	15	55	621	2431	3684	7175	1678	56539	48987
DRJR63	20	15	15	42	480	1904	2583	3462	1144	43790	39361
DRJR64	20	15	15	58	648	2802	4014	9142	1901	63119	55096
DRJR65	20	15	15	46	537	2122	3290	1383	1182	53135	46683
DRJR66	20	20	20	48	750	2665	5312	13501	2545	77265	68835
DRJR67	20	20	20	61	914	2785	5772	17957	3316	97523	87877
DRJR68	20	20	20	45	669	2347	4260	6665	2266	69345	62536
DRJR69	20	20	20	43	666	2464	4155	9316	2087	65238	59429
DRJR70	20	20	20	39	547	1729	3653	9778	1805	59058	52201

Tabelle A.25: *Zielfunktionswerte der Benchmark-Instanzen für* $D|resum, intree, res|multi$

A.7 Abbildungen

Die Abbildung A.1 gibt die Häufigkeiten der Clustergrößen bei verschiedenen Algorithmen an. Da die Klassifikationen drei Cluster enthalten, können die Clustergrößen von 1 bis 48 variieren. Die Spalten 1 bis 12 bezeichnen die Algorithmen in folgenden Form. Sei k die Nummer der jeweiligen Spalte. Für $k \in \{1, \ldots, 6\}$ wurde das hierarchische Clusterverfahren verwendet und für $k \in \{7, \ldots, 12\}$ das nicht-hierarchische. Für die Spalten $k \in \{1, 2, 3, 7, 8, 9\}$ wurde das Abstandsmaß $D(\alpha, \beta)$ verwendet. $T(\alpha, \beta)$ fand bei den anderen Anwendung. Schließlich wurden die Heterogenitätsmaße v_s in $k \in \{1, 4, 7, 10\}$, v_c in $k \in \{2, 5, 8, 11\}$ und v_a in $k \in \{3, 6, 9, 12\}$ eingesetzt.

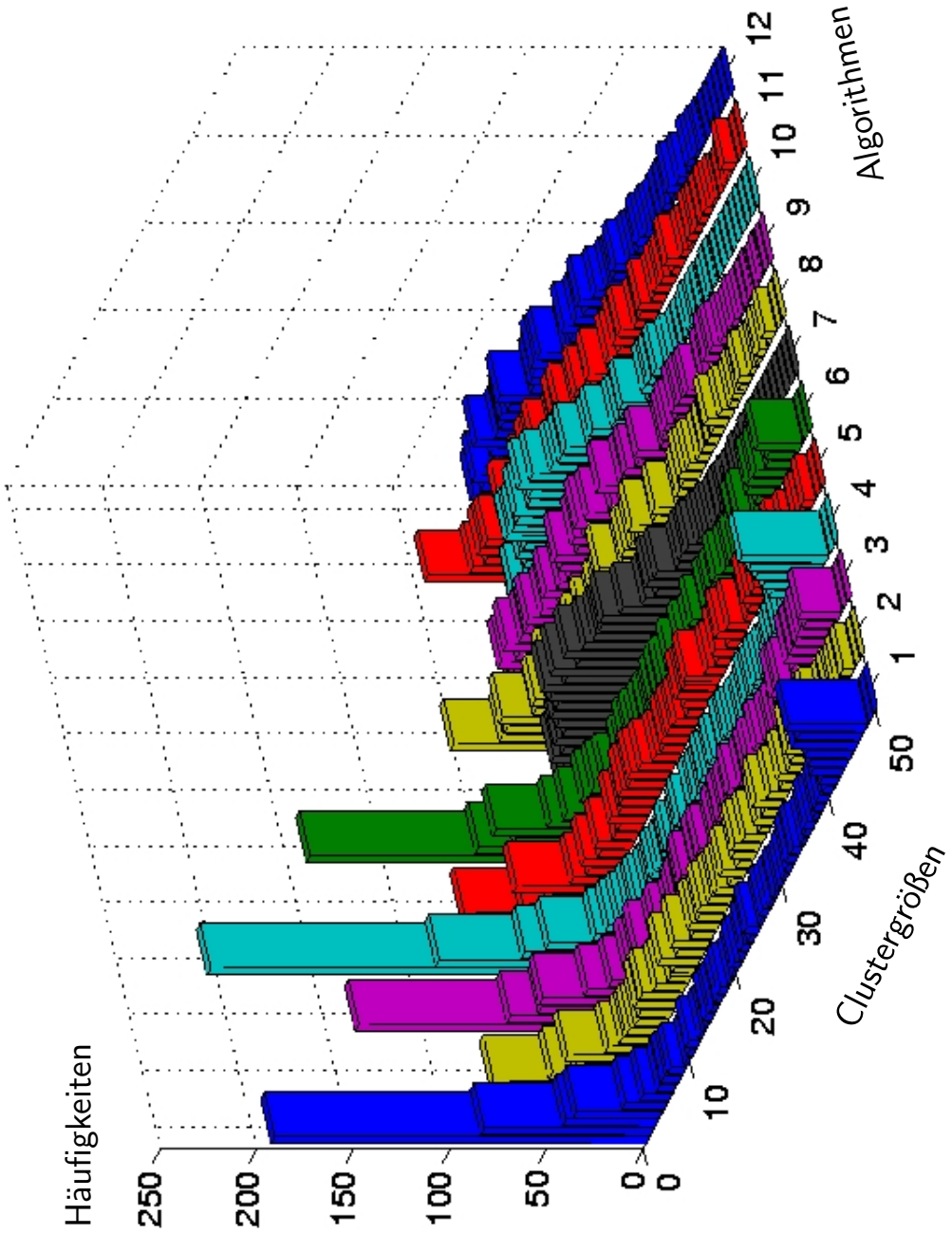

Abbildung A.1: *Häufigkeiten der Clustergrößen bei verschiedenen Algorithmen, Abstands- und Heterogenitätsmaßen*

Anhang B

Software

Im Rahmen dieser Arbeit wurde das Softwarepaket **„Visual Scheduler“** entwickelt. Vom Autor wurden die in der Arbeit vorgestellten Algorithmen implementiert und eine graphische Benutzerschnittstelle entwickelt. Weiterhin wurde der Generator für Probleminstanzen integriert.

Zur visuellen Entscheidungs-Unterstützung wurden Diagrammdarstellungen zum Vergleich von Lösungen der multikriteriellen Probleme entwickelt. Hierzu gehören Balken- und Liniendiagramme zum Zielfunktionsvergleich sowie Gantt-Diagramme und Histogramme zum Vergleich der Lösungsstrukturen. Ein Teil dieser Visualisierungen entstand im Rahmen eines Studentenprojekts mit Studenten der Fakultät Medien der Bauhaus-Universität Weimar im Wintersemester 2001/2002. Visualisierungen der Clusterung von Lösungen, der Struktur der Ausgangsdaten und den Abhängigkeiten zwischen Pareto-Lösungen und dominierten Lösungen gehören ebenfalls zum Umfang des Programmpakets.

B.1 Technischer Hintergrund

Für die Entwicklung des Softwarepakets gab es mehrere Grundanforderungen.

1. Die Implementation der Algorithmen und die Benutzerschnittstelle sollten unabhängig vom Betriebssystem sein.

2. Die Algorithmen sollten effizient und erweiterbar implementiert werden.

3. Alle benutzten Programmbibliotheken sollten frei verfügbar sein.

Deshalb wurden die Algorithmen in der **Sprache C** programmiert und die Oberfläche mit Hilfe der Bibliotheken des GIMP-Toolkits **GTK+ 1.2** entwickelt. Für Datenstrukturen, wie dynamische Felder und verkettete Listen, wurde die Bibliothek **Glib 1.2** verwendet, die zum Umfang von GTK+ gehört. In der Version 1.2 ist in der Glib kein Zufallszahlengenerator implementiert. Aus diesem Grund wurde der Mersenne Twister [39] als Zufallszahlengenerator verwendet.

Die Verwendung dieser Programmiersprachen und Bibliotheken führten zu einer hohen Portabilität des Programmpakets. Es steht unter verschiedenen Unix-Derivaten (einschließlich Linux) und Windows 2000, NT und XP zu Verfügung.

B.2 Funktionsumfang

Ergänzend zu den in dieser Arbeit verwendeten Zielfunktionen und Optimierungsheuristiken sind in dem Programmpaket weitere enthalten. Diese Ergänzungen werden im Folgenden kurz aufgeführt.

B.2.1 Zielfunktionen und Nachbarschaften

Zusätzlich zu den in der Arbeit hauptsächlich verwendeten Zielfunktionen C_{max}, $\sum C_j$, $\sum T_j$ und $|L|_{max}$ wurden weitere Zielfunktionen implementiert. Diese Zielfunktionen sind:

- die Summe der Durchlaufzeiten $\sum F_j$,
- die totale Earliness $\sum E_j$,
- die Anzahl der verspäteten Aufträge $\sum U_j$,
- die Anzahl der kritischen Vorgänge bezüglich der Zielfunktion C_{max}
- und die Summe der Leerzeiten auf den Maschinen.

Zu jeder dieser Zielfunktionen wird das Minimum gesucht. Alle diese Zielfunktionen können auch zur multikriteriellen Optimierung herangezogen werden. Dabei ist die lexikographische Optimierung sowie die Approximation der Menge Pareto-Lösungen möglich. Alle vier Ranking-Funktionen können zur Rangermittlung im Lösungspool verwendet werden.

Weiterhin bietet das Programm die Möglichkeit, die in Abschnitt 2.1.2 definierten Nachbarschaften auf die Menge der bezüglich der Zielfunktion C_{max} kritischen Vorgänge einzuschränken.

B.2.2 Lösungsverfahren

Alle in der Arbeit vorgestellten Lokale-Suche-Heuristiken wurden im Visual Scheduler implementiert und können in der genetischen lokalen Suche genutzt werden. Des Weiteren wurden die schnelle lokale Suche, die volle lokale Suche, das Simulated Annealing und die Tabu-Suche für die multikriterielle Optimierung erweitert. Die Erweiterung der schnellen und der vollen lokalen Suche bestand darin, die aktuelle und die Nachbarlösung anhand ihrer Ranking-Funktionswerte zu vergleichen. Durch diese Vergleichsmethode ist es möglich, dass Zyklen in der Nachbarschaftssuche auftreten. Dies ist aber durch die Verwendung der randomisierten Nachbarschaft in den Versuchsläufen nicht aufgetreten. Da eine Zeitschranke für die Laufzeit der Algorithmen angegeben werden muss, terminieren die Algorithmen in jedem Fall. Beide Varianten der lokalen Suche ermitteln jeweils nur eine Lösung.

Die Erweiterung des Simulated Annealings und der Tabu-Suche für die multikriterielle Optimierung ist umfangreicher. Die Tabu-Suche hat bei Verwendung der Nachbarschaften ohne Einschränkung auf die kritischen Vorgänge zu große Laufzeitnachteile. Aus diesem Grund wird hier nur das multikriterielle Simulated Annealing angegeben.

Algorithmus B.1 *Multikriterielles Simulated Annealing*
Seien N_β die verwendete Nachbarschaft, $F = \{f_k\}_{k=1,\dots,m}$ die Menge der Zielfunktionen, $W = \{w_i\}_{i=1,\dots,m}$ die Menge der Gewichte, T_0 die Starttemperatur, IA die Anzahl von Iterationen zwischen zwei Abkühlungsschritten, α der Abkühlungsfaktor, T_{stop} die Endtemperatur und P die Menge der gefundenen Pareto-Lösungen.

1. *Ermittle eine Startlösung $x \in \mathscr{X}$ und setze $P := \{x\}$.*

2. *Setze* $T := T_0$, $i := N - 1$ *und* $k := 0$.
3. *Wähle zufällig eine Permutation* σ *aus* S_{N-i}
4. *Für* $j := 1, \ldots, N - i$
   (a) *Sei* $x' := N_\beta^{\sigma(j),\sigma(j)+i}(x)$ *eine zulässige Lösung.*
   (b) *Falls* $\mathrm{F}^3(x') < \mathrm{F}^3(x)$, *dann setze* $x := x'$.
   Sonst akzeptiere Nachbarn x' *mit Wahrscheinlichkeit* $p = e^{\frac{\mathrm{F}^3(x)-\mathrm{F}^3(x')}{T}}$.
   (c) $k := k + 1$. *Falls* $k(mod\ IA) = 0$, *setze* $T := \alpha T$.
   (d) *Falls* x' *Pareto-optimal in* P, *dann setze* $P := P \cup \{x'\}$ *und entferne alle dominierten Lösungen aus* P.
5. *Falls* $T < T_{stop}$, *stopp. Gib* P *aus.*
6. *Setze* $i := i - 1$. *Falls* $i = 0$, *setze* $i := N - 1$.
7. *Gehe zu Schritt 3.*

Die Gewichte $W = \{w_i\}_{i=1,\ldots,m}$ werden vor jedem Lauf des Simulated Annealing zufällig gemäß Gleichverteilung bestimmt. Die Ergebnisse des Algorithmus B.1 liegen sowohl in der Qualität der Lösungen als auch im Laufzeitverhalten im Bereich der Algorithmen 3.1 und 3.2. Da der Algorithmus im Vergleich mit dem Sidestep-Algorithmus deutlich mehr Parameter aufweist, ist das Tuning für die verschiedenen Problemklassen und Zielfunktionen zeitaufwendiger. Daher wurde der Algorithmus nicht bei den Versuchen verwendet.

Symbolverzeichnis

Symbol	Kurzerklärung
$\emptyset$	leere Menge
$\|A\|$	Mächtigkeit der Menge A
$\lfloor x \rfloor$	$\max\{y \in \mathbb{Z} \mid y \leq x\}$
$\lceil x \rceil$	$\min\{y \in \mathbb{Z} \mid y \geq x\}$
$[x, y]$	$\{c \mid x \leq c \leq y\}$ abgeschlossenes Intervall von x bis y
(x, y)	$\{c \mid x < c < y\}$ offenes Intervall von x bis y
$[x, y)$	$\{c \mid x \leq c < y\}$ halboffenes Intervall von x bis y
$(x, y]$	$\{c \mid x < c \leq y\}$ halboffenes Intervall von x bis y
$\{i\|E\}$	Menge aller Elemente i mit der Eigenschaft E
$a(mod\ b)$	Rest bei Division von a durch b
$\mathscr{C}$	Klassifikation
C_i	i-ter Cluster einer Klassifikation
$D(\pi, \sigma)$	$\sum \|\pi(i) - \sigma(i)\|$, Abstandsmaß in S_n
d_j	Fälligkeitstermin eines Auftrages (due date)
$\mathrm{F}^{\alpha}(s)$	Ranking-Funktion, $\alpha \in \{1, 2, 3, 4\}$
$G = (V, A, E)$	disjunktiver Graph mit Knotenmenge V, konjunktiven Kanten A und disjunktiven Kanten E
$g(\mathscr{C})$	Gütemaß einer Klassifikation
GLS	genetische lokale Suche
$h(C)$	Maß für die Homogenität eines Clusters
IS	Anzahl der zulässigen Sidesteps im Sidestep-Algorithmus
$\mathscr{J}$	Menge der Jobs in einer Job-Shop-Scheduling-Instanz
JSP	Jop-Shop Scheduling-Problem
$\mathscr{M}$	Menge der Maschinen in einer Job-Shop-Scheduling-Instanz
$\mathbb{N}_0$	$\{0, 1, 2, \ldots\}$ Menge der natürlichen Zahlen einschließlich 0
$\mathbb{N}$	$\{1, 2, 3, \ldots\}$ Menge der natürlichen Zahlen ausschließlich 0
$N_k^{ij}(\pi)$	Nachbar von π, entsteht durch Vertauschung der Vorgänge $\pi(i)$ und $\pi(j)$ bei Verwendung der Nachbarschaft $N_{k,k\in\{1,\ldots,5\}}$ in der aufsteigenden Nummerierung π
$ND(A)$	Menge der nichtdominierten Lösungen in A
$O(g(n))$	$\{f\|\exists\, c > 0, n_0 > 0 : \forall\, n > n_0 : f(n) \leq cg(n)\}$
O_W, O_S, O_C	schwache, starke und komplette Outperformance-Relation
$\mathfrak{P}(A)$	Potenzmenge von A
PG	Poolgröße, Anzahl der Lösungen im Pool der GLS
p_i	Bearbeitungsdauer eines Vorgangs
PJSP	praktisches Jop-Shop Scheduling-Problem

Symbol	Kurzerklärung
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}^+$	Menge der nichtnegativen reellen Zahlen
$\mathscr{R}$	Menge der Ressourcen in einer Job-Shop-Scheduling-Instanz
ρ^P	Korrelationskoeffizient von Pearson
ρ^S	Rangkorrelationskoeffizient von Spearman
s_i	Startzeitpunkt eines Vorgangs
S_n	Menge der Permutationen von $\{1, 2, \ldots, n\}$
$\mathscr{T}$	Menge der Vorgänge in einer Job-Shop-Scheduling-Instanz
$T(\pi, \sigma)$	minimale Anzahl von Transpositionen, um π in σ zu überführen, Abstandsmaß in S_n
$v(C_i, C_j)$	Maß für die Heterogenität zweier disjunkter Cluster C_i und C_j
$\mathscr{X}$	Lösungsraum eines Scheduling-Problems
$\mathbb{Z}$	Menge der ganzen Zahlen

Literaturverzeichnis

[1] Adams, J.; Balas, E.; Zawack, D., The shifting bottleneck procedure for job shop scheduling. *Management Science 34*, No. 3, 391-401 (1988).

[2] Althöfer, I., 13 Jahre 3-Hirn – Meine Schach-Experimente mit Mensch-Maschine-Kombinationen. Lage/Lippe (1998).

[3] Armentano, V. A.; Scrich C. R., Tabu search for minimizing total tardiness in a job shop. *Int. J. Production Economics 63*, 131-140 (2000).

[4] Balas, E.; Vazacopoulos, A., Guided Local Search with Shifting Bottleneck for Job Shop Scheduling. *Management Science 44*, No. 2, 262-275 (1998).

[5] Blażewicz, J., Scheduling computer and manufacturing processes. Springer, Berlin (1996).

[6] Blażewicz, J.; Pesch, E.; Sterna, M., The disjunctive graph machine representation of the job shop scheduling problem. *Eur. J. Oper. Res. 127*, 317-331 (2000).

[7] Bräsel, H.; Harborth, M.; Tautenhahn, T.; Willenius, P., On the hardness of the classical job shop problem. *Annals of Operations Research 92*, 265-279 (1999).

[8] Brinkkötter, W.; Brucker, P., Solving Open Benchmark Problems for the Job Shop Problem. *Journal of Scheduling 4*, 53-64, (2001).

[9] Brucker, P.; Jurisch, B.; Sievers, B., A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics 49*, 107-127 (1994).

[10] Brucker, P., Scheduling algorithms. Springer, Berlin (1995).

[11] Carlier, J., The one-machine sequencing problem. *Eur. J. Oper. Res. 11*, 42-47 (1982).

[12] Dauzère-Pérès, S., A procedure for the one-machine sequencing problem with dependent jobs. *Eur. J. Oper. Res. 81*, 579-589 (1995).

[13] Demirkol, E.; Mehta, S.; Uzsoy, R., A Computational Study of Shifting Bottleneck Procedures for Shop Scheduling Problems. *Journal of Heuristics, Winter, 3(2)*, 111-137 (1997).

[14] Demirkol, E.; Mehta, S.; Uzsoy, R., Benchmarks for shop scheduling problems. *Eur. J. Oper. Res. 109*, 137-141 (1998).

[15] Diaconis, P., Group Representations in Probability und Statistics. *Lecture Notes - Monograph Series*, Vol. 11, Institute of Mathematical Statistics, Harvard University (1988).

[16] Domschke, W.; Scholl, A.; Voß, S., Produktionsplanung. Springer, Berlin (1997).

[17] Dong H. Beak; Sang Y. Oh; Wan C. Yoon, A visualized human-computer interactive approach to job shop scheduling. *Int. J. Computer Integrated Manufacturing*, No. 1, 75-83 (1999).

[18] Ehrgott, M.; Gandibleux, X., A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum 22*, 425-460 (2000).

[19] Esquivel S.; Ferrero S.; Gallard R.; Salto C.; Alfonso H.; Schütz M., Enhanced evolutionary algorithms for single and multiobjective optimization in the job shop scheduling problem. *Knowledge-Based Systems 15*, 13-25 (2002).

[20] Georgi, G., Job Shop Scheduling in der Produktion. Wirtschaftswissenschaftliche Beiträge 111, Physica-Verlag, Heidelberg (1995).

[21] Glover, F.; Kelly, J. P.; Laguna, M., Genetic algorithms and tabu search: hybrids for optimization. *Computers Ops Res. 22*, 111-134 (1995).

[22] Guéret, C.; Prins, C., A new lower bound for the open-shop problem. *Annals of Operations Research 92*, 165-183 (1999).

[23] Graham, R. L.; Lawler, E. L.; Lenstra, J. K.; Rinnooy Kan, A. H. G. , Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics 4*, 287-326 (1979).

[24] Hansen, M. P.; Jaszkiewicz, A., Evaluating the quality of approximations to the non-dominated set. *Technical report 07/98*, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark (1998).

[25] Hartung, J., Multivariate Statistik. Oldenbourg Verlag, München (1992).

[26] Hooker, J. N., Testing Heuristics: We Have It All Wrong. *Journal of Heuristics*, 33-42 (1995).

[27] Hastings, N. A. J.; Yeh, C.-H., Job oriented production scheduling. *Eur. J. Oper. Res. 47*, 35-48 (1990).

[28] He, Z.; Yang, T.; Deal, D. E., A multiple-pass heuristic rule for job shop scheduling with due dates. *Int. J. Prod. Res. 31*, No. 11, 2677-2692 (1993).

[29] Jain, A. S., A multi-level hybrid framework for the deterministic job-shop scheduling problem. Ph. D. Thesis, Department of Applied Physics and Electronic and Mechanical Engineering, University of Dundee (1998).

[30] Jain, A. S.; Meeran, S., Deterministic job-shop scheduling: Past, present and future. *Eur. J. Oper. Res. 113*, No. 2, 390-434 (1999).

[31] Jain, A.S.; Rangaswamy, B.; Meeran, S., New and „stronger“ job-shop neighbourhoods: A focus on the method of Nowicki and Smutnicki (1996). *J. Heuristics 6*, (4), 457-480 (2000).

[32] Kolonko, M., Some new results on simulated annealing applied to the job shop scheduling problem. *Eur. J. Oper. Res. 113*, No. 1, 123-136 (1999).

[33] Kyung Sam Park; Soung Hie Kim, Tools for interactive multiattribute decisionmaking with incompletely information. *Eur. J. Oper. Res. 98*, 111-123 (1997).

[34] van Laarhoven, P. J. M.; Aarts, E. H. L.; Lenstra, J. K., Job shop scheduling by simulated annealing. *Oper. Res. 40*, No. 1, 113-125 (1992).

[35] Lee, C.-Y.; Lei, L.; Pinedo, M., Current trends in deterministic scheduling. *Annals of Operations Research 70*, 1-41 (1997).

[36] Lootsma, F. A., Multicriteria decision analysis in a decision tree. *Eur. J. Oper. Res. 101*, 442-451 (1997).

[37] Manly, B. F. J., Randomization, Bootstrap and Monte Carlo Methods in biology. Chapman and Hall, London (1997).

[38] Martin, P.; Shmoys, D. B., A new approach to computing optimal schedules for the job-shop scheduling problem. *Proceedings of the 5th International IPCO Conference*, 389-403 (1996).

[39] Matsumoto, M.; Nishimura, T., Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation Vol. 8*, No. 1, 3-30 (1998).

[40] Mattfeld, D. C.; Bierwirth, C.; Kopfer, H., A Search space analysis of the Job Shop Scheduling Problem. *Annals of Operations Research 86*, 441-453 (1999).

[41] Neumann, K.; Schwindt, C.; Zimmermann, J., Project scheduling with time windows and scarce resources. *Lecture Notes in Economics and Mathematical Systems*, Vol. 508, Springer, Berlin Heidelberg New York (2001).

[42] Norman, B. A.; Bean, J. C., A genetic algorithm methodology for complex scheduling problems. *Nav. Res. Logist. 46*, No. 2, 199-211 (1999).

[43] Nowicki, E.; Smutnicki, C., A fast taboo search algorithm for the job shop problem. *Manage. Sci. 42*, No. 6, 797-813 (1996).

[44] Nowicki, E.; Smutnicki, C. New ideas in TS for job shop scheduling. *Preprint nr 50/2001*, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland (2001).

[45] Olson, D. L., Comparison of three multicriteria methods to predict known outcomes. *Eur. J. Oper. Res. 130*, 576-587 (2001).

[46] Pezzela, F.; Merelli, E., A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *Eur. J. Oper. Res. 120*, 297-310 (2000).

[47] Ramudhin, A.; Marier, P., The generalized shifting bottleneck procedure. *Eur. J. Oper. Res. 93*, No. 1, 34-48 (1996).

[48] Reeves, C. R., Landscapes, operators and heuristic search. *Annals of Operations Research 86*, 473-490 (1999).

[49] Rose, C., Mehrheitsbildung in der Kombinatorischen Optimierung. Dissertation, Fakultät für Mathematik und Informatik, Friedrich- Schiller-Universität Jena, Jena (2001).

[50] Sanderson, P. M., The Human Planning and Scheduling Role in Advanced Manufacturing Systems: An Emerging Human Factors Domain. *Human Factors*, 31(6), 635-666 (1989).

[51] Schoof, J., Kooperative Optimierung mit kommunizierenden Genetischen Algorithmen. Dissertation, Fakultät für Mathematik und Informatik, Julius- Maximilians-Universität Würzburg, Würzburg (1998).

[52] Schutten, J. M. J, Practical job shop scheduling. *Annals of Operations Research 83*, 161-177 (1998).

[53] Singer, M., Decomposition methods for large job shops. *Computers & Operations Research 28*, 193-207 (2001).

[54] Steinhöfel, K.; Albrecht, A.; Wong, C. K., Fast parallel heuristics for the job shop scheduling problem. *Computers & Operations Research 29*, 151-169 (2002).

[55] Storer, R. H.; Wu, S.D.; Vaccari, R., Problem and heuristic space search strategies for job shop scheduling. *ORSA J. Comput. 7*, No. 4, 453-467 (1995).

[56] Subramaniam, V.; Lee, G. K.; Hong, G. S.; Wong, Y. S.; Ramesh, T., Dynamic selection of dispatching rules for job shop scheduling. *Production Planning & Control 11*, No. 1, 73-81 (2000).

[57] Taillard, E. D., Benchmarks for basic scheduling problems. *Eur. J. Oper. Res. 64*, No. 2, 278-285 (1993).

[58] Taillard, E. D., Parallel taboo search techniques for the job shop scheduling problem. *ORSA J. Comput. 6*, No. 2, 108-117 (1994).

[59] T'kindt, V; Billaut, J.-C., Multicriteria scheduling problems: a survey. *RAIRO Oper. Res. 35*, 143-163 (2001).

[60] Vaessens, R. J. M.; Aarts, E. H. L.; Lenstra, J. K., Job shop scheduling by local search. *INFORMS J. Comput. 8*, No. 3, 302-317 (1996).

[61] Ling Wang; Da-Zhong Zheng, An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research 28*, 585-596 (2001).

[62] Yamada, T.; Nakano, R., Genetic Algorithms for Jop-Shop Scheduling Problems. *Proceedings of Modern Heuristic for Decision Support*, 67-81 (1997).

Index

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel und Literatur angefertigt habe.

Ort, Datum Unterschrift des Verfassers

Lebenslauf

Henning, André

geb. 12.05.1971	Eisfeld
1977-1987	Polytechnische Oberschule Effelder
1987-1990	Berufsschule der Elektrokeramischen Werke Sonneberg
1990	Abitur, Facharbeiter
1990-1991	Wehrdienst
1991-1997	Studium der Mathematik mit Nebenfach Informatik an der Friedrich-Schiller-Universität Jena
8/1994-3/1995	Auslandsstudium an der Universität York
05.02.1997	Mathematik-Diplom
1997-2001	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Mathematische Optimierung der Friedrich-Schiller-Universität Jena
seit 1999	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Mathematische Optimierung der Bauhaus-Universität Weimar

Ort, Datum Unterschrift