NXT
NACHHALTIGKEIT UND TECHNOLOGIE
REUTLINGEN UNIVERSITY

Optimierung des Mitarbeitereinsatzes in der Lohnfertigung

Vergleichende Analyse in der metallverarbeitenden Industrie

Thesis zur Erlangung des akademischen Grades

Bachelor of Science

an der Fakultät NXT Nachhaltigkeit und Technologie
der Hochschule Reutlingen

im Studiengang

Nachhaltige Technologie

Vorgelegt von:
Jean Dujardin
Wegstraße 123, 72764 Reutlingen
Matrikelnummer 1234567

Abgabedatum: 24. Dezember 2026

Erstprüferin: Prof. Dr. Katharina Erstmal
Zweitprüfer: Prof. Dr. Kornelius Hurtig

Keine echte Abschlussarbeit

Dieses Dokument ist ein Beispiel für die Klasse nxtthesis. Es zeigt, was die Klasse typografisch kann: Titelseite, Verzeichnisse, Abkürzungen und Symbole, Tabellen, Abbildungen, Formeln, Quelltext, Literatur und Index.

Die Erpolino Metallverarbeitung GmbH, alle Personen und alle Angaben zum Unternehmen sind frei erfunden. Die Rechnungen sind echt, sie beziehen sich aber auf kein reales Unternehmen. Das Dokument ist keine Prüfungsleistung und darf nicht als solche verwendet oder zitiert werden.

Kurzfassung

Die Arbeitsvorbereitung der Erpolino Metallverarbeitung GmbH verteilt die Arbeitsgänge eines Fertigungsloses bislang nach Erfahrung auf die Mitarbeitenden. Diese Arbeit formuliert die Verteilung als Optimierungsproblem, das die Gesamtdauer minimiert, und vergleicht ein exaktes Verfahren mit einer Metaheuristik.

Für das betrachtete Los aus 24 Arbeitsgängen und 8 Mitarbeitenden liefert das exakte Verfahren in 0,7 Sekunden eine beweisbar optimale Gesamtdauer von 236,20 Minuten. Die bisherige Praxis entspricht ungefähr einer Greedy-Regel und benötigt 266,67 Minuten. Simulated Annealing (SA) erreicht im Mittel 240,10 Minuten und trifft im besten Lauf das Optimum, braucht dafür aber mehr Rechenzeit als das exakte Verfahren.

Das wesentliche Ergebnis ist damit ein negatives: In der heutigen Größenordnung lohnt sich die Heuristik nicht. Ein Skalierungsversuch zeigt, dass sich das Verhältnis erst jenseits von etwa 60 Arbeitsgängen umkehrt.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ERP	Enterprise Resource Planning
GLPK	GNU Linear Programming Kit
LPT	Longest Processing Time
MILP	Mixed Integer Linear Programming
SA	Simulated Annealing

Symbolverzeichnis

$a_i(x)$	Auslastung des Mitarbeiters i in der Zuordnung x (min)
α	Abkühlfaktor
$C_{\max}$	Gesamtdauer eines Fertigungsloses (Makespan) (min)
$E(x)$	Energie einer Lösung im Verfahren
$e_i(q)$	Effizienz des Mitarbeiters i in der Qualifikation q
g_j	Grundzeit des Arbeitsgangs j (min)
I	Menge der Mitarbeitenden
J	Menge der Arbeitsgänge
λ	Gewicht des Glättungsterms in der Energie
p_{ij}	Bearbeitungszeit des Arbeitsgangs j durch Mitarbeiter i (min)
q_j	Qualifikation, die der Arbeitsgang j verlangt
T_k	Temperatur im Schritt k
x_{ij}	Zuordnungsvariable; 1, falls i den Arbeitsgang j übernimmt
Z	Menge der zulässigen Paare aus Mitarbeiter und Arbeitsgang

1 Einleitung

1.1 Ausgangslage und Problemstellung

Die Erpolino Metallverarbeitung GmbH mit Sitz in Reutlingen fertigt Präzisionsteile im Kundenauftrag. Als Lohnfertiger bearbeitet das Unternehmen keine eigenen Serienprodukte, sondern nimmt Aufträge unterschiedlicher Auftraggeber an, die sich in Stückzahl, Toleranzanforderung und Bearbeitungsumfang deutlich voneinander unterscheiden. Ein typischer Auftrag zerfällt dabei in mehrere *Arbeitsgänge*, etwa Drehen, Fräsen, Entgraten, Messen und Dokumentieren.

Die Zuteilung dieser Arbeitsgänge an die Beschäftigten ist nicht beliebig. Die derzeit 8 Mitarbeiter der Fertigung verfügen über unterschiedliche Qualifikationen: Nicht jede Person darf jeden Arbeitsgang ausführen, und wer ihn ausführen darf, benötigt dafür nicht zwangsläufig dieselbe Zeit. Welche Person welchen Arbeitsgang übernehmen kann, lässt sich in einer *Qualifikationsmatrix* festhalten, die für jede Kombination aus Mitarbeiter und Arbeitsgang die Bearbeitungsdauer oder aber die Unzulässigkeit der Zuordnung angibt.

Die Disposition erfolgt im Unternehmen bislang manuell. Die Fertigungsleitung verteilt die anstehenden Arbeitsgänge zu Wochenbeginn auf Basis von Erfahrung und mit Unterstützung einer Tabellenkalkulation. Dieses Vorgehen ist etabliert und liefert zulässige Pläne, es besitzt jedoch eine strukturelle Schwäche: Eine von Hand erstellte Zuteilung führt regelmäßig zu einer ungleichen Auslastung. Einzelne gut qualifizierte Mitarbeiter erhalten einen überproportionalen Anteil der Arbeit, während andere Kapazität ungenutzt bleibt. Da der Auftrag erst abgeschlossen ist, wenn der letzte Arbeitsgang beendet wurde, bestimmt die am stärksten belastete Person die Gesamtdauer. Diese Gesamtdauer wird in der Ablaufplanung als *Makespan* bezeichnet (Pinedo, 2016). Der Engpass entsteht somit nicht aus fehlender Kapazität, sondern aus deren ungünstiger Verteilung.

1.2 Zielsetzung

Die vorliegende Arbeit untersucht, wie sich die Zuteilung von Arbeitsgängen an qualifikationsverschiedene Mitarbeiter mit den Mitteln der mathematischen Optimierung planen lässt. Die leitende Forschungsfrage lautet: „Mit welchem Optimierungsverfahren lässt sich die Zuordnung von Arbeitsgängen zu unterschiedlich qualifizierten Mitarbeitern bei der Erpolino Metallverarbeitung GmbH so bestimmen, dass der Makespan minimal wird, und welches Verfahren ist für die tatsächlich auftretende Problemgröße angemessen?“

Zur Beantwortung wird das Problem zunächst als Mixed Integer Linear Programming (MILP) formuliert und mit einem exakten Verfahren gelöst. Als Vergleichsmaßstab dienen zum einen die Konstruktionsheuristik Longest Processing Time (LPT) (Graham, 1969), zum anderen das

metaheuristische Verfahren SA (Kirkpatrick et al., 1983). Grundlage der Untersuchung ist eine reale Instanz des Unternehmens mit 8 Mitarbeitern und 24 Arbeitsgängen.

Ein zentrales Ergebnis sei bereits hier angedeutet: Für diese Instanzgröße liefert das exakte Verfahren in weniger als einer Sekunde eine beweisbar optimale Lösung mit einem Makespan von 236,20 Stunden. Die Heuristiken erreichen diesen Wert nicht – LPT kommt auf 265,38 Stunden, SA auf 240,74 Stunden und damit auf einen Abstand von 1,92 Prozent zum Optimum. Der Einsatz einer Heuristik lohnt sich für die betriebliche Praxis der Erpolino Metallverarbeitung GmbH folglich nicht. Erst ab einer Größenordnung von etwa 60 Arbeitsgängen kehrt sich dieses Verhältnis um, weil die Rechenzeit des exakten Verfahrens dann merklich ansteigt. Dieser Befund ist kein Mangel der Arbeit, sondern ihr eigentliches Resultat: Er begründet, welches Verfahren empfohlen wird und ab wann diese Empfehlung neu zu prüfen ist.

1.3 Abgrenzung

Die Arbeit betrachtet ausschließlich die Zuordnung von Arbeitsgängen zu Mitarbeitern unter Minimierung des Makespans. Mehrere in der Praxis durchaus relevante Aspekte bleiben bewusst unberücksichtigt.

Nicht behandelt werden *Reihenfolgeabhängigkeiten*, also technologisch bedingte Vorrangbeziehungen zwischen Arbeitsgängen. Alle Arbeitsgänge gelten als voneinander unabhängig und zu jedem Zeitpunkt ausführbar. Ebenso bleiben *Rüstzeiten* außer Betracht; die Bearbeitungsdauer eines Arbeitsgangs hängt allein von der ausführenden Person ab, nicht von deren vorangegangener Tätigkeit. Nicht Gegenstand der Arbeit ist ferner die *Maschinenbelegung*: Es wird angenommen, dass die benötigten Betriebsmittel verfügbar sind und keine zusätzliche Ressourcenkonkurrenz erzeugen. Termine und Fälligkeiten einzelner Aufträge fließen nicht in die Zielfunktion ein, weshalb Kennzahlen wie die maximale Verspätung unberücksichtigt bleiben (Pinedo, 2016). Schließlich werden Schichtmodelle, Pausenregelungen und Abwesenheiten nicht modelliert; alle Mitarbeiter stehen über den Planungshorizont hinweg gleichermaßen zur Verfügung.

Diese Einschränkungen verkleinern den Modellumfang, ohne den Kern der Problemstellung zu berühren. Sie werden in Kapitel 7 als Ansatzpunkte für weiterführende Arbeiten wieder aufgegriffen.

1.4 Aufbau der Arbeit

Kapitel 2 legt die Grundlagen der Optimierung dar. Es führt in die lineare und ganzzahlige Optimierung ein (Wolsey, 2020), ordnet das betrachtete Zuordnungsproblem komplexitätstheoretisch ein (Garey & Johnson, 1979) und stellt die eingesetzten Lösungsverfahren vor.

Kapitel 3 arbeitet das lineare Problem aus. Die Zielfunktion, die Nebenbedingungen und die Behandlung unzulässiger Zuordnungen werden entwickelt; anschließend werden das exakte Verfahren, LPT und SA methodisch gegenübergestellt.

Kapitel 4 beschreibt die Erpolino Metallverarbeitung GmbH, ihre Fertigungsstruktur und die Erhebung der Daten, aus denen die Qualifikationsmatrix der untersuchten Instanz hervorgeht.

Kapitel 5 dokumentiert die Umsetzung. Das Modell wird in MathProg formuliert und mit dem GNU Linear Programming Kit gelöst (Makhorin, 2020); die Heuristiken sowie die Auswertung entstehen in Python (Harris et al., 2020; Hunter, 2007).

Kapitel 6 analysiert die Ergebnisse. Neben dem Vergleich der Verfahren auf der realen Instanz werden künstlich vergrößerte Instanzen herangezogen, um zu bestimmen, ab welcher Problemgröße der Einsatz einer Heuristik sinnvoll wird.

Kapitel 7 fasst die Arbeit zusammen, beantwortet die Forschungsfrage und benennt offenen Forschungsbedarf.

2 Grundlagen der Optimierung

Das in dieser Arbeit betrachtete Problem – die Zuordnung von Arbeitsgängen zu Mitarbeitern bei minimaler Gesamtdauer – ist kein Einzelfall, sondern eine Ausprägung einer seit Jahrzehnten untersuchten Problemklasse. Dieses Kapitel ordnet die Aufgabenstellung in die Theorie der Ablaufplanung ein, bestimmt ihre Komplexität und stellt die drei Lösungswege vor, die im weiteren Verlauf verglichen werden: ein exaktes Verfahren, zwei Konstruktionsheuristiken und ein stochastisches Verbesserungsverfahren. Die Darstellung folgt im Wesentlichen Pinedo (2016), der die Ablaufplanung als eigenständige Disziplin umfassend aufbereitet.

2.1 Zuordnungs- und Reihenfolgeprobleme

2.1.1 Ablaufplanung als Problemklasse

Die *Ablaufplanung* beschäftigt sich mit der Frage, wie eine Menge von Aufgaben auf eine Menge von Ressourcen verteilt und zeitlich angeordnet wird, sodass ein vorgegebenes Gütemaß möglichst gut erfüllt ist. Die Ressourcen heißen in der Literatur traditionell *Maschinen*, die Aufgaben *Jobs* – eine Sprechweise, die aus der Fertigungssteuerung stammt und auch dann beibehalten wird, wenn es sich, wie im vorliegenden Fall, um Menschen handelt. Diese terminologische Übertragung ist unproblematisch, solange man sich vergegenwärtigt, dass das Modell von allen Eigenschaften absieht, die über die Bearbeitungsdauer und die Zulässigkeit einer Zuordnung hinausgehen.

Zu unterscheiden sind zwei Fragestellungen, die in der Praxis oft verschmelzen. Ein *Zuordnungsproblem* fragt allein danach, *wer* welche Aufgabe übernimmt; ein *Reihenfolgeproblem* fragt zusätzlich, *wann* das geschieht. Sind die Aufgaben voneinander unabhängig, existieren keine Rüstzeiten und keine Fälligkeitstermine, so zerfällt das Reihenfolgeproblem: Die Reihenfolge, in der ein Mitarbeiter seine zugeteilten Arbeitsgänge abarbeitet, ist für die Gesamtdauer ohne Belang, weil sich die Summe seiner Bearbeitungszeiten dadurch nicht ändert. Es bleibt ein reines Zuordnungsproblem. Genau diese Situation liegt der vorliegenden Arbeit zugrunde, und sie ist der Grund dafür, dass sich das Problem so kompakt formulieren lässt.

2.1.2 Die Drei-Feld-Notation

Für die Klassifikation von Problemen der Ablaufplanung hat sich die von Graham (1969) eingeführte und später erweiterte *Drei-Feld-Notation* durchgesetzt. Ein Problem wird durch ein Tripel

$$\alpha \mid \beta \mid \gamma \tag{2.1}$$

beschrieben. Das Feld α benennt die *Maschinenumgebung*, das Feld β sammelt Nebenbedingungen und Sonderregeln, und γ gibt die *Zielfunktion* an. Die Notation ist deshalb so nützlich,

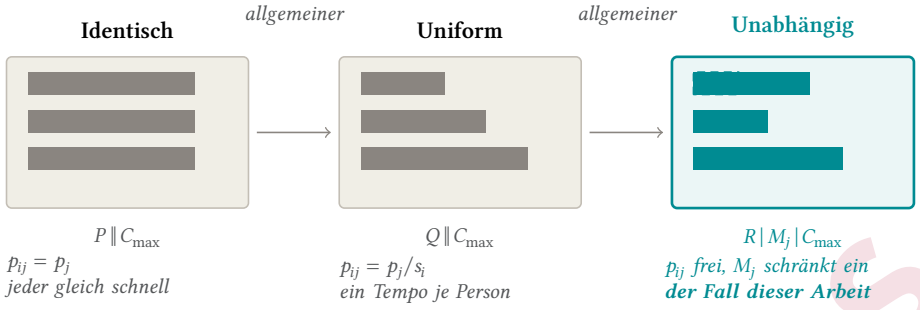


Abbildung 2.1: Einordnung des Problems: identische, uniforme und unabhängige parallele Maschinen

weil sie eine Aufgabenstellung in wenigen Zeichen so präzise festlegt, dass sich unmittelbar in der Literatur nachschlagen lässt, was über sie bekannt ist – insbesondere, ob sie effizient lösbar ist.

Für die Maschinenumgebung paralleler Maschinen sind drei Stufen zu unterscheiden, die in Abbildung 2.1 einander gegenübergestellt sind. Bei *identischen parallelen Maschinen* (P) benötigt jede Maschine für denselben Job dieselbe Zeit p_j. Bei *uniformen parallelen Maschinen* (Q) besitzt jede Maschine i eine Geschwindigkeit s_i, sodass sich die Bearbeitungsdauer als $p_{ij} = p_j / s_i$ ergibt; die Maschinen unterscheiden sich also nur um einen einzigen Faktor, und wer bei einem Job schneller ist, ist es bei allen. Bei *unabhängigen parallelen Maschinen* (R) schließlich ist p_{ij} eine beliebige Matrix: Ein Mitarbeiter kann bei einem Arbeitsgang der schnellste und bei einem anderen der langsamste sein.

2.1.3 Das Problem $R \mid M_j \mid C_{\max}$

Das Problem dieser Arbeit trägt die Signatur $R \mid M_j \mid C_{\max}$. Die drei Felder sind wie folgt zu lesen.

Das R steht für unabhängige parallele Maschinen. Diese Wahl ist keine vorsorgliche Verallgemeinerung, sondern durch den Gegenstand erzwungen: Die betrachteten Mitarbeiter unterscheiden sich nicht in einer pauschalen Arbeitsgeschwindigkeit, sondern in ihrer Erfahrung mit jeweils bestimmten Arbeitsgängen. Ein uniformes Modell würde die Daten sichtbar verfälschen.

Das M_j bezeichnet die *Maschinenzulässigkeit*: Für jeden Arbeitsgang j existiert eine Menge M_j derjenigen Mitarbeiter, die ihn überhaupt ausführen dürfen. Diese Restriktion bildet die *Qualifikation* ab. Wer für einen Arbeitsgang nicht qualifiziert ist, kommt für ihn nicht in Betracht – nicht zu höheren Kosten, sondern gar nicht. Modelltechnisch ist es daher sauberer und zugleich effizienter, unzulässige Paare erst gar nicht anzulegen, statt sie über eine große Strafkonstante auszuschließen.

Das $C_{\max}$ schließlich ist der *Makespan*, also der Zeitpunkt, zu dem auch der letzte Mitarbeiter fertig ist. Bezeichnet $x_{ij} \in \{0, 1\}$ die Entscheidung, ob Mitarbeiter i den Arbeitsgang j

übernimmt, so ist

$$C_{\max} = \max_i \sum_j p_{ij}\, x_{ij}. \tag{2.2}$$

Der Makespan misst nicht die geleistete Arbeit, sondern die verstrichene Zeit. Er wird allein vom am stärksten belasteten Mitarbeiter bestimmt – dem *Engpass*. Diese Eigenschaft ist charakteristisch und hat weitreichende Folgen: Sie macht den Makespan zu einem sinnvollen Maß für die Frage „Wann ist der Auftrag abgeschlossen?“, sie führt aber, wie in Abschnitt 2.5 zu zeigen sein wird, zu erheblichen Schwierigkeiten bei Verfahren, die sich von lokalen Verbesserungen leiten lassen.

2.2 Komplexität

2.2.1 NP-Schwere

Die naheliegende Idee, alle Zuordnungen aufzuzählen, scheitert an ihrer Anzahl. Bei n Arbeitsgängen und m zulässigen Mitarbeitern je Arbeitsgang existieren bis zu m^n Zuordnungen; der Suchraum wächst exponentiell. Schon bei moderaten Instanzgrößen übersteigt die vollständige Enumeration jede vertretbare Rechenzeit.

Dass es sich dabei nicht um einen Mangel an Einfallsreichtum handelt, sondern um eine Eigenschaft des Problems, zeigt die Komplexitätstheorie. Bereits der Spezialfall $P2 \parallel C_{\max}$ – zwei identische Maschinen, keine Nebenbedingungen – ist *NP-schwer*; er enthält das Problem *Partition*, eines der klassischen NP-vollständigen Probleme im Katalog von Garey und Johnson (1979), als Entscheidungsvariante. Da $R \mid M_j \mid C_{\max}$ diesen Spezialfall umfasst, ist es mindestens ebenso schwer. Nach gegenwärtigem Kenntnisstand existiert somit kein Algorithmus, der für jede Instanz in polynomieller Zeit eine beweisbar optimale Lösung liefert.

Es lohnt, die Tragweite dieser Aussage nicht zu überschätzen. NP-Schwere ist eine Aussage über das Verhalten im schlechtesten Fall und über beliebig große Instanzen. Sie schließt nicht aus, dass konkrete Instanzen praktischer Größe exakt und schnell lösbar sind – moderne Löser profitieren erheblich von Struktur, insbesondere von der durch M_j erzeugten Ausdünnung des Modells. Die NP-Schwere ist also kein Verbot, exakte Verfahren einzusetzen; sie ist eine Warnung davor, sich auf ihre Skalierbarkeit zu verlassen.

2.2.2 Approximierbarkeit

Wenn Optimalität nicht garantiert werden kann, stellt sich die Frage nach garantierter Nähe zum Optimum. Ein Verfahren heißt *ρ-Approximation*, wenn es in polynomieller Zeit eine Lösung liefert, deren Zielwert höchstens um den Faktor ρ über dem Optimum liegt.

Das grundlegende Resultat für unabhängige parallele Maschinen stammt von Lenstra et al. (1990). Sie zeigen, dass sich für $R \parallel C_{\max}$ eine *2-Approximation* in polynomieller Zeit konstruieren lässt. Das Verfahren beruht auf einem *Rundungsverfahren*: Zunächst wird die *LP-Relaxation* gelöst, deren Lösung Arbeitsgänge anteilig auf mehrere Maschinen verteilen darf; anschließend werden die gebrochenen Anteile so gerundet, dass der Fehler beschränkt bleibt. Dieselbe Arbeit zeigt zudem, dass keine ρ-Approximation mit $\rho < 3/2$ existieren kann, sofern nicht $\mathrm{P} = \mathrm{NP}$ gilt. Zwischen $3/2$ und 2 klafft bis heute eine Lücke.

Für die vorliegende Arbeit sind diese Ergebnisse vor allem als Maßstab von Bedeutung. Sie belegen, dass die Klasse R nicht beliebig unzugänglich ist, sie zeigen aber auch, dass die theoretischen Garantien großzügig ausfallen: Ein Faktor 2 bedeutet im schlechtesten Fall die doppelte Gesamtdauer. Die in dieser Arbeit eingesetzten Heuristiken tragen, wie sich zeigen wird, überhaupt keine Gütegarantie – liefern aber praktisch weit bessere Ergebnisse, als es die Theorie im schlechtesten Fall zusichern würde.

2.3 Exakte Verfahren

2.3.1 Ganzzahlige lineare Optimierung

Ein Modell der *ganzzahligen linearen Optimierung* (MILP) beschreibt ein Problem durch eine lineare Zielfunktion und lineare Nebenbedingungen über teils ganzzahligen Variablen (Wolsey, 2020). Das hier betrachtete Problem lässt sich so formulieren:

$$\min \quad C_{\max} \tag{2.3}$$

$$\text{u. d. N.} \quad \sum_{i \in M_j} x_{ij} = 1 \qquad \text{für alle } j \tag{2.4}$$

$$\sum_{j \,:\, i \in M_j} p_{ij}\, x_{ij} \le C_{\max} \qquad \text{für alle } i \tag{2.5}$$

$$x_{ij} \in \{0, 1\}. \tag{2.6}$$

Bedingung (2.4) stellt sicher, dass jeder Arbeitsgang genau einmal und nur an einen qualifizierten Mitarbeiter vergeben wird. Die Ungleichungen (2.5) verdienen besondere Beachtung: Sie fordern lediglich, dass $C_{\max}$ mindestens so groß ist wie die Auslastung jedes einzelnen Mitarbeiters. Erst im Zusammenspiel mit der Minimierung in (2.3) erzwingen sie die Gleichheit aus (2.2) – ein Standardkniff, mit dem sich das nichtlineare Maximum linear ausdrücken lässt.

2.3.2 LP-Relaxation und Schranken

Lässt man die Ganzzahligkeitsforderung (2.6) fallen und ersetzt sie durch $0 \le x_{ij} \le 1$, so entsteht die *LP-Relaxation*. Sie ist in polynomieller Zeit lösbar. Da jede zulässige Lösung des ganzzahligen Problems auch für die Relaxation zulässig ist, liefert deren Optimalwert eine *untere Schranke* für das Optimum. Jede konkrete Zuordnung wiederum liefert eine *obere Schranke*. Die Differenz zwischen beiden heißt *Dualitätslücke*; solange sie positiv ist, kann das Optimum irgendwo dazwischen liegen. Schließt sie sich, ist die Optimalität bewiesen.

Neben der LP-Relaxation existieren einfachere, kombinatorisch begründete Schranken. Für den Makespan ist etwa

$$C_{\max} \;\ge\; \frac{1}{m} \sum_{j} \min_{i \in M_j} p_{ij} \tag{2.7}$$

gültig: Selbst wenn jeder Arbeitsgang von dem für ihn schnellsten Mitarbeiter erledigt und die Last vollkommen gleichmäßig verteilt würde, wäre diese Zeit nicht zu unterbieten. Eine solche Schranke ist in aller Regel nicht erreichbar – sie ignoriert, dass die schnellsten Mitarbeiter einander im Weg stehen –, aber sie ist ohne nennenswerten Aufwand zu berechnen und erlaubt eine erste Einschätzung der Güte jeder gefundenen Lösung.

2.3.3 Branch-and-Bound

Das Standardverfahren zur exakten Lösung ganzzahliger Modelle ist *Branch-and-Bound*, das auf Land und Doig (1960) zurückgeht und bei Nemhauser und Wolsey (1988) ausführlich dargestellt ist. Es kombiniert zwei Ideen. Beim *Verzweigen* wird das Problem an einer Variablen mit gebrochenem Wert in Teilprobleme zerlegt, etwa in die Fälle $x_{ij} = 0$ und $x_{ij} = 1$; die Vereinigung der Teilprobleme deckt den ursprünglichen Lösungsraum vollständig ab. Beim *Beschränken* wird für jedes Teilproblem die Relaxation gelöst. Liegt deren Wert bereits über der besten bisher bekannten Lösung, so kann das Teilproblem samt allen seinen Nachfolgern verworfen werden, ohne es zu durchsuchen – es kann dort nichts Besseres mehr geben.

Genau in diesem Verwerfen liegt die Kraft des Verfahrens, und genau darin liegt auch die Bedeutung des Begriffs „beweisbar optimal“. Terminiert Branch-and-Bound regulär, so hat es nicht etwa alle Lösungen betrachtet; vielmehr hat es für jede nicht betrachtete Region gezeigt, dass sie keine bessere Lösung enthalten *kann*. Das Ergebnis ist damit nicht eine Lösung, die man nach längerem Suchen nicht mehr verbessern konnte, sondern eine Lösung mit mathematischem Beweis ihrer Optimalität. Dieser Unterschied ist der entscheidende Vorzug exakter Verfahren gegenüber allen Heuristiken: Sie liefern nicht nur eine Antwort, sondern zugleich die Gewissheit, dass es keine bessere gibt. Erkauft wird er mit einer Laufzeit, die im schlechtesten Fall exponentiell bleibt. In dieser Arbeit wird dafür der freie MILP-Löser GLPK eingesetzt (Makhorin, 2020).

2.4 Konstruktionsheuristiken

Eine *Konstruktionsheuristik* baut eine Lösung in einem Durchgang auf, ohne sie nachträglich zu verbessern. Sie trifft jede Entscheidung nach einer festen Regel und nimmt sie nie zurück. Der Reiz solcher Verfahren liegt in ihrer Geschwindigkeit und Nachvollziehbarkeit; ihr Preis ist das Fehlen jeder Gütegarantie.

Das *Greedy-Verfahren* betrachtet die Arbeitsgänge in ihrer Eingangsreihenfolge und weist jeden demjenigen qualifizierten Mitarbeiter zu, der ihn am frühesten abschließen würde. Es entscheidet also stets lokal optimal, ohne die Folgen zu bedenken. Der bekannte Schwachpunkt ist die Behandlung langer Arbeitsgänge: Trifft ein solcher erst spät ein, sind alle Mitarbeiter bereits belegt, und er verlängert den Makespan unmittelbar.

Die *LPT-Regel* (LPT) setzt genau hier an und dreht die Reihenfolge um: Die Arbeitsgänge werden absteigend nach ihrer Dauer sortiert und in dieser Reihenfolge vergeben. Die langen Brocken werden zuerst verteilt, solange noch Spielraum besteht; die kurzen dienen am Ende dem Feinausgleich. Bei unabhängigen Maschinen ist dabei zu klären, welche Dauer für die Sortierung maßgeblich ist, da p_{ij} von der Zuordnung abhängt – in dieser Arbeit wird die kürzestmögliche Dauer $\min_{i \in M_j} p_{ij}$ verwendet.

Für identische parallele Maschinen hat Graham (1969) für die LPT-Regel die klassische Schranke

$$\frac{C_{\max}^{\mathrm{LPT}}}{C_{\max}^{*}} \leq \frac{4}{3} - \frac{1}{3m} \tag{2.8}$$

bewiesen. Sie besagt, dass LPT auf $P \,||\, C_{\max}$ niemals mehr als rund ein Drittel über dem Optimum liegt – ein bemerkenswert starkes Ergebnis für ein derart einfaches Verfahren.

Es ist jedoch entscheidend, festzuhalten, dass diese Schranke im vorliegenden Fall *nicht* gilt. Grahams Beweis nutzt an zentraler Stelle aus, dass die Maschinen identisch sind: Nur dann besitzt jeder Job eine wohldefinierte, von der Maschine unabhängige Dauer p_j, nur dann ist die Sortierung nach Dauer eindeutig, und nur dann lässt sich der Beitrag des zuletzt beendeten Jobs gegen die Durchschnittslast abschätzen. Bei unabhängigen Maschinen bricht diese Argumentation zusammen. Die Sortierung nach der kürzestmöglichen Dauer ist eine Konvention und keine Eigenschaft des Jobs; ein nach ihr kurzer Arbeitsgang kann für den Mitarbeiter, der ihn tatsächlich erhält, sehr lang sein. Die LPT-Regel bleibt für $R \mid M_j \mid C_{\max}$ eine plausible und in der Praxis brauchbare Faustregel – aber eben nur das. Sie ohne diesen Vorbehalt mit Grahams Schranke zu versehen, wäre ein Fehlschluss, der in der Anwendungsliteratur nicht selten anzutreffen ist.

2.5 Simulated Annealing

2.5.1 Herkunft und Grundidee

Zwischen den schnellen, aber garantielosen Konstruktionsheuristiken und den exakten, aber teuren Verfahren stehen die *Metaheuristik*en. Sie verbessern eine vorhandene Lösung schrittweise, ohne den Lösungsraum systematisch zu erschöpfen. Das in dieser Arbeit eingesetzte Verfahren ist *Simulated Annealing* (SA).

Sein Name und seine Funktionsweise stammen aus der Metallurgie. Beim *Ausglühen* eines Werkstücks wird das Metall stark erhitzt und anschließend langsam abgekühlt. Bei hoher Temperatur besitzen die Atome genug Energie, um ihre Gitterplätze zu verlassen; mit sinkender Temperatur ordnen sie sich zunehmend in einem energiearmen, regelmäßigen Kristallgitter an. Schreckt man das Werkstück dagegen rasch ab, gefrieren die Fehlstellen ein, und das Material bleibt spröde. Die langsame Abkühlung ist es, die den energetisch günstigen Zustand ermöglicht.

Metropolis et al. (1953) entwickelten ein Verfahren, um dieses thermodynamische Verhalten zu simulieren. Unabhängig voneinander erkannten Kirkpatrick et al. (1983) und Černý (1985), dass sich das Vorgehen auf beliebige Optimierungsprobleme übertragen lässt, wenn man die Zielfunktion als *Energie* und einen Kontrollparameter als *Temperatur* auffasst. Eine ausführliche theoretische Behandlung findet sich bei van Laarhoven und Aarts (1987).

2.5.2 Das Metropolis-Kriterium

Der Kern des Verfahrens ist die Regel, nach der über die Annahme eines Kandidaten entschieden wird. Aus der aktuellen Lösung wird eine Nachbarlösung erzeugt und die Energiedifferenz ΔE bestimmt. Ist $\Delta E \leq 0$, die Lösung also nicht schlechter, wird sie stets angenommen. Ist sie schlechter, wird sie nicht etwa verworfen, sondern mit der Wahrscheinlichkeit

$$P(\text{Annahme}) = \exp\left(-\frac{\Delta E}{T}\right) \qquad \text{für } \Delta E > 0 \tag{2.9}$$

dennoch übernommen. Dieses *Metropolis-Kriterium* ist der eigentliche Grund für die Leistungsfähigkeit des Verfahrens. Ein Verfahren, das nur Verbesserungen akzeptiert, bleibt im ersten

lokalen Optimum stehen, das es erreicht. Die gelegentliche Annahme einer Verschlechterung erlaubt es, ein solches Tal wieder zu verlassen.

Die Temperatur T steuert dabei die Bereitschaft zur Verschlechterung. Bei großem T geht der Exponent gegen null und die Wahrscheinlichkeit gegen eins: Das Verfahren nimmt fast alles an und irrt nahezu zufällig durch den Suchraum. Bei kleinem T geht die Wahrscheinlichkeit gegen null, und das Verfahren verhält sich wie eine reine lokale Suche. Bemerkenswert ist ferner, dass (2.9) nicht nur von T, sondern auch von der Größe von ΔE abhängt: Kleine Verschlechterungen werden deutlich bereitwilliger hingenommen als große.

2.5.3 Abkühlung

Die *Abkühlung* überführt das Verfahren von der Exploration in die Verfeinerung. Praktisch verbreitet ist die *geometrische Abkühlung*

$$T_{k+1} = \alpha \cdot T_k, \qquad 0 < \alpha < 1, \tag{2.10}$$

bei der die Temperatur in jedem Schritt um einen festen Faktor sinkt und somit $T_k = T_0 \cdot \alpha^k$ gilt. Gibt man Anfangstemperatur T_0, Endtemperatur T_N und Schrittzahl N vor, so ergibt sich $\alpha = (T_N/T_0)^{1/N}$. Diese Parametrisierung ist der direkten Wahl von α vorzuziehen, weil sie an inhaltlich interpretierbaren Größen ansetzt.

Theoretisch lässt sich zeigen, dass SA unter einer hinreichend langsamen, logarithmischen Abkühlung mit Wahrscheinlichkeit eins gegen das globale Optimum konvergiert (van Laarhoven & Aarts, 1987). Dieses Resultat ist praktisch allerdings ohne Wert, da die erforderlichen Laufzeiten die vollständige Enumeration übersteigen. In der Anwendung kühlt man schneller ab und verzichtet damit auf jede Garantie – SA liefert eine gute Lösung, aber keinen Beweis.

2.5.4 Nachbarschaft

Die *Nachbarschaft* legt fest, welche Lösungen aus einer gegebenen Lösung in einem Schritt erreichbar sind. Ihre Gestaltung ist die eigentliche problemspezifische Entwurfsentscheidung: Ist sie zu klein, zerfällt der Suchraum in unerreichbare Gebiete; ist sie zu groß, verliert der Schritt seinen lokalen Charakter und das Verfahren entartet zur Zufallssuche.

In dieser Arbeit werden die beiden in Abbildung 2.2 dargestellten Operatoren kombiniert. Das *Verschieben* nimmt einem Mitarbeiter einen Arbeitsgang ab und übergibt ihn einem anderen qualifizierten Mitarbeiter. Das *Tauschen* kreuzt zwei Arbeitsgänge zwischen ihren Mitarbeitern, sofern beide für den jeweils anderen Arbeitsgang qualifiziert sind. Beide werden benötigt: Das Verschieben allein genügt nicht, denn sind zwei Mitarbeiter gleichermaßen ausgelastet, so verlagert jedes Verschieben das Problem nur, während allein ein Tausch die Last umverteilen kann, ohne sie an anderer Stelle zu erhöhen.

2.5.5 Das Plateauproblem

Abschließend ist auf eine Schwierigkeit hinzuweisen, die sich unmittelbar aus der Struktur des Makespans ergibt und für die Auslegung des Verfahrens von erheblicher Bedeutung ist. Nach

Abbildung 2.2: Die beiden Nachbarschaftsoperatoren: Verschieben und Tauschen

(2.2) ist $C_{\max}$ ein Maximum. Er ändert sich folglich nur dann, wenn ein Zug gerade den Engpass-Mitarbeiter betrifft. Jede Umverteilung zwischen zwei nicht ausgelasteten Mitarbeitern lässt die Zielfunktion vollkommen unverändert.

Verwendet man den Makespan unmittelbar als Energie, so ist die Suchlandschaft daher von ausgedehnten *Plateaus* durchzogen: Weite Bereiche besitzen exakt denselben Zielwert, und $\Delta E = 0$ liefert dem Verfahren keinerlei Hinweis, welche Richtung lohnend ist. Das Verfahren irrt über diese Ebenen, ohne geführt zu werden. Dieses *Plateauproblem* tritt bei allen Max- und Min-Zielfunktionen auf und ist keine Eigenheit des vorliegenden Falls.

Ein bewährter Ausweg besteht darin, die Energie um einen kleinen Zusatzterm zu ergänzen, der auch dann reagiert, wenn der Makespan es nicht tut. Bezeichnet a_i die Auslastung des Mitarbeiters i, so bietet sich

$$E = \max_i a_i \; + \; \lambda \sqrt{\sum_i a_i^2} \tag{2.11}$$

an. Der zweite Term, die euklidische Norm des Auslastungsvektors, sinkt bereits dann, wenn die Last gleichmäßiger verteilt wird – auch ohne dass der Makespan fällt. Damit erhält das Verfahren auf dem Plateau ein Gefälle, dem es folgen kann. Entscheidend ist die Wahl des Gewichts λ: Es muss klein genug sein, dass der Zusatzterm die Rangfolge zweier Lösungen mit verschiedenem Makespan niemals umkehrt. Er dient allein der Wegfindung, nicht der Bewertung. Die eigentliche Zielgröße bleibt der Makespan, und ausschließlich nach ihm wird die beste gefundene Lösung ausgewählt und berichtet. Welchen Unterschied diese Modifikation praktisch ausmacht, wird im Ergebniskapitel quantifiziert.

3 Ausarbeitung des linearen Problems

Kapitel 2 hat die Werkzeuge bereitgestellt, mit denen sich ein Zuordnungsproblem als ganzzahliges lineares Programm fassen lässt. Das vorliegende Kapitel wendet diese Werkzeuge auf die Fragestellung der Erpolino Metallverarbeitung GmbH an. Es entwickelt zunächst die Annahmen, unter denen die betriebliche Situation überhaupt einer mathematischen Beschreibung zugänglich wird, formuliert anschließend Mengen, Parameter und Variablen, stellt Zielfunktion und Nebenbedingungen auf und diskutiert die Eigenschaften der gewonnenen Formulierung. Den Abschluss bildet eine methodische Gegenüberstellung der Verfahren, die zur Lösung des Modells in Frage kommen.

3.1 Modellannahmen

Ein Modell ist niemals ein Abbild der Wirklichkeit, sondern eine bewusste Vereinfachung. Welche Vereinfachungen zulässig sind, entscheidet sich daran, ob sie den Kern der Fragestellung berühren. Für die hier untersuchte Zuordnung von Arbeitsgängen zu Mitarbeitern werden fünf Annahmen getroffen, die im Folgenden begründet werden.

Erstens bleiben *Rüstzeiten* unberücksichtigt. Die Bearbeitungsdauer eines Arbeitsgangs hängt allein davon ab, welche Person ihn ausführt, nicht davon, welche Tätigkeit dieser Person unmittelbar vorausging. In der Fertigung der Erpolino Metallverarbeitung GmbH fallen Rüstvorgänge zwar an, sie sind jedoch überwiegend an das Betriebsmittel und nicht an die Reihenfolge der Arbeitsgänge gebunden und daher bereits in den erhobenen Grundzeiten enthalten. Wären reihenfolgeabhängige Rüstzeiten zu berücksichtigen, so entstünde ein Problem mit Rundreisecharakter, dessen Komplexität die des Zuordnungsproblems deutlich übersteigt (Pinedo, 2016).

Zweitens werden keine *Reihenfolgeabhängigkeiten* modelliert. Alle Arbeitsgänge gelten als voneinander unabhängig und zu jedem Zeitpunkt ausführbar. Diese Annahme ist der eigentliche Grund dafür, dass das Problem ein reines Zuordnungsproblem bleibt und keine zeitliche Einplanung erfordert. Sobald feststeht, welche Person welche Arbeitsgänge übernimmt, ist die Reihenfolge innerhalb einer Person für die Zielgröße bedeutungslos, denn die Summe der Bearbeitungszeiten ändert sich durch Umsortieren nicht. Damit entfallen die Zeitpunktvariablen, die ein Ablaufplanungsmodell sonst mit sich brächte.

Drittens sind die Aufgaben *unteilbar*. Ein Arbeitsgang wird vollständig von genau einer Person ausgeführt; eine Aufteilung auf zwei Mitarbeiter oder ein Wechsel während der Bearbeitung ist ausgeschlossen. Diese Annahme entspricht der betrieblichen Praxis, in der ein Arbeitsgang zusammen mit dem zugehörigen Werkstück und dem Prüfprotokoll einer Person zugewiesen wird. Sie ist zugleich die Ursache dafür, dass das Modell ganzzahlige Variablen benötigt und nicht als lineares Programm gelöst werden kann. Ließe man Teilbarkeit zu, so wäre das Problem in polynomieller Zeit lösbar; die *Ganzzahligkeitsforderung* ist also nicht Beiwerk,

sondern der Grund für die Schwierigkeit des Problems (Garey & Johnson, 1979).

Viertens stehen alle Mitarbeiter über den Planungshorizont hinweg gleichermaßen zur Verfügung. Schichtmodelle, Pausen, Urlaub und Krankheit bleiben außer Betracht. Der Planungshorizont ist eine Kalenderwoche, für die die Arbeitsvorbereitung ein Fertigungslos freigibt; innerhalb dieser Woche gilt jede Person als ganztags einsatzbereit.

Fünftens ist die *Effizienz* einer Person je Qualifikation konstant. Lerneffekte, Ermüdung und Tagesform werden nicht abgebildet. Die Bearbeitungszeit ergibt sich damit deterministisch aus der Grundzeit des Arbeitsgangs und einem personenbezogenen Effizienzfaktor. Stochastische Schwankungen der Bearbeitungsdauer wären grundsätzlich modellierbar, führten aber auf ein stochastisches Optimierungsproblem und damit auf eine andere Klasse von Verfahren.

Diese Annahmen idealisieren die Fertigung erheblich. Sie lassen jedoch genau jenen Effekt unangetastet, um den es der Fertigungsleitung geht: die ungleiche Auslastung, die entsteht, wenn Qualifikationen ungleich verteilt sind.

3.2 Mengen, Parameter und Variablen

Sei I die Menge der Mitarbeiter der Fertigung und J die Menge der Arbeitsgänge des Planungshorizonts. Für die untersuchte Instanz gilt $|I| = 8$ und $|J| = 24$. Jeder Arbeitsgang $j \in J$ verlangt genau eine der 6 Qualifikationen; jeder Mitarbeiter $i \in I$ beherrscht eine Teilmenge davon.

Aus dieser Qualifikationsstruktur ergibt sich die zentrale Menge des Modells. Es bezeichne

$$Z \subseteq I \times J \tag{3.1}$$

die Menge der *zulässigen Paare*, also derjenigen Kombinationen (i, j), für die Mitarbeiter i die von Arbeitsgang j geforderte Qualifikation besitzt. In der Scheduling-Notation entspricht dies der Restriktion M_j, die für jeden Auftrag die Menge der zulässigen Maschinen einschränkt (Pinedo, 2016). Die Menge Z ist keine bloße Schreibvereinfachung, sondern trägt die gesamte Information über die *Qualifikationsschranken*; ihre Rolle wird in Abschnitt 3.4 ausführlich behandelt.

Für jedes zulässige Paar $(i, j) \in Z$ ist die *Bearbeitungszeit* $p_{ij} > 0$ in Minuten definiert. Sie wird nicht frei erhoben, sondern aus zwei Größen berechnet: der Grundzeit g_j des Arbeitsgangs, die die Arbeitsvorbereitung für eine Normalleistung vorgibt, und dem Effizienzfaktor $e_i(q_j)$, mit dem Mitarbeiter i die von Arbeitsgang j geforderte Qualifikation q_j ausübt. Es gilt

$$p_{ij} = \frac{g_j}{e_i(q_j)} \qquad \text{für alle } (i, j) \in Z. \tag{3.2}$$

Ein Effizienzfaktor von 1,00 bedeutet Normalleistung, ein Wert über 1,00 eine überdurchschnittliche und ein Wert darunter eine unterdurchschnittliche Leistung. Ein Mitarbeiter mit dem Faktor 1,25 benötigt für einen Arbeitsgang mit der Grundzeit 95 Minuten folglich 76 Minuten. Wichtig ist, dass die Effizienzfaktoren nicht in einer festen Rangfolge stehen: Eine Person kann beim Drehen überdurchschnittlich und beim Schleifen unterdurchschnittlich sein. Genau deshalb liegt kein Problem mit gleichförmigen Maschinen vor, bei dem sich alle Bearbeitungszeiten aus einer einzigen personenbezogenen Geschwindigkeit ergäben, sondern ein Problem

mit *unabhängigen parallelen Maschinen*, in der Kendall-artigen Notation $R \mid M_j \mid C_{\max}$ (Pinedo, 2016).

Das Modell benötigt zwei Arten von Variablen. Zum einen die *Binärvariable*

$$x_{ij} \in \{0, 1\} \qquad \text{für alle } (i, j) \in Z, \tag{3.3}$$

die den Wert 1 annimmt, wenn Mitarbeiter i den Arbeitsgang j übernimmt, und sonst 0. Zum anderen die stetige Variable

$$C_{\max} \geq 0, \tag{3.4}$$

die den *Makespan* abbildet, also den Zeitpunkt, zu dem auch der zuletzt fertig werdende Mitarbeiter seine Arbeit abgeschlossen hat. Bemerkenswert ist, dass $C_{\max}$ keine Entscheidungsvariable im eigentlichen Sinne ist: Ihr Wert ist durch die Belegung der x_{ij} vollständig bestimmt. Sie dient allein dazu, das Maximum über die Auslastungen, das für sich genommen keine lineare Funktion ist, in eine lineare Formulierung zu überführen.

3.3 Zielfunktion und Nebenbedingungen

Das vollständige Modell lautet

$$\begin{aligned}
\min \quad & C_{\max} && && (3.5)\\
\text{u. d. N.} \quad & \sum_{i \,:\, (i,j) \in Z} x_{ij} = 1 && \text{für alle } j \in J, && (3.6)\\
& \sum_{j \,:\, (i,j) \in Z} p_{ij}\, x_{ij} \leq C_{\max} && \text{für alle } i \in I, && (3.7)\\
& x_{ij} \in \{0, 1\} && \text{für alle } (i, j) \in Z, && (3.8)\\
& C_{\max} \geq 0. && && (3.9)
\end{aligned}$$

Die Zielfunktion (3.5) minimiert den Makespan. Sie besteht aus einer einzigen Variablen und ist damit denkbar einfach; die gesamte Modellierungsarbeit steckt in den Nebenbedingungen.

Die *Vergabebedingung* (3.6) stellt sicher, dass jeder Arbeitsgang genau einmal vergeben wird. Die Summation läuft ausschließlich über diejenigen Mitarbeiter, für die das Paar (i, j) zulässig ist. Die Gleichung verlangt zweierlei zugleich: Kein Arbeitsgang bleibt liegen, und kein Arbeitsgang wird doppelt ausgeführt. Da die Menge Z zu jedem $j \in J$ mindestens einen Mitarbeiter enthält – jede Qualifikation ist in der Fertigung mehrfach vorhanden –, ist das Modell zulässig.

Die *Auslastungsbedingung* (3.7) verknüpft die Zuordnungsvariablen mit dem Makespan. Die linke Seite summiert die Bearbeitungszeiten aller Arbeitsgänge, die Mitarbeiter i übernimmt, und beschreibt damit dessen Auslastung. Für jeden Mitarbeiter wird gefordert, dass diese Auslastung den Wert $C_{\max}$ nicht übersteigt. Es gibt genau 8 solcher Bedingungen, eine je Mitarbeiter, und 24 Vergabebedingungen.

Die Formulierung enthält weder Zeitpunkte noch Reihenfolgevariablen. Sie ist damit erheblich kleiner als ein allgemeines Ablaufplanungsmodell und verdankt diese Kompaktheit unmittelbar den Annahmen aus Abschnitt 3.1.

3.4 Eigenschaften des Modells

Vier Eigenschaften der Formulierung verdienen eine genauere Betrachtung, weil sie weniger selbstverständlich sind, als sie auf den ersten Blick erscheinen.

Die Ungleichung in der Auslastungsbedingung

Warum steht in (3.7) ein Kleiner-gleich-Zeichen und kein Gleichheitszeichen? Die Frage ist berechtigt, denn $C_{\max}$ soll ja gerade das Maximum der Auslastungen sein, und ein Maximum wird von mindestens einem Mitarbeiter angenommen.

Der Punkt ist, dass die Ungleichung zusammen mit der Minimierung genau das Gewünschte leistet. Die Bedingungen (3.7) erzwingen, dass $C_{\max}$ mindestens so groß ist wie jede einzelne Auslastung, also $C_{\max} \geq \max_{i \in I} \sum_j p_{ij} x_{ij}$. Die Zielfunktion drückt $C_{\max}$ so weit nach unten, wie es die Bedingungen erlauben. In jeder Optimallösung gilt daher Gleichheit für mindestens einen Mitarbeiter, und $C_{\max}$ nimmt exakt den Wert des Maximums an. Die Kombination aus unterer Schranke und Minimierungsrichtung ist die übliche lineare Darstellung einer Maximumsfunktion (Wolsey, 2020).

Ein Gleichheitszeichen wäre dagegen schlicht falsch. Es würde verlangen, dass alle 8 Mitarbeiter exakt dieselbe Auslastung erreichen. Bei unteilbaren Aufgaben und personenabhängigen Bearbeitungszeiten ist eine solche perfekt gleichmäßige Verteilung im Allgemeinen nicht erreichbar; das Modell wäre unzulässig und der Solver meldete, dass keine Lösung existiert – obwohl die Fertigung selbstverständlich jeden Tag zulässige Pläne umsetzt. Der Unterschied zwischen $\leq$ und $=$ entscheidet hier also nicht über Eleganz, sondern über Lösbarkeit.

Unzulässige Paare existieren nicht

Eine naheliegende Alternative zur Menge Z bestünde darin, für alle $8 \times 24 = 192$ Kombinationen eine Variable anzulegen und unzulässige Zuordnungen durch eine sehr große Bearbeitungszeit zu bestrafen. Dieses *Big-M*-Vorgehen ist verbreitet und in vielen Modellierungssituationen unvermeidlich – hier ist es jedoch ein Nachteil.

Der Grund liegt in der *LP-Relaxation*. Verfahren vom Typ *Branch-and-Bound* lösen zunächst das Modell ohne Ganzzahligkeitsforderung, verwenden dessen Zielfunktionswert als untere Schranke und verzweigen anschließend über fraktionale Variablen (Land & Doig, 1960; Nemhauser & Wolsey, 1988). Die Qualität dieser Schranke bestimmt maßgeblich, wie viele Knoten der Suchbaum umfasst. Eine große Konstante M erzeugt nun eine Relaxation, in der die bestrafte Variable einen kleinen Bruchteil ihres Wertes annehmen kann, ohne die Zielfunktion nennenswert zu belasten. Der relaxierte Zielfunktionswert liegt dann weit unter dem ganzzahligen Optimum, die *Dualitätslücke* ist groß, und der Suchbaum wächst entsprechend. Big-M-Formulierungen sind für schwache Relaxationen bekannt (Wolsey, 2020).

Lässt man die unzulässigen Paare dagegen von vornherein weg, so kann die Relaxation gar nicht erst auf sie ausweichen. Die Schranke wird schärfer, weil sie über einem kleineren und wahrheitsgetreuen Lösungsraum gebildet wird. Zugleich schrumpft das Modell. Der Vorteil ist damit doppelt und in beiden Hinsichten echt: Das Modell ist kleiner *und* seine Relaxation ist besser. Dass sich die Qualifikationsschranken so sauber ausdrücken lassen, ist ein Glücksfall der vorliegenden Struktur; er sollte genutzt werden.

Modellgröße

Die Auswirkung lässt sich beziffern. Bei 8 Mitarbeitern und 24 Arbeitsgängen wären 192 Kombinationen denkbar. Da jeder Mitarbeiter aber nur zwei bis drei der 6 Qualifikationen beherrscht, bleiben lediglich 84 zulässige Paare übrig. Das Modell besitzt also 84 Binärvariablen und eine stetige Variable, zusammen mit $24 + 8 = 32$ Nebenbedingungen. Die Qualifikationsschranken dünnen das Modell auf weniger als die Hälfte seines nominellen Umfangs aus.

Diese Beobachtung hat einen bemerkenswerten Nebeneffekt. Die Qualifikationsschranken erschweren die Planung aus betrieblicher Sicht, weil sie Handlungsspielraum nehmen. Aus Sicht des Solvers hingegen sind sie eine Hilfe: Sie verkleinern den Suchraum. Was den Menschen behindert, entlastet die Maschine.

Untere Schranken und ihre Schärfe

Eine untere Schranke für den Makespan lässt sich ohne jede Optimierung angeben. Kein Arbeitsgang kann schneller erledigt werden als von der dafür am besten geeigneten Person; die dabei anfallende Arbeit muss auf 8 Mitarbeiter verteilt werden. Damit gilt

$$C_{\max} \;\geq\; \frac{1}{|I|} \sum_{j \in J} \min_{i \,:\, (i,j) \in Z} p_{ij} \;=\; 205{,}44 \text{ Minuten.} \tag{3.10}$$

Eine zweite, ebenso einfache Schranke ist die längste unvermeidbare Einzelaufgabe: Da Aufgaben unteilbar sind, ist der Makespan mindestens so groß wie $\max_{j \in J} \min_{i \,:\, (i,j) \in Z} p_{ij}$, also so groß wie derjenige Arbeitsgang, der selbst unter der günstigsten Zuordnung am längsten dauert. Für die vorliegende Instanz ist die Lastschranke (3.10) die stärkere der beiden; Schranken dieser Art bilden den Ausgangspunkt der klassischen Gütegarantien für Listenverfahren (Graham, 1969) und der Approximationsalgorithmen für unabhängige parallele Maschinen (Lenstra et al., 1990).

Das tatsächliche Optimum liegt jedoch bei 236,20 Minuten. Die Lücke zur trivialen Schranke beträgt 14,97 Prozent. Die Schranke ist somit nicht scharf, und dafür gibt es zwei Gründe.

Der erste Grund sind die Qualifikationsschranken. Schranke (3.10) unterstellt, dass jeder Arbeitsgang von seinem jeweils besten Bearbeiter ausgeführt wird und die Last dennoch gleichmäßig verteilt werden kann. Beides zugleich ist unmöglich. Weist man jeden Arbeitsgang seinem schnellsten Bearbeiter zu, so häufen sich die Arbeitsgänge einer Qualifikation bei derjenigen Person, die diese Qualifikation am besten beherrscht, während andere Personen leer ausgehen. Wer die Last umverteilen will, muss Arbeitsgänge an langsamere Personen abgeben und damit die Gesamtarbeit erhöhen. Der Konflikt zwischen Bestbesetzung und Gleichverteilung ist die eigentliche Schwierigkeit des Problems – und genau der Punkt, an dem die Schranke die Wirklichkeit verfehlt.

Der zweite Grund ist die Unteilbarkeit. Selbst wenn die Qualifikationsstruktur eine ausgewogene Verteilung zuließe, ließe sich die mittlere Last (3.10) nur dann exakt erreichen, wenn sich die Arbeitsgänge beliebig fein aufteilen ließen. Da sie das nicht tun, bleibt stets ein Rest, um den die beste erreichbare Verteilung über dem Mittelwert liegt.

Die Schranke bleibt gleichwohl nützlich. Sie liefert eine Zahl, gegen die sich jede heuristische Lösung sofort einordnen lässt, und sie ist es, die Branch-and-Bound überhaupt erst erlaubt, Teilbäume zu verwerfen.

3.5 Vergleich nutzbarer Methoden

Für ein Modell dieser Struktur kommen drei Gruppen von Verfahren in Betracht. Sie unterscheiden sich in Gütegarantie, Rechenzeit, Skalierbarkeit, Implementierungsaufwand und Erweiterbarkeit.

Exakte Verfahren lösen das Modell (3.5) bis (3.9) in seiner ursprünglichen Form. Das Standardverfahren ist Branch-and-Bound (Land & Doig, 1960): Es löst die LP-Relaxation, verzweigt über eine fraktionale Binärvariable und verwirft Teilbäume, deren Schranke die beste bekannte Lösung nicht mehr unterbieten kann. Der Vorzug ist die *Gütegarantie*: Das Verfahren liefert nicht nur eine gute Lösung, sondern zugleich den Beweis, dass keine bessere existiert. Der Preis ist eine Rechenzeit, die im schlechtesten Fall exponentiell wächst – das zugrundeliegende Problem ist *NP-schwer* (Garey & Johnson, 1979). Der Implementierungsaufwand ist gering, sofern ein Solver eingesetzt wird; das Modell wird deklarativ notiert und von der Software gelöst (Makhorin, 2020). Auch die Erweiterbarkeit ist hoch: Zusätzliche Anforderungen wie Termine oder Obergrenzen je Person lassen sich als weitere lineare Nebenbedingungen ergänzen, ohne den Lösungsalgorithmus anzutasten.

Konstruktionsheuristiken bauen eine Lösung in einem einzigen Durchlauf auf und revidieren keine Entscheidung. Das *Greedy*-Verfahren geht die Arbeitsgänge in gegebener Reihenfolge durch und weist jeden derjenigen qualifizierten Person zu, die anschließend am frühesten fertig wäre. Die LPT-Regel sortiert die Arbeitsgänge zuvor absteigend nach Dauer und bearbeitet die langen zuerst; dahinter steht die Einsicht, dass eine spät eingeplante lange Aufgabe den Makespan unnötig in die Höhe treibt (Graham, 1969). Beide Verfahren sind in wenigen Zeilen implementiert und laufen in Bruchteilen einer Millisekunde. Eine Gütegarantie besitzen sie im vorliegenden Fall jedoch nicht: Die klassischen Schranken für LPT gelten für identische Maschinen, während hier unabhängige Maschinen mit Qualifikationsschranken vorliegen (Lenstra et al., 1990). Ihre Erweiterbarkeit ist gering – jede neue Anforderung verlangt einen Eingriff in die Konstruktionslogik.

Metaheuristiken verbessern eine vorhandene Lösung durch wiederholte lokale Veränderungen. SA akzeptiert dabei nicht nur Verbesserungen, sondern mit einer temperaturabhängigen Wahrscheinlichkeit auch Verschlechterungen, um lokale Optima wieder verlassen zu können; die Temperatur wird im Verlauf gesenkt, sodass das Verfahren am Ende nur noch Verbesserungen zulässt (Kirkpatrick et al., 1983). Verwandte Ansätze sind die *Tabu-Suche*, die kürzlich besuchte Lösungen für eine gewisse Zahl von Schritten sperrt und so Zyklen verhindert, sowie *genetische Algorithmen*, die eine Population von Lösungen führen und durch Rekombination und Mutation weiterentwickeln. Allen gemeinsam ist, dass sie keine Gütegarantie liefern und über eine Reihe von Parametern verfügen, deren Einstellung Erfahrung verlangt. Ihre Stärke ist die Skalierbarkeit: Die Rechenzeit wird durch die Iterationszahl vorgegeben und nicht durch die Instanzgröße bestimmt. Ihre Erweiterbarkeit ist mittelmäßig; zusätzliche Anforderungen lassen sich in die Bewertungsfunktion aufnehmen, doch die Nachbarschaft muss weiterhin zulässige Lösungen erzeugen.

Tabelle 3.1 stellt die Verfahren mit den Ergebnissen zusammen, die sie auf der Instanz der Erpolino Metallverarbeitung GmbH erzielen; die Vorgriffe auf Kapitel 6 sind hier nur als Beleg der methodischen Argumentation zu lesen. Das Bild ist eindeutig. Das MILP-Modell liefert den Makespan 236,20 Minuten und damit die beweisbar optimale Lösung, und zwar in weniger als

Verfahren	Makespan min	Abstand %	Rechenzeit ms	Gütegarantie
Untere Schranke (trivial)	205,44	−13,02	–	Schranke, keine Lösung
Greedy	266,67	12,90	0,0	keine
LPT	265,38	12,36	0,0	keine
Simulated Annealing	240,74	1,92	297,9	keine
MILP (GLPK)	236,20	0,00	676,5	beweisbar optimal
Simulated Annealing über 20 Läufe, je 60 000 Iterationen:				
Energie nur Makespan	240,71	1,91	–	bestes 239.86, schlecht. 240.84
Energie mit Glättung	240,10	1,65	–	bestes 236.20, schlecht. 243.68

Tabelle 3.1: Die untersuchten Verfahren auf der Instanz von Erpolino

einer Sekunde. LPT kommt auf 265,38 Minuten und verfehlt das Optimum um 12,36 Prozent; SA erreicht 240,74 Minuten und bleibt 1,92 Prozent darüber. Die Heuristiken sind schneller, doch bei einer Rechenzeit des exakten Verfahrens von unter einer Sekunde ist dieser Vorteil ohne praktischen Wert.

Für die vorliegende Problemgröße gibt es damit keinen Grund, auf eine Heuristik auszuweichen. Die Wahl fällt auf das exakte Verfahren, und zwar nicht, weil es grundsätzlich überlegen wäre, sondern weil die Instanz klein genug ist, um seine Schwäche – die im schlechtesten Fall exponentielle Rechenzeit – gar nicht erst wirksam werden zu lassen. Ab welcher Größe sich dieses Verhältnis umkehrt und die Heuristiken ihre Berechtigung erhalten, untersucht Kapitel 6 anhand künstlich vergrößerter Instanzen.

4 Das Unternehmen Erpolino

Die vorangegangenen Kapitel haben das Zuordnungsproblem in allgemeiner Form entwickelt. Damit aus dem Modell eine rechenbare Instanz wird, sind konkrete Zahlen nötig: Wer arbeitet im Betrieb, welche Tätigkeiten darf diese Person ausführen, wie schnell erledigt sie diese, und welche Arbeitsgänge stehen überhaupt an. Dieses Kapitel beschreibt das Unternehmen, aus dem diese Daten stammen, und legt offen, wie die Zahlen der Untersuchung zustande kommen.[1]

4.1 Unternehmen und Fertigung

Die Erpolino Metallverarbeitung GmbH mit Sitz in Reutlingen wird 1978 als Dreherei mit vier Beschäftigten gegründet. Der Standort liegt in der Region Neckar-Alb, einem von mittelständischer Zulieferindustrie geprägten Teil Baden-Württembergs, dessen Nähe zu den Ballungsräumen Stuttgart und Tübingen den Kundenkreis des Unternehmens bis heute prägt. Seit 2004 führt die zweite Generation der Gründerfamilie das Unternehmen; die Geschäftsanteile befinden sich vollständig in Familienbesitz. Diese Eigentümerstruktur ist für die Fragestellung der vorliegenden Arbeit insofern relevant, als Investitions- und Organisationsentscheidungen ohne externe Kapitalgeber getroffen werden und Veränderungen in der Planung entsprechend selten aus formalen Vorgaben, sondern aus praktischem Bedarf entstehen.

Gegenwärtig beschäftigt das Unternehmen rund 120 Mitarbeitende, von denen etwa 45 in der Fertigung tätig sind. Die übrigen Stellen verteilen sich auf Konstruktion, Vertrieb, Einkauf, Qualitätsmanagement, Verwaltung und Instandhaltung. Das Unternehmen betreibt keine eigene Produktentwicklung und vertreibt keine eigenen Erzeugnisse, sondern arbeitet ausschließlich in der *Lohnfertigung*: Es fertigt Präzisionsteile nach Zeichnungen und Spezifikationen seiner Auftraggeber. Der wirtschaftliche Kern des Geschäftsmodells liegt damit nicht im Produkt, sondern in der verfügbaren Fertigungskapazität und in der Fähigkeit, diese termingerecht bereitzustellen.

Die Kunden stammen im Wesentlichen aus drei Branchen. Der *Maschinenbau* bezieht Gehäuse-, Trag- und Rahmenteile in kleinen Stückzahlen, häufig als Ersatz- oder Vorserienteile. Die *Antriebstechnik* fragt Wellen, Buchsen, Lagerböcke und Getriebedeckel nach, meist in mittleren Losgrößen und mit engen Form- und Lagetoleranzen. Die *Medizintechnik* schließlich verlangt kleine Stückzahlen bei hohen Anforderungen an Oberflächengüte und Dokumentation. Keiner dieser Kunden vergibt Großserien; die typische *Losgröße* liegt zwischen einem Dutzend

[1] Die Erpolino Metallverarbeitung GmbH ist ein frei erfundenes Unternehmen. Sämtliche in dieser Arbeit genannten Angaben zu Geschichte, Größe, Kundenstruktur, Personal und Auftragsbestand sowie alle Zahlenwerte der Qualifikationsmatrix und des Fertigungsloses sind konstruiert und dienen ausschließlich dazu, das Verfahren an einer in sich stimmigen Instanz vorzuführen. Ein realer Betrieb dieses oder ähnlichen Namens ist nicht gemeint, und die Ergebnisse lassen keinen Rückschluss auf reale Betriebsdaten zu.

und einigen hundert Teilen. Für die Fertigung bedeutet dies einen häufigen Wechsel zwischen Aufträgen und einen entsprechend hohen Anteil an Rüst- und Einrichtvorgängen.

Die Fertigungstiefe umfasst sechs Bereiche: Drehen, Fräsen, Schweißen, Schleifen, CNC-Programmierung und Qualitätsprüfung. Der Maschinenpark besteht überwiegend aus CNC-Dreh- und Bearbeitungszentren, ergänzt um konventionelle Maschinen für Einzelteile und Nacharbeit. Wärmebehandlung und Oberflächenveredelung werden zugekauft und sind nicht Gegenstand dieser Arbeit. Die sechs Bereiche sind keine räumlich getrennten Abteilungen mit festem Personal, sondern Tätigkeitsfelder, zwischen denen die Beschäftigten je nach Auftragslage wechseln. Genau diese Flexibilität ist der Ausgangspunkt der Untersuchung: Sie eröffnet Spielräume in der Zuteilung, die bislang nicht systematisch genutzt werden.

4.2 Qualifikationen der Mitarbeitenden

Ob eine Person einen Arbeitsgang übernehmen kann, entscheidet sich an ihrer Qualifikation. Das Unternehmen unterscheidet die 6 Qualifikationen Drehen, Fräsen, Schweißen, Schleifen, CNC-Programmierung und Qualitätsprüfung. Sie werden über Ausbildung, innerbetriebliche Einweisung und in Teilen über externe Zertifikate erworben; das Schweißen und die Qualitätsprüfung unterliegen dabei formalen Nachweispflichten gegenüber den Kunden der Medizintechnik. Niemand in der Fertigung beherrscht alle sechs Felder, und niemand beherrscht nur eines: Die Beschäftigten verfügen in der Regel über zwei oder drei Qualifikationen.

Erfasst wird dieser Sachverhalt in einer *Qualifikationsmatrix*. Sie ordnet jedem Paar aus Mitarbeiter und Qualifikation entweder einen Zahlenwert oder die Angabe zu, dass die Qualifikation nicht vorliegt. Der Zahlenwert ist die *Effizienz* der Person in diesem Feld. Sie ist als Verhältniszahl definiert: Der Wert 1,00 entspricht der *Normalleistung*, also derjenigen Leistung, die der Arbeitsplanung als Vorgabewert zugrunde liegt. Ein Wert über 1,00 bedeutet, dass die Person schneller arbeitet als die Vorgabe, ein Wert darunter, dass sie länger braucht. Die Bearbeitungsdauer eines Arbeitsgangs ergibt sich daraus als Quotient aus der Grundzeit des Arbeitsgangs und der Effizienz der ausführenden Person. Eine Grundzeit von 120 Minuten führt bei einer Effizienz von 1,20 folglich auf 100 Minuten, bei einer Effizienz von 0,90 dagegen auf rund 133 Minuten.

Die im Unternehmen auftretenden Werte liegen zwischen 0,85 und 1,40. Diese Spanne ist erklärungsbedürftig, denn sie ist erheblich: Zwischen der langsamsten und der schnellsten Bearbeitung desselben Arbeitsgangs liegt ein Faktor von mehr als eineinhalb. Sie erklärt sich aus der Zusammensetzung der Belegschaft. Wer eine Tätigkeit seit Jahren täglich ausübt, erreicht Werte im oberen Bereich; wer die Qualifikation zwar besitzt, sie aber nur gelegentlich einsetzt, liegt unter der Normalleistung. Die Effizienz misst damit keine persönliche Leistungsbereitschaft, sondern Routine im jeweiligen Feld. Sie ist zudem qualifikationsbezogen und nicht personenbezogen: Dieselbe Person kann in einem Feld über und in einem anderen unter der Normalleistung liegen.

Fehlt die Qualifikation, so ist die Zuordnung nicht etwa langsam, sondern unzulässig. Für diesen Fall existiert kein Effizienzwert, und in der Matrix steht ein Strich. Diese Unterscheidung ist für die Modellierung wesentlich: Ein fehlender Eintrag darf nicht durch einen sehr kleinen Effizienzwert ersetzt werden, weil das Verfahren eine solche Zuordnung sonst zwar teuer, aber

grundsätzlich zulässig behandeln würde. Wie diese Schranke im linearen Modell umgesetzt wird, ist in Kapitel 3 dargelegt.

Mitarbeiter	Drehen	Fräsen	Schweißen	Schleifen	CNC-Programmierung	Qualitätsprüfung	Qualifikationen
Bauer	1,25	1,05	–	0,95	–	–	3
Cakir	0,90	1,30	–	–	1,10	–	3
Dobrev	–	–	1,35	1,00	–	–	2
Engelhardt	–	1,00	–	–	1,40	–	2
Fischer	1,10	–	0,95	–	–	1,15	3
Gruber	0,85	–	–	1,30	–	1,05	3
Haas	–	–	–	0,90	–	1,35	2
Ivanov	1,00	0,95	1,15	–	–	–	3
Mitarbeiter je Qualifikation	5	4	3	4	2	3	

Tabelle 4.1: Qualifikationsmatrix: Effizienz je Mitarbeiter und Qualifikation. Der Wert 1,00 entspricht der Normalleistung; ein Strich bedeutet, dass die Qualifikation nicht vorliegt und die Zuordnung damit unzulässig ist.

Tabelle 4.1 zeigt die Matrix für die 8 Mitarbeitenden, die in dieser Arbeit betrachtet werden. Zwei Beobachtungen sind für das weitere Vorgehen wichtig. Erstens ist die Matrix dünn besetzt: Von den $8 \times 6 = 48$ denkbaren Kombinationen sind nur 21 belegt. Zweitens ist die Besetzung ungleichmäßig. Fünf Personen dürfen drehen, aber nur zwei sind für die CNC-Programmierung qualifiziert, und beim Schweißen wie bei der Qualitätsprüfung sind es jeweils drei. Damit ist bereits an der Matrix ablesbar, wo Engpässe zu erwarten sind: Arbeitsgänge, die eine seltene Qualifikation verlangen, konkurrieren um dieselben wenigen Personen, unabhängig davon, wie gut die übrige Belegschaft ausgelastet ist.

4.3 Das Fertigungslos einer Kalenderwoche

Als Datengrundlage der Untersuchung dient das Fertigungslos einer Kalenderwoche, wie es die *Arbeitsvorbereitung* zu Wochenbeginn freigibt. Es umfasst 24 Arbeitsgänge, die aus Kundenaufträgen der drei genannten Branchen hervorgehen. Jeder Arbeitsgang ist durch drei Angaben bestimmt: eine Bezeichnung, die das zu bearbeitende Teil und die Tätigkeit benennt, die geforderte Qualifikation und eine *Grundzeit* in Minuten.

Die Grundzeit stammt aus dem Arbeitsplan des jeweiligen Teils. Sie ist die kalkulierte Bearbeitungszeit bei Normalleistung und schließt Nebenzeiten wie Aufspannen und Messen ein, nicht jedoch die *Rüstzeit* der Maschine, die im Unternehmen gesondert erfasst und in dieser Arbeit nicht modelliert wird. Die Grundzeiten des betrachteten Loses liegen zwischen 45 und 150 Minuten. Diese Spanne ist typisch für den Betrieb: Die kurzen Arbeitsgänge entfallen überwiegend auf die Qualitätsprüfung, die langen auf Fräs- und Schweißarbeiten an größeren Bauteilen.

Aus der Kombination von Fertigungslos und Qualifikationsmatrix ergibt sich die Größe der Probleminstanz. Von den $8 \times 24 = 192$ denkbaren Zuordnungen sind 84 zulässig; mehr als die Hälfte scheidet also allein an der fehlenden Qualifikation aus. Für jedes der 84 zulässigen Paare ist eine Bearbeitungszeit definiert, für alle übrigen keine.

Tabelle 4.2: Arbeitsgänge des Fertigungsloses mit der vom Verfahren gefundenen Zuordnung

Nr.	Bezeichnung	Grundzeit min	Qualifikation	Dauer min
A00	Flanschwelle A-102 drehen	95	Drehen	95,0
A01	Lagerbock B-204 fraesen	120	Fräsen	114,3
A02	Rahmen C-310 schweissen	140	Schweißen	121,7
A03	Passfläche D-011 schleifen	60	Schleifen	66,7
A04	Programm E-556 erstellen	110	CNC-Programmierung	78,6
A05	Charge F-078 pruefen	45	Qualitätsprüfung	33,3
A06	Antriebswelle A-115 drehen	130	Drehen	118,2
A07	Getriebedeckel B-221 fraesen	85	Fräsen	65,4
A08	Traverse C-333 schweissen	105	Schweißen	77,8

Fortsetzung nächste Seite

Tabelle 4.2: Arbeitsgänge (Fortsetzung)

Nr.	Bezeichnung	Grundzeit min	Qualifikation	Dauer min
A09	Dichtsitz D-042 schleifen	75	Schleifen	83,3
A10	Programm E-560 erstellen	90	CNC-Programmierung	64,3
A11	Charge F-081 pruefen	55	Qualitätsprüfung	40,7
A12	Spindel A-127 drehen	70	Drehen	56,0
A13	Konsole B-238 fraesen	150	Fräsen	115,4
A14	Behaelter C-350 schweissen	95	Schweißen	70,4
A15	Laufflaeche D-063 schleifen	110	Schleifen	84,6
A16	Programm E-571 erstellen	65	CNC-Programmierung	59,1
A17	Charge F-090 pruefen	80	Qualitätsprüfung	69,6
A18	Buchse A-134 drehen	50	Drehen	45,5
A19	Adapterplatte B-245 fraesen	70	Fräsen	66,7
A20	Konsolarm C-361 schweissen	125	Schweißen	92,6
A21	Fuehrung D-077 schleifen	90	Schleifen	69,2
A22	Programm E-583 erstellen	100	CNC-Programmierung	71,4
A23	Charge F-095 pruefen	60	Qualitätsprüfung	57,1

Tabelle 4.2 führt die 24 Arbeitsgänge mit Bezeichnung, Grundzeit, geforderter Qualifikation und der Dauer auf, die sich in der später berechneten Zuordnung tatsächlich ergibt. Der Vergleich der beiden Zeitspalten macht die Wirkung der Effizienz sichtbar: Wo die Dauer unter der Grundzeit liegt, hat das Verfahren den Arbeitsgang einer Person mit überdurchschnittlicher Routine zugewiesen; wo sie darüber liegt, war eine schnellere Person entweder nicht qualifiziert oder bereits stärker belastet als der Gesamtplan es zulässt. Dass das Verfahren einzelne Arbeitsgänge bewusst langsamer ausführen lässt, als es möglich wäre, ist kein Fehler, sondern Ausdruck der Zielsetzung: Minimiert wird nicht die Summe der Bearbeitungszeiten, sondern die Belastung der am stärksten ausgelasteten Person.

4.4 Bisherige Disposition und ihre Schwächen

Die Zuteilung der Arbeitsgänge erfolgt im Unternehmen bislang ohne rechnergestützte Optimierung. Die Fertigungsleitung öffnet zu Wochenbeginn eine Tabellenkalkulation, in der die freigegebenen Arbeitsgänge und die verfügbaren Personen aufgelistet sind, und trägt die Zuordnung von Hand ein. Die Qualifikationsmatrix existiert dabei durchaus, sie liegt jedoch als eigenes Tabellenblatt vor und wird beim Eintragen aus dem Gedächtnis herangezogen. Eine Prüfung, ob die eingetragene Person die geforderte Qualifikation überhaupt besitzt, findet nicht automatisch statt.

Das Vorgehen folgt einer erkennbaren Regel, auch wenn diese nirgends niedergeschrieben ist. Zuerst werden die Arbeitsgänge mit seltenen Qualifikationen verteilt, weil dort kaum Wahlmöglichkeiten bestehen. Danach werden die verbleibenden Arbeitsgänge der Reihe nach vergeben, wobei die Fertigungsleitung jeweils die Person auswählt, die den Arbeitsgang am schnells-

ten erledigen kann und noch als frei gilt. Diese Regel ist plausibel und liefert zulässige Pläne. Sie ist zugleich eine *Greedy-Regel* im Sinne von Kapitel 2: Sie trifft jede Entscheidung lokal und revidiert sie nicht, wenn sich später herausstellt, dass eine andere Wahl den Gesamtplan verbessert hätte.

Daraus folgt die zentrale Schwäche. Wer in einem Feld die höchste Effizienz besitzt, erhält dort tendenziell jeden Arbeitsgang und ist am Ende der Woche überlastet, während andere Kapazität ungenutzt bleibt. Da das Los erst abgeschlossen ist, wenn der letzte Arbeitsgang beendet wurde, bestimmt allein die am stärksten belastete Person den *Makespan*. Eine Zuordnung, die einen schnellen Mitarbeiter entlastet und dafür einen Arbeitsgang bewusst einer langsameren Person überträgt, kann den Gesamtplan folglich verkürzen, obwohl sie die Summe der Bearbeitungszeiten erhöht. Genau diese Möglichkeit ist einer Regel, die Arbeitsgang für Arbeitsgang die jeweils schnellste Person wählt, strukturell verschlossen.

Wie groß der dadurch verschenkte Spielraum ist, lässt sich beziffern. Die nachgebildete Greedy-Regel erreicht auf dem betrachteten Los einen Makespan von 266,67 Minuten, das exakte Verfahren dagegen 236,20 Minuten. Die Differenz von rund einer halben Stunde entsteht nicht durch zusätzliche Kapazität, sondern allein durch eine andere Verteilung derselben Arbeit auf dieselben Personen. Hinzu kommen zwei Nachteile, die sich nicht in Minuten ausdrücken lassen: Die Zuteilung ist an das Erfahrungswissen einzelner Personen gebunden und bei deren Abwesenheit nur eingeschränkt reproduzierbar, und sie ist gegenüber den Beschäftigten nicht begründbar, weil kein nachvollziehbares Kriterium vorliegt, nach dem sie zustande gekommen ist. Ein Optimierungsmodell beseitigt beide Nachteile als Nebeneffekt: Es macht die Regel explizit, nach der geplant wird.

5 Umsetzung

Das in Kapitel 3 entwickelte Modell wird auf zwei Wegen umgesetzt: als *Modellierungssprache* in MathProg, die ein Standardlöser bearbeitet, und als eigenes Verfahren in Python. Beide Umsetzungen arbeiten auf derselben Instanz. Das ist keine Selbstverständlichkeit und wird in Abschnitt 5.1 begründet.

5.1 Eine Quelle für die Daten

Wer zwei Verfahren vergleicht, vergleicht in Wahrheit oft zwei Datensätze. Werden die Zahlen für den Löser und für das eigene Programm getrennt gepflegt, laufen sie früher oder später auseinander, und der Vergleich misst dann den Unterschied der Eingaben statt den der Verfahren.

Deshalb liegt die Instanz genau einmal vor, nämlich in `Code/daten.py`. Aus dieser Quelle wird die Datendatei für GNU Linear Programming Kit (GLPK) erzeugt; dasselbe Modul liest das Python-Verfahren direkt ein. Die Bearbeitungszeit ergibt sich in beiden Fällen aus derselben Formel

$$p_{ij} = \frac{g_j}{e_i(q_j)}, \tag{5.1}$$

wobei g_j die *Grundzeit* des Arbeitsgangs j ist, q_j die benötigte Qualifikation und $e_i(q_j)$ die Effizienz des Mitarbeiters i in dieser Qualifikation. Ist $e_i(q_j)$ nicht definiert, fehlt die Qualifikation, und das Paar (i, j) existiert nicht.

Dieselbe Quelle speist auch die Abbildungen, die Tabellen und die Zahlen im Fließtext dieser Arbeit. Alle Zahlenwerte in diesem Text sind Makros, die aus den Rechenergebnissen erzeugt werden. Eine Zahl, die hier steht, kann deshalb nicht von der Rechnung abweichen, auf die sie sich beruft.

5.2 Das Modell in MathProg

MathProg (auch GMPL) ist die Modellierungssprache von GLPK (Makhorin, 2020). Sie beschreibt das Modell deklarativ: Mengen, Parameter, Variablen, Zielfunktion, Nebenbedingungen. Wie das Problem gelöst wird, steht nicht im Modell – das ist Sache des Lösers.

Die Mengen und die Zulässigkeit werden so erklärt:

```
set MITARBEITER;
set AUFGABEN;
set ZULAESSIG within {MITARBEITER, AUFGABEN};
```

```
param p{(i,j) in ZULAESSIG} > 0;

var x{(i,j) in ZULAESSIG}, binary;
var Cmax >= 0;
```

Listing 5.1: Mengen, Parameter und Variablen

Entscheidend ist die dritte Zeile. `ZULAESSIG` ist eine Teilmenge des Kreuzprodukts, und alle folgenden Größen sind nur über dieser Teilmenge erklärt. Eine Variable x_{ij} für ein Paar ohne passende Qualifikation entsteht damit gar nicht erst. Die Alternative – alle Paare anlegen und die unzulässigen mit einer großen Zahl bestrafen – wäre nicht nur größer, sondern brächte auch die in Abschnitt 3.4 beschriebene Schwächung der Relaxation mit sich.

Zielfunktion und Nebenbedingungen folgen unmittelbar der Formulierung aus Kapitel 3:

```
minimize Gesamtdauer: Cmax;

s.t. Vergabe{j in AUFGABEN}:
  sum{(i,j) in ZULAESSIG} x[i,j] = 1;

s.t. Auslastung{i in MITARBEITER}:
  sum{(i,j) in ZULAESSIG} p[i,j] * x[i,j] <= Cmax;
```

Listing 5.2: Zielfunktion und Nebenbedingungen

Das vollständige Modell steht in Anhang B. Aufgerufen wird es mit

```
glpsol --math erpolino.mod --data erpolino.dat
```

GLPK löst die Instanz in 0,7 Sekunden und meldet `INTEGER OPTIMAL SOLUTION FOUND`. Diese Meldung ist mehr als ein Ergebnis: Sie ist ein Beweis. Der Löser hat gezeigt, dass keine bessere Zuordnung existiert, nicht nur, dass er keine gefunden hat.

5.3 Das Verfahren in Python

Das Verfahren folgt dem klassischen Ablauf (Kirkpatrick et al., 1983): Von einer Startlösung ausgehend wird wiederholt eine Nachbarlösung erzeugt und angenommen oder verworfen. Als Startlösung dient die LPT-Regel.

5.3.1 Nachbarschaft

Zwei Operatoren erzeugen Nachbarlösungen, wie in Abbildung 2.2 dargestellt. Das *Verschieben* nimmt dem am stärksten ausgelasteten Mitarbeiter eine Aufgabe ab und gibt sie einem anderen qualifizierten Mitarbeiter. Das *Tauschen* kreuzt zwei Aufgaben zwischen ihren Mitarbeitern, sofern beide dafür qualifiziert sind.

Der zweite Operator ist nicht entbehrlich. Sind zwei Mitarbeiter beide ausgelastet, führt jedes Verschieben zwangsläufig zu einer Verschlechterung an anderer Stelle; nur ein Tausch kann die Last umverteilen, ohne sie zu erhöhen. Ohne den Tausch bleibt das Verfahren an solchen Konstellationen hängen.

5.3.2 Die Energie und das Plateauproblem

Naheliegend wäre, die Zielgröße selbst als *Energie* zu verwenden. Das funktioniert schlecht, und der Grund ist lehrreich. Der *Makespan* ist ein Maximum: Er ändert sich nur, wenn gerade der Engpass betroffen ist. Jeder Zug, der die übrigen Mitarbeiter betrifft, ist energieneutral. Das Verfahren läuft über weite *Plateaus* und erhält kein Signal, welche Richtung sich lohnt.

Abhilfe schafft ein zusätzlicher Term, der schon dann sinkt, wenn die Last gleichmäßiger verteilt wird:

$$E(x) = \max_{i \in I} a_i(x) + \lambda \sqrt{\sum_{i \in I} a_i(x)^2}, \qquad a_i(x) = \sum_{j\,:\,(i,j) \in Z} p_{ij}\, x_{ij}. \tag{5.2}$$

Der zweite Summand ist die euklidische Norm des *Auslastungsvektors*. Er macht die Landschaft begehbar, ohne die Zielgröße zu ersetzen: Das Gewicht $\lambda = 0{,}002$ ist so klein gewählt, dass der Term die Rangfolge zweier Lösungen mit unterschiedlichem Makespan nicht umdrehen kann. Die Auslastungen liegen bei rund 200 Minuten, der Term damit bei etwa 1,2; ein Unterschied im Makespan von einer Minute wiegt schwerer.

Bewertet wird während des Laufs über E, berichtet und gespeichert wird stets der echte Makespan. Was diese Änderung bewirkt, zeigt Abschnitt 6.2: Der Mittelwert über 20 Läufe sinkt von 240,71 auf 240,10 Minuten, und erst mit dem Glättungsterm wird das Optimum überhaupt erreicht.

5.3.3 Abkühlung

Die Temperatur fällt geometrisch, $T_{k+1} = \alpha\, T_k$, von $T_{\text{start}} = 25$ auf $T_{\text{ende}} = 0{,}05$ über 60 000 Schritte. Der Faktor ergibt sich daraus zu

$$\alpha = \left(\frac{T_{\text{ende}}}{T_{\text{start}}} \right)^{1/K} \approx 0{,}99990. \tag{5.3}$$

Abbildung 5.1 zeigt den Verlauf und die daraus folgende Annahmewahrscheinlichkeit für eine Verschlechterung. Zu Beginn werden auch deutliche Verschlechterungen angenommen, gegen Ende kaum noch: Das Verfahren geht von der Erkundung in die Feinsuche über.

5.3.4 Reproduzierbarkeit

Simulated Annealing ist ein stochastisches Verfahren. Ein einzelner Lauf sagt daher wenig aus, und ein Lauf ohne festen Startwert lässt sich nicht wiederholen. Beide Punkte werden hier ernst genommen: Alle Läufe verwenden feste Startwerte, und ausgewertet wird nicht ein Lauf, sondern die Streuung über 20 Läufe.

5.4 Werkzeuge

Das Verfahren ist in Python 3.13 ohne weitere Abhängigkeiten implementiert; die Auswertung nutzt NumPy (Harris et al., 2020) und matplotlib (Hunter, 2007). Als Löser dient GLPK in Version 5.0 (Makhorin, 2020). Der Quelltext des Verfahrens findet sich in Anhang A.

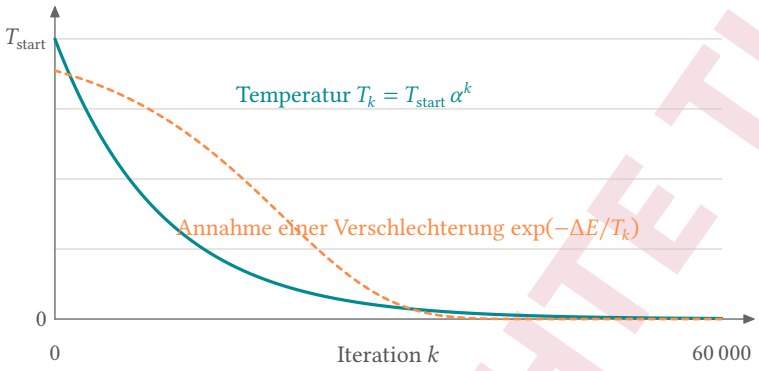


Abbildung 5.1: Geometrische Abkühlung über den Lauf und die daraus folgende Annahmewahrscheinlichkeit einer Verschlechterung. Gezeichnet mit MetaPost.

6 Analyse der Ergebnisse

Dieses Kapitel wertet die Läufe auf der Instanz aus Kapitel 4 aus. Zuerst wird die gefundene Zuordnung betrachtet, dann die Wirkung der geglätteten Energie, schließlich das Verhalten beider Verfahren bei wachsender Instanzgröße.

6.1 Die gefundene Zuordnung

Abbildung 6.1 stellt die Auslastung der Mitarbeitenden für die LPT-Startlösung und die Lösung des Verfahrens gegenüber. Der Unterschied ist weniger die Höhe als die Gleichmäßigkeit: LPT erzeugt ein deutliches Gefälle, das Verfahren zieht die Balken zusammen. Genau darum geht es – der *Makespan* ist die Höhe des höchsten Balkens, alles darunter ist Leerlauf.

Auffällig ist, dass selbst die optimale Lösung kein waagerechtes Profil erzeugt. Die Auslastungen streuen weiterhin um einige Minuten. Das ist keine Schwäche des Lösers, sondern eine Eigenschaft des Problems: Aufgaben sind unteilbar, und die Qualifikationen schränken ein, wer was übernehmen darf. Eine vollkommen gleichmäßige Verteilung ist im Allgemeinen unzulässig. Sie ist genau die Annahme, auf der die triviale untere Schranke von 205,44 Minuten beruht – und deshalb liegt das wahre Optimum mit 236,20 Minuten um 14,97 Prozent darüber.

Wohin die Arbeitszeit fließt, zeigt Abbildung 6.2. Die Bandbreite entspricht den Minuten, die von einer Qualifikation auf einen Mitarbeitenden entfallen. Sichtbar wird, dass sich die Last der seltenen Qualifikationen auf wenige Personen konzentriert: Wer als Einziger schweißen kann, bekommt die Schweißarbeit, unabhängig davon, wie ausgelastet er ohnehin ist.

6.2 Wirkung der geglätteten Energie

Abschnitt 5.3.2 hat die geglättete Energie mit dem Plateauproblem begründet. Ob das Argument trägt, ist eine empirische Frage. Abbildung 6.3 beantwortet sie über je 20 Läufe beider Varianten.

Der Befund ist deutlich. Mit der rohen Zielgröße liegen alle Läufe eng beieinander bei 240,71 Minuten im Mittel – eng, aber gleichmäßig zu hoch. Das Verfahren findet zuverlässig dasselbe lokale Optimum und kommt nicht darüber hinaus. Mit dem Glättungsterm sinkt der Mittelwert auf 240,10 Minuten, und der beste Lauf erreicht mit 236,20 Minuten das Optimum.

Bemerkenswert ist der Preis: Die Streuung nimmt zu. Die geglättete Variante ist im Mittel besser, aber weniger vorhersagbar. Wer einen einzelnen Lauf betrachtet, kann schlechter abschneiden als mit der rohen Variante. Das ist der übliche Handel zwischen Erkundung und Verlässlichkeit und ein Argument dafür, mehrere Läufe zu rechnen und den besten zu nehmen – was bei einer Rechenzeit von 298 Millisekunden je Lauf keine ernsthafte Hürde ist.

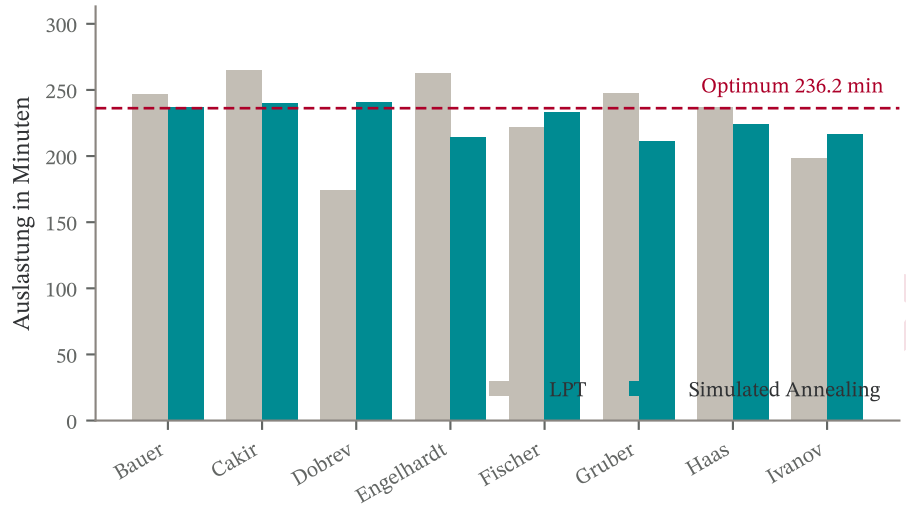


Abbildung 6.1: Auslastung der Mitarbeitenden bei der LPT-Startlösung und nach dem Verfahren. Der Makespan ist die Höhe des jeweils höchsten Balkens.

Abbildung 6.4 zeigt einen einzelnen Lauf im Verlauf. Die aktuelle Lösung schwankt anfangs stark – die Temperatur lässt Verschlechterungen zu –, und beruhigt sich mit sinkender Temperatur. Die beste gefundene Lösung fällt in Stufen: lange Phasen ohne Fortschritt, unterbrochen von Sprüngen. Genau dieses Muster erwartet man bei einer plateaulastigen Zielgröße.

6.3 Vergleich der Verfahren

Tabelle 3.1 auf Seite 18 hat die Verfahren bereits gegenübergestellt. Die Rangfolge ist eindeutig: Die bisherige Praxis entspricht ungefähr der Greedy-Regel mit 266,67 Minuten, LPT verbessert das auf 265,38 Minuten, SA auf 240,74 Minuten, und das exakte Verfahren erreicht 236,20 Minuten.

Gegenüber der heutigen Praxis spart die optimale Lösung rund eine halbe Stunde je Fertigungslos. Das ist der Ertrag, um den es geht, und er ist mit einem Standardlöser in weniger als einer Sekunde zu haben.

Damit stellt sich die Frage nach dem Sinn der Heuristik in aller Schärfe. Auf dieser Instanz ist SA in jeder Hinsicht unterlegen: Es braucht mehr Rechenzeit als der exakte Löser, findet ein um 1,92 Prozent schlechteres Ergebnis und gibt keinerlei Garantie. Wer nur diese Instanz betrachtet, sollte den Löser nehmen.

6.4 Verhalten bei wachsender Instanz

Der Vorteil des exakten Verfahrens ist allerdings an die Größe gebunden. Um zu prüfen, wo die Grenze liegt, wurden Zufallsinstanzen derselben Struktur mit wachsender Zahl von Arbeitsgängen erzeugt und beide Verfahren darauf angewendet. Dem Löser wurde eine Zeitschranke

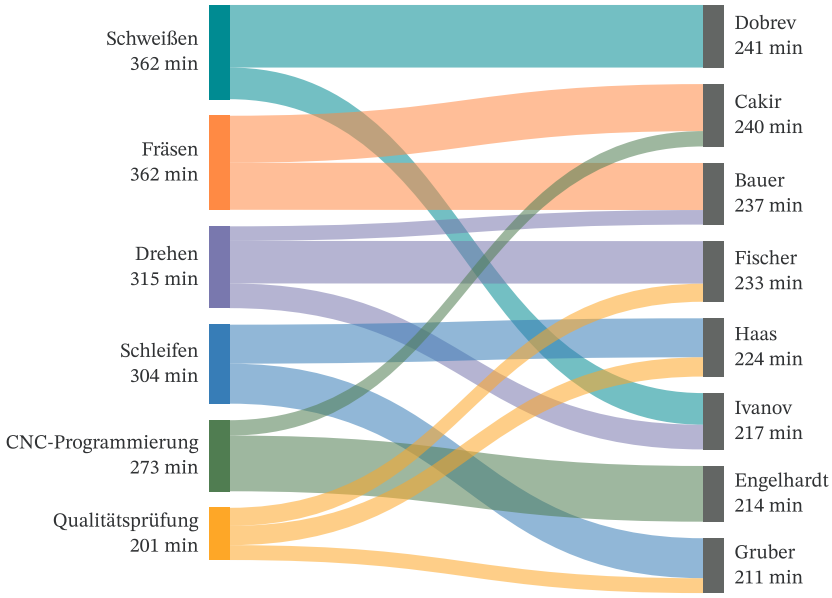


Abbildung 6.2: Fluss der Arbeitszeit von den Qualifikationen zu den Mitarbeitenden in der gefundenen Lösung. Die Bandbreite entspricht den Minuten.

von 30 Sekunden gesetzt.

Tabelle 6.1 und Abbildung 6.5 zeigen den Umschlag. Bis 40 Arbeitsgänge löst GLPK beweisbar optimal, teilweise in Sekundenbruchteilen. Ab 60 Arbeitsgängen läuft der Löser in die Zeitschranke und kann die Optimalität nicht mehr belegen. Von da an liefert SA in unter einer Sekunde Lösungen, die gleich gut oder besser sind als das, was der Löser in 30 Sekunden erreicht – bei 80 Arbeitsgängen um gut ein Prozent besser.

Zwei Beobachtungen verdienen Beachtung. Erstens verläuft die Rechenzeit des Lösers nicht monoton: Die Instanz mit 40 Arbeitsgängen ist leichter als die mit 24. Die Schwierigkeit einer Instanz hängt eben nicht allein an ihrer Größe, sondern an ihrer Struktur – eine bekannte, aber gern übersehene Eigenschaft ganzzahliger Probleme. Zweitens wächst die Rechenzeit von SA nur schwach, weil sie durch die feste Zahl von Iterationen bestimmt ist und nicht durch die Instanz. Das macht sie planbar, sagt aber nichts über die Güte.

6.5 Grenzen der Aussage

Die Ergebnisse dieses Kapitels gelten für die untersuchte Instanz und die untersuchten Zufallsinstanzen. Fünf Größen mit je einer Instanz sind eine schmale Grundlage; belastbar wäre eine Auswertung über mehrere Instanzen je Größe mit Angabe der Streuung. Ebenso wurde nur ein Löser betrachtet. Kommerzielle Löser sind auf ganzzahligen Problemen regelmäßig um Größenordnungen schneller als GLPK, was den Umschlagpunkt nach oben verschieben dürfte – vermutlich deutlich.

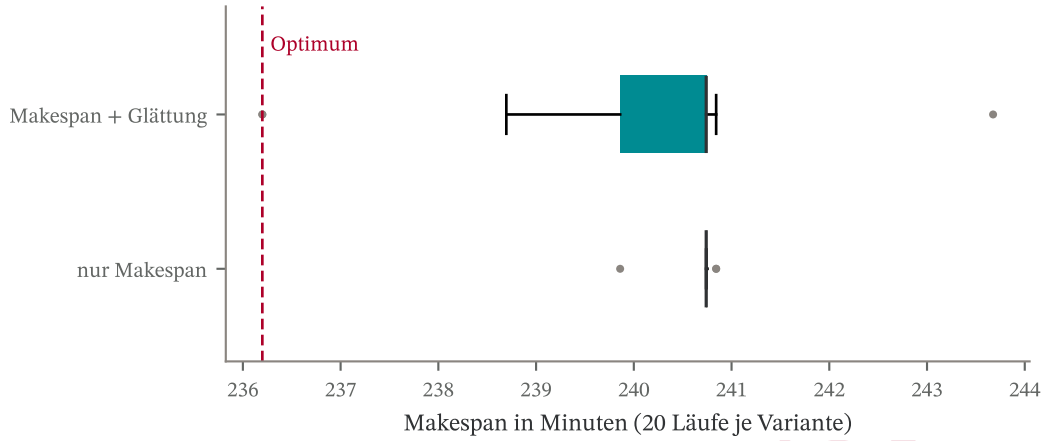


Abbildung 6.3: Streuung des Makespans über je 20 Läufe. Mit der rohen Zielgröße als Energie erreicht kein Lauf das Optimum; mit dem Glättungsterm gelingt es.

		MILP (GLPK)			Simulated Annealing		
Aufg.	Mitarb.	Makespan min	Zeit s	Status	Makespan min	Zeit s	Abstand %
24	8	276,77	1,1	optimal	280,80	0,3	1,46
40	10	381,45	0,0	optimal	381,45	0,4	0,00
60	12	529,71	30,0	Zeitschranke	529,33	0,5	−0,07
80	14	489,17	30,0	Zeitschranke	484,03	0,7	−1,05
120	18	707,27	30,0	Zeitschranke	707,27	1,0	0,00

Tabelle 6.1: Beide Verfahren auf Zufallsinstanzen wachsender Größe. Ein negativer Abstand bedeutet, dass Simulated Annealing die beste Lösung schlägt, die der Löser innerhalb der Zeitschranke gefunden hat.

Die Aussage „ab etwa 60 Arbeitsgängen lohnt sich die Heuristik“ ist deshalb genauer zu lesen als: „mit diesem Löser, dieser Zeitschranke und dieser Instanzstruktur“. Die Struktur des Arguments bleibt gültig, die Zahl ist keine Konstante.

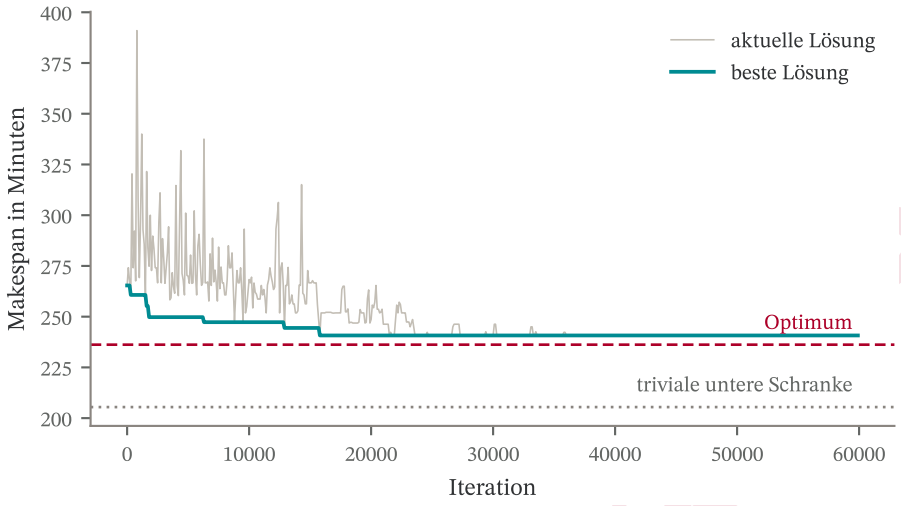
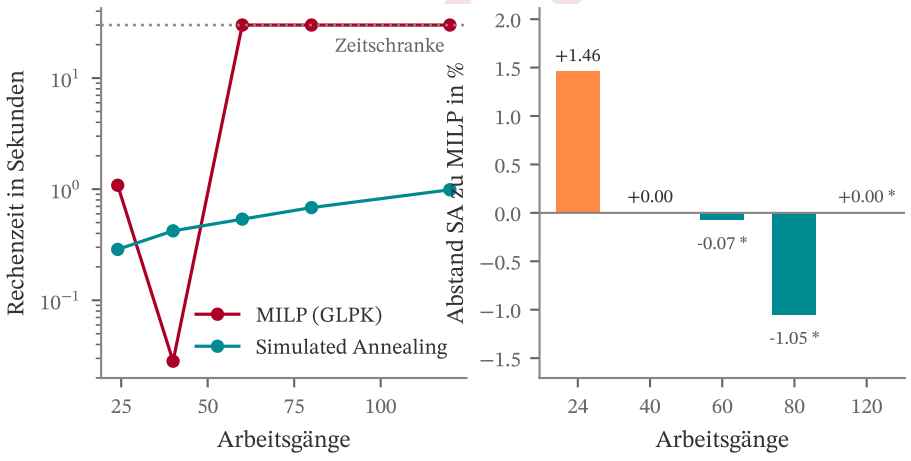


Abbildung 6.4: Verlauf eines einzelnen Laufs: aktuelle und beste Lösung, dazu Optimum und triviale untere Schranke.



* MILP an der Zeitschranke abgebrochen, Optimalität nicht bewiesen

Abbildung 6.5: Rechenzeit (links, logarithmisch) und Abstand der Lösungen (rechts) über die Zahl der Arbeitsgänge.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Diese Arbeit hat die Verteilung von Arbeitsgängen auf Mitarbeitende bei Erpolino als Optimierungsproblem formuliert und zwei Wege verglichen, es zu lösen.

Das Problem ließ sich als ganzzahliges lineares Programm formulieren, in der Notation des Scheduling als $R \mid M_j \mid C_{\max}$. Für das betrachtete Fertigungslos aus 24 Arbeitsgängen und 8 Mitarbeitenden liefert GLPK in 0,7 Sekunden eine beweisbar optimale Gesamtdauer von 236,20 Minuten. Gegenüber der bisherigen Praxis, die ungefähr einer Greedy-Regel entspricht und 266,67 Minuten benötigt, ist das eine Ersparnis von rund einer halben Stunde je Los.

Die Forschungsfrage lässt sich damit beantworten: Ja, die Verteilung lässt sich sinnvoll optimieren, und der Ertrag ist messbar. Der überraschende Teil der Antwort betrifft das Wie.

7.2 Stärken und Schwächen der Verfahren

7.2.1 Das exakte Verfahren

Seine Stärke ist die *Gütegarantie*. Der Löser liefert nicht nur eine Lösung, sondern den Beweis, dass es keine bessere gibt. Diese Aussage kann keine Heuristik machen, und sie ist in der Diskussion mit der Arbeitsvorbereitung viel wert: Sie beendet die Frage, ob nicht doch noch etwas herauszuholen wäre.

Hinzu kommt die Beschreibbarkeit. Das Modell in Anhang B umfasst rund 20 Zeilen und ist ohne Programmierkenntnisse lesbar. Ändert sich eine Anforderung, ändert sich eine Nebenbedingung – nicht ein Algorithmus.

Seine Schwäche ist das Laufzeitverhalten. Es ist im schlechtesten Fall exponentiell, und Abschnitt 6.4 zeigt, dass der schlechteste Fall nicht theoretisch bleibt: Ab 60 Arbeitsgängen erreicht GLPK die Zeitschranke, ohne die Optimalität zu belegen. Zudem ist die Rechenzeit nicht vorhersagbar – die Instanz mit 40 Arbeitsgängen war leichter als die mit 24.

7.2.2 Simulated Annealing

Seine Stärke ist die planbare Rechenzeit. Sie hängt an der Zahl der Iterationen, nicht an der Instanz, und wächst über den untersuchten Bereich nur schwach. Wo der Löser aufgibt, liefert das Verfahren weiterhin brauchbare Lösungen, bei 80 Arbeitsgängen sogar bessere als der Löser in 30 Sekunden.

Seine Schwächen sind gravierender, als es die Zahlen nahelegen. Es gibt keine Gütegarantie: Ohne Vergleichswert bleibt unbekannt, wie weit eine Lösung vom Optimum entfernt ist. Genau

dieser Vergleichswert fehlt aber in dem Bereich, in dem das Verfahren gebraucht wird – dort, wo der Löser versagt. Das ist kein Detail, sondern die eigentliche Schwäche der Methode.

Hinzu kommt die Empfindlichkeit gegenüber Entscheidungen, die nicht aus dem Problem folgen. Abschnitt 6.2 hat gezeigt, dass die Wahl der Energie über Erfolg und Misserfolg entscheidet: Mit der naheliegenden rohen Zielgröße erreicht kein einziger Lauf das Optimum, mit dem Glättungsterm gelingt es. Abkühlplan, Nachbarschaft und Iterationszahl wirken ähnlich. Nichts davon steht im Modell; alles davon muss erprobt werden.

7.2.3 Empfehlung

Für Erpolino in der heutigen Größenordnung lautet die Empfehlung eindeutig: das exakte Verfahren. Es ist schneller, besser und liefert eine Garantie. Die Heuristik ist die aufwendigere Lösung des leichteren Problems.

Das ist ein negatives Ergebnis für den Hauptteil dieser Arbeit, und es ist keines, das sich verstecken lässt. Es ist zugleich das nützlichste Ergebnis: Ein Unternehmen, das die Zuordnung verbessern will, sollte nicht mit einer Metaheuristik beginnen. Es sollte mit einem Modell beginnen.

Die Heuristik wird interessant, sobald die Instanz wächst – durch mehr Arbeitsgänge, weitere Standorte oder einen längeren Planungshorizont. Der gemessene Umschlag bei etwa 60 Arbeitsgängen gilt für GLPK; mit einem kommerziellen Löser dürfte er deutlich später liegen.

7.3 Ausblick

Drei Richtungen bieten sich an.

Das Modell erweitern. Die Annahmen aus Abschnitt 3.1 sind spürbare Vereinfachungen. *Rüstzeiten*, Reihenfolgeabhängigkeiten und Fälligkeitstermine fehlen. Termine ließen sich als Nebenbedingung ergänzen, ohne die Struktur zu ändern; Reihenfolgeabhängigkeiten führen dagegen auf ein anderes Problem – ein Job-Shop –, das erheblich schwerer ist (Pinedo, 2016).

Die Zielgröße hinterfragen. Der Makespan behandelt alle Aufgaben gleich. In der Praxis sind manche Aufträge dringender als andere. Eine gewichtete Zielgröße oder die Summe der Verspätungen wäre näher an dem, was die Arbeitsvorbereitung tatsächlich abwägt.

Die Datengrundlage verbreitern. Die Effizienzwerte der Qualifikationsmatrix stammen aus Erfahrungswerten. Ob sie stimmen, ist offen, und das Ergebnis hängt unmittelbar an ihnen. Aus dem Enterprise Resource Planning (ERP)-System ließen sich Ist-Zeiten gewinnen und die Matrix empirisch schätzen. Ohne diesen Schritt optimiert das Verfahren zuverlässig auf Zahlen, deren Herkunft niemand belegen kann – und das ist am Ende das größere Risiko als die Wahl des Verfahrens.

Literatur

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41–51. https://doi.org/10.1007/BF00940812

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Graham, R. L. (1969). Bounds on Multiprocessing Timing Anomalies. *SIAM Journal on Applied Mathematics*, *17*(2), 416–429. https://doi.org/10.1137/0117039

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, *28*(3), 497–520. https://doi.org/10.2307/1910129

Lenstra, J. K., Shmoys, D. B., & Tardos, É. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, *46*(1–3), 259–271. https://doi.org/10.1007/BF01585745

Makhorin, A. (2020). *GLPK – GNU Linear Programming Kit* (5.0). Free Software Foundation. Verfügbar 17. Juli 2026 unter https://www.gnu.org/software/glpk/

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems* (5. Aufl.). Springer. https://doi.org/10.1007/978-3-319-26580-3

van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated Annealing: Theory and Applications*. Springer. https://doi.org/10.1007/978-94-015-7744-1

Wolsey, L. A. (2020). *Integer Programming* (2. Aufl.). Wiley.

A Quelltext des Verfahrens

Der vollständige Quelltext liegt dem *ergänzenden Material* bei. Wiedergegeben ist hier der Kern des Verfahrens.

```
nachbar-anfang
```

Listing A.1: Nachbarschaft und Abkühlung des Verfahrens (Auszug aus `Code/sa.py`)

B Modell in MathProg

```
/* ------------------------------------------------------------------------
   Zuordnung von Arbeitsgängen zu Mitarbeitern bei minimaler Gesamtdauer.

   Erpolino Metallverarbeitung GmbH, Reutlingen.

   Scheduling-Notation: R|M_j|C_max -- unabhängige parallele Maschinen
   (jeder Mitarbeiter braucht für dieselbe Aufgabe unterschiedlich lange),
   Qualifikationsschranken (M_j), Zielgröße Gesamtdauer (C_max).

   Aufruf:  glpsol --math erpolino.mod --data erpolino.dat

   Die Datendatei erpolino.dat wird von daten.py erzeugt.
   ------------------------------------------------------------------------ */

set MITARBEITER;
set AUFGABEN;

/* Nur Paare, für die die Qualifikation vorliegt. Unzulässige Paare
   existieren gar nicht erst -- das ist schärfer und kleiner als sie mit
   einer großen Zahl zu bestrafen. */
set ZULAESSIG within {MITARBEITER, AUFGABEN};

/* Bearbeitungszeit in Minuten. */
param p{(i,j) in ZULAESSIG} > 0;

/* x[i,j] = 1, wenn Mitarbeiter i den Arbeitsgang j übernimmt. */
var x{(i,j) in ZULAESSIG}, binary;

/* Gesamtdauer: die Zeit, zu der auch der letzte Mitarbeiter fertig ist. */
var Cmax >= 0;

minimize Gesamtdauer: Cmax;

/* Jeder Arbeitsgang wird genau einmal vergeben. */
s.t. Vergabe{j in AUFGABEN}:
  sum{(i,j) in ZULAESSIG} x[i,j] = 1;

/* Cmax ist mindestens so groß wie die Auslastung jedes Mitarbeiters.
   Zusammen mit der Minimierung erzwingt das Cmax = max_i (Auslastung). */
s.t. Auslastung{i in MITARBEITER}:
  sum{(i,j) in ZULAESSIG} p[i,j] * x[i,j] <= Cmax;

solve;

/* ---- Ausgabe: maschinenlesbar für die Weiterverarbeitung ------------- */
```

```
printf "# Gesamtdauer (Cmax) in Minuten\n";
printf "CMAX %.4f\n", Cmax;

printf "# Zuordnung: Mitarbeiter Aufgabe Dauer\n";
for {(i,j) in ZULAESSIG: x[i,j] > 0.5}
  printf "ZUORDNUNG %s %s %.4f\n", i, j, p[i,j];

printf "# Auslastung je Mitarbeiter in Minuten\n";
for {i in MITARBEITER}
  printf "AUSLASTUNG %s %.4f\n", i, sum{(i,j) in ZULAESSIG} p[i,j] * x[i,j];

end;
```

Listing B.1: Das vollständige Modell in *MathProg* (`Code/erpolino.mod`)

Index

Verwendete KI-Tools

Für diese Arbeit wurden Werkzeuge der künstlichen Intelligenz zur sprachlichen Überarbeitung einzelner Absätze eingesetzt. Modellbildung, Implementierung, Auswertung und Bewertung der Ergebnisse stammen von der Verfasserin beziehungsweise vom Verfasser.

Erklärung zur wissenschaftlichen Arbeit

Ich versichere hiermit, die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Allgemeine Studien- und Prüfungsordnung für das Bachelor- und Masterstudium der Hochschule Reutlingen, die fachspezifische Studien- und Prüfungsordnung und die Regeln zur Sicherung der guten wissenschaftlichen Praxis der Hochschule Reutlingen beachtet zu haben.

Diese Arbeit oder Teile dieser Arbeit sind weder Bestandteil einer anderen Prüfungsleistung an dieser noch an einer anderen wissenschaftlichen Institution.

Ich versichere außerdem, dass die Nutzung von KI-Tools von der/m Prüfenden explizit erlaubt wurde und ich in meiner Arbeit davon Gebrauch gemacht habe. Die verwendeten KI-Tools sind der Arbeit nach fakultätsinternen Zitier- und Dokumentationsvorgaben angefügt. Ich habe inhaltsgenerierende KI-Tools lediglich unterstützend eingesetzt und die Kernthesen meiner Arbeit nicht von der KI übernehmen lassen. Ich bin mir bewusst, dass die Nutzung von KI-Tools keine Garantie für Richtigkeit bietet und ich die Verantwortung für alle KI-generierten Inhalte in dieser Arbeit trage.

Reutlingen, 24. Dezember 2026

Jean Dujardin