technische universität dortmund

IT in Produktion und Logistik

Technische Universität Dortmund
Fakultät Maschinenbau
Fachgebiet ITPL
Univ.-Prof. Dr.-Ing. Markus Rabe

Ant-Colony-Optimierung zur Variation modular modellierter Produktionssysteme und zum Scheduling von Auftragsreihenfolgen

---

Bachelorarbeit
Juli 2017

Bearbeiter: Dzemal Alic, Achmed Junusov
Matrikel-Nr.: 167000, 166397

Betreuer: Dipl.-Geoinf. Maik Deininger

Inhaltsverzeichnis

Abkürzungen und Formelzeichen

Abkürzung	Bezeichnung
A	Gesamtanzahl an Lösungsmöglichkeiten
A_{ges}	Gesamtanzahl der Arbeitsschritte
A_i	Arbeitsschritt *i*
ACO	Ant-Colony-Optimierung
ACS-Algorithmus	Ant-Colony-System-Algorithmus
AS	Arbeitsstation
AS-Algorithmus	Ant-System-Algorithmus
$FFWT$	Flow-Faktor
FW	Fitnesswert
$f(s^k)$	Wert der generierten Lösung
g	Anzahl der Gabelstapler
GPS	Ganzheitliches Produktionssystem
GS	Gabelstapler
J_i	Job *i*
m	Gesamtanzahl der Maschinen
M_k	Maschine *k*
MMAS-Algorithmus	MAX-MIN Ant-System-Algorithmus
N	Netzstruktur
n	Anzahl der Entscheidungsmöglichkeiten pro Station
N_i	Nachbarschaft des Knoten *i*
P	Plätze
${P_{ij}}^k$	Übergangswahrscheinlichkeit von Knoten *i* zu Knoten *j*
Q	Güte der Lösung
q	Zufällig generierte Zahl bei dem ACS-Algorithmus
q_0	Vom Anwender festgelegte Zahl bei dem ACS-Algorithmus
T	Transitionen
THORNs	Zeitbehaftete Hierarchische Objektorientierte Netze
V	Knotenmenge
v	Zug
X	Lösungsraum
x	Lösung
x'	Neue Lösung
Z_{ges}	Gesamtzykluszeit
α	Parameter für die Pheromonmenge
β	Parameter für die heuristische Information
ξ	Pheromonmenge bei dem lokalen Pheromonupdate
ϱ	Pheromonverdunstung
τ_0	Grundpheromonmenge

Abkürzung	Bezeichnung
τ_{ij}	Pheromonmenge von Knoten *i* zu Knoten *j*
τ_{max}	Maximale Pheromonmenge
τ_{min}	Minimale Pheromonmenge
$\Delta\tau_{ij}{}^{best}$	Pheromonmenge des besten Pfades
$\Delta\tau_{ij}{}^{k}$	Pheromonmenge der Ameise *k*
η_{ij}	Heuristische Information
$\emptyset FFWT$	Durchschnittlicher Flow-Faktor

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Im Zuge der Globalisierung herrscht ein stetig steigender Wettbewerb zwischen Unternehmen. Die Produktionsplanung hat sich dabei als ein wichtiger Bestandteil der Produktion und Logistik etabliert. Das moderne Unternehmensumfeld ist durch Marktturbulenzen gekennzeichnet. Aufgrund dieser Marktturbulenzen wird eine Wandlungs- und Anpassungsfähigkeit von Unternehmen gefordert, um eine möglichst schnelle und zeiteffiziente Anpassung des Produktionssystems durchführen zu können. So kann gewährleistet werden, dass das Unternehmen auf dem Markt konkurrenzfähig bleibt (vgl. Rauch, 2013). Anpassungen sind meist nur durch Experten mit umfangreichem Wissen über das Produktionssystem möglich. Zur Unterstützung von Anpassungen wird die Simulation als Hilfsmittel genutzt. Dabei wird durch die Anwendung von heuristischen Optimierungsverfahren eine optimale Lösung binnen einer bestimmten Zeit geliefert (vgl. Ehrgott, 2004). Heuristische Optimierungsverfahren unterstützen die Wahl von Entscheidungen mit hoher Tragweite, deren Folgen nicht ohne weiteres absehbar sind (vgl. Rabe, 2008). Durch eine schnell generierte Lösung wird dem Unternehmen ein Marktvorteil verschafft und dadurch das Potential zur Profitsteigerung erhöht (vgl. Rauch, 2013).

Das Ziel dieser Bachelorarbeit ist die Entwicklung und anschließende Umsetzung einer Ant-Colony-Optimierung (ACO), welche zu den heuristischen Optimierungsverfahren zählt. Dafür soll eine, für das vorliegende Produktionssystem, geeignete Ausführung der Ant-Colony-Optimierung gefunden werden. Diese Ausführung muss in ein vorgegebenes Simulationsprogramm implementiert werden. Dabei setzt sich das Simulationsprogramm aus zwei Unterprogrammen zusammen. Das erste Unterprogramm simuliert die Variation eines modular modellierten Produktionssystems. Diese soll eine Bewertung eines modularen Produktionssystems ermöglichen, das auf Zeitbehafteten Hierarchischen Objektorientierten Netzen (THORNs) basiert. Dabei wird die Anzahl der Maschinen im Produktionssystem variiert. So soll eine Maschinenkonstellation gefunden werden, die die Effizienz des Produktionssystems steigert. Die Effizienz beschreibt hierbei eine geringe Gesamtzykluszeit in Kombination mit einer hohen Maschinenauslastung. Das zweite Unterprogramm wird für das Scheduling von Auftragsreihenfolgen genutzt. Dabei soll eine Ant-Colony-Optimierung entwickelt werden, die eine optimale Reihenfolge von Einträgen in einer gegebenen Auftragsliste liefert. Durch die Variation der Auftragsreihenfolge soll die Gesamtzykluszeit der Auftragslisten für eine feste Maschinenkonstellation minimiert werden.

In dem zweiten Kapitel wird die zeitliche Entwicklung von Produktionssystemen, von D. Alic, erläutert. Dabei wird insbesondere das modulare Produktionssystem beschrieben. Außerdem werden das Job-Shop-Scheduling-Problem und geeignete Auswertungsmethoden erklärt. In dem dritten Kapitel werden die Grundlagen der Petri-Netze und THORNs, von D. Alic,

beschrieben. Dies ist erforderlich, da es sich bei dem vorgegebenen Simulationsprogramm zwar um eine Black-Box handelt, diese jedoch auf THORNs basiert. In dem vierten Kapitel erfolgt, von A. Junusov, die ausführliche Einführung in die Optimierungsverfahren. Dabei werden, nach der Unterteilung in exakte- und heuristische Optimierungsverfahren, ausgewählte Optimierungsverfahren näher betrachtet. Nachdem die essentiellen Unterschiede der beiden Optimierungsverfahren erläutert worden sind, werden metaheuristische Optimierungsverfahren vorgestellt. Das fünfte Kapitel befasst sich mit der umfassenden Beschreibung der ACO-Metaheuristik. Dabei werden, von A. Junusov, die unterschiedlichen Ausführungsmöglichkeiten und Pseudo-Codes der ACO-Metaheuristik untersucht und miteinander verglichen. Ab dem sechsten Kapitel werden die von D. Alic und A. Junusov gemeinsam erarbeiteten Themenbereiche dargestellt. In dem sechsten Kapitel wird das vorgegebene Produktionssystem vorgestellt. Daraufhin werden in dem siebten Kapitel unterschiedliche ACO-Metaheuristiken gegenübergestellt und hinsichtlich der Eignung für die Optimierungsprobleme verglichen. Anschließend wird für die Variation eines vorgegebenen Produktionssystems und für das Scheduling von Auftragsreihenfolgen die Wahl eines geeigneten Ameisenalgorithmus getroffen. Diese Algorithmen werden in das Simulationsprogramm implementiert und auf das Produktionssystem angewendet. Dies geschieht unter der Verwendung der Programmiersprache C++. Um Aussagen über das Verhalten der Algorithmen tätigen zu können, werden im achten Kapitel spezifische Parameter der ausgewählten ACO-Metaheuristik variiert und analysiert. Nach der Analyse der Ergebnisse werden diese diskutiert. Auf dieser Grundlage wird eine Bewertung der entwickelten ACO-Metaheuristik durchgeführt. Zum Abschluss werden in dem neunten Kapitel alle Ergebnisse zusammengetragen, zudem werden künftige Anpassungen und Modifikationen der ACO-Metaheuristik vorgeschlagen.

2 Produktionsplanung

Die Produktionsplanung ist gemeinsam mit der Produktionssteuerung ein Teilbereich des operativen Produktionsmanagements. Die Hauptaufgabe ist die Gewährleistung eines zeiteffizienten Ablaufs der Produktionsprozesse (vgl. Kistner und Steven, 2001). Die Produktionsplanung wird in die strategische, taktische und operative Produktionsplanung aufgeteilt. Dabei steht in diesem Kapitel die operative Produktionsplanung im Fokus. Die operative Produktionsplanung beschäftigt sich mit der optimalen Nutzung der vorhandenen Kapazitäten, um die Wirtschaftlichkeit der Produktion zu gewährleisten (vgl. März et al., 2011). Dabei steht die Optimierung der Produktionssysteme im Vordergrund. In diesem Kapitel werden ganzheitliche und modulare Produktionssysteme behandelt und miteinander verglichen.

2.1 Produktionssysteme

Ein Produktionssystem ist wie folgt definiert:

> *„Ein Produktionssystem besteht aus Hard- und Software. Als Hardware gelten dabei die Produktionsanlagen sowie die Lager-, Umschlag-, und Transporteinrichtungen. Als Software gilt das Regelwerk zur Gestaltung, Führung und Anpassung des Produktionsablaufs, um ein effizientes Zusammenspiel zwischen Organisation, Ressourcen, Menschen und Methoden zu gewährleisten.“* (vgl. Günther und Tempelmeier,1995)

Aus der Definition geht hervor, dass ein Produktionssystem nicht nur Elemente zum technischen Fertigungsprozess eines Produktes, sondern auch organisatorische Elemente zur Planung und Steuerung des Produktionsprozesses umfasst. Ziel der Unternehmen ist es mit Hilfe der Einführung und Entwicklung von Produktionssystemen ihre Produktivität und Effizienz zu steigern (vgl. Wildemann, 2004). Die Anforderungen und Erwartungen an das Produktionssystem haben sich jedoch im Laufe des 20. Jahrhunderts, aufgrund der Globalisierung und des technischen Fortschrittes, oft geändert. So dominierte die handwerkliche Produktion bis in die 20er Jahre vorherrschend. Ab den 20er Jahren ist die klassische industrielle Produktion durch die industrielle Revolution in den Vordergrund getreten. In den 50er Jahren sind dann schließlich die ersten Computer entwickelt worden, welche auch in die Produktion eingebunden worden sind. Mit diesem technischen Fortschritt ist auch eine große Wandlung in der Produktion bewirkt worden, da es durch die Nutzung von Computern zur Steuerung von Maschinen zum ersten Mal möglich gewesen ist, eine automatisierte Produktion zu betreiben (vgl. Rauch, 2013). Während in Europa und in den USA immer mehr Computer in der Produktion eingesetzt worden sind, ist in Japan das Toyota

Produktionssystem aus einer Notwendigkeit heraus entstanden (vgl. Ohno, 1993). Es ist eine Notwendigkeit, da sich Japan nach dem zweiten Weltkrieg in einer Wirtschaftskrise befunden hat. Der japanische Markt hat eine große Variantenvielfalt von den Unternehmen gefordert, wobei nicht genügend finanzielle Mittel für eine Massenproduktion zur Verfügung gestanden haben, wie es in Europa und in den USA der Fall gewesen ist. Aus diesem Grund ist Toyota gezwungen gewesen die Vermeidung von Verschwendung in den Vordergrund zu stellen (vgl. Rauch, 2013). Durch die Effizienz des Toyota Produktionssystems sind auch im europäischen Raum große Teile des Toyota Produktionssystems in eigene Produktionssysteme integriert worden. Das Resultat dieser Kombination verschiedener Produktionssysteme ist das Ganzheitliche Produktionssystem (GPS), welches sich in den 90er Jahren etabliert hat (vgl. Korge und Scholtz, 2004). Dieses Produktionssystem ist bis heute erhalten geblieben, wobei der mittlerweile globale Markt immer mehr Flexibilität und Wandlungsfähigkeit von den Unternehmen verlangt (vgl. Wiendahl et al., 2009). Im Folgenden wird auf die beiden aktuellsten Produktionssysteme eingegangen. So werden ganzheitliche und modulare Produktionssysteme näher erklärt und miteinander verglichen.

Ganzheitliche Produktionssysteme

Das Ganzheitliche Produktionssystem ist nach der deutschen MTM Vereinigung e.V wie folgt definiert:

> *„Als Ganzheitliches Produktionssystem (GPS) wird ein Modell zur Generierung einer den gesamten Wertschöpfungsprozess überdeckenden Gesamtheit von Standards zum Gestalten und Betreiben von Arbeitssystemen bezeichnet. Beim GPS werden interne Strategien (Wertschöpfungsstrategien) operationalisiert, indem die drei Ordnungskriterien Funktionen (funktionale Strategien), Ressourcen (Ressourcenstrategien) und Handlungsfelder so weit konkretisiert werden, bis dazu praktisch nützliche Standards und Regeln zu formulieren sind.“* (MTM, 2012)

Im vorherigen Kapitel ist die zeitliche Entwicklung von Produktionssystemen vorgestellt worden. Dabei ist das Ganzheitliche Produktionssystem zunächst in den 90er Jahren aufgetreten. Mittlerweile haben viele Unternehmen eigene Ganzheitliche Produktionssysteme aus dem Toyota Produktionssystem adaptiert. Diese eignen sich nun nicht mehr lediglich für die großen Unternehmen, sondern sind auch für kleine und mittelständische Unternehmen geeignet und werden teilweise sogar als Notwendigkeit angesehen (vgl. Feggeler und Neuhaus 2002). Das Ganzheitliche Produktionssystem wird auch außerhalb der Automobilindustrie in Unternehmen eingebunden. Um ein besseres Verständnis über Ganzheitliche Produktionssysteme zu vermitteln, wird zunächst der Begriff der

„Ganzheitlichkeit“ näher untersucht. Die folgenden vier Punkte sind charakteristisch für die „Ganzheitlichkeit“ (vgl. Spath, 2003):

1. **Umfassend**: Sämtliche Aufgaben, welche bei der Produktion auftreten können, werden in dem Produktionssystem berücksichtigt und etwaige Wege zur Lösung vorgeschlagen.
2. **Durchgängig**: Der Prozess ist lückenlos. Den Mitarbeitern sind alle erforderlichen Randbedingungen gegeben und bekannt. Darunter zählt die Qualifikation der Mitarbeiter, die notwendigen Befugnisse, sowie alle Werkzeuge und Hilfsmittel, welche benötigt werden, um die Prozesse durchführen zu können.
3. **Bruchlos**: Innerhalb des Unternehmens treten keine organisatorischen Lücken beziehungsweise Brücken auf. Unter Brüchen werden Unstimmigkeiten im Zeitmanagement zwischen der Produktion und der Konstruktion verstanden.
4. **Alle Aspekte berücksichtigend**: Die drei wichtigsten Komponenten in der Wertschöpfungskette werden berücksichtigt: Mensch, Organisation und Technik.

Wie bereits angedeutet ist das Ganzheitliche Produktionssystem ein Resultat aus der Modifikation des Toyota Produktionssystems, wobei ein Großteil der beinhalteten Aspekte identisch bleibt. In der Abbildung 2.1 ist eine schematische Darstellung des Toyota Produktionssystems abgebildet (vgl. Wienholdt und Meyer, 2007).

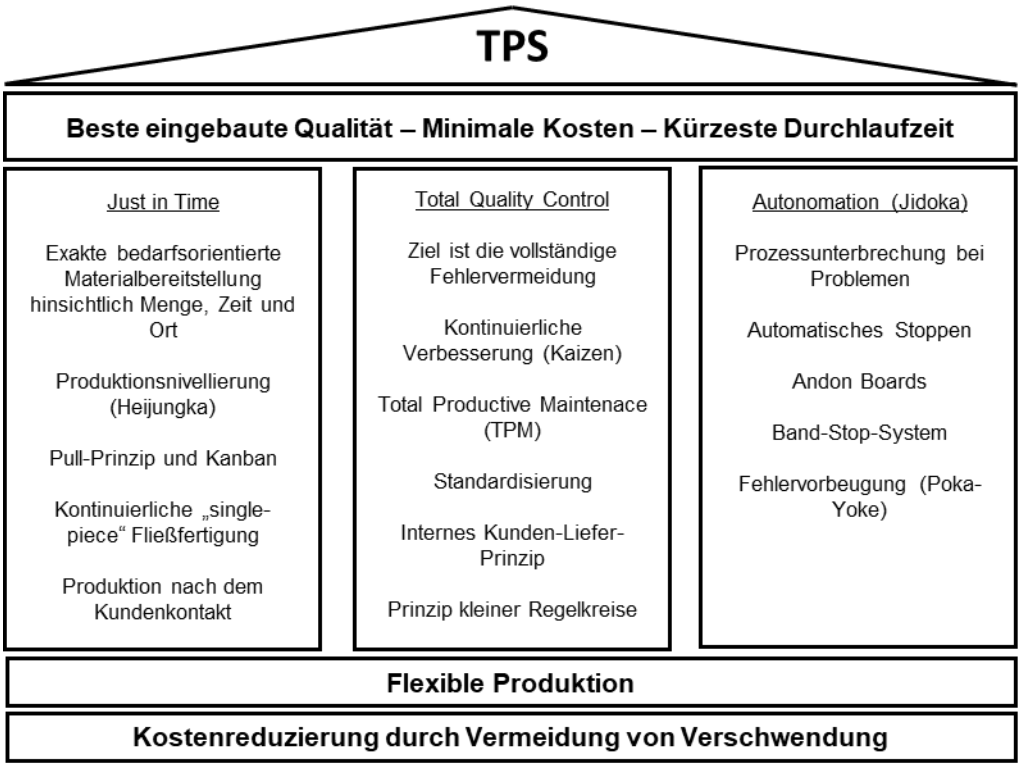


Abbildung 2.1: Toyota Produktionssystem

Wie zu erkennen ist, setzt sich das Fundament aus einer flexiblen Produktion und aus der Kostenreduzierung durch Vermeidung von Verschwendung zusammen. Das Fundament ist der wichtigste Bestandteil eines Gebäudes, ohne welchen kein weiterer Ausbau möglich ist. Diese zwei Aspekte sind durch den vom zweiten Weltkrieg geschwächten japanischen Markt vorgegeben worden und sind die Grundlage für die Säulen. Die drei Säulen repräsentieren das Zeitmanagement, die Qualitätskontrolle und die Autonomie der einzelnen Prozesse. Zum Schluss ist das Ziel des Toyota Produktionssystems auf den Säulen dargestellt. Das Ziel umfasst die drei Aspekte (vgl. Wienholdt und Meyer, 2007):

1. Beste eingebaute Qualität
2. Minimale Kosten
3. Kürzeste Durchlaufzeit

Da das Toyota Produktionssystem eine überschaubare Anzahl an Methoden besitzt, wird es auch als schlankes Produktionssystem bezeichnet (vgl. Dombrowski, 2015). Das Ganzheitliche Produktionssystem nutzt dies als sein Fundament und erweitert das schlanke Produktionssystem mit Methoden und Subsystemen, sodass das Produktionssystem ganzheitlich wird und die vier genannten Kriterien erfüllt.

Modulare Produktionssysteme

Wie in Abschnitt 2.1 erklärt sind Unternehmen einem immer turbulenter werdenden Umfeld ausgesetzt. Um auf diese Turbulenzen reagieren zu können, sind die Unternehmen darauf angewiesen ihre Flexibilität und Wandlungsfähigkeit stetig zu erhöhen. Die genannten Turbulenzen stehen für die wachsende Kundenorientierung der Unternehmen (vgl. Wiendahl et al., 2009). So stellt der aktuelle Trend die individuellen Kundenwünsche in den Vordergrund. Um diese zu berücksichtigen, sind die Unternehmen dazu gezwungen eine Vielzahl an Produkten und Produktarten anzubieten. Des Weiteren muss auch auf die variierenden Produktmengen reagiert werden können. In aktuellen wissenschaftlichen Diskussionen wird von der Modularisierung oder auch Rekonfigurierbarkeit der Produktionssysteme gesprochen (vgl. Matt, 2005). Die Modularisierung in den Produktionssystemen ist dabei auf Fabrikmodule und Prozessmodule bezogen. Dabei gilt ein Produktionssystem als ein modulares Produktionssystem, wenn es sechs Kriterien erfüllt (vgl. Koren und Shpitalni, 2010). Zu den Kriterien zählen die:

1. **Flexibilität**: Das Unternehmen muss in der Lage sein die Flexibilität der Anlage zu erhalten.
2. **Wandlungsfähigkeit**: Das Unternehmen muss in der Lage sein die Anlage in kurzer Zeit an Änderungen oder an neuen Produktanforderungen anpassen zu können.

3. **Skalierbarkeit**: Das Unternehmen muss in der Lage sein die Anlage in kurzer Zeit an neue Produktmengen anzupassen, indem diese um Komponenten erweitert wird oder Komponenten aus ihr entfernt werden.
4. **Modularität**: Das Unternehmen muss in der Lage sein die Anlage in Module unterteilen zu können und deren Anordnung manipulieren zu können.
5. **Integrierbarkeit**: Das Unternehmen muss in der Lage sein, Module in kurzer Zeit mit Hilfe von mechanischen oder steuerungstechnischen Schnittstellen in das Produktionssystem integrieren zu können.
6. **Diagnosefähigkeit**: Das Unternehmen muss in der Lage sein die Anlage zu jedem Zeitpunkt überwachen zu können, um Fehler schnell zu erkennen und auf diese reagieren zu können.

Das modulare Produktionssystem umfasst also viele Anforderungen an das Unternehmen. Diese stellen für das Unternehmen eine Herausforderung dar. Ist diese Herausforderung jedoch bewältigt worden, so wird das Unternehmen in der Lage sein, diverse Probleme der Produktionsplanung zu lösen. Zu den genannten Problemen der Produktionsplanung zählt unter anderem auch das Job-Shop-Problem beziehungsweise das Job-Shop-Scheduling.

2.2 Das Job-Shop-Scheduling

Das Job-Shop-Scheduling ist ein Bestandteil der Produktionsplanung. Dabei konzentriert sich das Job-Shop-Scheduling-Problem oder auch das Job-Shop-Problem darauf, die zeitliche Reihenfolge von Aufträgen festzulegen (vgl. Rabe und Deininger, 2013). Es werden n verschiedene Aufträge bzw. Jobs J_i auf m verschiedenen Maschinen M_k exakt einmal bearbeitet. Jeder Job setzt sich dabei aus a verschiedenen Arbeitsschritten A_1 ,…, A_a zusammen. Diese müssen in einer festgelegten Reihenfolge eingeplant werden. Die Summe aller Arbeitsschritte beträgt $A_{ges} = \sum_{i=1}^{n} a_i$. Falls der Spezialfall eintritt, dass $a_i = m$ ist, so beträgt die Summe aller Arbeitsschritte $A_{ges} = n \cdot m$. Jedem Arbeitsschritt A_i ist eine Bearbeitungszeit t_{ij} zugeordnet. Diese Bearbeitungszeit ist die Zeit, in welcher ein Job J_i auf einer Maschine M_i unterbrechungslos bearbeitet werden muss. Die Jobs unterscheiden sich darin, dass diese für ihre Fertigstellung in jeweils unterschiedlichen Reihenfolgen die Maschinen durchlaufen müssen. Die Jobs sind unabhängig voneinander. Zu beachten ist, dass ein Job nicht zeitgleich von mehreren Maschinen bearbeitet werden kann. Es ist ebenfalls nicht möglich, dass eine Maschine mehrere Jobs zur selben Zeit bearbeitet (vgl. Schutten, 1998). Die Zielgröße des Job-Shop-Scheduling ist die Gesamtzykluszeit Z_{ges}. Die Gesamtzykluszeit ist die Dauer, die benötigt wird, um sämtliche Arbeitsschritte aller Jobs auszuführen. Das Job-Shop-Scheduling verfolgt zwei Ziele. Das erste Ziel ist die Minimierung der Durchlaufzeiten der Aufträge, sodass diese annähernd den Förder- und Bearbeitungszeiten

entsprechen. Die durch Prozessstörungen bedingten Zwischenlagerzeiten der zu bearbeitenden Elemente sollen ebenfalls minimiert werden. Das zweite Ziel bezieht sich auf die Minimierung der Leerzeiten. Dazu müssen die Kapazitäten im Betrieb maximal ausgelastet werden, wobei dies durch die Reihenfolge der Aufträge umgesetzt werden soll. Um diese beiden Ziele zu realisieren, muss die Gesamtzykluszeit Z_{ges} minimiert werden (vgl. Graham, 1979). Ende der 70er Jahre ist der Beweis erfolgt, dass es sich bei Job-Shop-Problemen, welche die Gesamtzykluszeit Z_{ges} minimieren sollen, um NP-schwere Probleme handelt. Das Ausmaß dieser Problematik wird deutlich, wenn das von Muth und Thompson aufgestellte Problem betrachtet wird. Bei dem Problem ist ein Job-Shop-Problem mit zehn Jobs und zehn Maschinen untersucht worden. Dieses Problem ist länger als 20 Jahre ungelöst geblieben. Die Menge alle möglichen Ereignisse des Job-Shop-Problems errechnet sich durch $(n!)^m$, wobei in diesem Fall ein Ereignis eine Auftragsreihenfolge repräsentiert. Für das Problem von Muth und Thompson bedeutet dies, dass $(10!)^{10} \approx 395{,}94 \cdot 10^{63}$ mögliche Lösungen existieren (vgl. Schuster, 2003).

Um die Auslastung der Maschinen und die Gesamtzykluszeit der getroffenen Wahl beurteilen zu können, bietet sich das Gantt-Diagramm an. Für die Erstellung eines Gantt-Diagramms werden Informationen benötigt. Bei den benötigten Informationen handelt es sich um eine Auftragsliste in einer festgelegten Reihenfolge. Des Weiteren müssen Kenntnisse über die Bearbeitungszeiten der jeweiligen Aufträge auf den einzelnen Maschinen vorhanden sein (vgl. Yamada und Nakano, 1997). Zur Verdeutlichung werden die Daten aus der Tabelle 2.1 genutzt.

	J_1			J_2			J_3		
Ai	1	2	3	4	5	6	7	8	9
M_i	M_1	M_2	M_3	M_1	M_3	M_2	M_2	M_1	M_3
t_i	3	3	3	2	3	4	3	2	1

Tabelle 2.1: Daten des Job-Shop Problems

Die Tabelle 2.1 teilt sich in drei Jobs auf. Dabei ist jeder Arbeitsschritt eines Jobs einer Maschine zugeordnet. Außerdem sind die erforderlichen Bearbeitungszeiten für die jeweiligen Arbeitsschritte eingetragen. Die Daten aus der Tabelle 2.1 werden zur Erstellung des Gantt-Diagramms in der folgenden Abbildung 2.2 genutzt.

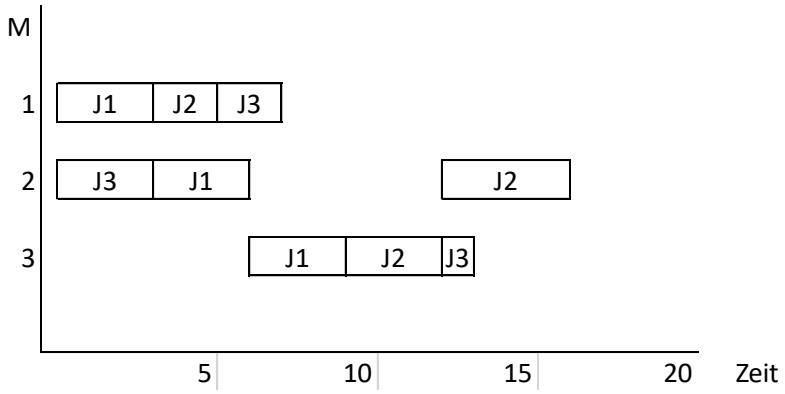


Abbildung 2.2: Gantt-Diagramm

In der Abbildung 2.2 ist das Gantt-Diagramm für die bisher behandelte Auftragsliste ersichtlich. Zu erkennen ist, welche Maschinen zu gegebenem Zeitpunkt gleichzeitig arbeiten. Außerdem kann auf der x-Achse die Gesamtzykluszeit Z_{ges} abgelesen werden. Mit Hilfe von mehreren Gantt-Diagrammen für verschiedene Auftragsreihenfolgen können Leerzeiten schnell erkannt werden. Durch die Nutzung von Gantt-Diagrammen kann das Ziel des Job-Shop-Problems erreicht werden. Dabei handelt es sich um die Maximierung der Maschinenauslastung und die zeitgleiche Minimierung der Gesamtzykluszeit (vgl. Graham, 1979).

3 Simulation

Die Simulation ist ein Hilfsmittel zur Analyse und Optimierung von zu planenden und bereits existierenden Systemen, welches immer mehr an Bedeutung gewinnt. In der Produktion und Logistik ist die gängigste Simulationsart die ereignisorientierte Simulation, mit welcher sich dieses Kapitel auseinandersetzt. Dabei stellt die Simulation ein Modell eines Produktionssystems dar. Zur Modellierung der Produktionssysteme werden Petri-Netze verwendet, wobei sich diese Arbeit auf die Erweiterung der Petri-Netze konzentriert. Bei der Erweiterung der Petri-Netzte handelt es sich um die Zeitbehafteten Hierarchischen Objektorientierten Netze.

3.1 Ereignisorientierte Simulation

Die Simulation wird nach der Richtlinie VDI 3633 Blatt 1 definiert als das

> „*Nachbilden eines dynamischen Prozesses in einem System mithilfe eines experimentierfähigen Modells, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind.*“

Es ist zu beachten, dass die Simulation keine Lösung, sondern lediglich eine Bewertung liefert. Diese Bewertung wird als Simulationsexperiment bezeichnet. Ein Simulationsexperiment ist nach VDI 2008, Abschnitt 1.4 die

> „*gezielte empirische Untersuchung des Verhaltens eines Modells durch wiederholte Simulationsläufe mit systematischer Parameter- oder Strukturvariation.*“

Die Qualität der Bewertung beziehungsweise des Simulationsexperiments ist abhängig vom Anwender. Ein erfahrener Anwender ist in der Lage mehr Informationen aus einem Simulationsmodell zu gewinnen als ein unerfahrener Anwender aus demselben Simulationsmodell. Es wird zwischen der kontinuierlichen und der ereignisorientierten Simulation unterschieden. Die kontinuierliche Simulation zeichnet sich dadurch aus, dass die Modellvariablen kontinuierliche Funktionen der Simulationszeit sind. In der kontinuierlichen Simulation existieren keine klar voneinander abgegrenzten Objektzustände. Bei der ereignisorientierten Simulation hingegen sind die Modellvariablen keine kontinuierlichen Funktionen der Zeit. Es werden lediglich die Zeitpunkte verzeichnet, zu welchen Ereignisse stattfinden. Dabei beschreibt ein Ereignis eine Zustandsänderung (vgl. Pidd, 2004). In der Realität ist die Zeit jedoch ein kontinuierlicher Fluss, in welchem keine Sprünge zu verzeichnen sind. Ein Simulationsprogramm, welches eine ereignisorientierte Simulation nutzt, zieht nur die Zeitpunkte in Erwägung, die auch den weiteren Verlauf der Simulation beeinflussen. Zur Veranschaulichung wird ein Beispiel aus der Produktion gewählt. Die Ankunft von Materialien an einer Maschine ist ein Ereignis. Sonstige Bewegungen, die zwischen dem Versand und der Ankunft der Materialien an der Maschine stattfinden, sind für die Simulation irrelevant und es

wird lediglich eine Förderzeit auf die Simulationszeit addiert. Wichtig ist die Bearbeitungszeit an der Maschine. Diese Zeit wird durch das Eintreffen der Materialien an der Maschine generiert. So addiert das Simulationsprogramm die Zeit bis zu dem Eintritt des Ereignisses auf die bisher verstrichene Zeit. Dies ist der Grund, weshalb in ereignisorientierten Simulationen Zeitsprünge zu verzeichnen sind (vgl. Siemens, 2014).

3.2 Petri-Netze

Petri-Netze sind Instrumente zur graphischen Beschreibung und Analyse von Systemen. Mit ihrer Hilfe können dynamische Systeme mit nicht-deterministischen und nebenläufigen Abläufen modelliert, simuliert und anschließend analysiert werden (vgl. Reising, 2010). Petri-Netze sind stark mathematisch geprägt. Dennoch sind sie durch die simple visuelle Darstellungsweise sehr übersichtlich und anschaulich. Dadurch ist es möglich, schnell das Verständnis komplexer Systeme aufzubauen. Aus diesem Grund sind Petri-Netze beliebte Hilfsmittel in der Produktion und Logistik, obwohl Petri-Netze ursprünglich im Jahr 1962 für die Automatentechnik entwickelt worden sind (vgl. Petri, 1962).

Zunächst werden die fundamentalen Bestandteile von Petri-Netzen untersucht. Dabei handelt es sich um Plätze, Transitionen und Kanten.

Plätze

Die Struktur von Petri-Netzen enthält zwei Arten von Elementen. Dabei umfasst die erste Art von Elementen die Plätze. Die Visualisierung von Plätzen erfolgt, wie in der Abbildung 3.1 ersichtlich, mittels einer Ellipse oder eines Kreises. Mit einem Platz *P* können ausschließlich passive Komponenten nachgebildet werden. Passive Komponenten sind in der Lage Objekte zu lagern, Daten zu speichern oder sich in einem Zustand zu befinden (vgl. Balzert, 2009).

Abbildung 3.1: Plätze

Transitionen

Bei der zweiten Art von Elementen in der Struktur von Petri-Netzen handelt es sich um die Transitionen. Die Visualisierung von Transitionen erfolgt, wie in der Abbildung 3.2 ersichtlich, mittels eines Rechtecks oder eines Quadrates. Mit einer Transition *T* können ausschließlich aktive Komponenten nachgebildet werden. Aktive Komponenten sind in der Lage Objekte zu erzeugen, zu verändern, zu verbrauchen oder zu transportieren (vgl. Balzert, 2009).

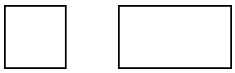


Abbildung 3.2: Transitionen

Kanten

Kanten dienen der Verbindung von Plätzen und Transitionen. Dabei handelt es sich um gerichtete Kanten. Aus diesem Grund werden diese, wie in der Abbildung 3.3, als Pfeile dargestellt. Kanten werden nie zur Nachbildung von Komponenten der Systeme genutzt. Kanten stellen ausschließlich abstrakte oder auch lediglich gedankliche Beziehungen zwischen Systembestandteilen dar. Unter gedanklichen Beziehungen werden unter anderem logische Zusammenhänge oder Zugriffsrechte verstanden. Einzelne Kanten können dabei mit der mathematischen Schreibweise für Strecken zwischen zwei Punkten adressiert werden. So wird eine Kante, welche eine Komponente A mit einer Komponente B verbindet, als $\overline{ab}$ bzw. $\overline{ba}$ bezeichnet (vgl. Balzert, 2009).

Abbildung 3.3: Kante

In Petri-Netzen können Kanten von Plätzen ausgehen und zu Transitionen führen oder von Transitionen ausgehen und zu Plätzen führen. Das Verbinden von zwei Plätzen oder zwei Transitionen ist nicht erlaubt. Bisher sind die graphischen Aspekte von Petri-Netzen betrachtet worden. Diese basieren jedoch auf mathematischen Grundlagen. Die beiden Strukturelemente „Plätze“ und „Transitionen“ werden als Knoten angesehen. So sind Petri-Netze Graphen, welche eine Knotenmenge V beschreiben. Dabei werden die Plätze P und die Transitionen T als disjunkte Mengen definiert (vgl. Reising, 2010). Es gilt:

$$V = P \cup T, P \cap T = \emptyset$$

Um sicherzustellen, dass keine Verbindung innerhalb der Menge P oder innerhalb der Menge T hergestellt werden kann, wird die folgende Regel aufgestellt:

$$F \subset (T \times P) \cup (P \times T) \subset V \times V$$

Folglich wird die Menge $N = (P, T, F)$ als Netzstruktur bezeichnet. Um die Orientierung der Kanten festzustellen, wird der Vor- und Nachbereich eingeführt. Der Vorbereich informiert darüber, woher die Kanten kommen, wobei der Nachbereich darüber informiert, wohin die Kanten führen. Auch diese Eigenschaft von Petri-Netzen basiert auf mathematischen Grundsteinen (vgl. Reising, 2010). So ist der Vorbereich definiert als:

$$\circ x = \{y | yFx\}$$

Analog zum Vorbereich ist der Nachbereich definiert als:

$$x \circ = \{y | yFx\}$$

Die bisher vorgestellten Komponenten von Petri-Netzen genügen lediglich der statischen Nachbildung von Prozessen. In der Produktion und Logistik sind jedoch immer Materialflüsse zu simulieren. Objekte, welche beispielsweise während eines Prozesses auf mehreren Maschinen bearbeitet werden, werden als fließend angesehen. Um diese in Petri-Netzen nachzubilden, werden Markierungen eingeführt. Dabei ist eine Markierung als Verteilung von Marken auf Plätzen zu verstehen. Die Visualisierung der Marken erfolgt mittels Symbolen, welche sich innerhalb der Ellipsen oder Kreise befinden. Oft genügt eine abstrakte Darstellung durch einen Punkt. Hiermit wird meistens gekennzeichnet, dass eine bestimmte Bedingung erfüllt ist. Außerdem können auch reale Flussobjekte als Punkt dargestellt werden, wobei dies lediglich empfehlenswert ist, wenn nur eine Art von Flussobjekten existiert. Bei komplexeren Systemen mit unterschiedlichen Flussobjekten bieten sich dagegen verschiedene Symbole an. Die Symbole repräsentieren dabei Elemente aus der Realität. In der folgenden Abbildung 3.4 sind oft genutzte Symbole für Markierungen zu sehen (vgl. Balzert, 2009).

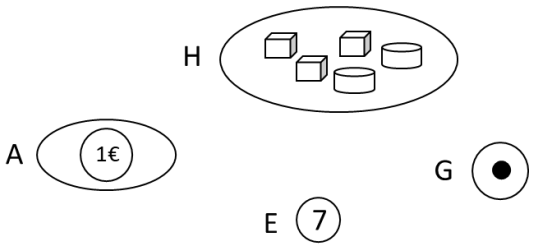


Abbildung 3.4: Symbole

Eine weitere Möglichkeit die Anschaulichkeit von Petri-Netzen zu steigern, ist die Beschriftung der Kanten und Transitionen mit Ausdrücken. Die Ausdrücke stellen oft Funktionen oder Variablen dar. Nachdem ein System vollständig mit Variablen modelliert worden ist, können die Variablen durch beliebige Elemente ersetzt werden. Dies ist der Vorteil, den Ausdrücke mit sich bringen, da so eine erhöhte Flexibilität bei der Anwendung der Petri-Netze möglich ist. Eine Beschriftung der Transitionen umfasst Variablen als Parameter, welche verschiedene Modi der Transitionen beschreiben. An dieser Stelle ist zu beachten, dass eine Transition lediglich dann eintreten kann, wenn die Beschriftung den Wert „true“ annimmt. In der folgenden Abbildung 3.5 sind gängige Beschriftungen von Kanten und Transitionen dargestellt (vgl. Balzert, 2009).

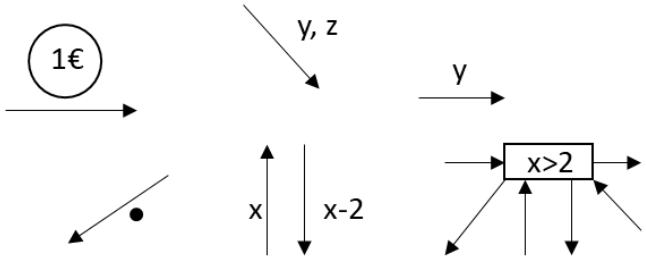


Abbildung 3.5: Beschriftete Kanten

In der Produktion und Logistik werden hauptsächlich objektorientierte Petri-Netze verwendet. Es existieren drei verschiedene Ansätze bezüglich der Integrationsrichtung zwischen Petri-Netzen und objektorientierten Konzepten (vgl. Zapf und Heinzl, 1998):

1. Ansätze zur Einbettung von Petri-Netzen in Objekten
2. Ansätze zur Einbettung von Objekten in Petri-Netze
3. Ansätze zur beidseitigen Integration von Objekten und Petri-Netzen

In dieser Arbeit stehen dabei die Ansätze zur Einbettung von Objekten in Petri-Netze im Vordergrund. Bei diesem Ansatz werden die Marken der Petri-Netze mit objektorientierten Datentypen hinterlegt, wobei die Transitionen durch objektorientierte Programmiersprachen definiert werden. Um ein System mit objektorientierten Petri-Netzen beschreiben zu können, werden die Begriffe „Objekte“, „Klassen“ und „Methoden eingeführt (vgl. Schnieder, 1999):

1. **Objekte**: Ein System setzt sich aus Bausteinen zusammen. Fundamentale Bausteine des Systems werden als Objekte bezeichnet. Objekten können dabei individuelle Eigenschaften hinterlegt werden.
2. **Klassen**: Objekte mit verschiedenen Eigenschaften werden als unterschiedliche Objekttypen angesehen. Eine Klasse fasst ähnliche Objekttypen zusammen.
3. **Methoden**: Mit Hilfe von Methoden können die Verhaltensweisen von Objekten implementiert werden. Dabei ist die Aufgabe von Methoden die Manipulation einzelner Bestandteile von Objekten.

Methoden besitzen wichtige Eigenschaften, welche für objektorientierte Petri-Netze von großer Bedeutung sind (vgl. Schnieder, 1999). Dabei handelt es sich um die:

1. **Abstraktion**: Komplexe Zusammenhänge der Realität können simplifiziert werden.
2. **Vererbung**: Methoden und Variablen können an Unterklassen weitergegeben werden. In den Unterklassen ist die Erweiterung der geerbten Variablen und Methoden möglich. Außerdem können Unterklassen zusätzliche Variablen und Methoden einbinden.
3. **Kapselung**: Objektdetails können vor Teilen des Programms verborgen werden.
4. **Polymorphie**: Die Eigenschaft einer Methode in unterschiedlichen Klassen dieselbe Signatur besitzen und dennoch erneut implementiert werden zu können.

THORNs zählen ebenfalls zu einem Ansatz, mit welchem Objekte in Petri-Netze eingebunden werden.

3.3 Zeitbehaftete Hierarchische Objektorientierte Netze

In dieser Arbeit soll ein modulares Produktionssystem nachgebildet werden, um mit Hilfe von Simulationen die Module in Abhängigkeit von der Auftragslage auszuwechseln. Dadurch soll eine erhöhte Flexibilität des Produktionssystems gewährleistet werden. Um diesen Anforderungen gerecht zu werden, genügen klassische Petri-Netze oft nicht. Aus diesem Grund ist es sinnvoller THORNs zu verwenden. THORNs sind erweiterte Petri-Netze. Die grundlegenden Strukturen von Petri-Netzen wie „Stellen“, „Plätze“, „Kanten“ und „Transitionen“ sind auch fundamental für THORNs (vgl. Wieting, 1996). Jeder dieser Bestandteile ist dabei erweitert und verändert. So werden keine attributlosen Marken wie bei Petri-Netzen genutzt. THORNs nutzen Objekte, welche in der Programmiersprache C++ programmiert werden können. Dadurch ist es möglich, komplexe Objekte mit unterschiedlichen Objekttypen zu erstellen. Die Komplexität der Objekte ist erforderlich, wenn komplexe Systeme nachgebildet werden sollen. Die Objekte sind auf den Stellen platziert. Die Stellen der THORNs besitzen die vier Attribute „Name“, „Typ“, „Struktur“ und „Kapazität“. Der Name der Stelle ist ein Attribut, welches lediglich der Übersichtlichkeit dient. Dieser kann vom Anwender frei gewählt werden. Der Typ einer Stelle legt fest, welche Objekttypen auf der jeweiligen Stelle platziert werden dürfen. Die Struktur einer Stelle ist ein Attribut, welches Reihenfolgebeziehungen zwischen den Objekten auf einer Stelle herstellt. Dabei wird zwischen den Strukturen „Multiset“, „Stack“, „Queue“ und „Priority Queue“ unterschieden (vgl. Schöf et al., 1997). Stellen, welche die Struktur „Multiset“ annehmen, legen keine Regeln für die Objekte fest. Stellen, welche die Struktur „Stack“ aufweisen, legen die Reihenfolgebeziehung nach dem Prinzip „first in last out“ fest. Die Objekte werden also auf den Stellen gestapelt. Dadurch ist das letzte Objekt, das auf dem Stapel platziert worden ist, das erste Objekt, welches vom Stapel entnommen werden kann. Das Komplement zur Struktur „Stack“ ist die Struktur „Queue“. Die Struktur „Queue“ legt die Reihenfolgebeziehung nach dem Prinzip „first in first out“ fest. Die Objekte auf der Stelle werden also aneinandergereiht, wobei das erste Objekt in der Warteschlange als erstes aus dieser austritt. Die letzte Struktur ist die „Priority Queue“. Diese Struktur bietet dem Anwender der Simulation die Möglichkeit benutzerdefinierte Prioritätsfunktionen für die jeweilige Stelle festzulegen. Das vierte Attribut von Stellen in THORNs ist die „Kapazität“. Die Kapazität legt die Anzahl an Objekten fest, mit welcher die jeweilige Stelle belegt werden darf. Nachdem die Stellen, welche die erste Art von Elementen in Petri-Netz- und THORN-Strukturen darstellt, miteinander verglichen worden sind, wird nun die zweite Art von Elementen in den genannten Strukturen gegenübergestellt (vgl. Schöf et al., 1997). Bei der zweiten Art von Strukturelementen handelt es sich um die

Transitionen. Transitionen in THORNS besitzen die sechs Parameter „Name“, „Schaltkapazität“, „Schaltbedingung“, „Schaltaktion“, „Verzögerungszeit“ und „Schaltdauer“. Der Parameter „Name“ dient lediglich zur Übersichtlichkeit und überlässt dem Anwender der Simulation freie Wahl über die Namensgebung. Der Parameter „Schaltkapazität“ gibt die maximale Anzahl von Parallelschaltungen einer Transition mit sich selbst an. Die Schaltkapazität wird durch eine natürliche Zahl determiniert oder durch den Wert Ω als unbeschränkt definiert. Mit Hilfe des Parameters „Schaltbedingungen“ ist es dem Anwender der Simulation möglich, Bedingungen an die Objekte aus dem Vorbereich zu stellen. Um eine Schaltung einer Transition zu gewährleisten, ist es zunächst notwendig sämtliche festgelegten Schaltbedingungen zu erfüllen. Der Parameter „Schaltaktion“ legt fest, welche Aktionen durch das Schalten einer Transition ausgelöst werden. Die Aktionen der Transition beziehen sich dabei sowohl auf Objekte des Vorbereichs, als auch auf Objekte des Nachbereichs. Die Parameter „Verzögerungszeit“ und „Schaltdauer“ sind zwar zwei eigenständige Parameter, jedoch legen sie gemeinsam das zeitliche Verhalten von Transitionen fest. Die Verzögerungszeit legt fest, wie lange eine Transition ohne Unterbrechungen aktiv zu sein hat, um eine Schaltung zu ermöglichen. Ist eine Schaltung erfolgt, so wird der Parameter „Schaltdauer“ wirksam. Dieser Parameter legt das Zeitintervall vom Beginn der Schaltung bis zum Ende der Schaltung fest. Der Anwender hat die Möglichkeit das zeitliche Verhalten der Transition in Abhängigkeit von Objekten aus dem Vorbereich zu definieren (vgl. Wieting, 1996). Der dritte Bestandteil klassischer Petri-Netze sind die Kanten. Auch dieser Bestandteil ist in THORNs in erweiterter Ausführung vorzufinden. Kanten besitzen in THORNs die beiden Parameter „Name“ und „Kantentyp“. In Abhängigkeit vom Kantentyp können die Kanten um weitere Parameter erweitert werden. Zu diesen zusätzlichen Parametern zählen die Parameter „Gewicht“ und „Variablenname“. Der Parameter „Variablenname“ dient lediglich zur Übersicht und überlässt dem Anwender der Simulation freie Wahl über die Namensgebung. Kanten in THORNs können die Typen „standard“, „enabling“, „inhibitor“ oder „consuming“ annehmen. Die Visualisierungen aller wichtigen Bestandteile der THORNs sind in der folgenden Abbildung 3.6 zusammengefasst (vgl. Rabe und Deininger, 1997).

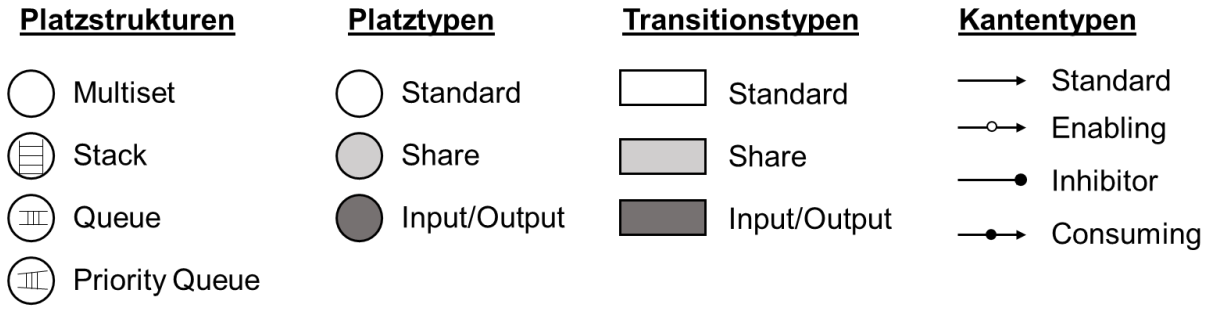


Abbildung 3.6: Platzstrukturen

THORNs werden meistens in einer Black-Box verwendet. Dies bedeutet, dass der Anwender der Simulation die Bestandteile der THORNs zwar nutzen und verändern kann, jedoch werden diese nicht visuell ausgegeben.

4 Optimierungsverfahren

Optimierungsverfahren können zum einen in exakte Optimierungsverfahren und zum anderen in heuristische Optimierungsverfahren gegliedert werden (vgl. Domschke et al., 1997). Dabei wird unter einer Optimierung eine Verbesserung des aktuellen Zustands oder die Findung eines Maximums beziehungsweise Minimums verstanden (vgl. VDI 1996). In diesem Kapitel werden zunächst beide Optimierungsverfahren erläutert und miteinander verglichen. Daraufhin wird die Bedeutung der Simulation für Optimierungsverfahren verdeutlicht. Da in der vorliegenden Bachelorarbeit eine ACO-Metaheuristik entwickelt wird, werden heuristische Optimierungsverfahren detaillierter betrachtet.

4.1 Exakte Optimierungsverfahren

Bei den exakten Optimierungsverfahren handelt es sich um Optimierungsverfahren, die auf dem Prinzip der Enumeration basieren. In einer endlich großen Anzahl an Schritten ist es dabei möglich, das globale Optimum eines Optimierungsproblems zu finden (vgl. Rieck, 2009). Es besteht zudem die Möglichkeit, dass eine optimale Lösung nicht gefunden wird und das Problem nicht gelöst werden kann (vgl. Acker, 2011). Mit steigender Komplexität steigen gleichzeitig die geforderte Rechnerleistung und die Laufzeit (vgl. Schwartz, 2004). Möglichkeiten, Optimierungsprobleme wie das Job-Shop-Problem oder die Variation von modularen Produktionssystemen zu lösen, bestehen in der sogenannten gemischt-ganzzahligen Optimierung (vgl. Troßmann, 1996). Um dieses Optimierungsverfahren besser nachvollziehen und im Anschluss mit heuristischen Optimierungsverfahren vergleichen zu können, wird es im Folgenden kurz erläutert.

Gemischt-ganzzahliges Optimierungsproblem

Da gemischt-ganzzahlige Optimierungsverfahren zu den exakten Optimierungsverfahren gehören, basieren sie auf dem Prinzip der Enumeration (vgl. Kallrath, 2013). So werden alle potentiellen Lösungsmöglichkeiten erzeugt und untersucht, um die optimale Lösung zu finden. Die Wahl der richtigen Variablen und die Beschreibung dieser spielen hierbei eine große Rolle (vgl. Kallrath, 2013).

4.2 Heuristische Optimierungsverfahren

Heuristische Optimierungsverfahren sind eine besondere Form von Optimierungsverfahren. Dabei wird nach der Durchführung eines heuristischen Optimierungsverfahrens das globale Optimum des Problems nicht unbedingt gefunden. Das Optimierungsverfahren strebt viel mehr nach einer guten Lösung in einer angemessenen Zeit (vgl. Ehrgott, 2004). Dies bedeutet, dass

am Ende eine Menge heuristisch effizienter Lösungen vorliegen. Als heuristische Lösung wird eine Lösung bezeichnet, welche von keiner anderen bekannten Lösung übertroffen wird (vgl. Huckert, 1980). Wie bereits erwähnt, kommen heuristische Optimierungsverfahren überwiegend dann zum Einsatz, wenn NP-schwere Probleme vorliegen (vgl. Kistner und Steven, 2001). Bei NP-schweren Problem steigt die Komplexität der Optimierungsprobleme mit steigender Anzahl an Knoten exponentiell. Mit Hilfe von Ant-Colony-Optimierung lassen sich solche Probleme gut lösen (vgl. Pinedo, 1995). Ein Optimierungsverfahren zählt zu den heuristischen Verfahren, wenn drei Eigenschaften erfüllt sind (vgl. Streim, 1975):

1.) Willkürliche Entscheidungsoperatoren kommen nicht zum Einsatz
2.) Im Suchprozess werden potentielle Lösungen ausgeschlossen
3.) Es gibt keine Garantie, dass eine globale optimale Lösung gefunden wird

Heuristische Optimierungsverfahren lassen sich wie folgt gliedern (vgl. Domschke et al., 1997):

1.) Relaxationsbasierte Verfahren
2.) Unvollständig ausgeführte exakte Verfahren
3.) Eröffnungsverfahren
4.) Verbesserungsverfahren
5.) Unterschiedliche Kombinationen der einzelnen Verfahren.

Eine weitere wichtige Unterteilung der Heuristischen Optimierungsverfahren ist die Unterteilung in die deterministische und die stochastische Heuristik (vgl. Domschke et al., 1997). Dabei wird bei den deterministischen Heuristiken immer dasselbe Ergebnis erzielt. Stochastische Lösungsverfahren hingegen, beginnen immer mit einer zufälligen Anfangskombination, was dazu führt, dass es immer zu unterschiedlichen Ergebnissen führt (vgl. Domschke et al., 1997). Der Hauptunterschied zu den exakten Optimierungsverfahren liegt darin, dass das Ergebnis der heuristischen Verfahren, hinsichtlich der Güte der Lösung, nicht beurteilt werden kann (vgl. Klemmt, 2012). Um einen besser Überblick über die Heuristiken zu bekommen, werden im Folgenden die ersten vier heuristischen Optimierungsverfahren kurz erläutert.

Relaxationsbasierte Verfahren

Bei diesem heuristischen Verfahren werden Nebenbedingungen vereinfacht. So können gute Lösungen des Optimierungsproblems in geringer Zeit gefunden werden (vgl. Domschke et al., 1997).

Unvollständig ausgeführte exakte Verfahren

Dieses Verfahren beinhaltet die Funktionsweise schon im Namen. Es werden exakte Verfahren nur bis zu einem bestimmten Zeitpunkt ausgeführt und dann beendet. Der Zeitpunkt

kann beispielsweise eine definierte, erreichte Güte der Lösung sein (vgl. Domschke et al., 1997).

Eröffnungsverfahren

Beim Eröffnungsverfahren wird die Lösung des Problems durch sukzessive Vervollständigung von Teillösungen ermittelt (vgl. Zäpfel und Braune, 2005). Ein Beispiel für Eröffnungsverfahren ist das Prioritätsregelverfahren, welches vor allem beim Job-Shop-Problem angewendet wird (vgl. Acker, 2011). Wichtig zu wissen ist, dass die durch ein Eröffnungsverfahren ermittelte Lösung auch als Startlösung für das im späteren Verlauf näher erläuterte Verbesserungsverfahren genutzt werden kann (vgl. Müller-Merbach, 1970). Das Prioritätsregelverfahren erfolgt nach Domschke (1997) in zwei aufeinander aufbauenden Schritten. Zum einen erfolgt das Prioritätsregelverfahren nach der Bestimmung der Rangwerte für ein Objekt mit Hilfe einer Prioritätsregel und zum anderen nach der Bestimmung der Reihenfolge der Aufträge auf Basis der Rangwerte. Prioritätsregelverfahren haben jedoch eine geringe Lösungsqualität. Die Lösung, die am Ende geliefert wird, weicht durchschnittlich um ca. 20 % von der optimalen Lösung ab. Daher ist dieses Verfahren nicht bei strengen Anforderungen von Lösungen geeignet (vgl. Schwartz, 2004).

Verbesserungsverfahren

Verbesserungsverfahren benötigen bereits zu Beginn eine zulässige Lösung. Diese Lösung kann beispielsweise zufällig erzeugt werden (vgl. Domschke et al., 1997). Ausgehend von dieser zufälligen Lösung x wird eine verbesserte Lösung x' gesucht. Dabei wird die Lösung x nur marginal geändert, um auf x' zu kommen (vgl. Domschke und Drexl, 2005). Es wird also nicht der komplette Lösungsraum X untersucht (vgl. Dowsland, 1993). Dieses Prinzip ist auch bekannt als „find a feasible solution and then search for better ones“ (vgl. Hillier, 1969). Der Lösungsraum, der untersucht wird, wird Nachbarschaftsraum genannt. Dieser ist definiert als Teilmenge des gesamten Lösungsraums X (vgl. Dowsland, 1993). Bei den marginalen Änderungen der Lösung gibt es jedoch Regeln, die zu beachten sind. Demnach ist eine Transformation nur soweit zulässig, wenn sie (vgl. Domschke und Scholl, 2006):

1.) nur der Veränderung an einer Stelle entspricht
2.) nur die Elemente vertauscht
3.) nur die Elemente verschiebt

Eine Transformation von der Lösung x zu x' wird auch als Zug v bezeichnet (Domschke und Drexl 2005). Es gibt auch die Möglichkeit die Änderungen pro Transformation deutlich größer zu wählen. Dies führt jedoch dazu, dass die Annäherung an das Optimum übersehen werden kann (vgl. Dueck et al., 1993). Daher ist es vorteilhaft Änderungen zu vorzunehmen, bei denen sich die Lösung der Nachbarschaft nur minimal ändert (vgl. Schwartz, 2004). Verbesserungs-

verfahren lassen sich in zwei Hauptkategorien gliedern (vgl. Domschke et al., 1997). Diese werden im Folgenden näher beschrieben.

Reine Verbesserungsverfahren

Lösungsverfahren, welche nur Nachbarschaftslösungen akzeptieren, deren Zielfunktionswert besser ist als der der vorhandenen Lösung, werden als reine Verbesserungsverfahren bezeichnet. Dieses Verfahren bringt jedoch einige Nachteile mit sich. Der größte Nachteil ist, dass mit diesem Verbesserungsverfahren ein lokales Maximum oder Minimum gefunden werden kann, welches stark vom globalen Extremum abweicht (vgl. Brucker, 2004). In der folgenden Abbildung 4.1 wird das Problem visualisiert.

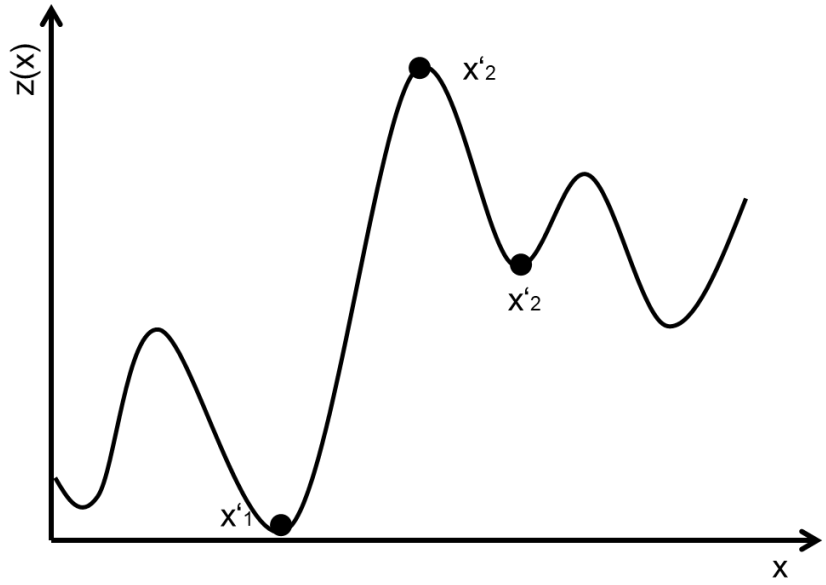


Abbildung 4.1: Lösungslandschaft (nach Domschke, 1997)

In der Abbildung 4.1 ist eine Funktion *z(x)* zu sehen, welche minimiert werden soll. Da bei reinen Verbesserungsverfahren bei der Iteration nur eine Minimierung der Zielfunktion zugelassen wird, wird beispielsweise das Minimum x'_1 nie verlassen. x'_1 ist zwar ein Optimum, jedoch nur ein lokales Optimum. Somit wird das globale Optimum x'_3 zu keinem Zeitpunkt erreicht (vgl. Domschke et al., 1997).

Lokale Suchverfahren

Verfahren, bei denen auch schlechtere Lösungen der Zielfunktion akzeptiert werden, werden lokale Suchverfahren genannt (vgl. Domschke und Scholl, 2006). Dies ist der Hauptunterschied zu den reinen Verbesserungsverfahren. Somit kann das Minimum x'_1 aus der Abbildung 4.1 verlassen werden und es besteht die Möglichkeit das globale Minimum zu erreichen. Es ist somit abhängig von der Zeit, ob das globale Minimum erreicht wird (vgl. Domschke et al., 1997). Nach diesem Prinzip sind Metaheuristiken entwickelt worden, die

Lösungsverfahren bei Optimierungsproblemen unterstützen sollen (vgl. Domschke und Scholl, 2006). Metaheuristische Optimierungsverfahren werden im folgenden Kapitel näher erläutert.

4.3 Metaheuristische Optimierungsverfahren

Metaheuristische Optimierungsverfahren unterscheiden sich von heuristischen Optimierungsverfahren nicht essentiell. Der einzige Unterschied liegt darin, dass metaheuristische Optimierungsverfahren auf eine Vielzahl von Optimierungsproblemen anwendbar sind, wobei heuristische Optimierungsverfahren sich auf ein spezielles Optimierungsproblem beziehen (vgl. Pirlot, 1996). Im Folgenden werden einige metaheuristische Optimierungsverfahren näher beschrieben.

Simulated Annealing

Das Simulated Annealing wird bei der Unterteilung der heuristischen Optimierungsverfahren den Verbesserungsverfahren zugeordnet, welche auf der lokalen Suche beruhen (vgl. Zäpfel und Braune, 2005). Dabei wird zu Beginn zunächst eine zufällige Startlösung erzeugt. Anschließend wird mit den bereits beschriebenen Transformationsvorschriften nach einer Nachbarschaftslösung gesucht und miteinander verglichen. So erfolgt nach der Generierung einer x' ein Vergleich mit der aktuellen Lösung x. Ist die Lösung x' besser als die aktuelle Lösung x, so wird x durch x' ersetzt. Trifft der Fall ein, dass die Lösung x' schlechter als x ist, so wird die schlechtere Lösung mit einer festgelegten Wahrscheinlichkeit zugelassen. Zeitgleich zu der Suche gibt es einen Streuparameter t_i, welcher nach jeder Iteration sinkt. Für $i \to \infty$ Iterationen soll dabei der Streuparameter t_i gegen den Wert Null laufen (vgl. Geiger, 2005). Das Abbruchkriterium bei der Suche kann von dem Anwender selbst festgelegt werden. Als Beispiel wäre die Erreichung einer maximalen Iteration als Abbruchkriterium möglich (vgl. Zäpfel und Braune, 2005).

Tabu-Suche

Die Tabu-Suche ist ebenfalls ein auf lokale Suchverfahren basierendes Verbesserungsverfahren. Hierbei wird bei jedem Iterationsschritt die gesamte Nachbarschaft nach besseren Zielwerten durchsucht (vgl. Zäpfel und Braune, 2005). Es beginnt mit einer zulässigen Lösung x (vgl. Geiger, 2005). Die sogenannte Tabu-Liste ist zu Beginn dabei komplett leer. Die Tabu-Liste enthält alle Lösungen, die bereits betrachtet worden sind. Diese wird verwendet, um nicht in Zyklen zu suchen und immer wieder dieselben Lösungen zu generieren. Das bedeutet, dass es nicht möglich ist, dieselbe Lösung zweimal zu betrachten (vgl. Zäpfel und Braune, 2005). Ein Ausnahmefall dafür ist die Findung einer besseren Lösung. Schlechtere Lösungen werden bei der Tabu-Suche auch zugelassen (vgl. Geiger, 2005).

Vergleichsweise zum Simulated Annealing ist dies jedoch nur möglich, wenn keine Verbesserung realisierbar ist (vgl. Domschke et al., 1997). Ein mögliches Abbruchkriterium bei der Tabu-Suche ist das Erreichen einer maximalen Anzahl an Iterationsschritten ohne Verbesserung der Lösung (vgl. Zäpfel und Braune, 2005).

5 Ameisenalgorithmus

Der Ameisenalgorithmus gehört zu den jüngsten Metaheuristiken. Daher herrscht in diesem Bereich noch ein großes Erforschungspotential (vgl. Dorigo et al.,1991_Buchous). Dieser zählt jedoch bereits zu den erfolgreichsten Algorithmen, die eine Form der Schwarmsysteme wiederspiegeln. Schwarmsysteme beschreiben Systeme, die durch kollektives Verhalten zahlreiche Vorteile aufweisen (vgl. Biethahn, 2004). Ameisenalgorithmen lassen sich der Klasse der metaheuristischen Optimierungsverfahren zuordnen. Dabei wird bei dem Ameisenalgorithmus die Natur als Vorbild genommen. Ameisen zeigen bei ihrer Futtersuche ein kollektives Problemlösungsverhalten auf, welches auf unterschiedliche Optimierungsprobleme angewendet werden kann (vgl. Dorigo et al., 2000). Um dieses Problemverhalten zu verdeutlichen, wird im weiteren Verlauf der Ameisenalgorithmus näher betrachtet.

5.1 Grundlagen des Ameisenalgorithmus

Ameisen in der Natur sind in der Lage, durch indirekte Kommunikation untereinander, den kürzesten Weg zur Futterquelle zu finden. Um das Prinzip des Ameisenalgorithmus zu verdeutlichen, wird im Folgenden das Experiment von Goss dargestellt (vgl. Goss u. a., 1989).

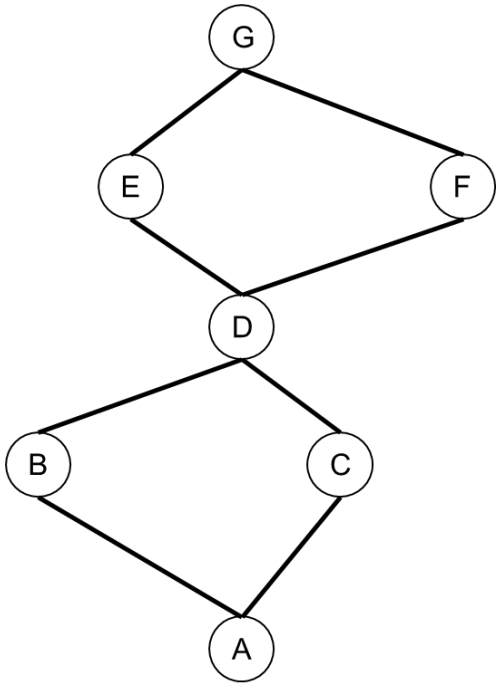
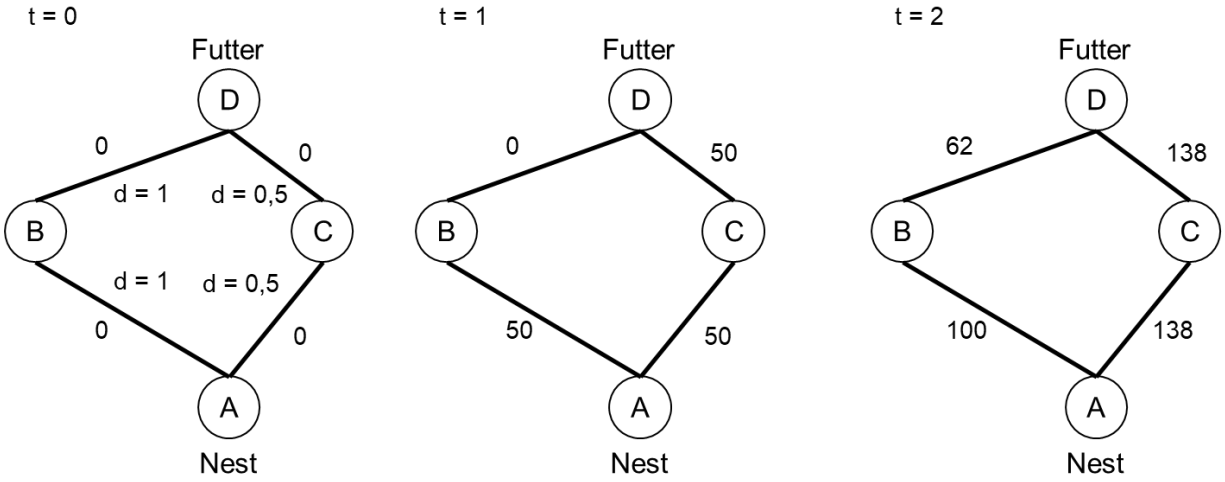


Abbildung 5.1: Doppelbrücke von Goss

In der Abbildung 5.1 ist der Versuchsaufbau des Experiments von Goss zu sehen. Dabei haben die Ameisen auf dem Weg zur Futterquelle zweimal die Möglichkeit zwischen einem längeren und einem kürzeren Weg zu wählen. Obwohl zu Beginn alle Wege gleichermaßen

durchlaufen werden, stellt sich nach einiger Zeit der Weg ACDEG für den größten Teil der Ameisen ein (vgl. Goss, 1989). Die Erklärung für dieses Phänomen liegt dabei in der Verwendung von Pheromonen. Ameisen hinterlassen bei ihrer Futtersuche Pheromone, welche von nachkommenden Ameisen wahrgenommen werden. Ohne Pheromone ist die Wegentscheidung der Ameisen zufällig. Beeinflusst wird die Entscheidung der Ameisen erst durch die Wahrnehmung von Pheromonen. Liegen zwei Wegmöglichkeiten mit zwei unterschiedlichen Pheromonkonzentrationen vor, so wird der Weg mit der höheren Pheromonkonzentration von den Ameisen bevorzugt. Da die Ameisen, die zu Beginn zufällig den kürzesten Weg gewählt haben, sich eher auf dem Rückweg befinden, liegt auf diesem Weg eine erhöhte Pheromonkonzentration vor. Diese bringt die nachkommenden Ameisen dazu, ebenfalls diesen Weg zu wählen, weshalb nach bestimmter Zeit der kürzeste Weg am häufigsten durchlaufen wird. Die folgende Abbildung 5.2 zeigt einen Prozess der Selbstorganisation durch indirekte Kommunikation (vgl. Dorigo et al., 1996).



Annahmen:

- Zu jedem Zeitpunkt t starten 100 Ameisen vom Nest zur Futtersuche.
- Die Ameisen können in einer Zeiteinheit eine Distanz d von 1 zurücklegen.
- Jede Ameise legt eine Pheromonmenge von 1 auf dem zurückgelegten Weg ab.
- Die Verdunstung der Pheromone wird nicht berücksichtigt.

Abbildung 5.2: Ablauf des Ameisenalgorithmus

Zum Zeitpunkt t = 0 befinden sich 100 Ameisen am Startpunkt im Nest und erreichen Punkt A. Da zu Beginn noch keine Pheromoninformation für die Ameisen vorhanden ist, entscheiden diese rein zufällig, welche Route sie wählen. Daher besteht am Anfang eine Wahrscheinlichkeit von 50 %, dass die Ameisen die Route über den Punkt B wählen und 50% das die Ameisen die Route C wählen (vgl. Dorigo et al., 1996).

Zum Zeitpunkt t=1 laufen also 50 Ameisen über den Punkt C und erreichen den Punkt D und somit kurz darauf das Ziel. Zum selben Zeitpunkt erreichen die restlichen 50 Ameisen erst den

Punkt B, welcher doppelt so weit entfernt ist. Wenn die Ameisen sich auf den Rückweg begeben, müssen sie erneut entscheiden, welchen Weg sie wählen. Nun haben die Ameisen jedoch eine andere Ausgangslage. Auf der einen Route liegt bereits eine Pheromonspur. Diese Spur ist nur auf der Route zu Punkt C vorhanden, da die Ameisen, welche die Route B gewählt haben, noch nicht an der Futterquelle angekommen sind. Daher entscheidet sich der größte Teil der 50 Ameisen für die Route über den Punkt C. Es wird in dem Beispiel von Goss von 76% ausgegangen. Dies bedeutet, dass sich 38 der 50 Ameisen für die Route über C entscheiden. Zum selben Zeitpunkt starten weitere 100 Ameisen vom Nest und gehen auf die Futtersuche. Da zu Beginn jeweils 50 Ameisen über B und 50 über C gelaufen sind, ist die Pheromonkonzentration der beiden Spuren gleich groß. Dies hat zur Folge, dass sich wieder 50 Ameisen für den Weg über B und 50 Ameisen über C entscheiden (vgl. Dorigo et al., 1996).

Zum Zeitpunkt t=2 erreichen bereits die ersten 38 Ameisen ihr Nest. Parallel dazu erreichen 50 Ameisen, die zum Zeitpunkt t=0 die Route über den Punkt B gewählt hatten, den Futterplatz. Wie bereits bei den vorherigen Ameisen, entscheidet sich ein Großteil dieser für den Weg über den Punkt C, da dieser eine erhöhte Pheromonkonzentration enthält. Die Ameisen, die nun von ihrem Nest aus starten, wählen mit einer höheren Wahrscheinlichkeit den Weg über den Punkt C. So kommt es dazu, dass mit fortschreitender Zeit der Weg über C deutlich bevorzugt wird (vgl. Goss, 1989).

Um die Futtersuche der Ameisen auch mit Hilfe von Formeln darstellen zu können, wird im Folgenden der Ameisenalgorithmus nach Dorigo (1999) beschrieben. Die Ausgangsposition ist in diesem Fall identisch mit dem vorherigen Versuch. Auf der Futtersuche ist zu Beginn keine Pheromonmenge vorhanden, weshalb Ameisen rein zufällig die Wegentscheidung treffen. Bei den nachkommenden Ameisen tritt dieser Fall nicht ein. In der folgenden Formel 5.1 wird die Entscheidungsregel dargestellt. Sie zeigt mit welcher Wahrscheinlichkeit sich Ameisen für die jeweiligen Wege entscheiden. Dabei wird die Entscheidung proportional zu der Menge an der bisherigen Pheromonmenge getroffen (vgl. Dorigo, 1999).

$$P_{ij}{}^{k} = \begin{cases} \dfrac{\tau_{ij}}{\sum_{j \in N_i} \tau_{ij}} & falls\ j \in N_i \\ 0 & falls\ j \notin N_i \end{cases} \qquad (5.1)$$

Hierbei wählen die Ameisen ebenfalls mit einer höheren Wahrscheinlichkeit die Wege, die die vorherigen Ameisen schon hinterlegt haben. Dafür wird an jedem Punkt der aktuelle Wert für τ_{ij} festgehalten. Dieser Wert ist auch nur für die Ameisen zugänglich, die genau diesen Punkt überqueren. So wird die indirekte Kommunikation der Ameisen nachgeahmt. Nachdem eine Ameise einen Punkt erreicht, wird der Wert über die lokale Information τ_{ij} gelesen und eine Zufallszahl in einem Intervall von [0,1) erzeugt. Je nachdem welcher Wert erzeugt wird, entscheidet sich die Ameise für den zugehörigen Weg, welcher in das

Wahrscheinlichkeitsintervall fällt. Erreichen die Ameisen den nächsten Punkt, so steigt der Wert der Pheromonmenge des Punktes um einen konstanten Faktor. Somit ist dieser Punkt beliebter für die nachkommenden Ameisen. Der gesamte Prozess wiederholt sich so lange, bis sich ein Weg herauskristallisiert, der am effektivsten für die Futtersuche ist. In der folgenden Abbildung 5.3 ist die Findung der Futtersuche durch künstliche Ameisen nochmal verdeutlich (vgl. Dorigo, 1999).

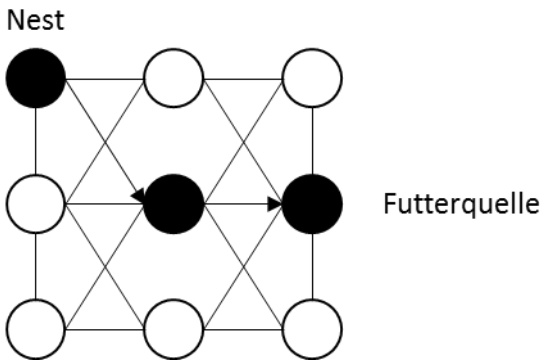


Abbildung 5.3: Futtersuche der künstlichen Ameisen (vgl. Dorigo, 1999)

Dieser Versuch zeigt, dass auch künstliche Ameisen in der Lage sind den kürzesten Weg zur Futterquelle zu finden. Jedoch ist dies nur als Ergebnis der gesamten Kolonie möglich, was wiederum zeigt, dass der Ameisenalgorithmus zu einer Form der Schwarmintelligenz gehört (vgl. Dorigo, 1999).

Da der Ameisenalgorithmus zu den metaheuristischen Optimierungsverfahren gehört und diese auf eine breite Menge an Optimierungsproblemen anzuwenden sein müssen, ist der Algorithmus noch weiterentwickelt worden. Die Modifikationen werden im Folgenden genauer beschrieben.

5.2 Ant-Colony-Optimierung-Metaheuristik

Bei der Ant-Colony-Optimierung handelt es sich um eine Metaheuristik, welche unterschiedliche Ausführungsmöglichkeiten besitzt. Es gibt zudem eine Vielzahl an Modifikationen, die den Ameisenalgorithmus erweitern, jedoch werden sie nicht alle berücksichtigt. Diese müssen auch nicht die Natur als Vorbild nehmen, jedoch sind einige Parallelen trotzdem zu erkennen. Sie sind notwendig, um die Einsatzmöglichkeiten und die Leistungsfähigkeit des Ameisenalgorithmus zu steigern. Der erste bedeutende Unterschied ist, dass sich künstliche Ameisen auf einem Graphen bewegen, welcher das zu lösende Problem abbildet. In der Realität bewegen sich Ameisen komplett frei. In der ACO-Metaheuristik bewegt sich die Ameise von einem Zustand zum anderen. Der zweite Unterschied zu den realen Ameisen ist, dass künstliche Ameisen zu ihrem künstlichen Gedächtnis in der Form der Pheromonspur noch ein zusätzliches Gedächtnis besitzen, das vorherige Schritte von einem

Zustand zum anderen Zustand speichert. Eine weitere Diskrepanz zu realen Ameisen, welche berücksichtigt werden kann, ist die Güte der Futterquelle. Abhängig von der Größe oder Qualität der Futterquelle ist es möglich, dass künstliche Ameise unterschiedliche Pheromonmengen ausscheiden. Zudem ist die Aktualisierung der Pheromonspur ein bedeutender Unterschied. Während diese Aktualisierung bei realen Ameisen parallel im Hintergrund abläuft, besteht die Möglichkeit bei künstlichen Ameisen, dass die erste Aktualisierung der Pheromonspur erst nach der Findung der ersten Lösung der Futtersuche stattfindet. Der letzte Unterschied zum Ameisenalgorithmus ist die Verwendung beziehungsweise Erweiterung durch Einbringung weiterer Optimierungsverfahren wie dem lokalen Suchverfahren oder ähnlichem. Wichtig ist dabei die Berücksichtigung der Steigerung des Rechenaufwands bei Erhöhung der Nachbarschaft oder Einbringung weiterer Modifikationen (vgl. Dorigo, 1999).

Bei der Betrachtung der unterschiedlichen Ausführungsmöglichkeiten der Ant-Colony-Optimierung, kann festgestellt werden, dass sich die ACO-Algorithmen in der Wegentscheidung und in dem Pheromonupdate unterscheiden. Außerdem gibt es bei diesen Algorithmen unterschiedliche Ansätze die Pheromonmenge zu definieren. Um das Prinzip der unterschiedlichen ACO-Algorithmen besser nachvollziehen zu können, werden diese im Folgenden detailliert erläutert. Dabei handelt es sich bei den zu untersuchenden Algorithmen um den Ant-System-, Ant-Colony-System- und den MAX-MIN-Ant-System-Algorithmus.

Ant System

Bei dem Ant System (AS) handelt es sich um den ersten ACO-Algorithmus, der entwickelt worden ist (vgl. Dorigo, 1996). Es gibt mehrere Ausführungen des AS-Algorithmus, wobei sich die folgenden Formeln auf die Ausführung des Ant System von Dorigo beziehen. Die Pheromonspuren τ_{ij} werden zunächst als konstant festgelegt. Es besteht auch die Möglichkeit Pheromonspuren einer guten Lösung zu verwenden, um bereits mit guten Lösungen anzufangen. Bei der Wegentscheidung, die die künstlichen Ameisen treffen, wird die folgende Formel 5.2 verwendet (vgl. Dorigo, 1996):

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{u \in N_i^k} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}} & falls\ j \in N_i^k \\ 0 & falls\ j \notin N_i^k \end{cases} \tag{5.2}$$

Hierbei ist deutlich zu erkennen, dass die Übergangswahrscheinlichkeit p_{ij}^{k} den Wert Null annimmt, wenn j kein Element der Nachbarschaft ist. Bei den restlichen Möglichkeiten wird eine Wahrscheinlichkeit in Abhängigkeit der Pheromonmenge und der heuristischen Information berechnet. Dabei gibt es die beiden Parameter α und β, welche die Gewichtung der Pheromonmenge und der heuristischen Information wiederspiegeln. So wird bei der Wahl

von α = 0 die Menge die Pheromone nicht berücksichtigt. Andersherum wird bei der Wahl von β = 0 die heuristische Information vernachlässigt. Steigen die Parameter, so haben sie unterschiedliche Auswirkungen auf den weiteren Verlauf der Wegentscheidung von künstlichen Ameisen (vgl. Dorigo, 2004). Nachdem die künstlichen Ameisen Lösungen generiert haben, folgt das Pheromonupdate, um den nachkommenden Ameisen neue Informationen zu liefern, die sie nutzen können. Bei dem AS-Algorithmus gibt es auch beim Pheromonupdate unterschiedliche Ausführungsmöglichkeiten. Im weiteren Verlauf wird jedoch nur die Ant-Cycle-Variante beschrieben (vgl. Dorigo, 1996). Hierbei wird die zurückgelegte Pheromonmenge nach der Konstruktion einer Lösung auf alle Pfade addiert. Außerdem spielt bei dem AS-Algorithmus die Pheromonverdunstung eine bedeutende Rolle. Diese wird in der Formel 5.3 für das Pheromonupdate ersichtlich (vgl. Dorigo, 1996):

$$(1-\rho)\cdot\tau_{ij}+\sum_{k=1}^{m}{\Delta\tau_{ij}}^{k}\rightarrow\tau_{ij} \qquad \forall(i,j)\in L \qquad (5.3)$$

Der Parameter ρ beschreibt die Pheromonverdunstung, welche Werte zwischen 0 und 1 annehmen kann. ${\Delta\tau_{ij}}^{k}$ ist die von der Ameise *k* hinterlegte Pheromonmenge. Mit Hilfe der Verdunstungsrate ist es möglich, die Suche der Ameisen zu beeinflussen. So wird durch eine hohe Wahl der Verdunstungsrate eine breite Suche des Lösungsraums sichergestellt. Durch eine niedrige Wahl der Pheromonverdunstung hingegen, stagniert die Konstruktion der Lösung bereits nach einer geringen Anzahl an Iterationen (vgl. Engelbrecht, 2005). Die Pheromonmenge ${\Delta\tau_{ij}}^{k}$ berechnet sich bei dem AS-Algorithmus wie folgt (vgl. Dorigo, 1996):

$${\Delta\tau_{ij}}^{k}=\begin{cases}\frac{Q}{f(s^k)}\ falls\ (i,j) & Element\ der\ Lösung\ s^k\\ 0 & sonst\end{cases} \qquad (5.4)$$

Dabei beschreibt der Parameter Q die Güte der Lösung und s^k die generierte Lösung der künstlichen Ameise k.

Um die Umsetzung des Algorithmus zu veranschaulichen, wird im Folgenden eine mögliche Variante des Pseudo-Codes beschrieben (vgl. Bonabeau et al., 1999). Dabei wird lediglich das schnelle Erreichen eines Endpunktes angestrebt.

1: **Name:** AS-Algorithmus
2: **For each** Kante (i, j)
3: $\tau_{ij} = \tau_0$
4: **End For**
5: $L^+ = 1$ //Es gibt noch kein Minimum erreicht
6: **For** t = 1 **to** t_{max} //Hauptschleife
7: Fülle die Liste J_i^k für alle Ameisen k und alle Knoten i

8: **For** k = 1 **to** m //Für jede Ameise eine Tour finden
9: Knoten i = Anfangsknoten
10: T^k besteht nur aus Anfangsknoten
11: **While** Knote i **not** = Endknoten
12: Wähle den nächsten Knoten j mit der Wahrscheinlichkeit
13: $$p_{ij}^k = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{u \in N_i^k} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}$$
14: wobei der Knoten j in J_i^k enthalten sein muss
15: Aktualisiere Liste J_i^k
16: Füge den Knoten j zu T^k hinzu
17: Knoten i = Knoten j
18: **End While**
19: **End For**
20: **For** k = 1 **to** m
21: Berechne die Tourlänge L^k der Tour T^k der Ameise k
22: **If** $L^k < L^+$ oder $L^+ = -1$ **then**
23: Aktualisiere T^+ und L^+
24: **End If**
25: **End for**
26: **For each** Kante (i, j) //Berechne Pheromonupdate
27: $$(1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta {\tau_{ij}}^k \rightarrow \tau_{ij}$$
28: Wobei
29: $$\Delta {\tau_{ij}}^k = \frac{Q}{f(s^k)} \; falls \; (i,j)$$
30: **End For**
31: **End For**

Ant Colony System

Der Ant-Colony-System-Algorithmus (ACS) ist eine modifizierte Variante des AS-Algorithmus (vgl. Dorigo, 1997). Dabei unterscheidet sich der Algorithmus sowohl in der Wegentscheidung als auch bei dem Pheromonupdate der künstlichen Ameisen. Bei der Wegentscheidung verwendet der ACS-Algorithmus zwei unterschiedliche Vorgehensweisen. Zum einen behilft sich der Algorithmus mit den Informationen der vorherigen künstlichen Ameisen. Zum anderen wird auch beim ACS-Algorithmus wie bei dem AS-Algorithmus eine Wahrscheinlichkeitsberechnung für die möglichen Pfade durchgeführt. Dabei wird die Vorgehensweise zufällig durch einen gleichverteilten Parameter q_0 gewählt. Die Vorgehensweisen lassen sich wie folgt beschreiben (vgl. Dorigo, 2004):

$$j = \begin{cases} argmax_{l \in N_i^k}\{\tau_{il} \cdot [\eta_{il}]^\beta\} & falls\ q \leq q_0 \\ J & sonst \end{cases} \quad (5.5)$$

Liegt der zufällig generierte Parameter q unter einem konstanten festgelegten Parameter q_0, so folgt die nachkommende künstliche Ameise *k* dem besten Pfad aus den vorherigen Iterationen. Liegt der Wert für q über dem Wert q_0, so wird eine Wahrscheinlichkeitsberechnung durchgeführt. Diese wird wie folgt berechnet (vgl. Dorigo, 2004):

$$J = p_{ij}^k = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{u \in N_i^k} [\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta} \quad (5.6)$$

Es wird deutlich, dass die Wahl des Parameters q_0 einen großen Einfluss auf den Ablauf der künstlichen Ameisen hat. So wird durch eine hohe Wahl des Parameters q_0 eine breite Suche des Lösungsraumes verhindert. Andersherum wird durch eine niedrige Wahl des Parameters q_0 zwar eine breite Suche des Lösungsraums gewährleistet, jedoch werden gute Lösungen aus vorherigen Iterationen kaum berücksichtigt. Aufgrund der Tatsache, dass bei den AS- und MMAS-Algorithmen in jedem Fall Rechenoperationen stattfinden, besitzen diese vergleichsweise zu den ACS-Algorithmen deutlich höhere Laufzeiten (vgl. Dorigo, 2004).

Ein deutlicher Unterschied des ACS-Algorithmus zum AS-Algorithmus liegt im Pheromonupdate vor. Bei dem ACS-Algorithmus wird zwischen dem lokalen und dem globalen Pheromonupdate unterschieden. Beim globalen Pheromonupdate hinterlegt lediglich die Ameise, die die beste Lösung generiert hat, eine Pheromonmenge. Auf den restlichen Pfaden wird dabei nichts geändert. Somit ergibt sich die Formel 5.7 für das globale Pheromonupdate wie folgt (vgl. Dorigo, 2004):

$$(1 - \rho) \cdot \tau_{ij} + \Delta {\tau_{ij}}^{best} \rightarrow \tau_{ij} \qquad \forall (i,j) \in L \quad (5.7)$$

Die Pheromonmenge $\Delta {\tau_{ij}}^{best}$ des besten Pfades wird dabei mit der folgenden Formel 5.8 berechnet (vgl. Dorigo, 2004).

$$\Delta {\tau_{ij}}^{best} = \frac{1}{f(s^{best})} \quad (5.8)$$

Bei dem lokalen Pheromonupdate handelt es sich um eine Maßnahme die künstlichen Ameisen über Pfade laufen zu lassen, welche bisher selten gewählt worden sind. So wird eine frühzeitige Stagnation der Ergebnisse verhindert. Dafür wird, nachdem eine Ameise k einen Pfad *ij* gewählt hat, die Pheromonmenge auf dem Pfad verringert. Dies hat zur Folge, dass dieser Pfad für die nachkommenden Ameisen unattraktiv wird. Das lokale Pheromonupdate wird mittels der Formel 5.9 beschrieben (vgl. Dorigo, 2004):

$$(1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_0 \rightarrow \tau_{ij} \quad (5.9)$$

Die Formel zeigt, dass das lokale Update der Pheromonmenge nicht unter den Wert τ_0 sinken kann. Wichtig ist zu beachten, dass der Wert für τ_0 deutlich geringer als $\frac{1}{f(s^{best})}$ sein muss, um plausible Ergebnisse für die Pheromonmenge auf den Pfaden zu erhalten (vgl. Dorigo, 2004).

Analog zum AS-Algorithmus wird auch hierbei für die Umsetzung des Algorithmus der Pseudo-Code betrachtet (vgl. Bonabeau et al., 1999).

1: **Name:** ACS-Algorithmus
2: **For each** Kante (i, j)
3: $\tau_{ij} = \tau_0$
4: **End For**
5: $L^+ = -1$ //es gibt noch keine minimale Tourlänge
6:
7: **For** t = 1 **to** t_{max} //Hauptschleife
8: Fülle die Liste für alle Ameisen k und alle Knoten J_i^k
9:
10: **For** k = 1 **to** m // Für jede Ameise eine Tour finden
11: Knoten i = Anfangsknoten
12: T^k besteht nur aus dem Anfangsknoten
13:
14: **While** Knoten i **not** = Endknoten
15: **If** wenigstens ein Knoten j in der Kandidatenliste von i enthalten
16: ist, der noch nicht besucht wurde **then**
17:
18: Errechne eine Zufallszahl q
19:
20: Wähle den nächsten Knoten j aus den Knoten der
21: Kandidatenliste mittels der Formel:

22:

$$j = \begin{cases} argmax_{l \in N_i^k}\{\tau_{il} \cdot [\eta_{il}]^\beta\} & falls\ q \leq q_0 \\ J & sonst \end{cases}$$

23: Mit

$$J = p_{ij}^k = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{u \in N_i^k}[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}$$

24: gewählt wird.
25: **Else**
26: Wähle den nähest gelegenen Knoten aus J_i^k als Knoten j
27**:** **End If**

28: Berechne die neue Menge an Pheromonen laut der lokalen
29: Pheromon-Update-Regel:

30:
$$(1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_0 \rightarrow \tau_{ij}$$

31: Aktualisiere die Liste J_i^k
32: Aktualisiere die Kandidatenliste von Knoten i

33: Füge den Knoten j in T^k hinzu
34: Knoten i = Knoten j
35: **End While**
36: **End For**
37:
38: **For** k = 1 **to** m
39: Berechne die Tourlänge L^k der Tour T^k der Ameise k
40: **If** $L^k < L^+$ oder $L^+ = -1$ **then**
41: Aktualisiere T^+ und L^+
42: **End if**
43: **End For**
44: **For each** Kante (i, j) ∈ T^+ //Pheromonupdate
45: Berechne die neue Menge an Pheromonen laut der Pheromonupdate- Regel:
46:
47: $$(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}{}^{best} \rightarrow \tau_{ij}$$

48: wobei $\Delta\tau_{ij} = \frac{1}{L^+}$
49: **End For**
50: **End For**

MAX-MIN Ant System

Bei dem MAX-MIN-Ant-System-Algorithmus (MMAS) gibt es im Vergleich zum AS-Algorithmus drei wesentliche Unterschiede (vgl. Stützle, 2000). Der erste Unterschied ist, dass das Pheromonupdate wie beim ACS-Algorithmus nur auf den Pfaden mit den besten Lösungen erfolgt. Außerdem werden die Pheromonmengen in einem bestimmten Intervall $[\tau_{min}, \tau_{max}]$ gehalten, um so frühzeitige Stagnation zu verhindern. Zusätzlich dazu wird die zu Beginn gewählte Pheromonmenge bewusst hoch gewählt, um so eine breite Suche des Lösungsraums zu ermöglichen. Die Wegentscheidung erfolgt bei dem MMAS-Algorithmus nach der Formel 5.2, welche bereits bei der Wegentscheidung des AS-Algorithmus verwendet worden ist (vgl. Stützle, 2000).

Das Pheromonupdate, das nur auf den Pfaden mit der besten Lösung erfolgt, wird wie folgt berechnet (vgl. Stützle, 2000):

$$(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}{}^{best} \rightarrow \tau_{ij} \qquad \forall(i,j) \in L \qquad (5.10)$$

Die Formel 5.10 ist identisch zu der Formel 5.7 des Pheromonupdates beim ACS-Algorithmus. Hier besteht jedoch die Möglichkeit für $\Delta\tau_{ij}{}^{best}$ unterschiedliche Pheromonmengen zu wählen. Zum einen kann die beste Lösung der aktuellen Iteration gewählt werden. Zum anderen kann die beste Lösung des gesamten Suchprozesses gewählt werden (vgl. Stützle, 2000). Der Pseudo-Code für den MMAS-Algorithmus erfolgt analog zum AS- und ACS-Algorithmus. Da

der Pseudo-Code bei diesen Algorithmen bereits genau beschrieben worden ist, wird beim MMAS-Algorithmus nicht mehr darauf eingegangen.

6 Das Simulationsmodell

In der vorliegenden Bachelorarbeit wird ein modulares Produktionssystem nachgebildet. Dabei setzt sich das Produktionssystem aus fünf Arbeitsstationen zusammen. Die Arbeitsstationen beinhalten eine variable Anzahl an Maschinen. Es besteht die Möglichkeit jede Arbeitsstation mit einer bis zu fünf Maschinen zu belegen. Zur Veranschaulichung des Produktionssystems wird die Abbildung 6.1 betrachtet (vgl. Law 2007, S. 694).

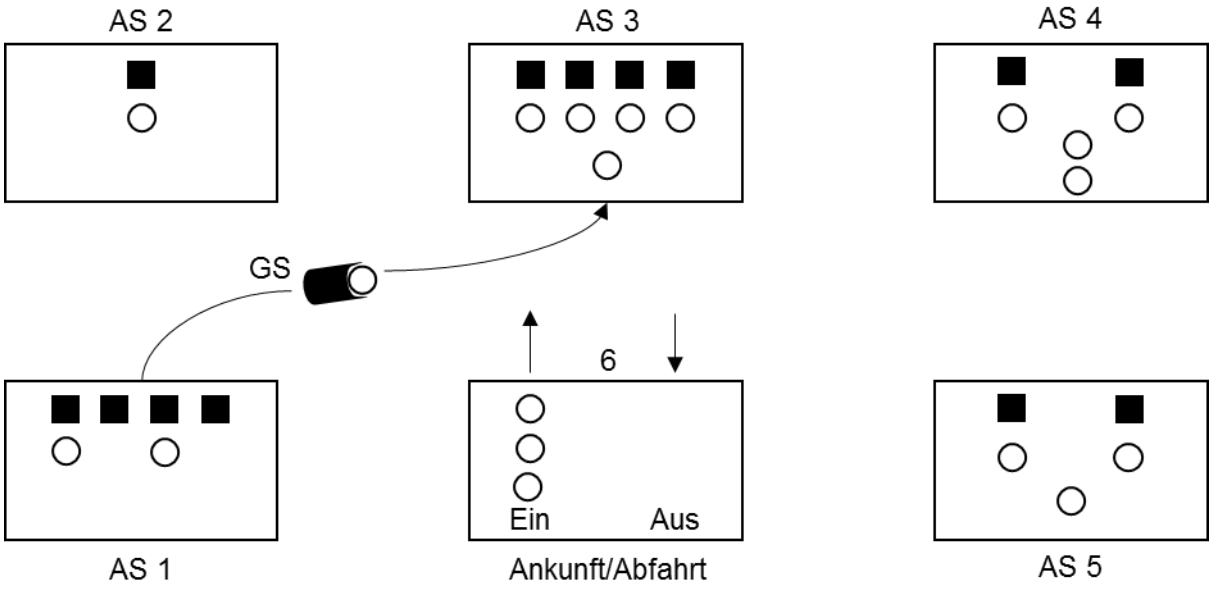


Abbildung 6.1: Das Produktionssystem

In der Abbildung 6.1 sind sechs Stationen dargestellt. Fünf dieser sechs Stationen sind Arbeitsstationen. Die letzte Station dient als Zwischenlager für die Aufträge. Die Aufträge sind als Kreise und die Maschinen als schwarze Quadrate dargestellt. Zu Beginn wird eine Auftragsliste generiert. Die Aufträge werden dabei in den unterschiedlichen Arbeitsstationen bearbeitet. Der Transport der Aufträge erfolgt mittels Gabelstaplern, welche in der Abbildung 6.1 durch „GS“ gekennzeichnet sind. Dem Produktionssystem stehen drei Gabelstapler zur Verfügung. Aus der Bearbeitung der Aufträge resultieren zwei Problemstellungen. Das erste Problem befasst sich mit der Maschinenkonstellation innerhalb der einzelnen Arbeitsstationen. Hierbei ist eine festgelegte Reihenfolge von Aufträgen vorgegeben, welche in jedem Lauf zufällig generiert wird. Das Ziel ist zum einen die Minimierung der Zykluszeit. Dazu ist es erforderlich die Maschinenkonstellation zu finden, welche die vorgegebenen Aufträge in kürzester Zeit bearbeitet. Zum anderen wird die Maximierung der Maschinenauslastung angestrebt (vgl. Kapitel 2.2). In der Abbildung 6.1 sind die Maschinen in der Arbeitsstation 1 unterlastet und in der Arbeitsstation 4 überlastet. So sind in der abgebildeten Momentaufnahme in der Arbeitsstation 1 mehr Maschinen als Aufträge vorhanden. Dagegen befinden sich in der Arbeitsstation 4 mehr Aufträge als Maschinen. Dies führt dazu, dass die

Aufträge nicht bearbeitet werden können. Dieser Aspekt ist wichtig für die Effizienz des Produktionssystems. Es wäre möglich, jede Arbeitsstation mit der maximalen Anzahl an Maschinen zu füllen. Dabei würde die Zykluszeit auf der einen Seite schrumpfen, auf der anderen Seite wären einige Maschinen über einen großen Zeitraum unbenutzt. Um die Effizienz des Produktionssystems zu steigern, ist es also notwendig die Standzeiten der Maschinen zu verringern (vgl. Kapitel 2.2). Repräsentativ für die Effizienz des modular modellierten Produktionssystems ist der Fitnesswert (FW). Um auf den Fitnesswert schließen zu können, muss zunächst der Flow-Faktor (FFWT) berechnet werden. Dieser ergibt sich aus der folgenden Formel 6.1:

$$FFWT = \frac{(Endzeit - Startzeit)}{(Transport + Bearbeitung)} \tag{6.1}$$

Der Flow-Faktor gibt die Effizienz einzelner Maschinen im Produktionssystem wieder. Dabei wird die tatsächliche Bearbeitungszeit in Verhältnis zur optimalen Bearbeitungszeit gesetzt. In dem vorliegenden Simulationsmodell ist ein Flow-Faktor von drei optimal. Aus dem Flow-Faktor resultiert anschließend der Fitnesswert. Dieser lässt sich wie folgt berechnen:

$$FW = |(\emptyset FFWT - 3)| \cdot Anzahl\ der\ Maschinen \tag{6.2}$$

Für jede Maschine wird ein individueller Flow-Faktor berechnet. Um eine Aussage über das Produktionssystem treffen zu können, wird der durchschnittliche Flow-Faktor aller Maschinen berechnet. Dieser wird mit drei subtrahiert und anschließend zum Betrag genommen. So wird sichergestellt, dass der beste Flow-Faktor gegen den Wert Null strebt. Durch die Multiplikation mit der Anzahl der Maschinen wird die Effizienz des Produktionssystems in Abhängigkeit der Maschinenauslastung berücksichtigt. Jede Maschinenkonstellation wird 20 Mal durchlaufen und erzeugt 20 Fitnesswerte. Aus diesen Fitnesswerten wird ein durchschnittlicher Fitnesswert berechnet, welcher vom Simulationsprogramm ausgegeben wird. Je geringer der Fitnesswert, desto besser und effizienter ist das Produktionssystem. Das globale Optimum konvergiert gegen den Wert Null.

In der folgenden Tabelle 6.1 sind die Bearbeitungszeiten der einzelnen Jobtypen in den jeweiligen Arbeitsstationen zu sehen. Dabei spielt es keine Rolle, von welcher Maschine ein Jobtyp innerhalb einer Arbeitsstation bearbeitet wird.

Jobtyp	AS 1	AS 2	AS 3	AS 4	AS 5
1	0,15	0,1	0,25		0,3
2	0,2		0,3	0,15	
3	0,35	0,15	0,2	0,2	0,1

Tabelle 6.1: Bearbeitungszeiten

Es ist deutlich zu erkennen, dass nicht alle Jobtypen in allen Arbeitsstationen bearbeitet werden. Es ist außerdem für jeden Jobtyp festgelegt, in welcher Reihenfolge dieser die Arbeitsstationen durchläuft. Diese Reihenfolge ist in der Tabelle 6.2 für jeden Jobtypen dargestellt.

Jobtyp	Bearbeitungsreihenfolge in AS
1	3, 1, 2, 5
2	4, 1, 3
3	2, 5, 1, 4, 3

Tabelle 6.2: Bearbeitungsreihenfolge

7 Ant-Colony-Optimierung in der Simulation

In diesem Kapitel wird die Ant-Colony-Optimierung an dem vorliegenden Simulationsmodell angewendet. Dazu werden unterschiedliche Ausführungsmöglichkeiten der Ant-Colony-Optimierung gegenübergestellt und miteinander verglichen. Es werden Aspekte wie die Wegentscheidung, die Pheromonverdunstung und das Pheromonupdate untersucht (vgl. Kapitel 5.2). Nach der Gegenüberstellung der verschiedenen Ausführungsmöglichkeiten werden zwei Ausführungen ausgewählt. Die erste Ausführung wird zur Bestimmung der Maschinenkonstellation des modular modellierten Produktionssystems genutzt. Die zweite Ausführung wird für das Scheduling von Auftragsreihenfolgen genutzt.

7.1 Ant-Colony-Optimierung zur Variation des Produktionssystems

Da das Simulationsmodell eine Black-Box ist, steht dem Anwender der Simulation keine visuelle Darstellung der möglichen Maschinenkonstellationen zur Verfügung (vgl. Kapitel 3.3). Um dennoch die Vorgehensweise der künstlichen Ameisen nachvollziehen zu können, wird die Abbildung 7.1 betrachtet.

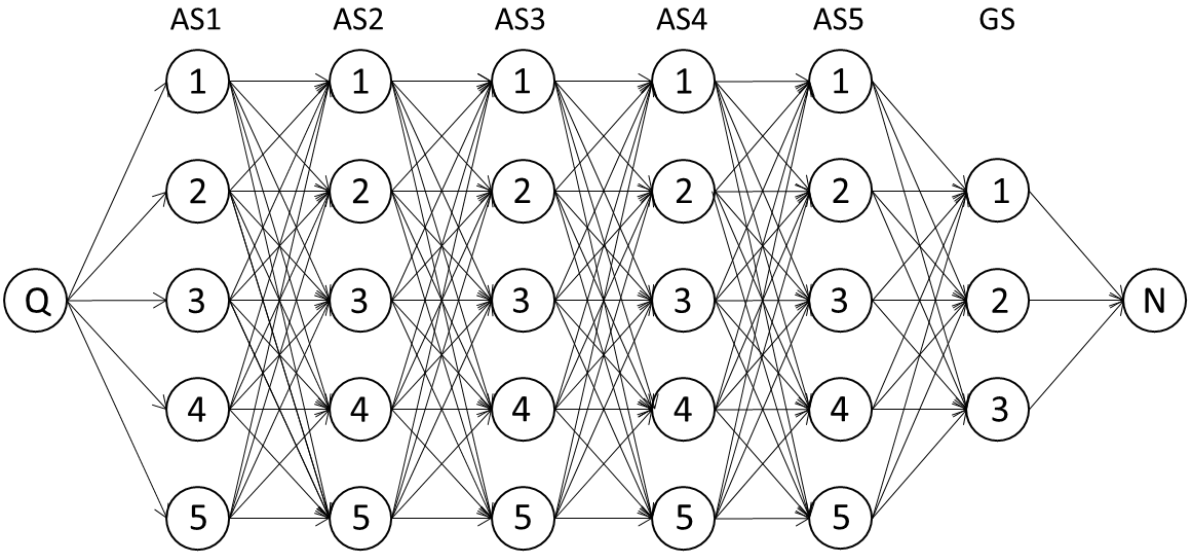


Abbildung 7.1: Mögliche Maschinenkonstellationen

Wie in der Realität verfolgen die künstlichen Ameisen in dem Simulationsmodell ebenfalls das Ziel, den kürzesten Weg zur Futterquelle zu finden. Anders als bei realen Ameisen sind die künstlichen Ameisen in ihrer Pfadfindung eingeschränkt (vgl. Kapitel 5.1). Diese festgelegten Pfade sind in der Abbildung 7.1 ersichtlich. Jede der fünf Arbeitsstationen kann mit Maschinen befüllt werden. Die Anzahl der Maschinen kann dabei zwischen eins und fünf variieren. In dem vorliegenden Simulationsmodell gibt es für die unterschiedlichen Anzahlen der Maschinen mehrere Möglichkeiten diese darzustellen. In dieser Bachelorarbeit ist die vorliegende

Problemstellung durch zwei verschiedene Ansätze gelöst worden. Zur Veranschaulichung des ersten Lösungsansatzes wird die folgende Abbildung 7.2 betrachtet.

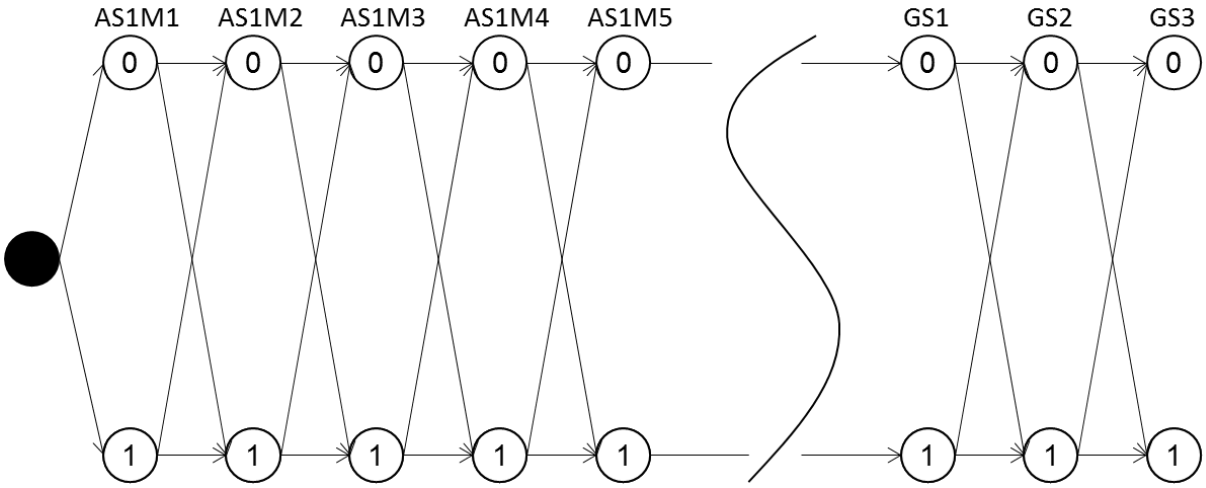


Abbildung 7.2: Belegung einzelner Maschinen

Innerhalb des Programms existiert ein Vektor, welcher die Belegung jeder Arbeitsstation mit Maschinen beschreibt. Außerdem wird die Anzahl der Gabelstapler innerhalb des Vektors festgelegt. Der Vektor setzt sich aus booleschen Werten zusammen. So stehen der künstlichen Ameise an jeder Station zwei potentielle Wege zur nächsten Station zur Verfügung. Von Station zu Station wird entschieden, ob die jeweilige Maschine in der Arbeitsstation existiert. So werden insgesamt 25 Entscheidungen für die Maschinen getroffen. Anschließend ist die Anzahl der Transportmittel zu bestimmen, welche die Aufträge von einer Arbeitsstation zur anderen transportiert. Dazu stehen in dem Simulationsmodell drei Gabelstapler zur Verfügung. Diese sind ebenfalls als drei Stationen in der Abbildung 7.1 dargestellt. Das beschriebene Vorgehen der künstlichen Ameise liefert eine Lösungsmöglichkeit. Die Gesamtanzahl der Lösungsmöglichkeiten umfasst dabei:

$$A = n^{m+g} \tag{7.1}$$

Dabei beschreibt n die Anzahl der Entscheidungsmöglichkeiten pro Station. Diese können die Werte „0“ oder „1“ annehmen. Entscheidet sich die künstliche Ameise für den Wert „0“, so gibt es in dieser Station keine Maschine. Wird jedoch der Wert „1“ angenommen, so existiert in dieser Station eine Maschine. Die Variable m ist als die Gesamtanzahl der Maschinen im Produktionssystem definiert. Der letzte Parameter g repräsentiert die Anzahl der Gabelstapler. In dem vorliegenden Simulationsmodell nehmen die Variablen Werte an, welche in der folgenden Tabelle 7.1 abgebildet sind.

Variable	Wert
n	2
m	25
g	3

Tabelle 7.1: Werte des Produktionssystems

Die Werte aus Tabelle 7.1 werden in die Formel 7.1 eingesetzt. So beläuft sich die Gesamtanzahl der Lösungsmöglichkeiten auf:

$$A = 2^{25+3} = 268.435.456 \quad (7.2)$$

Unter diesen Möglichkeiten befinden sich Maschinenkonstellationen, welche nicht auftreten dürfen. So muss in jeder Arbeitsstation mindestens eine Maschine vorhanden sein, um einen Fitnesswert generieren zu können. Eine Arbeitsstation setzt sich aus 5 Stationen der Abbildung 7.2 zusammen. Aufgrund der Tatsache, dass die Bearbeitung eines Jobtyps unabhängig von der Maschine innerhalb einer Arbeitsstation abläuft, repräsentiert eine hohe Anzahl an Vektoren identische Maschinenkonstellationen. Nach der Umsetzung dieser Implementierungsmöglichkeit sind hohe Simulationszeiten festgestellt worden. Dabei sind immens hohe Maschinenkonstellationen erzeugt worden, welchen keinen Fitnesswert liefern. Aus diesem Grund wird eine alternative Lösungsmöglichkeit umgesetzt. Bei dieser Lösungsmöglichkeit wird der Vektor nicht direkt verändert. Es wird zunächst ein Baum erzeugt. Dieser schließt sowohl die nicht möglichen Maschinenkonstellationen als auch die identischen Maschinenkonstellationen aus. Der erzeugte Baum wird in der folgenden Abbildung 7.3 dargestellt.

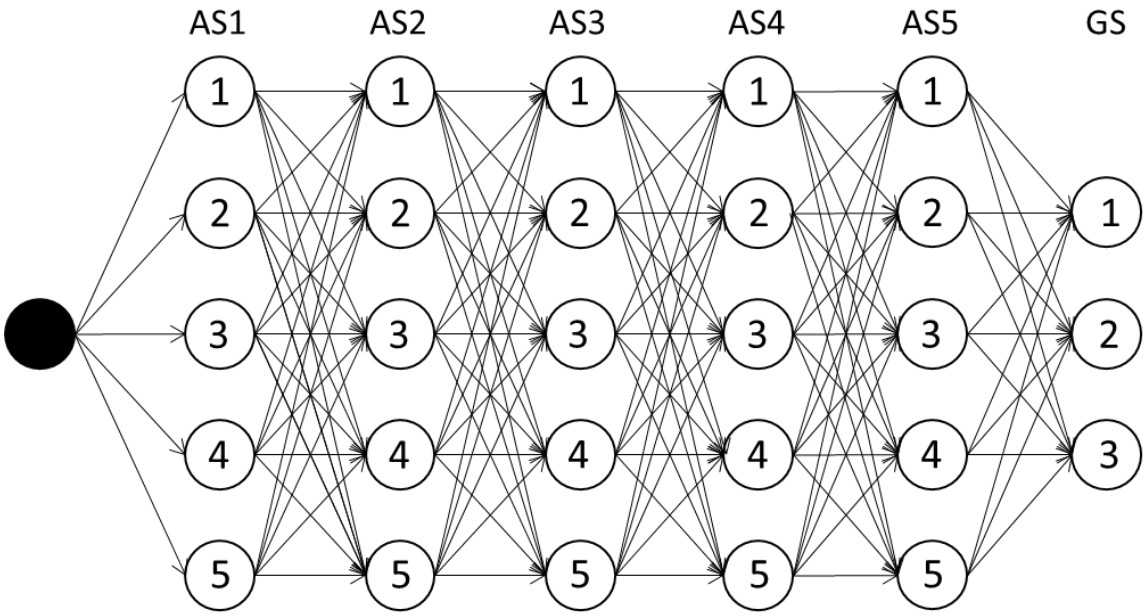


Abbildung 7.3: Belegung der Arbeitsstation

Nach dem Ausschluss der unmöglichen und identischen Maschinenkonstellationen, lässt sich nun die Gesamtanzahl der Lösungsmöglichkeiten wie folgt berechnen:

$$A = 5^5 \cdot 3 = 9.375 \tag{7.3}$$

In der Simulation laufen die künstlichen Ameisen einzelne Pfade des Baums ab. Anschließend werden die Pfade mit Hilfe einer Übersetzungsfunktion an den Vektor weitergegeben. Erst dann ist die Berechnung des Fitnesswerts möglich. Bei der Gegenüberstellung der Implementierungsmöglichkeiten wird deutlich, dass die Gesamtanzahl der Lösungsmöglichkeiten durch die Erzeugung des Baumes stark reduziert wird. Daraus resultieren kürzere Simulationsseiten und eine bessere Ausgabe (vgl. Kapitel 5.2).

Bei solchen Anwendungsbeispielen mit einer hohen Anzahl an Lösungsmöglichkeiten, spielen Metaheuristiken für die Optimierung eine immer größer werdende Rolle. Diese sind in der Lage, in einer bestimmten Zeit eine optimale Lösung zu finden. Dabei entspricht die optimale Lösung nicht zwingend dem globalen Optimum (vgl. Kapitel 4.3). Zur Findung einer optimalen Lösung wird in dem vorliegenden Simulationsmodell die ACO-Metaheuristik verwendet. Es existieren unterschiedliche Ausführungsmöglichkeiten der Ant-Colony-Optimierung (vgl. Kapitel 5.2). Die Aufgabe liegt nun darin, die für das vorliegende Simulationsmodell geeignetste Ausführung der Ant-Colony-Optimierung zu finden.

7.1.1 Auswahl der Ant-Colony-Optimierung-Ausführung

Die Hauptmerkmale bei der Ausführung der Ant-Colony-Optimierung sind die Wegentscheidung, die Pheromonverdunstung und das Pheromonupdate (vgl. Kapitel 5.2).

Wegentscheidung

Zur Beurteilung der Wegentscheidung werden die ACO-Algorithmen AS, MMAS und ACS in Betracht gezogen. Der Aspekt der Wegentscheidung ist bei den Algorithmen AS und MMAS identisch (vgl. Kapitel 5.2). Um die Unterschiede zwischen den drei ACO-Ausführungen zu verdeutlichen, wird für den weiteren Verlauf die Abbildung 7.4 betrachtet.

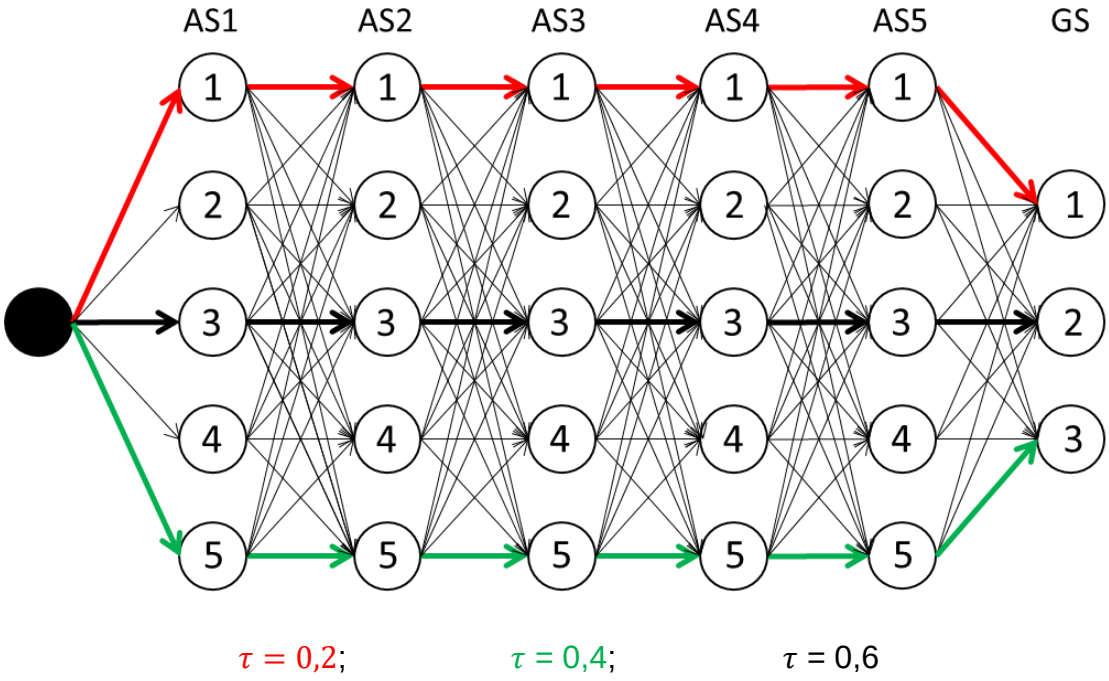


Abbildung 7.4: Pheromonspuren nach der ersten Iteration

Sowohl beim AS als auch beim MMAS und ACS ist jeder Pfad mit derselben Pheromonmenge gewichtet. Aus diesem Grund erfolgt die Wegentscheidung der künstlichen Ameise rein zufällig, wobei eine Gleichverteilung hinterlegt ist. Läuft die künstliche Ameise einen Pfad ab, so wird der Fitnesswert des Pfades berechnet. Der Fitnesswert ist essentiell für die Berechnung der Pheromonmenge τ_{ij}. Erhöht sich der Fitnesswert, so erhöht sich ebenfalls die Pheromonmenge des Pfades. Die Pheromonmenge lässt sich in Abhängigkeit des Fitnesswertes wie folgt berechnen:

$$\tau_{ij} = \frac{1}{\sum FW_{ij}} \tag{7.4}$$

In der Simulation wird diese Pheromonmenge von den künstlichen Ameisen auf dem Rückweg zum Nest zurückgelegt. Dabei wählen die künstlichen Ameisen denselben Pfad für den Rückweg, welchen sie als Hinweg gewählt haben (vgl. Kapitel 5.2). Erst nach diesem Schritt wird die berechnete Pheromonmenge auf die aktuell vorhandene Pheromonmenge addiert. Die Abbildung 7.4 zeigt schematisch die Pheromonspuren nach der ersten Iteration. Der beste Fitnesswert resultiert aus dem schwarzen Pfad. Dies ist der Grund, weshalb auf diesem Pfad die höchste Pheromonmenge vorhanden ist. Die Übergangswahrscheinlichkeit der künstlichen

Ameisen vom Knoten i zum Knoten j ist nun nicht mehr gleichverteilt. Die Übergangswahrscheinlichkeit bei dem AS- und MMAS-Algorithmus berechnet sich durch die Formel 5.2 (vgl. Kapitel 5.2):

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{u \in N_i^k} [\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta} & falls\ j \in N_i^k \\ 0 & falls\ j \notin N_i^k \end{cases}$$

Um die Formel an dem vorliegenden Simulationsmodell zu verdeutlichen, wird die Annahme getroffen, dass $\propto = 1$ und $\beta = 0$ sind. In diesem Fall werden lediglich die Pheromonspuren zur Wegentscheidung berücksichtigt (vgl. Kapitel 5.2). Da j bei der Variation modularer Produktions-systeme immer Element von N_i^k ist, tritt der zweite Fall für die Übergangswahrscheinlichkeit nie ein. Im Folgenden wird der schwarze Pfad untersucht. Somit ergibt sich folgende Rechnung:

$$p_{ij}^k = \frac{[\tau_{ij}]^1 \cdot [\eta_{ij}]^0}{\sum_{u \in N_i^k} [\tau_{ij}]^1 \cdot [\eta_{ij}]^0} = \frac{[0{,}6]^1}{0{,}6 + 0{,}4 + 0{,}2} = 0{,}5 \qquad (7.5)$$

Die Wahrscheinlichkeit, dass eine künstliche Ameise bei der zweiten Iteration den schwarzen Pfad wählt, beträgt also 50 %. Analog berechnen sich die Übergangswahrscheinlichkeiten für die restlichen Pfade. Die Wahrscheinlichkeiten für alle Pfade sind in der folgenden Tabelle 7.2 dargestellt.

Pfad	Wahrscheinlichkeit [%]
Schwarz	50
Rot	17
Grün	33

Tabelle 7.2: Wahrscheinlichkeiten für Maschinenkonstellationen

Nachdem nun die Wegentscheidung des AS- und MMAS-Algorithmus betrachtet worden ist, wird nun die Entscheidungsformel des ACS-Algorithmus auf das vorliegende Simulationsmodell angewendet. Hierbei wird eine pseudozufällig proportionale Regel verwendet, welche den nächsten Knoten j der künstlichen Ameise bestimmt. Diese lautet wie folgt (vgl. Kapitel 5.2):

$$j = \begin{cases} argmax_{l \in N_i^k} \{\tau_{il} \cdot [\eta_{il}]^\beta\} & falls\ q \leq q_0 \\ J & sonst \end{cases}$$

Im Gegensatz zur Formel 5.2 wird hier keine Übergangswahrscheinlichkeit berechnet. Abhängig von der Wahl von q_0, wird die Gewichtung der Pheromonmenge beeinflusst (vgl. Kapitel 5.2). Um die Entscheidungsformel zu verdeutlichen, wird diese an dem vorliegenden

Simulationsmodell angewendet. Hierbei wird für die Vergleichbarkeit ebenfalls für $\beta = 0$ gewählt. Für den Parameter q_0 wird der Wert 0,6 festgelegt. Dieser Wert ist willkürlich gewählt und kann von dem Anwender der Simulation variiert werden. Die Variable q wird zufällig generiert. Dabei kann sie Werte zwischen 0 und 1 annehmen.

$$j = \begin{cases} schwarzer\ Pfad & falls\ q \leq 0{,}6 \\ p_{ij}^{k} = \dfrac{[\tau_{ij}]^1 \cdot [\eta_{ij}]^0}{\sum_{u \in N_i^k} [\tau_{ij}]^1 \cdot [\eta_{ij}]^0} & sonst \end{cases} \tag{7.6}$$

Nimmt die Zufallszahl q einen Wert an, welcher kleiner als 0,6 ist, so wählt die künstliche Ameise nach Formel 7.6 den schwarzen Pfad aus der Abbildung 7.4. Dies hat den Grund, dass sich auf dem schwarzen Pfad die größte Pheromonmenge befindet. Der schwarze Pfad repräsentiert die in der Tabelle 7.3 dargestellte Maschinenkonstellation.

Arbeitsstation / GS	Anzahl Maschinen / Auswahl GS
WS1	3
WS2	3
WS3	3
WS4	3
WS5	3
GS	2

Tabelle 7.3: Maschinenkonstellation des schwarzen Pfades

Nimmt die Zufallszahl q einen Wert an, welcher größer als 0,6 ist, wird auf die Formel 7.6 Bezug genommen, durch welche die Erkundung des Suchraumes realisiert wird (vgl. Kapitel 5.2). In Abhängigkeit von q_0 existiert also ein Intervall, in dem keine Rechenvorgänge stattfinden. Befindet sich die Zufallsvariable q innerhalb dieses Intervalls, wird ausschließlich der Pfad mit der höchsten Pheromonmenge selektiert. Es werden statt Rechenoperationen lediglich Vergleiche der Pheromonmengen durchgeführt. Bei dem AS- und MMAS-Algorithmus finden in jedem Fall Rechenoperationen statt. Dies hat zur Folge, dass die Laufzeiten von AS- und MMAS-Algorithmen stets höher sind als die Laufzeiten des ACS-Algorithmus (vgl. Kapitel 5.2). Aus diesem Grund ist die Verwendung des ACS-Algorithmus in Bezug auf die Wegentscheidung vorteilhafter. Da die Nutzung von heuristischen Informationen für das vorliegende Produktionssystem nicht geeignet sind, wird die Formel zur Wahrscheinlichkeitsberechnung wie folgt geändert:

$$p_{ij}^{k} = \frac{\tau_{ij} + \tau_0}{\sum_{u \in N_i^k} \tau_{ij} + \tau_0} \tag{7.7}$$

Bei dem Wert τ_0 handelt es sich um eine Grundpheromonmenge, welche auf allen Kanten gleich groß ist. Diese wird verwendet, um Pfade zu berücksichtigen, welche von keinen Ameisen durchlaufen worden sind. Je größer die Grundpheromonmenge, desto geringer ist der Einfluss von durchlaufenen Pfaden. Um den Einfluss der Pheromonmenge auf die Wegentscheidung näher zu untersuchen, wird im späteren Verlauf dieser Arbeit der Parameter q_0 variiert. Nachdem das Kriterium der Wegentscheidung analysiert worden ist, wird als nächstes die Pheromonverdunstung untersucht.

Pheromonverdunstung

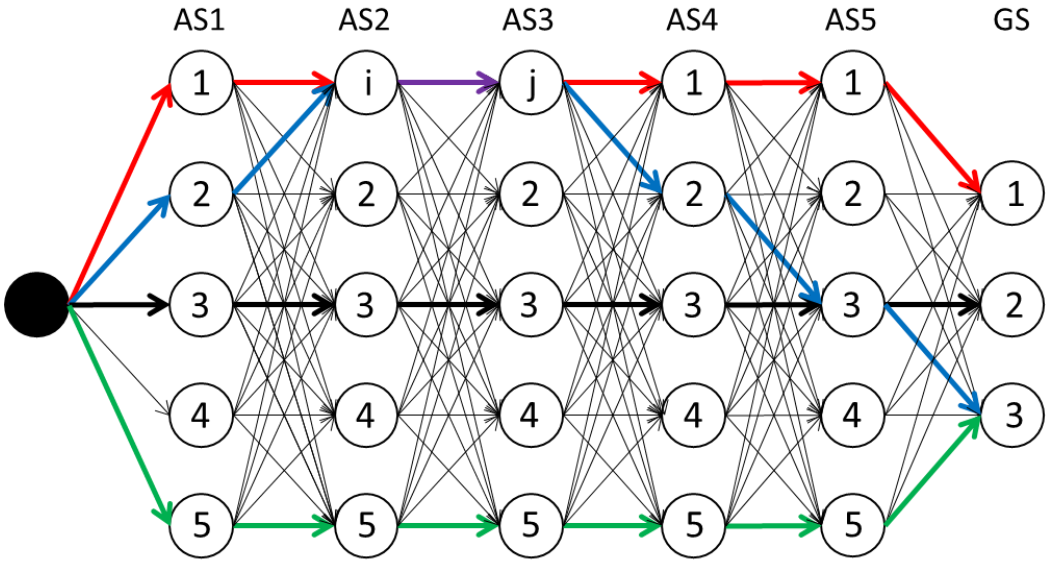
Die Pheromonverdunstung ist ein besonderer Aspekt der ACO-Metaheuristik. Die Berücksichtigung der Pheromonverdunstung bringt bei den künstlichen Ameisen einen entscheidenden Vorteil mit sich. Um eine schnelle Konvergenz an einem lokalen Optimum zu vermeiden, wird die hinterlegte Pheromonmenge nach jedem Iterationsschritt reduziert. Dies ermöglicht eine breite Suche innerhalb des Lösungsraums, wodurch die Wahrscheinlichkeit das globale Optimum zu finden erhöht wird. In dieser Bachelorarbeit wird die Pheromonverdunstung variiert. Dazu werden Läufe mit geringer Pheromonverdunstung und Läufe mit hoher Pheromonverdunstung betrachtet. Die Wahl einer hohen Pheromonverdunstung wird gewählt, um eine frühzeitige Stagnation des Fitnesswertes zu verhindern. Anschließend wird die Auswirkung auf den Fitnesswert analysiert. Das dritte zu untersuchende Kriterium, welches zur Wahl der ACO-Ausführung betrachtet wird, ist das Pheromonupdate.

Pheromonupdate

Das Pheromonupdate des AS-Algorithmus wird als erstes untersucht. Bei diesem Algorithmus werden die Pheromonspuren erst verändert, wenn alle Ameisen die Lösungsfindung beendet haben. Das Pheromonupdate erfolgt durch (vgl. Kapitel 5.2):

$$(1-\rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta {\tau_{ij}}^{k} \rightarrow \tau_{ij} \qquad \forall (i,j) \in L$$

Hierbei wird sowohl die Verstärkung als auch die Verdunstung der Pheromonspuren berücksichtigt, um die Pheromonintensität auf der jeweiligen Kante zu bestimmen. Bisher ist lediglich die erste Iteration betrachtet worden. Vor dieser Iteration sind keine Pheromonspuren auf den Kanten vorhanden. Um die Funktionsweise des Pheromonupdates zu verdeutlichen, wird deshalb ein bestimmter Pfad aus der zweiten Iteration hinzugezogen.



$\tau = 0{,}2$; $\tau = 0{,}4$; $\tau = 0{,}6$; $\tau = 0{,}3$;

Abbildung 7.5: Pheromonspuren nach der zweiten Iteration

Wie in der Abbildung 7.5 zu sehen, wird der Pfad von Knoten i zu Knoten j von unterschiedlichen künstlichen Ameisen durchlaufen. Die Kombination des roten und blauen Pfades ergibt den in der Abbildung 7.5 zu sehenden violetten Pfad. Die aktuelle Pheromonmenge τ_{ij} zwischen den beiden Knoten lässt sich nach dem Pheromonupdate wie folgt berechnen.

$$(1 - 0{,}6) \cdot 0{,}2 \; + 0{,}3 = \tau_{ij} = 0{,}38 \qquad (7.8)$$

0,6 ist dabei die frei gewählte Phermonverdunstung. Bei den restlichen Knoten verläuft die Aktualisierung der Pheromonmenge analog.

Im Vergleich zum AS-Algorithmus unterscheidet sich das Pheromonupdate bei dem MMAS-Algorithmus. Die Intensivierung der Pheromonmenge wird lediglich auf dem besten Pfad durchgeführt (vgl. Kapitel 5.2). Dies bedeutet, dass der Pfad mit dem besten Fitnesswert eine Pheromonzunahme erhält, wobei die restlichen Pfade ungeändert bleiben. Die Pheromonverdunstung hingegen bleibt bei allen Pfaden erhalten.

$$(1 - \rho) \cdot \tau_{ij} + \; \Delta\tau_{ij}{}^{best} \rightarrow \tau_{ij} \qquad \forall (i,j) \in L$$

Angewendet auf die Knoten i und j aus der Abbildung 7.5 ergibt sich folgende Rechnung:

$$(1 - 0{,}6) \cdot 0{,}2 + \; 0 = \tau_{ij} = 0{,}08 \qquad (7.9)$$

Da es sich bei diesem Pfad nicht um die beste Lösung mit dem höchsten Fitnesswert handelt, findet auch keine Pheromonzunahme statt. Dies gilt ebenfalls für die restlichen Pfade. Einzig beim schwarzen Pfad ist eine Pheromonintensivierung möglich.

Beim ACS-Algorithmus gibt es zwei unterschiedliche Pheromonupdates. Zum einen das globale Pheromonupdate, zum anderen das lokale Pheromonupdate. Für den weiteren Verlauf wird lediglich das globale Pheromonupdate betrachtet. Dieses berechnet sich durch die Formel 5.7:

$$(1-\rho)\cdot\tau_{ij} + \varrho\cdot\Delta\tau_{ij}{}^{best} \rightarrow \tau_{ij} \qquad \forall(i,j)\in s^{best}$$

Auch hier findet die Erhöhung der Pheromonkonzentration lediglich für den Pfad mit dem besten Fitnesswert statt. Der Unterschied zum Pheromonupdate des MMAS-Algorithmus ist, dass ein Teil der Pheromonmenge des besten Pfades unmittelbar verdunstet. Es wird somit ein gewichteter Durchschnitt von der neu abgelegten und der alten Pheromonmenge berechnet.

Unter Berücksichtigung des Kriteriums der Wegentscheidung überwiegen die Vorteile des ACS-Algorithmus. Vor allem die geringen Laufzeiten des ACS sind ein entscheidender Faktor bei der Wahl zwischen den drei unterschiedlichen Ameisenalgorithmen (vgl. Kapitel 5.2). Für den Aspekt des Pheromonupdates wird jedoch das Vorgehen des AS-Algorithmus umgesetzt, um Pheromone auf allen Wege intensivieren zu können. So wird versucht eine frühzeitige Stagnation auszuschließen (vgl. Kapitel 5.2).

7.1.2 Pseudo-Code

1: **Name:** ACS-Algorithmus zur Variation des Produktionssystems
2: **Eingabe:** Anzahl Ameisen, Anzahl Iterationen, Grundpheromon τ_0, q_0 , Verdunstungsrate
3:
4: **begin**
5: *Erzeuge Baum;*
6: *Wähle zufällige Pfade im Baum;*
7: Pheromonmenge $\tau(i) = \frac{1}{Fitnesswert(i)}$; // Kanten des Baums mit Pheromon gewichten
8:
9: Neue Pheromonmenge auf Kante i : $\tau(i) = \tau_0 + \tau_{dazu}\,(i)$;
10:
11: **For** g = 2 **to** AnzIterationen **do**
12:
13: **begin**
14: *Führe für alle Ameisen gleichzeitig aus:*
15:
16: For k = 0 to k = 5 **do**
17: **begin**
18: **If** k < 5 **then**
19: AnzNachbarKanten = 5 // 5 Arbeitsstationen
20: **Else**
21: AnzNachbarKanten = 3 // 3 Gabelstapler
22:
23: *Generiere q*
24: **If** q < q0 **then**
25: *Wähle Kante mit höchster Pheromongewichtung*
26: **else**

27: **begin**
28: $p(i) = \frac{\tau_{neu}(i)}{\sum \tau_{Nachbarschaft}}$ // Wahscheinlichkeit einer Kante
29: *Definiere Intervalle abhängig von den berechneten Wahrscheinlichkeiten*
30: *generiere Zufallszahl;*
31: *→ ermittle in welchem Intervall die Zufallszahl liegt;*
32: *→ wähle diese Kante;*
33: **end**
34:
35: *Ermittle Fitnesswert des gelaufenen Pfades und bestimme dazugekommene*
36: *Pheromonmenge*
37: *Aktualisiere Pheromonmenge für jede Kante i:* $\tau(i) = \tau_{dazu} + \tau\,(i)$*;*
38:
39: **end**
40: *Trage Pfade mit Fitnesswerten in Liste ein ;*
41: *Verdunste Pheromon:* $\tau(i) = \tau(i) \cdot \rho$*;*
42**:**
43: **end**

Bei der Eingabe im Programm muss der Anwender fünf Parameter festlegen. Er muss entscheiden, wieviele Ameisen die Pfade gleichzeitig ablaufen und ihre Pheromone versprühen. Des Weiteren muss die Anzahl der Iterationen festgelegt werden. Um diese Anzahl festzulegen, muss der Anwender der Simulation wissen, wie genau seine Lösung sein muss und wieviel Zeit er für eine Simulation einplant. Der dritte Parameter, welcher festgelegt werden muss, ist die Grundpheromonmenge τ_0 . Mit der Festlegung dieser Variable entscheidet der Anwender, mit welcher Wahrscheinlichkeit die Ameisen Wege laufen sollen, welche von keiner Ameise zuvor gelaufen worden sind (vgl. Kapitel 5.2). Um diese Abhängigkeit der Wegentscheidung von der Pheromonmenge genauer einstellen zu können, muss der Anwender einen Wert für q_0 festlegen. Dabei handelt es sich bei q_0 um eine spezifische Variable des ACS-Algorithmus. Die letzte Entscheidung, welche der Anwender vor der Ausführung des Programms treffen muss, ist der Wert der Verdunstungsrate. Dieser Parameter bietet dem Anwender die Möglichkeit festzulegen, wie intensiv die Informationen von Vorgängern genutzt werden sollen (vgl. Kapitel 5.2). Nachdem diese Parameter festgelegt worden sind, hat das Programm alle notwendigen Informationen, um die Simulation zu beginnen. Der erste Schritt nach der Eingabe der Parameter ist das Erzeugen des Baums. Bei dem Baum handelt es sich um ein Konstrukt, welches aus Kanten und Knoten besteht (vgl. Abbildung 7.3). Der Baum ist in Bezug auf die Variation des modularen Produktionssystems statisch. Es wird also bei jeder Simulation unabhängig von der Eingabe des Benutzers derselbe Baum erzeugt. Die Anzahl der Kanten des Baums beträgt

$$AnzKanten = 5 + 4 * 5 * 5 + 3 * 3 = 114 \qquad (7.10)$$

Die Kanten des Baums können gewichtet werden. Die Gewichtung repräsentiert in dem vorliegenden Simulationsmodell die Pheromonmenge. Die Gewichtung der Kanten ist nach der Erzeugung des Baums zunächst für jede Kante identisch (vgl. Kapitel 5.1). Dabei nimmt

diese den Wert der Grundpheromonmenge τ_0 an. Aufgrund der identischen Gewichtung von allen Kanten wählt die erste Generation der Ameisen zufällige Pfade. Nachdem alle Ameisen die Pfade abgelaufen sind, werden die Fitnesswerte der einzelnen Pfade bestimmt. Aus den Fitnesswerten berechnet das Programm die Pheromonmenge, welche auf den jeweiligen Pfaden versprüht werden soll. Diese dazugekommene Pheromonmenge τ_{dazu} wird auf die Grundpheromonmenge τ_0 addiert. Die Gewichtung der Kanten in dem Baum ist nun nicht mehr gleichverteilt. Ab der zweiten Generation bis zur letzten Generation der Ameisen wird die Wegentscheidung des ACS-Algoritmus verwendet. Die Anzahl der Generationen von Ameisen wird in dem vorliegenden Simulationsmodel durch die Anzahl der Iterationsschritte nachgebildet. Aus diesem Grund läuft die FOR-Schleife von g=2 bis zur Gesamtanzahl an Iterationsschritten. Damit eine Ameise einen vollständigen Pfad abläuft, muss diese sechs Wegentscheidungen treffen, weshalb das Programm eine FOR-Schleife von k=0 bis k=5 laufen lässt. Dabei stehen der Ameise bei den ersten fünf Entscheidungen jeweils fünf Entscheidungsmöglichkeiten zur Verfügung. Dies hat den Grund, dass sich die ersten fünf Entscheidungen auf die Maschinenanzahl der jeweiligen Arbeitsstation beziehen. Die sechste und letzte Entscheidung der Ameise bezieht sich auf die Anzahl der verwendeten Gabelstapler. Vor jeder Entscheidung generiert das Programm zufällig eine Zahl *q*. Anschließend wird eine Fallunterscheidung druchgeführt. Tritt der Fall ein, dass der zufällig generierte Wert für *q* kleiner ist als der Wert von q_0, so wählt die Ameise die Kante, welche die größte Gewichtung besitzt. Tritt der Fall ein, dass der generierte Wert für *q* größer ist, als der Wert von q_0, so entscheidet sich die Ameise durch eine Wahrscheinlichkeitsverteilung für eine Kante. Dazu berechnet das Programm die Wahrscheinlichkeit jeder Kante in Abhängigkeit der Pheromonmenge, indem die Pheromonmenge einer Kante in Verhältnis zu allen Kanten der erreichbaren Nachbarschaft gesetzt wird (vgl. Kapitel 5.2). Diese Wahrscheinlichkeiten werden vom Programm in Intervalle umgewandelt. Die Summe dieser Intervalle ergibt den Wert „1“. Anschließend wird eine weitere Zufallszahl zwischen „0“ und „1“ generiert. Je nachdem, in welchem der berechneten Intervalle diese Zufallszahl liegt, wird die dazugehörige Kante gewählt. Dieser Vorgang wird innerhalb der inneren FOR-Schleife wiederholt, bis jede Ameise sechs Wegentscheidungen getroffen hat. Nachdem die FOR-Schleife beendet worden ist, ist jede Ameise einen vollständigen Pfad abgelaufen. Jede Ameise generiert einen Fitnesswert, aus welchem sich die dazugekommene Pheromonmenge berechnet. Diese dazugekommene Pheromonmenge wird auf die bisherigen Pheromonmengen der Kanten des jeweiligen Pfades addiert. Diese Pheromonmengen werden zum Ende der äußeren FOR-Schleife verdunstet. Dies geschieht, indem die Gewichtungen der Kanten mit der Verdunstungsrate multipliziert werden. Die gelaufenen Pfade mit den Fitnesswerten werden von dem Programm in eine Liste geschrieben. An dieser Stelle ist in der Simulation die zweite Iteration beendet worden. Die Simulation ist beendet, sobald die äußere FOR-Schleife so oft durchgelaufen ist, dass die vom

Anwender festgelegte Anzahl an Iterationen erreicht worden ist. Die Ausgabe des Simulationsprogramms ist eine Liste, in welcher die Maschinenkonstellationen mit den entsprechenden Fitnesswerten ausgegeben werden. Auf Basis dieser Ausgabe erfolgt die Auswertung die Simulationsergebnisse.

7.2 Ant-Colony-Optimierung zum Scheduling von Auftragsreihenfolgen

Analog zur Nutzung der Ant-Colony-Optimierung zur Variation des modularen Produktionssystems erfolgt auch das Scheduling der Auftragsreihenfolgen innerhalb einer Black-Box (vgl. Kapitel 3.3). Es wird eine Auftragsliste generiert. Der Inhalt der Auftragsliste ändert sich nicht. Es ändert sich lediglich die Reihenfolge der Aufträge innerhalb der Liste. In dieser Bachelorarbeit enthält eine Auftragsliste 1.200 Aufträge. Aus der hohen Anzahl der Aufträge resultiert eine immens hohe Anzahl an potentiellen Pfaden, welche den künstlichen Ameisen zur Verfügung stehen. Eine solch hohe Anzahl an potentiellen Pfaden ist selbst für einen leistungsstarken Computer nicht realisierbar. Aus diesem Grund wird für den weiteren Verlauf eine Reduktion der Auftragsliste vorgenommen. Die Vorgehensweise der Lösungsfindung ändert sich dabei nicht. Die Auftragsliste wird auf neun Aufträge reduziert. In der folgenden Abbildung 7.6 ist eine schematische Darstellung der möglichen Auftragsreihenfolgen einer Liste abgebildet.

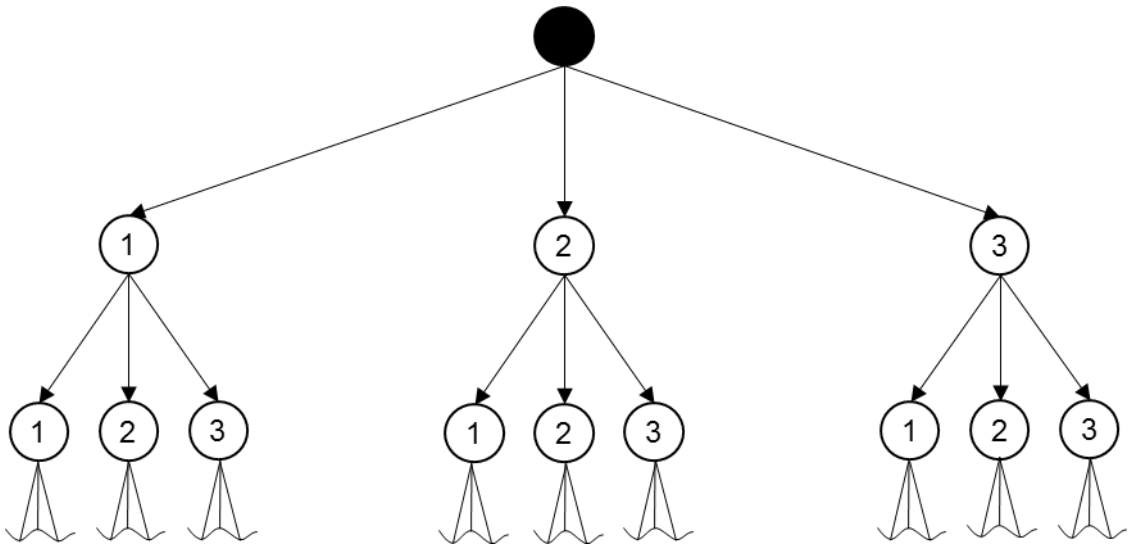


Abbildung 7.6: Mögliche Auftragsreihenfolgen

Der Aufbau der Abbildung 7.6 erfolgt nach einem ähnlichen Prinzip wie in Kapitel 7.1. Auch hier wird ein Baum erzeugt. Der Unterschied ist hierbei jedoch, dass sich die Auswahlmöglichkeiten nach jeder Station um eine Auswahlmöglichkeit verringert. Die Gesamtanzahl der Lösungsmöglichkeiten umfasst dabei:

$$A = n! \tag{7.11}$$

Der Parameter n ist als die Anzahl der Einträge in der Auftragsliste definiert. In dem vorliegenden Modell beträgt die Gesamtanzahl der Lösungsmöglichkeiten

$$A = 9! = 362.880 \tag{7.12}$$

Zum Vergleich beträgt die Gesamtanzahl der Lösungsmöglichkeiten an der Auftragsliste mit 1.200 Einträgen

$$A = 1200! = 6{,}35 \cdot 10^{3175} \tag{7.13}$$

Dies ist ein geeignetes Beispiel, um zu verdeutlichen aus welchem Grund Metaheuristiken angewendet werden. Der Rechenaufwand der Simulation ist bei einer solch hohen Anzahl an Lösungsmöglichkeiten extrem hoch.

7.2.1 Auswahl der Ant-Colony-Optimierung-Ausführung

Analog zur Variation des modularen Produktionssystems wird eine geeignete Ausführungsmöglichkeit der Ant-Colony-Optimierung gesucht. Dabei werden ebenfalls die Kriterien Wegentscheidung, Pheromonverdunstung und Pheromonupdate untersucht.

Wegentscheidung

Analog zu Kapitel 7.1.1 werden auch hier die ACO-Algorithmen AS, MMAS und ACS miteinander verglichen. Betrachtet wird die folgende Abbildung 7.7.

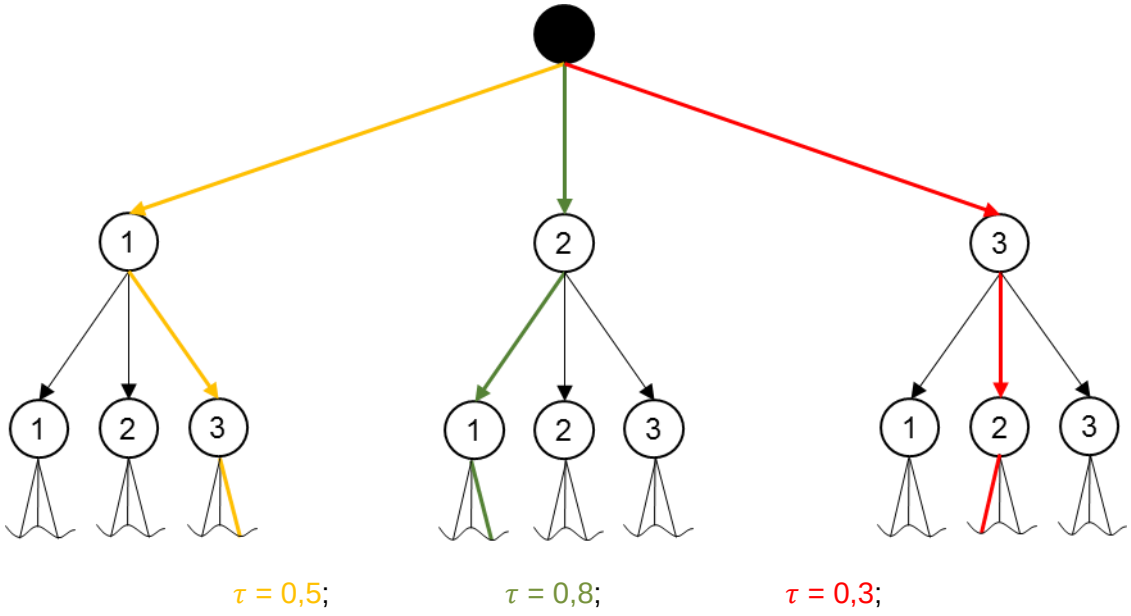


Abbildung 7.7: Pheromonspuren nach der ersten Iteration

Es ist zu berücksichtigen, dass die Maschinenkonstellation in diesem Fall festgelegt ist und nicht variiert wird. Die Aufgabe besteht darin, für diese Maschinenkonstellation die Auftragsreihenfolge zu finden, welche den besten Fitnesswert liefert. Dabei ist in der Abbildung 7.7 beispielhaft eine Auftragsliste mit drei Einträgen zu sehen. Jeder der drei Jobtypen, welche

in dem Simulationsmodel auftreten können, ist dabei in der Auftragsliste vorhanden. In der Abbildung 7.7 sind die Ergebnisse nach der ersten Iteration dargestellt. Die Wahrscheinlichkeit, dass die künstlichen Ameisen einen bestimmten Pfad ablaufen ist zunächst für jeden Pfad gleich. Erst nachdem die Pfade belaufen worden sind, ändert sich die Wahrscheinlichkeitsverteilung für die Pfade. Diese ist von der Pheromonmenge beziehungsweise vom Fitnesswert der einzelnen Pfade abhängig (vgl. Kapitel 5.2). Die Pheromonmengen sind ebenfalls in der Abbildung 7.7 abgebildet. Ein grundlegender Unterschied zur Variation der Produktionssysteme ist, dass die Anzahl der Nachbarschaften der Knoten geringer ist. Es ist abhängig davon, wie nah die künstliche Ameise an der Futterquelle ist.

Beim AS- und MMAS-Algorithmus wird die Übergangswahrscheinlichkeit durch die Formel 5.2 berechnet. Auch hier wird die Annahme getroffen, dass $\propto = 1$ und $\beta = 0$ sind. Für den roten Pfad ergibt sich somit

$$p_{ij}^{k} = \frac{[\tau_{ij}]^1 \cdot [\eta_{ij}]^0}{\sum_{u \in N_i^k} [\tau_{ij}]^1 \cdot [\eta_{ij}]^0} = \frac{[0{,}3]^1}{0{,}5 + 0{,}8 + 0{,}3} = 0{,}1875 \qquad (7.14)$$

Die Wahrscheinlichkeit, dass eine künstliche Ameise bei der zweiten Iteration den roten Pfad wählt, beträgt also 18,75%. Die Übergangswahrscheinlichkeiten der übrigen Pfade berechnen sich analog. Diese sind in der folgenden Tabelle 7.4 zusammengeführt.

Pfad	Wahrscheinlichkeit [%]
Gelb	31,25
Grün	50
Rot	18,75

Tabelle 7.4: Wahrscheinlichkeiten der Auftragsreihenfolgen

Der grüne Pfad liefert also den besten Fitnesswert. Dieser Pfad beschreibt in dem vorliegenden Simulationsmodel die Auftragsreihenfolge:

1. Job-Typ 2
2. Job-Typ 1
3. Job-Typ 3
4. …

Bei dem ACS-Algorithmus erfolgt die Wegentscheidung durch die Formel 5.5. Wird $q_0 = 0{,}6$ festgelegt, so entscheiden sich die künstlichen Ameisen für den besten Pfad, falls $q \leq 0{,}6$ generiert wird. Der beste Pfad in Abbildung 7.7 ist der grüne Pfad. Wird $q > 0{,}6$ generiert, so werden die Übergangswahrscheinlichkeiten für die einzelnen Pfade in Abhängigkeit der Pheromonmenge berechnet.

Pheromonverdunstung

Auch beim Scheduling von Auftragsreihenfolgen werden unterschiedliche Werte für die Pheromonverdunstung gewählt. In diesem Fall ist es möglich, dass die Wahl einer hohen oder niedrigen Pheromonverdunstung eine andere Auswirkung auf die Untersuchung des Lösungsraums aufweist. Daher ist die Berücksichtigung der Pheromonverdunstung essentiell. Durch die Wahl der niedriger Pheromonverdunstungen wird eine schnelle Festlegung einer potentiellen Lösung angestrebt, da dies schon nach kurzer Laufzeit zur Stagnation der Fitnesswerte führt (vgl. Kapitel 5.2). Zum anderen wird eine hohe Pheromonverdunstung gewählt, um den anderen Fall zu untersuchen.

Pheromonupdate

Das Pheromonupdate des AS-Algorithmus ist bereits in dem Kapitel 7.1.1 erläutert worden. Um diesen Algorithmus am Beispiel für das Scheduling von Auftragsreihenfolgen anzuwenden, wird die folgende Abbildung 7.8 betrachtet.

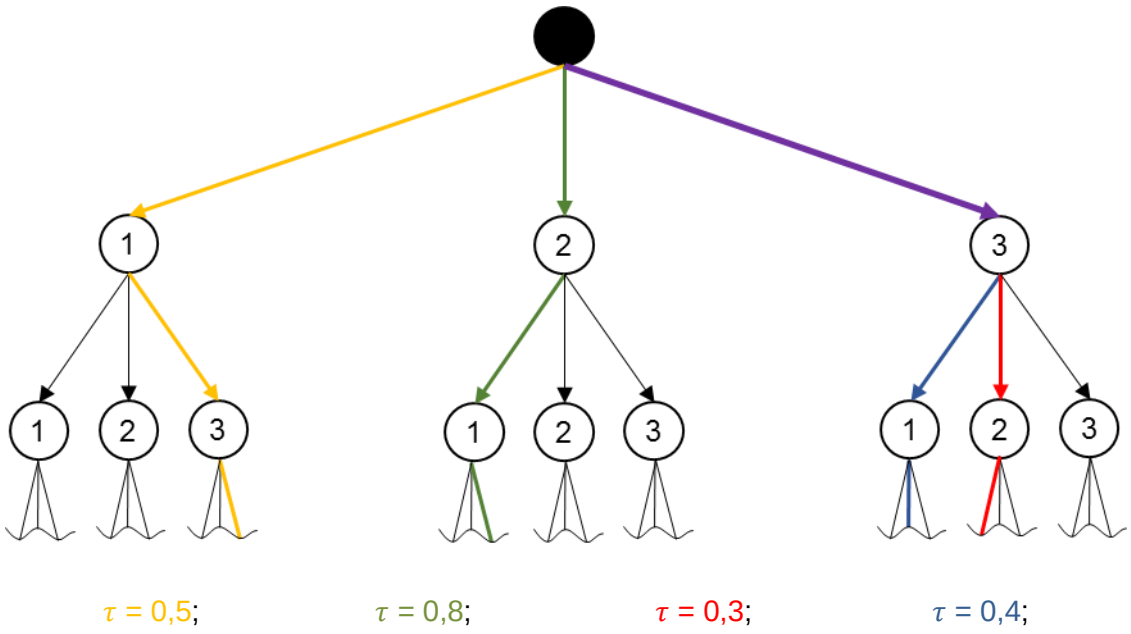


Abbildung 7.8: Pheromonspuren nach der zweiten Iteration

In der Abbildung 7.8 ist ein Pfad aus der zweiten Iteration hinzugekommen. Dabei überschneidet sich dieser vom Root zum Knoten „3“ mit dem roten Pfad. Daraus resultiert das Pheromonupdate mit einer Pheromonverdunstung von 0,6 nach dem AS-Algorithmus wie folgt:

$$(1-\rho)\cdot\tau_{ij}+\sum_{k=1}^{m}\Delta\tau_{ij}{}^{k}\rightarrow\tau_{ij}=\ (1-0{,}6)\cdot 0{,}3+0{,}4=0{,}52 \qquad (7.15)$$

Nach Formel 5.7 ergeben sich ebenfalls die restlichen Pheromonmengen.

Bei dem MMAS- und ACS-Algorithmus findet die Pheromonintensivierung beim Scheduling von Auftragsreihenfolgen nur für den grünen Pfad statt. Dies hat den Grund, dass wie bereits in Kapitel 7.1.1 erwähnt, die Intensivierung nur auf dem Pfad mit dem besten Fitnesswert erfolgt. Die Berechnung der Pheromonmenge nach dem Pheromonupdate erfolgt dabei analog zu den Berechnungen aus dem Kapitel 7.1.1.

Da es sowohl bei der Variation der Produktionssysteme als auch beim Scheduling von Auftragsreihenfolgen keine bemerkenswerten Unterschiede gibt, wird auch hier der AS-Algorithmus für das Pheromonupdate bevorzugt. Daher wird im weiteren Verlauf dieser Bachelorarbeit mit denselben Formeln wie in Kapitel 7.1 gerechnet.

7.2.2 Pseudo-Code

Name: ACS-Algorithmus für das Scheduling von Auftragsreihenfolgen
Eingabe: Anzahl Jobtypen, Anzahl Ameisen, Anzahl Iterationen, Grundpheromon τ_0, q_0, Verdunstungsrate
begin
Erzeuge Baum;
Wähle zufällige Pfade im Baum;
Pheromonmenge $\tau(i)=\frac{1}{Fitnesswert(i)}$; // Kanten des Baums mit Pheromon gewichten

Neue Pheromonmenge auf Kante i : $\tau(i)=\tau_0+\tau_{dazu}\,(i)$;

For g = 2 **to** AnzIterationen **do**

begin
Führe für alle Ameisen gleichzeitig aus:

Prüfe Anzahl Nachbarschaften
Generiere q
    If q < q0 **then**
        Wähle Kante mit höchster Pheromongewichtung
    else
        begin
        $p(i)=\frac{\tau_{neu}(i)}{\sum\tau_{Nachbarschaft}}$ // Wahscheinlichkeit einer Kante
        Definiere Intervalle abhängig von den berechneten Wahrscheinlichkeiten
        generiere Zufallszahl;
        → ermittle in welchem Intervall die Zufallszahl liegt;
        → wähle diese Kante;
        end

Ermittle Fitnesswert des gelaufenen Pfades und bestimme dazugekommene

30: *Pheromonmenge;*
31: *Aktualisiere Pheromonmenge für jede Kante i:* $\tau(i) = \tau_{dazu} + \tau\,(i)$*;*
32:
33: **end**
34: *Trage Pfade mit Fitnesswerten in Liste ein ;*
35: *Verdunste Pheromon:* $\tau(i) = \tau(i) \cdot \rho$*;*
36:
37: **end**

Analog zum Vorgehen des Algorithmus für die Maschinenkonstellation, erfordert auch der Algorithmus für das Scheduling von Auftragsreihenfolgen zunächst die Eingabe der Parameter. Bei diesen Parametern handelt es sich um die

1. Anzahl Jobtypen
2. Anzahl Ameisen
3. Anzahl Iterationen
4. Grundpheromon τ_0
5. q_0-Wert ρ
6. Verdunstungsrate

Es ist also festzustellen, dass die Eingabe des Algorithmus um den Parameter „Anzahl Jobtypen“ erweitert worden ist. Mit Hilfe dieses Parameters wird dem Anwender der Simulation die Möglichkeit gegeben explizite Auftragslisten zu optimieren. Hat der Anwender die Parameter dem Simulationsprogramm übermittelt, so wird als nächstes der Baum für die eingegebene Auftragsliste erzeugt. Im Gegensatz zur Funktionsweise des Simulationsmodells für die Maschinenkonstellation, wird in diesem Fall ein dynamischer Baum benötigt. Die Anzahl der Kanten, Knoten und erreichbaren Nachbarschaften hängt von der eingegebenen Auftragsliste ab und ist nicht für jeden Fall identisch. Aus diesem Grund benötigt das Simulationsprogramm mehr Zeit für das Aufbauen des Baums. Die Anzahl der Kanten in diesem Baum umfasst maximal:

$$AnzKanten = \sum_{i=1}^{7} 3^i + 3^7 \cdot 4 = 12.027 \qquad (7.16)$$

Jede dieser Kanten kann gewichtet werden. Diese Gewichtung wird für das Versprühen von Pheromonen genutzt. Die Grundpheromonmenge τ_0 wird nach der Erzeugung des Baums gleichverteilt auf dem Baum hinterlegt. Hierdurch wird erzielt, dass die Ameisen ihre Wegentscheidung im ersten Durchlauf basierend auf einer Gleichverteilung treffen. Nach dem Ablaufen der ersten Iteration wird die Gewichtung der Kanten manipuliert. Dies geschieht in Abhängigkeit der generierten Fitnesswerte. So ist die dazugekommene Pheromonmenge einer einzelnen Kante als Kehrwert des Fitnesswertes definiert. Diese Pheromonmenge wird mit der bestehenden Pheromonmenge addiert. Aus dieser Aktualisierung der Pheromon-

menge resultiert, dass keine Gleichverteilung mehr vorliegt. Von nun an wird der AS-Algorithmus zur Wegentscheidung verwendet. Die Aspekte der Wegentscheidung, des Pheromonupdates und der Pheromonverdunstung erfolgen analog zum Vorgehen des Pseudo-Codes für die Maschinenkonstellation.

8 Auswertung der Simulation

In diesem Kapitel wird das vorliegende Simulationsmodell nach der Implementierung des ausgewählten Algorithmus betrachtet. Wichtig ist, dass sich die Implementierung für die Variation der Produktionssysteme von der Implementierung des Scheduling von Auftragsreihenfolgen unterscheidet. Aus diesem Grund werden beide Teilprobleme unabhängig voneinander ausgewertet. Ziel ist es, eine optimale Maschinenkonstellation und eine optimale Auftragsreihenfolge für das Produktionssystem zu ermitteln. Es werden dabei unterschiedliche Parameter variiert und die Auswirkung auf das Simulationsmodell untersucht.

8.1 Bestimmung der Maschinenkonstellation

Bei der Findung einer optimalen Maschinenkonstellation werden Läufe mit unterschiedlichen Parametereinstellungen simuliert. Es wird versucht die spezifischen Parameter des ausgewählten Algorithmus möglichst gut auf das Simulationsmodell einzustellen. Bei diesen Parametern handelt es sich um die Verdunstungsrate und um die Wahl des qo-Wertes. Außerdem wird die Anzahl der Iterationsschritte und die Anzahl der Läufe pro Iterationsschritt variiert. Die Anzahl der Läufe pro Iterationsschritt beschreibt dabei die Anzahl der künstlichen Ameisen.

Zunächst wird der Einfluss der Verdunstungsrate ρ verdeutlicht. Dabei werden bei der Variation der Verdunstungsraten die restlichen Parameter konstant gehalten. Es werden Läufe mit Verdunstungsraten $\rho = 10\%$; $\rho = 30\%$ und $\rho = 60\%$ getestet. Der qo-Wert wird dabei auf 0,5 festgelegt. Außerdem finden zehn Läufe pro Iterationsschritt statt. Im ersten Iterationsschritt erfolgt die Wegentscheidung der künstlichen Ameisen gleichverteilt. Diese Konfiguration wird zum einen mit einer Anzahl von zehn Iterationsschritten und zum anderen mit 50 Iterationsschritten ausgeführt. In diesem Fall handelt es sich um voneinander unabhängige Läufe. Als Ergebnis der Simulation werden die Fitnesswerte der jeweiligen Konfiguration ausgegeben. In der folgenden Tabelle 8.1 sind die Ergebnisse der Simulationsexperimente abgebildet.

Anfangsverteilung zufällig; $qo = 0{,}5$; 10 Ameisen	$Verdunstungsrate\ \rho\ [\%]$		
	10		
10 Iterationen	8438,1	6751,9	7366,2
50 Iterationen	121,1	87,0	137,4
Anfangsverteilung zufällig; $qo = 0{,}5$; 10 Ameisen	$Verdunstungsrate\ \rho\ [\%]$		
	30		
10 Iterationen	8983,0	5534,1	6819,9
50 Iterationen	11,1	86,4	35,8
Anfangsverteilung zufällig; $qo = 0{,}5$; 10 Ameisen	$Verdunstungsrate\ \rho\ [\%]$		
	60		
10 Iterationen	9098,0	7293,4	8593,8
50 Iterationen	422,7	339,2	583,6

Tabelle 8.1: Fitnesswerte verschiedener Verdunstungsraten

Um statistische Unsicherheiten ausschließen zu können, wird jede Konfiguration drei Mal simuliert. Aus diesen drei Simulationsergebnissen wird anschließend der Durchschnitt berechnet. Die durchschnittlichen Fitnesswerte sind in der folgenden Tabelle 8.2 zusammengefasst.

Anfangsverteilung random; $qo = 0{,}5$; 10 Ameisen	$Verdunstungsrate\ \rho\ [\%]$		
	10	30	60
10 Iterationen	4525,3	5112,38	6328,45
100 Iterationen	3369,59	1697,28	5355,48

Tabelle 8.2: Durchschnittliche Fitnesswerte verschiedener Verdunstungsraten

Die erste Generation der künstlichen Ameisen läuft die möglichen Pfade zufällig ab. Diese Ameisen hinterlegen Pheromonspuren. Dadurch verläuft die Wegentscheidung der künstlichen Ameisen ab der zweiten Generation in Abhängigkeit der Pheromonmenge. Zunächst ist in der Tabelle 8.2 deutlich zu erkennen, dass die Fitnesswerte nach 50 Iterationen

stets besser sind als die Fitnesswerte nach zehn Iterationen. Dies ist ein Indikator dafür, dass das Optimierungsverfahren funktioniert. Dabei ist für jeden Lauf eine zufällige Startkonstellation der Maschinen gewählt worden. Der Einfluss der zufälligen Startkonstellation wird in Tabelle 8.2 ersichtlich. So weisen die Fitnesswerte trotz identischer Konfiguration eine hohe Streuung auf. Werden die Simulationsergebnisse der Läufe mit einer Verdunstungsrate von 10% betrachtet, so werden mehrere Auffälligkeiten festgestellt. Die geringe Verdunstungsrate von 10% lässt eine breite Durchsuchung des Lösungsraumes nicht zu. Dies bedeutet, dass nach der Findung eines lokalen Optimums die Entwicklung der Fitnesswerte nicht stark vom gefundenen lokalen Optimum abweichen (vgl. Kapitel 5.2). Vergleichsweise zu den Pheromonverdunstungen von 10% und 60%, liefert die Simulation mit einer Pheromonverdunstung von 30% die besten Fitnesswerte. Die höheren Fitnesswerte bei der Wahl der Pheromonverdunstung von 60% begründen sich durch die geringe Abhängigkeit von der Pheromonmenge. Die hohe Pheromonverdunstung von 60% ermöglicht zwar eine breite Suche des Lösungsraums, jedoch löst diese sich zu schnell von lokalen Optima. Daher ist die Wahl der Pheromonverdunstung von 30% am geeignetsten. Dies hat den Grund, dass sowohl eine breite Suche des Lösungsraums als auch die Nutzung der Informationen der vorherigen Ameisen ermöglicht wird. Um die Güte der Optimierung zu untersuchen, werden Läufe einzeln betrachtet. Dazu wird die Verbesserung des Fitnesswertes in Abhängigkeit der Anzahl der Iterationsschritte in der folgenden Abbildung 8.1 dargestellt.

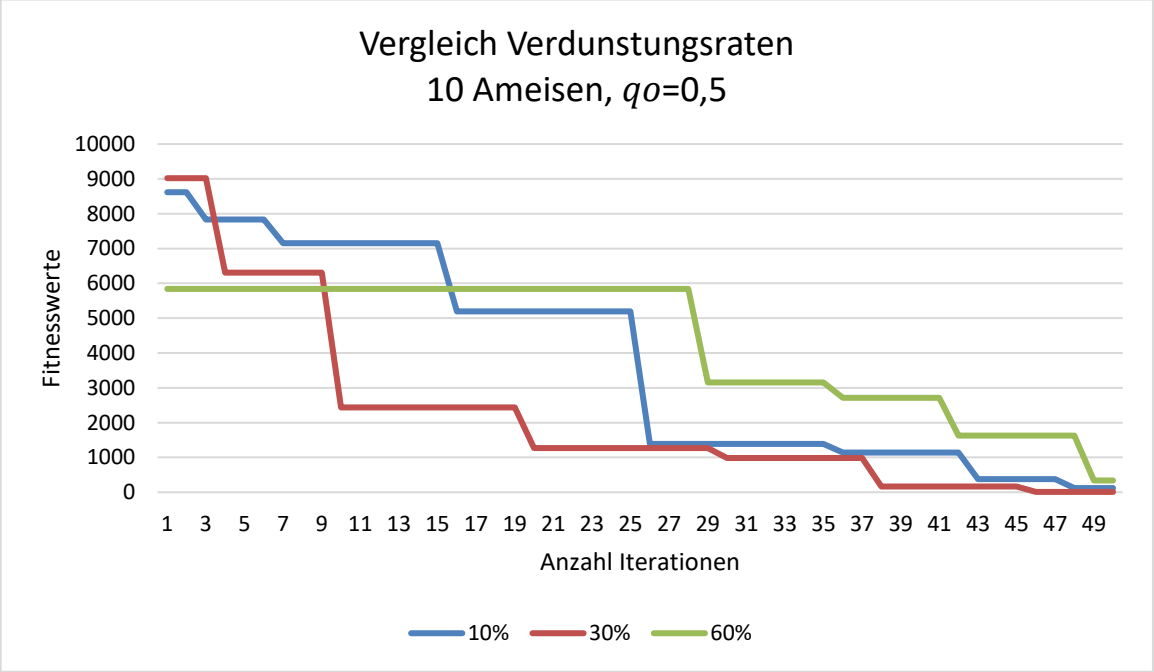


Abbildung 8.1: Vergleich Verdunstungsraten

Es sind die besten Läufe aus der Tabelle 8.1 für die jeweilige Verdunstungsrate abgebildet. Nach jeder Iteration wird geprüft, ob ein besserer Fitnesswert als in der vorherigen Iteration

erzielt worden ist. So wird lediglich der beste gefundene Fitnesswert bis zur jeweiligen Iteration abgebildet. Zunächst ist an dem Trend der Graphen deutlich zu sehen, dass das Optimierungsverfahren funktioniert. Bei der Verdunstungsrate von 60% werden die Maschinenkonstellationen stark zufällig generiert. Auf der einen Seite besteht die Möglichkeit, dass durch den Zufall ein sehr guter Fitnesswert gefunden wird. Auf der anderen Seite tritt am häufigsten der Fall ein, dass die Fitnesswerte schlecht sind. Dies liegt daran, dass es mehr Maschinenkonstellationen mit schlechten Fitnesswerten als mit guten Fitnesswerten gibt. Bei dem Vergleich der Verdunstungsraten von 10% und 30% fällt auf, dass beide Verdunstungsraten nach 50 Iterationen gute Fitnesswerte liefern. Durch die breitere Suche des Lösungsraums bei der Verdunstungsrate von 30%, werden weniger Iterationsschritte zur Findung von guten Fitnesswerten benötigt als bei einer Verdunstungsrate von 10%.

Nun wird die Eingabe des qo-Wertes variiert, um so den Einfluss des Parameters auf die Fitnesswerte untersuchen zu können. In der folgenden Tabelle 8.3 sind die durchschnittlichen Fitnesswerte für drei unterschiedliche qo-Werte abgebildet.

Anfangsverteilung random; 10 Ameisen;	qo-Wert		
	0,2	0,5	0,8
10 Iterationen	5661,6	5435,2	9414,1
50 Iterationen	182,5	16,4	901,7

Tabelle 8.3: Durchschnittliche Fitnesswerte verschiedener qo-Werte

Auch hier ist aus der Tabelle 8.3 zu entnehmen, dass die Fitnesswerte sich mit einer zunehmenden Anzahl an Iterationen verbessern. Indem ein qo-Wert von 0,2 gewählt wird, wird nur selten auf den besten Fitnesswert zugegriffen und die meisten Wegentscheidungen erfolgen durch eine Wahrscheinlichkeitsverteilung auf jedem Pfad. Wird ein qo-Wert von 0,8 gewählt, sind die nachfolgenden Maschinenkonstellationen sehr stark abhängig von der Maschinenkonstellation mit dem besten Fitnesswert (vgl. Kapitel 5.2). Aus diesem Grund benötigt das Optimieren der Fitnesswerte bei einer ungünstigen Startkonstellation in diesem Fall einen großen Zeitaufwand. Bei einem qo-Wert von 0,5 wird die Wegentscheidung durch die Wahrscheinlichkeitsverteilung und die Wegentscheidung in Abhängigkeit des Pfades mit dem besten Fitnesswert gleich stark berücksichtigt. Es ist zu erkennen, dass bei der vorgegebenen Konfiguration aus der Tabelle 8.3 für den qo-Wert von 0,5 die besten Fitnesswerte generiert worden sind. Um den Einfluss des qo-Wertes auf das Optimierungsverhalten innerhalb eines Laufs zu untersuchen, wird die folgende Abbildung 8.2 betrachtet.

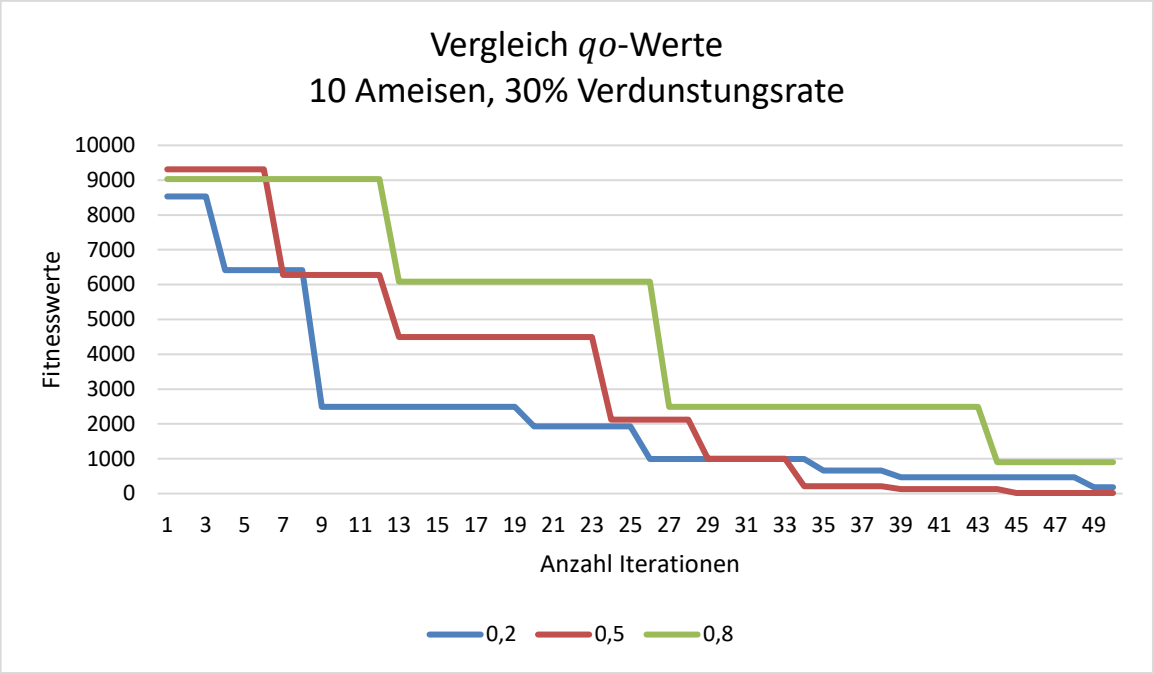


Abbildung 8.2: Vergleich qo-Werte

Bei den anfänglichen Iterationen werden ähnliche Fitnesswerte gefunden. Aus der Abbildung 8.2 ist zu entnehmen, dass die Optimierung mit einem qo-Wert von 0,2 bis zur 29ten Iteration am schnellsten erfolgt. Ab der 30ten Iteration werden durch einen qo-Wert von 0,5 bessere Fitnesswerte erzielt. Die ersten Fitnesswerte der Läufe sind weit vom Optimum entfern. Bei einem qo-Wert von 0,2 binden sich die Ameisen der aktuellen Iteration nicht so stark an die besten Fitnesswerte der vorherigen Iterationen wie bei einem qo-Wert von 0,5. Dies hat zur Folge, dass mit einem qo-Wert von 0,2 bereits nach wenigen Iterationen bessere Fitnesswerte als mit einem qo-Wert von 0,5 gefunden werden. Auf lange Sicht liefert ein qo-Wert von 0,5 bessere Resultate. Dies hat den Grund, dass ein qo-Wert von 0,5 ab der Findung eines guten Fitnesswertes vorteilhafter ist. Bei einem qo-Wert von 0,8 lösen sich die Ameisen nur mit geringer Wahrscheinlichkeit von den besten Fitnesswerten der vorherigen Iterationen. Dies führt dazu, dass der Graph in Abbildung 8.2 über viele Iterationen ein konstantes Verhalten aufweist. Aus demselben Grund sind lediglich drei Verbesserungen des Fitnesswertes zu verzeichnen.

Der dritte Einflussfaktor, der in dem vorliegenden Simulationsmodell variiert wird, ist die Anzahl der Ameisen. Wie in der Tabelle 8.1 ersichtlich worden ist, hängt der generierte Fitnesswert sehr stark von der zufällig generierten Startkonstellation ab. Je höher die Anzahl der Läufe pro Iteration, desto höher ist die Wahrscheinlichkeit, dass eine gute Startkonstellation erzeugt wird. Diese Startkonstellation dient als gute Voraussetzung für das weitere Optimieren. Daher ist die Berücksichtigung dieses Faktors sehr wichtig. In der folgenden Tabelle 8.4 sind die

Simulationsergebnisse für Läufe mit fünf, zehn und fünfzig Ameisen abgebildet. Dabei wird eine Verdunstungsrate von 30% und ein qo-Wert von 0,5 gewählt.

Anfangsverteilung random; qo-Wert;	Anzahl der Ameisen		
	5	10	50
10 Iterationen	6090,7	4994,1	1212,8
50 Iterationen	39,5	28	16,5

Tabelle 8.4: Durchschnittliche Fitnesswerte verschiedener Ameisenanzahlen

Wie aus der Tabelle 8.4 ersichtlich wird, liefert die Konfiguration mit 50 Ameisen den deutlich besten Fitnesswert. Der Einfluss der Ameisen wird bereits nach zehn Iterationen deutlich. Der generierte Fitnesswert bei einer Anzahl von 50 Ameisen, ist bereits nach zehn Iterationen deutlich besser als der Fitnesswert bei einer Anzahl von zehn Ameisen. In der folgenden Abbildung 8.4 ist der Verlauf der Fitnesswerte von Läufen mit unterschiedlichen Anzahl an Ameisen abgebildet.

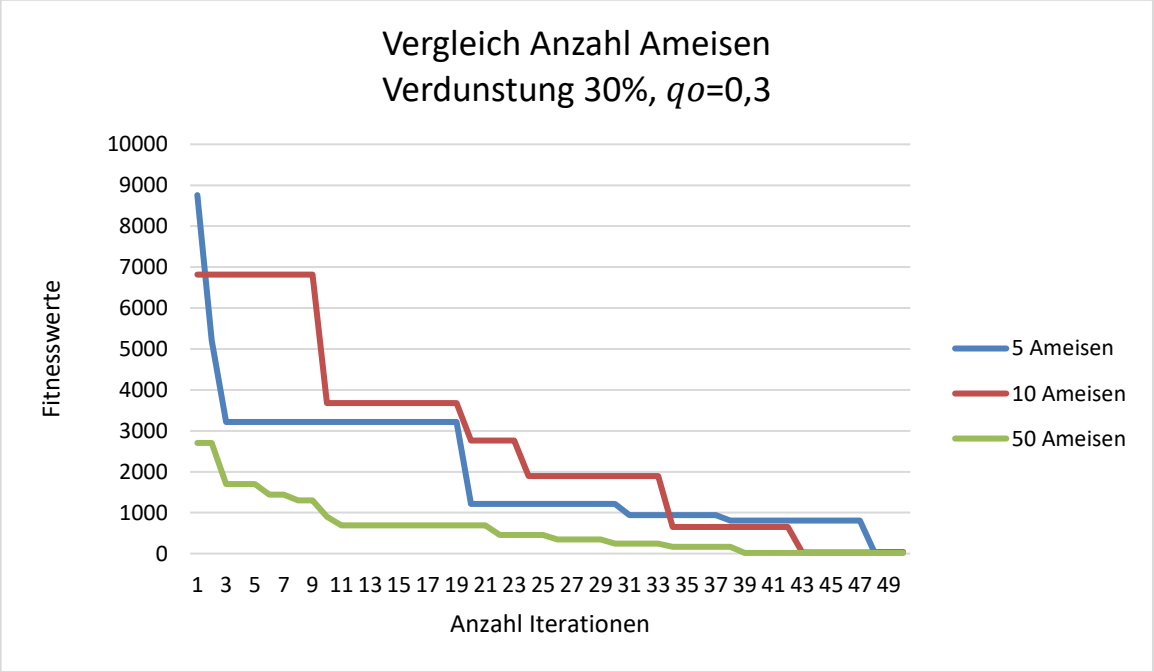


Abbildung 8.3: Vergleich Anzahl der Ameisen

Anders als bei den bisher untersuchten Einflussfaktoren, unterscheiden sich die Fitnesswerte der ersten Iterationen sehr stark. Dies hat den Grund, dass durch eine hohe Anzahl der Ameisen die Wahrscheinlichkeit einen guten Fitnesswert zu finden erhöht wird. Der Abbildung 8.3 ist zu entnehmen, dass je höher die Anzahl der Ameisen ist, desto öfter eine Verbesserung des Fitnesswertes stattfindet. Daher sind im Graphen mit zehn Ameisen mehr Sprünge als bei

dem Graphen mit fünf Ameisen vorhanden. Die meisten Sprünge beinhaltet der Graph des Laufs mit 50 Ameisen.

Bisher sind nur zufällige Startkonstellationen generiert worden. Um zu verdeutlichen, wie stark die Fitnesswerte einzelner Startkonstellationen voneinander abweichen können, werden zwei beispielhafte Startkonstellationen durchgeführt und miteinander verglichen. Die erste Startkonstellation erfasst den Fall, dass alle Arbeitsstationen die maximale Anzahl an Maschinen beinhalten. Bei der zweiten Startkonstellation beinhalten die Arbeitsstationen jeweils eine Maschine. Die Simulationsergebnisse beider Konstellationen sind in den folgenden zwei Tabellen abgebildet.

Anfangsverteilung AS/GS max.	$Verdunstungsrate\ \rho$		
	0,1	0,3	0,6
10 Iteration	1963,1	1164,4	1625,1
50 Iterationen	1,2	0,5	0,8
Anfangsverteilung AS/GS max.	q_0-Wert		
	0,2	0,5	0,8
10 Iteration	1457,5	1113,1	1360,8
50 Iterationen	0,7	0,3	0,6
Anfangsverteilung AS/GS max.	Anzahl der Ameisen		
	5	10	50
10 Iteration	1827,6	915,4	9,6
50 Iterationen	38,9	1,1	0,4

Tabelle 8.5: Fitnesswerte bei maximaler Anzahl der Maschinen

Anfangsverteilung M/WS & GS = 1	$Verdunstungsrate\ \rho$		
	0,1	0,3	0,6
10 Iteration	8090,7	7056,1	8225,1
50 Iterationen	6072,1	5780,5	7420,9
Anfangsverteilung M/WS & GS = 1	q_0-Wert		
	0,2	0,5	0,8
10 Iteration	4457,5	5743,1	7410,8
50 Iterationen	2356,1	1053,4	2789,7
Anfangsverteilung M/WS & GS = 1	Anzahl der Ameisen		
	5	10	50
10 Iteration	6151,4	4445,2	1211,5
50 Iterationen	1113,2	928,6	510,5

Tabelle 8.6: Fitnesswerte bei minimaler Anzahl der Maschinen

Bei der Untersuchung der Tabelle 8.5 fällt zunächst auf, dass viele Fitnesswerte sehr nah an dem globalen Optimum liegen. So ist mit einer Verdunstungsrate von 30%, einem q_0-Wert von 0,5 nach 50 Iterationen und 50 Ameisen ein Fitnesswert von 0,5 erreicht worden. Auch in diesem Fall wird ersichtlich, dass mit steigender Anzahl an Iterationen eine stetige Verbesserung des Fitnesswertes erfolgt. Auch bei der zweiten Startkonfiguration findet die Verbesserung der Fitnesswerte statt. Die Intervalle der erzeugten Fitnesswerte unterscheiden sich jedoch stark. Das Intervall der Fitnesswerte der ersten Startkonstellation befindet sich zwischen 0,5 und 1963,1. Während sich das Intervall der Fitnesswerte der ersten Startkonstellation in der Nähe des globalen Optimums befindet, liefert die zweite Startkonstellation erheblich schlechtere Lösungen. So befindet sich das Intervall der zweiten Startkonstellation zwischen 510,5 und 8225,1. Der schlechteste Fitnesswert aus der Tabelle 8.5 ist also besser als der beste Fitnesswert aus der Tabelle 8.6. Um den Einfluss der Startkonstellationen auf die Entwicklung der Fitnesswerte untersuchen zu können, wird die folgende Abbildung 8.4 betrachtet. Dabei wird für den besseren Vergleich eine dritte Startkonstellation mit jeweils drei Maschinen in jeder Arbeitsstation hinzugefügt.

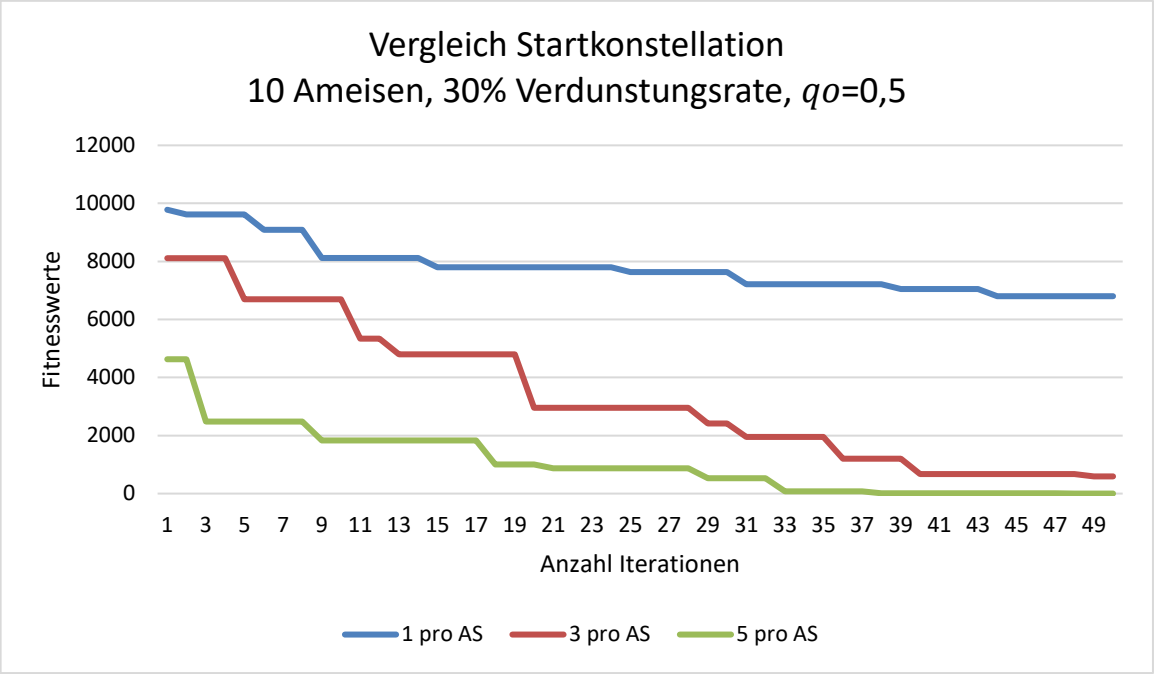


Abbildung 8.4: Vergleich Startkonstellation

Aus der Abbildung 8.4 wird ersichtlich, dass die Startkonstellation mit fünf Maschinen pro Arbeitsstation die besten Fitnesswerte liefert. Daraus folgt, dass die Abarbeitung der Auftragsliste eine hohe Anzahl an Maschinen pro Arbeitsstation verlangt.

Aus der Gegenüberstellung der drei Startkonstellationen folgt die Erkenntnis, dass die Wahl der Startkonstellation einen großen Einfluss auf die ausgegebene Lösung besitzt. Es ist zwar möglich mit einer schlechten Startkonstellation Fitnesswerte in der Nähe des globalen Optimums zu finden, jedoch ist der dafür benötigte Zeitaufwand immens. Um den Zeitaufwand deutlich zu minimieren, ist umfangreiches Wissen des Anwenders über das Produktionssystem notwendig.

8.2 Scheduling von Auftragsreihenfolgen

Bei der Auswertung des Scheduling von Auftragsreihenfolgen wird hauptsächlich die Anzahl der einzelnen Jobtypen variiert. Dabei wird das Verhalten des Algorithmus hinsichtlich dieses Parameters untersucht. Der Einfluss der übrigen Parameter ist bereits im Kapitel 8.1 analysiert worden. Um die Funktionalität des Optimierungsverfahrens zu verdeutlichen, werden einzelne Auftragsreihenfolgen in Gantt-Diagrammen dargestellt.

Zu Beginn wird zunächst eine beliebige Konstellation von Jobtypen simuliert. Dabei enthält die Auftragsliste immer genau neun Einträge. Für diese geringe Anzahl an Einträgen in der Auftragsliste, wird jede Arbeitsstation mit lediglich einer Maschine belegt. Dies hat den Grund, dass ansonsten die Maschinen unterlastet sind. Wird die Anzahl der Maschinen pro Arbeitsstation erhöht, so tritt der Fall ein, dass alle Jobs gleichzeitig bearbeitet werden. So

werden in jedem Lauf identische Fitnesswerte generiert. In der folgenden Tabelle 8.7 sind die Fitnesswerte aus verschiedenen Simulationsexperimenten mit einer Maschine pro Arbeitsstation dargestellt.

Anfangsverteilung AS/GS max.	$Verdunstungsrate\ \rho$		
	0,1	0,3	0,6
10 Iteration	31,93	31,42	32,01
100 Iterationen	31,24	30,87	31,39
Anfangsverteilung AS/GS max.	q_0-Wert		
	0,2	0,5	0,8
10 Iteration	31,84	31,52	32,08
100 Iterationen	30,97	30,46	31,47
Anfangsverteilung AS/GS max.	Anzahl der Ameisen		
	5	10	50
10 Iteration	32,11	31,04	30,93
100 Iterationen	31,25	30,61	30,19

Tabelle 8.7: Fitnesswerte bei beliebiger Konstellation von Jobtypen

Wie in der Tabelle 8.7 zu erkennen ist, unterscheiden sich die Fitnesswerte lediglich sehr wenig voneinander. Dies folgt aus der Anzahl der Einträge in der Auftragsliste. Für das Scheduling von Auftragsreihenfolgen ist die Auftragsliste von 1200 Einträgen auf neun Einträge reduziert worden. Des Weiteren wird in der Tabelle 8.7 deutlich, dass sowohl bei dem Scheduling von Auftragsreihenfolgen als auch bei der Variation des modularen Produktionssystems die gleichen Konfigurationen die besten Fitnesswerte liefern. Mit 50 Ameisen, einer Verdunstungsrate von 30% und einem q_0-Wert von 0,5 resultieren die besten Ergebnisse. Da durch eine solch hohe Anzahl von Ameisen die Simulationszeit stark erhöht wird, wird diese auf 10 Ameisen festgelegt. Der Einfluss der spezifischen Parameter des Algorithmus ist beim Scheduling von Auftragsreihenfolgen identisch wie bei der Bestimmung der Maschinenkonstellation (vgl. Kapitel 8.1). Aus diesem Grund wird nun der Einfluss der Anzahl der Jobtypen in der Auftragsliste auf den Fitnesswert untersucht. In der folgenden Abbildung 8.5 sind die Fitnesswerte von vier verschiedenen Läufen mit unterschiedlicher Anzahl an Jobtypen über die Anzahl der Iterationen abgebildet.

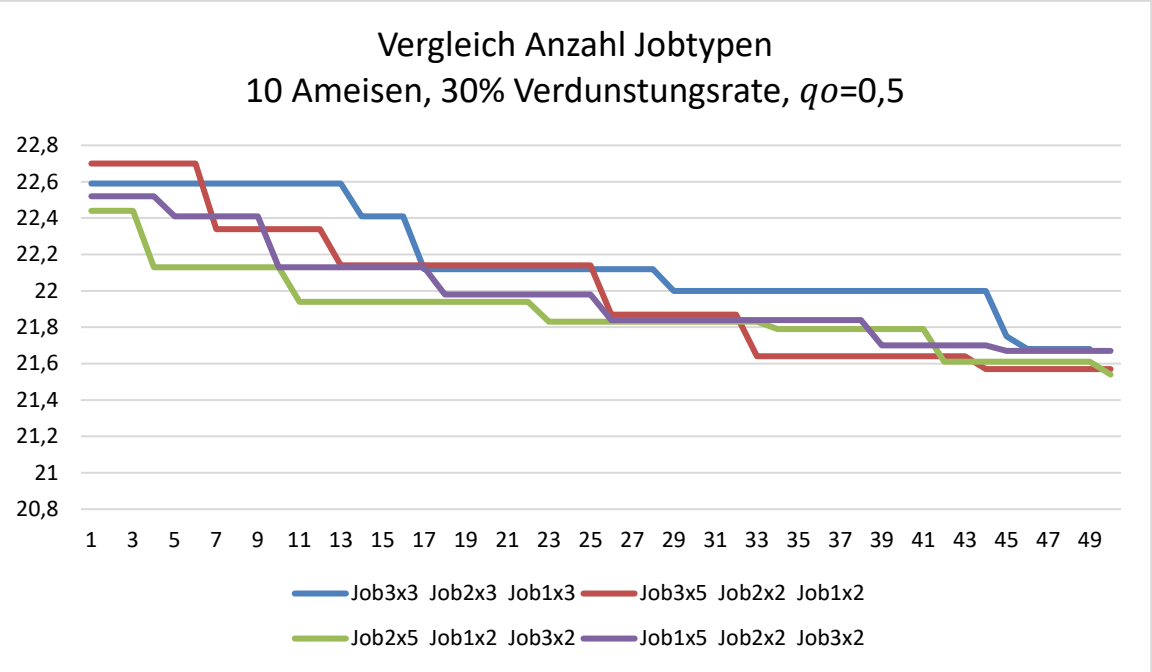


Abbildung 8.5: Vergleich Anzahl Jobtypen

Wie in der Abbildung 8.5 ersichtlich wird, wird jeder Lauf mit steigender Anzahl an Iterationsschritten optimiert. Der Unterschied der Fitnesswerte nach einer Iteration und nach 50 Iterationen ist jedoch sehr gering. Die größte Verbesserung des Fitnesswertes weist der rote Graph auf. Die durch den grünen Graphen dargestellte Auftragsliste besitzt sowohl nach der ersten als auch nach der letzten Iteration den besten Fitnesswert. Dies zeigt, dass die Zusammenstellung der Auftragsliste einen bedeutenden Einfluss auf die Fitnesswerte hat. Mit Hilfe der Abbildung 8.5 ist jedoch eine Aussage über den Einfluss der Auftragsreihenfolge auf die Gesamtzykluszeiten nicht möglich. Um diesen Einfluss untersuchen zu können, werden in der folgenden Abbildung 8.6 zwei unterschiedliche Auftragsreihenfolgen in zwei Gantt-Diagramm gegenübergestellt (vgl. Kapitel 2.2).

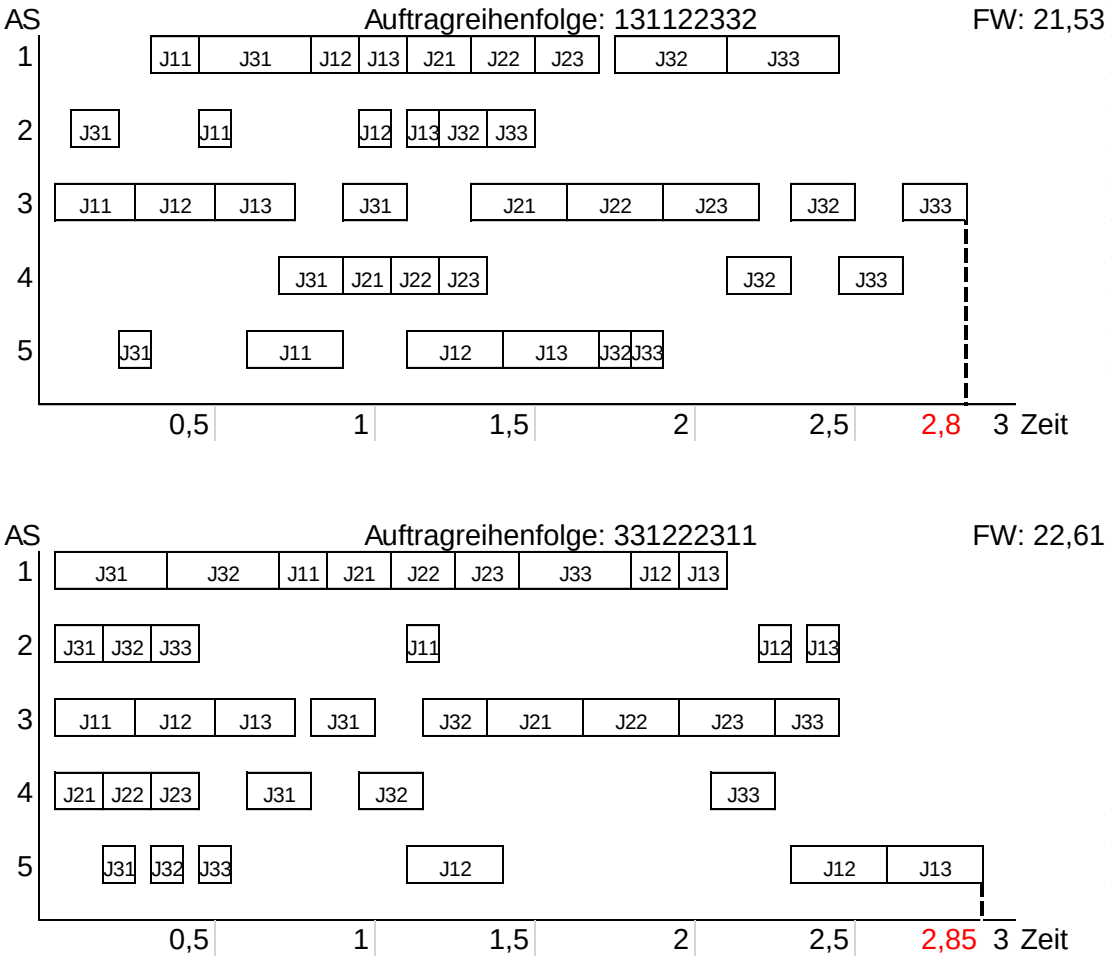


Abbildung 8.6: Auswertung im Gantt-Diagramm

In der Abbildung 8.6 ist eine Auftragsliste mit zwei unterschiedlichen Auftragsreihenfolgen dargestellt. Diese Auftragsliste enthält alle Jobtypen jeweils drei Mal. Dabei handelt es sich bei dem unteren Gantt-Diagramm um die Auftragsreihenfolge nach der ersten Iteration. Bei dem oberen Gantt-Diagramm handelt es sich um die Auftragsreihenfolge nach der 50. Iteration. Der Fitnesswert nach der 50. Iteration ist um 1,08 besser als der Fitnesswert nach der ersten Iteration. Dies resultiert aus der um 0,05 Zeiteinheiten kürzeren Gesamtzykluszeit, welche in der Abbildung 8.6 ersichtlich wird. Somit ist zwar eine Optimierung der Auftragsreihenfolge der Auftragsliste erfolgt, jedoch ist die Optimierung der Gesamtzykluszeit marginal. Dies hat den Grund, dass lediglich neun Aufträge in einer Auftragsliste enthalten sind und die Leerzeiten nicht stark reduziert werden können. Mit steigender Anzahl an Aufträgen ist eine bessere Optimierung der Gesamtzykluszeit möglich. Dafür wird jedoch eine ausreichend hohe Rechnerleistung benötigt (vgl. Kapitel 5.2).

9 Zusammenfassung und Fazit

Im Fokus dieser Bachelorarbeit wird die Ant-Colony-Optimierung für die Produktionsplanung genutzt. Dafür dient ein festgelegtes Produktionssystem als Grundlage. Hierbei wird die Ant-Colony-Optimierung zum einen zur Variation modular modellierter Produktionssysteme und zum anderen zum Scheduling von Auftragsreihenfolgen verwendet.

Nach der Entwicklung einer Ant-Colony-Optimierung für die Variation des Produktionssystems und für das Scheduling von Auftragsreihenfolgen, ist diese umgesetzt worden. Dazu sind für beide Problemstellungen zunächst Bäume erstellt worden. Diese Bäume repräsentieren den Lösungsraum des jeweiligen Problems. Des Weiteren beinhalten die Bäume die von den künstlichen Ameisen belaufenen Pfade. Nachdem die Ant-Colony-Optimierung für das Produktionssystem umgesetzt worden ist, erfolgte die Auswertung und Analyse. Hierbei wurden alle spezifischen Parameter der ACO-Metaheuristik variiert und ihre Auswirkung auf die Simulationsergebnisse untersucht. Als Simulationsergebnisse sind die Fitnesswerte der einzelnen Maschinenkonstellationen beziehungsweise Auftragsreihenfolgen ausgegeben worden. Zu den spezifischen Parametern zählen die Anzahl der Ameisen, die Verdunstungsrate, der q_0-Wert und die Grundpheromonmenge. Weitere Parameter des Simulationsmodells sind die Anzahl der Iterationen und die Zusammenstellung der Auftragsliste. Dabei wird die Zusammenstellung der Auftragsliste lediglich bei dem Scheduling der Auftragsreihenfolgen eingegeben.

Zunächst konnte festgestellt werden, dass unabhängig von der Wahl der genannten Parameter eine Optimierung der Fitnesswerte erfolgt. Bei der Auswertung der Simulation ist hervorgekommen, dass je höher die Anzahl der Ameisen gewählt wird, desto weniger Iterationsschritte für die Findung guter Fitnesswerte benötigt werden. Außerdem hat sich ein q_0-Wert von 0,5 als beste Einstellung der untersuchten Werte erwiesen. Bei der Untersuchung der Verdunstungsrate ist festgestellt worden, dass sowohl eine niedrige als auch eine hohe Wahl der Verdunstungsrate unvorteilhaft für die Findung guter Fitnesswerte ist. Bei der Bestimmung der Maschinenkonstellation stellte sich heraus, dass für eine gute Lösung des Problems binnen kurzer Zeit ein tiefes Verständnis über das Produktionssystem vorliegen muss. Ohne diese Kenntnis ist dafür eine deutlich höhere Simulationszeit notwendig. Außerdem ist die Differenz zwischen den Fitnesswerten nach der ersten und nach der letzten Iteration sehr hoch. Dies ist beim Scheduling der Auftragsreihenfolge nicht der Fall. Es findet zwar eine Optimierung statt, jedoch ist diese aufgrund der wenigen Einträge in der Auftragsliste sehr gering. Die Anzahl der Einträge musste wegen der geringen Rechnerleistung stark reduziert werden.

Beide Algorithmen funktionieren und liefern optimierte Ergebnisse. Somit können diese als Grundlage für zukünftige Forschung im Rahmen der simulationsbasierten Optimierung verwendet werden. Mit dem technischen Fortschritt und steigender Rechnerleistung, können die Algorithmen auf immer komplexere Systeme angewendet werden. So müsste die Auftragsliste bei dem Scheduling von Auftragsreihenfolgen nicht reduziert werden. Zudem besteht die Möglichkeit die Algorithmen zu modifizieren. Somit kann beispielsweise eine Kandidatenliste erstellt werden, welche eine breite Suche des Lösungsraums bereits in anfänglichen Iterationen ermöglicht und eine Suche in Zyklen verhindert. Dies ist lediglich ein Beispiel für mögliche Modifikationen der Algorithmen, um Simulationsergebnisse in der Nähe des globalen Optimums innerhalb einer kurzen Zeit zu erhalten.

Literaturverzeichnis

Acker I (2011) Methoden der mehrstufigen Ablaufplanung in der Halbleiterindustrie. Wiesbaden: Gabler

Balzert H (2009) Lehrbuch der Software-Technik. 3. Aufl. Heidelberg [u.a.]: Spektrum Akademischer Verlag, S. 303-309

Bonabeau E; Dorigo M; Theraulaz G (1999) Swarm Intelligence: From Nature to Artificial Systems, Oxford University Press, New York, S.45

Brucker P (2004) Scheduling algorithms. Berlin, Heidelberg Springer

Dombrowski U (2015) Ganzheitliche Produktionssysteme, Braunschweig

Domschke W (1997) Scholl, A.: Voß, Produktionsplanung: Ablauforganisatorische Aspekte. Berlin, Heidelberg, New York: Springer, S. 51

Domschke W; Drexl A (2005) Einführung in Operation Research. Berlin: Springer

Domschke W; Scholl A (2006) Heuristische Verfahren. Jena: Wirtschaftswissenschaftliche Fakultät, Friedrich-Schiller-Universität Jena

Dong M (2016) Untersuchung von Validierungsmöglichkeiten für modulare, auf objektorientierten Petri-Netzen basierende Simulationsmodelle, Dortmund

Dorigo M; Di Caro, G. (1999) The Ant Colony Optimization Meta-Heuristic. In: New Ideas in Optimization. Corne, D. and Dorigo, M. and Glower, F. McGrawHill, London, S. 13ff.

Dorigo M; Di Caro G; Gambardella L.M. (1999) Ant Algorithms for Discrete Optimization. In: Artificial Life 5, Nr. 2, S. 137–172

Dorigo M; Gambardella L.M. (1997) Ant colony system: A cooperative learning approach to the traveling salesman problem. In: IEEE Transactions on Evolutionary Computation 1, Nr. 1, S. 53-66

Dorigo M; Maniezzo V; Colorni A (1996) The ant system: optimization by a colony of cooperating agents. In: IEEE Transactions on Systems, Man and Cybernetics_Part B 26, Nr. 1, S. 29-41

Dorigo M; Stützle T (2004) Ant Colony Optimization. Cambridge: MIT Press

Dowsland K.A. (1993) Simulated annealing. In: Reeves, C.R. (Hrsg.): Modern heuristic techniques for combinatorial problems. New York: John Wiley & Sons, S.20-69

Dueck G, Scheuer T; Wallmeier H.-M. (1993) Toleranzschwelle und Sintflut: Neue Ideen zur Optimierung. Spektrum der Wissenschaft, 3, S. 42-51

Eckhardt I (2015) Konzeptentwicklung für die Kopplung heuristischer Optimierung und ereignisdiskreter Simulation für Ablaufplanungsprobleme, Dortmund

Ehrgott M; Gandibleux X (2004) Approximative solution methods for multiobjective combinatorial optimization. TOP: An Official Journal of the Spanish Society of Statistics and Operation Research

Eigner M (2012): Informationstechnologie für Ingenieure. Berlin, Heidelberg: Springer Berlin Heidelberg

ElMaraghy H, Wiendahl H.-P. (2009) Changeability – An Introduction. In: ElMaraghy, H.(Hrsg.): Changeable and reconfigurable Manufacturing Systems. Berlin/Heidelberg
Engelbrecht, A.P. 2005 Fundamentals of Computational Swarm Intelligence. Chichester: Wiley

Feggeler A; Neuhaus R (2002): Was ist neu an Ganzheitlichen Produktionssystemen? In:Institut für angewendete Arbeitswissenschaft e.V. (Hrsg.): Ganzheitliche Produktionssysteme– Gestaltungsprinzipien und deren Verknüpfung. Köln: Wirtschaftsverlag Bachem, S. 18-26.

Geifer M.J. (2005) Multikriterielle Ablaufplanung. Wiesbaden: Deutscher Universitäts-Verlag

Goss S; Aron S.; Deneburg, J. L.; Pasteels J. M. (1989) Self-organized Shortcuts in the Argentine Ant. In: Naturwissenschaften 76, S. 579–581

Graf S (2003) Auswahl und Implementierung eines Ameisenalgorithmus‘ zur Steuerung von Patienten im Planspiel „INVENT“, Wien

Graham R.L. (1979) Optimization and approximation in deterministic sequencing and scheduling, New York

Günther H.-O.; Tempelmeier H (1995): Produktionsmanagement – Einführung und Übungsaufgaben. Berlin/Heidelberg

Hillier F.F. (1969) Efficient heuristic procedures for integer linear programming with an interior. Operation Research 17, 4, S.600-637

Huckert K; Rhode R; Roglin O; Weber R (1980) On the Interactive Solution to a Multicriteria Scheduling Problem. In: Zeitschrift für Operations-Research 24, S. 47–60

Kallrath J (2013) Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis. Wiesbaden: Springer

Kistner K.-P.: Steven M (2001) Produktionsplanung. Heidelberg: Physica-Verlag

Klemmt A (2012) Ablaufplanung in der Halbleiter- und Elektronikproduktion: Hybride Optimierungsverfahren und Dekompositionstechniken. Wiesbaden: Springer Vieweg

Koren Y; Shpitalni M (2010): Design of reconfigurable manufacturing systems. In: Journal of Manufacturing Systems 29, Nr. 4, S.130-141.

Korge A; Scholtz O (2004): Produzierende Unternehmen innovative organisieren und führen. In: wt Werkstattstechnik online 94, Nr. 1-2, S. 2-6.

März L; Krug W; Rose O; Weigert G (2011): Simulation und Optimierung in Produktion und Logistic. Praxisorientierter Leitfaden mit Fallbeispielen. Heidelberg

Matt D.T. (2005): Design of self-contained, adaptable factory modules. In: Proceedings of the1st International Conference on Changeable, Agile, Reconfigurable and Virtual Production. München

MTM 2012 Deutsche MTM Vereinigung e.V.: Glossar: Ganzheitliches Produktionssystem, elektronischveröffentlicht:URL:https://www.dmtm.com/forschung/glossar/index.php?buchstabe=G&PHPSESSID=4979e7b0c646cbdc3c797786731ac5a4 [Stand 16.07.2017]

Mühlenbrock M (2016) Spezifikation und Klassifizierung von Objekten in Produktionssystemen zur Simulation mittels Petri-Netzen

Müller-Merbach H (1970) Optimale Reihenfolgen. Berlin, Heidelberg: Springer

Ohno T (1993): Das Toyota-Produktionssystem. Frankfurt

Petri C.A.(1962): Kommunikation mit Automaten, Darmstadt

Pidd M (1992): Computer Simulation in management science. Verlag: John Wiley & Sons Ltd; Auflage: 3 Sub

Pinedo M (1995) Scheduling: Theory, Algorithms, and Systems. Englewood Cliffs, New Jersey

Pirlot M (1996) General local search methods. European Journal of Operation Research 92, 3, S 493-511

Pothmann N (2007) Kreuzungsminimierung für k-seitige Buchzeichnungen von Graphen mit Ameisenalgorithmen, Dortmund

Rabe M; Deininger M (2015) Discrete Event Simulation of Modular Produciton System Models using Petri Nets, Dortmund

Rabe M (2008) Verifikation und Validierung für die Simulation in Produktion und Logistik, Kassel

Rabe M; Deininger M (2013) Fabrikmodelle für Job-Shop-Scheduling Algorithmen in Changing-Steady-State-Systemen, Dortmund

Rauch E (2013) Konzept eines wa

Reising W (2010) Petrinetze – Modellierungstechnik, Analysemethoden, Fallstudien, Berlin
Rieck, J. 2009 Tourenplanung mittelständiger Speditionsunternehmen: Modelle und Methoden Wiesbaden: Gabler

Schnieder E (1999): Methoden der Automatisierung, Braunschweig/ Wiesbaden: Verlag Vieweg

Schöf S; Wieting R; Sonnenschein M (1997) Distributed Net Simulation: DNS; Abschlußbericht:OFFIS, Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme, Forschungsbereich 4, Systemmodellierung

Schuster C.J. (2003) No-wait Job-Shop-Scheduling: Komplexität und Local Search, Duisburg

Schutten J (1998) Practical job shop scheduling. Annals of Operations-Research 83, Enschede, Niederlande

Schwartz F (2004) Störungsmanagement in Produktionssystemen. Aachen: Shaker

Siemens (2014) Tecnomatix Plant Simulation 12 Schritt-für-Schritt-Hilfe

Spath D. (2003): Ganzheitlich produzieren: Innovative Organisation und Führung, Stuttgart

Streim H (1975) Heuristische Lösungsverfahren: Versuch einer Begriffserklärung. Zeitschrift für Operation Research 19, S.143-162

Stützle T; Hoos H.H. (2000) MAX MIN Ant system. In: Journal of Future Generation Computer Systems 16, Nr. 8, S. 889-914

Troßmann E (1996) Ablaufplanung bei Einzel- und Serienproduktion. In: Kern, W.; Schröder, H.; Weber, J.: Handwörterbuch der Produktionswirtschaft. Stuttgart: Schäffer-Poeschel

Vennemann A (2015) Vorgehensweise zur Aufbereitung von Eingangs- und Ergebnisdaten einer ereignisdisktreten Simulation eines Logistiknetzwerks des Werkstoffhandels zur glaubwürdigen Messbarkeit von komplexen Data-Warehouse-Kennzahlen, Dortmund

Wiendahl H.-P.; Reichardt J (2009) Nyhuis, P.: Handbuch Fabrikplanung – Konzept, Gestaltungund Umsetzung wandlungsfähiger Produktionsstätten. München/Wien

Wienholdt, H.; Meyer, J.C. (2007): Kundenindividuelle Produkte zu Kosten der Massenproduktion.In: UdZ Unternehmen der Zukunft (FIR Zeitschrift für Betriebsorganisation und Unternehmensentwicklung) 8, Nr. 3, S. 30-33.

Wieting R (1996): Handbuch zur THORN Entwicklungsumgebung. Oldenburg

Wildemann H (2004): Der Wertbeitrag der Produktion – Entwicklungspfade von Produktionssystemen. In: ZfB Zeitschrift für Betriebswirtschaft 74 Nr. 4, S. 49-68.

VDI (1996) VDI-Richtlinie 3633 „Begriffsdefinitionen“, Entwurf (Gründruck). Beuth, Berlin

VDI (2008) VDI-Richtlinie 3633 Blatt 1 „Simulation von Logistik-, Materialflussund Produktionssystemen“. Beuth, Berlin

Yamada T; Nakano R (1997): Genetic algorithms in engineering systems, 1997, Cambridge

Zapf M; Heinzl A (1998): Ansätze zur Integration von objektorientierten Konzepten und Petri-Netzen, Bayreuteh: Universität Bayreuth

Zäpfel G; Braune R (2005) Moderne Heuristiken der Produktionsplanung: am Beispiel der Maschinenbelegung. München: Vahlen