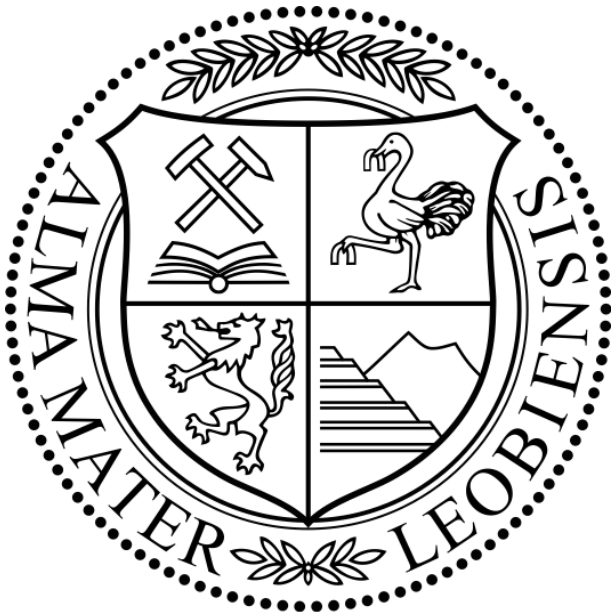
Ein kombiniertes Zuordnungs- und Reihenfolgeproblem in der Produktionsplanung

Christina Fuchs

Oktober 2011

Diplomarbeit von Christina Fuchs ©



betreut von

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Norbert Seifter

Lehrstuhl für Angewandte Mathematik
Montanuniversität Leoben
Österreich

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich diese Arbeit selbstständig verfasst, andere als die angegebenen Quellen nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Leoben, am 11. Oktober 2011 Christina Fuchs, BSc

Danksagung

Ich möchte mich recht herzlich bei allen Personen bedanken, die mir beim Verfassen meiner Diplomarbeit geholfen haben.

Ein ganz besonderer Dank gebührt meinen Eltern, Monika und Gebhard. Durch ihr bedingungsloses Vertrauen und ihre uneingeschränkte Unterstützung in allen Lebenslagen, war es mir erst möglich mein Studium erfolgreich zu absolvieren.

Bei meinem Betreuer Norbert möchte ich mich für die engagierte Zusammenarbeit, die Hilfe bei der Literaturrecherche und die Bereitstellung seiner Fachkompetenzen bedanken.

Ein großes Dankeschön möchte ich auch meiner restlichen Familie und meinen Freunden aussprechen. Besonders hervorheben möchte ich dabei Christoph. Sein kritisches Interesse und seine konstruktiven Ratschläge haben maßgeblich zum Erfolg dieser Arbeit beigetragen.

Kurzfassung

Diese Diplomarbeit untersucht eine in der Literatur in dieser Form noch undokumentierte Aufgabenstellung. Im Konkreten handelt es sich um eine Kombination aus einem *mehrere Maschinen Zuordnungs-* und einem *Reihenfolgeproblem*. Dabei sollen viele Aufträge effizient auf eine definierte Anzahl an identischen Produktionslinien, welche sich aus unterschiedlichen Maschinen zusammensetzen, verteilt und innerhalb dieser sinnvoll gereiht werden. Zur Lösung der beschriebenen Problemstellung werden verschiedene Algorithmen entwickelt, angepasst und implementiert. In diesem Zusammenhang ergeben sich Verfahrenskombinationen, die das Produktionsprogramm schrittweise erstellen und verbessern, mit dem Ziel die maximale Gesamtdurchlaufzeit der Produktionslinien zu minimieren. Die logische Abfolge der Prozesskette lässt sich wie folgt beschreiben: Zu Beginn werden die Aufträge nach gewissen Prioritätsregeln den verfügbaren Produktionslinien zugeordnet. Dann erfolgt die Optimierung der zuvor erstellten Auftragsverteilung mit anschließender Verbesserung der Abarbeitungsreihenfolge. Zuletzt werden entstandene Kapazitätslücken mittels Korrekturprozedur vermindert. Die Evaluierung der Methoden liefert eine Empfehlung für die stabilsten und zuverlässigsten Algorithmen, welche die Aufgabenstellung innerhalb einer vorgegebenen, beschränkten Laufzeit bewältigen.

Abstract

This master thesis investigates a yet undocumented topic. In detail the paper deals with a combined *multi machine scheduling-* and a *sequencing problem*. The task is, to allocate efficiently several jobs to a defined number of identical production lines, which are composed of different machines. Additionally, the jobs should be arranged within the lines in an order, which reduces the time used to execute all tasks. Various algorithms have been developed, adapted and implemented to solve the specified problem. The methods are combined to build and improve the production plan incrementally. The goal is, to minimize the highest total completion time of all production lines. The process steps are: First the jobs are assigned to the available production lines based on certain priority rules. The generated production plan is optimized and the job sequence is improved. Last but not least capacity gaps are closed through a correction procedure. The evaluation gives an advice about the most stable and reliable algorithms, which are able to find good solutions under certain runtime restrictions.

Inhaltsverzeichnis

Kapitel 1

Einführung

Das Thema dieser Diplomarbeit sind Zuordnungs- und Reihenfolgeprobleme in der Produktionsplanung und -steuerung. Um Zuordnungs- und Reihenfolgeprobleme zu lösen, werden zwei Ansätze kombiniert: Schedulingalgorithmen[1] und Reihenfolgealgorithmen. Die nachfolgenden Erläuterungen liefern eine kurze Beschreibung dieser beiden Vorgangsweisen.

1.1 Scheduling

Die ersten Schedulingalgorithmen wurden bereits in den fünfziger Jahren entwickelt. Seither haben sich die Einsatzgebiete dieser mathematischen Modelle jedoch stark ausgedehnt. Ursprünglich wurden diese Algorithmen dazu verwendet, die Leistung von Computersystemen zu optimieren. Heutzutage allerdings decken sie ein breites Spektrum an Anwendungsbereichen ab. Beispiele dafür sind die Fertigungsindustrie, Dienstleistungsunternehmen, Informations- und Kommunikationsnetze, sowie Transport- und Distributionslogistik. [3]

Da Schedulingalgorithmen für die Fertigungsindustrie, im Konkreten für die Produktionsplanung und -steuerung analysiert werden, wird nun im Detail auf diese spezifische Aufgabenstellung eingegangen. Die betrachteten Probleme können außerdem in fünf Hauptgruppen unterteilt werden.

Bei der Ablauf- und Terminplanung steht die Zuteilung von Aufgaben (Aufträgen) zu vorhandenen Kapazitäten (Produktionslinien) im Mittelpunkt. Der wichtigste Aspekt dabei ist, dass mit der generierten Aufgabenverteilung ein oder mehrere definierte Ziele (z.B. Termin-

[1] Englisch für Ablaufplanung, Terminplanung

treue, Minimierung der Durchlaufzeit, Minimierung der Kosten) erreicht werden. Um die Zielrealisierung zu gewährleisten, werden komplexe mathematische Modelle und Heuristiken verwendet. [16]

Neben den mathematischen Modellen und Heuristiken trägt vor allem die Vielfalt der unterschiedlichen Eigenschaften einer Produktion zur umfangreichen Problemdefinition bei. Einige dieser Charakteristika sollen hier erwähnt werden: Die Anzahl identischer Produktionslinien, die Anzahl der Arbeitsplätze auf jeder Maschine, die Aufteilung der Auftragsabarbeitung, die Abhängigkeit zwischen Aufträgen und die Liefertermine von Aufträgen. Anwendungsgebiete für Scheduling sind: [3]

Auftragsplanung

Charakteristikum: Kurzfristige Auftragsplanung.
Ziel: Minimierung der Gesamtdurchlaufzeit, Minimierung der Anzahl verspäteter Aufträge.
Anwendung: Herstellung von Produktionsgütern.

Eine oder mehrere Maschinen

Ein Auftrag muss nur auf einer beliebigen Produktionslinie bearbeitet werden.

Job Shop

Ein Auftrag muss auf mehreren Produktionslinien bearbeitet werden.

Flow Shop

Erweiterung des Job Shop Problems. Die Reihenfolge der Auftragsabarbeitung muss immer gleich bleiben. → Kein Überholen möglich (FIFO - Prinzip).

Produktionsplanung

Charakteristikum: Mittel- bis langfristige Bestandsplanung.
Ziel: Minimierung der Umrüstkosten, Minimierung des Lagerbestands.
Anwendung: Raffinerien, Papierfabriken.

Lot Scheduling

Mehrere Produkte müssen sequenziell auf einer Produktionslinie hergestellt werden.
→ Umrüstkosten.

Planung von Fördersystemen

Charakteristikum: Taktung von Fördersystemen.
Ziel: Maximierung der Durchsatzmenge.
Anwendung: Automobilindustrie.

Projektmanagement

Charakteristikum: Abhängigkeit zwischen den Aufgaben, unendliche Ressourcen.
Ziel: Frühestmögliche Beendigung der letzten Aufgabe.
Anwendung: Consulting Projekte, Errichtung von Gebäuden.

Supply Chain Management

Charakteristikum: Hierarchisches Konzept, welches eine Kombination aus Auftrags- und Produktionsplanungsmodell darstellt.
Ziel: Minimierung des Lagerbestands, Minimierung der Transportkosten.
Anwendung: Supply Chain. [16]

1.2 Reihung

Bei der Reihenfolgeoptimierung steht die Findung einer Aufgabenabarbeitungssequenz im Mittelpunkt. Für die Bestimmung dieser zeitlichen Reihenfolge existieren verschiedene Ansätze, die nun im Detail erklärt werden.

Kombinatorische Methoden

Die Verwendung einer *kombinatorischen Methode* beinhaltet die Konstruktion aller möglichen Permutationen an Produktionsprogrammen. Anschließend wird der Belegungsplan, der ein vorgegebenes Zielkriterium am besten erfüllt, ausgewählt. Da die Anzahl der möglichen Abarbeitungsreihenfolgen mit $n!$ steigt, wenn n der Anzahl der Aufträge entspricht, ist dieser Ansatz für größere Probleme unbrauchbar. [21]

Ganzzahlige Lineare Planung

Bei der *ganzzahligen linearen Planung* wird das logistische System mit Hilfe von Gleichungen und Ungleichungen beschrieben. Dadurch können Einschränkungen des Systems, wie etwa Maschinenbelastungen, Liefertermine, Transportzeiten und vieles mehr, berücksichtigt

werden. Da die Lösungsverfahren, wie zum Beispiel die Varianten des Simplex zur Lösung ganzzahliger linearer Optimierungsprobleme, im schlechtesten Fall exponentielle Laufzeit haben, ist auch diese Methode in der Praxis oft nicht anwendbar. [21]

Prioritätsregeln

Bei der Reihung mit Hilfe von *Prioritätsregeln* handelt es sich um ein Näherungsverfahren. Dabei wird das Produktionsprogramm anhand eines Vergleichsparameters festgelegt. Diese Kenngröße beschreibt verschiedene Eigenschaften des logistischen Systems. Nachfolgend werden exemplarisch einige dieser Reihenfolge- und Prioritätsregeln angeführt. [1]

FIFO (First In First Out)

Der Auftrag, der als erster ins System eingebucht wird, wird als erster auf der Produktionslinie bearbeitet.

KOZ (Kürzeste Operationszeit)

Der Auftrag mit der geringsten Bearbeitungszeit wird als erster auf der Produktionslinie bearbeitet.

LOZ (Längste Operationszeit)

Der Auftrag mit der größten Bearbeitungszeit wird als erster auf der Produktionslinie bearbeitet.

GSZ (Geringste Schlupfzeit)

Der Auftrag mit der geringsten Differenz zwischen Restbearbeitungszeit und Liefertermin wird als erster auf der Produktionslinie bearbeitet.

FPE (Frühester Plan - Endtermin)

Der Auftrag mit dem frühesten Fertigstellungsdatum wird als erster auf der Produktionslinie bearbeitet. [1]

1.3 Problemstellung dieser Arbeit

Es soll eine Produktion simuliert werden, die aus mehreren identischen Produktionslinien l_i besteht, wobei $i = \{1, \ldots, s\}$ und $s \in \{1, 2, 3, 4\}$ die Anzahl an Produktionslinien ist. Jede dieser Produktionslinien setzt sich aus mehreren Maschinen m_j zusammen, wobei $j = \{1, \ldots, k\}$ und $1 \leq k \leq K$ die Anzahl der verfügbaren Maschinen ist. Dabei ist zu be-

achten, dass jeder Auftrag auf jeder Maschine bearbeitet werden muss, sich Aufträge aber nicht überholen dürfen. Des Weiteren wird eine Kundenauftragsliste mit Aufträgen a_x in dieses Produktionssystem eingebucht, wobei $x = \{1, \ldots, n\}$ und $1 \leq n \leq 200$ die Anzahl an zu bearbeitenden Aufträgen ist. Für jeden dieser Aufträge gibt es maschinenabhängige Bearbeitungszeiten tb_x. Außerdem müssen Rüstzeiten tr_x miteinbezogen werden, die sowohl vom vorherigen Auftrag, als auch von der aktuellen Maschine abhängen. Da jeder Auftrag ein Unikat ist, kann ein Auftrag a_x nicht der Vorgänger a_y von sich selbst sein. Das heißt, wenn $y = x$, dann $r_{jy}^{(x)} = 0$.

Bearbeitungszeit $tb_x = \begin{bmatrix} b_1^{(x)} & \cdots & b_k^{(x)} \end{bmatrix}$ **Rüstzeit** $tr_x = \begin{bmatrix} r_{11}^{(x)} & \cdots & r_{1n}^{(x)} \\ \vdots & \ddots & \vdots \\ r_{k1}^{(x)} & \cdots & r_{kn}^{(x)} \end{bmatrix}$

Da die entwickelten Scheduling- und Reihenfolgealgorithmen in einem Simulationsprogramm für die Fertigungsindustrie eingesetzt werden, spielt die Zeit, welche benötigt wird, um das Produktionsprogramm zu erstellen, eine sehr wichtige Rolle. Die Vorgabe dafür lautet maximal 10 Sekunden, auf einem derzeit durchschnittlichen Computer.

Zu guter Letzt wird die Aufgabenstellung mit Hilfe der oben angeführten Einteilung kategorisiert. Dieses Ablauf- und Terminplanungsproblem kann der Klasse der *Auftragsplanungsprobleme*, genauer dem *mehrere Maschinen Problem*, zugeordnet werden. Der Reihenfolgealgorithmus verwendet das *kombinatorische Prinzip*. Allerdings werden nicht, wie beschrieben, alle möglichen Permutationen an Auftragsreihenfolgen generiert. Stattdessen wird versucht, durch kontinuierliche Verbesserung einer Startlösung, eine Minimierung der maximalen Gesamtdurchlaufzeit zu erreichen.

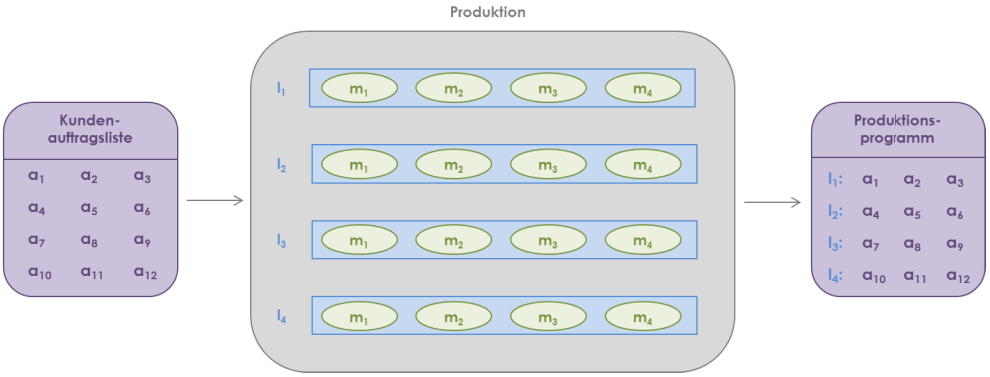


Abbildung 1.1: Diese Grafik zeigt die eben beschriebene Aufgabenstellung, die eine Kombination aus einem *mehrere Maschinen Scheduling-* und einem *Reihenfolgeproblem* ist.

1.4 Forschungsfrage

Das Ziel dieser Diplomarbeit ist es, diverse Scheduling- und Reihenfolgealgorithmen in JAVA[2] zu implementieren und diese zu bewerten. Mit dieser Evaluierung soll es möglich sein den Algorithmus zu identifizieren, welcher das beste Produktionsprogramm liefert und somit die Kostenfunktion minimiert. Mit anderen Worten, es wird die Produktion mit der geringsten Gesamtdurchlaufzeit T gesucht, wobei $T = \max_{1 \leq i \leq s}\{T_i\}$.

[2] © Oracle, United States of America, www.oracle.com

Kapitel 2

Logistikrelevanz

Dieses Kapitel befasst sich mit der Eingliederung von Scheduling- und Reihenfolgealgorithmen in die logistische Wertschöpfungskette.

2.1 Logistikkette

Eine *Logistikkette* beschreibt das logistische System eines Industrieunternehmens. Das bedeutet, dass sowohl Transport- und Lager- als auch Produktionsprozesse abgebildet werden. Mit Hilfe dieser Prozesse ist es möglich, die Bereiche Beschaffung, Produktion, Distribution und Entsorgung zu definieren. Hervorzuheben ist dabei das Tätigkeitsfeld *Produktionslogistik*. Sie nimmt nicht nur eine zentrale Stellung in der Wertschöpfungskette ein, sondern weist auch eine enge Verzahnung mit Transport- und Lagerprozessen auf. Darüber hinaus befasst sie sich intensiv mit der *Produktionsplanung und -steuerung*. [1]

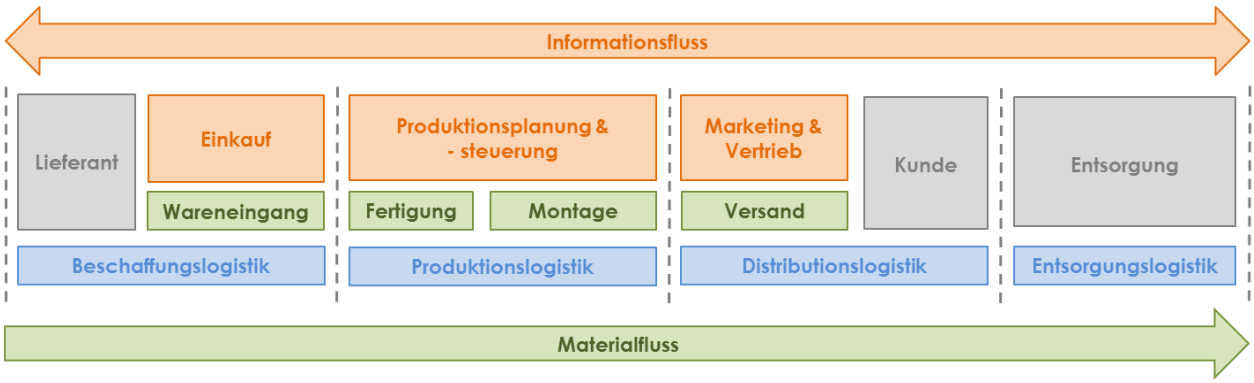


Abbildung 2.1: Diese Grafik zeigt eine Logistikkette. [9]

2.2 Produktionslogistik

Da bereits erwähnt wurde, wie bedeutend die Produktionsplanung und -steuerung im Gesamtsystem eines Fertigungsbetriebes ist, wird dieser Bereich nun im Detail analysiert. Ihre Aufgabengebiete können anhand von drei Planungsebenen abgesteckt werden.

Laut VDI - Richtlinie kann der Begriff der Produktionsplanung und -steuerung wie folgt definiert werden: *Die Produktionsplanung und -steuerung umfasst die räumliche, zeitliche und mengenmäßige Planung, Steuerung und Kontrolle des gesamten Geschehens im Produktionsbereich.* [1]

Strategische Produktionsplanung

Im Zuge der *strategischen Produktionsplanung* werden die Geschäftsfelder, sowie die Produktionsstandorte und die Produktionsorganisation langfristig festgelegt. [1]

Taktische Produktionsplanung

Mittelfristig werden in der *taktischen Produktionsplanung* die Produktpalette und die zur Herstellung dieser Produkte benötigten Betriebsmittel definiert. Darauf aufbauend muss die Humanressourcenplanung erfolgen. [1]

Operative Produktionsplanung

Die letzte Stufe bildet die kurzfristige oder *operative Produktionsplanung*, die in folgende Tätigkeitsschwerpunkte aufgeteilt wird.

Planung des Produktionsprogramms

Charakteristikum: Festlegung der Absatzmenge der Produkte, Kapazitätsabgleich.
Einflussfaktoren: Entwicklung der Absatzmärkte, Verfügbarkeit der Kapazitäten in den einzelnen Produktionsstandorten, Rahmenverträge mit Lieferanten, saisonale Bedarfsschwankungen.

Materialbedarfsplanung

Charakteristikum: Bestimmung der benötigen Menge an Baugruppen, Einzelteilen und Rohstoffen.
Methode: Stücklistenauflösung → MRP (Material Requirements Planning).

Ablauf- und Terminplanung

Charakteristikum: Festlegung der Losgrößen, Zuordnung der Aufträge zu den verfügbaren Kapazitäten, Festlegung der Abarbeitungsreihenfolge innerhalb einer Kapazität. [1]

Abschließend ist zu sagen, dass das beschriebene Scheduling- und Reihenfolgeproblem dieser Diplomarbeit der Kategorie *operative Produktionsplanung* zugeteilt wird. Dezidiert handelt es sich dabei um den letzten Schritt dieser Planungsebene, die *Ablauf- und Terminplanung*.

Kapitel 3

Mathematische Problemspezifikation

Dieses Kapitel befasst sich mit der Klassifizierung von NP - Problemen, sowie der Einordnung des Schedulingproblems dieser Diplomarbeit in dieses Schema.

Die nächsten Abschnitte liefern eine klare Abgrenzung zwischen den Begriffen NP - Problem, P - Problem und NP - vollständiges Problem. Zuvor muss allerdings noch erwähnt werden, dass alle Aufgabenstellungen, die zu der Klasse der NP - Probleme gehören, sogenannte Entscheidungsprobleme sind. Das bedeutet, dass ein Algorithmus auf eine definierte Eingabe, nur *ja* oder *nein* als Antwort zurück gibt.

3.1 NP - Problem

Non deterministic polynomial time Probleme sind Aufgabenstellungen die nicht deterministisch, mit anderen Worten höchstens durch das Generieren zufälliger Ergebnisse, in polynomialer Laufzeit gelöst werden können. Daraus ergibt sich die Frage: Gibt es auch Algorithmen, die deterministisch und in polynomialer Zeit die optimale Lösung für ein NP - Problem generieren? [13]

3.2 P - Problem

Zu der Klasse der *polynomial time* Probleme zählen alle Entscheidungsprobleme für die bereits bewiesen ist, dass es effiziente, deterministische[1] Algorithmen gibt, welche die Aufgabenstellung in Polynomialzeit bewältigen. Das heißt, die obere Schranke für die Laufzeit

[1] Jeder Handlungsschritt ist zu jedem Zeitpunkt eindeutig definiert.

kann mit $\mathcal{O}(n^k)$ definiert werden, wobei n die Größe der Aufgabenstellung und $k \in \mathbb{N}$ eine Konstante ist. Diese Probleme können auch als NP - leicht bezeichnet werden. Beispiele dafür sind Sortieralgorithmen.

Weiters gilt, dass alle P - Probleme der Menge der NP - Probleme angehören $\mathcal{P} \subseteq \mathcal{NP}$. Folglich ergibt sich die Diskussion, ob $\mathcal{P} = \mathcal{NP}$ ist. Diese Frage beschäftigt Wissenschaftler schon seit Jahrzehnten. Zum heutigen Zeitpunkt kann diese Frage nicht eindeutig beantwortet werden. Da bisher aber noch kein Algorithmus gefunden wurde, mit dem es möglich ist, ein NP - vollständiges Problem in polynomialer Laufzeit exakt zu lösen, gilt $\mathcal{P} \neq \mathcal{NP}$. [2]

3.3 NP - schweres Problem

Im Allgemeinen ist ein *NP - schweres* Problem mindestens so schwer, wie jede andere Aufgabenstellung aus $\mathcal{NP}$. Konkret wird von einem NP - schweren Problem Q gesprochen, wenn es möglich ist, ein Problem L polynomial auf ein Problem Q zu reduzieren ($L \,\alpha\, Q$) und wenn $L \in \mathcal{NP}$.

Polynomiale Reduktion

Ein Problem A ist polynomial reduzierbar auf ein Problem B ($A \,\alpha\, B$), wenn es einen Algorithmus mit polynomialer Laufzeit gibt, der Instanzen von A in Instanzen von B überführt, sodass die Instanzen von A und die transformierten Instanzen das selbe Ergebnis liefern. [18]

3.4 NP - vollständiges Problem

Zu guter Letzt wird der Begriff *NP - vollständig* definiert. Ein Problem Q kann der Gruppe der NP - vollständigen Probleme zugeordnet werden, wenn es folgende Eigenschaften erfüllt.

- Q gehört zur Klasse der NP - Probleme, d.h. $Q \in \mathcal{NP}$.
- Q ist NP - schwer, d.h. ($L \,\alpha\, Q$) und $L \in \mathcal{NP}$. [2]

Der erste Beweis für ein NP - vollständiges Problem wurde von Stephen A. Cook im Jahre 1971 erbracht. Ihm ist es gelungen, aufzuzeigen, dass das Erfüllbarkeitsproblem[2] dieser speziellen Problemgruppe angehört.

[2] Englisch: Boolean Satisfiability Problem

In den folgenden Jahren hat Richard Karp eine Vielzahl an weiteren NP - vollständigen Problemen identifiziert und die Beweise dafür geliefert. [18]

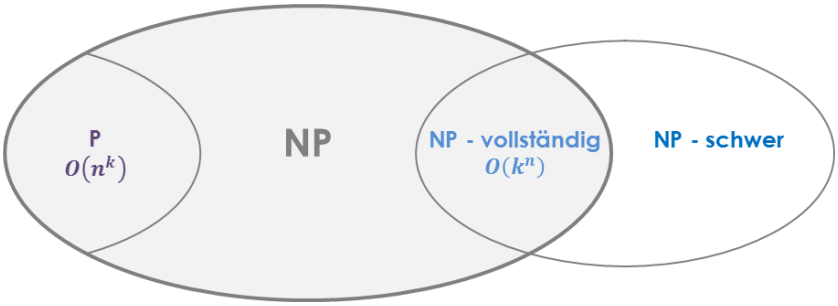


Abbildung 3.1: Diese Grafik zeigt das Zusammenspiel von NP - Problemen, P - Problemen und NP - vollständigen Problemen.

Beispiele

Zum Abschluss werden noch einige wichtige Beispiele genannt, die neben den meisten Maschinen Schedulingproblemen zur Gruppe der NP - vollständigen Probleme gehören.

Cliquenproblem

Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit V Knoten und E Kanten. Besitzt G eine Clique bzw. einen vollständigen Teilgraphen[3] mit $k \in \mathbb{N}$ Knoten?

Hamiltonkreis

Gegeben sei ein gerichteter Graph $G = (V, E)$. Besitzt G einen Hamiltonkreis oder mit anderen Worten einen geschlossenen Pfad, der jeden Knoten genau einmal enthält?

Rucksackproblem

Gegeben sei eine Menge an Objekten $(Gewicht, Wert)$ die optimal in einen Rucksack gepackt werden sollen. Ist es möglich eine Packvariante zu finden, die den Gesamtwert maximiert und dabei das festgesetzte Maximalgewicht nicht überschreitet?

Partitions- oder Zahlenaufteilungsproblem

Gegeben sei eine Menge $M = \{a_1, \ldots, a_n\}$ für die gilt $a_i \in \mathbb{N}$, wobei $1 \leq i \leq n$. Gibt es eine Aufteilung der Menge $I = \{1, \ldots, n\}$ in zwei disjunkte Teilmengen $S_1 \subset I$ und $S_2 \subset I$, sodass $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i$? [13]

[3] Jeder Knoten ist mit jedem anderen Knoten aus dieser Menge verbunden.

3.5 Nachweis der NP - Vollständigkeit

Vor einigen Jahren ist ein Artikel mit dem Titel *Complexity of Maschine Scheduling Problems* [13] erschienen. Dieser Artikel befasst sich hauptsächlich mit der Eingliederung von Maschinen Schedulingproblemen in das NP - Klassifikationsschema. Die Herleitungen dafür sind mit Zitaten belegt.

In diesem Bericht wird erwähnt, dass sich alle traditionellen Maschinen Schedulingprobleme mit Hilfe von fünf Attributen $n \,|\, m \,|\, l, \lambda \,|\, \mathcal{K}$ beschreiben lassen.

- n ... Anzahl der Aufträge der Kundenauftragsliste
- m ... Anzahl identischer Maschinen
- l ... Typ des Maschinen Schedulingproblems (Flow Shop, Job Shop, ...)
- λ ... Zusatzbedingung (Abhängigkeit von Aufträgen, ...)
- $\mathcal{K}$ ... Kostenfunktion

Dabei kann bereits die Modifikation eines einzigen Parameters darüber entscheiden, ob es sich um ein NP - vollständiges Problem handelt oder nicht. Das nachfolgende Beispiel soll diesen Sachverhalt verdeutlichen. [13]

Im Zuge dieser Arbeit sollen Algorithmen entwickelt werden, die die maximale Gesamtdurchlaufzeit $T = \max_{1 \leq i \leq s}\{T_i\}$ minimieren.

Würde dieser Parameter $\mathcal{K}$ leicht variiert werden und die Aufgabenstellung wäre nun die Minimierung der Gesamtdurchlaufzeit $T = \sum_{i=1}^{s}$ aller Produktionslinien, wobei jede Produktionslinie nur aus einer Maschine besteht, dann würde plötzlich ein P - Problem vorliegen. [2]

Unter anderem wird eine Parameterkombination beschrieben, die vereinfacht die Problemstellung dieser Arbeit widerspiegelt. Die Rede ist von einem 2 - Maschinen Schedulingproblem, bei welchem die maximale Gesamtdurchlaufzeit minimiert werden soll. Dabei liegen keine zusätzlichen Restriktionen oder Bedingungen vor. Diese Aufgabenstellung wird bereits der Gruppe der NP - vollständigen Probleme zugeordnet. Diese Arbeit befasst sich jedoch mit einer Auftragszuordnung auf identische Produktionslinien, die sich wiederum aus mehreren Maschinen zusammensetzen. Das bedeutet, das Problem wird um maschinen- und auftragsabhängige Rüstzeiten und somit um ein Reihenfolgeproblem erweitert. Da es sich bei der simplen 2 - Maschinen Ausprägung um ein NP - vollständiges Problem handelt, kann mit ziemlicher Sicherheit davon ausgegangen werden, dass diese komplexere Variante ebenfalls dieser Problemgruppe angehört. [13]

Kapitel 4

Lösungsansätze

Bis jetzt hat sich diese Arbeit vor allem mit der Einordnung und Klassifizierung des kombinierten *Zuordnungs- und Reihenfolgeproblems* für *mehrere Maschinen* befasst. Ab diesem Zeitpunkt soll der Fokus auf die Lösung der Aufgabenstellung gelegt werden.

Prinzipiell wird zwischen *Heuristik* und *Optimierungsverfahren* unterschieden. Ein *Optimierungsverfahren* liefert mit absoluter Sicherheit die optimale Lösung zu einem bestimmten Problem. Der Nachteil dieser Algorithmen ist die extrem lange Rechenzeit. Diese Schwäche macht sich hauptsächlich bei NP - schweren Problemen bemerkbar. Aus diesem Grund finden Heuristiken hier ihr Einsatzgebiet. [5]

Eine *Heuristik* ist eine Vorgehensweise mit Hilfe derer in akzeptabler Zeit eine gute zulässige, aber nur sehr selten die optimale Lösung gefunden wird. Des Weiteren lassen sich Heuristiken in die nachfolgend angeführten Gruppen unterteilen. Nach jeder Verfahrensbeschreibung wird unter dem Punkt *implementiertes Verfahren* bereits auf die umgesetzten Algorithmen verwiesen. In den nächsten Kapiteln wird im Detail auf die Beschreibung und Implementierung eingegangen. [5]

4.1 Eröffnungsverfahren

Ein *Eröffnungsverfahren* wird dazu verwendet, um eine erste zulässige Lösung einer Aufgabenstellung zu finden. Dabei ist die Qualität des Ergebnisses von der Ausgestaltung des Algorithmus, der Einbeziehung von Nebenbedingungen sowie von der dafür in Kauf genommenen Rechenzeit abhängig. [5]

Uninformierte Verfahren

Ein weiterer passender Fachausdruck für *uninformiertes Verfahren* wäre *blinde Suche*. Wie der Begriff schon vorwegnimmt, laufen diese Algorithmen sehr starr nach einer allgemein gültigen Regel ab. Die konkrete Modellstruktur wird nicht zur Lösungsfindung herangezogen. [5]

Beispiel: Nord - West - Eckenregel.

Greedy Heuristiken

Greedy Verfahren versuchen mittels schrittweiser Verbesserung den bestmöglichen Zielfunktionswert der aktuellen Teillösung zu generieren ohne dabei zukünftige Konstruktionsschritte miteinzubeziehen. Diese „gierigen“Algorithmen werden auch als *statische Prioritätsregelverfahren* bezeichnet. [5]

Beispiel: Matrix - Minimum Methode.
Implementierte Verfahren: Best - Fit Algorithmus, First - Fit Algorithmus.

Look - Ahead Heuristiken

Bei *Look - Ahead* oder *vorausschauenden Verfahren* wird nach jedem Schritt abgeschätzt, welche Auswirkung die aktuelle Zuordnung auf zukünftige Verteilungen und somit auf die Qualität der Lösung hat. Das heißt, das Auswahlkriterium für den folgenden Verfahrensschritt wird immer neu gesetzt. Diese Algorithmen werden auch als *dynamische Prioritätsregelverfahren* bezeichnet. [5]

Beispiel: Vogelsche Approximation.

4.2 Verbesserungs- und Suchverfahren

Die erste Stufe bei der Anwendung von *Verbesserungs-* und *Suchverfahren* ist die Bestimmung einer zulässigen Startlösung. Sie wird entweder mittels Eröffnungsverfahren oder zufällig generiert. Die anschließende Nachbarschaftssuche verbessert diese Ausgangssituation schrittweise. Dabei stehen verschiedene Möglichkeiten zur Verfügung.

Erstens ist es möglich, Elemente zu vertauschen. Dabei wechseln ein Auftrag a_x und ein Auftrag a_y ihre Positionen innerhalb einer Produktionslinie l_i. Es wäre auch vorstellbar, dass zwei Aufträge von unterschiedlichen Produktionslinien ausgetauscht werden.

Eine weitere Option wäre eine Verschiebung von Elementen. Ein Auftrag a_x wird im Produktionsprogramm um einen oder mehrere Plätze nach hinten oder vorne verschoben. Somit wird eine Änderung in der Abarbeitungsreihenfolge hervorgerufen.

Zu guter Letzt ist es noch möglich, einzelne Werte von Elementen zu ändern. Bei dieser Variante der Nachbarschaftssuche, wird der aktuelle Index der Produktionslinie, auf welcher der Auftrag a_x bearbeitet wird, um eins erhöht. [4]

Beispiele: k - Opt, Laha & Sarin, Threshold Accepting, Tabu Search, Ameisenalgorithmen.
Implementierte Verfahren: Simulated Annealing, Genetische Algorithmen, Korrekturalgorithmus, Insertion Search.

Reine Verbesserungsverfahren

Ein *reines Verbesserungsverfahren* wird beendet, sobald ein neuer Iterationsschritt keine Verbesserung zur vorherigen Lösung liefert. Problematisch ist die Tatsache, dass es sich bei diesem Endergebnis meist um ein lokales Optimum handelt. [5]

Lokale Suchverfahren

Im Gegensatz zu den reinen Verbesserungsverfahren besteht bei *lokalen Suchverfahren* die Möglichkeit, durch die Akzeptanz schlechterer Ergebnisse, lokale Optima zu verlassen. Diese nicht ausgeschlossene zwischenzeitliche Abwärtsentwicklung gestattet es somit, einen größeren Lösungsraum abzusuchen. Folglich wird die Wahrscheinlichkeit, dass das globale Optimum gefunden wird, erhöht. [5]

4.3 Relaxationsbasierte Verfahren

Bei *relaxationsbasierten Verfahren* wird die Ausgangsaufgabenstellung vereinfacht. Zu diesem Zweck werden Nebenbedingungen eliminiert, Variablentypen vereinfacht (z.B. Vernachlässigung der Ganzzahligkeit) und Zielfunktionen neu ausformuliert. Das simplifizierte Problem wird anschließend gelöst. Zum Schluss wird das Ergebnis auf den ursprünglichen Sachverhalt rücktransformiert. [1]

4.4 Unvollständig ausgeführte Optimierungsverfahren

Bei der Anwendung von *unvollständig ausgeführten Optimierungsverfahren* werden Algorithmen, die das exakte Ergebnis liefern, ausgeführt. Sie werden jedoch nach einer gewissen Rechenzeit vorzeitig abgebrochen. Die bis zu diesem Zeitpunkt beste gefundene Lösung, repräsentiert das Ergebnis. [1]

Beispiel: Branch and Bound.
Implementiertes Verfahren: Minimum Bin Slack Algorithmus.

4.5 Ausblick auf die Lösungsimplementierung

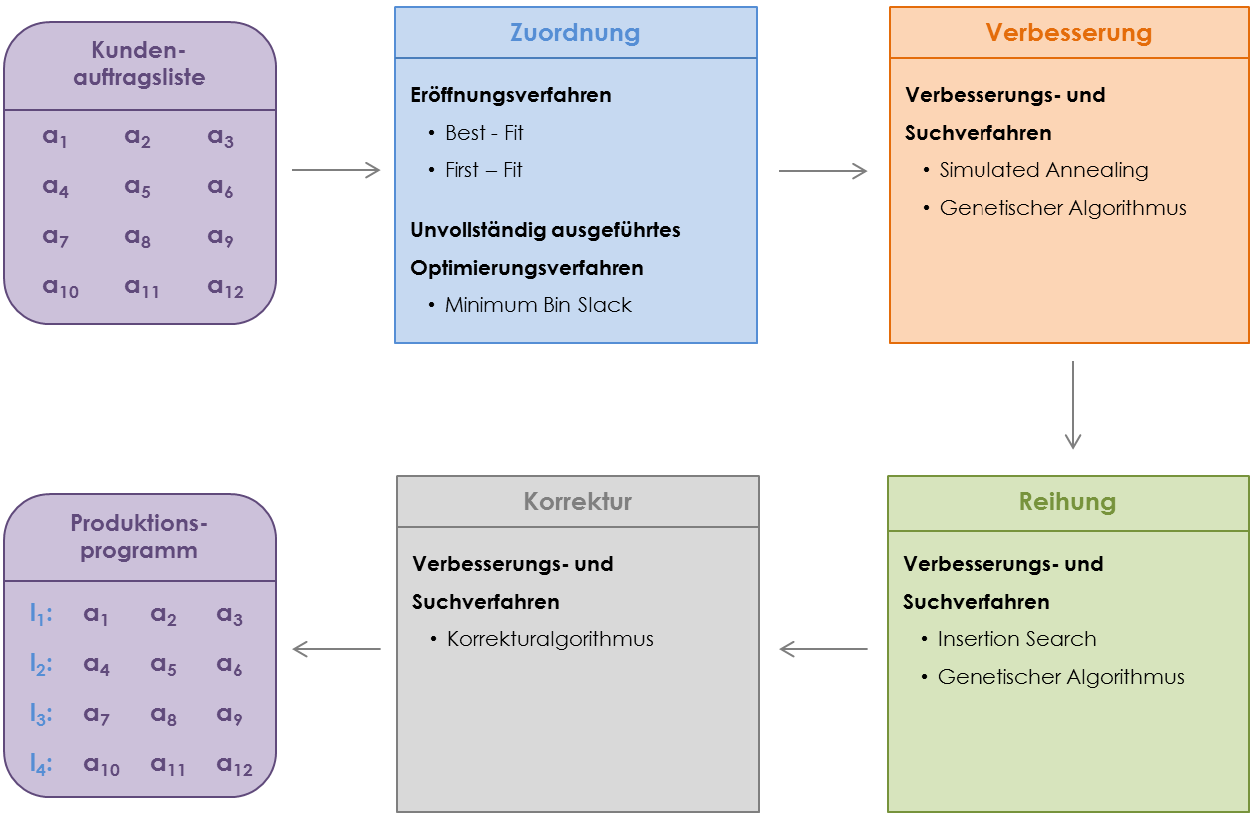


Abbildung 4.1: Diese Grafik veranschaulicht das Zusammenspiel jener Verfahren, die im Zuge dieser Diplomarbeit implementiert, getestet und evaluiert werden.

Kapitel 5

Umsetzung

Im Laufe dieses Abschnittes werden alle wichtigen Begriffe, die in den Kapiteln *Zuordnung, Verbesserung, Reihung und Korrektur* verwendet werden, definiert und erläutert. Des Weiteren wird ausführlich auf die Berechnung der verschiedenen Parameter eingegangen.

5.1 Objekte

Zu Beginn werden die verwendeten Simulationsobjekte kurz vorgestellt.

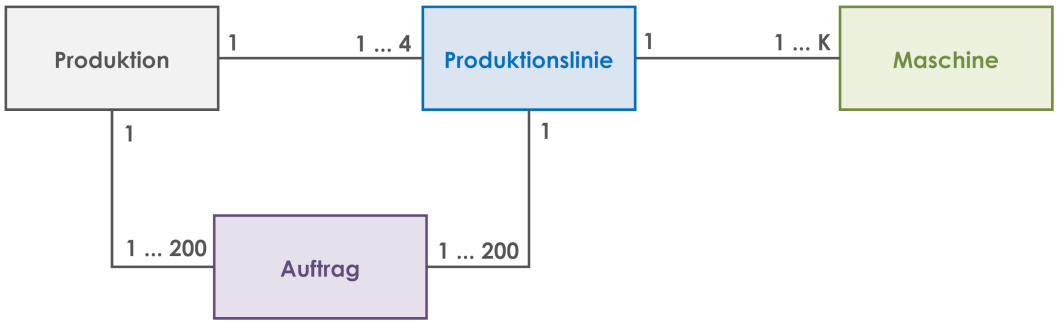


Abbildung 5.1: Diese Grafik zeigt das Klassendiagramm der Simulationsobjekte.

Produktion

Eine *Produktion* P umfasst eine oder mehrere Abteilungen eines Unternehmens, die für die Herstellung und Montage von Halbfertig- und Endprodukten verantwortlich sind. In dieses logistische System wird eine Kundenauftragsliste $A = [a_1, \ldots, a_n \leftrightarrow \{\,\}]$ eingebucht. Die leere Menge $\{\,\}$ bedeutet, dass keiner der Aufträge einer Produktionslinie zugewiesen wurde.

Produktionslinie

Eine Produktion besteht aus einer oder mehreren identischen *Produktionslinien* l_i.

Maschine

Jede Produktionslinie setzt sich aus einer oder mehreren *Maschinen* m_j zusammen. Die Bearbeitung der Aufträge auf den Maschinen erfolgt sequenziell. Das bedeutet, einem Auftrag a_x ist es nicht möglich, einen anderen Auftrag a_y zu überholen. → First In First Out - Prinzip.

Auftrag

Alle *Aufträge* a_x der Kundenauftragsliste $A = [a_1, \ldots, a_n \leftrightarrow \{\,\}]$ werden im *Zuordnungsschritt* auf die Produktionslinien verteilt. Das erstellte Produktionsprogramm $A = [a_1, \ldots, a_n \leftrightarrow \{A_1, \ldots, A_s\}]$ wird mittels *Verbesserungsalgorithmus* modifiziert, wobei A_i der Kundenauftragsliste der Produktionslinie l_i entspricht. Im nächsten Schritt wird die Abarbeitungsreihenfolge der Aufträge auf den einzelnen Produktionslinien angepasst. Zum Abschluss werden entstandene Kapazitätslücken mittels *Korrekturverfahren* eliminiert.

5.2 Definitionen

s ... Anzahl der Produktionslinien l_i, wobei $1 \leq i \leq s$

k ... Anzahl der Maschinen m_j je Produktionslinie, wobei $1 \leq j \leq k$

n ... Gesamt Anzahl der Aufträge a_x der Kundenauftragsliste A, wobei $1 \leq x \leq n$

n_i ... Anzahl der Aufträge a_x die auf der Produktionslinie l_i bearbeitet werden bzw. Anzahl der Aufträge a_x der Kundenauftragsliste A_i

n^* ... Anzahl der Restaufträge

$b_j^{(x)}$ ... Bearbeitungszeit von Auftrag a_x auf Maschine m_j

$r_{jy}^{(x)}$ ... Rüstzeit von Maschine m_j bei aktuellem Auftrag a_x und Vorgängerauftrag a_y

bE_x ... Exakte Bearbeitungszeit von Auftrag a_x $\quad bE_x = \sum_{j=1}^{k} b_j^{(x)}$

rM_x ... Durchschnittliche Rüstzeit von Auftrag a_x $\quad rM_x = \sum_{j=1}^{k} (\frac{1}{n-1} \sum_{\substack{y=1 \\ y \neq x}}^{n} r_{jy}^{(x)})$

rE_x ... Exakte Rüstzeit von Auftrag a_x $\quad rE_x = \sum_{j=1}^{k} r_{jy}^{(x)}$ und $y = \text{fix}$

t_x ... Durchlaufzeit von Auftrag a_x $\quad t_x = bE_x + rM_x$ oder $t_x = bE_x + rE_x$
Bei konkreter Anwendung von t_x wird auf die Berechnungsvariante verwiesen.

A^* ... Absteigend sortierte Kundenauftragsliste A; Kriterium: $t_x = bE_x + rM_x$

T_i ... Durchlaufzeit der Produktionslinie l_i $\quad T_i = \sum_{x=1}^{n_i} t_x$ und $t_x = bE_x + rE_x$

K_i ... Kapazität der Produktionslinie l_i $\quad K_i = \frac{1}{s} \sum_{x=1}^{n} t_x$ und $t_x = bE_x + rM_x$

K_i' ... Restkapazität der Produktionslinie l_i $\quad K_i' = K_i - T_i$

K^* ... Kapazitätserweiterung $\quad K^* = \frac{1}{s} \sum_{x=1}^{n^*} t_x$ und $t_x = bE_x + rM_x$

5.3 Musterbeispiel

Um einen Einblick in die Funktionsweise und den Ablauf der implementierten Algorithmen zu erhalten, wird in den nächsten Kapiteln immer wieder auf das nachfolgende Musterbeispiel zurückgegriffen.

Produktion P, wobei zu Beginn $A = [a_1, \ldots, a_6 \leftrightarrow \{\,\}]$

Maschinen $M = [m_1, m_2]$, $k = 2$

Produktionslinien $L = [l_1, l_2]$, $s = 2$

Kundenauftragsliste $A = [a_1, a_2, a_3, a_4, a_5, a_6]$, $n = 6$

Kapazität $K_1 = K_2 = \frac{1}{2} \sum_{x=1}^{6} t_x = \frac{1}{2} \cdot (11.8 + 14.4 + 18.8 + 13.4 + 20.8 + 6.8) = 43.0$

Auftrag a_x	**Bearbeitungszeit** $tb_x = [b_1^{(x)} \ \cdots \ b_k^{(x)}]$	**Rüstzeit** $tr_x = [r_{11}^{(x)} \ \cdots \ r_{kn}^{(x)}]$
a_1	$tb_1 = [2, 5]$	$tr_1 = r_{1y}^{(1)} = r_{2y}^{(1)} = [0, 2, 5, 1, 0, 4]$
a_2	$tb_2 = [7, 3]$	$tr_2 = r_{1y}^{(2)} = r_{2y}^{(2)} = [1, 0, 4, 2, 1, 3]$
a_3	$tb_3 = [2, 10]$	$tr_3 = r_{1y}^{(3)} = r_{2y}^{(3)} = [1, 5, 0, 5, 5, 1]$
a_4	$tb_4 = [7, 4]$	$tr_4 = r_{1y}^{(4)} = r_{2y}^{(4)} = [0, 0, 0, 0, 4, 2]$
a_5	$tb_5 = [9, 9]$	$tr_5 = r_{1y}^{(5)} = r_{2y}^{(5)} = [0, 2, 3, 1, 0, 1]$
a_6	$tb_6 = [1, 1]$	$tr_6 = r_{1y}^{(6)} = r_{2y}^{(6)} = [0, 2, 5, 2, 3, 0]$

Tabelle 5.1: Kundenauftragsliste - vorgegebene Werte

Mit Hilfe der Daten aus Tabelle 5.1 ist es möglich, die exakte Bearbeitungs- und mittlere Rüstzeit, sowie die durchschnittliche Durchlaufzeit eines Auftrags zu berechnen.

Auftrag a_x	**BearbeitungExakt** bE_x	**RüstMittel** rM_x	**Durchlaufzeit** $t_x = bE_x + rM_x$
a_1	$bE_1 = 7.0$	$rM_1 = 4.8$	$t_1 = 11.8$
a_2	$bE_2 = 10.0$	$rM_2 = 4.4$	$t_2 = 14.4$
a_3	$bE_3 = 12.0$	$rM_3 = 6.8$	$t_3 = 18.8$
a_4	$bE_4 = 11.0$	$rM_4 = 2.4$	$t_4 = 13.4$
a_5	$bE_5 = 18.0$	$rM_5 = 2.8$	$t_5 = 20.8$
a_6	$bE_6 = 2.0$	$rM_6 = 4.8$	$t_6 = 6.8$

Tabelle 5.2: Kundenauftragsliste - berechnete Werte

Abbildung 5.2: Auf dieser Grafik wird die Ausgangssituation der Produktion dargestellt.

Kapitel 6

Zuordnung

Dieser Teil der Arbeit befasst sich mit der *Zuordnung* von Aufträgen zu den verfügbaren Produktionslinien. Die größte Herausforderung dabei ist es, geeignete Algorithmen für das vorliegende Maschinen Schedulingproblem zu finden. Leider hat sich herausgestellt, dass in der Literatur keine spezifischen Methoden für diese Aufgabenstellung dokumentiert sind. Aus diesem Umstand heraus, hat sich die Idee entwickelt, Heuristiken zu verwenden, die das allseits bekannte Bin - Packing[1] Problem lösen. Der Grund dafür liegt auf der Hand. Bin - Packing Algorithmen sind sehr einfache Approximationsverfahren, die leicht und schnell auf andere Problemstellungen adaptiert werden können und dabei gute Ergebnisse liefern.

Außerdem muss berücksichtigt werden, dass es sich bei dieser Aufgabenstellung um ein *Offline - Problem* handelt. Bei einem Offline - Problem ist der Gesamtumfang der Produktion bereits bei der Initialisierung der Lösungsfindung bekannt. Somit ist es nicht möglich während des Simulationsprozesses, Aufträge und Produktionslinien hinzuzufügen oder zu löschen. Aus diesem Grund werden alle Aufträge der Kundenauftragsliste bereits vor der Verteilung absteigend nach ihrer Durchlaufzeit t_x sortiert. Folglich werden Aufträge mit sehr großer Durchlaufzeit frühzeitig berücksichtigt und eingeplant. [10]

Zu guter Letzt wird noch angemerkt, dass die Kapazität K_i mittels durchschnittlicher Rüstzeit rM_x berechnet wird, da zu Beginn keine Information über die Reihenfolge der Auftragsabarbeitung und den jeweiligen Vorgängerauftrag existiert. Außerdem werden während des Zuordnungsschrittes die Durchlaufzeit t_x und damit die Gesamtdurchlaufzeit T_i nur angenähert. Zu diesem Zeitpunkt der Simulation ist es nicht notwendig bei jeder Iteration die exakte Gesamtdurchlaufzeit zu berechnen. Weiters kann somit Rechenzeit eingespart werden. Die Bestimmung des Zielfunktionswertes beruht jedoch auf exakten Berechnungen.

[1] Objekte sollen in eine minimale Anzahl an Behältern gepackt werden.

6.1 Best - Fit

Beschreibung

Beim *Best - Fit* Prinzip wird der Auftrag a_x der Produktionslinie l_i zugeordnet, die noch am wenigsten ausgelastet ist und somit die geringste Gesamtdurchlaufzeit $min\{T_1, \ldots, T_s\}$ aufweist. Die Durchlaufzeit von Auftrag a_x wird folgendermaßen berechnet: $t_x = bE_x + rM_x$. [5]

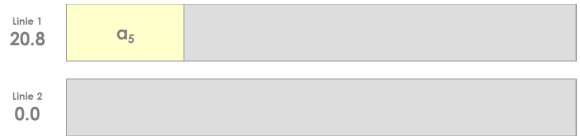
Pseudocode 1 Best - Fit

1: Sortiere die Aufträge der Kundenauftragsliste A so, dass $t_1 \geq t_2 \geq \ldots \geq t_n$
2: **For All** Aufträge a_x
3: $l_{Min} \leftarrow$ Suche die Produktionslinie l_i mit minimaler Belastung
4: Auftrag a_x zu Produktionslinie l_{Min} hinzufügen
5: **EndFor**

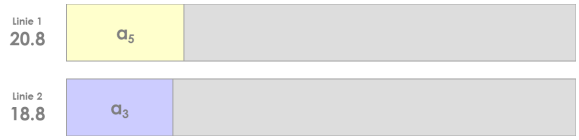
Musterbeispiel

Sortierte Kundenauftragsliste: $A^* = [a_5, a_3, a_2, a_4, a_1, a_6]$

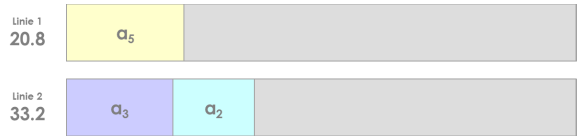
Durchlaufzeiten: 20.8, 18.8, 14.4, 13.4, 11.8, 6.8



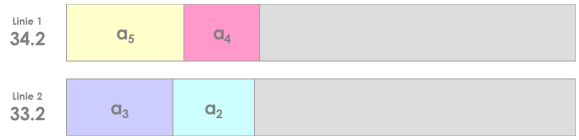
(a) Auftrag a_5 wird Produktionslinie l_1 zugeordnet.



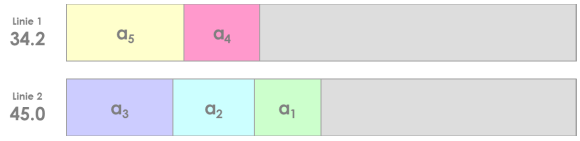
(b) Auftrag a_3 wird Produktionslinie l_2 zugeordnet.



(c) Auftrag a_2 wird Produktionsliniel_2 zugeordnet.



(d) Auftrag a_4 wird Produktionslinie l_1 zugeordnet.



(e) Auftrag a_1 wird Produktionslinie l_2 zugeordnet.

(f) Auftrag a_6 wird Produktionslinie l_1 zugeordnet.

Ergebnis $T = max\{T_1, T_2\} = \{43.0, 41.0\} = \mathbf{43.0}$, wobei $T_i = \sum_{x=1}^{n_i} t_x$ und $t_x = bE_x + rE_x$

Abbildung 6.1: Anhand dieser Grafik wird der Ablauf des Best - Fit Algorithmus dargestellt.

6.2 First - Fit

Beschreibung

Beim *First - Fit* Prinzip wird der Auftrag a_x der ersten Produktionslinie l_i zugeteilt, die noch genügend Restkapazität K_i' zur Verfügung hat. Die Durchlaufzeit von Auftrag a_x wird folgendermaßen berechnet: $t_x = bE_x + rM_x$. [10]

Es kann jedoch der Fall eintreten, dass die Durchlaufzeit $t_x > K_i'$, für alle $i = \{1, \ldots, s\}$. Das bedeutet, dass der Auftrag a_x keiner Produktionslinie l_i zugeordnet werden kann. Aus diesem Grund wird die beschriebene Standardvorgehensweise um folgende Besonderheit ergänzt. Alle Aufträge, die keiner Produktionslinie zugeteilt werden können, werden gespeichert. Die Kapazität K_i der Produktionslinien wird mit $K_i = K_i' + K^* = K_i' + \frac{1}{s}\sum_{x=1}^{n^*} t_x$ erweitert, wobei n^* der Anzahl der Restaufträge entspricht. Anschließend wird der Ablauf erneut gestartet, mit dem Ziel, alle Restaufträge in das Produktionssystem einzugliedern.

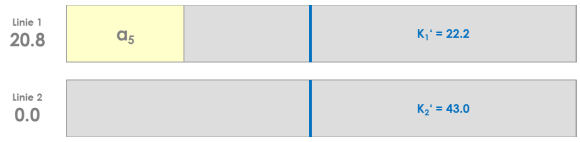
Pseudocode 2 First - Fit

```
1: Sortiere die Aufträge der Kundenauftragsliste A so, dass t_1 ≥ t_2 ≥ ... ≥ t_n
2: If Kundenauftragsliste A leer Then
3:    return
4: Else
5:    For All Aufträge a_x
6:       For All Produktionslinien l_i
7:          If t_x ≤ K_i' Then
8:             Auftrag a_x zu Produktionslinie l_i hinzufügen und x++
9:          EndIf
10:      EndFor
11:      If Auftrag a_x kann keiner Produktionslinie l_i zugeordnet werden Then
12:         Auftrag a_x zur Liste der Restaufträge hinzufügen
13:      EndIf
14:   EndFor
15: EndIf
16: Rekursiver Aufruf (Kundenauftragsliste ← Restaufträge) ↷ Schritt 2
```

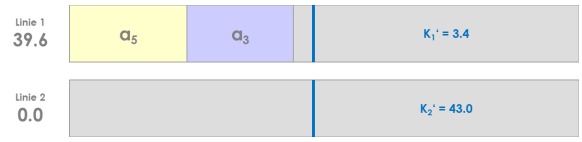
Musterbeispiel

Sortierte Kundenauftragsliste: $A^* = [a_5, a_3, a_2, a_4, a_1, a_6]$

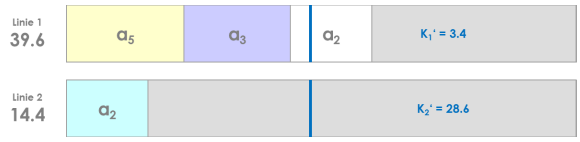
Durchlaufzeiten: 20.8, 18.8, 14.4, 13.4, 11.8, 6.8



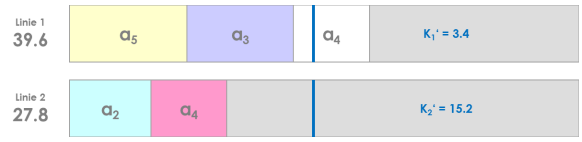
(a) Auftrag a_5 wird Produktionslinie l_1 zugeordnet.



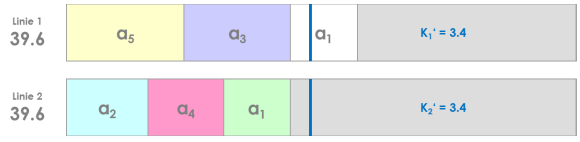
(b) Auftrag a_3 wird Produktionslinie l_1 zugeordnet.



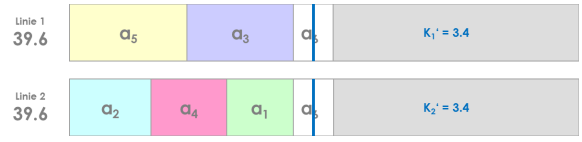
(c) Auftrag a_2 wird Produktionslinie l_2 zugeordnet.



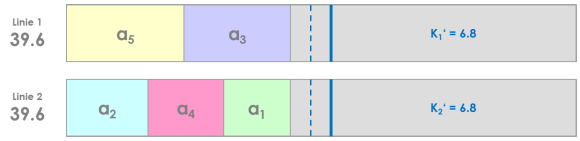
(d) Auftrag a_4 wird Produktionslinie l_2 zugeordnet.



(e) Auftrag a_1 wird Produktionslinie l_2 zugeordnet.



(f) Auftrag a_6 kann weder Produktionslinie l_1 noch Produktionslinie l_2 zugeordnet werden.



(g) Die Kapazität der beiden Produktionslinien wird erweitert. $K^* = 6.8 \cdot \frac{1}{2} = 3.4$

(h) Auftrag a_6 wird Produktionslinie l_1 zugeordnet.

Ergebnis $T = max\{T_1, T_2\} = \{52.0, 30.0\} = \mathbf{52.0}$, wobei $T_i = \sum_{x=1}^{n_i} t_x$ und $t_x = bE_x + rE_x$

Abbildung 6.2: Anhand dieser Grafik wird der Ablauf des First - Fit Algorithmus dargestellt.

6.3 Minimum Bin Slack

Beschreibung

Das Prinzip des *Minimum Bin Slack* Algorithmus ist einfach. Es soll jene Auftragsverteilung gefunden werden, welche die zur Verfügung stehende Restkapazität K_i' der Produktionslinie l_i in weiterer Folge auch *Slack - Packing* genannt, minimiert. Die Durchlaufzeit von Auftrag a_x wird folgendermaßen berechnet: $t_x = bE_x + rM_x$.

Die Produktionslinien werden unabhängig voneinander, der Reihe nach betrachtet. Zuerst wird die sortierte Kundenauftragsliste A^* durchlaufen. Dabei werden alle Aufträge a_x für die $t_x \leq K_i'$ gilt, der Produktionslinie l_i zur Abarbeitung zugewiesen. Ist das gesamte Kapazitätskontingent ausgeschöpft, wird der letzte hinzugefügte Auftrag wieder aus dem Produktionsprogramm A_i gelöscht. Von diesem Auftrag ausgehend wird nach neuen Aufträgen gesucht, die das verbleibende Slack - Packing weiter minimieren und somit das gewünschte Ergebnis verbessern. Dieser Ablauf wird solange fortgesetzt, bis alle Auftragskombinationen getestet worden sind.

Wie bereits mehrmals erwähnt wurde, handelt es sich bei der Laufzeit um einen kritischen Faktor. Folglich müssen einige Abbruchkriterien und Sonderbedingungen definiert werden, die gewährleisten, dass der Minimum Bin Slack Algorithmus trotz beschränkter Laufzeit, die hier mit 3 Sekunden angenommen wird, brauchbare Ergebnisse liefert. Alle Aufträge die nicht mit der Minimum Bin Slack Methode verteilt werden können, werden mittels Best - Fit in das bestehende Produktionsprogramm eingegliedert. [7]

Kapazitätserweiterung

Mit Hilfe des Minimum Bin Slack Algorithmus wird eine Minimierung der Restkapazität $K_i' = K_i - T_i$ erzielt. Dieser Slack - Packing Wert lässt sich aus der Kapazität K_i und der Gesamtdurchlaufzeit T_i der Produktionslinie l_i errechnen. Wie bereits in der Einleitung dieses Kapitels erwähnt wurde, wird die Kapazität nur ungefähr mittels mittlerer Rüstzeit angenähert und liefert somit nur einen groben, aber sehr wichtigen Richtwert. Die Idee ist, durch schrittweise Vergrößerung der Kapazität eine bessere Approximation dieses Wertes zu erhalten. Nach mehreren Tests hat sich herauskristallisiert, dass diese Ressourcenerweiterung in einigen Fällen ein bedeutend besseres Produktionsprogramm liefert. Die Umsetzung dieser Weiterentwicklung ist in Schritt 2 des Pseudocodes ersichtlich.

Rechenzeitüberprüfung

Da die Rechenzeit eine bedeutende Rolle spielt, wird vor jedem Rekursionsschritt überprüft, ob die Laufzeitgrenze von 3 Sekunden noch nicht überschritten wurde. Wird diese Bedingung verletzt, wird der Algorithmus sofort abgebrochen und das beste bisher gefundene Produktionsprogramm A_i wird zurückgegeben. Zeile 12 des Pseudocodes spiegelt dies wider.

Slack - Packing Abbruch

Der aktuelle Zuordnungsschritt wird frühzeitig beendet, wenn eine sehr gute Lösung für die Produktionslinie l_i gefunden wurde. Ein Produktionsprogramm wird als sehr gut eingestuft, wenn die Bearbeitung aller Aufträge der Produktionslinie l_i 99% der errechneten Kapazität K_i in Anspruch nimmt. Diese Bedingung wird in Schritt 20 des Pseudocodes sichtbar.

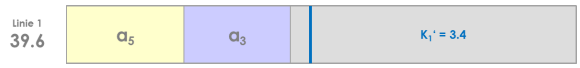
Pseudocode 3 Minimum Bin Slack

```
1: Sortiere die Aufträge der Kundenauftragsliste A so, dass t_1 ≥ t_2 ≥ ... ≥ t_n
2: For c = 1.0 → 1.1
3:    c ← c + 0.001
4:    For All Produktionslinien l_i
5:       index ← 0
6:       If Ende der Kundenauftragsliste A erreicht Then
7:          return
8:       Else
9:          For All Aufträge a_x, wobei x ← index
10:            If t_x ≤ K'_i Then
11:               Auftrag a_x zu Produktionslinie l_i hinzufügen
12:               If Laufzeit ≥ 3 Sekunden Then
13:                  return ▶ Bestes bisher gefundenes Produktionsprogramm A_i
14:               Else
15:                  Rekursiver Aufruf (index++) ↷ Schritt 6
16:                  If Slack - Packing der neuen Auftragskombination < K'_i Then
17:                     A_i ← neue Auftragskombination; K'_i ← Slack - Packing
18:                  EndIf
19:                  If K'_i ≤ K_i · 0.01 Then
20:                     return ▶ Produktionsprogramm A_i
21:                  EndIf
22:                  Auftrag a_x von Produktionslinie l_i löschen
23:               EndIf
24:            EndIf
25:         EndFor
26:      EndIf
27:   EndFor
28:   If Aufträge konnten nicht alle verteilt werden Then
29:      Best - Fit Algorithmus (Restaufträge)
30:   EndIf
31:   If T < T_Best Then
32:      A_Best ← A; T_Best ← T
33:   EndIf
34: EndFor
```

Musterbeispiel

Sortierte Kundenauftragsliste: $A^* = \{a_5, a_3, a_2, a_4, a_1, a_6\}$

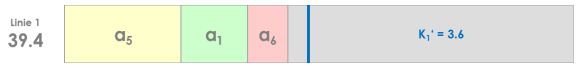
Durchlaufzeiten: 20.8, 18.8, 14.4, 13.4, 11.8, 6.8



(a) Auftrag a_5 und Auftrag a_3 werden zur Produktionslinie l_1 hinzugefügt. Für alle weiteren Aufträge a_x gilt: $t_x > K_1'$

(b) Auftrag a_3 wird gelöscht. Auftrag a_2 und Auftrag a_6 werden hinzugefügt. Für alle restlichen Aufträge a_x gilt: $t_x > K_1'$

(c) Auftrag a_6 wird gelöscht. Auftrag a_2 wird ebenfalls gelöscht. Der nächste Auftrag nach Auftrag a_2 ist Auftrag a_4. Auftrag a_4 und Auftrag a_6 werden hinzugefügt.



(d) Auftrag a_6 wird gelöscht. Auftrag a_4 wird ebenfalls gelöscht. Der nächste Auftrag nach Auftrag a_4 ist Auftrag a_1. Auftrag a_1 und Auftrag a_6 werden hinzugefügt.

(e) Auftrag a_6 wird gelöscht. Auftrag a_1 wird ebenfalls gelöscht. Der nächste Auftrag nach Auftrag a_1 ist Auftrag a_6. Auftrag a_6 wird hinzugefügt.

(f) Auftrag a_6 wird gelöscht. Auftrag a_5 wird ebenfalls gelöscht. Der nächste Auftrag nach Auftrag a_5 ist Auftrag a_3. Auftrag a_3 wird hinzugefügt. ...

Der Ablauf wird wie beschrieben fortgesetzt. Das Produktionsprogramm A_1 für Produktionslinie l_1 ist $A_1 = \{a_5, a_2, a_6\}$. Die übrigen Aufträge werden nach dem gleichen Schema auf Produktionslinie l_2 verteilt. Das Resultat ist $A_2 = \{a_3, a_4\}$. Zu guter Letzt wird der Restauftrag a_1 mittels Best - Fit Methode, ebenfalls Produktionslinie l_2 zugeordnet.

(g) Diese letzte Grafik bildet das Gesamtergebnis und somit das endgültige Produktionsprogramm ab.

Ergebnis $T = max\{T_1, T_2\} = \{36.0, 32.0\} = \mathbf{36.0}$, wobei $T_i = \sum_{x=1}^{n_i} t_x$ und $t_x = bE_x + rE_x$

Abbildung 6.3: Anhand dieser Grafik wird der Ablauf des Minimum Bin Slack Algorithmus dargestellt.

Kapitel 7

Verbesserung

Der nächste Schritt auf dem Weg zu einem sehr guten Produktionsprogramm ist die *Verbesserung* der zuvor erstellten Auftragsverteilung. Zu diesem Zweck werden *Verbesserungs-* und *Suchverfahren* eingesetzt. Dieses Kapitel wird sich ausführlich mit dieser Thematik befassen.

Die gängigsten Verbesserungs- und Suchverfahren, die immer wieder in der Fachliteratur erwähnt werden, sind Simulated Annealing, Threshold Accepting, Tabu Search und genetische Algorithmen. Sie basieren alle auf dem gleichen Prinzip. Von einer oder mehreren Startlösungen ausgehend, wird nach verschiedenen Prinzipien der Lösungsraum untersucht. Das Unterscheidungsmerkmal dieser Methoden liegt auch in der Auswahl und Akzeptanz der neu generierten Produktionsprogramme. Auf Simulated Annealing, sowie genetische Algorithmen wird im weiteren Verlauf dieses Kapitels im Detail eingegangen.

Zu guter Letzt wird noch angemerkt, dass die Durchlaufzeit t_x und damit die Gesamtdurchlaufzeit T_i exakt berechnet werden. Dies ist möglich, da die verwendete Ausgangslösung ein Produktionsprogramm und Informationen über die Auftragsreihenfolge bereitstellt.

Threshold Accepting

Bei *Threshold Accepting* handelt es sich um eine abgewandelte Form von Simulated Annealing. Bei diesem Verfahren werden schlechtere Lösungen nur dann akzeptiert, wenn die Änderung des Zielfunktionswerts ($T_{P_g} - T_{Best}$) eine gewisse Schwelle nicht unterschreitet. P_g repräsentiert dabei die aktuelle Produktion. Während der Simulation wird dieser Grenzwert kontinuierlich auf null reduziert. Bei Threshold Accepting besteht, wie auch bei anderen Verfahren, die Gefahr in einem lokalen Optimum hängen zu bleiben. [4]

Tabu Search

Tabu Search ist ein Verfahren, welches nach jedem Iterationsschritt die Nachbarschaftslösung mit dem besten Zielfunktionswert $min\{T_{P_1}, \ldots, T_{P_z}\}$ auswählt. Dabei können auch Produktionsprogramme selektiert werden, die eine größere Gesamtdurchlaufzeit, als die beste bisher gefundene Auftragsverteilung, aufweisen. Um nach der Akzeptanz einer Verschlechterung nicht wieder zu einer besseren, bereits gefunden Lösung zurückzukehren, werden diese Lösungen immer *tabu* gesetzt. Die Speicherung und Überprüfung der Tabus ist sehr aufwendig. Auch in diesem Fall besteht die Gefahr, in einem lokalen Optimum hängen zu bleiben. [1]

Abschließend soll noch kurz darauf eingegangen werden, wieso weder Threshold Accepting noch Tabu Search implementiert wurden. Bei der Anwendung von Simulated Annealing ist es auf Grund der wahrscheinlichkeitsbedingten Lösungsakzeptanz leichter, ein gefundenes lokales Optimum wieder zu verlassen und somit einen größeren Lösungsraum zu durchforsten. Genetische Algorithmen wiederum haben den Vorteil, dass während des gesamten Ablaufs immer mehrere Lösungen parallel verfügbar sind. Auch dieser Umstand hat zur Folge, dass eine breitere Nachbarschaftsumgebung abgesucht werden kann.

7.1 Simulated Annealing

Beschreibung

Simulated Annealing versucht, ausgehend von einem bereits bestehenden Produktionsprogramm, durch kontinuierliches Verändern der Auftragsverteilung eine bessere Lösung zu finden. Dabei wird der Abkühlungsprozess eines Metalls als Grundprinzip verwendet. [15]

Initialisierung

Zu Beginn muss eine Ausgangslösung generiert werden. Bei dieser konkreten Anwendung wird ein erstes gültiges Produktionsprogramm mittels Zuordnungsalgorithmus oder zufällig erzeugt. Wie bereits bekannt ist, kann diese Zuordnung mittels Best - Fit, First - Fit oder Minimum Bin Slack Algorithmus erfolgen. [15]

Lösungsevaluierung

Anschließend wird mit der Nachbarschaftssuche begonnen. Zu diesem Zweck werden zwei unterschiedliche Methoden (Min - Max- und Random Nachbarschaft) verwendet, die im Anschluss erläutert werden. Nach jedem ausgeführten Iterationsschritt erfolgt eine Evaluierung

des neu erstellten Produktionsprogrammes. Dabei werden zwei Fälle unterschieden. Eine bessere Lösung wird immer sofort akzeptiert, eine schlechtere hingegen wird mit einer gewissen Wahrscheinlichkeit bP angenommen. [10]

Mit Hilfe der Boltzmann Wahrscheinlichkeit $bP = e^{-\Delta E/T}$ wird die Akzeptanz suboptimaler Lösungen gesteuert. Außerdem wird durch ihre Verwendung, die Chance, ein lokales Optimum wieder zu verlassen, erhöht. Des Weiteren kann auf Basis der Wahrscheinlichkeitsberechnung der erwähnte Abkühlungsprozess im Detail analysiert werden. Nach der Erhitzung eines Metalls bewegen sich Moleküle frei und mit sehr großer Geschwindigkeit. Während der kontinuierlichen Abkühlung wird die Geschwindigkeit und somit die innere Energie stetig reduziert. Dabei haben die Moleküle genügend Zeit stabile Kristalle zu bilden und den optimalen Energiezustand anzunehmen. [19]

Umgelegt auf die Implementierung des Algorithmus bedeutet dies folgendes: Zu Beginn der Lösungsraumerforschung ($T_{SA} = 10000$) werden schlechtere Ergebnisse mit sehr großer Wahrscheinlichkeit akzeptiert. Damit ist es möglich, einen Großteil der Nachbarschaften zu testen. Im Laufe der Untersuchung wird die Temperatur kontinuierlich mit dem Faktor $\alpha = 0.99$ multipliziert ($T_{SA} \leftarrow T_{SA} \cdot \alpha$). Je näher das Ende des Abkühlungsprozesses rückt, desto starrer werden die Lösungen. Die Werte der Parameter wurden mit Hilfe von Testläufen und unter Einbeziehung der Laufzeitbeschränkung gesetzt. [15]

Nachbarschaftssuche

Bevor im Detail auf die beiden Nachbarschaftssuchverfahren eingegangen wird, soll noch erwähnt werden, dass diese beiden Methoden nicht in Anlehnung an einen Artikel oder ein Buch entwickelt wurden. Dies wäre gar nicht möglich gewesen, da der Begriff Nachbarschaftssuche in der Literatur immer nur erwähnt, verwendete Methoden, im Zusammenhang mit dem hier betrachteten Problem, aber nie näher beschrieben werden.

Min - Max Nachbarschaft

Bei der *Min - Max* Nachbarschaftssuche werden zwei Aufträge nach folgendem Prinzip vertauscht. Zuerst werden die beiden Produktionslinien l_{Min} mit minimaler und l_{Max} mit maximaler Gesamtdurchlaufzeit bestimmt. Anschließend werden alle Aufträge der beiden Kundenauftragslisten A_{Min}, A_{Max} absteigend nach der Durchlaufzeit ($t_x = bE_x + rM_x$) sortiert. Von Produktionslinie l_{Min} wird nun ein Auftrag a_x ausgewählt, der sich in der hinteren Hälfte der temporären Kundenauftragsliste befindet und somit eine kurze Durchlaufzeit aufweist. Von Produktionslinie l_{Max} wird ein Auftrag a_y ausgewählt, der sich in der vorderen Hälfte der temporären Kundenauftragsliste befindet und somit eine lange Durchlaufzeit hat. Dann

erfolgt die Vertauschung der Aufträge a_x und a_y.

Random Nachbarschaft

Bei der *Random* Nachbarschaftssuche werden zwei Produktionslinien l_I, l_{II} und zwei Aufträge a_x, a_y zufällig bestimmt und vertauscht. Dabei müssen folgende Fälle unterschieden werden. Sind die beiden ausgewählten Produktionslinien identisch ($l_I = l_{II}$), so werden zufällig zwei Aufträge der selben Produktionslinie bestimmt und vertauscht. Weiters kann es auch möglich sein, dass zwei unterschiedliche Produktionslinien ($l_I \neq l_{II}$) Gegenstand der Betrachtung werden. In diesem Fall werden zwei Aufträge verschiedener Produktionslinien ausgewählt. Bei der Vertauschung ändern Auftrag a_x und a_y ihre Position und ihre Produktionslinie. Außerdem besteht die Möglichkeit, dass nur ein Auftrag a_x von Produktionslinie l_I selektiert wird. Dann wird dieser Auftrag am Ende der Kundenauftragsliste von Produktionslinie l_{II} hinzugefügt.

Pseudocode 4 Simulated Annealing

```
1: While T_SA ≥ T_Ende
2:   counter ← 0
3:   While counter ≤ 10
4:     Generiere ein neues Produktionsprogramm
5:     If T der neuen Auftragskombination < T Then
6:       A ← neue Auftragsverteilung; T ← max{T_1, ..., T_s}
7:       If T < T_Best Then
8:         A_Best ← A; T_Best ← T
9:       EndIf
10:    Else
11:      zP ← Generiere eine Zufallszahl [0.0, 1.0[
12:      bP ← Berechne die Boltzmann Wahrscheinlichkeit
13:      If bP ≥ zP Then
14:        A ← neue Auftragsverteilung; T ← max{T_1, ..., T_s}
15:        counter ← ∞
16:      Else
17:        counter++
18:      EndIf
19:    EndIf
20:  EndWhile
21:  T_SA ← T_SA · α
22: EndWhile
```

Musterbeispiel

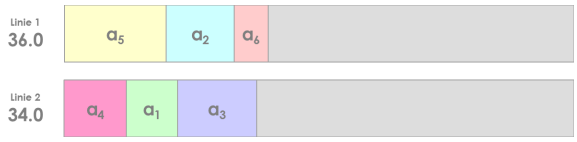
Das verwendete Produktionsprogramm wurde zufällig erstellt.

Auftragsverteilung $A_1 = [a_5, a_3, a_2] \rightarrow T_1 = 58.0$

Auftragsverteilung $A_2 = [a_4, a_1, a_6] \rightarrow T_2 = 22.0$

Min - Max Nachbarschaft ($T_{SA} = 10000$)
$l_{Min} \leftarrow l_2$: Auftrag a_6 wird zufällig ausgewählt.
$l_{Max} \leftarrow l_1$: Auftrag a_3 wird zufällig ausgewählt.
Die beiden Aufträge werden vertauscht.

$max\{36.0, 34.0\} \leq T = 58.0$
Produktionsprogramm wird angenommen!

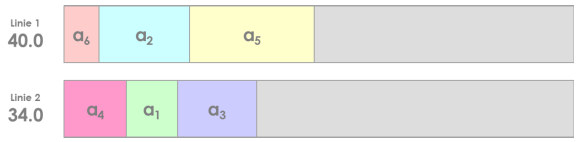


(a) Auftrag a_6 wird zu Produktionslinie l_1 hinzugefügt. Auftrag a_3 wird zu Produktionslinie l_2 hinzugefügt.

Random Nachbarschaft ($T_{SA} = 9900$)
1. Fall: $l_I = l_{II} \leftarrow l_1$
Auftrag a_5 und a_6 werden zufällig ausgewählt.
Die beiden Aufträge werden vertauscht.

$max\{40.0, 34.0\} \not\leq T = 36.0$

$bP = e^{-(40.0-36.0)/(9.900)} = 0.9995 \geq zP = 0.41$
Produktionsprogramm wird angenommen!

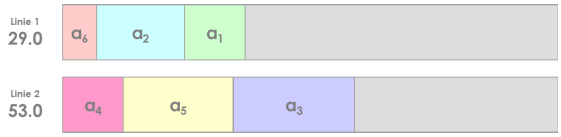


(b) Auftrag a_5 und a_6 werden vertauscht.

Random Nachbarschaft ($T_{SA} = 9801$)
2. Fall: $l_I \neq l_{II} \leftarrow l_1, l_2$
l_1: Auftrag a_5 wird zufällig ausgewählt.
l_2: Auftrag a_1 wird zufällig ausgewählt.
Die beiden Aufträge werden vertauscht.

$max\{29.0, 53.0\} \not\leq T = 40.0$

$bP = e^{-(53.0-40.0)/(9.801)} = 0.9987 \not\geq zP = 0.999$
Produktionsprogramm wird nicht angenommen!

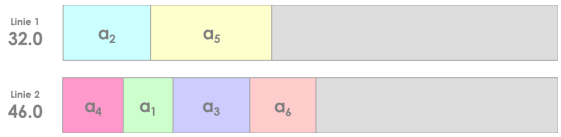


(c) Auftrag a_1 wird zu Produktionslinie l_1 hinzugefügt. Auftrag a_5 wird zu Produktionslinie l_2 hinzugefügt.

Random Nachbarschaft ($T_{SA} = 9703$)
3. Fall: $l_I \neq l_{II} \leftarrow l_1, l_2$
l_1: Auftrag a_6 wird zufällig ausgewählt.
l_2: Kein Auftrag wird ausgewählt.
Auftrag a_6 wird hinzugefügt.

$max\{32.0, 46.0\} \not\leq T = 40.0$

$bP = e^{-(46.0-40.0)/(9.703)} = 0.9994 \not\geq zP = 0.12$
Produktionsprogramm wird angenommen!



(d) Auftrag a_6 wird zu Produktionslinie l_2 hinzugefügt. Dieser Ablauf würde fortgesetzt werden bis $T_{SA} \leq T_{Ende} = 1$ wäre.

Abbildung 7.1: Anhand dieser Grafik werden der Ablauf der Simulated Annealing Methode und der beiden Nachbarschaftssuchverfahren dargestellt.

7.2 Genetische Algorithmen

Beschreibung

Genetische Algorithmen basieren auf einem ähnlichen Prinzip wie Simulated Annealing. Von mehreren bestehenden Produktionsprogrammen ausgehend, wird versucht, durch kontinuierliches Verändern der Auftragsverteilung, bessere Lösungen zu finden. Die Grundidee bilden dabei die biologische Evolution und die Theorien rund um Genetik und Darwinismus („Survival of the Fittest“). Bahnbrechende Erkenntnisse auf diesem Gebiet sind vor allem John Holland zu verdanken, der als Begründer der genetischen Algorithmen gilt. [6]

Zuerst sollen kurz die biologischen Grundlagen näher erläutert werden. Bei genetischen Algorithmen steht die Vererbung und somit der Zellkern im Mittelpunkt der Betrachtung. Das Kernplasma besteht aus *Chromosomen*. Diese Chromosomen enthalten die *Gene* und werden als Träger der Erbinformation bezeichnet. In weiterer Folge werden diese Chromosomen auch *Individuen* genannt. Unterschiedliche Gene definieren verschiedene Eigenschaften eines Lebewesens. Die konkrete Ausprägung eines Merkmals heißt *Allel*. Weiters werden zwei Zellteilungen unterschieden, mit Hilfe derer sich Chromosomen vermehren können. Die erste Möglichkeit ist die ungeschlechtliche Mitose. Bei dieser Form der Zellteilung entsteht eine Tochterzelle, die das selbe Erbgut wie ihre Mutterzelle besitzt. Mitose setzt bei der Regeneration von Zellen nach einer Verletzung ein. Die zweite Variante bildet die geschlechtliche Meiose. In diesem Fall erfolgt durch die Kreuzung zweier Zellen eine Vermischung des Erbguts. Auf das Prinzip der Meiose aufbauend, kann mit der technischen Ausführung fortgefahren werden. [14]

Vorher soll noch kurz darauf verwiesen werden, dass genetische Algorithmen sehr viel Interpretationsspielraum und Variantenvielfalt vereinen. Eigentlich würde die Untersuchung der Anwendbarkeit von genetischen Algorithmen für das mehrere Maschinen Schedulingproblem eine eigene Diplomarbeit darstellen. Um die Relation zu den anderen Kapiteln nicht zu verlieren, wird nicht im Detail auf alle möglichen Formen, sondern nur auf eine kleine Anzahl davon eingegangen. Die Auswahl fiel dabei auf unkomplizierte Methoden, die sehr schnell und effizient implementiert werden können.

Initialisierung

Zu Beginn des Verfahrens müssen mehrere Ausgangslösungen, sogenannte Individuen, generiert werden. Ein Individuum repräsentiert dabei eine Kundenauftragsverteilung $A_g = [a_1, \ldots, a_n \leftrightarrow \{A_1, \ldots, A_s\}]$ der Produktion P_g. Bei dieser konkreten Anwendung werden In-

dividuen mittels Zuordnungsalgorithmus oder zufällig erzeugt. Wie bereits bekannt ist, kann diese Zuordnung mittels Best - Fit, First - Fit oder Minimum Bin Slack erfolgen. Alle erstellten Produktionsprogramme können unter dem Begriff *Ausgangspopulation* zusammengefasst werden. Die Größe der Population bzw. die Anzahl der Individuen aI ist problemabhängig.

Fitnessfunktion

Mit Hilfe der *Fitnessfunktion* wird die Qualität der Merkmalausprägungen eines Individuums messbar gemacht.

Die Fitness $f(P_g)$ eines Individuums entspricht in dieser Arbeit der maximalen Gesamtdurchlaufzeit $T_g = \max\{T_1, \ldots, T_s\}$ der Produktion P_g. Da es sich um ein Minimierungsproblem handelt, muss die Fitnessfunktion folgendermaßen berechnet werden $g(P_g) = C - f(P_g)$, wobei C eine Konstante ist. Wäre die Fitnessfunktion äquivalent zur definierten Fitness $f(P_g)$, würde eine Produktion mit einer großen Gesamtdurchlaufzeit, bevorzugt vererbt werden. Das Resultat wäre in diesem Fall eine Maximierung der Zielfunktion. Weiters ist zu sagen, dass die Wahl der Konstante $C = 1.1 \cdot c$ ein sensibles Unterfangen ist. In dieser Arbeit entspricht c der größten Gesamtdurchlaufzeit aller Individuen der Ausgangspopulation. Diese Konstante c wird mit einem Faktor multipliziert. Damit soll verhindert werden, dass die Fitness einen Wert ≤ 0 annimmt.

In weiterer Folge kann die relative Fitness $r(P_g)$ eines Individuums berechnet werden.

$$r(P_g) = \frac{g(P_g)}{\sum_{g=1}^{z} g(P_g)}, \text{ wobei } \quad 0 \leq r(P_g) \leq 1 \quad \text{und} \quad \sum_{g=1}^{z} r(P_g) = 1$$

Selektion

Der nächste Schritt ist die *Selektion*. Dabei werden Individuen aus der zuvor erstellten Ausgangspopulation ausgewählt. Alle selektierten Produktionsprogramme P_g werden Teil des *Paarungspools* $= \{P_1, \ldots, P_z\}$, wobei $(z = aI - 3)$. Nach Abschluss der Selektion wird der Paarungspool um das beste und die beiden schlechtesten Individuen der aktuellen Population erweitert.

Da diese Beschreibung sehr allgemein ist, gibt es genügend Möglichkeiten für die tatsächliche Ausformulierung des Algorithmus. Auf jeden Fall ist zu beachten, dass der Paarungspool eine ausgewogene Kombination aus guten bisher gefundenen Lösungen, sowie suboptimalen Ergebnissen ist. Anderenfalls läuft der genetische Algorithmus Gefahr, in einem lokalen Optimum hängenzubleiben. Aus einer Fülle an Selektionsmethoden wurden die beiden nachfol-

genden ausgewählt. Der Vollständigkeit halber sollen die anderen Verfahren erwähnt werden: Heirat - Selektion, Lineares Ranking, Exponentielle Selektion, Uniforme Selektion.

Fitnessproportionale Selektion

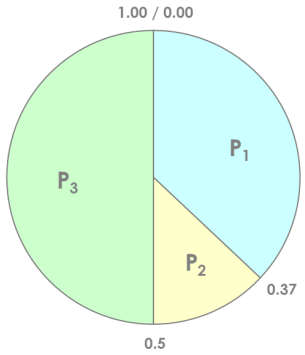
Bei der *fitnessproportionalen* oder auch *Roulette - Wheel* Selektion werden Individuen mit einer gewissen Wahrscheinlichkeit, die proportional zu ihrem relativen Fitnesswert $r(P_g)$ ist, ausgewählt. Damit soll erreicht werden, dass ein gutes Produktionsprogramm mit einer höheren Wahrscheinlichkeit vererbt und somit Teil des Paarungspools wird. Veranschaulicht wird dieses Verfahren mit einer Rouletteschеibe. Diese Scheibe wird in verschieden große Sektoren eingeteilt, wobei ein Sektor einem Individuum und die Größe der relativen Fitness entspricht. Mit Hilfe einer Zufallszahl $zZ \in [0.0, 1.0[$ wird anschließend ein Abschnitt bestimmt und das entsprechende Individuum ausgewählt. [14]

Musterbeispiel

Ausgangspopulation mit Fitness $f(P_g) = [43.0, 52.0, 38.0]$

Ausgangspopulation mit Fitness $g(P_g) = [14.2, 5.2, 19.2]$ mit $C = 52.0 \cdot 1.1 = 57.2$

$r(P_1) = \frac{14.2}{(14.2+5.2+19.2)} = 0.37 \quad r(P_2) = \frac{5.2}{38.6} = 0.13 \quad r(P_3) = \frac{19.2}{38.6} = 0.50$



Zufallszahl $zZ = 0.41$ wird generiert!
Produktion P_1 deckt das Intervall $[0.00, 0.37]$ ab.
Die Zufallszahl liegt nicht im angegeben Intervall.
Produktion P_2 deckt das Intervall $]0.37, 0.50]$ ab.
Die Zufallszahl liegt im angegebenen Intervall.
Produktion P_2 wird zum Paarungspool hinzugefügt.

Abbildung 7.2: Diese Grafik zeigt das Prinzip der fitnessproportionalen Selektion.

Turnierauswahl

Bei der *Turnierauswahl* werden zufällig z Individuen ausgewählt. Jenes Individuum mit der höchsten Fitness $max\{g(P_1), ..., g(P_z)\}$ wird in den Paarungspool aufgenommen. [14]

Musterbeispiel

Ausgangspopulation mit Fitness $f(P_g) = [43.0, 52.0, 38.0]$

Ausgangspopulation mit Fitness $g(P_g) = [14.2, 5.2, 19.2]$ mit $C = 52.0 \cdot 1.1 = 57.2$

Produktion P_1 mit $g(P_1) = 14.2$ und Produktion P_2 mit $g(P_2) = 5.2$ werden zufällig ausgewählt. Produktion P_1 wird zum Paarungspool hinzugefügt.

Kreuzung

Die nächste Stufe ist die *Kreuzung*[1] von Individuen. Bei diesem Rekombinationsschritt werden zufällig zwei Elternindividuen, in diesem Fall Produktionen, aus dem Paarungspool ausgewählt. Im Anschluss daran erfolgt die Fortpflanzung (Meiose). Das Resultat sind Kinderindividuen, die alle unter dem Begriff *Nachkommen* zusammengefasst werden.

Da der Begriff Fortpflanzung sehr allgemein gehalten ist, können mehrere Kreuzungsvarianten unterschieden werden. Zwei davon werden im Detail erläutert. Alle weiteren Verfahren werden hier kurz angeführt: 1 - Punkt Crossover, N - Punkt Crossover, Shuffle Crossover, Intermediärer Crossover, Linearer Crossover, Tausch Crossover.

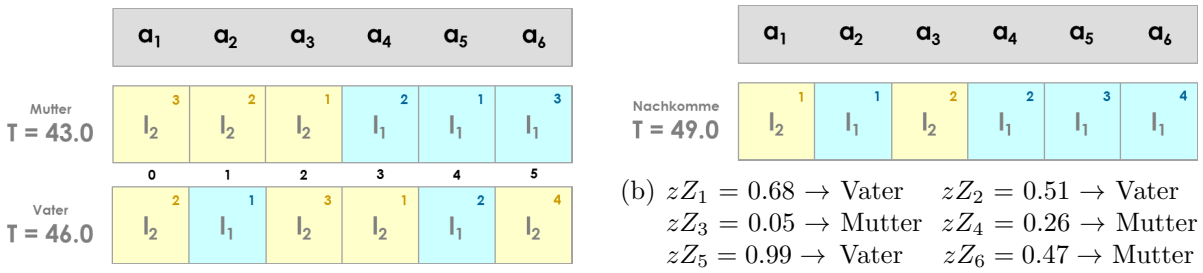
Gleichmäßiges Crossover

Ein neuer Nachkomme wird beim *gleichmäßigen Crossover* nach folgendem Prinzip erstellt. Es wird eine Zufallszahl $zZ \in [0.0, 1.0[$ erzeugt. Ist diese Zufallszahl $zZ < 0.5$ wird das Allel der Mutter vererbt. Ist die Zahl $zZ \geq 0.5$ wird die Genausprägung des Vaters verwendet. [14]

Musterbeispiel

Paarungspool mit Fitness $f(P_g) = [43.0, 52.0, 38.0, 46.0]$

Produktion P_1 und Produktion P_4 wurden zufällig ausgewählt. Produktion P_1 wird zum Mutter- und Produktion P_2 wird zum Vaterindividuum.



(b) $zZ_1 = 0.68 \rightarrow$ Vater $zZ_2 = 0.51 \rightarrow$ Vater
$zZ_3 = 0.05 \rightarrow$ Mutter $zZ_4 = 0.26 \rightarrow$ Mutter
$zZ_5 = 0.99 \rightarrow$ Vater $zZ_6 = 0.47 \rightarrow$ Mutter

Abbildung 7.3: Diese Grafik zeigt das Prinzip der gleichmäßigen Kreuzung.

[1] Englisch: Crossover

2 - Punkt Crossover

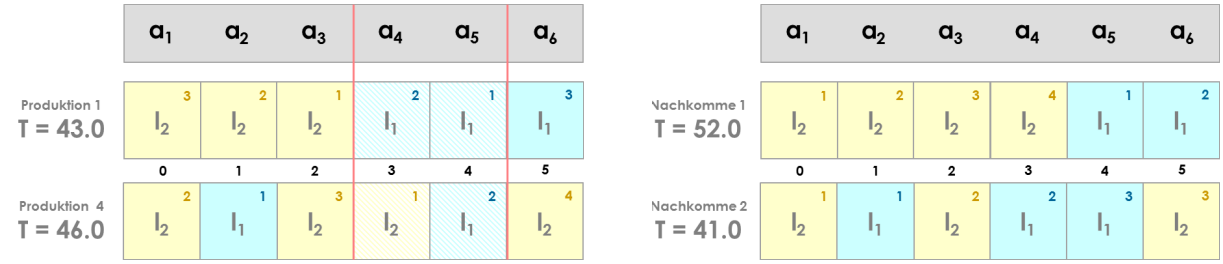
Beim *2 - Punkt Crossover* werden zwei neue Nachkommen nach folgendem Prinzip erzeugt. Es werden zufällig zwei Positionen p_1 und p_2, wobei $0 \leq p_1 \leq p_2 \leq n-1$, im Individuum bestimmt. Alle Aufträge, die innerhalb dieses Bereichs liegen, werden vertauscht. Im Konkreten wird dabei die Information, auf welcher Produktionslinie der Auftrag abgearbeitet wird, geändert. Die weiteren Aufträge werden 1:1 übernommen. Im biologischen Sinne bedeutet dies, dass die Allele der Gene der Elternindividuen bei der Fortpflanzung teilweise ausgetauscht werden. [14]

Musterbeispiel

Paarungspool mit Fitness $f(P_g) = [43.0, 52.0, 38.0, 46.0]$

Produktion P_1 und Produktion P_4 wurden zufällig ausgewählt.

Die Position wurden bestimmt: $p_1 = 3$ und $p_2 = 4$



(a) Die Allele von Auftrag a_4 und Auftrag a_5 werden vertauscht.

Abbildung 7.4: Diese Grafik zeigt das Prinzip der 2 - Punkt Kreuzung.

Mutation

Beim letzten Verfahrensschritt der genetischen Algorithmen handelt es sich um die *Mutation* von Individuen. Dabei wird versucht die unkontrollierbare Veränderung von Genen in der Natur nachzuahmen. Dieser Vorgang wird mit einer Mutationsrate mP gesteuert. Dieser Wert sollte allerdings nicht allzu groß angenommen werden, da gute Allelausprägungen somit leicht zerstört werden können. Wie bereits bei der Selektion und der Kreuzung angesprochen wurde, gibt es auch hier eine Unmenge an Mutationsarten. Einige davon sollen genannt werden: Normalverteilte Mutation, Verschiebe Operator, Translokations Operator, Mix Operator, Schüttel Operator.

Gleichverteilte Mutation

Alle Nachkommen werden einzeln und nach der Reihe durchlaufen. Bei der *gleichverteilten Mutation* wird für jedes Gen des Individuums eine Zufallszahl zZ erzeugt. Ist diese Zahl kleiner als die Mutationswahrscheinlichkeit mP, wird die Ausprägung geändert. In diesem Fall wird der Abarbeitungsort des Auftrags um eine Produktionslinie verschoben. [14]

Musterbeispiel

(a) Die Ausprägung von Auftrag a_6 wird mutiert.

Abbildung 7.5: Diese Grafik zeigt das Prinzip der gleichverteilten Mutation.

Ersetzungsstrategie

Nach Abschluss der Mutation wird die vorhandene Menge an Nachkommen um das beste und die beiden schlechtesten Individuen der Ausgangspopulation erweitert. Diese erweiterte Gruppe an Individuen bildet die neue Ausgangspopulation. Der beschriebene Ablauf beginnt von vorne. Beim Abbruchkriterium werden 2 Fälle unterschieden: Einerseits wird das Verfahren abgebrochen, wenn die letzten paar Populationen kein verbessertes Produktionsprogramm geliefert haben. Andererseits wird der Ablauf beendet, wenn bereits eine gewisse Anzahl an Populationen erstellt wurde. Je größer diese Werte sind, desto bessere Ergebnisse liefert der Algorithmus. Allerdings muss der Aspekt der Rechenzeit miteinbezogen werden.

Kapitel 8

Reihung

Zuerst werden alle Aufträge auf die verfügbaren Produktionslinien verteilt. Anschließend wird die erstellte Auftragsverteilung verbessert. Nun wird der Ablauf durch die Anpassung der *Abarbeitungsreihenfolge* fortgesetzt. Zu diesem Zweck wird jede Produktionslinie einzeln betrachtet. Das Ziel ist es, eine Auftragskonstellation zu finden, die die Gesamtdurchlaufzeit der Produktionslinie minimiert. Dieser Abschnitt der Arbeit wird einige vielversprechende Methoden vorstellen, die häufig im Zusammenhang mit Reihenfolgeverbesserung genannt werden.

Da Insertion Search, das Verfahren von Laha & Sarin und die k - Opt Methode ebenfalls der Gruppe der *Verbesserungs-* und *Suchverfahren* angehören, basieren sie auf demselben Prinzip, wie Simulated Annealing, Threshold Accepting, Tabu Search und genetische Algorithmen. Von einer oder mehreren Startlösungen ausgehend, wird nach verschiedenen Prinzipien der Lösungsraum untersucht. Den Hauptunterschied bilden dabei die angewandten Methoden zur Lösungserstellung. Auf Insertion Search wird in Kürze im Detail eingegangen. Weiters soll aufgezeigt werden, welche Abänderung des genetischen Algorithmus notwendig ist, um eine Reihenfolgeverbesserung durchzuführen.

Zu guter Letzt wird noch angemerkt, dass die Durchlaufzeit t_x und damit die Gesamtdurchlaufzeit T_i exakt berechnet werden. Dies ist möglich, da die verwendete Ausgangslösung ein Produktionsprogramm und Informationen über die Auftragsreihenfolge bereitstellt.

Laha & Sarin

Beim Verfahren von *Laha & Sarin* wird die Minimierung der Gesamtdurchlaufzeit T_i der Produktionslinie l_i angestrebt. Zu Beginn werden alle Aufträge aufsteigend nach ihrer Durchlaufzeit $t_1 \leq t_2 \leq \ldots \leq t_n$ sortiert. Anschließend erfolgt die Generierung einer Ausgangslösung.

Dazu werden die ersten beiden Aufträge a_1 und a_2 verwendet. Es wird überprüft, welche der beiden möglichen Auftragsanordnungen $\hat{A}_i = [a_1, a_2]$ oder $\hat{A}_i = [a_2, a_1]$ das bessere Ergebnis liefert. Auf Basis dieses unvollständigen Produktionsprogramms $\hat{A}_i$, werden der Reihe nach alle weiteren Aufträge in die Kundenauftragsliste eingegliedert. Folgendes Prinzip findet hier seine Anwendung. Zuerst wird ein Auftrag a_x an alle möglichen Positionen der aktuellen Kundenauftragsliste $\hat{A}_i$ gesetzt. Die Auftragsreihenfolge, welche die kleinste Gesamtdurchlaufzeit aufweist, wird angenommen, $\hat{A}_i$ wird um einen Auftrag erweitert. Dann werden alle weiteren Auftragskombinationen getestet, die durch Verschieben der bereits eingeplanten Aufträge $a_y \neq a_x$ möglich sind. Das beste daraus resultierende Produktionsprogramm wird gespeichert und bildet die Grundlage für den nächsten Iterationsschritt. Alle Aufträge, die noch nicht Teil der Kundenauftragsliste sind, werden nach dem selben Prinzip verteilt. Die beiden Entwickler dieses Verfahrens bestätigen, dass mit Hilfe dieser Prozedur sehr gute Ergebnisse erzielt werden. Leider ist dieses Verfahren auf Grund der enormen Laufzeit $\mathcal{O}(n_i^4 \cdot k)$ uninteressant für die Lösung der Aufgabenstellung dieser Arbeit. [12]

k - Opt

Das k - *Opt* Verfahren wird meistens im Zusammenhang mit der Lösung eines Traveling Salesman[1] Problems erwähnt. Aus diesem Grund wird die Funktionsweise auf diese Aufgabenstellung aufbauend, erläutert. Von einer bereits gefundenen Startlösung f ausgehend, wird versucht, eine Nachbarschaft $N_k(f)$ zu finden, deren Rundreisenlänge geringer ist als die der Ausgangstour. Eine neue Nachbarschaft $N_k(f)$ entsteht durch das Löschen von k Kanten aus der Startlösung f und durch anschließendes Hinzufügen von k neuen Kanten. Es sei erwähnt, dass die Wahl des Parameters k dabei eine große Herausforderung darstellt. Je größer k ist, desto bessere Ergebnisse liefert das Verfahren üblicherweise, aber desto länger dauert die Berechnung einer neuen Nachbarschaft. [8]

8.1 Insertion Search

Beschreibung

Insertion Search versucht, ausgehend von der Kundenauftragsliste A_i einer Produktionslinie l_i, durch verschieben eines Auftrages a_x in einem definierten Intervall, eine bessere Abarbeitungsreihenfolge zu finden. [20]

[1] Problem des Handlungsreisenden: Findung einer minimalen Rundreise in einem vorgegebenen Graph.

Zu Beginn wird eine Position p in der Kundenauftragsliste bestimmt. Normalerweise erfolgt die Selektion dieses Parameters zufällig. Bei der implementierten Variante jedoch wird die Position p mit Hilfe des Auftrags a_x gesetzt. Der Auftrag a_x entspricht dabei dem Auftrag, der aktuell die größte Rüstzeit in der Kundenauftragsliste A_i verursacht. Anschließend wird dieser Auftrag um $\alpha = \frac{n}{4}$ Positionen im Intervall $I_1 = [p+1, p+2, \ldots, min\{p+\alpha, n_i - 1\}]$ und $I_2 = [p-1, p-2, \ldots, max\{0, p-\alpha\}]$ verschoben. Der Parameter α wird auch Vertauschungsreichweite genannt. Wird bei keiner der Vertauschungen eine bessere Auftragsreihenfolge gefunden, wird der Parameter p zufällig bestimmt. Wurde die Hälfte aller Aufträge der Kundenauftragsliste A_i neu gereiht, wird das Suchverfahren beendet.

Pseudocode 5 Insertion Search

```
1: For All Produktionslinien l_i
2:   counter ← 0
3:   If n_i ≥ 2 Then
4:     While counter ≤ n_i/2
5:       p ← Position Auftrag a_x (Auftrag mit max. Rüstzeit oder zufälliger Auftrag)
6:       For k = p + 1 → min{p + α, n_i − 1}
7:         Verschiebe den Auftrag a_x an Position k
8:         If T der neuen Auftragskombination < T_Best Then
9:           A_Best ← neue Auftragsverteilung; T_Best ← T
10:        EndIf
11:      EndFor
12:      For k = p − 1 → max{0, p − α}
13:        Verschiebe den Auftrag a_x an Position k
14:        If T der neuen Auftragskombination < T_Best Then
15:          A_Best ← neue Auftragsverteilung; T_Best ← T
16:        EndIf
17:      EndFor
18:      counter + +
19:    EndWhile
20:  EndIf
21: EndFor
```

Musterbeispiel

Das verwendete Produktionsprogramm wurde zufällig erstellt. Produktionslinie l_1 wird im weiteren Verlauf des Beispieles näher betrachtet.

Auftragsverteilung $A_1 = [a_5, a_4, a_6, a_1, a_2] \rightarrow T_1 = 70.0$

Auftragsverteilung $A_2 = [a_3] \rightarrow T_2 = 12.0$

Ausgangssituation
Reichweite $\alpha = 6/4 = 1.5 \sim 1$

Auftrag a_4 an Position $p = 1$ weist die größte Rüstzeit $r_{j5}^{(4)} = 8$ auf, wobei $j = \{1, 2\}$.

(a) Der ausgewählte Auftrag a_4 wird im gekennzeichneten Intervall neu eingegliedert.

Vertauschung im Intervall $I_1 = [2]$
Auftrag a_4 wird um eine Position nach hinten vorschoben.

$62.0 \leq T_{Best} = 70.0$
Produktionsprogramm wird angenommen!

(b) Auftrag a_4 wird an Position $p = 2$ verschoben. Auftrag a_4 und Auftrag a_6 werden dabei vertauscht.

Vertauschung im Intervall $I_2 = [0]$
Auftrag a_4 wird um eine Position nach vorne vorschoben.

$66.0 \not\leq T_{Best} = 62.0$
Produktionsprogramm wird nicht angenommen!

(c) Auftrag a_4 wird an Position $p = 0$ verschoben. Auftrag a_4 und Auftrag a_5 werden dabei vertauscht. Dieser Ablauf würde nach demselben Prinzip noch weiter fortgesetzt werden.

Abbildung 8.1: Anhand dieser Grafik wird der Ablauf von Insertion Search dargestellt.

8.2 Genetische Algorithmen

Beschreibung

Auf das Prinzip der *genetischen Algorithmen* sowie die Erläuterung der einzelnen Verfahrensschritte wurde bereits in Kapitel *Verbesserung* sehr genau eingegangen. Aus diesem Grund werden in diesem Abschnitt der Arbeit nur Implementierungsunterschiede erläutert, Methodenwissen zu dieser Thematik wird allerdings vorausgesetzt. Die einzige bedeutende Veränderung kann bei der Definition eines *Individuums* vermerkt werden.

Zum Zweck der Verbesserung entspricht ein Individuum einer Produktion P. Das Ziel dabei ist es, die maximale Gesamtdurchlaufzeit T eines Individuums zu minimieren.

Im Zusammenhang mit der Reihenfolgeverbesserung kann ein Individuum wie folgt beschrieben werden. Ein Individuum kann als Produktionslinie l_i der Produktion P angesehen werden. Das Hauptaugenmerk wird dabei auf die Minimierung der Gesamtdurchlaufzeit T_i der Produktionslinie l_i gelegt. Genauer gesagt, wird von mehreren bestehenden Produktionsprogrammen A_i ausgehend versucht, durch kontinuierliches Verändern der Abarbeitungsreihenfolge, eine bessere Kundenauftragsliste zu finden. Unter Einbeziehung dieser Variation können alle bereits erläuterten Selektionsverfahren, Kreuzungsmethoden und Mutationsarten angewandt werden.

Kapitel 9

Korrektur

Abgeschlossen wird das vierstufige Lösungssystem mit einem *Korrekturalgorithmus*. Bevor dieser Schritt näher erläutert wird, sollen noch einmal kurz die anderen Verfahrensschritte in Erinnerung gerufen werden.

Begonnen wird mit der Zuordnung der Aufträge zu den verfügbaren Produktionslinien, die erforderlichen Berechnungen beruhen dabei auf Näherungswerten. Beim Zuordnungsschritt wird immer die gesamte Produktion P betrachtet. Im Anschluss daran wird die erstellte Auftragsverteilung verbessert, dabei steht wieder die gesamte Produktion P im Mittelpunkt der Betrachtung. Ab diesem Zeitpunkt werden alle erforderlichen Parameter exakt kalkuliert. Der nächste Schritt ist die Anpassung der Abarbeitungsreihenfolge. Zu diesem Zweck werden alle Produktionslinien l_i unabhängig voneinander betrachtet. Diese getrennte Perspektive ist auch der Grund dafür, dass zum Abschluss eine Korrektur erfolgen muss.

Mit Hilfe des Korrekturalgorithmus soll die Differenz $D = |T_{Min} - T_{Max}|$ der Produktionslinie l_{Min} mit minimaler und l_{Max} mit maximaler Gesamtdurchlaufzeit verringert werden. Somit ist es nachträglich möglich, die negative Auswirkung, welche die separierte Betrachtung der Produktionslinien mit sich bringt, zu schmälern.

Beschreibung

Das Ziel des Korrekturalgorithmus ist die Minimierung der Differenz $D = |T_{Min} - T_{Max}|$. Das Verfahren versucht, durch Umschichten der Aufträge entstandene Kapazitätslücken zu nutzen und dadurch eine ausgewogene Belastung aller Produktionslinien zu erreichen.

Zu Beginn wird die Produktionslinie l_{Min} mit minimaler und l_{Max} mit maximaler Gesamtdurchlaufzeit bestimmt. Danach wird die Kundenauftragsliste A_{Max} von Produktionslinie l_{Max} durchlaufen. Dabei wird nach Aufträgen a_x Ausschau gehalten, die folgende Kriteri-

en erfüllen: Die Gesamtdurchlaufzeit T_{Min} der Produktionslinie l_{Min} muss nach Hinzufügen des Auftrags a_x kleiner sein, als die Gesamtdurchlaufzeit T_{Max}. Weiters muss nach anschließendem Löschen des Auftrags a_x von Produktionslinie l_{Max} gewährleistet werden, dass die definierte Differenz D reduziert wurde. Die Suche beginnt von vorne, wenn ein passender Auftrag gefunden werden konnte. Anderenfalls wird das Verfahren beendet. Spätestens nach 50 Iterationsschritten jedoch wird der Algorithmus abgebrochen, um die Einhaltung der Laufzeitschranke sicherzustellen.

Pseudocode 6 Korrekturalgorithmus

1: $anzahlIterationen \leftarrow 0$
2: **If** $anzahlIterationen \geq 50$ **Then**
3: return
4: **Else**
5: $l_{Min} \leftarrow l_i$ mit minimaler Gesamtdurchlaufzeit $T_{Min} = min\{T_1, \ldots, T_s\}$
6: $l_{Max} \leftarrow l_i$ mit maximaler Gesamtdurchlaufzeit $T_{Max} = max\{T_1, \ldots, T_s\}$
7: **For All** Aufträge a_x der Kundenauftragsliste A_{Max}
8: Auftrag a_x zu Produktionslinie l_{Min} hinzufügen
9: **If** $T_{Min} < T_{Max}$ **Then**
10: Auftrag a_x von Produktionslinie l_{Max} löschen
11: **If** $|T_{Min} - T_{Max}| < D_{Best}$ **Then**
12: $P_{Best} \leftarrow$ neue Produktion; $D_{Best} \leftarrow |T_{Min} - T_{Max}|$
13: Ausgangszustand der Produktion herstellen
14: **Else**
15: Ausgangszustand der Produktion herstellen
16: **EndIf**
17: **Else**
18: Auftrag a_x von Produktionslinie l_{Min} löschen
19: **EndIf**
20: **EndFor**
21: **If** Es wurde ein passender Auftrag a_x gefunden **Then**
22: Rekursiver Aufruf ($P \leftarrow P_{Best}$; $anzahlIterationen++$) $\curvearrowright$ Schritt 2
23: **Else**
24: return
25: **EndIf**
26: **EndIf**

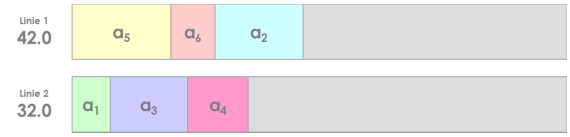
Musterbeispiel

Das verwendete Produktionsprogramm wurde zufällig erstellt.

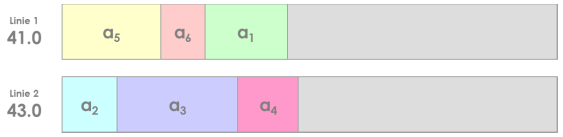
Auftragsverteilung $A_1 = [a_5, a_6] \rightarrow T_1 = 26.0$

Auftragsverteilung $A_2 = [a_2, a_1, a_3, a_4] \rightarrow T_2 = 46.0$

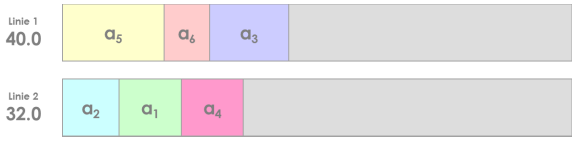
Differenz $D_{Best} = |T_{Min} - T_{Max}| = |26.0 - 46.0| = 20.0$



(a) Auftrag a_2 wird zu Produktionslinie l_1 hinzugefügt. Da $42.0 < 46.0$ und $10.0 < 20.0$, wird das Produktionsprogramm angenommen!



(b) Auftrag a_1 wird zu Produktionslinie l_1 hinzugefügt. Da $43.0 < 46.0$ und $2.0 < 10.0$, wird das Produktionsprogramm angenommen!



(c) Auftrag a_3 wird zu Produktionslinie l_1 hinzugefügt. Da zwar $40.0 < 46.0$, aber $8.0 \not< 2.0$, wird das Produktionsprogramm nicht angenommen!

Der Ablauf wird mit dem Produktionsprogramm, das in Schritt (b) erstellt wurde, wie beschrieben fortgesetzt.

Abbildung 9.1: Anhand dieser Grafik wird der Ablauf des Korrekturalgorithmus dargestellt.

Kapitel 10

Ergebnispräsentation und Fazit

Das letzte Kapitel dieser Arbeit befasst sich intensiv mit der Evaluierung der implementierten Verfahren. Das Ziel der Analyse soll es sein, eine Empfehlung für die stabilsten und zuverlässigsten Algorithmen, welche die Aufgabenstellung innerhalb einer vorgeschriebenen, beschränkten Laufzeit bewältigen, zu liefern.

Zu Beginn erfolgt eine Beschreibung der Testfälle. Im Anschluss daran wird die Verteilung der Daten und folglich die Wahl der statistischen Testmethode beschrieben. Zuletzt werden die gewonnenen Ergebnisse präsentiert und interpretiert.

10.1 Erarbeitung der Testfälle

Bevor überhaupt mit der Auswertung begonnen werden kann, müssen konkrete Testfälle formuliert und Datensätze erhoben werden. Laut Definition der Problemstellung gibt es fünf variable Parameter: Bearbeitungszeit tb_x, Rüstzeit tr_x, Anzahl der Aufträge n, Anzahl der Produktionslinien s, Anzahl der Maschinen k. Alle möglichen Ausprägungen dieser Kriterien würden, ohne dabei unterschiedliche Bearbeitungs- und Rüstzeiten mit einfließen zu lassen, bereits über $n \cdot s \cdot k = 200 \cdot 4 \cdot 20 = 16000$ Testfälle je Algorithmus ergeben. Dies entspricht einer reinen Rechenzeit von zirka einem Tag. Eine detaillierte Variation von Bearbeitungs- und Rüstzeiten würde den berechneten Wert noch um einiges vervielfachen. Schlussfolgernd muss gesagt werden, dass dies den Rahmen der Diplomarbeit sprengen würde. Aus diesem Grund wurde eine kleine Anzahl an konkreten Parameterwerten ausgewählt. Mit Hilfe dieser Selektion wird ein repräsentativer Teil der realistischen Aufgabenkombinationen abgedeckt.

Die Datensätze die im Zuge einer Simulation generiert werden, werden gesammelt und in folgendem Format gespeichert.

Die Werte der Abszisse bilden die Bearbeitungszeiten tb_x der Aufträge a_x. Dabei wird die Bearbeitungszeit aus einem Zufallsintervall ausgewählt. Beginnend mit dem Intervall $[10, 20]$ wird die obere Schranke solange um den Wert 30 erhöht, bis das Intervall $[10, 1450]$ erreicht wird.

Bei den entsprechenden Messwerten der Ordinate handelt es sich um die maximale Gesamtdurchlaufzeit $\overline{T}$ der erstellten Produktionsprogramme.

Bei der Veranschaulichung anhand eines Diagramms, wird auf der y - Achse die Differenz $D = (\overline{T} - K_i)$ aufgetragen. Diese Differenz gibt an, wie weit die gemessene, maximale Gesamtdurchlaufzeit $\overline{T}$ vom definierten Richtwert, der angenäherten Kapazität $K_i = \frac{1}{s} \sum_{x=1}^{n} t_x$, wobei $t_x = bE_x + rM_x$, entfernt ist. Genauer gesagt, wird die maximale Gesamtdurchlaufzeit $\overline{T}$ durch den Mittelwert von $z = 50$ Testiterationen T_g berechnet.

$$\overline{T} = \frac{1}{z} \sum_{g=1}^{z} T_g$$

Variation der Aufträge

Der erste Testfall untersucht das Verhalten der Algorithmen bei steigender Auftragszahl. Es werden folgende Zustände des Systems angenommen: $n = [10, 25, 50, 100, 200]$. Zu diesem Zweck werden die restlichen Parameter mit folgenden Werten hinterlegt:

Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 1$

Variation der Produktionslinien

Der zweite Testfall untersucht das Verhalten der Algorithmen bei sinkender Anzahl der Produktionslinien. Es werden folgende Zustände des Systems angenommen: $s = [4, 3, 2]$. Zu diesem Zweck werden die restlichen Parameter mit folgenden Werten hinterlegt:

Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Aufträge $n = 50$
Anzahl der Maschinen $k = 1$

Variation der Maschinen

Der dritte Testfall untersucht das Verhalten der Algorithmen bei steigender Anzahl der Maschinen. Es werden folgende Zustände des Systems angenommen: $k = [1, 5, 10, 15]$. Zu diesem Zweck werden die restlichen Parameter mit folgenden Werten hinterlegt:

Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Aufträge $n = 50$
Anzahl der Produktionslinien $s = 4$

Variation der Rüstzeit

Der vierte Testfall untersucht das Verhalten der Algorithmen bei Rüstzeiten, die eine Abhängigkeit zu ihrer Bearbeitungszeit aufweisen. Es werden folgende Zustände des Systems angenommen: $tr_x = [0.25 \cdot tb_x, 0.5 \cdot tb_x, 1 \cdot tb_x]$. Zu diesem Zweck werden die restlichen Parameter mit folgenden Werten hinterlegt:

Anzahl der Aufträge $n = 50$
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 1$

10.2 Kolmogoroff Smirnow Test

Zu Beginn der Datenauswertung ist es wichtig die Testergebnisse hinsichtlich ihrer Wahrscheinlichkeitsverteilung zu prüfen. Es hat sich herausgestellt, dass die Gesamtdurchlaufzeiten normalverteilt sind.

Zu diesem Zweck wird der *Kolmogoroff Smirnow* Test verwendet. Dieses Verfahren wird immer dann angewandt, wenn überprüft werden soll, ob eine Stichprobe einer gewissen Verteilung genügt.

Das Prinzip lässt sich wie folgt beschreiben: Der Abstand zwischen der empirischen Summenfunktion $\tilde{F}(x)$, sowie der kumulierten Verteilungsfunktion $F(x)$ einer Referenzverteilung wird gemessen und in Beziehung zum erwünschten Signifikanzniveau α gesetzt. Auf Basis dieses Verhältnisses wird darüber entschieden, ob die Gesamtdurchlaufzeiten mit Signifikanz $\alpha = 0.05$ normalverteilt sind oder nicht.

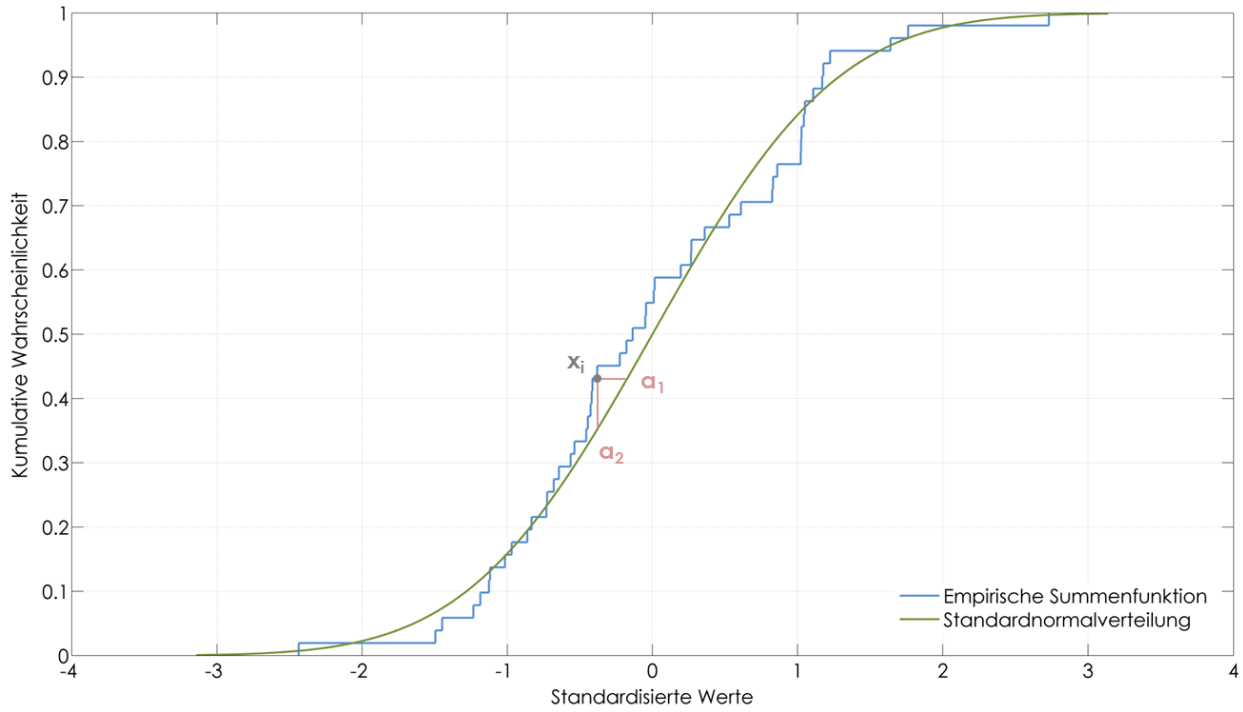


Abbildung 10.1: Anhand dieser Grafik wird veranschaulicht, dass es sich bei den gemessenen Gesamtdurchlaufzeiten, die mit Hilfe der empirischen Summenformel dargestellt werden, um normalverteilte Daten handelt.
Die Daten wurden anhand folgendes Szenarios generiert:
Best Fit, $n = 200$, $s = 4$, $k = 1$, $tb_x \in [10, 360]$ Minuten, $tr_x \in [0, 60]$ Minuten

Im Detail läuft der Kolmogoroff Smirnow Test wie folgt ab. Zu Beginn werden die relativen Häufigkeiten der diskreten Gesamtdurchlaufzeiten ermittelt, aufaddiert und mit Hilfe der Truppenfunktion $\tilde{F}(x)$ beschrieben. Anschließend wird für jeden Messwert der horizontale a_1 und der vertikale a_2 Abstand zwischen der empirischen Summenfunktion und der kumulativen Häufigkeitsverteilung berechnet:

$$a_1 = |\tilde{F}(x_{i-1}) - F(x)| \quad \text{und} \quad a_2 = |\tilde{F}(x_i) - F(x_i)| \tag{10.1}$$

Dabei gilt für die kumulative Wahrscheinlichkeitsverteilung $F(x)$:

$$F(x) = \phi\left(\frac{x_i - \bar{x}}{s}\right) \tag{10.2}$$

Für den normalisierten Wert $\phi(z)$ kann anschließend das entsprechende Quantil aus der Tabelle der Standardnormalverteilung abgelesen werden. Zu diesem Zweck müssen der Mittelwert $\bar{x}$ und die Standardabweichung s für Stichproben genauer definiert werden:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{und} \quad s = \sqrt{s^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{10.3}$$

Aus allen berechneten Werten a_1, a_2 wird der Größte, der gleichzeitig den maximalen Abstand zwischen den beiden Kurven repräsentiert, ausgewählt. Dieser Abstand wird mit a bezeichnet. Nun wird mittels bekannter Stichprobengröße n und der Signifikanzgröße α der Wert c aus der Verteilungstabelle des Kolmogoroff Smirnow Test abgelesen.

Die Stichprobe gilt als normalverteilt, wenn $a < c$. [11]

10.3 t - Test

Der nächste Schritt ist die Auswertung der Daten. Da die Gesamtdurchlaufzeiten normalverteilt sind, kann der *t - Test* hier angewendet werden. Mit Hilfe der Mittelwerte $\bar{x}_1, \bar{x}_2$ zweier unabhängiger Stichproben, kann überprüft werden, ob die dazugehörigen Mittelwerte μ_1, μ_2 der Grundgesamtheit gleich oder verschieden sind.

Schlussfolgernd kann die Nullhypothese wie folgt formuliert werden:
$H_0 : \mu_1 = \mu_2$
$H_1 : \mu_1 > \mu_2$

Der konkrete Ablauf dieser statistischen Testmethode lässt sich wie folgt beschreiben: Zu Beginn wird der Wert t auf Basis folgender Berechnungsgrundlage bestimmt. Dabei entsprechen n_1, n_2 der Anzahl der Elemente der jeweiligen Stichprobe:

$$t = \sqrt{\frac{n_1 n_2 (n_1 + n_2 - 2)}{n_1 + n_2}} \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}} \tag{10.4}$$

Liegen zwei Stichproben mit identischer Größe $n_1 = n_2$ vor, kann die Formel vereinfacht werden:

$$t = \sqrt{n} \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} \tag{10.5}$$

Nun wird mit Hilfe der Anzahl an Freiheitsgraden $n_1 + n_2 - 2$ und dem festgelegten Signifikanzniveau α der Wert c aus der t - Test Verteilungstabelle abgelesen.

Die Nullhypothese wird angenommen, wenn $t < c$. Das würde bedeuten, dass keiner der evaluierten Algorithmen ein signifikant besseres Produktionsprogramm liefert als der andere. [11]

10.4 Wald - Wolfowitz Test

Bei Daten die keiner konkreten Wahrscheinlichkeitsverteilung zugeordnet werden können, kann ein *nichtparametrisches Verfahren* zur Analyse verwendet werden. Dieser Abschnitt soll die theoretischen Grundlagen des *Wald - Wolfowitz* Tests, der dieser Gruppe angehört, näher erläutern.

Beim Wald - Wolfowitz handelt sich um eine nichtparametrische Methode, die die Zufälligkeit einer Testdatenfolge verifiziert. In diesem konkreten Fall könnte damit überprüft werden, ob sich die Gesamtdurchlaufzeiten der Produktionsprogramme, die mit Hilfe eines bestimmten Algorithmus erstellt wurden, signifikant vom Rest unterscheiden. Der Ablauf kann wie folgt definiert werden.

Bevor mit der Auswertung der Testdaten begonnen werden kann, muss eine Hypothese formuliert werden. In dieser Arbeit gilt:
H_0 : Keiner der beiden Algorithmen ist signifikant besser!
H_1 : Eines der beiden Verfahren ist signifikant besser!

Der nächste Schritt ist die Aufbereitung der Messergebnisse. Normalerweise werden diese durch zwei unabhängige Datensätze, die auch als Klassen bezeichnet werden, dargestellt. Diese beiden Teilmengen werden aufsteigend nach einem gewissen Kriterium sortiert und im Zuge dessen zusammengefügt. Anschließend wird der spezifische Wert jedes Testergebnisses durch seine zugehörige Klasse ersetzt. Folgende Datensequenz, die chronologische Abfolge genannt wird, wäre möglich 1 1 0 1 1 0 0 1. Daraus kann $n_1 = 5$ und $n_2 = 3$ abgeleitet werden, wobei n_1 der Anzahl der Elemente, die Klasse 1 zugeordnet werden und n_2 der Menge aller Elemente, die aus Klasse 2 ausgewählt wurden, entspricht. Umgelegt auf diese Arbeit würde dies bedeuten, dass in n_1 von $n_1 + n_2$ Fällen, das erste Verfahren das Produktionsprogramm mit geringerer Gesamtdurchlaufzeit generiert hat.

Aufbauend auf die zuvor erstellte Datensequenz müssen sogenannte Runs identifiziert werden. Ein Run fasst dabei eine Abfolge an Elementen mit der gleichen Klassenzugehörigkeit zusammen. Die Summe aller Runs wird mit r bezeichnet.

Angewandt auf das vorher angeführte Beispiel würde dies folgendes bedeuten:
$| 1\ 1 | 0 | 1\ 1 | 0\ 0 | 1 | \rightarrow r = 5.$

Zu guter Letzt muss der Parameter r einer Signifikanzprüfung unterzogen werden. Allgemein gilt, je größer r, desto wahrscheinlicher ist es, dass das Ergebnis zufälliger Natur ist. Außerdem muss das Signifikanzniveau α, das ein Maß für die Irrtumswahrscheinlichkeit darstellt, definiert werden. Normalerweise wird $\alpha = 0.05$ gesetzt. Die Prüfmethode hängt von der Größe der Aufgabenstellung ab.

Kleiner Testdatensatz

Im Falle eines kleinen Testdatensatzes, $n_1 \leq 20$ und $n_2 \leq 20$, wird die Anzahl aller möglichen Permutationen an Anordnungen als Berechnungsgrundlage verwendet.

$$\binom{n_1 + n_2}{n_1} = \binom{n_1 + n_2}{n_2} \tag{10.6}$$

Auf diese Beziehung aufbauend, kann die Wahrscheinlichkeit, dass die Anzahl der ermittelten Runs r oder weniger als r ist, bestimmt werden. Für eine gerade Anzahl $k = \frac{r}{2}$ an Runs gilt:

$$p(r \leq r') = \frac{1}{\binom{n_1+n_2}{n_1}} \sum_{r=2}^{r'} 2\binom{n_1 - 1}{k-1}\binom{n_2 - 1}{k-1} \tag{10.7}$$

Des Weiteren kann die Wahrscheinlichkeit für eine ungerade Anzahl $k = \frac{r-1}{2}$ an Runs wie folgt definiert werden:

$$p(r \leq r') = \frac{1}{\binom{n_1+n_2}{n_1}} \sum_{r=2}^{r'} \left[\binom{n_1 - 1}{k-1}\binom{n_2 - 1}{k-2} + \binom{n_1 - 1}{k-2}\binom{n_2 - 1}{k-1}\right] \tag{10.8}$$

Üblicherweise werden mit Hilfe dieser Formeln Tabellen erzeugt, mit denen es sehr leicht möglich ist, den entsprechenden Wert für r' abzulesen. Ist der abgelesene Wert r' kleiner als der Wert r, der auf Basis des Testdatensatzes ermittelt wurde, wird die Nullhypothese H_0 abgelehnt. [17]

Großer Testdatensatz

Im Falle eines großen Testdatensatzes $n_1 > 20$ oder $n_2 > 20$, wird die Normalverteilung zur Approximation verwendet. Für den Mittelwert μ_r gilt:

$$\mu_r = \frac{2n_1n_2}{n_1 + n_2} + 1 \tag{10.9}$$

Die Standardabweichung σ_r kann wie folgt berechnet werden:

$$\sigma_r = \sqrt{\frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}} \tag{10.10}$$

Mittels Mittelwert und Standardabweichung lässt sich die Formel für den Wert z definieren:

$$z = \frac{r - \mu_r}{\sigma_r} \tag{10.11}$$

Damit bei einer Aufgabenstellung mit relativ kleinem n_1, n_2 sichergestellt werden kann, dass der Wert z richtig angenähert wird, erfolgt eine Stetigkeitskorrektur der Kurve:

$$z = \frac{|r - \mu_r| - 0.5}{\sigma_r} \tag{10.12}$$

Ausgehend von diesem Wert z und dem Signifikanzniveau α kann das entsprechende Quantil in der Tabelle für die Normalverteilung nachgeschlagen werden. Der Wert des Quantils wird verdoppelt und bildet nun den Vergleichswert r'. Es gilt wiederum: Ist der berechnete Wert r' kleiner als der Wert r, der auf Basis des Testdatensatzes ermittelt wurde, dann wird die Nullhypothese H_0 abgelehnt. [17]

Resümee zur Anwendbarkeit

Bevor auf die Untersuchungsresultate eingegangen wird, soll gleich vorweggenommen werden, dass der Wald - Wolfowitz Test für die Datenkonstellation dieser Aufgabenstellung ungeeignet ist. Es hat sich herausgestellt, dass das Verfahren sowohl bei händischer Berechnung, als auch bei der Kalkulation mittels Matlab[1] sehr oft kein sinnvolles Ergebnis liefert. Obwohl anhand einer Kurve, die den Verlauf der maximalen Gesamtdurchlaufzeiten abbildet, eindeutig zu erkennen ist, dass ein Algorithmus bis auf wenige Ausreißer eindeutig besser ist als ein Vergleichsverfahren, liefert der Test als Antwort „H_0 wird angenommen“. Das bedeutet,

[1] © The MathWorks Inc., Natick, MA, United States of America, www.mathworks.com

dass das Ergebnis laut Wald - Wolfowitz Test zufälliger Natur ist.

Veranschaulicht kann dies auf der nachfolgenden Grafik werden. Auf der x - Achse werden zu diesem Zweck Bearbeitungszeiten im Intervall [10,1450] aufgetragen. Auf der y - Achse wird die Differenz $D = (\bar{T} - K_i)$ abgebildet, die angibt, wie groß der Unterschied zwischen der gemittelten, maximalen Gesamtdurchlaufzeit und der angenäherten Kapazität K_i der Produktionslinien ist. Es ist eindeutig erkennbar, dass mittels Best - Fit und First - Fit Produktionsprogramme generiert werden, deren Richtwert D geringer und somit besser als der des Minimum Bin Slack Algorithmus ist. Trotzdem ergibt die Evaluierung mittels Wald - Wolfowitz Test, dass keine Aussage darüber getroffen werden kann, welcher der verglichenen Methoden effizientere Lösungen erzeugt.

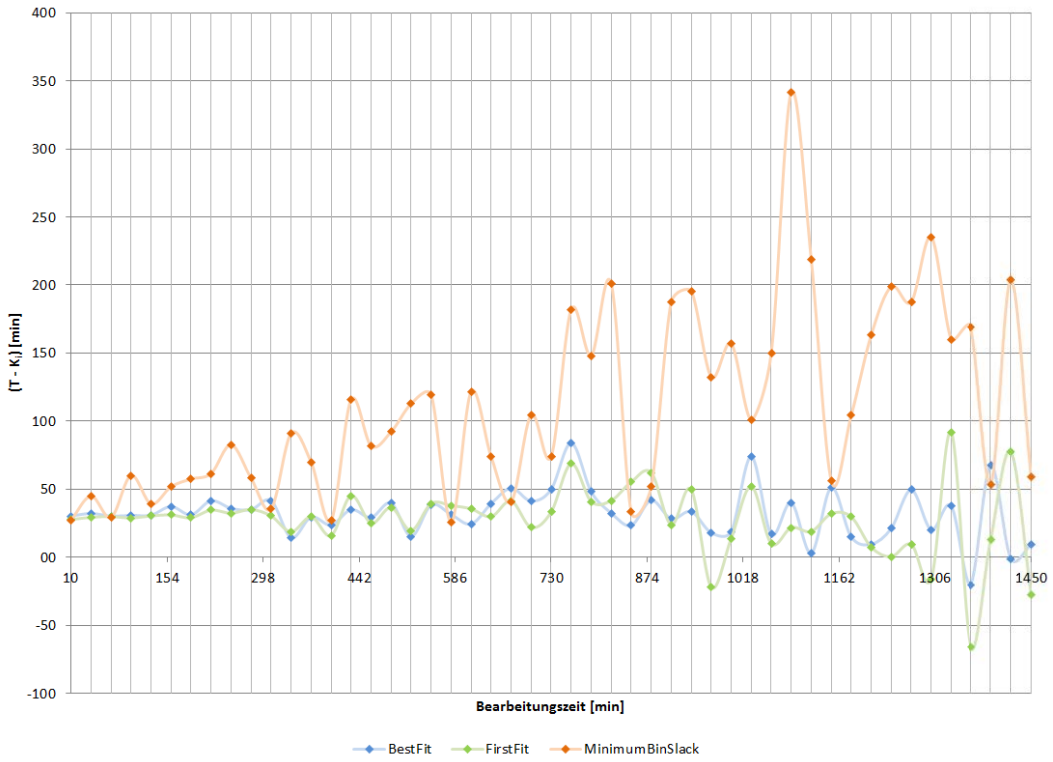


Abbildung 10.2: Anhand dieser Grafik wird dargestellt, dass der Wald - Wolfowitz keine korrekte Aussage über die Qualität der Algorithmen trifft.

10.5 Auswertung

Dieser letzte Abschnitt der Arbeit präsentiert die Evaluierungsergebnisse, die auf Basis der beschriebenen Testfälle generiert und mit Hilfe des t - Tests ausgewertet wurden. Der t - Test wurde dem Wald - Wolfowitz Test vorgezogen, da dieser aus den beschriebenen Gründen leider keine sinnvollen Ergebnisse liefert.

Die genauen Daten zu den nachfolgend angeführten Diagrammen befinden sich im Anhang.

Genetische Algorithmen

Es sei bereits zu Beginn vorweggenommen, dass die genetischen Algorithmen für diese Art der Aufgabenstellung reichlich ungeeignet sind. Auf Grund der Laufzeitbeschränkung ist es nicht möglich bei komplexen Problemen (große Auftragszahlen, hohe Rüstzeiten, viele Maschinen,...) sinnvolle Aufgabenverteilungen zu erzielen, da die Algorithmen frühzeitig abgebrochen werden. Diese Methode würde bei Vernachlässigung der Laufzeitschranke mit Sicherheit effiziente Produktionsprogramme erzeugen, die mit den sehr guten Ergebnissen der einfachen Approximationsverfahren mithalten könnten.

Im Konkreten konnten folgende Schwachstellen identifiziert werden, die auf die vorgegebene Laufzeitschranke zurückgeführt werden können:

- Die Anzahl der Individuen ist zu gering!
- Es können nicht genügend Populationen erzeugt werden!
- Die konkreten Werte der Parameter wurden nicht aufgrund umfangreicher Testläufe bestimmt, sondern großteils der Fachliteratur entnommen!
- Die Selektionsmethoden, Kreuzungsverfahren und die Mutationsprozedur wurden möglicherweise nicht optimal auf diese Problemstellung angepasst!

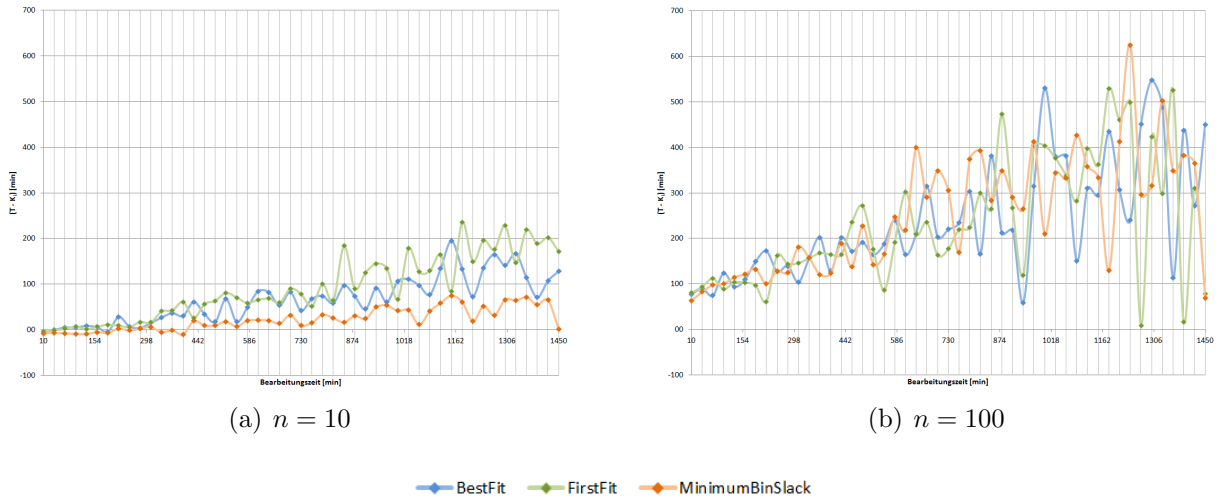
Da das Hauptthema dieser Arbeit nicht die Untersuchung der Anwendbarkeit von genetischen Algorithmen für das hier vorliegende Problem war, muss kurz und knapp gesagt werden, dass die Verfahren in dieser Form unbrauchbar sind. Da das große Potential dieser Methoden bereits sehr oft nachgewiesen wurde, wäre es sicher wünschenswert, diese Thematik in einer eigenen Studie genauer zu hinterfragen.

Zuordnung

Die Zuordnung der Aufträge zu den verfügbaren Produktionslinien erfolgt mittels Best - Fit, First - Fit oder Minimum Bin Slack. Folgende Resultate konnten mit Hilfe der statistischen Analyse gewonnen werden.

Variation der Aufträge

Bei der Variation der Aufträge fällt auf, dass bei einer geringen Anzahl der Minimum Bin Slack Algorithmus sehr gut funktioniert und Ergebnisse liefert die um $\sim 3.7\%$ besser sind, als die des Best - Fit und First - Fit Verfahrens. Bei steigender Auftragszahl $n \geq 25$, werden die Produktionsprogramme, die mittels Minimum Bin Slack generiert werden immer schlechter. Die Auftragsverteilungen der beiden anderen Verfahren jedoch, können als gleich gut eingestuft werden. Die sinkende Tendenz des Minimum Bin Slack ist auf die Tatsache zurückzuführen, dass bei einer geringeren Anzahl an Aufträgen alle Permutationen an Auftragskombinationen ausprobiert werden können, dies bei größeren Mengen auf Grund der Laufzeitbeschränkung jedoch nicht mehr möglich ist. Allgemein gilt, je mehr Aufträge den verfügbaren Produktionslinien zugeordnet werden sollen, desto größer ist die Differenz zwischen maximaler Gesamtdurchlaufzeit und errechnetem Richtwert.



(a) $n = 10$ (b) $n = 100$

Abbildung 10.3: Diese Grafik zeigt die negative Entwicklung des Minimum Bin Slack Verfahrens bei steigender Auftragszahl.

Variation der Produktionslinien

Außerdem wurde festgestellt, dass die Anzahl der verfügbaren Produktionslinien keinen Einfluss auf die Qualität der erstellten Produktionsprogramme der einzelnen Approximationsverfahren hat.

Variation der Maschinen

Bei steigender Maschinenanzahl wird deutlich, dass es sich bei der First - Fit Methode um das beste Verfahren handelt. Damit werden Produktionsprogramme erzeugt, die um $\sim 1.0\%$ besser sind als jene Auftragsverteilungen, die mittels Best - Fit oder Minimum Bin Slack generiert werden.

Variation der Rüstzeit

Bei der Auswertung des letzten Testfalls war es offensichtlich, dass sowohl Best - Fit als auch First - Fit Auftragsverteilungen erzeugen, deren maximale Gesamtdurchlaufzeit um $\sim 1.7\%$ geringer ist, als die des Minimum Bin Slack Verfahrens.

Fazit Zuordnung

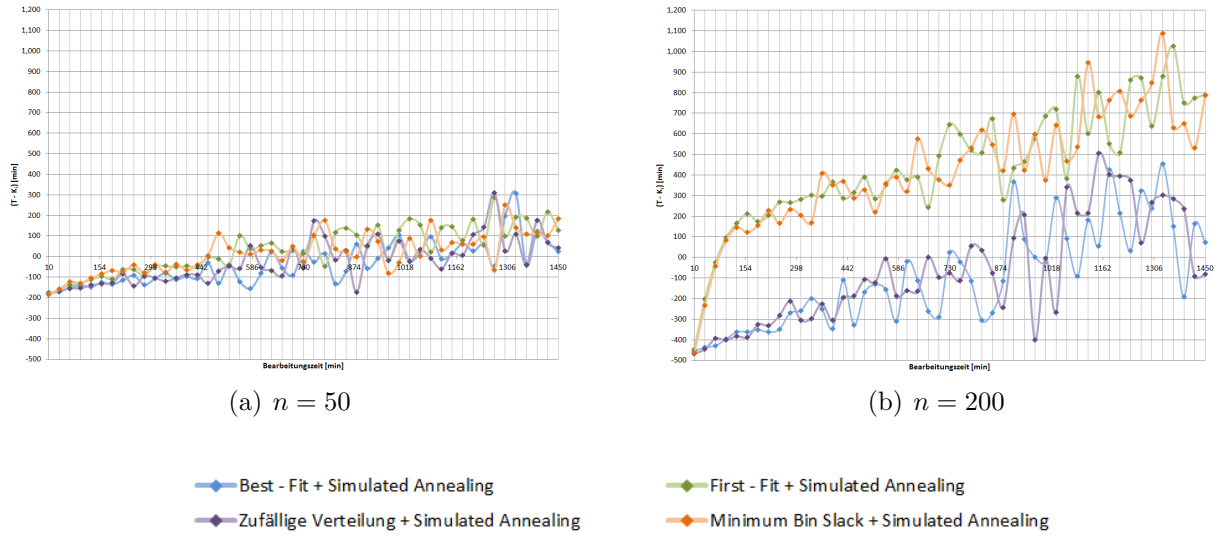
Zusammenfassend kann gesagt werden, dass bei sehr kleinen Auftragszahlen $n \leq 25$ auf jeden Fall der *Minimum Bin Slack* Algorithmus angewandt werden muss. Bei steigender Auftrags- und Maschinenanzahl ist es jedoch sinnvoller *Best - Fit* oder *First - Fit* auszuwählen. Obwohl die First - Fit Methode bei vielen Maschinen ein eindeutig effizienteres Produktionsprogramm liefert, sollte bei einer konkreten Aufgabenstellung auch immer das Best - Fit Verfahren angewandt werden. Die zwei Methoden agieren nach verschiedenen Prinzipien und wirken sich daher unterschiedlich auf spezifische Probleme aus. Auf Grund ihrer effizienten Arbeitsweise muss auch bei der Anwendung beider Algorithmen kaum Zeitverlust in Kauf genommen werden.

Verbesserung

Die Verbesserung der zuvor erstellten Auftragsverteilung erfolgt mittels Simulated Annealing und genetischen Algorithmen. Zu diesem Zweck werden Ausgangslösungen mit Best - Fit, First - Fit, Minimum Bin Slack und zufällig erzeugt. Alle möglichen Algorithmenkombinationen wurden getestet. Folgende Resultate konnten mit Hilfe der statistischen Analyse gewonnen werden.

Variation der Aufträge

Es fällt auf, dass bei einer geringen Anzahl an Aufträgen alle Kombinationen aus Zuordnungsalgorithmus und Simulated Annealing gleich gute Ergebnisse liefern. Bei steigender Auftragszahl $n \geq 100$, erhöhen die Produktionsprogramme die mittels Best - Fit oder zufällig initialisiert und mit Hilfe von Simulated Annealing verbessert wurden, ihre Effizienz.



(a) $n = 50$ (b) $n = 200$

Abbildung 10.4: Diese Grafik zeigt, dass bei steigender Auftragszahl die Kombination aus Best - Fit mit Simulated Annealung sowie die Generierung einer zufälligen Lösung mit anschließender Verbesserung, die besten Ergebnisse liefern.

Variation der Produktionslinien

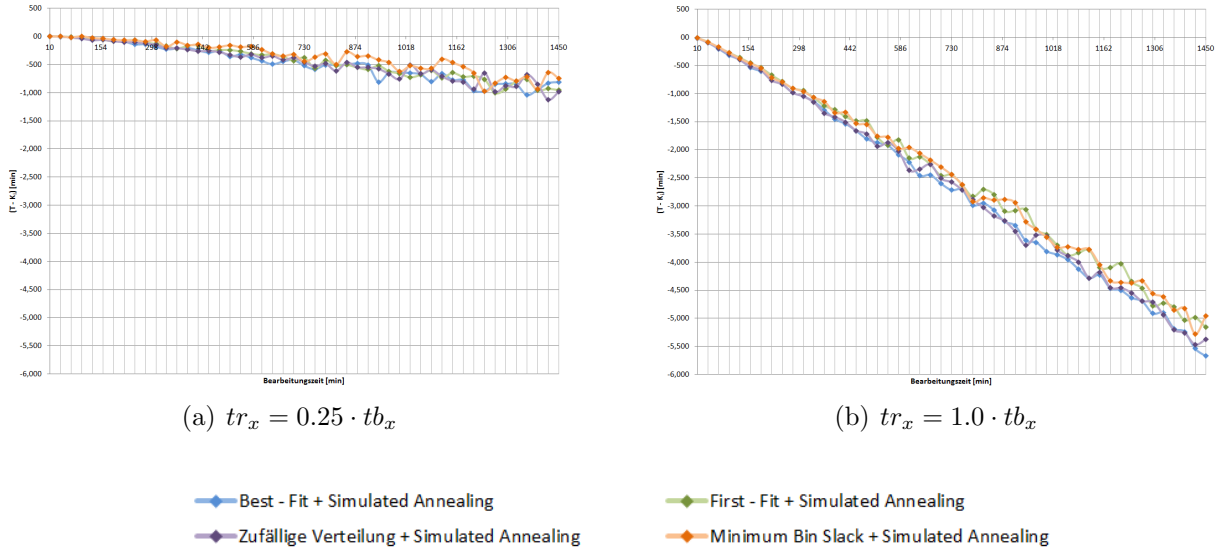
Weiters wurde festgestellt, dass die Anzahl der verfügbaren Produktionslinien keinen Einfluss auf die Qualität der erstellten Produktionsprogramme der einzelnen Approximationsverfahren hat. Darüber hinaus muss erwähnt werden, dass es bei kleineren Mengen $s = 3$ oder $s = 2$ an identischen Produktionslinien leichter ist, den Richtwert K_i dauerhaft zu unterschreiten.

Variation der Maschinen

Bei der Untersuchung von Aufgabenstellungen mit unterschiedlicher Maschinenanzahl hat sich herausgestellt, dass es kein spezielles Verfahren gibt, welches sich hinsichtlich der erstellten Produktionsprogramme signifikant vom Rest unterscheidet. Dabei gilt, je mehr Maschinen vorhanden sind, desto größer wird die Differenz $D = (\bar{T} - K_i)$.

Variation der Rüstzeit

Bei der Variation der Rüstzeiten wurde herausgefunden, dass die Best - Fit Methode und die zufällige Verteilung mit anschließender Lösungsraumuntersuchung, die sinnvollsten Kombinationen darstellen. Sie weisen gegenüber den anderen Verfahren eine um $\sim 2.6\%$ geringere Gesamtdurchlaufzeit auf. Außerdem kann der nachfolgenden Grafik entnommen werden, dass bei steigenden Rüstzeiten die berechneten Differenzen D immer kleiner werden, das bedeutet, dass die Gesamtdurchlaufzeiten weit unter dem angenäherten Richtwert, der Kapazität K_i, liegen.



(a) $tr_x = 0.25 \cdot tb_x$ (b) $tr_x = 1.0 \cdot tb_x$

Abbildung 10.5: Diese Grafik veranschaulicht die Tatsache, dass bei steigenden Rüstzeiten, die berechnete Differenz D immer kleiner wird.

Nachbarschaftssuche

Bei der Analyse der Nachbarschaftssuchverfahren hat sich ergeben, dass die Min - Max und die Random Methode meistens gleich gute Produktionsprogramme generieren. Da bei Aufgabenstellungen mit sehr hohen Rüstzeiten die *Random Nachbarschaftssuche* jedoch eindeutig bessere Ergebnisse liefert, wird diese bevorzugt angewandt. Dies kann darauf zurückgeführt werden, dass beim ausgewählten Verfahren bereits gezielt die Reihenfolge der Abarbeitungssequenz verändert wird.

Fazit Verbesserung

Zusammenfassend kann gesagt werden, dass im Zuge der Zuordnung und Verbesserung, der *Best - Fit* Algorithmus, sowie das Verfahren der *zufälligen Verteilung*, mit anschließender Anwendung von *Simulated Annealing* mit *Random Nachbarschaftssuche*, die besten Alternativen darstellen.

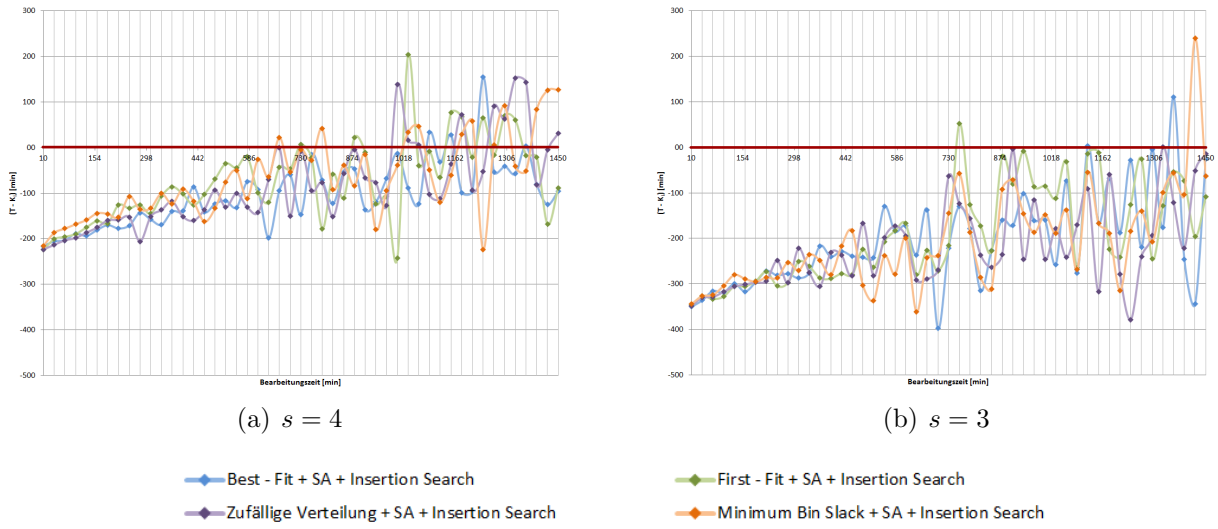
Im Mittel wurde durch die zusätzliche Anpassung der Auftragsverteilung eine Verbesserung von 7.1% gegenüber der Zuordnung erzielt.

Reihung

Die Verbesserung der Abarbeitungsreihenfolge der erstellten und angepassten Auftragsverteilung erfolgt mittels Insertion Search und genetischen Algorithmen. Zu diesem Zweck werden Produktionsprogramme mit Best - Fit, First - Fit, Minimum Bin Slack oder zufällig erzeugt. Diese Ausgangslösungen werden mittels Simulated Annealing unter Anwendung der Random Nachbarschaftssuche modifiziert. Alle möglichen Algorithmenkombinationen wurden getestet. Folgende Resultate konnten mit Hilfe der statistischen Analyse gewonnen werden.

Variation der Produktionslinien

Außerdem wurde festgestellt, dass die Anzahl der Produktionslinien keinen direkten Einfluss auf die Verfahrensauswahl hat. In diesem Zusammenhang kann also nicht eindeutig gesagt werden, welcher der implementierten Algorithmen zu bevorzugen ist. Allerdings hat sich auch bei dieser erweiterten Lösungsform gezeigt, dass es bei einer geringeren Anzahl an Produktionslinien $s = 3$ oder $s = 2$ einfacher ist, den Richtwert K_i dauerhaft zu unterschreiten.



(a) $s = 4$ (b) $s = 3$

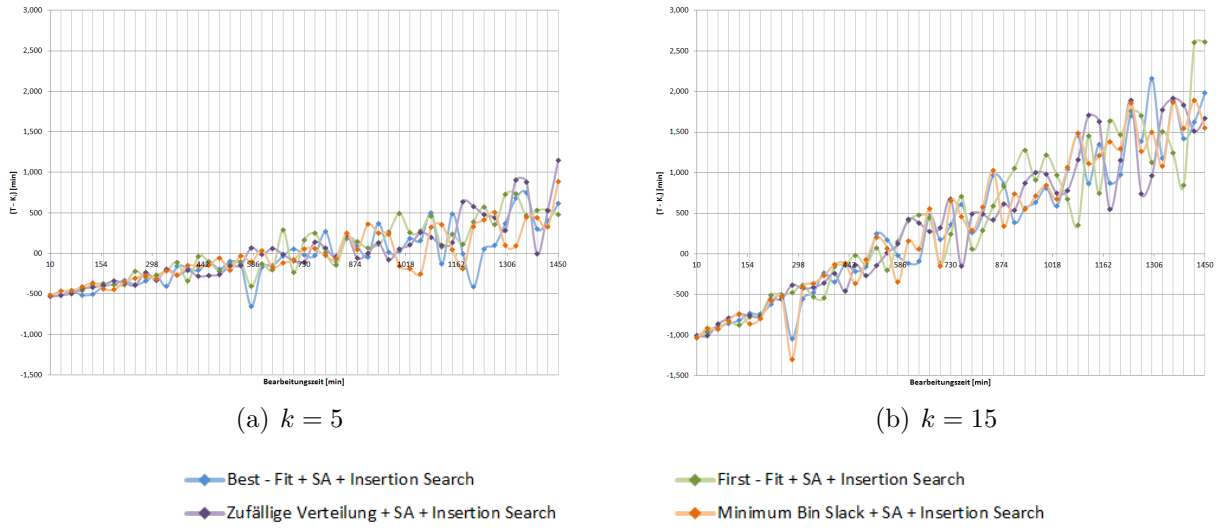
Abbildung 10.6: Diese Grafik zeigt, dass es bei sinkender Anzahl an vorhandenen Produktionslinien einfacher möglich ist, den angestrebten Richtwert K_i dauerhaft zu unterschreiten.

Variation der Aufträge

Bei der Analyse von Problemen mit variierender Auftragszahl fällt auf, dass die Qualität der Algorithmen die selbe Tendenz wie im Abschnitt Verbesserung, aufweisen. Bei einer geringen Anzahl an Aufträgen liefern alle Kombinationen aus Zuordnungsalgorithmus, Simulated Annealing und Insertion Search gleich gute Ergebnisse. Bei steigender Auftragszahl $n \geq 100$, erhöhen die Produktionsprogramme deren Ausgangslösung mittels Best - Fit oder zufällig initialisiert wurden, ihre Effizienz.

Variation der Maschinen

Bei der Untersuchung von Aufgabenstellungen mit unterschiedlicher Maschinenanzahl hat sich auch dieses Mal herausgestellt, dass es kein spezielles Verfahren gibt, welches sich hinsichtlich der erstellten Produktionsprogramme signifikant vom Rest unterscheidet. Dabei gilt, je mehr Maschinen vorhanden sind, desto größer wird die Differenz $D = (\bar{T} - K_i)$.



(a) $k = 5$ (b) $k = 15$

Abbildung 10.7: Diese Grafik zeigt, dass die Differenz D bei steigender Maschinenanzahl, größer wird.

Variation der Rüstzeit

Best - Fit und die zufällige Verteilung mit anschließender Lösungsraumuntersuchung und Reihenfolgeverbesserung haben sich als sinnvollste Kombinationen herausgestellt. Sie sparen gegenüber den anderen Methoden $\sim 2.1\%$ der Gesamtdurchlaufzeit ein. Außerdem kann bei diesem Testfall beobachtet werden, dass bei steigenden Rüstzeiten die berechneten Differenzen D immer kleiner werden. Das bedeutet, dass die Gesamtdurchlaufzeiten weit unter dem angenäherten Richtwert, der Kapazität K_i, liegen.

Fazit Reihung

Zusammenfassend kann gesagt werden, dass für die Kombination aus Zuordnung, Verbesserung und Reihung der *Best - Fit* Algorithmus sowie die *zufällig Verteilung* von Aufträgen auf die Produktionslinien mit anschließender Anwendung von *Simulated Annealing* und *Insertion Search* die beste Alternative darstellen.

Im Mittel wurde durch die zusätzliche Anpassung der Auftragsabarbeitungssequenz der verfügbaren Produktionslinien eine Optimierung von 2.9% gegenüber der Zuordnung mit anschließender Verbesserung der Auftragsverteilung erzielt.

Korrektur

Die abschließende Reduktion der Kapazitätslücken, die während der Erstellung und Verbesserung des Produktionsprogrammes und der Anpassung der Abarbeitungsreihenfolge entstanden sind, wird mit einer speziellen Prozedur verwirklicht. Zu diesem Zweck werden Produktionsprogramme mit Best - Fit, First - Fit, Minimum Bin Slack oder zufällig erzeugt. Diese Ausgangslösungen werden mittels Simulated Annealing unter Anwendung der Random Nachbarschaftssuche modifiziert. Zuletzt wird die Abarbeitungsreihenfolge mit Insertion Search verbessert. Alle möglichen Algorithmenkombinationen wurden getestet. Folgende Resultate konnten mit Hilfe der statistischen Analyse gewonnen werden.

Variation der Aufträge

Bei der Auswertung der Testfälle mit unterschiedlichen Auftragszahlen hat sich herauskristallisiert, dass es kein Verfahren gibt, welches durch signifikant bessere Produktionsprogramme hervorsticht. Es kann vermerkt werden, dass es eine ganz leichte Tendenz zum Best - Fit Verfahren gibt, die mittels t - Test aber nicht bewiesen werden kann.

Variation der Produktionslinien

Außerdem wurde festgestellt, dass die Anzahl der verfügbaren Produktionslinien keinen Einfluss auf die Qualität der erstellten Produktionsprogramme der einzelnen Approximationsverfahren hat. Dies ist keine Überraschung, da bereits die Auswertung der Zuordnungs-, Verbesserungs- und Reihenfolgealgorithmen die selben Erkenntnisse geliefert haben. Darüber hinaus muss auch hier erwähnt werden, dass es bei kleineren Mengen $s = 3$ oder $s = 2$ an identischen Produktionslinien leichter ist, den vorgegebenen Richtwert K_i dauerhaft zu unterschreiten.

Variation der Maschinen

Bei der Untersuchung von Aufgabenstellungen mit steigender Maschinenanzahl hat sich herausgestellt, dass die Anwendung des Korrekturverfahrens fast nie in einer Verbesserung der Auftragsverteilung resultiert. Daher gilt auch in diesem Fall, dass es kein spezielles Verfahren gibt, welches sich hinsichtlich der erstellten Produktionsprogramme signifikant vom Rest unterscheidet.

Variation der Rüstzeit

Bei der Variation der Rüstzeiten hat sich ergeben, dass es keine Verfahrenskombination gibt, die eindeutig das beste Ergebnis liefert.

Fazit Korrektur

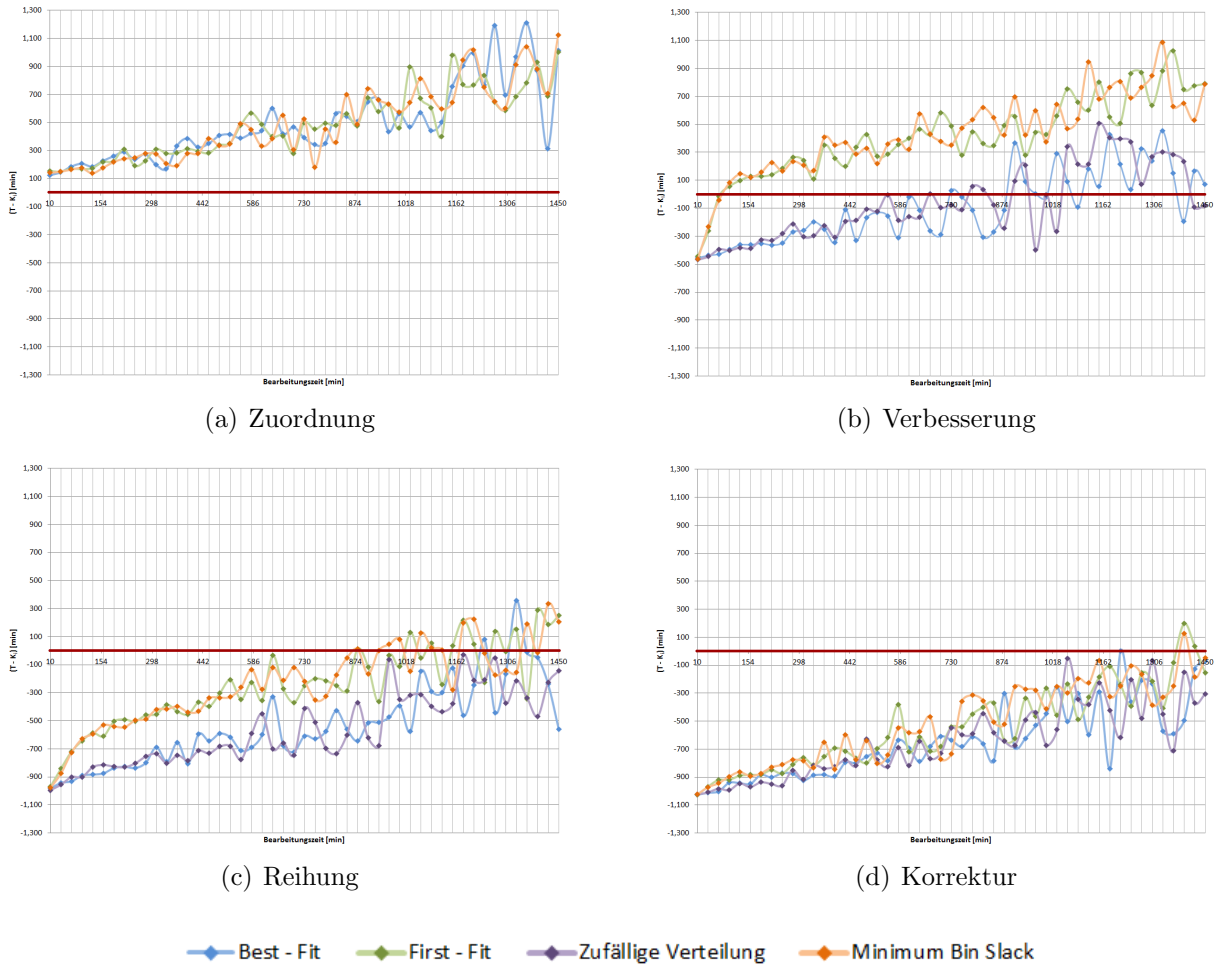
Zusammenfassend muss leider gesagt werden, dass bei keinem der analysierten Testfälle eine klare Tendenz zu einer konkreten Algorithmenkombination zu erkennen war.

Im Mittel allerdings wurde durch den nachträglichen Korrekturschritt eine Verbesserung von 0.9% gegenüber der Reihung erzielt.

10.6 Schlusswort

Die Evaluierung der verschiedenen Algorithmen anhand der definierten Testfälle hat ergeben, dass im Zuge der Erstellung eines effizienten Produktionsprogrammes immer die gesamte Prozesskette von der Zuordnung, über die Verbesserung, bis hin zur Reihung und abschließender Korrektur, durchlaufen werden muss. Dies ist wichtig, da in jeder Stufe eine signifikante Reduktion der maximalen Gesamtdurchlaufzeit zu verzeichnen ist.

Im Detail bedeutet das, dass die maximale Gesamtdurchlaufzeit des erstellten Produktionsprogrammes während des Simulationsprozesses im Durchschnitt um $\sim 10.9\%$ gesenkt werden kann.



(a) Zuordnung

(b) Verbesserung

(c) Reihung

(d) Korrektur

Abbildung 10.8: Anhand dieser Grafik wird aufgezeigt, dass jede einzelne Prozessstufe eine Reduktion der maximalen Gesamtdurchlaufzeit des erstellten Produktionsprogrammes bewirkt.

Hier noch einmal eine kurze Zusammenfassung der wichtigsten Erkenntnisse.

Bei der Analyse der einzelnen Verfahren, die im Zuordnungsschritt für die Erstellung einer Ausgangslösung verwendet werden, hat sich herausgestellt, dass die Wahl der besten Methode sehr stark von den restlichen Prozessschritten abhängig ist.

Zur Erinnerung: Nach der Zuordnung galt der First - Fit Algorithmus, als das geeignetste Verfahren. Nach Abschluss der Verbesserung und der Anpassung der Abarbeitungsreihenfolge waren die Best - Fit Variante und die zufällige Verteilung eindeutig die besten Alternativen. Nach der Anwendung der Korrekturprozedur hat sich ergeben, dass alle verwendeten Algorithmenkombinationen nahezu identische Produktionsprogramme hinsichtlich ihrer Gesamtdurchlaufzeit liefern, wobei die Verfahrenskonstellation, bei der mit Hilfe der Best - Fit Methode eine Ausgangslösung erzeugt wird, leicht zu bevorzugen ist. Bei der endgültigen Empfehlung einer Methode die immer im Zusammenhang mit den nachfolgenden Prozessschritten betrachtet werden muss, können zwei mögliche Fälle unterschieden werden:

- Muss aufgrund von Zeitmangel ein konkreter Algorithmus für die Erstellung einer Basislösung ausgewählt werden, dann fällt die Entscheidung auf die **Best - Fit** Methode.
- Ist hingegen genügend Zeit ($\sim$ 40 Sekunden) vorhanden, sollte die Simulation mit unterschiedlichen Ausgangslösungen, die mittels **Best - Fit, First - Fit, Minimum Bin Slack** und durch **zufällige Verteilung** erzeugt werden, durchgeführt werden.

Im Zuge der Auswertung der Verbesserungsverfahren hat sich herauskristallisiert, dass **Simulated Annealing** zusammen mit der **Random Nachbarschaftssuche** die beste Alternative darstellt. Diese Methodenkombination hat sich eindeutig gegen Simulated Annealing unter Anwendung der Min - Max Nachbarschaftssuche, sowie die verschiedensten Varianten der genetischen Algorithmen durchgesetzt.

Bei der Auswertung der Testfälle wurde klar, dass es sich bei **Insertion Search** um das Verfahren handelt, mit welchem die Auftragsreihenfolgen der einzelnen Produktionslinien bestmöglich verbessert werden können. Die genetischen Algorithmen, die auf Grund der Laufzeitbeschränkung nicht optimal eingesetzt werden können und entsprechend schlechte Ergebnisse liefern, konnten mit den sehr guten Produktionsprogrammen von Insertion Search nicht annähernd mithalten.

Den Abschluss bildet der **Korrekturalgorithmus**, der unter allen Umständen angewandt werden muss.

Unter Einbeziehung der beschriebenen Testfälle und aller Algorithmenkombinationen hat sich ergeben, dass mit den implementierten Verfahren, Produktionsprogramme generiert werden, die in $\sim$ 80% der Fälle unter dem berechneten Richtwert $K_i = \frac{1}{s}\sum_{x=1}^{n} t_x$, wobei $t_x = bE_x + rM_x$, liegen. Dieser Prozentsatz erlaubt es zu sagen, dass es mit Hilfe der implementierten Approximationsverfahren, Verbesserungs- und Suchmethoden, sowie der Korrekturprozedur gelungen ist, sehr gute Ergebnisse zu erzielen.

10.7 Ausblick

Diese Diplomarbeit hat einige Erweiterungspotentiale im Bezug auf die Problemstellung, sowie die Lösungsimplementierung hervorgebracht, die nun kurz angeführt werden.

- Zuerst muss erwähnt werden, dass das behandelte Problem sehr allgemein gehalten wurde. Das bedeutet, dass keine konkreten Informationen zu Bearbeitungs- und Rüstzeiten, sowie zu den verwendeten Maschinen verfügbar sind. Wären detaillierte Informationen zum Unternehmen oder zum Einsatzgebiet vorhanden, wäre es sehr leicht möglich, die Verfahren besser auf die Aufgabenstellung abzustimmen.
- Auf Grund des riesigen Potentials von genetischen Algorithmen sollten diese im Zusammenhang mit einem mehrere Maschinen Scheduling- und einem Reihenfolgeproblem genauer erforscht und besser auf die Problemstellung angepasst werden.
- Weiters wäre es sehr interessant zu testen, wie sich die beschriebenen Verfahren bei einem Problem verhalten, bei welchem es jederzeit möglich ist, neue Aufträge in das System einzubuchen.
- Eine Erweiterung der Simulation um geplante sowie ungeplante Wartungs- und Instandhaltungsmaßnahmen würde die Aufgabenstellung noch realistischer gestalten.
- Außerdem wäre es sicher wünschenswert die Eigenschaften eines jeden Auftrags um gewisse Restriktionen, wie zum Beispiel den Liefertermin, zu erweitern.

Abschließend soll gesagt werden, dass dieses sowie alle weiteren Zuordnungs- und Reihenfolgeprobleme wirklich sehr interessante Themen sind, die genügend Forschungsbedarf und Verbesserungspotentiale aufweisen. Das Wichtigste dabei ist, dass diese Art an Aufgabenstellungen für jedes produzierende Unternehmen von größter Bedeutung und praktischer Relevanz sind.

Anhang

Anhang A

Testdatensätze und Diagramme

An dieser Stelle sollen exemplarisch einige der über ~ 1200 behandelten Testfälle angeführt werden. Im Konkreten werden die berechneten Gesamtdurchlaufzeiten, die den Diagrammen zugrunde liegen, aufgelistet. Außerdem wird beispielhaft die Anwendung des t - *Tests* erläutert.

A.1 Zuordnung: Anzahl der Aufträge $n = 10$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 67 näher:

Bearbeitungszeit $tb_x \in [10, 1450]$ Minuten
Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 1$
Anzahl der Aufträge $n = 10$

Bearbeitungs -zeiten tb_x [min]	Richtwert K_i [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie					
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
10	88.9	81.6	-7.3	85.3	-3.6	81.0	-7.9
40	125.9	125.7	-0.2	124.9	-1.0	119.2	-6.7
70	163.8	169.0	5.2	165.8	2.1	156.1	-7.6
100	200.9	205.9	5.0	208.1	7.2	191.9	-9.1
130	239.2	248.0	8.8	240.3	1.0	230.3	-9.0
160	276.9	282.7	5.8	283.6	6.8	271.8	-5.0
190	315.9	311.8	-4.1	327.0	11.1	309.2	-6.8

220	351.7	380.3	28.6	360.8	9.1	354.5	2.8
250	385.8	393.1	7.3	392.0	6.2	384.4	-1.4
280	423.7	427.9	4.2	440.8	17.1	426.2	2.5
310	463.8	479.0	15.2	480.0	16.2	470.2	6.4
340	500.3	527.1	26.7	540.9	40.5	495.0	-5.4
370	538.5	575.4	36.9	581.0	42.5	537.3	-1.2
400	577.8	608.3	30.5	639.1	61.3	567.8	-9.9
430	612.8	674.0	61.2	639.1	26.4	632.4	19.6
460	653.7	688.3	34.6	710.1	56.4	663.1	9.4
490	688.3	705.8	17.6	751.9	63.6	698.0	9.8
520	724.9	792.7	67.8	805.9	81.0	742.9	17.9
550	764.3	781.9	17.5	835.0	70.7	771.7	7.4
580	802.4	851.6	49.2	861.3	58.9	822.9	20.6
610	831.6	916.2	84.6	897.4	65.8	852.8	21.1
640	869.9	951.9	82.0	938.5	68.6	890.5	20.6
670	914.9	968.6	53.8	974.3	59.4	928.7	13.8
700	948.1	1030.6	82.5	1038.2	90.1	980.1	32.0
730	989.0	1031.0	42.0	1067.2	78.2	998.8	9.8
760	1038.1	1106.1	68.0	1089.7	51.6	1053.4	15.4
790	1065.4	1139.7	74.2	1165.5	100.0	1098.4	32.9
820	1094.9	1153.2	58.3	1158.8	63.9	1120.9	26.0
850	1130.4	1227.3	96.8	1314.4	184.0	1146.6	16.2
880	1178.2	1251.7	73.5	1268.4	90.2	1208.7	30.6
910	1213.5	1258.6	45.2	1338.7	125.2	1237.8	24.3
940	1246.6	1337.8	91.3	1391.6	145.0	1296.5	49.9
970	1280.5	1341.6	61.1	1414.7	134.2	1333.8	53.3
1000	1323.0	1429.3	106.3	1390.0	66.9	1364.7	41.7
1030	1370.8	1481.4	110.6	1549.2	178.4	1413.9	43.1
1060	1415.3	1512.6	97.3	1542.2	127.0	1426.9	11.6
1090	1439.1	1516.3	77.1	1568.3	129.2	1480.0	40.9
1120	1468.1	1602.4	134.2	1633.0	164.9	1526.9	58.8
1180	1550.4	1683.5	133.1	1786.2	235.9	1611.7	61.4
1210	1599.9	1673.0	73.0	1749.6	149.7	1619.3	19.3
1240	1626.1	1761.8	135.7	1822.5	196.3	1678.1	52.0

1270	1663.5	1828.5	165.0	1839.3	175.8	1694.8	31.3
1300	1696.7	1838.1	141.4	1924.8	228.1	1762.8	66.1
1330	1737.9	1905.0	167.1	1885.3	147.4	1802.4	64.5
1360	1785.1	1899.4	114.2	2003.8	218.7	1856.4	71.3
1390	1814.3	1885.5	71.2	2002.8	188.5	1869.7	55.4
1420	1844.6	1952.1	107.5	2046.5	201.9	1909.6	65.0
1450	1901.4	2030.0	128.6	2073.1	171.7	1903.1	1.7

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

2479.0	2199.0	2553.0	2353.0	2315.0	2153.0	2309.0	1355.0
1573.0	2791.0	2583.0	2471.0	1250.0	2164.0	2437.0	2214.0
1394.0	1852.0	1536.0	2557.0	2521.0	1644.0	2113.0	1811.0
1194.0	2569.0	2034.0	2434.0	2299.0	2657.0	1782.0	1898.0
2183.0	1463.0	2267.0	2836.0	2337.0	2148.0	1337.0	2972.0
1662.0	1773.0	1698.0	1836.0	1644.0	2440.0	1461.0	2141.0
2272.0	1692.0	Testdatensatz First - Fit					

(b) Datensätze der 50 Testiterationen des First - Fit Verfahrens für den markierten Testfall $tb_x = [10, 1450]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

1598.0	1477.0	2567.0	2519.0	1367.0	1470.0	2398.0	1393.0
2089.0	1811.0	1938.0	2079.0	1697.0	1949.0	1381.0	1574.0
1570.0	2311.0	1628.0	1861.0	1781.0	1766.0	1899.0	1460.0
1563.0	2638.0	2457.0	2321.0	2233.0	1980.0	2069.0	1863.0
2702.0	1517.0	1468.0	1965.0	2624.0	1665.0	1612.0	1474.0
1676.0	2081.0	1649.0	2460.0	1591.0	1994.0	2285.0	1783.0
1955.0	1946.0	Testdatensatz Minimum Bin Slack					

(c) Datensätze der 50 Testiterationen des Minimum Bin Slack Verfahrens für den markierten Testfall $tb_x = [10, 1450]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.1: Datensätze - Zuordnung: Anzahl der Aufträge $n = 10$

t - Test

Es werden die Mittelwerte des First - Fit $\bar{x}_1$ und des Minimum Bin Slack Algorithmus $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{2073.1 - 1903.1}{\sqrt{453.1^2 + 377.8^2}} = 2.04$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t \not< c \rightarrow H_0$ wird verworfen, d.h. der Mittelwert $\bar{x}_1$ ist signifikant größer als der Mittelwert $\bar{x}_2$!

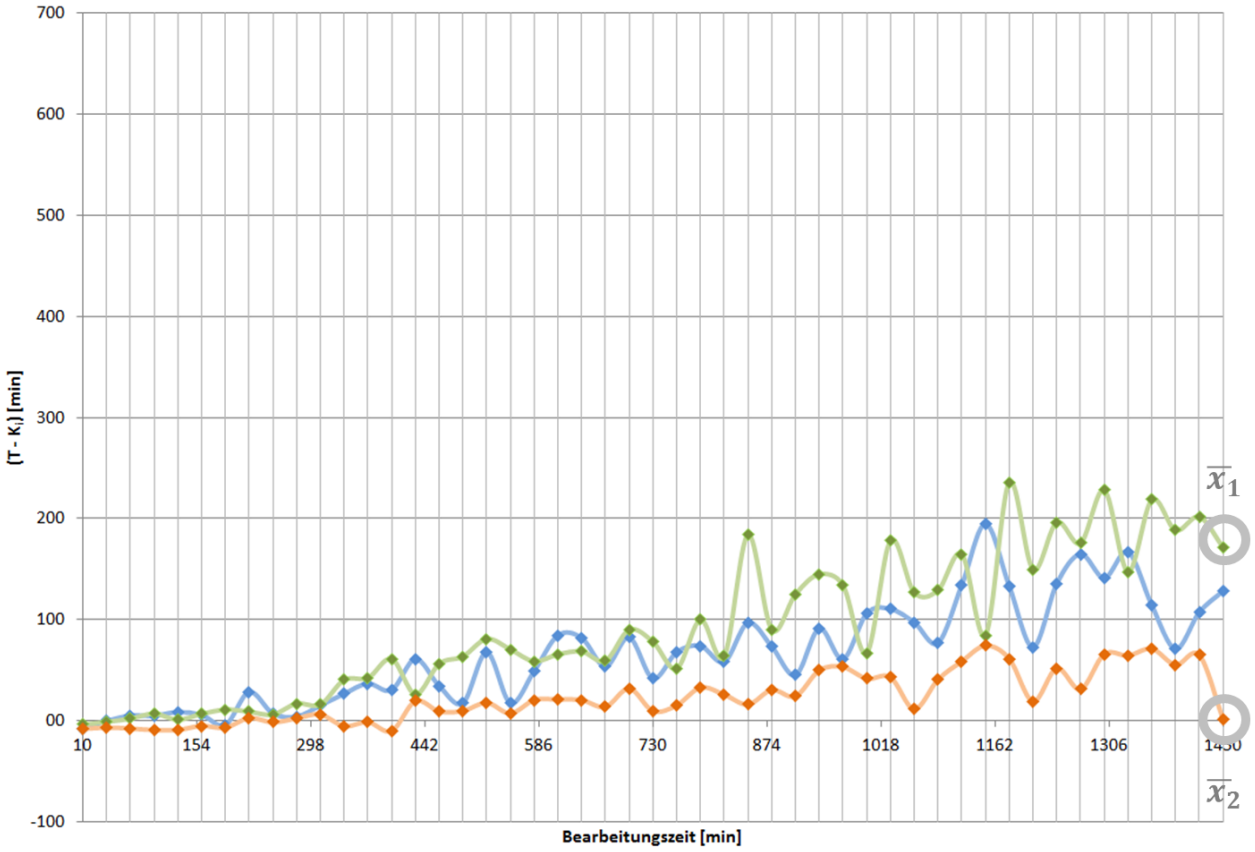


Abbildung A.2: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 67 abgebildet wurde.

A.2 Zuordnung: Anzahl der Aufträge $n = 100$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 67 näher:

Bearbeitungszeit $tb_x \in [10, 1450]$ Minuten
Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 1$
Anzahl der Aufträge $n = 100$

Bearbeitungs -zeiten tb_x [min]	Richtwert K_i [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie					
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
10	888.7	965.9	77.2	969.4	80.7	951.9	63.3
40	1262.1	1351.8	89.7	1356.0	93.9	1344.5	82.4
70	1638.4	1712.8	74.4	1750.6	112.2	1736.5	98.0
100	2012.9	2136.8	123.8	2101.8	88.8	2113.3	100.4
130	2389.8	2483.7	93.9	2493.4	103.6	2504.3	114.5
160	2767.5	2877.2	109.7	2870.6	103.1	2889.0	121.5
190	3150.6	3300.0	149.4	3247.2	96.6	3283.0	132.4
220	3527.9	3700.1	172.2	3588.7	60.9	3627.8	100.0
250	3897.9	4025.1	127.3	4060.3	162.5	4026.5	128.7
280	4262.3	4401.7	139.4	4406.4	144.1	4387.5	125.2
310	4647.7	4751.8	104.0	4793.8	146.0	4828.9	181.1
340	5022.5	5179.7	157.2	5179.1	156.6	5179.7	157.2
370	5393.4	5595.6	202.2	5560.9	167.5	5514.1	120.7
400	5761.7	5889.8	128.2	5926.8	165.1	5885.0	123.3
430	6144.3	6345.9	201.6	6308.9	164.6	6333.1	188.8
460	6526.8	6698.7	171.9	6762.4	235.6	6664.7	137.9
490	6889.0	7079.8	190.8	7160.1	271.1	7116.6	227.6
520	7273.7	7438.0	164.3	7450.1	176.4	7416.2	142.5
550	7647.0	7835.1	188.2	7733.0	86.0	7812.2	165.2
580	8017.7	8257.5	239.9	8209.5	191.9	8264.7	247.1
610	8402.0	8566.6	164.6	8703.5	301.5	8619.7	217.7
640	8777.6	8989.3	211.7	8986.6	209.0	9177.4	399.9
670	9143.3	9457.6	314.2	9378.7	235.3	9433.9	290.5

700	9525.0	9728.5	203.5	9688.3	163.3	9873.4	348.4
730	9896.0	10115.9	219.8	10072.9	176.9	10201.4	305.4
760	10262.4	10496.7	234.4	10481.9	219.5	10432.0	169.6
790	10671.5	10974.9	303.4	10895.1	223.6	11045.5	374.0
820	11036.8	11202.6	165.8	11336.0	299.3	11429.6	392.8
850	11445.4	11826.5	381.2	11709.5	264.2	11728.7	283.3
880	11763.3	11975.2	211.9	12235.9	472.6	12111.5	348.2
910	12103.2	12321.1	217.9	12370.5	267.3	12393.8	290.6
940	12517.4	12575.7	58.3	12636.0	118.6	12782.1	264.7
970	12895.4	13209.6	314.2	13299.6	404.2	13307.5	412.1
1000	13255.4	13785.5	530.1	13659.2	403.8	13465.5	210.2
1030	13633.7	14013.8	380.1	14010.4	376.7	13977.5	343.8
1060	14030.9	14411.8	380.9	14368.3	337.4	14362.7	331.8
1090	14458.0	14608.9	150.9	14739.9	281.8	14884.2	426.2
1120	14785.5	15096.1	310.6	15182.5	396.9	15143.5	357.9
1150	15121.7	15416.2	294.5	15483.8	362.1	15454.8	333.2
1180	15559.7	15993.9	434.2	16088.8	529.1	15689.7	130.0
1210	15867.8	16174.3	306.5	16327.9	460.1	16280.5	412.7
1240	16249.7	16489.9	240.1	16749.0	499.3	16874.5	624.7
1270	16700.0	17151.1	451.1	16708.7	8.7	16996.4	296.4
1300	17062.4	17610.2	547.7	17485.7	423.3	17378.3	315.9
1330	17415.3	17903.6	488.3	17713.3	298.0	17917.8	502.5
1360	17784.8	17897.9	113.1	18310.3	525.5	18133.2	348.5
1390	18229.1	18666.5	437.4	18245.3	16.2	18611.2	382.1
1420	18535.3	18807.0	271.7	18845.4	310.1	18900.5	365.2
1450	18940.8	19391.1	450.3	19018.8	78.0	19010.3	69.5

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

19482.0	20679.0	19684.0	18293.0	19933.0	17193.0	18930.0	18219.0
20178.0	19590.0	20259.0	17026.0	18712.0	17729.0	18970.0	19938.0
18379.0	20218.0	17834.0	20691.0	16466.0	18840.0	21099.0	18238.0
19450.0	19323.0	19429.0	19352.0	19667.0	18491.0	20629.0	17999.0
19352.0	18276.0	18880.0	21174.0	18416.0	19096.0	18848.0	17808.0
19170.0	18619.0	19976.0	18832.0	19200.0	18765.0	17842.0	18164.0
19342.0	18258.0	**Testdatensatz First - Fit**					

(b) Datensätze der 50 Testiterationen des First - Fit Verfahrens für den markierten Testfall $tb_x = [10, 1450]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

20154.0	17624.0	18849.0	19744.0	18312.0	19926.0	19761.0	19634.0
18817.0	19264.0	18068.0	17447.0	19775.0	19969.0	18086.0	18629.0
18809.0	19503.0	17207.0	17530.0	19225.0	20523.0	18969.0	19150.0
18048.0	20599.0	19821.0	21083.0	20007.0	18497.0	18720.0	17773.0
19327.0	19394.0	18335.0	18105.0	19668.0	18950.0	18428.0	19830.0
17225.0	18078.0	18912.0	20326.0	19310.0	19338.0	19982.0	17696.0
18908.0	19181.0	Testdatensatz Minimum Bin Slack					

(c) Datensätze der 50 Testiterationen des Minimum Bin Slack Verfahrens für den markierten Testfall $tb_x = [10, 1450]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.3: Datensätze - Zuordnung: Anzahl der Aufträge $n = 100$

t - Test

Es werden die Mittelwerte des First - Fit $\bar{x}_1$ und des Minimum Bin Slack Algorithmus $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{19018.8 - 19010.3}{\sqrt{1032.0^2 + 937.2^2}} = 0.04$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t < c \rightarrow H_0$ wird nicht verworfen, d.h. der Mittelwert $\bar{x}_1$ unterscheidet sich nicht signifikant vom Mittelwert $\bar{x}_2$!

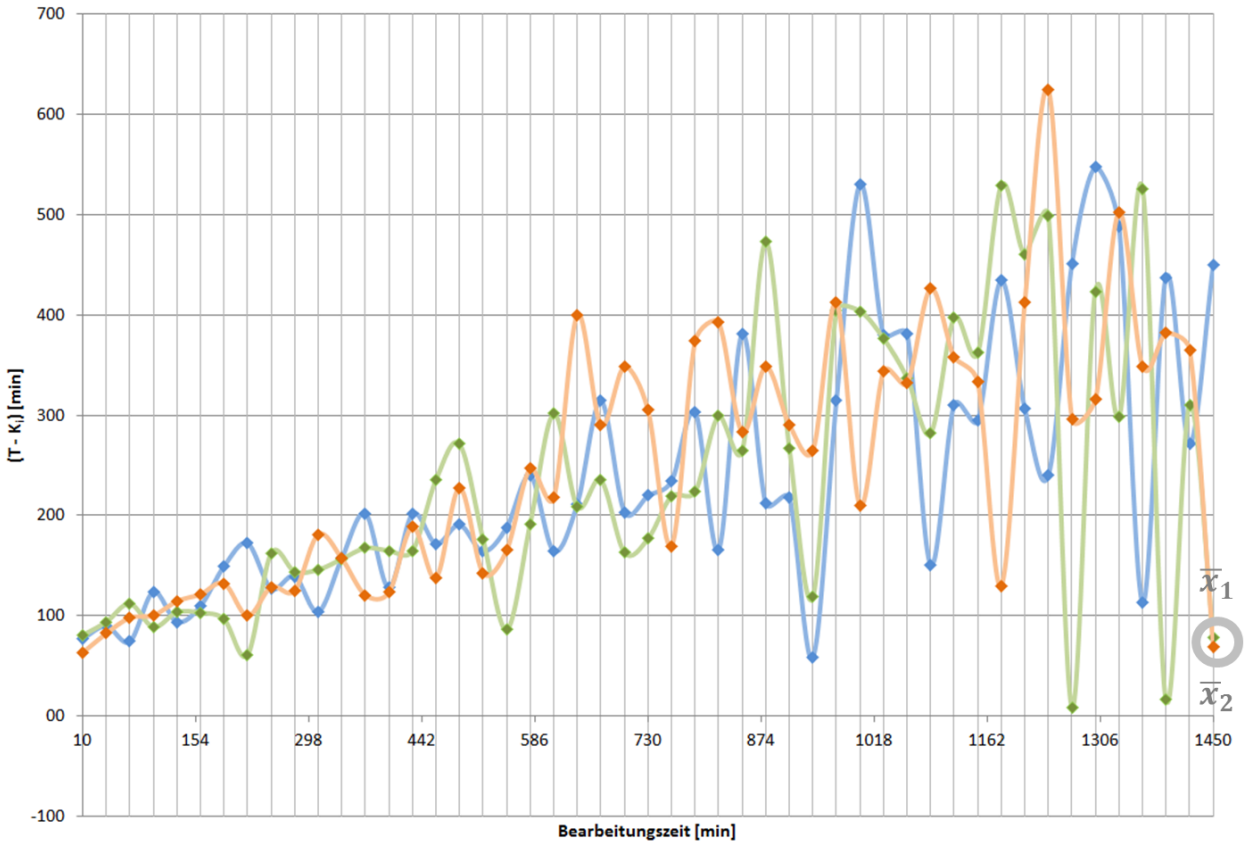


Abbildung A.4: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 67 abgebildet wurde.

A.3 Optimierung: Rüstzeit $tr_x = 0.25 \cdot tb_x$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 70 näher:

Bearbeitungszeit $tb_x \in [10, 40]$ Minuten
Rüstzeit $tr_x = 0.25 \cdot tb_x$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 1$
Anzahl der Aufträge $n = 50$

Bearbeitungs -zeiten tb_x [min]	Richtwert Ki [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie							
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	RandomFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
		Random	Random	Random	Random	Random	Random	Random	Random
10	81.1	81.2	0.1	82.5	1.4	83.4	2.3	81.9	0.8
40	319.3	308.7	-10.7	312.1	-7.3	316.9	-2.4	319.6	0.2
70	550.3	531.8	-18.6	541.3	-9.1	534.5	-15.8	533.4	-17.0
100	784.7	746.8	-37.9	774.1	-10.6	745.7	-39.0	788.5	3.8
130	1022.1	976.9	-45.2	982.5	-39.6	955.3	-66.8	994.4	-27.7
160	1251.5	1198.2	-53.3	1193.7	-57.8	1191.2	-60.2	1213.7	-37.7
190	1488.7	1408.1	-80.6	1402.9	-85.8	1401.3	-87.4	1429.6	-59.1
220	1725.2	1634.6	-90.5	1653.4	-71.8	1624.2	-101.0	1657.1	-68.0
250	1956.3	1814.4	-141.9	1892.2	-64.1	1853.4	-103.0	1886.6	-69.8
280	2193.9	2055.1	-138.8	2092.8	-101.1	2068.4	-125.5	2104.0	-90.0
310	2422.6	2238.0	-184.5	2286.6	-136.0	2275.0	-147.6	2354.4	-68.1
340	2659.0	2431.1	-228.0	2487.6	-171.5	2449.7	-209.3	2478.5	-180.6
370	2898.0	2684.5	-213.5	2678.0	-220.0	2689.8	-208.2	2797.1	-100.9
400	3122.2	2919.6	-202.6	2956.6	-165.7	2889.8	-232.4	2966.6	-155.6
430	3363.7	3115.0	-248.7	3205.8	-157.9	3102.0	-261.7	3218.4	-145.3
460	3597.3	3311.2	-286.2	3345.0	-252.3	3341.8	-255.5	3395.6	-201.7
490	3833.5	3573.6	-259.9	3592.7	-240.8	3554.8	-278.6	3646.5	-186.9
520	4060.4	3702.9	-357.5	3819.9	-240.4	3731.1	-329.3	3903.7	-156.7
550	4313.5	3993.2	-320.3	4046.3	-267.3	3949.9	-363.7	4123.6	-189.9
580	4541.8	4161.9	-379.9	4233.5	-308.3	4219.0	-322.7	4364.1	-177.6
610	4781.5	4344.8	-436.7	4450.9	-330.5	4402.1	-379.4	4546.5	-234.9
640	5017.1	4522.4	-494.7	4701.2	-315.9	4670.2	-346.9	4704.0	-313.0
670	5235.0	4789.7	-445.3	4876.2	-358.8	4817.0	-418.1	4883.5	-351.5
700	5474.2	5056.4	-417.8	5041.8	-432.4	5092.6	-381.6	5154.2	-320.0
730	5710.7	5188.4	-522.3	5335.1	-375.7	5252.3	-458.5	5266.3	-444.4
760	5963.2	5382.1	-581.2	5393.9	-569.4	5434.7	-528.6	5592.8	-370.4
790	6178.1	5672.4	-505.7	5752.7	-425.3	5692.6	-485.4	5864.3	-313.8
820	6399.0	5890.2	-508.8	5881.5	-517.5	5786.5	-612.6	5901.6	-497.4
850	6634.5	6133.1	-501.5	6130.3	-504.2	6171.9	-462.7	6357.5	-277.0
880	6878.0	6400.9	-477.1	6325.3	-552.7	6330.5	-547.5	6521.8	-356.2
910	7112.9	6604.3	-508.6	6525.1	-587.8	6562.0	-550.9	6768.2	-344.7
940	7340.1	6529.8	-810.2	6818.8	-521.2	6762.9	-577.1	6919.8	-420.2
970	7604.2	6966.0	-638.2	6983.7	-620.5	6936.5	-667.6	7141.7	-462.5
1000	7841.8	7196.4	-645.4	7183.5	-658.3	7082.9	-758.9	7219.8	-622.1
1030	8043.5	7395.2	-648.3	7316.7	-726.8	7536.7	-506.7	7519.5	-524.0
1060	8286.7	7610.6	-676.2	7606.8	-679.9	7622.6	-664.1	7721.8	-565.0
1090	8527.2	7727.1	-800.1	7921.5	-605.7	7935.1	-592.1	7956.0	-571.2

1120	8768.0	8105.8	-662.2	8035.3	-732.7	8054.5	-713.5	8366.1	-401.9
1150	8973.6	8197.1	-776.5	8326.5	-647.1	8182.7	-790.9	8506.8	-466.8
1180	9227.0	8452.8	-774.2	8509.7	-717.3	8417.7	-809.4	8687.3	-539.7
1210	9451.4	8489.6	-961.8	8745.0	-706.4	8510.6	-940.7	8798.9	-652.5
1240	9728.8	8755.5	-973.4	8966.3	-762.6	9076.7	-652.1	8755.4	-973.4
1270	9964.7	9111.1	-853.6	8959.2	-1005.4	8982.5	-982.1	9133.4	-831.2
1300	10154.5	9310.5	-843.9	9216.1	-938.4	9278.8	-875.7	9422.9	-731.6
1330	10408.9	9557.2	-851.8	9610.5	-798.5	9515.7	-893.3	9618.4	-790.6
1360	10585.3	9543.5	-1041.8	9821.0	-764.3	9905.3	-680.0	9863.3	-722.0
1390	10863.7	9916.0	-947.7	9909.5	-954.2	10016.6	-847.2	9923.9	-939.9
1420	11105.8	10274.9	-830.8	10174.9	-930.9	9981.7	-1124.1	10464.3	-641.4
1450	11351.1	10541.4	-809.7	10397.4	-953.7	10364.3	-986.8	10606.6	-744.5

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

306.0	257.0	314.0	295.0	305.0	271.0	293.0	309.0
316.0	324.0	344.0	280.0	310.0	335.0	323.0	285.0
296.0	286.0	287.0	351.0	297.0	312.0	352.0	294.0
316.0	309.0	323.0	306.0	302.0	282.0	338.0	312.0
305.0	317.0	298.0	306.0	335.0	305.0	317.0	276.0
363.0	302.0	335.0	324.0	298.0	321.0	333.0	322.0
280.0	267.0	**Testdatensatz Best - Fit**					

(b) Datensätze der 50 Testiterationen des Best - Fit Verfahrens für den markierten Testfall $tb_x = [10, 40]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

319.0	304.0	316.0	317.0	308.0	319.0	323.0	254.0
345.0	294.0	286.0	320.0	335.0	293.0	329.0	304.0
332.0	308.0	286.0	327.0	357.0	281.0	302.0	329.0
307.0	329.0	309.0	301.0	295.0	276.0	285.0	333.0
338.0	304.0	313.0	339.0	298.0	321.0	317.0	288.0
320.0	331.0	325.0	300.0	341.0	319.0	329.0	319.0
289.0	290.0	**Testdatensatz First - Fit**					

(c) Datensätze der 50 Testiterationen des First - Fit Verfahrens für den markierten Testfall $tb_x = [10, 40]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.5: Datensätze - Optimierung: Rüstzeit $tr_x = 0.25 \cdot tb_x$

t - Test

Es werden die Mittelwerte des Best - Fit $\bar{x}_1$ und des First - Fit Algorithmus $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_2 > \mu_1$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{312.1 - 308.7}{\sqrt{20.4^2 + 22.6^2}} = 0.79$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t < c \rightarrow H_0$ wird nicht verworfen!

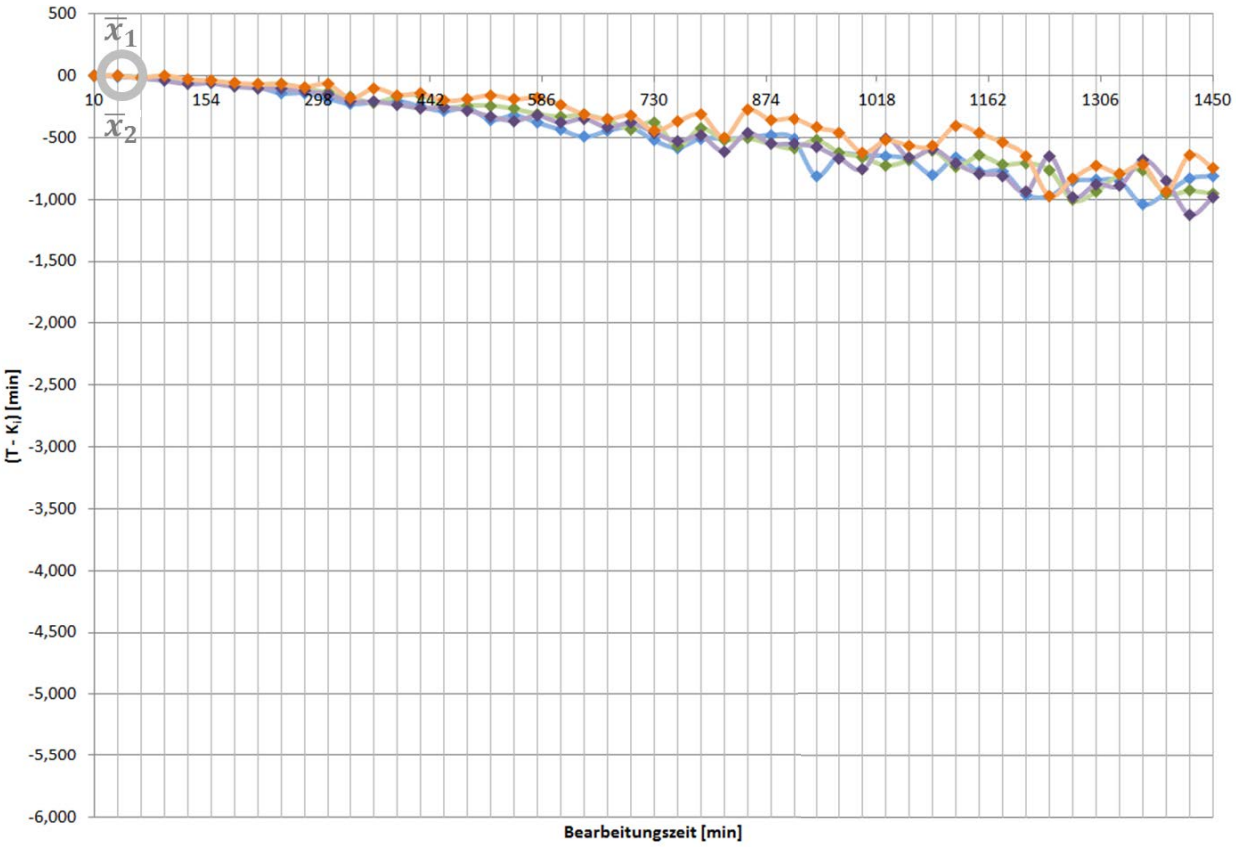


Abbildung A.6: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 70 abgebildet wurde.

A.4 Optimierung: Rüstzeit $tr_x = 1.00 \cdot tb_x$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 70 näher:

Bearbeitungszeit $tb_x \in [10, 610]$ Minuten
Rüstzeit $tr_x = 1.00 \cdot tb_x$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 1$
Anzahl der Aufträge $n = 50$

Bearbeitungs -zeiten tb_x [min]	Richtwert Ki [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie							
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	RandomFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
		Random	Random	Random	Random	Random	Random	Random	Random
10	131.4	117.9	-13.5	119.8	-11.6	118.7	-12.7	119.4	-12.0
40	506.5	412.1	-94.4	426.7	-79.8	414.0	-92.4	422.8	-83.7
70	884.1	681.2	-202.9	704.2	-180.0	689.3	-194.8	710.3	-173.8
100	1257.5	937.8	-319.7	986.7	-270.8	956.4	-301.1	979.9	-277.6
130	1631.5	1236.4	-395.1	1269.6	-361.9	1230.5	-401.0	1256.7	-374.8
160	2009.2	1473.4	-535.8	1552.7	-456.5	1494.3	-514.9	1544.1	-465.1
190	2380.1	1777.9	-602.2	1838.0	-542.0	1793.0	-587.0	1832.9	-547.2
220	2766.0	2072.1	-693.9	2095.3	-670.7	1997.9	-768.1	2049.4	-716.6
250	3137.7	2308.2	-829.5	2351.1	-786.6	2303.6	-834.1	2339.8	-797.9
280	3511.7	2528.9	-982.8	2605.6	-906.2	2526.7	-985.1	2608.3	-903.4
310	3885.6	2837.2	-1048.4	2936.0	-949.6	2836.6	-1049.0	2921.2	-964.4
340	4259.6	3116.7	-1142.9	3186.5	-1073.1	3103.3	-1156.3	3186.4	-1073.2
370	4629.5	3338.1	-1291.4	3411.1	-1218.3	3279.0	-1350.5	3486.7	-1142.8
400	5014.0	3556.7	-1457.3	3724.2	-1289.8	3599.1	-1415.0	3668.4	-1345.7
430	5391.3	3852.7	-1538.5	3979.4	-1411.8	3882.6	-1508.7	4054.1	-1337.2
460	5749.3	4085.9	-1663.4	4260.4	-1488.9	4089.3	-1660.0	4214.8	-1534.5
490	6130.6	4322.9	-1807.6	4641.4	-1489.2	4411.2	-1719.4	4577.5	-1553.0
520	6510.6	4635.7	-1874.9	4734.1	-1776.5	4572.1	-1938.5	4753.4	-1757.2
550	6886.5	4962.1	-1924.3	4970.2	-1916.3	5012.2	-1874.3	5109.9	-1776.6
580	7255.0	5166.6	-2088.4	5430.9	-1824.1	5231.2	-2023.8	5276.2	-1978.8
610	7670.3	5448.9	-2221.4	5520.7	-2149.6	5304.1	-2366.2	5708.2	-1962.1
640	8004.5	5541.8	-2462.8	5876.2	-2128.3	5662.7	-2341.9	5938.9	-2065.6
670	8397.7	5950.8	-2446.9	6136.1	-2261.6	6135.6	-2262.1	6214.7	-2183.0
700	8744.3	6144.5	2599.9	6284.4	2459.9	6232.9	2511.5	6437.1	-2307.3
730	9124.5	6406.2	-2718.3	6680.6	-2443.9	6547.1	-2577.4	6679.5	-2445.0
760	9521.5	6810.1	-2711.4	6896.0	-2625.5	6807.9	-2713.6	6898.2	-2623.3
790	9895.6	6910.0	-2985.6	7063.4	-2832.2	7008.3	-2887.2	6968.1	-2927.4
820	10272.2	7321.4	-2950.8	7561.8	-2710.4	7243.5	-3028.6	7416.7	-2855.5

850	10613.2	7538.7	-3074.5	7815.0	-2798.2	7436.0	-3177.2	7714.5	-2898.8
880	11033.7	7755.5	-3278.1	7942.7	-3091.0	7770.2	-3263.5	8150.7	-2882.9
910	11415.9	8067.6	-3348.3	8334.6	-3081.3	7961.5	-3454.4	8473.7	-2942.2
940	11770.2	8155.1	-3615.1	8702.5	-3067.7	8069.7	-3700.5	8490.2	-3280.0
970	12163.5	8513.9	-3649.6	8743.6	-3419.9	8639.2	-3524.3	8748.6	-3414.9
1000	12522.4	8713.6	-3808.9	9009.8	-3512.6	8981.7	-3540.7	8963.1	-3559.3
1030	12926.6	9059.6	-3867.1	9222.8	-3703.8	9139.2	-3787.4	9192.0	-3734.6
1060	13261.0	9301.6	-3959.4	9381.9	-3879.2	9367.6	-3893.4	9536.4	-3724.6
1090	13626.6	9499.7	-4126.9	9793.1	-3833.5	9626.5	-4000.1	9854.3	-3772.3
1120	14005.1	9718.7	-4286.4	10218.4	-3786.6	9722.4	-4282.6	10233.9	-3771.2
1150	14409.0	10178.6	-4230.4	10315.9	-4093.1	10230.2	-4178.8	10357.5	-4051.5
1180	14781.1	10315.2	-4465.9	10684.3	-4096.8	10327.8	-4453.3	10446.6	-4334.4
1210	15197.4	10695.8	-4501.6	11169.8	-4027.6	10738.3	-4459.1	10839.1	-4358.4
1240	15516.8	10881.4	-4635.5	11169.2	-4347.6	10970.0	-4546.8	11145.0	-4371.8
1270	15883.5	11186.2	-4697.4	11415.3	-4468.2	11190.1	-4693.5	11547.4	-4336.1
1300	16300.9	11387.2	-4913.8	11519.6	-4781.4	11593.1	-4707.8	11740.6	-4560.3
1330	16649.0	11743.0	-4906.0	11922.1	-4726.9	11710.5	-4938.5	12027.8	-4621.2
1360	17027.6	11846.9	-5180.8	12226.3	-4801.3	11822.3	-5205.3	12169.4	-4858.2
1390	17373.2	12135.5	-5237.7	12336.1	-5037.1	12111.7	-5261.5	12550.1	-4823.1
1420	17734.0	12194.2	-5539.8	12746.9	-4987.0	12268.1	-5465.8	12457.1	-5276.9
1450	18124.6	12452.3	-5672.3	12965.2	-5159.3	12749.1	-5375.5	13167.6	-4957.0

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

5247.0	5067.0	5732.0	4773.0	5197.0	5221.0	4614.0	6779.0
5843.0	5410.0	5114.0	6628.0	6986.0	5419.0	5396.0	5086.0
6060.0	5741.0	4838.0	5024.0	5756.0	5105.0	5255.0	5106.0
5501.0	5388.0	5237.0	5667.0	5651.0	5423.0	5992.0	5664.0
5284.0	5718.0	5707.0	5217.0	5935.0	5701.0	5583.0	5431.0
5143.0	5342.0	5557.0	6537.0	5007.0	5164.0	5414.0	5923.0
5707.0	5746.0	Testdatensatz First - Fit					

(b) Datensätze der 50 Testiterationen des First - Fit Verfahrens für den markierten Testfall $tb_x = [10, 610]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

5896.0	5259.0	5265.0	5134.0	5430.0	5356.0	5745.0	5250.0
6018.0	5189.0	5939.0	4461.0	4777.0	6011.0	5248.0	5170.0
5276.0	4990.0	5084.0	5241.0	5267.0	5399.0	4762.0	4971.0
5171.0	5057.0	5673.0	5310.0	5409.0	5970.0	4980.0	5511.0
5715.0	4603.0	4864.0	4931.0	5562.0	5642.0	5784.0	5123.0
4882.0	5231.0	5507.0	5078.0	5616.0	5474.0	5557.0	5085.0
5363.0	4967.0	Testdatensatz Random - Fit					

(c) Datensätze der 50 Testiterationen der zufälligen Verteilung für den markierten Testfall $tb_x = [10, 610]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.7: Datensätze - Optimierung: Rüstzeit $tr_x = 1.00 \cdot tb_x$

t - Test

Es werden die Mittelwerte des First - Fit Algorithmus $\bar{x}_1$ und der zufälligen Verteilung (Random Fit) $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_2 > \mu_1$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{5520.7 - 5304.1}{\sqrt{485.8^2 + 363.8^2}} = 2.52$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t \not< c \rightarrow H_0$ wird verworfen!

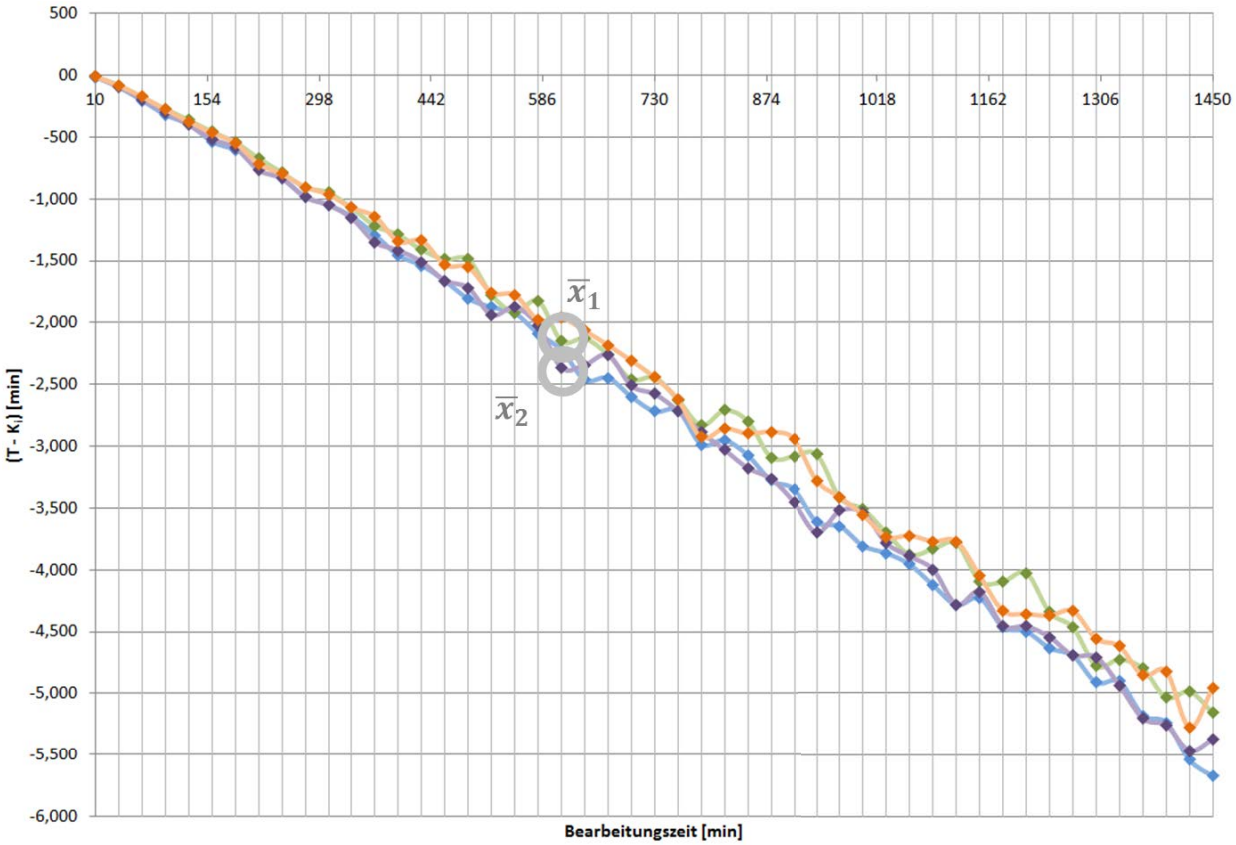


Abbildung A.8: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 70 abgebildet wurde.

A.5 Reihung: Anzahl der Produktionslinien $s = 4$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 72 näher:

Bearbeitungszeit $tb_x \in [10, 820]$ Minuten
Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 1$
Anzahl der Aufträge $n = 50$

Bearbeitungs -zeiten tb_x [min]	Richtwert Ki [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie							
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	RandomFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
		Random	Random	Random	Random	Random	Random	Random	Random
10	444.7	228.3	-216.4	220.7	-224.0	221.3	-223.4	229.2	-215.5
40	633.1	427.5	-205.6	432.3	-200.8	419.8	-213.3	446.0	-187.1
70	822.2	618.7	-203.5	626.5	-195.7	618.7	-203.5	644.5	-177.7
100	1008.8	820.4	-188.4	819.2	-189.6	810.8	-198.0	840.6	-168.2
130	1198.6	1004.8	-193.8	1023.4	-175.2	1012.0	-186.6	1039.7	-158.9
160	1380.3	1199.6	-180.7	1219.3	-161.0	1204.9	-175.4	1236.1	-144.2
190	1570.3	1400.4	-169.9	1404.0	-166.3	1411.0	-159.3	1424.7	-145.6
220	1755.2	1577.7	-177.5	1629.6	-125.6	1596.3	-158.9	1602.3	-152.9
250	1945.0	1774.0	-171.0	1812.0	-133.0	1792.6	-152.4	1838.1	-106.9
280	2135.5	1992.4	-143.1	2009.9	-125.6	1929.5	-206.0	1999.6	-135.9
310	2322.4	2164.7	-157.7	2177.3	-145.1	2171.1	-151.3	2189.7	-132.7
340	2513.9	2344.5	-169.4	2407.4	-106.5	2377.0	-136.9	2413.0	-100.9
370	2699.0	2559.2	-139.8	2613.0	-86.0	2580.9	-118.1	2575.3	-123.7
400	2889.7	2750.5	-139.2	2787.7	-102.0	2738.2	-151.5	2798.9	-90.8
430	3066.1	2979.6	-86.5	2939.9	-126.2	2906.1	-160.0	2947.7	-118.4
460	3262.3	3121.3	-141.0	3160.0	-102.3	3126.1	-136.2	3100.1	-162.2
490	3441.7	3318.4	-123.3	3372.5	-69.2	3348.1	-93.6	3308.4	-133.3
520	3633.8	3517.3	-116.5	3598.9	-34.9	3502.8	-131.0	3558.4	-75.4
550	3816.8	3685.5	-131.3	3772.6	-44.2	3716.1	-100.7	3766.3	-50.5
580	4014.5	3939.6	-74.9	3994.1	-20.4	3883.9	-130.6	3902.4	-112.1
610	4204.8	4112.0	-92.8	4106.1	-98.7	4062.3	-142.5	4178.6	-26.2
640	4380.3	4182.0	-198.3	4260.2	-120.1	4310.2	-70.1	4316.4	-63.9
670	4572.3	4478.2	-94.1	4529.3	-43.0	4572.2	-0.1	4594.7	22.4
700	4758.8	4699.2	-59.6	4711.5	-47.3	4608.6	-150.2	4705.5	-53.3
730	4939.1	4792.5	-146.6	4945.6	6.5	4931.0	-8.1	4934.7	-4.4
760	5103.5	5088.6	-14.9	5081.4	-22.1	5008.7	-94.8	5075.7	-27.8
790	5320.4	5248.6	-71.8	5142.3	-178.1	5243.0	-77.4	5362.3	41.9
820	5501.1	5378.4	-122.7	5443.0	-58.1	5349.4	-151.7	5408.9	-92.2

850	5697.2	5642.9	-54.3	5586.4	-110.8	5639.5	-57.7	5658.2	-39.0
880	5893.8	5846.4	-47.4	5916.0	22.2	5889.2	-4.6	5810.1	-83.7
910	6073.7	5937.7	-136.0	6063.4	-10.3	6007.0	-66.7	6058.5	-15.2
940	6264.5	6145.0	-119.5	6140.6	-123.9	6187.1	-77.4	6084.4	-180.1
970	6493.0	6425.1	-67.9	6385.3	-107.7	6365.9	-127.1	6398.5	-94.5
1000	6633.0	6620.5	-12.5	6390.6	-242.4	6771.0	138.0	6594.0	-39.0
1030	6803.7	6714.7	-89.0	7006.9	203.2	6820.2	16.5	6836.9	33.2
1060	7016.0	6892.8	-123.2	6976.7	-39.3	7021.8	5.8	7061.8	45.8
1090	7199.3	7232.4	33.1	7190.8	-8.5	7096.0	-103.3	7150.0	-49.3
1120	7401.3	7369.2	-32.1	7336.1	-65.2	7289.2	-112.1	7281.7	-119.6
1150	7586.9	7614.9	28.0	7663.3	76.4	7551.1	-35.8	7526.6	-60.3
1180	7774.7	7675.9	-98.8	7842.2	67.5	7846.5	71.8	7803.4	28.7
1210	7976.4	7881.0	-95.4	7955.8	-20.6	7883.0	-93.4	8034.0	57.6
1240	8138.3	8292.5	154.2	8203.1	64.8	8086.2	-52.1	7914.8	-223.5
1270	8343.1	8289.6	-53.5	8327.0	-16.1	8433.9	90.8	8348.3	5.2
1300	8523.4	8482.7	-40.7	8593.4	70.0	8586.5	63.1	8615.1	91.7
1330	8659.9	8602.1	-57.8	8720.6	60.7	8812.7	152.8	8618.4	-41.5
1360	8926.8	8930.3	3.5	8909.5	-17.3	9069.4	142.6	8875.8	-51.0
1390	9093.1	9012.2	-80.9	9072.5	-20.6	9011.0	-82.1	9176.4	83.3
1420	9267.9	9143.2	-124.7	9099.7	-168.2	9262.5	-5.4	9393.5	125.6
1450	9471.6	9376.1	-95.5	9383.0	-88.6	9503.1	31.5	9598.3	126.7

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

5881.0	4783.0	4918.0	5707.0	5578.0	4780.0	6286.0	5204.0
5097.0	5067.0	5927.0	5024.0	5214.0	5075.0	5071.0	4867.0
5272.0	5293.0	6327.0	5241.0	5478.0	5251.0	4888.0	5707.0
4630.0	5578.0	5174.0	5518.0	4599.0	5131.0	5608.0	4475.0
4913.0	5207.0	5579.0	5152.0	5737.0	5309.0	6129.0	5728.0
5969.0	5997.0	5771.0	4686.0	5978.0	5263.0	5541.0	5522.0
5744.0	5885.0	**Testdatensatz Best - Fit**					

(b) Datensätze der 50 Testiterationen des Best - Fit Verfahrens für den markierten Testfall $tb_x = [10, 820]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

5445.0	5310.0	5504.0	5374.0	5458.0	5303.0	5224.0	5795.0
5773.0	5073.0	4951.0	5698.0	5405.0	5554.0	5285.0	5344.0
4859.0	5341.0	4653.0	5052.0	5748.0	5707.0	4485.0	5862.0
4917.0	5360.0	5855.0	5607.0	5085.0	5998.0	5649.0	5676.0
5879.0	5381.0	5175.0	5494.0	5714.0	5149.0	6367.0	4631.0
5264.0	5305.0	5646.0	5445.0	5150.0	4547.0	5464.0	4913.0
5501.0	4885.0	**Testdatensatz Random - Fit**					

(c) Datensätze der 50 Testiterationen der zufälligen Verteilung für den markierten Testfall $tb_x = [10, 820]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.9: Datensätze - Reihung: Anzahl der Produktionslinien $s = 4$

t - Test

Es werden die Mittelwerte des Best - Fit Algorithmus $\bar{x}_1$ und der zufälligen Verteilung (Random Fit) $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{5378.4 - 5349.4}{\sqrt{448.5^2 + 371.8^2}} = 0.35$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t < c \rightarrow H_0$ wird nicht verworfen!

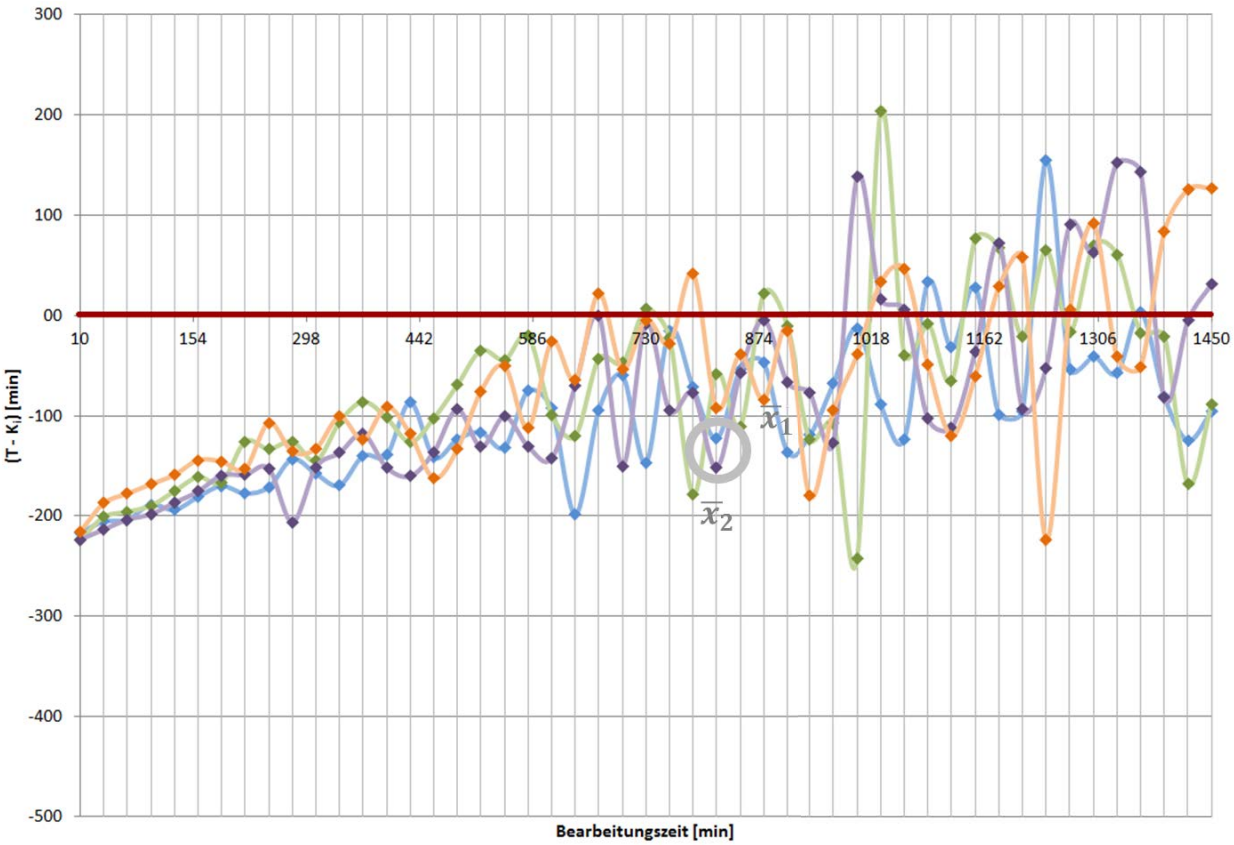


Abbildung A.10: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 72 abgebildet wurde.

A.6 Reihung: Anzahl der Produktionslinien $s = 3$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 72 näher:

Bearbeitungszeit $tb_x \in [10, 1210]$ Minuten
Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Produktionslinien $s = 3$
Anzahl der Maschinen $k = 1$
Anzahl der Aufträge $n = 50$

Bearbeitungs -zeiten tb_x [min]	Richtwert Ki [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie							
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	RandomFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
		Random	Random	Random	Random	Random	Random	Random	Random
10	592.5	242.4	-350.1	246.4	-346.1	244.4	-348.1	248.3	-344.2
40	842.8	506.7	-336.1	516.0	-326.9	512.9	-329.9	516.2	-326.7
70	1092.6	777.1	-315.5	759.5	-333.1	764.1	-328.5	768.5	-324.1
100	1342.6	1023.5	-319.1	1014.8	-327.8	1025.3	-317.4	1038.6	-304.0
130	1589.8	1290.5	-299.3	1284.6	-305.2	1284.1	-305.7	1309.7	-280.1
160	1851.3	1534.7	-316.5	1546.0	-305.3	1550.3	-301.0	1562.5	-288.8
190	2092.3	1796.5	-295.8	1798.7	-293.6	1796.1	-296.2	1797.3	-295.0
220	2353.9	2082.5	-271.4	2081.0	-272.9	2060.6	-293.3	2068.0	-285.9
250	2594.1	2312.9	-281.3	2289.5	-304.6	2345.6	-248.6	2307.6	-286.5
280	2844.4	2566.9	-277.6	2546.9	-297.5	2547.0	-297.4	2591.7	-252.8
310	3099.8	2812.6	-287.2	2849.1	-250.8	2877.8	-222.0	2829.7	-270.1
340	3348.4	3072.7	-275.7	3086.7	-261.7	3073.4	-275.1	3112.7	-235.7
370	3593.4	3377.1	-216.3	3306.3	-287.0	3288.2	-305.2	3344.6	-248.7
400	3846.1	3606.3	-239.8	3557.9	-288.2	3615.4	-230.8	3566.0	-280.1
430	4090.8	3862.6	-228.2	3813.3	-277.4	3854.0	-236.8	3873.8	-217.0
460	4346.5	4107.7	-238.7	4067.1	-279.4	4064.6	-281.9	4162.8	-183.6
490	4595.3	4353.9	-241.5	4371.5	-223.9	4428.8	-166.6	4292.2	-303.1
520	4865.4	4622.4	-243.0	4602.1	-263.3	4583.3	-282.1	4529.1	-336.3
550	5089.4	4959.4	-130.0	4882.2	-207.2	4890.7	-198.7	4851.1	-238.3
580	5342.3	5157.8	-184.5	5158.9	-183.5	5169.7	-172.6	5063.9	-278.5
610	5591.3	5418.8	-172.5	5424.1	-167.2	5396.1	-195.2	5391.9	-199.4
640	5871.8	5635.7	-236.1	5592.5	-279.4	5580.5	-291.3	5510.8	-361.1
670	6101.9	5964.1	-137.8	5875.6	-226.3	5813.2	-288.7	5859.7	-242.2
700	6365.4	5968.0	-397.5	6093.6	-271.8	6096.6	-268.8	6128.1	-237.3
730	6578.0	6356.9	-221.0	6362.6	-215.3	6514.3	-63.7	6432.9	-145.1
760	6857.0	6727.9	-129.1	6908.8	51.8	6732.8	-124.2	6799.9	-57.2
790	7087.1	6908.8	-178.3	6961.6	-125.4	6930.7	-156.3	6901.0	-186.1
820	7379.9	7064.7	-315.1	7207.1	-172.8	7143.1	-236.7	7093.7	-286.1

850	7622.1	7395.8	-226.3	7395.1	-227.0	7358.5	-263.6	7311.4	-310.8
880	7865.9	7706.4	-159.5	7845.4	-20.5	7630.1	-235.8	7774.0	-91.9
910	8094.9	7923.5	-171.4	8014.2	-80.7	8090.7	-4.2	8024.2	-70.7
940	8344.5	8243.2	-101.3	8336.4	-8.1	8098.4	-246.1	8198.5	-146.0
970	8635.4	8474.1	-161.3	8548.4	-86.9	8520.1	-115.3	8449.1	-186.3
1000	8881.4	8722.0	-159.4	8796.5	-84.9	8635.6	-245.8	8733.5	-147.9
1030	9096.9	8839.8	-257.1	8984.7	-112.2	8919.0	-177.9	8907.7	-189.2
1060	9331.1	9257.7	-73.5	9298.9	-32.2	9089.9	-241.2	9193.3	-137.8
1090	9642.7	9366.7	-276.0	9378.1	-264.6	9472.2	-170.4	9374.6	-268.1
1120	9826.4	9829.2	2.8	9812.6	-13.8	9735.1	-91.3	9771.2	-55.2
1150	10102.5	9935.9	-166.5	10091.0	-11.5	9785.9	-316.6	9935.2	-167.2
1180	10404.0	10336.5	-67.5	10180.5	-223.6	10344.0	-60.0	10215.7	-188.3
1210	10654.8	10466.9	-187.9	10413.9	-240.9	10376.1	-278.7	10340.5	-314.3
1240	10874.0	10846.2	-27.8	10748.4	-125.7	10495.3	-378.7	10690.1	-184.0
1270	11085.3	10865.8	-219.6	11060.0	-25.3	10845.6	-239.8	10945.9	-139.5
1300	11388.8	11384.0	-4.8	11143.7	-245.1	11195.6	-193.2	11181.2	-207.6
1330	11575.5	11399.6	-175.9	11447.3	-128.3	11576.2	0.7	11476.6	-99.0
1360	11865.3	11976.1	110.8	11811.0	-54.3	11744.0	-121.2	11809.0	-56.3
1390	12104.9	11859.4	-245.5	12030.8	-74.2	11883.4	-221.5	12000.7	-104.3
1420	12413.7	12069.6	-344.1	12218.3	-195.4	12362.2	-51.5	12653.2	239.5
1450	12618.9	12592.9	-25.9	12510.0	-108.8	12605.2	-13.7	12556.1	-62.7

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

9821.0	10766.0	11651.0	10916.0	11424.0	9378.0	8887.0	9819.0
10803.0	10747.0	10412.0	10271.0	10085.0	10066.0	10729.0	9910.0
11074.0	10308.0	10243.0	10268.0	10838.0	10677.0	9775.0	10886.0
10982.0	11069.0	11561.0	10924.0	10008.0	9408.0	9002.0	11272.0
10922.0	10466.0	10958.0	9617.0	10329.0	9739.0	10353.0	11639.0
10169.0	10329.0	9994.0	10599.0	10941.0	8725.0	10124.0	10663.0
10263.0	8997.0	Testdatensatz Random - Fit					

(b) Datensätze der 50 Testiterationen der zufälligen Verteilung für den markierten Testfall $tb_x = [10, 1210]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

10502.0	10252.0	9166.0	11826.0	10368.0	10382.0	11457.0	10201.0
10502.0	10036.0	11562.0	9242.0	10619.0	11600.0	10095.0	9469.0
8874.0	9976.0	12141.0	10844.0	9432.0	10937.0	11473.0	10285.0
9340.0	10758.0	9859.0	9969.0	9650.0	10936.0	9526.0	9912.0
8620.0	10922.0	9664.0	9734.0	11381.0	10732.0	9410.0	10557.0
10095.0	11877.0	10833.0	10378.0	10891.0	10468.0	9028.0	10233.0
11792.0	9221.0	Testdatensatz Minimum Bin Slack					

(c) Datensätze der 50 Testiterationen des Minimum Bin Slack Verfahrens für den markierten Testfall $tb_x = [10, 1210]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.11: Datensätze - Reihung: Anzahl der Produktionslinien $s = 3$

t - Test

Es werden die Mittelwerte der zufälligen Verteilung $\bar{x}_1$ und des Minimum Bin Slack Algorithmus $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{10376.1 - 10340.5}{\sqrt{702.6^2 + 857.3^2}} = 0.23$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t < c \rightarrow H_0$ wird nicht verworfen!

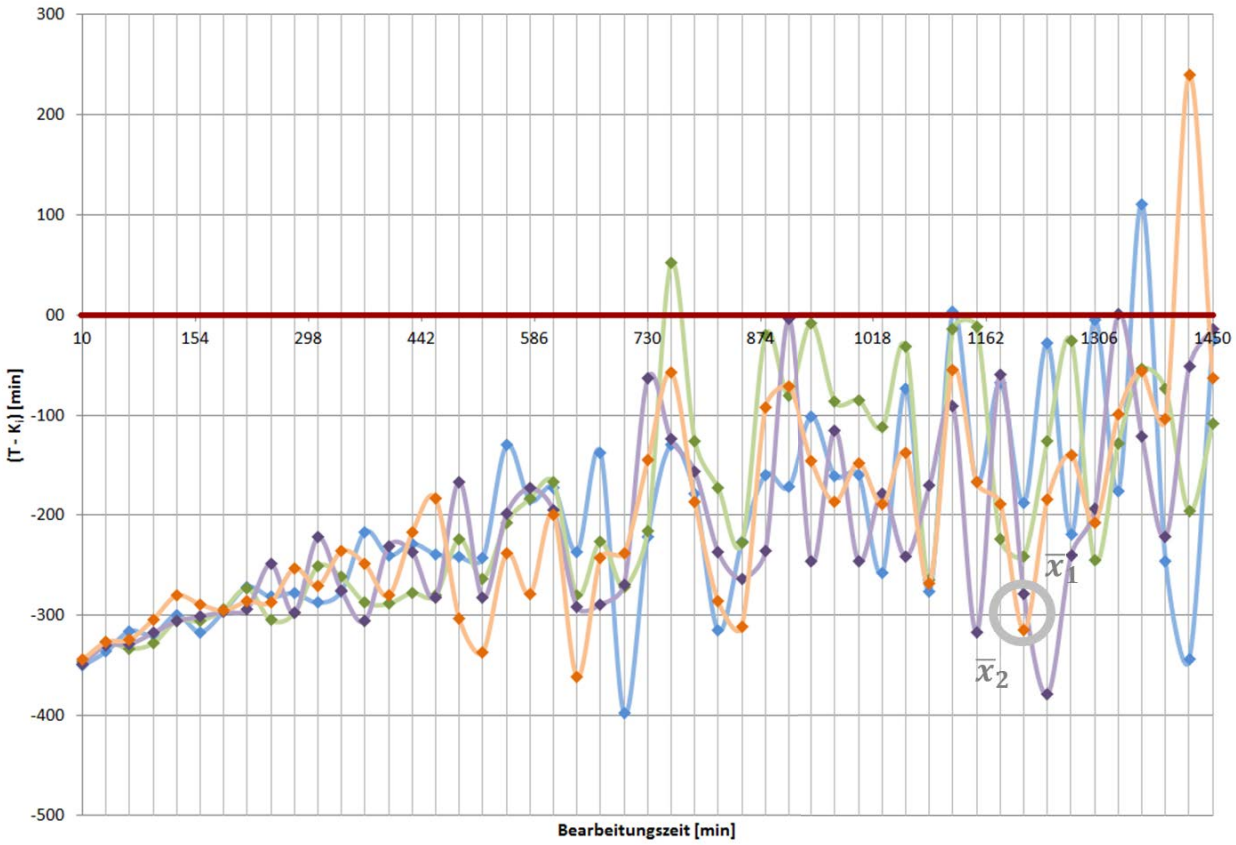


Abbildung A.12: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 72 abgebildet wurde.

A.7 Reihung: Anzahl der Maschinen $k = 5$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 73 näher:

Bearbeitungszeit $tb_x \in [10, 580]$ Minuten
Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 5$
Anzahl der Aufträge $n = 50$

Bearbeitungs -zeiten tb_x [min]	Richtwert Ki [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie							
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	RandomFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
		Random	Random	Random	Random	Random	Random	Random	Random
10	2221.4	1688.4	-533.0	1707.6	-513.8	1691.2	-530.2	1704.9	-516.5
40	3161.0	2645.3	-515.7	2645.0	-516.0	2644.1	-516.9	2694.7	-466.3
70	4099.5	3634.6	-465.0	3616.3	-483.3	3605.3	-494.3	3641.9	-457.7
100	5041.6	4527.1	-514.4	4583.7	-457.9	4608.6	-433.0	4628.8	-412.8
130	5969.1	5467.1	-502.0	5585.7	-383.4	5548.6	-420.6	5602.4	-366.8
160	6919.8	6512.3	-407.5	6548.7	-371.1	6530.7	-389.1	6478.5	-441.3
190	7858.4	7476.0	-382.4	7469.8	-388.7	7520.1	-338.3	7410.7	-447.7
220	8794.3	8458.1	-336.1	8408.6	-385.7	8427.5	-366.8	8440.4	-353.8
250	9734.2	9349.2	-385.0	9515.4	-218.8	9342.5	-391.7	9428.8	-305.4
280	10650.6	10308.2	-342.4	10357.0	-293.6	10415.2	-235.4	10377.5	-273.1
310	11616.8	11340.2	-276.6	11351.6	-265.2	11281.8	-335.0	11304.2	-312.6
340	12531.5	12123.1	-408.4	12322.4	-209.1	12337.5	-194.0	12316.9	-214.6
370	13450.0	13293.4	-156.7	13341.7	-108.4	13182.0	-268.1	13182.2	-267.8
400	14428.9	14244.8	-184.1	14090.7	-338.3	14218.3	-210.6	14280.6	-148.4
430	15335.6	15125.4	-210.2	15300.1	-35.5	15054.1	-281.5	15177.7	-157.9
460	16311.6	16204.7	-106.9	16207.8	-103.8	16039.2	-272.4	16166.7	-144.9
490	17230.9	17037.4	-193.6	17012.1	-218.9	16971.0	-259.9	17173.0	-58.0
520	18184.3	18085.3	-99.0	18047.8	-136.5	18030.9	-153.4	17977.3	-207.0
550	19122.7	18983.7	-139.0	19016.3	-106.4	18975.3	-147.4	19093.8	-29.0
580	20079.3	19426.5	-652.8	19674.9	-404.4	20148.0	68.7	19978.3	-101.0
610	20993.7	20823.4	-170.3	20862.5	-131.2	20978.6	-15.0	21026.5	32.8
640	21898.5	21749.9	-148.6	21697.1	-201.5	21960.5	62.0	21727.0	-171.6
670	22827.2	22795.4	-31.8	23115.7	288.5	22813.5	-13.7	22712.7	-114.5
700	23837.3	23893.9	56.6	23602.6	-234.7	23748.1	-89.2	23756.4	-80.9
730	24709.9	24690.9	-19.0	24875.8	165.9	24602.4	-107.5	24765.4	55.5
760	25654.7	25626.9	-27.7	25901.9	247.2	25794.1	139.4	25712.1	57.5
790	26666.1	26932.7	266.6	26675.5	9.4	26731.1	65.0	26637.8	-28.3
820	27583.5	27483.3	-100.2	27440.0	-143.5	27543.4	-40.0	27516.4	-67.0

850	28451.0	28633.6	182.6	28637.3	186.3	28680.9	229.9	28702.2	251.2
880	29480.5	29569.4	89.0	29625.9	145.4	29424.3	-56.2	29524.5	44.0
910	30335.2	30292.6	-42.6	30402.9	67.7	30337.2	2.0	30694.8	359.6
940	31284.2	31654.1	369.9	31397.0	112.8	31413.1	128.9	31534.0	249.8
970	32270.9	32287.9	17.0	32536.9	266.0	32194.5	-76.4	32506.9	236.0
1000	33192.1	33216.7	24.6	33683.5	491.5	33244.8	52.8	33036.8	-155.3
1030	34135.2	34316.5	181.3	34389.8	254.6	34240.6	105.4	33944.9	-190.3
1060	35102.9	35258.4	155.4	35380.3	277.3	35360.5	257.6	34850.6	-252.4
1090	35957.1	36457.2	500.1	36416.8	459.7	36157.3	200.2	36279.6	322.5
1120	36930.5	36798.3	-132.1	37012.3	81.9	37031.0	100.5	37286.8	356.4
1150	37889.7	38378.2	488.5	38124.2	234.5	38020.6	130.9	37937.3	47.5
1180	38878.5	38869.8	-8.7	38989.0	110.5	39515.9	637.4	38689.8	-188.7
1210	39789.4	39375.0	-414.4	40180.0	390.6	40368.7	579.3	40118.7	329.3
1240	40664.9	40717.9	53.1	41236.1	571.2	41143.5	478.7	41075.4	410.5
1270	41587.2	41686.3	99.1	41939.0	351.9	42029.1	442.0	42089.4	502.2
1300	42568.8	42939.8	371.0	43296.0	727.1	42852.7	283.8	42666.0	97.2
1330	43506.6	44187.9	681.3	44242.0	735.5	44414.7	908.1	43599.1	92.5
1360	44503.0	45251.0	747.9	44967.1	464.1	45384.3	881.3	44947.2	444.2
1390	45412.6	45713.0	300.4	45943.7	531.1	45405.2	-7.4	45851.7	439.1
1420	46353.2	46751.2	398.0	46891.1	537.9	46876.7	523.5	46682.2	329.0
1450	47081.8	47695.7	613.9	47564.1	482.3	48226.4	1144.5	47970.0	888.1

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

19527.0	19245.0	19080.0	19891.0	18784.0	18568.0	19826.0	19108.0
18938.0	19630.0	19438.0	18930.0	19390.0	20090.0	20042.0	19986.0
20093.0	18998.0	18779.0	19324.0	19700.0	20426.0	18635.0	18806.0
19334.0	18932.0	18859.0	18602.0	20690.0	19277.0	19195.0	20034.0
19151.0	19397.0	19766.0	19636.0	19647.0	19227.0	20380.0	18969.0
19169.0	18970.0	19286.0	20575.0	19022.0	19245.0	18887.0	20334.0
20016.0	19493.0	Testdatensatz Best - Fit					

(b) Datensätze der 50 Testiterationen des Best - Fit Verfahrens für den markierten Testfall $tb_x = [10, 580]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

19950.0	19975.0	19649.0	18964.0	19827.0	19147.0	19076.0	19199.0
20504.0	19933.0	19006.0	20066.0	19822.0	19426.0	19819.0	19456.0
19352.0	19911.0	20152.0	19797.0	19691.0	19211.0	19040.0	20233.0
19418.0	20120.0	20273.0	18829.0	19467.0	19487.0	19806.0	19931.0
19763.0	19405.0	18758.0	20269.0	20161.0	20571.0	18627.0	20059.0
19133.0	20288.0	19556.0	19680.0	20408.0	20559.0	19150.0	20542.0
18363.0	19918.0	Testdatensatz First - Fit					

(c) Datensätze der 50 Testiterationen des First - Fit Verfahrens für den markierten Testfall $tb_x = [10, 580]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.13: Datensätze - Reihung: Anzahl der Maschinen $k = 5$

t - Test

Es werden die Mittelwerte des Best - Fit $\bar{x}_1$ und des First Fit Algorithmus $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_2 > \mu_1$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{19674.9 - 19426.5}{\sqrt{539.6^2 + 546.0^2}} = 2.29$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t \not< c \rightarrow H_0$ wird verworfen!

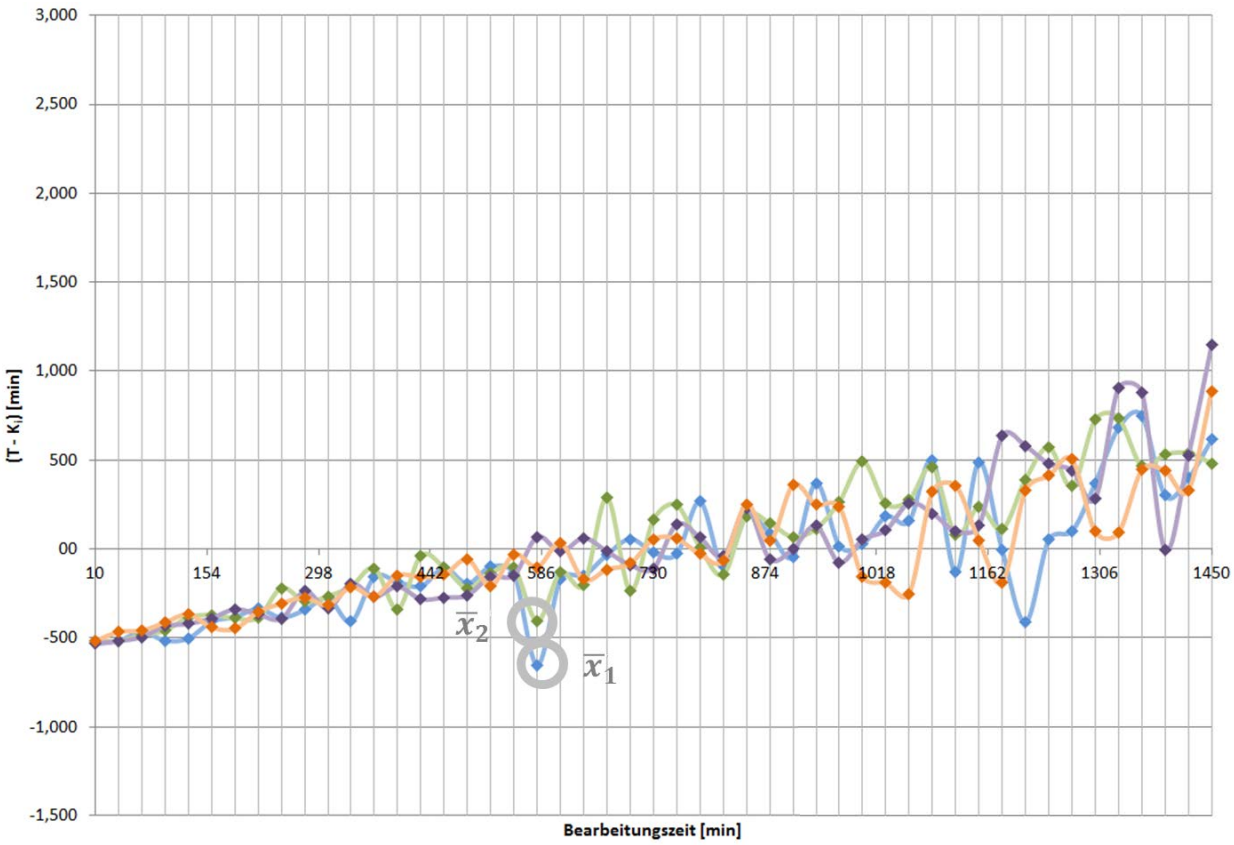


Abbildung A.14: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 73 abgebildet wurde.

A.8 Reihung: Anzahl der Maschinen $k = 15$

Folgende Parametereinstellungen beschreiben den Testfall von Seite 73 näher:

Bearbeitungszeit $tb_x \in [10, 280]$ Minuten
Rüstzeit $tr_x \in [0, 60]$ Minuten
Anzahl der Produktionslinien $s = 4$
Anzahl der Maschinen $k = 15$
Anzahl der Aufträge $n = 50$

Bearbeitungs -zeiten tb_x [min]	Richtwert Ki [min]	Mittlere, maximale Gesamtdurchlaufzeit einer Produktionslinie							
		BestFit	Differenz Richtwert	FirstFit	Differenz Richtwert	RandomFit	Differenz Richtwert	MinimumBin Slack	Differenz Richtwert
		Random	Random	Random	Random	Random	Random	Random	Random
10	6663.6	5654.4	-1009.2	5627.3	-1036.3	5656.6	-1007.0	5627.9	-1035.7
40	9481.8	8539.3	-942.6	8520.9	-960.9	8472.3	-1009.5	8566.6	-915.2
70	12295.9	11400.9	-895.1	11383.7	-912.2	11430.8	-865.2	11365.1	-930.8
100	15105.3	14246.2	-859.1	14265.0	-840.3	14311.3	-794.0	14284.3	-821.0
130	17929.5	17109.1	-820.4	17052.1	-877.4	17184.9	-744.6	17193.5	-736.0
160	20754.6	20019.4	-735.2	19977.2	-777.4	19995.3	-759.3	19892.6	-862.0
190	23578.5	22836.6	-741.9	22823.8	-754.7	22807.0	-771.5	22781.5	-797.0
220	26361.9	25739.6	-622.4	25851.8	-510.1	25798.9	-563.1	25788.2	-573.7
250	29170.4	28664.6	-505.8	28666.8	-503.6	28612.3	-558.0	28645.4	-524.9
280	32025.8	30979.2	-1046.6	31546.2	-479.6	31642.6	-383.3	30721.8	-1304.0
310	34801.5	34247.3	-554.2	34417.5	-384.0	34375.7	-425.8	34392.3	-409.2
340	37627.1	37148.5	-478.5	37097.0	-530.1	37213.1	-414.0	37263.3	-363.7
370	40418.9	40185.8	-233.1	39873.5	-545.4	40062.1	-356.8	40148.6	-270.3
400	43311.3	42961.7	-349.6	43155.3	-156.0	43070.3	-240.9	43179.2	-132.1
430	46060.1	45928.0	-132.1	45924.7	-135.4	45602.3	-457.8	45943.3	-116.8
460	48902.3	48686.8	-215.5	48883.1	-19.2	48764.1	-138.2	48537.2	-365.1
490	51730.7	51576.1	-154.6	51604.8	-125.9	51461.6	-269.2	51655.9	-74.8
520	54468.5	54714.1	245.6	54537.9	69.4	54327.4	-141.1	54668.4	199.9
550	57421.2	57594.1	172.8	57216.4	-204.8	57432.6	11.4	57487.5	66.2
580	60181.3	60160.7	-20.6	60333.2	151.9	60306.9	125.6	59832.3	-349.0
610	62972.7	62855.6	-117.0	63379.5	406.8	63399.0	426.3	63131.7	159.0
640	65794.3	65703.8	-90.6	66271.8	477.5	66173.3	379.0	65852.1	57.7
670	68573.4	69045.2	471.8	69017.1	443.7	68848.5	275.1	69126.8	553.4
700	71426.4	71603.0	176.6	71306.8	-119.6	71746.2	319.8	71278.4	-148.0
730	74215.6	74578.5	362.9	74456.6	240.9	74888.5	672.9	74870.8	655.1
760	77110.0	77715.8	605.8	77819.4	709.4	76957.6	-152.4	77569.1	459.2
790	79828.4	80091.5	263.2	79887.5	59.1	80316.0	487.6	80117.5	289.1
820	82703.4	83190.9	487.4	82988.1	284.7	83194.4	490.9	83279.9	576.4

850	85472.9	86441.0	968.1	86064.6	591.7	85889.6	416.7	86499.9	1026.9
880	88272.5	89131.7	859.2	89103.6	831.1	88888.5	616.0	88609.6	337.1
910	91060.4	91447.1	386.6	92114.2	1053.8	91599.8	539.4	91796.4	735.9
940	94002.1	94563.8	561.8	95277.9	1275.8	94870.8	868.8	94549.0	546.9
970	96807.0	97440.9	633.8	97716.9	909.9	97805.9	998.9	97519.4	712.4
1000	99479.8	100291.4	811.6	100694.4	1214.6	100457.8	978.0	100321.9	842.1
1030	102311.9	102899.9	588.0	103282.4	970.5	103059.8	747.9	102987.2	675.3
1060	105241.1	106282.6	1041.5	105917.8	676.7	106019.9	778.8	106304.1	1063.0
1090	107958.4	109413.5	1455.2	108312.6	354.3	109117.2	1158.8	109442.6	1484.2
1120	110679.4	111543.5	864.1	112129.2	1449.7	112385.8	1706.4	111792.7	1113.2
1150	113582.5	114928.9	1346.4	114330.7	748.2	115212.4	1629.9	114790.9	1208.4
1180	116449.6	117321.9	872.3	118083.7	1634.1	116997.6	548.0	117828.6	1379.0
1210	119194.4	120168.2	973.7	120659.5	1465.0	120343.5	1149.1	120492.5	1298.0
1240	121871.4	123574.5	1703.2	123629.6	1758.2	123764.0	1892.7	123730.3	1859.0
1270	124769.9	126153.5	1383.6	126467.5	1697.6	125509.7	739.9	126029.4	1259.5
1300	127598.7	129755.9	2157.2	128723.7	1124.9	128559.5	960.7	129097.1	1498.4
1330	130646.1	131823.0	1176.9	132153.3	1507.2	132419.2	1773.1	131722.6	1076.5
1360	133353.8	135237.6	1883.8	134597.0	1243.2	135266.9	1913.2	135216.5	1862.7
1390	136325.3	137746.4	1421.1	137168.2	842.9	138155.7	1830.4	137870.5	1545.1
1420	138949.1	140568.4	1619.3	141549.8	2600.7	140459.2	1510.1	140835.8	1886.7
1450	141694.2	143674.7	1980.5	144301.2	2607.0	143359.2	1665.0	143243.0	1548.8

(a) Datensätze der gemittelten, maximalen Gesamtdurchlaufzeiten und der errechneten Differenzen $D = (\bar{T} - K_i)$ aller getesteten Algorithmen.

30509.0	31273.0	30705.0	32045.0	31667.0	30135.0	31009.0	31734.0
31622.0	30668.0	30433.0	30493.0	30984.0	30402.0	32274.0	32465.0
30330.0	32741.0	30810.0	29852.0	30786.0	30790.0	31237.0	30868.0
31211.0	31430.0	30373.0	31090.0	29970.0	30550.0	30198.0	31596.0
31494.0	30425.0	31723.0	30976.0	31937.0	30658.0	30212.0	31779.0
30504.0	31898.0	31021.0	30143.0	30718.0	30721.0	30316.0	30603.0
30597.0	30984.0	Testdatensatz Best - Fit					

(b) Datensätze der 50 Testiterationen des Best - Fit Verfahrens für den markierten Testfall $tb_x = [10, 280]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_1$, sowie die Standardabweichung s_1 berechnet werden.

30220.0	30570.0	30981.0	30617.0	30046.0	31592.0	30107.0	31055.0
30506.0	30543.0	31270.0	31830.0	31758.0	30511.0	31040.0	30662.0
30673.0	29784.0	30455.0	31318.0	30295.0	30058.0	30307.0	31494.0
30038.0	30311.0	31200.0	31334.0	30324.0	29614.0	30051.0	31657.0
30086.0	29737.0	30029.0	31151.0	30975.0	31031.0	30176.0	31553.0
31311.0	30980.0	31112.0	30782.0	30238.0	31040.0	30962.0	30892.0
31045.0	30771.0	Testdatensatz Minimum Bin Slack					

(c) Datensätze der 50 Testiterationen des Minimum Bin Slack Verfahrens für den markierten Testfall $tb_x = [10, 280]$. Auf Basis dieser Werte können der Mittelwert $\bar{x}_2$, sowie die Standardabweichung s_2 berechnet werden.

Abbildung A.15: Datensätze - Reihung: Anzahl der Maschinen $k = 15$

t - Test

Es werden die Mittelwerte des Best - Fit $\bar{x}_1$ und des Minimum Bin Slack Algorithmus $\bar{x}_2$ miteinander verglichen.

Die Nullhypothese lässt sich wie folgt beschreiben. $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$

$$t = \sqrt{n}\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2 + s_2^2}} = \sqrt{50}\frac{30979.2 - 30721.8}{\sqrt{675.4^2 + 568.2^2}} = 2.06$$

Anzahl der Freiheitsgrade: $n_1 + n_2 - 2 = 50 + 50 - 2 = 98$

Signifikanzniveau: $\alpha = 0.05$

Tabelle des t - Tests: $c = 1.65$

$t \not< c \rightarrow H_0$ wird verworfen!

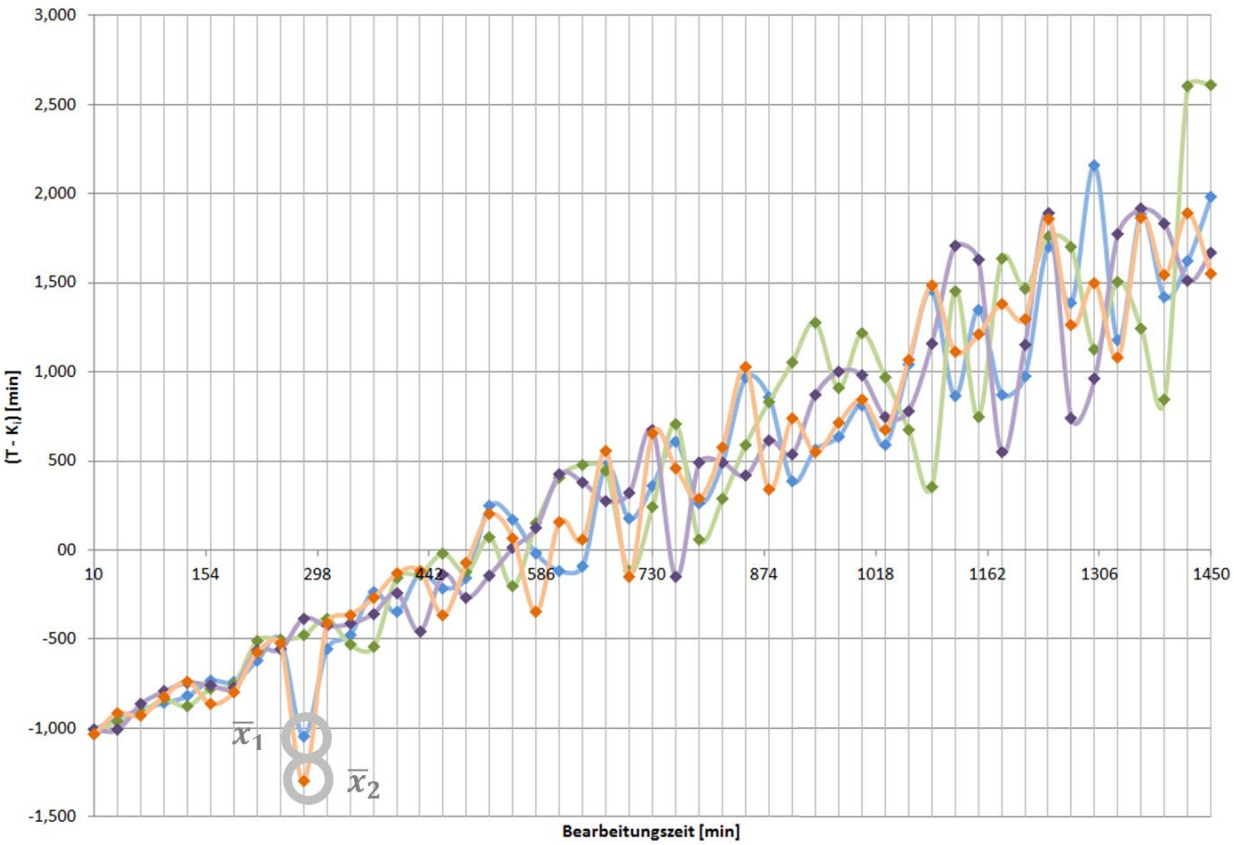


Abbildung A.16: Anhand dieser Grafik wird veranschaulicht, dass das statistische Ergebnis mit der grafischen Beschreibung übereinstimmt und nachvollzogen werden kann. Es handelt sich dabei um das gleiche Diagramm, das schon auf Seite 73 abgebildet wurde.

Abbildungsverzeichnis

Literaturverzeichnis

[1] D. Arnold, H. Isermann, A. Kuhn, K. Furmans, and H. Tempelmeier. *VDI-Handbuch der Logistik*. Springer Verlag, Berlin, Heidelberg, 2008.

[2] P. Brucker. *Complex Scheduling*. Springer Verlag, Berlin, Heidelberg, 2009.

[3] P. Brucker. *Scheduling Algorithms*. Springer Verlag, Berlin, Heidelberg, 2009.

[4] W. Domschke and A. Drexl. *Einführung in Operations Research*. Springer Verlag, Berlin, Heidelberg, 2007.

[5] W. Domschke and A. Scholl. Heuristische Verfahren. Technical report, Wirtschaftswissenschaftliche Fakultät der Friedrich-Schiller-Universität Jena, 2008.

[6] W. Domschke and A. Scholl. *Logistik: Rundreisen und Touren*. Oldenbourg Wissenschaftsverlag, München, 2010.

[7] K. Fleszar and K.S. Hindi. New heuristics for one-dimensional bin-packing. Technical report, Department of Systems Engineering of the Brunel University in Uxbridge, 2000.

[8] D. Jungnickel. *Graphs, Networks and Algorithms*. Springer Verlag, Berlin, Heidelberg, 1999.

[9] R. Koether. *Taschenbuch der Logistik*. Carl Hanser Verlag, München, Wien, 2004.

[10] B. Korte. *Kombinatorische Optimierung*. Springer Verlag, Berlin, Heidelberg, 2008.

[11] E. Kreyszig. *Statistische Methoden und ihre Anwendungen*. Vandenhoeck und Ruprecht, Göttingen, 1975.

[12] D. Laha and S.C. Sarin. Aheuristic to minimize total flowtime in permutation flowshop. Technical report, Department of Mechanical Engineering of the Jadavpur University in India and Grado Department of Industrial and Systems Engineering of the Virginia Tech in the USA, 2008.

[13] J.K. Lenstra, A.H.G. Rinnooy Kaan, and P. Brucker. Complexity of maschine scheduling problems. *Annals of Discrete Mathematics 1*, 1977.

[14] W.M. Lippe. *Soft-Computing: Mit neuronalen Netzen, Fuzzy-Logic und evolutionären Algorithmen.* Springer Verlag, Berlin, Heidelberg, 2005.

[15] C. Low. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Technical report, Department of Industrial Engineering of the Da-Yeh University in Taiwan, 2005.

[16] M.L. Pinedo. *Planning and Scheduling in Manufacturing and Services.* Springer Verlag, Berlin, Heidelberg, 2009.

[17] S. Siegel. *Nonparametric Statistics for the behavioral Science.* McGraw - Hill Book Company, New York, Toronto, London, 1956.

[18] J.D. Smith. *Design and Analysis of Algorithms.* Springer Verlag, Berlin, Heidelberg, 2007.

[19] Y.H. Song. *Modern Optimisation Techniques in Power Systems.* Springer Verlag, Berlin, Heidelberg, 1999.

[20] L.Y. Tseng and Y.T. Lin. A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. Technical report, Institute of Networking and Multimedia / Department of Computer Science and Engineering of the National Chung Hsing University in Tawain, 2009.

[21] E. Winter, R. Mosena, and L. Roberts. *Gablers Wirtschaftlexikon: Die ganze Welt der Wirtschaft: Betriebswirtschaft, Volkswirtschaft, Wirtschaftsrecht, Recht und Steuern.* Gabler Verlag, Wiesbaden, 2009.