



Protokoll

Lösung von Distributed Job-Shop Problemen

Carl Martin

Geboren am: 2. Februar 1996 in Schlema

Studiengang: Wirtschaftsingenieurwesen, 7. FS

Matrikelnummer: 4054734

23. Mai 2018

Betreuender Hochschullehrer

Prof. Dr. Udo Buscher

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Algorithmenverzeichnis	V
Abkürzungsverzeichnis	VI
Symbolverzeichnis	VI
1 Problemstellung und Motivation	1
2 Anwendungsbeispiele in der Praxis	3
3 Mathematische Problemformulierung	5
4 Verfahrenslösung	6
4.1 Regel-basierte Heuristiken	6
4.1.1 Aufteilung von Jobs auf Produktionsstätten	6
4.1.2 Einplanung der Aufträge auf Maschinen	7
4.1.3 Anwendungsbeispiel	7
4.2 Greedy Heuristiken	12
4.2.1 Greedy-Heuristik 1	12
4.2.2 Greedy-Heuristik 2	15
4.2.3 Greedy-Heuristik 3	16
A Programmcodes	18
Literaturverzeichnis	19

Abbildungsverzeichnis

2.1	Produktionsstandorte im Infineon Produktionsnetzwerk	3
4.1	Zuordnung von Aufträgen zu Produktionsstandorten	7
4.2	Interaktives Beispiel - Einplanung nach der SPT Regel	11
4.3	Interaktives Beispiel - Einplanung nach der LPT Regel	12
4.4	Interaktives Beispiel - Einplanung nach der ersten Greedy Heuristik . . .	15
4.5	Interaktives Beispiel - Einplanung nach der zweiten Greedy Heuristik . .	16
4.6	Interaktives Beispiel - Einplanung nach der dritten Greedy Heuristik . . .	17

Tabellenverzeichnis

4.1	Jobs und Operationsfolge für das Anwendungsbeispiel	7
4.2	Ermitteltes Arbeitspensum und Rank	8
4.3	Arbeitstabelle zur Aufteilung von Jobs auf Produktionsstätten	9
4.4	Einplanung von Jobs nach der SPT Regel - Erster Iterationsschritt	10
4.5	Einplanung von Jobs nach der SPT Regel - Ergebnis	10
4.6	Aus der Operationsfolge extrahierte Informationen	14
4.7	Einplanung der Jobs nach der ermittelten Operationsfolge	14

Abkürzungsverzeichnis

DJS	Distributed Job Shop
LPT	Longest Processing Time
SPT	Shortest Processing Time

1 Problemstellung und Motivation

Frühere Epochen der Großserienfertigung waren durch eine fortschreitende Zentralisierung der Betriebe gekennzeichnet, die auf die Zeit der industriellen Revolution und die Entstehung des Fabriksystems aus der früheren Handwerksproduktion zurückgeht. Die technischen Entwicklungen dieser Zeit wurden begleitet durch die Entstehung von Fabrikssystemen und den damit verbundenen Produktivitäts- und Kostenvorteilen. Zwar können Fabriken effizient produzieren, jedoch ist dieses zentralisierte Paradigma auch durch langwierige, oft langsam reagierende Lieferketten geprägt. In den letzten drei Jahrzehnten hat Globalisierung die Industrielandschaft mit mehreren internationalen Produktionsstandorten, die regionale und globale Märkte bedienen, stark verändert.¹

Bei der zentralisierten Werkstattfertigung - also im klassischen Job Shop - wird angenommen, dass es eine einzige Produktionsstätte mit m Maschinen gibt. Das Problem besteht darin n jobs mit jeweils eigenen Prozessrouten so einzutakten, dass eine Zielfunktion minimiert wird. Meist wird hierfür die Fertigungsdauer, also die benötigte Zeit zur Fertigstellung aller Jobs, verwendet. Es werden bei den Lösungsverfahren meist folgende Annahmen getroffen:

- Maschinen & Jobs sind kontinuierlich verfügbar
- Rüstzeit können ignoriert werden oder sind in den Prozesszeiten integriert
- Ein Job kann nur auf einer Maschine gleichzeitig bearbeitet werden
- Eine Maschine kann nur einen Job gleichzeitig bearbeiten

Der Trend zu mehreren Fabriken, die die gleichen Güter für unterschiedliche Märkte produzieren, verändert auch die Anforderungen und Problemstellungen der Maschinenbelegungsplanung. Im Distributed Job Shop (DJS), bestehend aus f Produktionsstätten mit jeweils m Maschinen, wird die Planung komplexer - es müssen zwei Entscheidungen getroffen werden:

¹Vgl. BABBAGE (2005): *From The Economy of Machinery and Manufactures*, S. 135f.

1. Verteilung der Jobs auf die Produktionsstätten
2. Einplanung der Jobs auf die Maschinen

Im DJS Problem werden zusätzlich zum klassischen Job Shop folgende Annahmen getroffen:

- Jobs können nicht auf mehreren Produktionsstätten bearbeitet werden
- Alle Produktionsstätten haben jeweils einen identischen Maschinenpark und können so jeden Auftrag bearbeiten

Die Maschinenbelegungsplanung im DJS kann durch die Adaption von bestehenden Regelbasierten Heuristik oder durch Greedy Heuristiken gelöst werden. Die im Paper NADERI / AZAB 2014: *Modeling and heuristics for scheduling of distributed job shops* entwickelten Heuristiken sollen im Folgenden näher erläutert werden.

2 Anwendungsbeispiele in der Praxis

Gerade die Halbleiterindustrie lebt schon seit einigen Jahren im DJS innerhalb eines globalen Produktionsnetzwerkes. Zu beobachten ist, dass viele Halbleiterfertiger gerade in der Frontend Produktion (Aufbringen der Transistoren auf dem Wafer) über ein weltweites Produktionsnetzwerk verfügen. In den verschiedenen Fabriken werden oft auch identische Produkte gefertigt, um den regionalen Bedarf zu sättigen.

Infineon, beispielsweise, hat 12 Frontend Produktionsstätten weltweit.

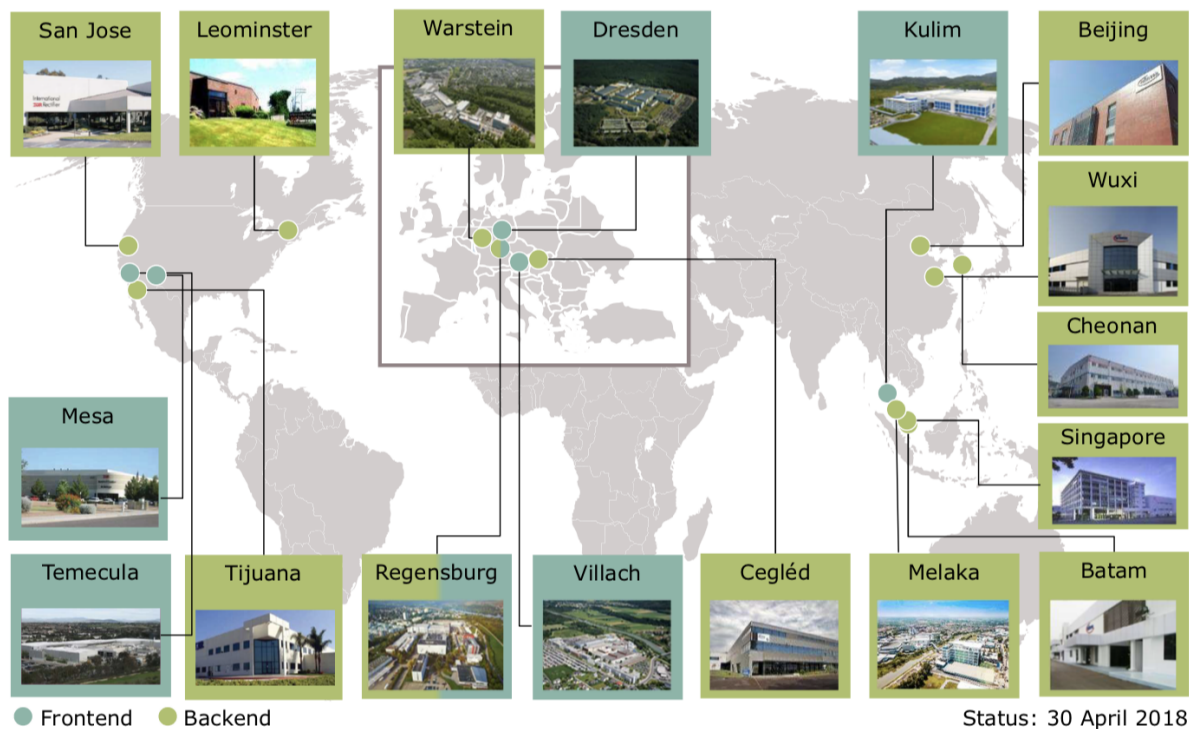


Abbildung 2.1: Produktionsstandorte im Infineon Produktionsnetzwerk¹

Jeder Wafer hat dabei ein spezielles Rezept (Operationsreihenfolge) und wird in einer vorgegebenen Reihenfolge auf unterschiedlichen Maschinen bearbeitet. Selbst die Annahme,

¹Eigene Abbildung

dass Rüstzeiten aus der Betrachtung ausgeschlossen werden, trifft bei fast allen Prozessen aufgrund einem hohen Automatisierungsgrad zu.

3 Mathematische Problemformulierung

Bei der mathematischen Formulierung des Distributed Job Shops DJ_m wird folgende Notation nach PINEDO 2016 verwendet:

- n — Anzahl an der Jobs $j, k = \{0, 1, 2, \dots, n\}$, wobei es sich bei Job 0 um einen Dummy Job zur Identifizierung des ersten auf einer Maschine zu prozessierenden Jobs handelt
- m — Anzahl der Maschinen $i, l = \{1, 2, \dots, m\}$
- f — Anzahl der Produktionsstätten $r = \{1, 2, \dots, f\}$
- $p_{j,i}$ — Porzessierungszeit von Job j auf Maschine i

Nach Pinedo sollte das Scheduling Problem durch ein Triplet aus Maschinenumgebung, Prozesscharakteristiken und Zielfunktion beschrieben werden. Im Falle des Distributed Job Shops kann das Problem wie folgt beschrieben werden:

$$J_{f,m} || C_{max} \tag{3.1}$$

Im Gegensatz zum klassischen Job Shop Problem, was durch $J_m || C_{max}$ beschrieben wird, muss im Distributed Job Shop die Produktionsstätte f ebenfalls mit in Betracht gezogen werden.

4 Verfahrenslösung

4.1 Regel-basierte Heuristiken

Bei den Regel-basierten Heuristiken wird, wie im ersten Kapitel erklärt, in zwei Schritten vorgegangen.

4.1.1 Aufteilung von Jobs auf Produktionsstätten

Bei der Aufteilung der Jobs auf die Produktionsstätte werden folgende Aspekte mit in Betracht gezogen:

- Einbeziehung der Prozesszeiten je Auftrag j auf den verschiedenen Maschinen als Kennzahl für Arbeitspensum
- Möglichst gleichmäßige Aufteilung der Prozesszeiten auf die jeweiligen Maschinen der Produktionsstätten (Ungleichverteilung kann zu schlechter Kapazitätsauslastung führen)
- Jobs mit gleichen Routen sollten auf verschiedene Produktionsstätten verteilt werden, da sie die Fertigungsdauer erhöhen können durch potenziell erhöhte Standzeiten

Diese Punkte wurden in der entwickelten Formel für die Kalkulation des Arbeitspensums berücksichtigt, welches als Grundlage für Verteilung der Jobs dient¹:

$$\text{Arbeitspensum}(j, i) = \left(\sum_{k \in R_{j,i}} p_{j,k} \right) + p_{j,i} \quad \forall_{ij} \quad (4.1)$$

wobei $R_{j,i}$ dem Satz aller der Maschine i vorgeschalteten Maschinen aus Job j entspricht. Bei der Verteilung von Jobs auf Fabriken soll das maximale Arbeitspensum je Fabrik und Maschine minimiert werden.

¹NADERI / AZAB (2014): *Modeling and heuristics for scheduling of distributed job shops*, S. 7756.

²Eigene Abbildung

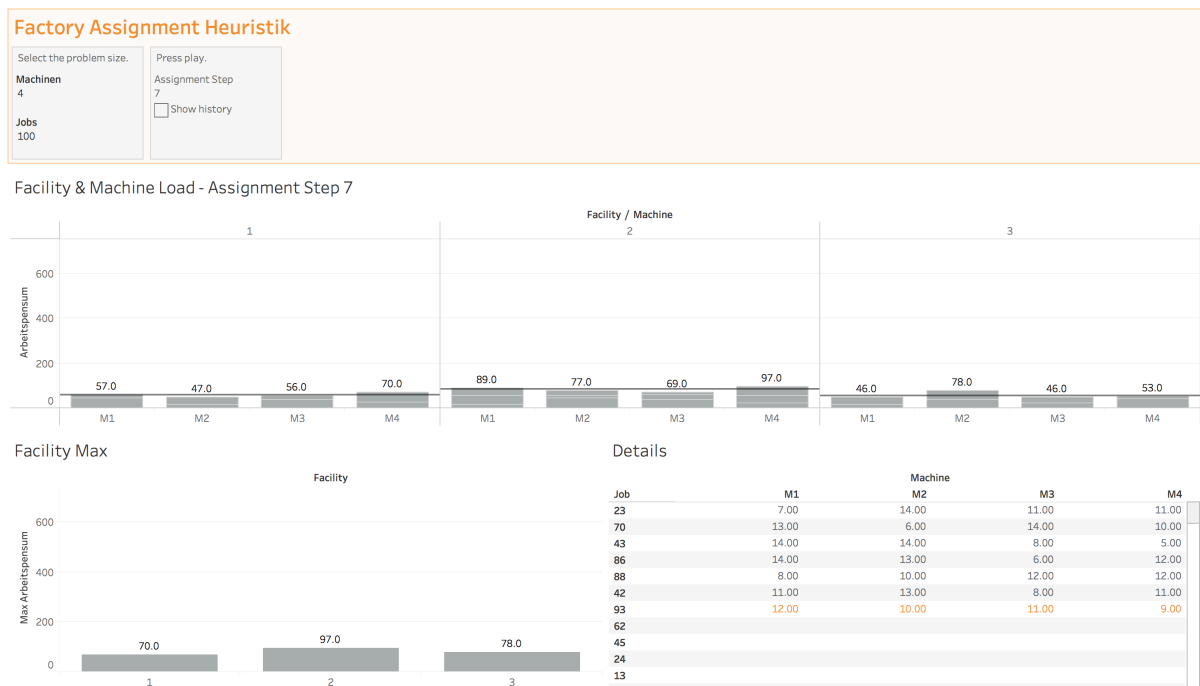


Abbildung 4.1: Graphische Aufbereitung der Zuordnung von Aufträgen zu Produktionsstandorten²

4.1.2 Einplanung der Aufträge auf Maschinen

Mithilfe der bekannten Heuristiken Shortest Processing Time (SPT) und Longest Processing Time (LPT) (LPT) werden die den Produktionsstätten zugeordneten Jobs auf den einzelnen Maschinen eingeplant.

4.1.3 Anwendungsbeispiel

Im folgenden Problem sollen $j = 4$ Jobs auf $f = 2$ Produktionsstätten mit jeweils $m = 3$ Maschinen verteilt werden. In Tabelle 4.1 sind die Prozesszeiten der Jobs auf den Maschinen und deren Route angegeben.

Job	M1	M2	M3	Route
1	3	7	4	{3, 2, 1}
2	6	5	3	{2, 1, 3}
3	12	1	6	{1, 3, 2}
4	2	9	5	{3, 2, 1}

Tabelle 4.1: Jobs und Operationsfolge für das Anwendungsbeispiel

Aufteilung von Jobs auf Produktionsstätten

Mithilfe der Formel für Arbeitspensum kann dieses leicht errechnet werden. In der Tabelle 4.2 wird zusätzlich die Summe des Arbeitspensums für jeden Job berechnet und ein Rank nach absteigender Summe des Arbeitspensums vergeben.

Job	M1	M2	M3	Summe	Rank
1	$11 + 3 = 14$	$4 + 7 = 11$	$0 + 4 = 4$	29	4
2	11	5	14	30	3
3	12	19	18	49	1
4	16	14	5	35	2

Tabelle 4.2: Ermitteltes Arbeitspensum und Rank

Nach Ermittlung des Arbeitspensums und Ranks kann jetzt mit der Aufteilung der Jobs auf die einzelnen Produktionsstätten begonnen werden.

Iteration 1

Verteilung der ersten f Jobs nach dem ermittelten Rank auf die Produktionsstätten und Bestimmung des maximalen Workloads auf der Produktionsstätte.

- Job $j = 3$ wird in Produktionsstätte $f = 1$ gefertigt — maximaler Workload beträgt 19.
- Job $j = 4$ wird in Produktionsstätte $f = 2$ gefertigt — maximaler Workload beträgt 16.

Die in dieser Iteration ermittelten Ergebnisse werden in Tabelle 4.3 eingetragen.

Iteration 2 ... x

1. Ermittlung des nächsten nichtverteilten Jobs nach dem ermitteltem Rank.
2. Addition vom Arbeitspensum des Jobs zu dem jeweiligen bereits verteilten Arbeitspensum für die Produktionsstätten.
3. Ermittlung des Maximum des Arbeitspensums pro Produktionsstätte.
4. Verteilung des Jobs auf die Produktionsstätte mit dem minimalen Maximum.

Iteration	Produktions- stätte	Job	M1	M2	M3	Max	Auswahl
1	1	3	12	19	18	19	x
	2	4	16	14	5	16	x
2	1	2	12 + 11	19 + 5	18 + 14	32	
			23	24	32		
	2	2	16 + 11	14 + 5	5 + 14	27	x
			27	19	19		
3	1	1	12 + 14	19 + 11	18 + 4	30	x
			26	30	22		
	2	1	27 + 14	19 + 11	19 + 4	41	
			41	30	23		

Tabelle 4.3: Arbeitstabelle zur Aufteilung von Jobs auf Produktionsstätten

Daraus ergibt sich, dass die Jobs wie folgt auf die Produktionsstätte verteilt werden:

1. Jobs: 3, 1
2. Jobs: 4, 2

Zu Beginn des Kapitels wurde als Merkmal einer guten Jobzuordnung angemerkt, dass Jobs mit identischen Routen in unterschiedlichen Produktionsstätten gefertigt werden sollten. In diesem Beispiel wird sehr schön sichtbar, dass durch das im Paper entwickelte Vorgehen Jobs 1 & 4 mit identischen Routen unterschiedlichen Produktionsstätten zugeordnet wurden. So kann die Fertigungsdauer minimiert werden.

Einplanung der Aufträge auf Maschinen mittels SPT

Zuerst wird Tabelle 4.4 erstellt und für jede Maschine nach der Prozesszeit aufsteigend sortiert.

Iteration 1 ... x

Jeweils die erste Zeile pro Maschine betrachten. Sind alle vorhergehenden Operationen abgeschlossen?

1. Ja? Das Maximum aus (Maximale Endzeit der Maschine, Maximale Endzeit des Jobs) ist die neue Startzeit. Die neue Endzeit ergibt sich aus der neuen Startzeit + Prozesszeit. Fortfahren mit nächster Maschine.

2. Nein? Fortfahren mit nächster Zeile der Maschine.

Nach Durchführung des ersten Iterationsschrittes ergibt sich Tabelle 4.4.

Machine	Job	Operation	Prozesszeit	Startzeit	Endzeit
M1	1	3	3	x	
M1	3	1	12		
M2	3	3	1	x	
M2	1	2	7		
M3	1	1	4	0	4
M3	3	2	6		

Tabelle 4.4: Einplanung von Jobs nach der SPT Regel - Erster Iterationsschritt

Nach Durchführung aller Iterationsschritte ergibt sich Tabelle 4.5.

Machine	Job	Operation	Prozesszeit	Startzeit	Endzeit
M1	1	3	3	11	14
M1	3	1	12	0	12
M2	3	3	1	18	19
M2	1	2	7	4	11
M3	1	1	4	0	4
M3	3	2	6	12	18

Tabelle 4.5: Einplanung von Jobs nach der SPT Regel - Ergebnis

In der Präsentation wurde zu Illustrationszwecken unter anderem folgendes interaktives Beispiel verwendet:

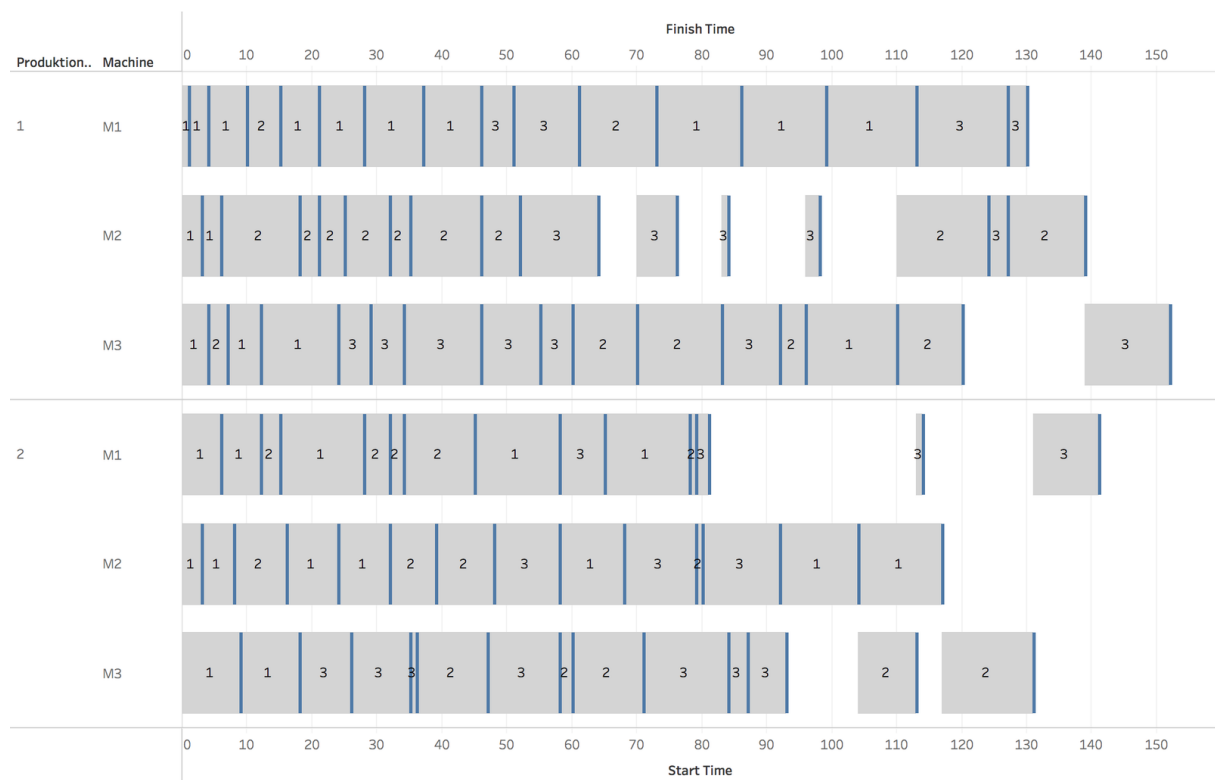


Abbildung 4.2: Darstellung der nach SPT eingeplanten Operationen im Gantt-Diagramm³

Einplanung der Aufträge auf Maschinen mittels LPT

Die Schritte für LPT sind identisch zu den von SPT mit der Ausnahme, dass die Tabelle 4.4 zu Beginn nicht aufsteigend, sondern absteigend sortiert wird.

In der Präsentation wurde zu Illustrationszwecken unter anderem folgendes interaktives Beispiel verwendet:

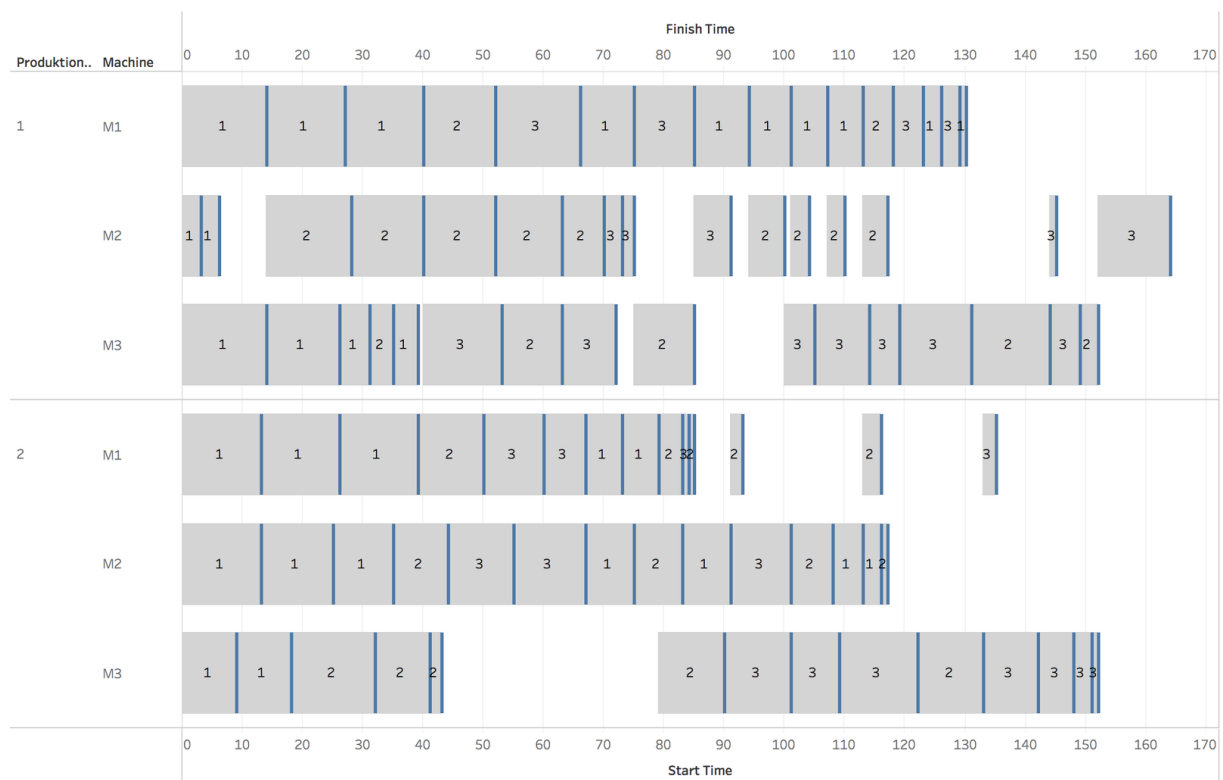


Abbildung 4.3: Darstellung der nach LPT eingeplanten Operationen im Gantt-Diagramm⁴

4.2 Greedy Heuristiken

In NADERI / AZAB 2014: *Modeling and heuristics for scheduling of distributed job shops* werden weiterhin drei hoch-performante Greedy-Heuristiken eingeführt. Diese Heuristiken beruhen auf dem “insertion neighborhood concept”. D.h. es wird durch kontinuierliches Einsetzen eines Prozessschrittes an verschiedenen Stellen in der Prozesskette die Reihenfolge ermittelt, welche zur kürzesten Fertigungsdauer führt.

4.2.1 Greedy-Heuristik 1

Allgemeines Vorgehen

Schritt 1. Zuerst wird, wie im Abschnitt 4.1.1 eingeführt, die Zuteilung von Jobs auf die einzelnen Produktionsstätten mithilfe der Kenngröße des Arbeitspensums durchgeführt.

⁴Eigene Abbildung

Schritt 2. Für jede Produktionsstätte wird eine zufällige Reihenfolge der Operationen (Operationsfolge) bestimmt, wobei eine spezielle Array-Datenstruktur verwendet wird. Anstatt sowohl Job, Machine und Operationsnummer zu verwenden, wird nur die Jobnummer verwendet welche m Mal wiederholt wird. Durch die Position und die Anzahl der Vorkommnisse der Jobnummer vorher, kann wieder auf Maschine und Operationsnummer zurückgeschlossen werden. Produktionsstätte 1, aus dem vorher vorgestelltem Beispiel, wurden die Jobs (1,3) zugewiesen, wobei jeder der Jobs auf drei Maschinen bearbeitet wird. Somit ergibt sich, dass beide Jobnummer jeweils drei Mal vorkommen müssen. Die zufällige generierte Reihenfolge der Operationen könnte daher z.B. [1, 3, 1, 1, 3, 3] lauten. Das würde bedeuten, dass zuerst Job 1 auf Maschine M3 bearbeitet wird, bevor mit der Bearbeitung von Job 3 auf Maschine M1 begonnen wird.

Schritt 3. Für jede Produktionsstätte wird der Länge des Arrays entsprechend viele Iterationen durchgeführt. Der Laufindex i wird für die Iterationszahl eingeführt und es wird die Variable `best_seq` im Programmcode verwendet, in der die beste ermittelte Operationsfolge gespeichert wird. Die Jobindizes werden nacheinander aus der ursprünglichen Operationsfolge genommen und in alle möglichen Positionen in der Operationsfolge getestet. Schließlich wird die beste Position ausgewählt. Die Prozedur wird für den nächsten Job in der Operationsfolge wiederholt.

Anwendungsbeispiel

Zufällige Operationsfolge: [1, 3, 1, 1, 3, 3]

Iteration 1:

Es wird der Job an erster Stelle der Operationsfolge ermittelt — Job 1. Es gibt keine weiteren Jobindizes in `best_seq`, weshalb [1] als neue `best_seq` gespeichert wird.

Iteration 2:

1. Ermittle den Jobindex an der zweiten Stelle der originalen Operationsfolge. Im Beispiel wäre dies Job 3.
2. Da `best_seq = 1` entspricht ergeben sich folgende Permutationen: {[1, 3], [3, 1]}. Angenommen, dass Sequenz [1, 3] die bessere ist — `best_seq` entspricht nun [1, 3].

Iteration 3:

1. Ermittle den Jobindex an der dritten Stelle der originalen Operationsfolge. Im Beispiel ist dies Job 1.

2. Da bereits im vorherigen Schritt festgestellt wurde, dass $[1, 3]$ die beste Sequenz war, müssen nicht mehr alle Permutationen getestet werden. Stattdessen müssen nur noch Permutationen getestet werden, die durch Einsetzen des Jobindex an jeder Position der `best_seq` generiert werden. Es ergeben sich daher folgende relevante Permutationen: $\{[1, 1, 3], [1, 1, 3], [1, 3, 1]\}$.
3. Es werden wieder die Permutationen getestet und die Beste wird in `best_seq` gespeichert.

Iteration 4 ... x:

Es werden die Schritte aus Iteration 3 wiederholt.

Testen von Operationsfolgen

Operationsfolgen werden wie folgt eingeplant und getestet:

1. Über die gegebene Operationsfolge wird durch die Position des Jobindex und Anzahl der vorher vorkommenden identischen Jobindizes die Maschine ermittelt. Basierend auf Operationsfolge $[1, 1, 3]$ kann auf folgende Informationen geschlossen werden:

Job	Machine	Operation	Prozesszeit
1	M3	1	4
1	M2	2	7
3	M1	1	12

Tabelle 4.6: Aus der Operationsfolge extrahierte Informationen

2. In der durch die Operationsfolge bestimmten Reihenfolge werden die Jobs auf den Maschinen eingeplant. Der Startzeitpunkt auf der jeweiligen Maschine entspricht dem Maximum des letzten Endzeitpunktes der Maschine und des Jobs.

Job	Machine	Operation	Prozesszeit	Startzeit	Endzeit
1	M3	1	4	0	4
1	M2	2	7	4	11
3	M1	1	12	0	12

Tabelle 4.7: Einplanung der Jobs nach der ermittelten Operationsfolge

3. Der Maximale Endzeitpunkt entspricht der Fertigungsdauer. Es wird die Sequenz mit der geringsten Fertigungsdauer in jeder Iteration gewählt.

Im Folgenden findet sich ein Screenshot für das in der Präsentation verwendete interaktive Beispiel:

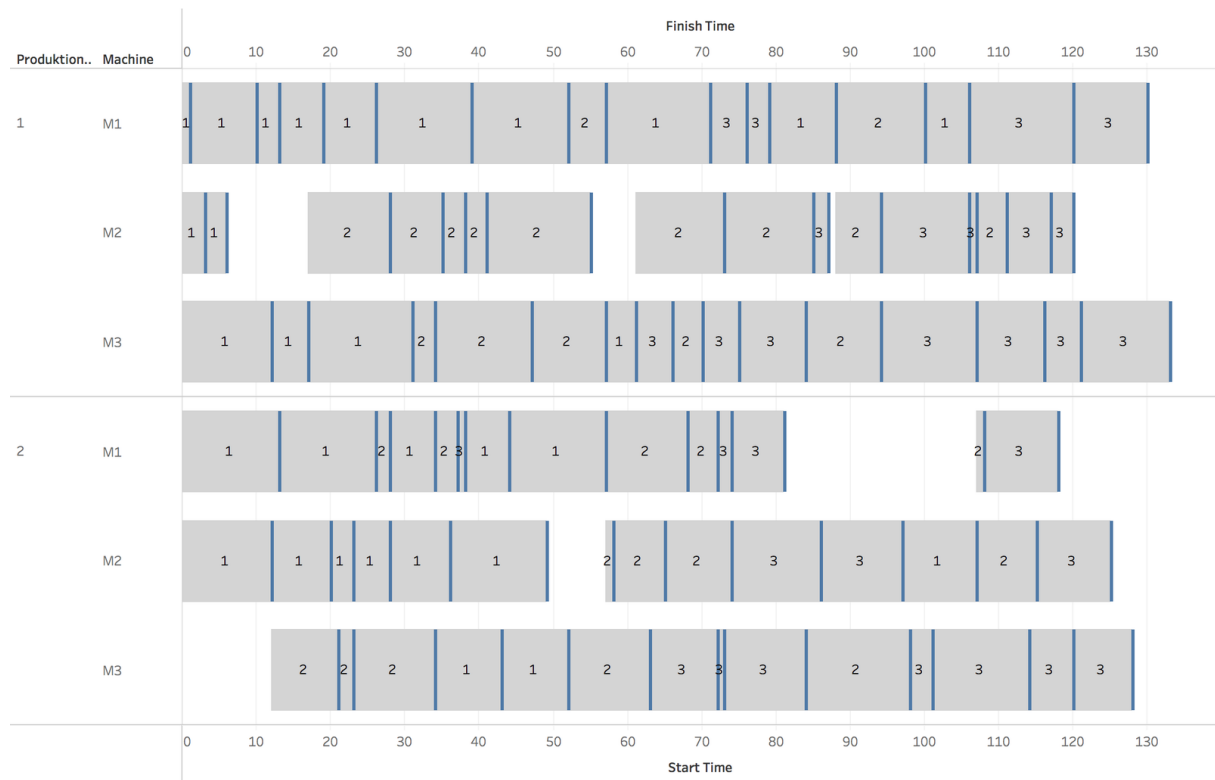


Abbildung 4.4: Darstellung der nach der ersten Greedy Heuristik eingeplanten Operationen im Gant-Diagramm⁵

4.2.2 Greedy-Heuristik 2

Die zweite Heuristik funktioniert ähnlich der ersten Heuristik. Der einzige Unterschied ist die Zuordnung der Jobs zu den Produktionsstätten.

Schritt 1. Zuerst wird die Liste der Jobs absteigend nach der Fertigungsdauer sortiert. Die ersten f Jobs werden jeweils einer Produktionsstätte zugeordnet.

Schritt 2. Die restlichen Jobs werden weiter in der absteigenden Reihenfolge bearbeitet und jeweils der Produktionsstätte mit der bisher niedrigsten gesamten Fertigungsdauer

⁵Eigene Abbildung

zugeordnet. Es wird nun der dritte Schritt aus der ersten Greedy-Heuristik angewendet.

Im Folgenden findet sich ein Screenshot für das in der Präsentation verwendete interaktive Beispiel:

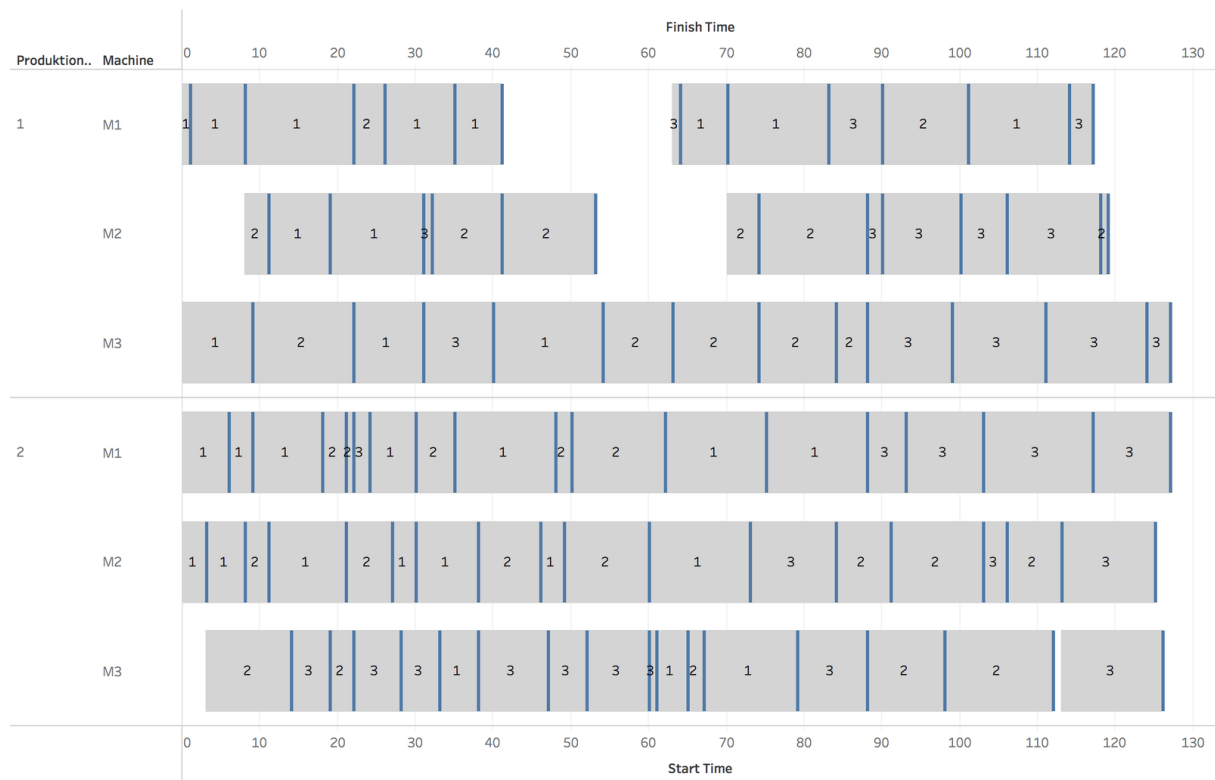


Abbildung 4.5: Darstellung der nach der zweiten Greedy Heuristik eingeplanten Operationen im Gant-Diagramm⁶

4.2.3 Greedy-Heuristik 3

Die dritte Heuristik funktioniert ebenfalls ähnlich der ersten Heuristik. Der Unterschied ist, dass alle Permutationen inklusive die Zuordnung zu allen Produktionsstätten in jeder Iteration getestet wird.

Schritt 1. Zuerst wird die Liste der Jobs erstellt. Dabei erfolgt keine Sortierung.

Schritt 2. Für jede Kombination aus Job und Fabrik wird der dritte Schritt aus der ersten Greedy Heuristik angewendet. Der Job wird der Fabrik zugeordnet, dessen Fertigungsdauer

⁶Eigene Abbildung

am Ende minimal ist.

Im Folgenden findet sich ein Screenshot für das in der Präsentation verwendete interaktive Beispiel:

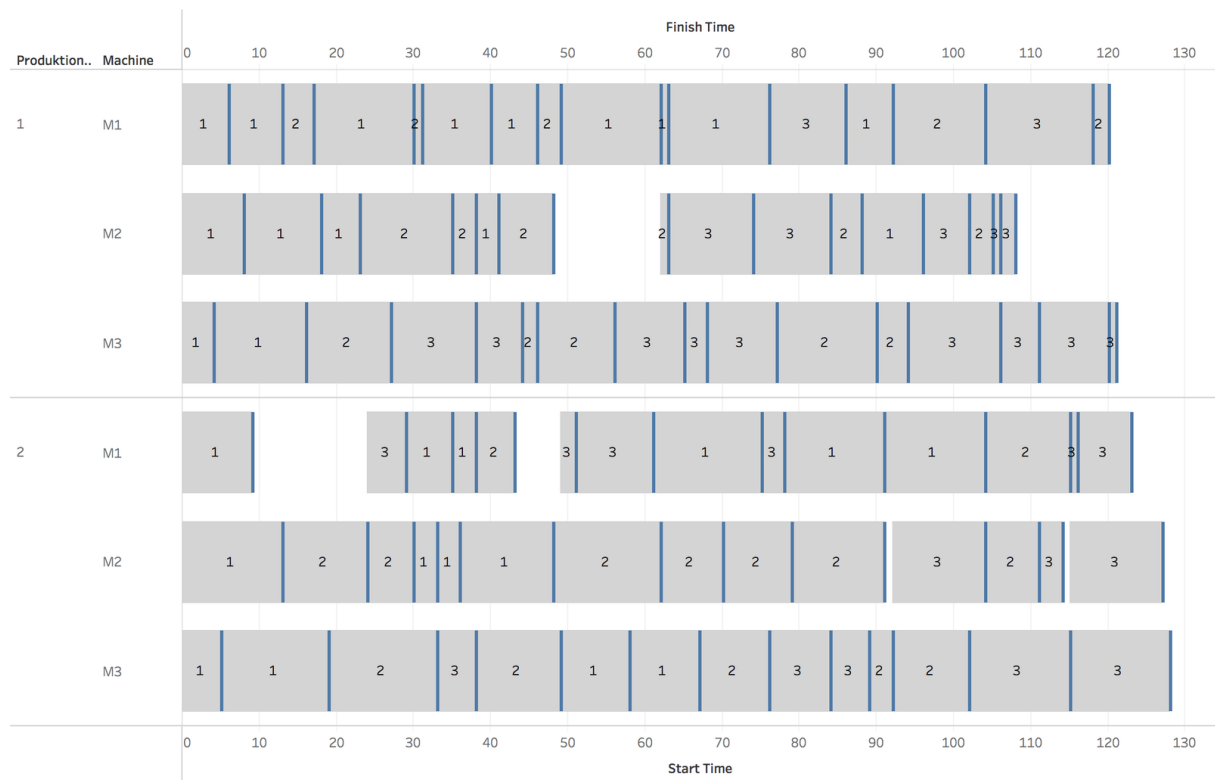


Abbildung 4.6: Darstellung der nach der dritten Greedy Heuristik eingeplanten Operationen im Gant-Diagramm⁷

⁷Eigene Abbildung

A Programmcodes

Der Programmcode, welche im Rahmen dieser Arbeit entwickelt wurde, wurde auf GitHub (<https://git.io/vhTwq>) veröffentlicht und kann unter der MIT License kostenfrei weiter- und wiederverwendet werden.

Literaturverzeichnis

BABBAGE, C. (2005): From The Economy of Machinery and Manufactures. In: *Readings In The Economics Of The Division Of Labor: The Classical Tradition*. World Scientific, S. 131–148.

NADERI, B. / AZAB, A. (2014): Modeling and heuristics for scheduling of distributed job shops. In: *Expert Systems with Applications*, Vol. 41, Nr. 17, S. 7754–7763.

PINEDO, M. L. (2016): Scheduling: theory, algorithms, and systems. Springer.