Institut für Rechnergestützte Ingenieursysteme
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Diplomarbeit Nr. 3732

Anwendung von künstlichen neuronalen Netzen in Steuerung und Analyse von intralogistischen Materialflusssystemen (MFS) im Rahmen von Industrie 4.0

Benjamin Schehrer

Studiengang:	Informatik
Prüfer:	Univ-Prof. Hon-Prof. Dr. Dieter Roller
Betreuer:	Univ-Prof. Hon-Prof. Dr. Dieter Roller
begonnen am:	01.10.2015
beendet am	31.03.2016
CR-Klassifikation:	F.1.1, I.2.8, I.2.11, I.6.3, I.6.8

Zusammenfassung

In modernen Intralogistikanlagen ist optimale Materialflusssteuerung die wichtigste Funktion neben der Verwaltung der eingelagerten Waren. Um eine kostspielige Konfiguration der oft sehr komplexen Anlagen zu vermeiden, ist es notwendig, den Konfigurationsprozess zu automatisieren.

Das Ziel dieser Arbeit soll ein Prototyp zur Ermittlung der optimalen Konfiguration, im Hinblick auf Leistung und Verschleiß, von intralogistischen Materialflüssen sein.

Hierzu werden im Rahmen dieser Diplomarbeit zunächst Inhouse-Experten zu den unterschiedlichen Konfigurationsparametern befragt. Auf Basis der gewonnen Daten soll ein abstraktes Modell zur Modellierung der verschiedenen Intralogistikanlagen erstellt werden.

Des Weiteren sollen die unterschiedlichen Methoden zur automatisierten Optimierung der Lagerflüsse untersucht und die Methode der künstlichen neuronalen Netze in einem Prototyp, unter Verwendung des abstrakten Modells, umgesetzt werden.

Der Prototyp soll in der Lage sein, eine optimale Konfiguration für ein gegebenes Modell eines intralogistischen Materialflusssystems zu ermitteln.

Die ermittelte Konfiguration soll in einem nächsten Schritt gegen eine Standardinstallation der Software viadat und gegen eine Simulation in der Simulations- und Emulationssoftware Emulate 3D validiert werden können.

Für eine spätere Automatisierung der Validierung soll die Software Emulate 3D auf vorhandene Programmierschnittstellen untersucht werden. Des Weiteren sollen die Möglichkeiten der Anbindung der Modelllösung an die Standardsoftware viadat evaluiert werden.

Abstract

In modern intralogistics systems, the optimized material flow control is the most important function, along with the management of the stored goods. To avoid the expensive configuration of the facilities, most of which are highly complex, it is required to automate the configuration process.

The goal of this diploma thesis is to generate a prototype used to identify the optimum configuration of intralogistic material flows with regard to performance and wear.

In the context of this diploma thesis, first of all in-house experts are to be questioned concerning the different configuration parameters. On the basis of the obtained data, an abstract model used to model intralogistics facilities of different types is to be generated.

Moreover, different methods of automated optimization of the material flows are to be examined, and the method of artificial neural networks is to be implemented in a prototype, on the basis of the abstract model.

The prototype must be able to identify the optimum configuration for a given model of an intralogistic material flow system.

In the next step, it is intended to validate the identified configuration against a standard installation of the viadat software and against a simulation in the simulation and emulation software Emulate 3D.

With regard to the automation of the validation at a later time, the software Emulate 3D is to be examined for existing programming interfaces. Another task is to evaluate the option of connecting the model solution to the standard software viadat.

Inhaltsverzeichnis

1 EINLEITUNG

Im Folgenden werden die Kooperationspartner, Problemstellung, die konkrete Aufgabenstellung sowie die genaue Gliederung der vorliegenden Arbeit erläutert.

1.1 Kooperationspartner

Diese Arbeit wird vom Institut für Rechnergestützte Ingenieursysteme (IRIS) der Universität Stuttgart betreut und geprüft. Durchgeführt wird die vorliegende Arbeit in Zusammenarbeit mit der viastore Gruppe, bestehend aus viastore SYSTEMS GmbH und viastore SOFTWARE GmbH, und dem Institut für Rechnergestützte Ingenieursysteme (IRIS).

1.2 Danksagung

Ich möchte mich bei Univ-Prof. Hon-Prof. Dr. Dieter Roller für die Betreuung und für hilfreiche Hinweise zur Durchführung dieser Arbeit bedanken.

Weiterhin möchte ich mich bei Dirk Gehlich für die Erläuterungen zum Materialfluss sowie Michael Tophinke, Carsten Röhl und Thomas Jetter für unterstützende Beiträge zu dieser Arbeit bedanken. Ebenso danke ich Harald Metzger für seine Einführung in die Funktionsweise der Kommunikation zwischen der Software viadat und der SPS der Fördertechnik.

Darüber hinaus bedanke ich mich bei Dr. Martin Krebs für die Möglichkeit der Realisierung dieser Arbeit.

1.3 Problemstellung

In modernen Logistiksystemen ist die Optimierung der internen Abläufe von hoher Bedeutung für die Funktion des gesamten Logistiksystems. Hierzu wird der Materialfluss innerhalb solcher Logistiksysteme durch den Einsatz von Lagerverwaltungssystemen (LVS) gesteuert, diese besitzen i. d. R. eine zentrale Materialflusssteuerungseinheit. Die Materialflusssteuerungseinheit übernimmt die Koordination aller im Logistiksystem anfallenden Materialflussprozesse. Zur Steuerung der Materialflüsse innerhalb eines Logistiksystems wird im Kontext dieser Arbeit die Warehousemanagementsoftware viadat, im Folgenden als „*Software viadat*“ bezeichnet, der Firma viastore SOFTWARE GmbH eingesetzt. Abbildung 1-1 zeigt das Funktionsprinzip der Steuerung der Logistikprozesse durch den Einsatz eines LVS.

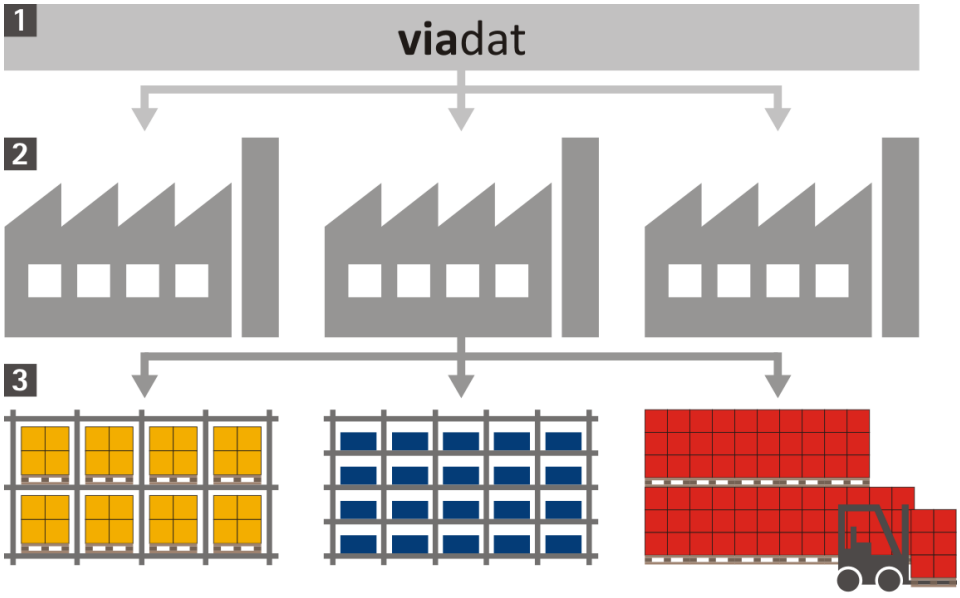

Abbildung 1-1: Das LVS viadat als zentrale Steuereinheit des Logistiksystems [via15]

Das Logistiksystem wird durch das LVS-Modul von der Software viadat (1) gesteuert. Das LVS überwacht alle Abläufe innerhalb eines Logistiksystems und speichert hierzu eine Fülle an Daten um die Materialflussprozesse optimal steuern zu können. Darüber hinaus kommuniziert es mit sog. Host-Systemen, auch Enterprise Ressource Planning System (ERP) genannt.
Ein Logistiksystem kann in mehrere Standorte (2) unterteilt sein, die alle ihre eigenen Materialflussprozesse haben. Die verschiedenen Standorte können in unterschiedliche Lagertypen (3), wie z. B. automatische oder manuelle Lager, unterteilt werden.

Hierbei wird das System zur Steuerung von Materialflüssen auch als Materialflusssystem (MFS) bezeichnet. Die Konfiguration eines Materialflusssystems (MFS) und der darin enthaltenen Flüsse geschieht momentan auf Basis von Erfahrungswerten der entsprechenden Experten und ist ein sehr langwieriger und auch kostspieliger Prozess. Die Konfiguration eines MFS wird i. d. R. in mehreren Iterationsschritten vollzogen. Am Anfang steht je nach Komplexität des MFS die Simulation der Materialflussprozesse durch die Simulationssoftware Emulate 3D. [Dem15]

Im Anschluss werden auf Basis der durch die Simulation gewonnenen Erkenntnisse die ersten Parameter zur Konfiguration des MFS, wie etwa die Anzahl der gleichzeitig in einer Strecke A befindlichen Fördereinheiten, definiert und das MFS vorkonfiguriert. Danach folgen i. d. R. mehrere Testreihen im laufenden Betrieb der Anlage, bis die geforderte Leistung erreicht ist, in jedem dieser Teilschritte müssen durch den Kunden Ressourcen in Form von Bedienern, FEs, etc. bereitgestellt werden. Jeder Teilschritt ist ein langwieriger Prozess, da eine Parameteränderung nur durch Beobachtungen und Vergleiche mit den Vorgaben verifiziert werden kann. Die durch dieses iterative Verfahren gewonnenen Parameter sind eine rein statische Konfiguration und können nur schwer an neue Gegebenheiten angepasst werden. Die Analyse der Ausgangssituation verdeutlicht die Notwendigkeit für ein MFS, das sich auch neuen Situationen anpassen kann.

Hierzu sollen künstliche neuronale Netze, auch neuronale Netze genannt, auf ihre Anwendbarkeit zur Analyse und Steuerung von intralogistischen Materialflüssen untersucht werden. Hierbei spielen vor allem Kosteneffizienz und Schnelligkeit bei der Entscheidungsfindung eine wichtige Rolle. Diese Arbeit stützt sich auf die Validierung eines geeigneten Systems und arbeitet unabhängig von der Software viadat der Firma viastore SOFTWARE GmbH. [via15]

1.4 Aufgabenstellung

Das Ziel dieser Arbeit besteht darin, die Anforderungen aktueller Materialflusssysteme zu untersuchen und ein Konzept zu entwickeln, das Materialflüsse durch die Verwendung neuronaler Netze analysiert und ggf. optimiert. Dabei soll das Hauptaugenmerk auf dem Durchsatz in Abhängigkeit des Verschleißes liegen.

Das Konzept soll in Form eines Prototyps durch die Simulation mittels der Software Emulate 3D [Dem15] und durch Simulation mit der Software viadat validiert werden. Hierbei spielt die Umsetzbarkeit und Leistung des Prototyps eine wichtige Rolle. Die zu verwendende Programmiersprache spielt eine untergeordnete Rolle.

1.5 Gliederung

Die vorliegende Arbeit gliedert sich in vier Teile, die dem Vorgehen bei der Erstellung und Auswertung eines ereignisorientierten Simulators zur Bewertung und Analyse eines Multiagenten-Reinforcement-Learning-Systems entsprechen.

Im ersten Teil werden das Materialflusssystem und seine Komponenten eingeführt und erläutert. Im Anschluss werden die aktuellen Methoden zur Analyse und Steuerung von MFS auf ihre praktische Anwendbarkeit und den Komplexitätsgrad untersucht.

Im zweiten Teil der Arbeit wird ausgehend von einer Referenzanlage der Firma viastore SYSTEMS ein abstraktes Materialflussmodell zur Verwendung eines Multiagentensystems erstellt. Aufbauend auf dem abstrakten Materialflussmodell wird ein Lösungsansatz skizziert.

Der dritte Teil der Arbeit beschreibt die konkrete Anwendung neuronaler Netze auf das im ersten Teil definierte Materialflussmodell. Hierzu wird ein geeignetes neuronales Netz zur

Anwendung ausgewählt und anhand eines Prototyps am Beispiel eines automatischen Lagersystems umgesetzt. Das Ergebnis des Prototyps soll eine optimale Parametrierung der Anlage sein.

Im letzten Teil der Arbeit soll untersucht werden, inwiefern der Prototyp an die Software viadat gekoppelt werden kann. Hierbei soll insbesondere die Ankopplung des Prototyps an den Livebetrieb der Anlage erörtert werden. Abschließend soll ein Ausblick über die Möglichkeit des Einsatzes von neuronalen Netzen im Hinblick auf Predictive Maintenance bzw. Predictive Analytics gegeben werden.

2 MATERIALFLUSS

Im nachfolgenden Kapitel werden die grundlegenden Begriffe des Materialflusses beschrieben.

2.1 Materialflusssystem

Der typische Verlauf eines Materialflusses in intralogistischen Materialflusssystemen (MFS) gestaltet sich vom Wareneingang (WE) bis zum Warenausgang (WA). I. d. R. werden der WE und der WA durch verschiedene Lagerbereiche, zumeist verteilt über unterschiedliche Standorte, voneinander getrennt. Dies setzt wiederum ein System zur Verteilung der Materialien in den einzelnen Bereichen voraus. Ein Materialflusssystem gliedert sich in die folgenden fünf Grundfunktionen:

- Bearbeiten (B)
- Fördern (F)
- Verteilen (V)
- Warten (W)
- Zusammenführen (Z)

Mit den oben genannten Grundfunktionen können alle Vorgänge innerhalb eines Materialflusssystems dargestellt werden. [Arn09]

Somit kann jedes Materialflusssystem über ein Flussdiagramm dargestellt werden. Abbildung 2-1 zeigt eine vereinfachte Darstellung des Materialflusses vom Wareneingang hin zum Warenausgang.

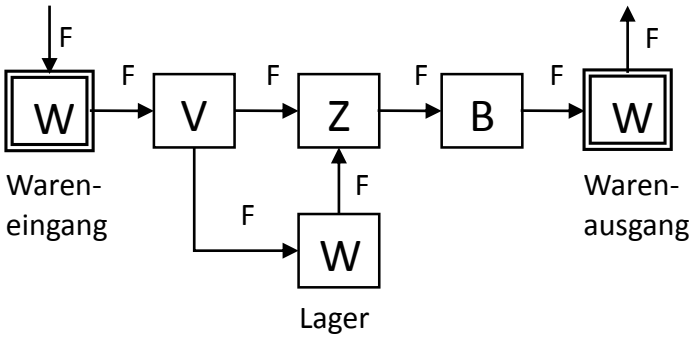

Abbildung 2-1: Materialflussdiagramm

Für den Durchlauf des Materials in Abbildung 2-1 ergeben sich nach Warenannahme zwei mögliche Materialflüsse:

a) Im Falle einer Einlagerung wird die Fördereinheit (FE) in das Lager befördert und wartet auf eine Auslagerungsanforderung. Danach wird die FE zurück auf das MFS befördert, bearbeitet, in den Warenausgang befördert und wartet abschließend im Warenausgang auf einen Weitertransport.
b) Wird die FE sofort bearbeitet, so wird die FE nach dem Wareneingang direkt zur Bearbeitung, danach in den Warenausgang befördert und wartet abschließend im Warenausgang auf einen Weitertransport.

2.1.1 Komponenten eines Materialflusssystems

Ein MFS besteht typischerweise aus verschiedenen Komponenten. Abbildung 2-2 zeigt am Beispiel einer fiktiven Anlage den Materialfluss und dessen Komponenten innerhalb eines MFS.

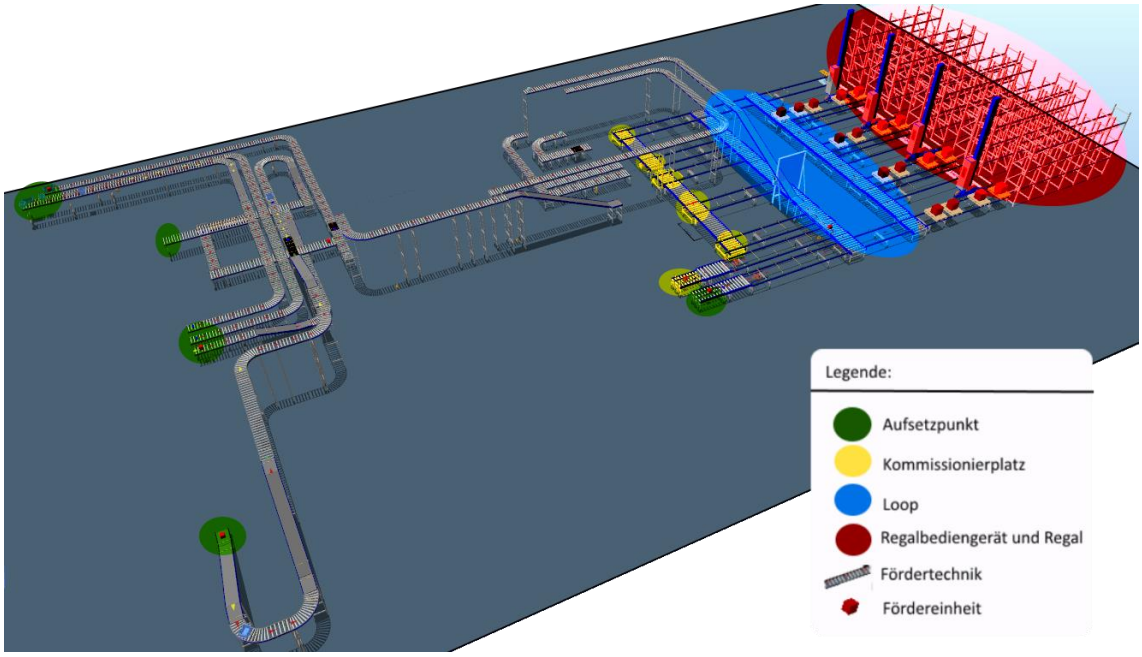

Abbildung 2-2: Materialflusssystem [via151]

In einem MFS sind i. A. die folgenden Komponenten zu finden:

- **Fördertechnik (grau)**

  Die Fördertechnik ist die Verbindung unterschiedlicher Komponenten innerhalb eines Materialflusssystems, sie wird durch den Einsatz einer speicherprogrammierbaren Steuerung (SPS) gesteuert. Üblicherweise werden die Fördereinheiten mit einer festen Geschwindigkeit durch die einzelnen Fördertechnikelemente bewegt. In der Fördertechnik sind i. A. Scanner verbaut mit deren Hilfe die Bewegungen der FE aufgezeichnet werden können. Mit den durch die Scanner erfassten Daten kann der Materialfluss gesteuert werden. [via15]

- **Regal und Regalbediengerät** (rot)

  Regalbediengeräte (RBGs) dienen zur Entnahme von Material aus den Regalen im Lager. RBGs haben eine feste Ein- und Ausgangsleistung und sind durch eine begrenzte Aufnahmekapazität gekennzeichnet.

- **Loop** (blau)

  Ein Loop ist eine Fördertechnikkomponente und dient zur Steuerung des Materialflusses. Seine primäre Funktion ist das Verteilen der Fördereinheiten auf die Kommissionierplätze. Ferner wird durch den Einsatz eines oder mehrerer Loops der gesamte Materialfluss flüssig gehalten.

- **Aufsetzpunkt (grün)**

  Ein Aufsetzpunkt dient zur Einbringung von Fördereinheiten in das automatische Lagersystem.

- **Kommissionierplatz** (gelb)

  An einem Kommissionierplatz (K-Platz) wird die Ware aus den Fördereinheiten entnommen und anschließend zum Versand befördert.

- **Fördereinheit**

  Eine Fördereinheit (FE) ist die kleinste beförderbare Einheit innerhalb eines MFS. In einer Fördereinheit wird die Ware vom Lager zu den K-Plätzen befördert. Jede Fördereinheit ist durch eine eindeutige Nummer gekennzeichnet, so kann das MFS den Zustand einer FE zu jeder Zeit eindeutig feststellen.

- **Transportauftrag**

  Durch die Anforderung eines Behälters aus dem Lager wird dieser nach Entnahme aus dem Lager zur Fördereinheit (FE). Ein Transportauftrag (TA) definiert den Start und das Ziel einer FE. Diese können nach der Erstellung des Transportauftrags nicht mehr geändert werden.

- **Fördertechnikplätze**

  Die einzelnen Fördertechnikplätze, auch Wegpunkte genannt, innerhalb der Fördertechnik sind i. d. R. durch eine eindeutige Adresse, in Form einer Nummer z. B. 4711, gekennzeichnet. Innerhalb eines MFS sind die Fördertechnikplätze mit einem Scanner zur Identifikation der FE mittels eines angebrachten Barcodes ausgerüstet.

- **Speicherprogrammierbare Steuerung (SPS)**

  Durch den Einsatz einer speicherprogrammierbaren Steuerung (SPS) können die Bauteile der Fördertechnik gesteuert werden. Die SPS besteht aus einer Eingabeeinheit, einem Steuerwerk, Programmspeicher, Merker, Zeitgeber, einer Ausgabeeinheit und einer Schnittstelle zum Programmiergerät. Die Funktionalität einer SPS wird von außen durch einen klar definierten Ablauf im Programmspeicher abgelegt. [Her05]
  Zur Steuerung der Fördertechnik verwendet viastore SYSTEMS GmbH i. A. Produkte der Reihe Siemens Simatic S7. [Sie16]

Das Zusammenspiel der Komponenten wird im kommenden Absatz, in Abhängigkeit des Vorgangs bzw. der Aktion, kurz skizziert.

2.1.2 Aktionen in einem Materialflusssystem

In Abhängigkeit der Aktion entstehen unterschiedliche Wege, die eine FE im MFS nehmen kann.

- **Auslagerung**

  Im Lager (rot) werden die Lagereinheiten (LEs) vom Regalbediengerät (RBG) entnommen und über eine Förderstrecke als Fördereinheit (FE) zum Loop (blau) transportiert. Die FEs durchkreisen so lange das Loop, bis das entsprechende Ziel, der Kommissionierplatz (blau) frei ist. Nach Abschluss der Kommissionierung werden die Behälter entweder als Leerbehälter oder Rücklagerung zurück ins Lager gespeist.

- **Rücklagerung**

  Nach der Entnahme der gewünschten Menge werden die FEs über den Aufsetzpunkt (grün) neben den Kommissionierungsplätzen zurück ins Lager geschleust.

- **Einlagerung**

  Über die Aufsetzpunkte (grün) werden neue Leerbehälter als FE in das Lager eingespeist. Die leeren FE fließen ebenfalls über das Loop (blau) in das Lager zurück.

2.1.3 Materialströme

Aus den soeben eingeführten Aktionen lassen sich die folgenden Ströme ableiten:

- **Auslagerstrom**

  Für FEs, die aus dem Lager zu den Kommissionierplätzen fließen.

- **Rücklagerstrom**

  Für FEs, die von den Kommissionierplätzen zurück ins Lager fließen.

- **Einlagerstrom**

  Für neu einzulagernde FEs, die von den Aufsetzpunkten zurück ins Lager fließen.

- **Leerbehälterstrom**

  Für leere FEs, die nach Abschluss der Kommission von den Aufsetzpunkten zurück ins Lager fließen.

2.2 Materialflusssteuerung

Ein Materialflusssystem hat zur Aufgabe, gleichzeitig den Fluss vieler Fördereinheiten effizient zu gestalten, mit dem Ziel den Durchsatz an Fördereinheiten zu optimieren. Der Materialfluss muss getaktet werden, um Kapazitätsengpässe zu vermeiden. Hierfür ist es wichtig, dass alle Engpässe eines MFS bekannt sind, da sie die tatsächliche Geschwindigkeit des gesamten Materialflusses bestimmen. Damit der Materialfluss an Engpässen nicht zum Stocken kommt, müssen Staus vermieden werden. Das Ziel der Materialflusssteuerung ist ein ruhiger, kontinuierlicher Durchlauf von Fördereinheiten. [Dic09]

Abbildung 2-3 zeigt die Unterschiede in der Geschwindigkeit in gleichmäßig getaktetem bzw. geregeltem und ungetaktetem bzw. ungeregeltem Materialfluss.

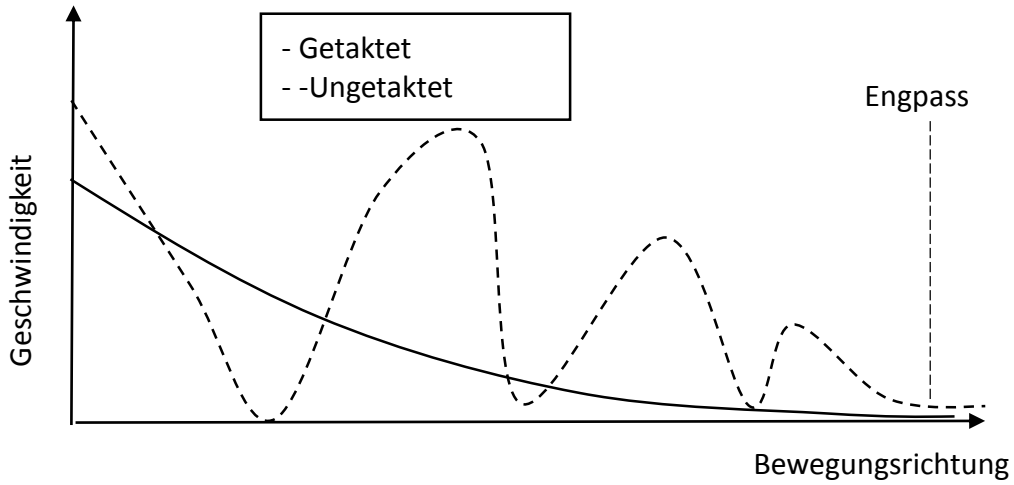

Abbildung 2-3: Verhältnis von Geschwindigkeit zu Bewegungsrichtung [Dic09]

Es ist zu erkennen, dass es bei ungeregeltem bzw. ungetaktetem Materialfluss häufiger zu Staus im Materialfluss kommen kann. Bei einer Regelung des Materialflusses nimmt die Geschwindigkeit des Materialflusses bis zum Erreichen des Engpasses stetig ab.

Im Zuge der Materialflusssteuerung ist es von immenser Bedeutung, dass das System die Information über den Standort einer jeden Fördereinheit zu jeder Zeit abrufen kann, um so zusammen mit dem gegebenen Ziel der FE einen optimalen Materialflussprozess zu gestalten. [Arn09]

2.2.1 Ziele in der Materialflusssteuerung

Wie jeder Prozess wird auch die Materialflusssteuerung betrieben, um gewisse Ziele zu erreichen. Die Ziele der Materialflusssteuerung können Tabelle 2-1 entnommen werden.

Tabelle 2-1: Ziele in der Materialflusssteuerung nach [Gud121]

Ziel	Beispiel
Auslastung	Maximale Auslastung des MFS
Leistung	Maximaler Durchsatz in allen Richtungen Maximaler Durchsatz am Kommissionierplatz
Zeit	Minimale Auftragsdurchlaufzeit in allen Richtungen
Stau	Minimale Warteschlangen
Sicherheit	Minimale Ausfallwahrscheinlichkeit

Da sich, nach [Gud121], diese Ziele nicht vereinen lassen, muss eine Rangfolge für die Ziele definiert werden.

2.3 Materialflussprozess

Moderne Intralogistikanlagen bestehen aus verschiedenen Komponenten. Durch das Zusammenspiel aller Komponenten entsteht ein Materialflussprozess ähnlich eines Netzwerkflusses in vernetzten Systemen. Im Allgemeinen besteht dieser Prozess aus drei Komponenten:

- Quellen (Q)
- Senken (S)
- Vorgängen (V) [Arn09]

Abbildung 2-4 zeigt einen exemplarischen Materialflussprozess auf Basis des Netzwerkflusses.

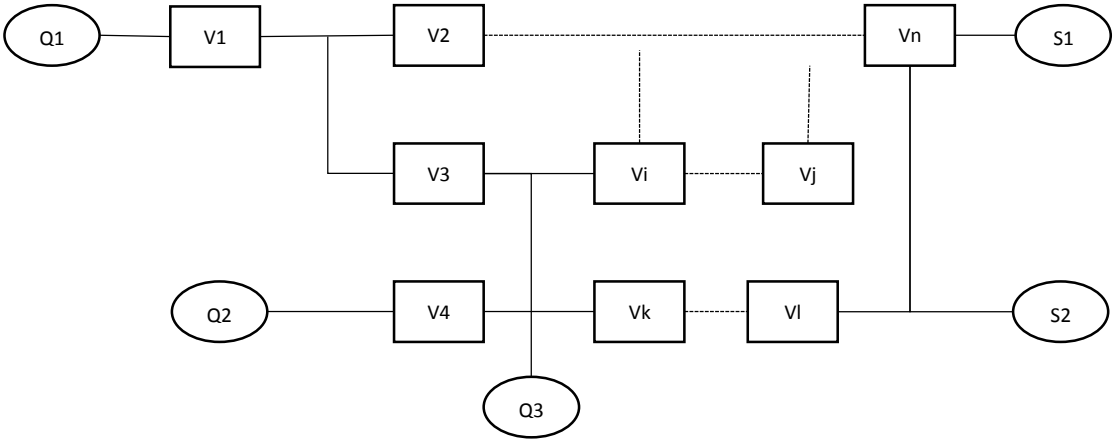

Abbildung 2-4: Netzwerkstruktur des Materialflussprozesses [Arn09]

Das Material innerhalb eines Materialflussprozesses läuft in Fördereinheiten (FE) von einer Quelle über einen oder mehrere Vorgänge hin zu einer Senke. Die Vorgänge innerhalb des Materialflussprozesses können sowohl seriell als auch parallel ablaufen, da z. B. das gleiche Material innerhalb eines Materialflusses verschiedene Vorgänge beanspruchen kann.

Mit Erhöhung der Flexibilität einer Anlage steigt auch der Vernetzungsgrad. Da durch die zu Grunde liegende Netzwerkstruktur Vorfahrtsregeln, auf Basis der Prioritäten der Fördereinheiten, definiert und das Fassungsvermögens der Warteräume des Netzwerks beachtet werden müssen. [Arn09]

2.4 Kennzahlen von Materialflusssystemen

Die Leistung eines MFS wird i. d. R. mit der Hilfe der folgenden zwei Kennzahlen beurteilt.

2.4.1 Durchsatz

Der Durchsatz λ wird durch die Summe n der pro Zeiteinheit (ZE) beförderten FE beschrieben [ten11]:

$$\lambda = \frac{n}{ZE} \tag{2.1}$$

2.4.2 Durchlaufzeit

Die Durchlaufzeit t_{DL} beschreibt die Zeitspanne vom Zeitpunkt der Einspeisung t_E einer FE bis zum Zeitpunkt der Entnahme t_A [Nyh12]:

$$t_{DL} = t_A - t_E \tag{2.2}$$

2.4.3 Füllstand

Der Füllstand bzw. Auslastungsgrad ρ beschreibt das Verhältnis von momentanem Durchsatz λ zu Grenzdurchsatz γ und ist definiert als: [Arn09]

$$\rho = \frac{\lambda}{\gamma} * 100\ \% \tag{2.3}$$

In den nachfolgenden Abschnitten werden in einem ersten Schritt die aktuellen Methoden zur Analyse und Steuerung von MFS vorgestellt. Im Anschluss wird auf Grundlage der in dieser Arbeit betrachteten Referenzanlage ein allgemeiner Ansatz zur Modellierung von MFS vorgestellt. Daraufhin wird der im Verlauf dieser Arbeit erarbeitete Lösungsansatz vorgestellt und in einem ereignisbasierten Simulator auf seine Anwendbarkeit untersucht, abschließend wird ein Integrationsansatz des Lösungsansatzes in die Software viadat vorgestellt.

3 METHODEN ZUR ANALYSE UND STEUERUNG VON MATERIALFLUSSSYSTEMEN

Zur Analyse und Steuerung von intralogistischen MFS haben sich verschiedene Ansätze etabliert. In diesem Kapitel werden die klassischen Ansätze sowie Ansätze aus dem Internet der Dinge und aus dem Bereich des maschinellen Lernens vorgestellt.

3.1 Graphen und Diagramme

Zur systematischen Darstellung von MFS und ihren zugehörigen Prozessen eignen sich die in 2.3 eingeführten Graphen und Diagramme. Sollen in der Analyse der Prozesse auch Aktivitäten oder Nebenläufigkeiten dargestellt werden, so müssen andere Darstellungsmöglichkeiten in Betracht gezogen werden. Die SADT-Methode, Materialflussgraphen und die Petri-Netze [Arn09] werden im Folgenden als klassische Beispiele für die Analyse und Beschreibung von MFS kurz erläutert.

3.1.1 Materialflussgraph

Nach [Arn09] ist Materialflussgraph ein gerichteter Graph G mit:

$$G = (V, E) \tag{3.1}$$

den Knoten:

$$V = \{1,2, \dots n\} \tag{3.2}$$

wobei $|V| = n$ der Anzahl an Elementen im MFS entspricht. Und den Kanten:

$$E = \{p_1, p_2, \dots, p_m\} \tag{3.3}$$

Um die Vorgänge innerhalb eines MFS besser quantifizieren zu können, benötigen die Kanten und Knoten Bewertungen, wie z. B.

- Grenzdurchsätze γ_{ij}
- Betriebliche Durchsätze γ_{ij}
- Allgemeine Kosten der Übergänge c_{ij} [Arn09]

Um auch die Knoten quantifizieren zu können, kann nach [Arn09] der Graph G um die Beschränkung für Knoten, z. B. Kapazitäten β der einzelnen Warteschlangen, erweitert werden. [Arn09]

3.1.2 SADT-Diagramm

Die SADT[1]-Methode stammt ursprünglich aus dem Softwareengineering und ist eine formale Beschreibung von Aktivitäten und den dazu benötigten oder entstehenden Datenflüssen. Durch den Einsatz der SADT-Diagrammsprache können komplexe Sachverhalte einfach und verständlich in Diagrammen dargestellt werden. [Arn09] Abbildung 3-1 zeigt exemplarisch einen Auszug aus einem SADT-Diagramm.

[1] SADT (Engl.) = Structured Analysis and Design Technique

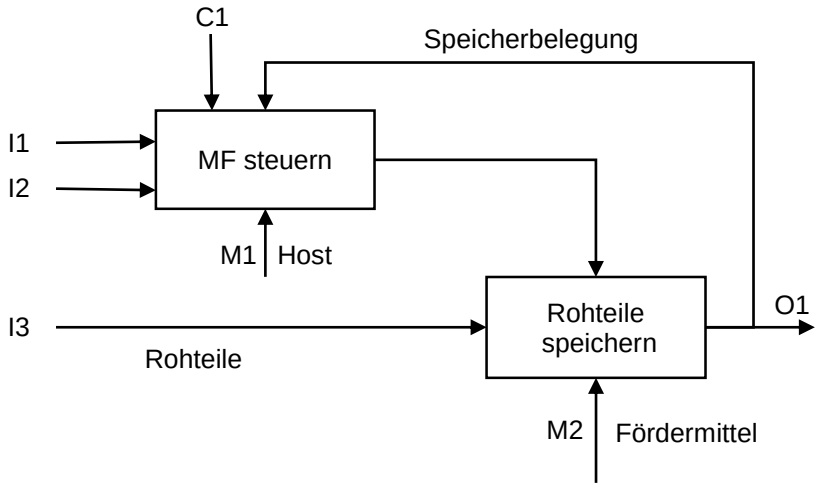

Abbildung 3-1: SADT-Diagramm-Auszug nach [Arn09]

Die in Abbildung 3-1 gezeigten Aktivitäten sind in den Kästen abgebildet. Auf der linken Seite der Kästen stehen die benötigten Daten bzw. Objekte, die zur Durchführung der Aktivität notwendig sind und werden als Input (I) bezeichnet. Die, als Output (O) bezeichneten, Pfeile auf der rechten Seite symbolisieren die durch die Aktivität erzeugten Objekte. Steuerinformationen, sog. Controls (C), werden von oben und die dazu benötigten Mittel (M) von unten an die Aktivität gelegt. [Arn09]

Durch die Verknüpfung der Diagramme aller am MF-Prozess beteiligten Gruppen entsteht nach [Arn09] ein komplettes Diagramm des Materialflusses. Wobei die Outputs des einen Diagramms zu Inputs des anderen werden können.

Ferner sind durch die Darstellung als SADT-Diagramm sowohl der MF-Prozess als auch der dazugehörige Datenfluss gemeinsam abgebildet. [Arn09]

3.1.3 Petri-Netze

Durch die Verwendung von Petri-Netzen können insbesondere die sich ändernden Situationen und Ereignisse innerhalb eines MFS dargestellt werden. Durch die Verwendung der Petri-Netze kann der Entwurf der Steuerung des MFS stark vereinfacht werden. [Arn09]

Petri-Netze sind gerichtete Graphen, deren Knoten aus zwei Teilmengen, den Ereignissen und Situationen, bestehen. Ereignisse, bzw. Transitionen, beschreiben Zeitpunkte im Prozessablauf und werden als Rechtecke dargestellt. Situationen, bzw. Bedingungen, sind zeitverbrauchende Vorgänge oder Zustände und sind als kreisförmige Knoten dargestellt. In Petri-Netzen ist nur die Verbindung von Ereignissen und Situationen gestattet. [Arn09]

Um von einer statischen zu einer dynamischen Prozessbeschreibung zu wechseln, empfiehlt [Arn09] die Verwendung von Marken in den Situationen des Petri-Netzes. Die Marken haben dabei die Funktion einer Schaltung mit fest vorgegebenen Regeln. In Abbildung 3-2 ist der Schaltvorgang abgebildet. Links ist der Ausgangszustand und rechts der Zustand nach dem Schalten dargestellt.

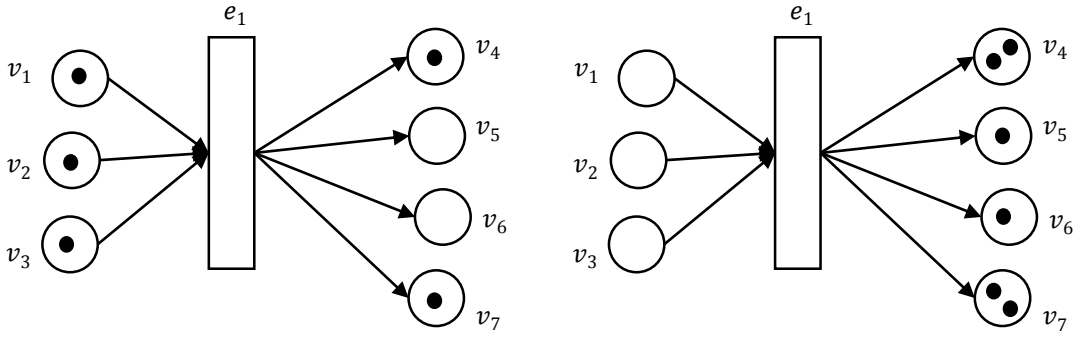

Abbildung 3-2: Transition – Ausgangszustand (links) und Zustand nach der Schaltung (rechts)

Damit der Eintritt eines Ereignisses stattfindet, müssen sämtliche Vorgängersituationen mit einer Marke versehen werden. So wird beim Schaltvorgang bei jeder Vorgängersituation eine Marke entfernt und bei den Nachfolgesituationen eine Marke erzeugt. Um Konflikte in den Schaltungen zu vermeiden, können z. B. zusätzliche Situationen eingeführt werden. [Arn09]

3.2 Simulation und Emulation

Das Hauptwerkzeug zur Beurteilung von MFS ist die Simulation der Anlage. Hierzu gibt es verschiedene Ansätze. Um die Idee hinter einer Simulation zu verstehen, wird hier zuerst der allgemeine Begriff der Simulation erläutert. Im Anschluss werden Methoden zur Simulation von Anlagen vorgestellt.

Unter Simulation wird i. A. die Nachbildung eines Systems mit der Darstellung von dynamischen Prozessen in experimentierbaren Modellen zur Simulierung verstanden. Die so erlangten Resultate können im Anschluss auf die Realität übertragen werden. [Chr13]
Grundlage für eine Simulation ist ein abstraktes Simulationsmodell. Der Simulationsprozess bildet nach [Chr13] den in Abbildung 3-3 abgebildeten Kreislauf.

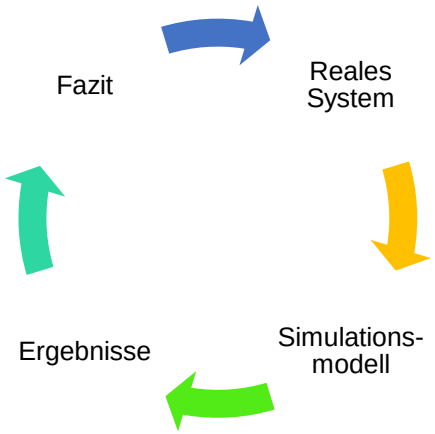

Abbildung 3-3: Simulationskreislauf nach [Chr13]

Zu Beginn der Simulation steht immer das reale System. Aus diesem wird im nächsten Schritt durch Abstraktion ein Simulationsmodell erstellt. Auf Basis des Simulationsmodells werden Experimente durchgeführt. Die so gewonnenen Ergebnisse werden ausgewertet, interpretiert und auf das Ausgangssystem übertragen. Der hier abgebildete Kreislauf wird so oft durchlaufen, bis die Ergebnisse den Vorgaben entsprechen.

Die Simulation von Systemen wird in diskrete und kontinuierliche Simulation unterschieden. [Chr13]

Die Eigenschaften der beiden Simulationsarten nach [Chr13] können Tabelle 3-1 entnommen werden.

Tabelle 3-1: Vergleich diskrete und kontinuierliche Simulation

Simulation / Eigenschaft	Diskret	Kontinuierlich
Zeitabstände	veränderlich	äquidistant
Ablauf	ereignisorientiert	stetige Veränderung in der Zeit
Ereignisse	abhängig von vorangegangen Ereignissen	kontinuierlich
Systemzustände	Abhängig von vorangegangen Ereignissen und Eingabeparametern	Abhängig von Eingabeparametern

Die Unterscheidung von diskreter und kontinuierlicher Simulation wird hier nur der Vollständigkeit halber aufgeführt. Die im Folgenden beschriebenen numerischen Simulationen beruhen alle auf der Simulation von diskreten Ereignissen. [Chr13]

3.2.1 Diskrete numerische Simulation

Im Kontext der numerischen Simulation kann durch den Einsatz von Simulationswerkzeugen das Verhalten eines Prozesses simuliert und somit Gesetzmäßigkeiten abgeleitet werden. Die diskrete numerische Simulation wird dabei unterschieden in:

- Ereignisorientierte
- Prozessorientierte
- Periodenorientierte Simulation [Hed13]

Zur Durchführung von numerischen Simulationen gibt es verschiedene Werkzeuge wie z. B. MATLAB [Mat15] oder Analytica [Lum15]. Auf die genaue Verwendung dieser Werkzeuge wird hier jedoch nicht eingegangen.

Ereignisorientierte Simulation

Bei der ereignisorientierten Simulation werden alle Ereignisse, die bei der Durchführung eines diskreten Prozesses auftreten, simuliert. Die Zeit wird nach [Mar03] als abzählbare Menge $\mathcal{T} = \{t_1, t_2, \dots, t_n\}$ abgebildet, somit entspricht jedem Ereignis eineindeutig ein $t \in \mathcal{T}$.
Bei der Simulation eines bestimmten Ereignisses wird die von diesem Ereignis abhängige Ereignisroutine ausgeführt.

Die Ereignisroutine ist nach [Hed13] ein Quellcode als Teil der Simulationssoftware, der die folgenden Aufgaben bearbeitet:

1. Neuen Zustand berechnen
2. Planung neuer zukünftiger Ereignisse
3. Statistische Auswertung

Ereignisroutinen werden ausschließlich für unabhängige Ereignisse erstellt. In der aktuellen Ereignisroutine werden alle Berechnungen, die für die abhängigen Ereignisse notwendig sind, bereits ausgeführt. Abbildung 3-4 skizziert den Ablauf einer diskreten Simulation.

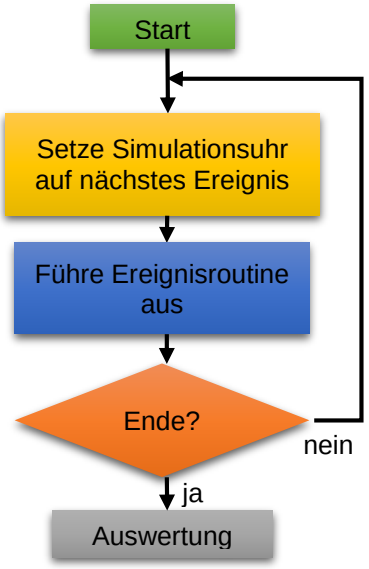

Abbildung 3-4: Ablauf der diskreten Simulation [Mar03]

Beim Start der Simulation wird die Simulation mit dem ersten Ereignis $\mathcal{T} = \{t_0\}$ begonnen. Für jeden Ereigniszeitpunkt $t \in \mathcal{T}$ werden durch das Ausführen der Ereignisroutine für ein Objekt

$o \in \mathcal{O}$ dessen Zustand $\mathcal{X}_t(\mathcal{A}(o))$ neu berechnet und ggf. neue Ereignisse zur Menge der Ereignisse $\mathcal{T}$ hinzugefügt. Ist die Simulation noch nicht beendet, wird die Simulationsuhr mit $t = \min(\mathcal{T})$ auf das nächste Ereignis gesetzt. Weiter gibt [Mar03] an, dass durch den Einsatz der diskreten Simulation eine beliebige genaue Approximation des Zielsystems erreicht werden kann. Jedoch steigt mit der Genauigkeit auch der Berechnungsaufwand. Häufig wird die diskrete Simulation für den Leistungsnachweis eines Systems oder für Prognosen zu Echtzeit verwendet. [Mar03]

Prozessorientierte Simulation

Die prozessorientierte Simulation beruht auf dem Ansatz der ereignisorientierten Simulation, bei der Prozesse durch gewisse Ereignisse ausgelöst werden. Im Gegensatz zur ereignisgesteuerten Simulation werden hier jedoch zusätzlich zu den zukünftigen Ereignissen auch die aktuellen, eventuell parallel laufenden, Ereignisse betrachtet. Hierbei werden die Objekte als dynamische Einheiten, die sich durch ein System bewegen, betrachtet. Die Wege des einzelnen Objekts werden als Prozess abgebildet, wodurch es zu quasi-parallelen Prozessen kommen kann. Um eine Parallelisierung zu verhindern, werden Regeln definiert, die die Ausführung der Prozesse durch die Bildung von Warteschlangen sequenzieren. Dafür werden für jeden Prozess die folgenden Zustände definiert:

- **Aktiv mit Kontrolle**

  Der Prozess erhält die Kontrolle über die Simulation, wenn sein nächstes Ereignis das global nächste ist und er in der aktiven Phase ist. Es wird der Zustand aller sich ändernden Objekte berechnet und Folgeereignisse werden geplant.

- **Aktiv ohne Kontrolle**

  Das Ereignis ist schon geplant, der Prozess wartet aber noch auf die Abarbeitung der Ereignisse vorhergehender Prozesse.

- **Passiv**

  Der Prozess wartet auf das ihn auslösende Ereignis. [Hed13]

Periodenorientierte Simulation

Bei der periodenorientierten Simulation werden die Zustandsänderungen eines Systems erst nach gegebenen äquidistanten Zeitintervallen Δt betrachtet. Die Simulationsuhr wird nach Ausführung aller Berechnungen zu einem Zeitpunkt t_i auf

$$t_{i+1} = t_i + \Delta t \tag{3.4}$$

gestellt. Im Gegensatz zur ereignisorientierten Simulation kann die Zustandsänderung auch unabhängig vom Ereignis betrachtet werden, die Zustandsänderung folgt in der Regel durch die Vorgabe eines konkreten Algorithmus. [Hed13]

Ferner gibt [Hed13] an, dass zur Simulierung von „realen“ zufällig verteilten Ereignissen, diese mithilfe von stochastisch verteilten Zufallsvariablen beschrieben werden können.

3.2.2 3D-Simulationssoftware

Durch den Einsatz von 3D-Simulationssoftware kann der Materialfluss in Echtzeit simuliert und analysiert werden. Hierzu muss jedoch das gesamte Anlagenbild in die virtuelle Umgebung übertragen werden. Die meisten Werkzeuge liefern hierzu ein sog. Baukastensystem, mit dem die einzelnen Bestandteile des Materialflusssystems abgebildet werden können. Hier können die einzelnen Parameter der jeweiligen Objekte eingestellt werden.

Durch den Einsatz von Simulationssoftware können jedoch meist schon sehr detaillierte Aussagen über Engpässe, Durchsätze und Auswirkungen von Störungen getroffen werden. Typische Beispiel für Simulationssoftware sind Sim3D von Emulate 3D [Dem15], aber auch das von [Ben12] genannte ISG virtuos.

3.2.3 Emulation

[Gün10] beschreibt die Emulation als Spezialfall der Simulation, erweitert um reale Funktionskomponenten. Im Gegensatz zur Simulation wird die Emulation jedoch zur Voraussage des Verhaltens eines bestimmten Systems unter Realbedingungen verwendet.

Meist wird das Simulationsmodell um die Funktionalitäten des zu verwendenden Steuerungssystems erweitert. Durch diese Erweiterung wird die Laufzeit der Emulation meist auf die Echtzeit beschränkt, da z. B. Taktzeiten von Schaltungen nicht verkürzt werden können.

Die Emulation wird meist als Werkzeug zur Entwicklung und Inbetriebnahme von Anlagen eingesetzt, sie ersetzt damit kostspielige Tests der Anlage im Realbetrieb. Die Emulation kann im Gegensatz zur Simulation nur durch den Einsatz von Emulationssoftware wie z. B. Emulate3D von Emulate 3D [Dem15] realisiert werden.

3.3 Internet der Dinge

Der Ansatz des Internets der Dinge beschreibt im Wesentlichen die vollständige Vernetzung aller Komponenten und den Austausch ihrer Kontextinformationen über das Internet. [Bau14] Ferner agieren die einzelnen Objekte des Systems als eigenständige Einheiten und tauschen ihre Daten in Echtzeit aus. Die benötigten Daten werden lokal gespeichert und die Entscheidungen vor Ort gefällt. [Gün10]

Auf Basis des Internets der Dinge schlägt [Gün10] die Verwendung von Agentensystemen zur Steuerung von intralogistischen Materialflüssen vor. Darüber hinaus überträgt er die in 3.1 beschriebenen Simulations- und Emulationsverfahren auf das Internet der Dinge.

3.3.1 Agentensysteme

Wie eingangs beschrieben, eignen sich Agentensysteme zur Steuerung von intralogistischen MFS. Zur Realisierung eines Agentensystems werden die Bestandteile eines MFS entsprechend ihrer Eigenschaften und Fähigkeiten in Funktionseinheiten, sog. Entitäten, zerlegt. Entitäten sind die kleinste nicht mehr weiter zerlegbare Einheit und besitzen die folgenden drei Grundfunktionen:

- Kommunikation mit anderen Entitäten
- Mitteilung eigener Fähigkeiten und Eigenschaften
- Administration des eigenen Zustands und Abschicken von Rück- und Statusmeldungen [Gün10]

Je nach Funktionalität der entsprechenden Entität kann diese noch um weitere Fähigkeiten erweitert werden. Jeder Entität wird zur Bearbeitung ihrer Aufgabe ein Agent zugeordnet. Tabelle 3-2 zeigt die Entitäten eines Agentensystems für ein MFS mit exemplarischen Eigenschaften.

Tabelle 3-2: Entitäten eines Agentensystems nach [Gün10]

Entität	Allgemeine Eigenschaften	Beispiel
Förder-einheit	Kleinste einzeln bewegbare Menge im MFS Identifikation mittels Autoidenttechnologien	Behälter, Palette
Modul	Autonom agierende Fördertechnikelemente erfüllen logistische Funktionen der Fördereinheiten an Entscheidungspunkten	Regalbediengeräte, Rollenbahn
Software-dienst	Reine Software Dienen zur Visualisierung, Optimierung, etc. Oft Schnittstellen zu anderen Programmen	Visualisierungssoftware, Schnittstellen zu anderen Systemen

Die hier eingeführten Module können gemäß den in Abschnitt 2.1 eingeführten Grundbestandteilen eines MFS klassifiziert werden.
Agentensysteme kommen mittlerweile schon bei der dezentralen Ermittlung des kürzesten Weges unter Verwendung des Dijkstra-Algorithmus[2] zum Einsatz. Hier wird bei jedem Entscheidungspunkt die Verfügbarkeit der aktuellen Route geprüft und ggf. eine neue Route unter Verwendung des Dijkstra-Algorithmus berechnet. [Gün10] [Göh13]

3.3.2 Simulation und Emulation

[Gün10] schlägt vor, den klassischen Ansatz für die Simulation und Emulation eines MFS so zu erweitern, dass die im vorherigen Abschnitt beschriebenen Agenten in das Materialflussmodell integriert werden können. Hierzu muss das diskrete Simulationsmodell so erweitert werden, dass die Antwortzeiten, Nachrichtenverluste, etc. der Agenten simuliert werden können.

Ferner schlägt [Gün10] vor, die oft sehr komplexen und großen Simulationsmodelle zu partitionieren und auf mehreren Rechnern parallel auszuführen, um so die Ausführungsgeschwindigkeit zu erhöhen.

Um die Emulation durch den Einsatz von Agentensystem zu beschleunigen wird ein standardisierter Baukasten eingeführt. Die Einführung eines solchen Baukastens erleichtert die Gestaltung eines Simulationsmodells. Hierzu muss für jede Funktion des Systems ein entsprechender Baukasten in Form eines Steuerungsagenten vorhanden sein.

Des Weiteren wird das Konzept so erweitert, dass für jeden Steuerungsagent ein Emulatoragent vorhanden ist, der die Funktion auf Maschinenebene abbildet. Dieses Vorgehen legt die Erstellung eines Baukastens für Emulationsagenten nahe. [Gün10]
Somit kann das Verhalten eines Systems komplett nachgebildet werden.

3.3.3 Cloudbasierte Materialflusssteuerung

Um den steigenden Anforderungen an MFS-Systeme zu begegnen, kann die Steuerung modularisiert und als Cloud-Service zur Verfügung gestellt werden. Durch die Verlagerung der Steuerungsfunktion entfällt die Notwendigkeit der Verwendung von Hardwareschnittstellen zur Kommunikation zwischen den einzelnen Steuerungselementen. [Bau14]

Auf Basis des Software-as-a-Service-Konzepts [Clo15] der Cloud-Technologie kann die Steuerung kontinuierlich an die jeweiligen Bedürfnisse angepasst werden.
[Bau14] schlägt zur Umsetzung einer cloudbasierten Materialflusssteuerung den in Abbildung 3-5 skizzierten Ansatz vor.

[2] Zur genauen Funktionsweise des Dijkstra-Algorithmus vgl. [Geh07]

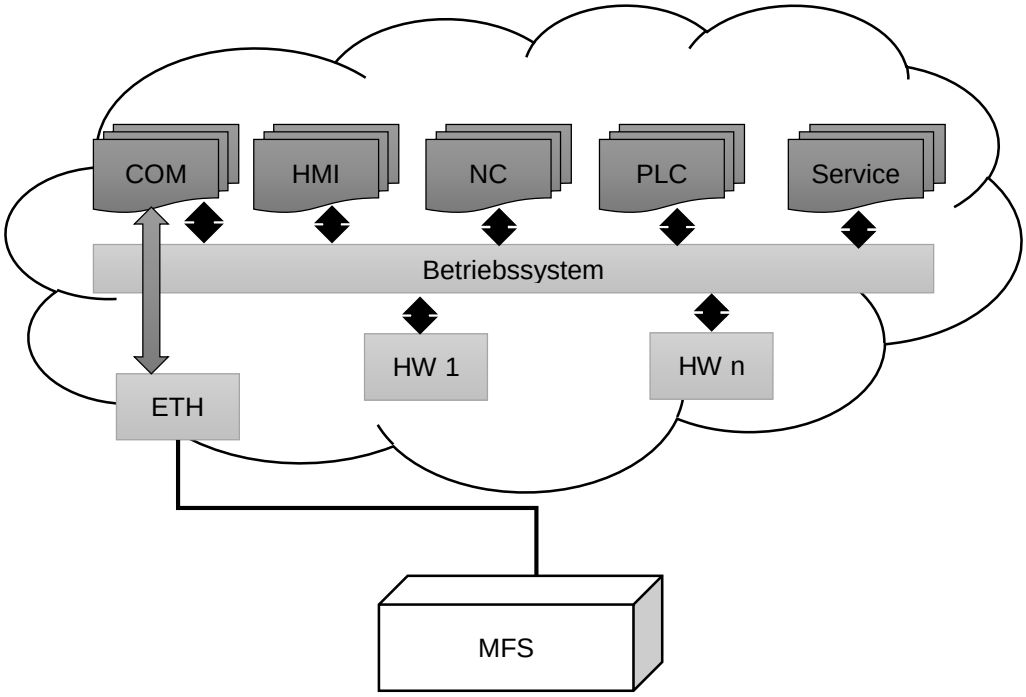

Abbildung 3-5: Cloudbasierte Materialflusssteuerung

In dem hier gezeigten Ansatz werden alle zur Steuerung benötigten Module wie COM, HMI, NC, PC und zusätzliche Services in die Cloud verlagert. Die Aktoren und Sensoren des MFS sind über eine Netzwerkschnittstelle mit der Cloud verbunden und können die entsprechenden Module bei Bedarf in Echtzeit abrufen. Die einzelnen Module werden über ein in der Cloud vorhandenes Betriebssystem gruppiert. Die Steuerung über ein Betriebssystem ermöglicht auch die mehrmalige Instanziierung eines gleichen Moduls. Das Betriebssystem ist auch für die Skalierung der Rechenleistung verantwortlich. [Bau14]

Des Weiteren gibt [Bau14] die folgenden durch den Einsatz von Cloud-Technologie entstehenden Vorteile an:

- Skalierbare Steuerung
- Längere Verfügbarkeit der Steuerungsplattform
- Steigerung der Flexibilität
- Effizientere Analyse und Simulation der Systeme
- Erhöhung der Datensicherheit
- Erhöhung der Verfügbarkeit
- Bessere Darstellung der Informationen

3.4 Maschinelles Lernen

Um qualifizierte Aussagen über den Betriebszustand einer Anlage, insbesondere ihrer kritischen Komponenten, treffen zu können, müssen die Betriebsdaten der Komponenten gesammelt und ausgewertet werden. Hierzu z. B. können durch den Einsatz von Algorithmen aus dem Bereich des Datamining wie z. B. Predictive Analytics Aussagen über den Zustand einer Anlage getroffen werden. [Alp08]

3.4.1 Datamining

Durch den Einsatz von Dataminingtechniken kann mittels der Erkennung von Mustern im Verhalten einer Komponente, wie z. B. der Temperaturveränderung, eine Aussage über den Zustand der Komponente getroffen werden, Wartungen werden planbar und die Ausfälle von Anlagen minimiert, da Ersatzteile schon im Voraus bestellt werden können. [Bau14]

So kann z. B. der von [Alz15] eingeführte Ansatz zur Analyse von Einschreibungen in Kursen mithilfe von Assoziationsregeln problemlos auf die Analyse des Betriebszustandes und der

Voraussage von Ausfällen übertragen werden. So können in der Analyse und Steuerung von MFS auch die von [Alz15] vorgestellten Methoden zur Mustererkennung in großen Datenmengen angewandt werden.

Um den benötigten Datenstrom zu reduzieren, eignet sich das Vorgehen mittels Big Data.

3.4.2 Big Data

Die für die im Bereich Predictive Analytics benötigten Datenmengen wachsen exponentiell mit dem Betrieb der Anlage an. Um die auszuwertende Datenmenge zu reduzieren, werden die Daten bezüglich der sog. „3V“ (Volume, Variety, Velocity) verdichtet, d. h. es werden „Ausreißer“ in den Daten gesucht. Um die Verdichtung effizient zu betreiben, wird die komplexe Ereignisanalyse (CEP) eingesetzt. [Bau14]

Mittels der komplexen Ereignisanalyse werden die Betriebsdaten verdichtet, um relevante Informationen zu gewinnen. Hierzu werden die folgenden Funktionen auf den Betriebsdaten ausgeführt:

- Identifizierung von
  - Ausreißern
  - Trends
- Bildung von
  - Summen
  - Minima
  - Maxima
  - Durchschnittswerten über festgelegte Zeiteinheiten

und ausschließlich das Ergebnis dieser Funktionen weitergeleitet. Auf Basis dieser Daten werden dann Aussagen über das Verhalten von Maschinen bzw. Komponenten getroffen. [Bau14]

In den folgenden Kapiteln wird die in dieser Arbeit vorliegende Referenzanlange analysiert und auf Grundlage dieser Analyse ein allgemeiner Modellierungsansatz für MFS vorgestellt. Anschließend wird der im Verlauf dieser Arbeit erarbeitete Lösungsansatz erläutert und in einem ereignisbasiertem Simulator umgesetzt und auf seine Anwendbarkeit zur Optimierung von MFS untersucht. Abschließend wird ein Vorschlag zur Integration des erarbeiteten Lösungsansatzes in die Software viadat vorgestellt.

4 Analyse der Referenzanlage und Lösungsansatz

Zur Identifikation der Einflussparameter auf den Materialfluss wird eine Referenzanlage der Firma viastore SYSTEMS GmbH [via152] auf mögliche Optimierungsmöglichkeiten untersucht. Hierzu werden die ausschlaggebenden Bestandteile der Referenzanlage einzeln erläutert und danach im Gesamtkontext der Referenzanlage dargestellt. Im Anschluss werden die aus der Referenzanlage abgeleiteten Modellelemente vorgestellt. Daraufhin werden die Modellelemente im Kontext der in Abschnitt 2.1 eingeführten Materialflüsse betrachtet und die Referenzanlage als daraus entstehendes Gesamtmodell dargestellt. Abschließend wird auf Basis der Analyse des vorliegenden Modells der zugrunde liegende Lösungsansatz grob skizziert.

4.1 Analyse der Referenzanlage

Das zugrunde liegende Schaubild der Referenzanlage der Firma viastore SYSTEMS GmbH besteht aus drei Auslagerbereichen, die über eine Fördertechnik mit den entsprechenden Entnahmezonen verbunden sind. Des Weiteren besitzt die Referenzanlage einen Wareneingang und Aufnahmeplätze für Leerbehälter in festgelegten Bereichen.

Die Funktionalität sowie der Ablauf der einzelnen Bereiche wird im Kommenden ausgehend von Abbildung 4-1 erläutert.

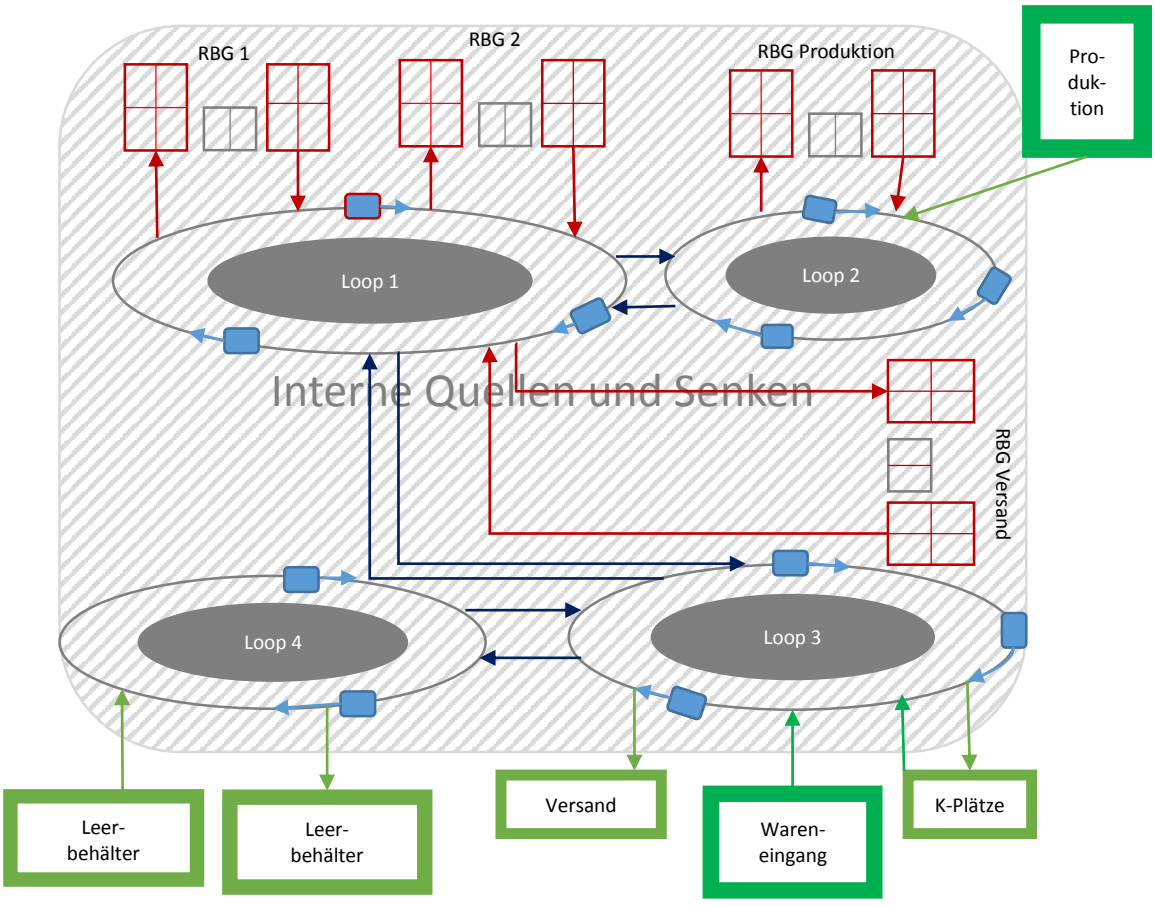

Abbildung 4-1: Schematische Darstellung der Referenzanlage

Die Quellen und Senken eines MFS werden in interne (rot) und externe (grün) Quellen und Senken unterschieden. Interne Quellen und Senken können als eine Art Black-Box interpretiert werden, deren Ein- und Ausgangsleistung nicht optimiert werden kann. Die Leistung von externen Quellen und Senken hingegen kann optimiert werden.

Auf eine detaillierte Darstellung der in 2.1 vorgestellten Fördertechnikplätze wird hier aus Übersichtsgründen verzichtet. Abbildung 4-2 zeigt zur Veranschaulichung die in Loop 1 vorhandenen Plätze der Fördertechnik.

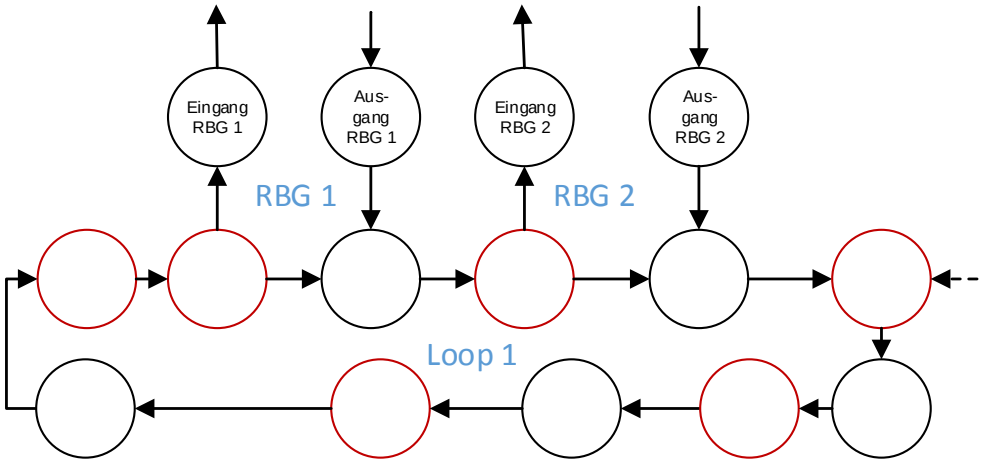

Abbildung 4-2: Auszug aus dem Gesamtschaubild von Loop 1

Die roten Knoten symbolisieren einen Fördertechnikplatz mit Scanner zur Erfassung von FEs und Rückmeldung der erfassten Scanner-Daten an die Software viadat, schwarze Knoten repräsentieren Knoten innerhalb der Fördertechnik ohne Scanner.

Im obigen Beispiel fließt eine FE vom Ausgang eines RBG über die entsprechenden Knoten innerhalb der Fördertechnik zum Zielknoten. Bei den rot markierten Knoten wird jeweils eine Ankunftsmeldung von der Fördertechnik an die Software viadat geschickt.

4.1.1 Bestandteile und Besonderheiten der Referenzanlage

Im Folgenden werden die Bestandteile und Besonderheiten der Referenzanlage erläutert. Zu allererst werden die Eigenschaften der Quellen und Senken erläutert und im Anschluss die Besonderheiten beschrieben. Alle angegebenen Leistungen sowie Füllstandsbeschränkungen wurden [via155] und [via154] entnommen.

4.1.2 Eingänge

Die Eingänge, auch Quellen genannt, sind

- das RBG 1
- das RBG 2
- das RBG „Produktion“
- das RBG „Versand“
- der Wareneingang
- die Aufsetzpunkte für Leerbehälter und Rücktransport an den K-Plätzen und
- der Aufsetzpunkt für LEs aus der Produktion.

Durch die oben aufgeführten Quellen gelangen neue FEs in das MFS. Die Leistung der einzelnen Quellen kann Tabelle 4-1 entnommen werden. [via152]

Tabelle 4-1: Maximale betriebliche Leistung der Quellen

Quelle	Leistung
RBG 1	112,5 FE/h
RBG 2	112,5 FE/h
RBG „Produktion“	102 FE/h
RBG „Versand“	212 FE/h
Wareneingang	76 FE/h
Aufsetzpunkt K-Platz 1 und 2	100 FE/h
Aufsetzpunkt K-Platz 3 und 4	100 FE/h
Zwischenlagerplatz K-Platz 1	28 FE/h
Zwischenlagerplatz K-Platz 2	28 FE/h
Zwischenlagerplatz K-Platz 3	28 FE/h
Zwischenlagerplatz K-Platz 4	28 FE/h

Die in Tabelle 4-1 aufgeführten Leistungen beschreiben die Frequenz, mit der FEs in das MFS eingeschleust werden können. Die Quellen werden in interne Quellen z. B. RBG 1 und externe Quellen z. B. Wareneingang unterschieden. Externe Quellen zeichnen sich durch eine Interaktion mit einem Bediener innerhalb des MFS aus. Interne Quelle sind i. d. R. automatische Einheiten z. B. Regalbediengeräte mit einer fixen betrieblichen Ausgangsleistung.

4.1.3 Ausgänge

Die Ausgänge, auch Senken genannt, der Referenzanlage sind die K-Plätze, mit insgesamt vier Ausgängen, und die Versandplätze, mit insgesamt 20 Plätzen. Tabelle 4-2 zeigt die Leistung der Senken der Referenzanlage.

Tabelle 4-2: Maximale betriebliche Leistung der Senken

Senke	Leistung
K-Platz 1	50 FE/h
K-Platz 2	50 FE/h
K-Platz 3	50 FE/h
K-Platz 4	50 FE/h
RBG 1	141,5 FE/h
RBG 2	141,5 FE/h
RBG „Produktion“	116 FE/h
RBG „Versand“	331 FE/h
Verpackungsplatz 1 bis 7	jeweils 8 FE/h
Verpackungsplatz 8 bis 10	jeweils 5 FE/h

Die Leistung an den entsprechenden Senken hängt einzig von der Leistung der in Abschnitt 2.1 beschriebenen Fördertechnik ab. Wie auch die Quellen werden die Senken ebenfalls in externe und interne Senken unterschieden. Die Ausgangsleistung von externen Senken kann nicht optimiert werden, da sie durch die entsprechende Geräteleistung beschränkt ist. Es können auch hier ausschließlich die Leistungen externer Senken in Abhängigkeit der Leistung der entsprechenden Quellenleistung optimiert werden.

Die hier aufgeführten Leistungen der Quellen und Senken gelten für die spätere Betrachtung des Modells als die zu erreichenden Zielvorgaben.

4.1.4 Leistungsbeschränkende Füllmengen

Für die letzten Abschnitte von Loop zu RBG bzw. Loop zu K-Platz gelten die in Tabelle 4-3 aufgelisteten maximalen Füllmengen.

Tabelle 4-3: Materialflusselemente und ihre Füllmengen

Materialflusselement	Füllmenge
RBG 1	Max. 7 FEs eingehend und ausgehend
RBG 2	Max. 7 FEs eingehend und ausgehend
RBG „Produktion“	Max. 7 FEs eingehend und ausgehend
K-Platz 1	Max. 7 FEs eingehend
K-Platz 1 und 2	Max. 7 FEs ausgehend
K-Platz 2	Max. 7 FEs eingehend
K-Platz 3	Max. 7 FEs eingehend
K-Platz 3 und 4	Max. 7 FEs ausgehend
K-Platz 4	Max. 7 FEs eingehend
Loop 1	Max. 28 FEs gleichzeitig
Loop 2	Max. 18 FEs gleichzeitig
Loop 3	Max. 28 FEs gleichzeitig
Loop 4	Keine Beschränkung vorhanden

Durch die maximalen Staukapazitäten der einzelnen RBGs und K-Plätze ergibt sich eine Einschränkung der Leistung der Referenzanlage. So kann z. B. ein Stau in allen K-Plätzen einen Stau in den vorhergehenden Loops, im Extremfall in den RBGs, verursachen und das komplette MFS zum Erliegen bringen.

4.1.5 Auftragsreine Transportaufträge

Ein Kundenauftrag kann aus beliebig vielen Positionen bestehen. Jede Position generiert eine Warenanforderung im Lager, die aus mehreren Transportaufträgen bestehen kann. Um eine Vermischung von verschiedenen Kundenaufträgen bzw. Positionen zu vermeiden, müssen Transportaufträge auftragsrein bearbeitet werden, d. h. sobald eine FE mit einer Auftragsnummer einen K-Platz betritt, ist dieser Auftrag, bis zu seiner Fertigstellung, fest mit dem K-Platz verknüpft und es kann kein anderer Transportauftrag an diesem K-Platz bearbeitet werden.

4.1.6 Materialflüsse

Um das Geschehen und die Abhängigkeiten innerhalb eines komplexen Materialflusses zu verstehen, ist es notwendig, den Materialfluss in die in 2.1 eingeführten Materialströme zu zerlegen. Die auftretenden Ströme sowie ihre jeweiligen Abhängigkeiten werden in den kommenden Abschnitten erläutert.

Auslagerfluss

Abbildung 4-3 zeigt die schematische Darstellung des Auslagerflusses der Referenzanlage.

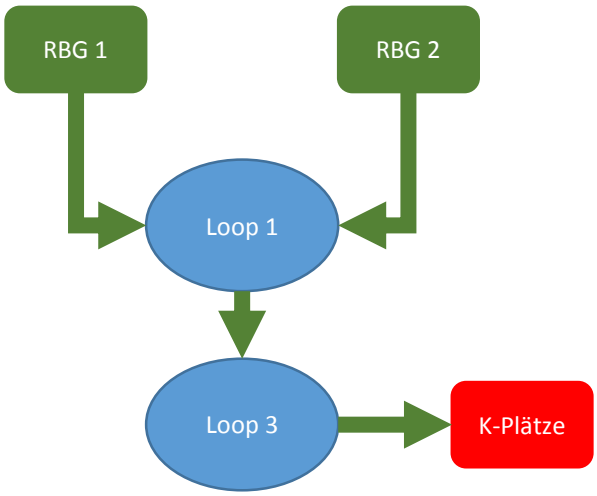

Abbildung 4-3: Schematische Darstellung des Auslagerflusses

Die FEs aus den RBG 1 und RBG 2 (grün) fließen über Loop 1 und Loop 3 (blau) zu den Kommissionierplätzen (K-Plätzen, rot). Nach Ankunft an den K-Plätzen wird der Kommissioniervorgang gestartet und die FEs bleiben bis zum Abschluss des Kommissioniervorgangs durch den Bediener im K-Platz.

Rücklagerfluss

Abbildung 4-4 zeigt den Ablauf des Rücklagerflusses in der Referenzanlage.

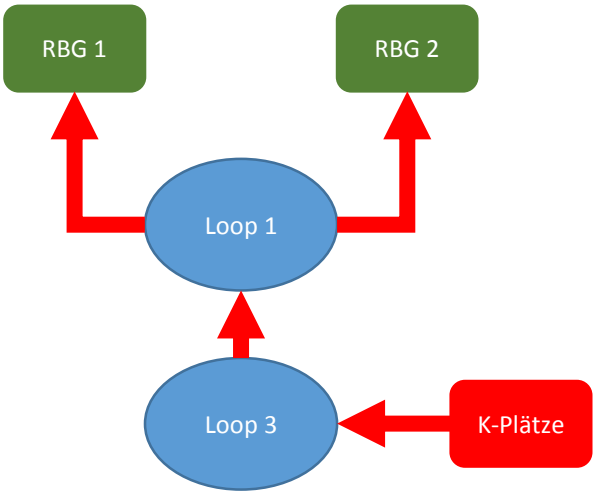

Abbildung 4-4: Schematische Darstellung des Rücklagerflusses

Nach Abschluss der Kommission werden nicht leere FEs von den K-Plätzen über Loop 3 und Loop 1 zurück zum entsprechenden RBG befördert. Von den RBGs werden sie am Ziellagerplatz eingelagert.

Umlagerflüsse

Durch den Einsatz der RBG „Versand“ und RBG „Produktion“ ergeben sich die in Abbildung 4-5 dargestellten roten und grünen Umlagerflüsse.

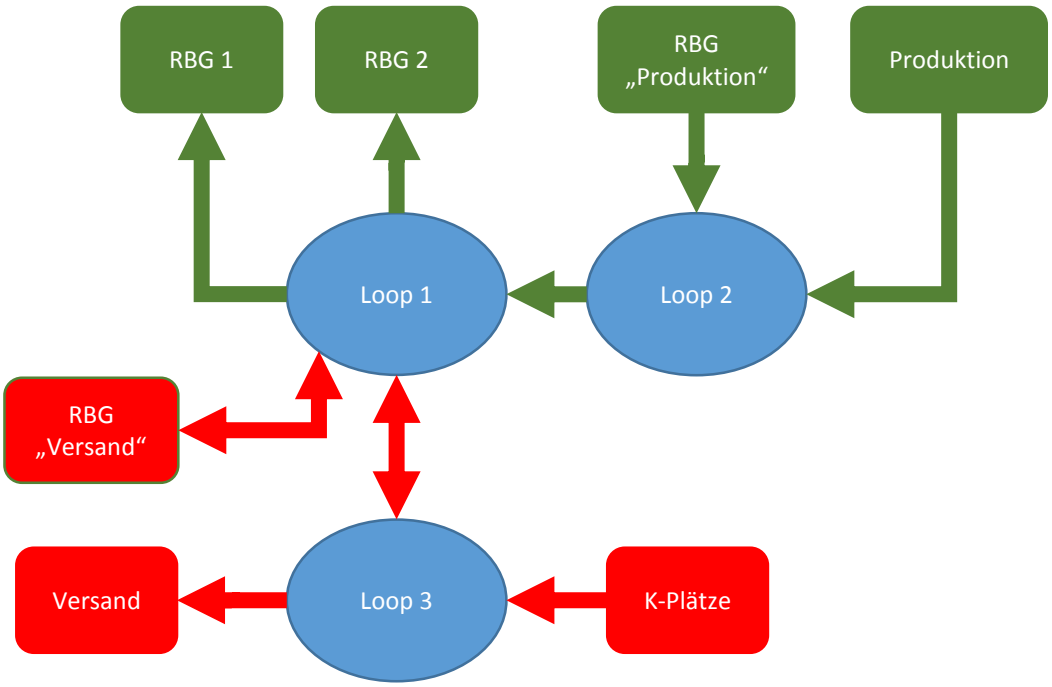

Abbildung 4-5: Schematische Darstellung der Umlagerflüsse

- **Roter Umlagerfluss**

  Die FEs werden nach Abschluss des Kommissioniervorgangs von den K-Plätzen über Loop 3 und Loop 1 zum RBG „Versand“ transportiert. Ist der Kommissionierauftrag abgeschlossen, werden die entsprechenden FEs vom RBG „Versand“ über Loop 1 und Loop 3 zu den Versandplätzen befördert.

- **Grüner Umlagerfluss**

  Die fertig produzierten Waren werden in FEs aus der Produktion über Loop 2 und Loop 1 zu den RBG 1 und RBG 2 ins Lager befördert.

Einlagerfluss

In Abbildung 4-6 ist der Einlagerfluss abgebildet. Neue Ware wird in FEs über Loop 3 und Loop 1 zu den RBG 1 und 2 oder über Loop 3, Loop 1 und Loop 2 zum RBG „Produktion“ befördert.

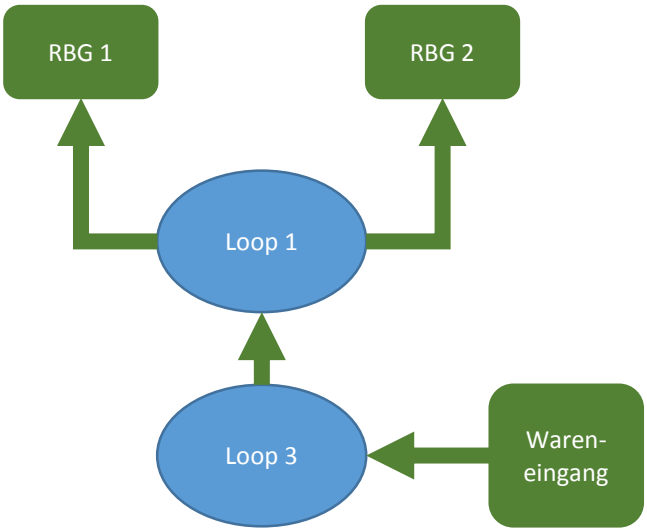

Abbildung 4-6: Schematische Darstellung des Einlagerflusses

Leerbehälterfluss

In Abbildung 4-7 ist der Leerbehälterfluss der Referenzanlage dargestellt. Leere FEs werden über eine spezielle Bahn zu Loop 4 befördert, von wo aus sie zurück zu den RBG 1 und 2 befördert werden. Ebenso wird über den Leerbehälterausgang an Loop 4 der Wareneingang kontinuierlich mit leeren Behältern versorgt.

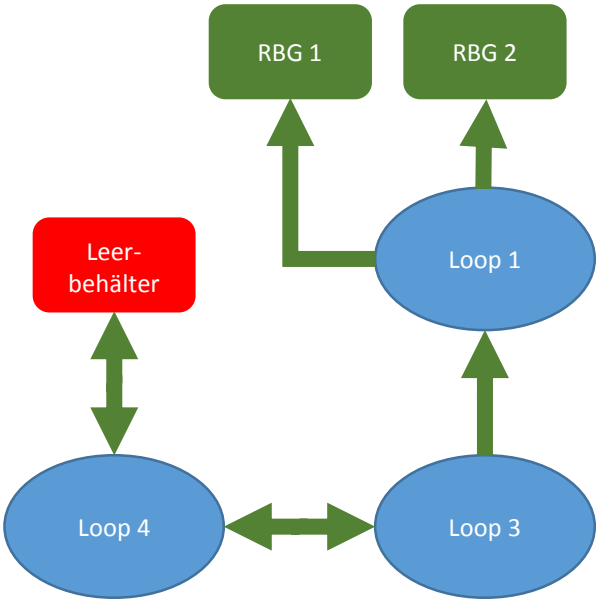

Abbildung 4-7: Schematische Darstellung des Leerbehälterflusses

Für einen besseren Überblick über die vorhandene Problematik wird zuerst der momentane Optimierungsprozess beschrieben. Im Anschluss wird das Modell der Referenzanlage eingeführt. Danach wird ein grober Lösungsansatz skizziert.

4.2 Modellierungsansatz der Referenzanlage

Durch die Analyse der Referenzanlage ergab sich ein Ansatz zur Darstellung des Materialflusses innerhalb eines komplexen MFS. So kann der Fluss von FEs innerhalb eines MFS als der in 3.1 eingeführte Materialflussgraph von einer Quelle hin zu einer oder mehreren Senken dargestellt werden. Der klassische Materialflussgraph (vgl. Abbildung 4-2) wird hierbei in einer leicht

abgewandelten Form betrachtet. Die Knoten innerhalb des Graphs werden als Materialflusselemente bezeichnet und die Kanten symbolisieren den Übergang von einem Materialflusselement zum nächsten. Das hieraus entstehende Materialflussmodell wird im Kommenden näher erläutert.

4.2.1 Materialflusselemente

Die einzelnen Elemente (vgl. 2.1) innerhalb eines MFS werden zu Materialflusselementen (MFEs) zusammengefasst. Die FEs bewegen sich von Materialflusselement (MFE) zu Materialflusselement und verbleiben mit einer gewissen Transportdauer innerhalb des entsprechenden MFE. Materialflusselemente werden anhand ihrer übergeordneten Funktion in befördernde, erzeugende und verbrauchende MFEs unterschieden.

Die genaue technische Funktionalität, wie z. B. Senkrechtfördern oder manuelle Beförderung durch den Einsatz von Staplern, ist in dieser Betrachtung nicht relevant. So können zusammenhängende Bestandteile eines MFS, wie z. B. Senkrechtförderanlagen oder Loops, zu einem übergeordneten Materialflusselement zusammengefasst werden. Abbildung 4-8 zeigt die Materialflusselemente des hier vorgeschlagenen Modells.

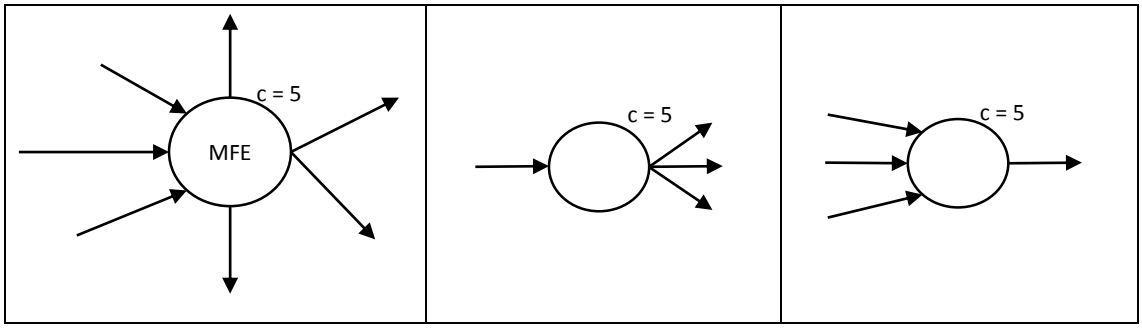

Abbildung 4-8: Beförderndes (links), erzeugendes (Mitte) und verbrauchendes (rechts) MFE

Die Kapazität c der hier vorgestellten MFEs kann durch physische Gegebenheiten, wie z. B. maximal förderbare FEs pro Stunde oder auch der maximalen Aufnahmekapazität, beschränkt werden.

Beförderndes Materialflusselement

Die in Richtung des befördernden Materialflusselements gerichteten Pfeile symbolisieren eingehende Verbindungen von vorhergehenden MFEs und die ausgehenden Pfeile repräsentieren ausgehende Verbindungen zu nachfolgenden MFEs. Des Weiteren verdeutlicht Abbildung 4-8 das komplexe n:m-Verhältnis von Eingängen zu Ausgängen, da i. d. R. ein MFE mit mehreren MFEs verbunden sein kann.

Erzeugendes Materialflusselement

Das erzeugende Materialflusselement hat genau einen Eingang und kann mehrere Ausgänge besitzen. Es erzeugt mit einer vorgegebenen Leistung neue FEs, die in das MFS eingeschleust werden. Die Eigenschaft eines erzeugenden MFE hängt in erster Linie von seiner genauen Funktion ab, so kann z. B. ein K-Platz durchaus als erzeugendes MFE gesehen werden, wenn er die kommissionierte Ware nach Abschluss des Vorgangs in neue FEs packt und diese in das MFS einschleust.

Verbrauchendes Materialflusselement

Das verbrauchende Materialflusselement hat mehrere Eingänge und genau einen Ausgang. FEs die das verbrauchende MFE verlassen, werden nicht wieder in das MFS eingeschleust.

4.2.2 Verbindungen

Verbindungen zwischen den einzelnen MFEs symbolisieren den Übergang von MFE_i zu MFE_{i+1}. Die Leistung von Verbindungen ist durch Kundenwünsche oder physische Gegebenheiten eindeutig festgelegt. Abbildung 4-9 zeigt exemplarisch ein RBG und Loop 1 als MFEs mit Verbindungen und ihren Leistungen.

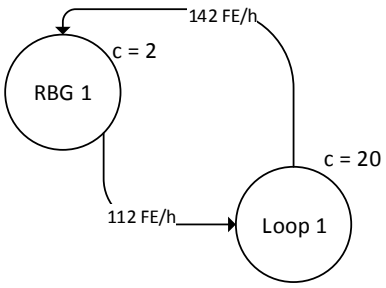

Abbildung 4-9: RBG-Element und Loop-Element mit festen Leistungen

Die festgelegte Ausgangsleistung liegt bei 112 Behältern pro Stunde, die Eingangsleistung bei 142 Behältern pro Stunde. Des Weiteren kann das RBG 1 aufgrund seiner Kapazität $c = 2$ maximal 2 FEs gleichzeitig befördern und das Loop 1 bedingt durch seine Kapazität $c = 20$ maximal 20 FEs aufnehmen.

4.3 Modell der Referenzanlage

Auf Basis der in den Abschnitten 2.1 und 4.2 eingeführten Modellelemente und Materialflüsse wird das Referenzschaubild für den Auslager- und Rücklagerfluss exemplarisch modelliert. Im Anschluss wird das Gesamtmodell der Referenzanlage angegeben.

4.3.1 Auslager- und Rücklagerfluss

Abbildung 4-11 zeigt die Modelle für den Auslager- und den Rücklagerfluss. In den Verbindungen der einzelnen Materialflusselemente sind die geforderten Ein- bzw. Ausgangsleistungen eingetragen.

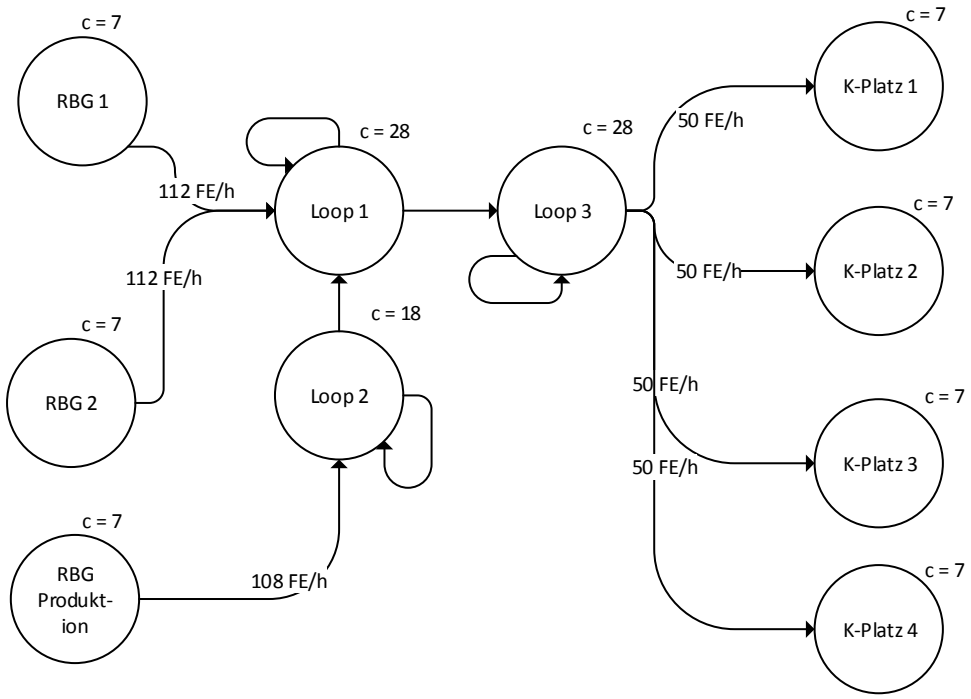

Abbildung 4-10: Auslagerflussmodell

Abbildung 4-11 zeigt den Materialfluss von den RBGs zu den entsprechenden K-Plätzen, hier repräsentieren die K-Plätze Senken für die RBGs.

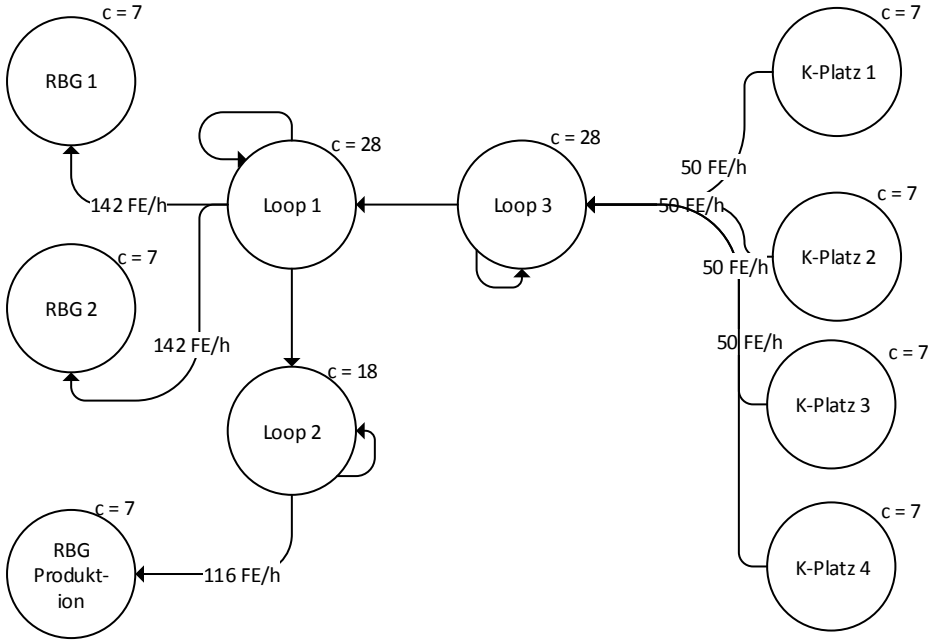

Abbildung 4-11: Rücklagerflussmodell

Abbildung 4-11 zeigt den Rücklagerfluss von den K-Plätzen als Quellen zu den entsprechenden RBGs als Senken. Wie in Abschnitt 4.1.1 beschrieben, fließen die FEs über die entsprechenden Loops zu den K-Plätzen und zurück. Die RBGs und K-Plätze besitzen jeweils eine Kapazität von maximal 7 gleichzeitig darin befindlichen FEs und die Loops können jeweils maximal 28 FEs gleichzeitig aufnehmen.

Um einen optimalen Materialfluss zu erreichen, muss zum einen eine Blockade der MFEs Loop 1, Loop 2 und Loop 3 und zum anderen ein unnötig langer Aufenthalt in den MFEs Loop

1, Loop 2 und Loop 3 vermieden werden, da diese MFE die Transportdauer einer FE am meisten beeinflussen.

Eine Zusammenführung aller Teilmodelle in ein Gesamtmodell verdeutlicht die Komplexität der Referenzanlage und die damit verbundenen Optimierungsmöglichkeiten.

4.3.2 Gesamtmodell der Referenzanlage

Abbildung 4-12 zeigt das Gesamtmodell aller Teilflüsse der Referenzanlage.

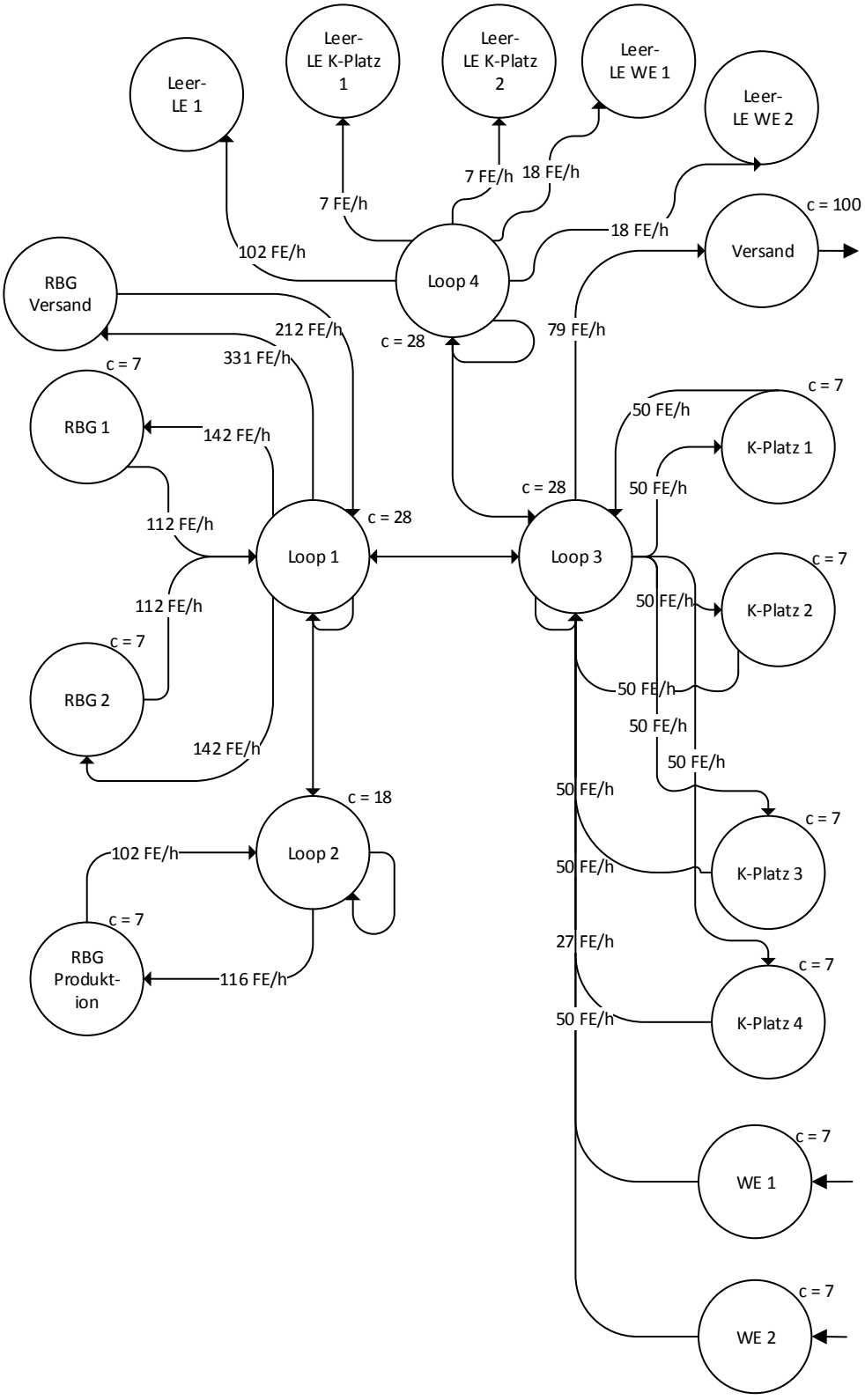

Abbildung 4-12: Gesamtmodell der Referenzanlage

Das Gesamtmodell der Referenzanlage birgt einige Besonderheiten. So bedeutet die Kapazität von insgesamt 100 FEs für den Versand, dass maximal 100 FEs an allen 20 Versandsenken aufgenommen werden können.

Aus der Gesamtbetrachtung aller Teilflüsse wird deutlich, dass der Auslagerfluss die größte Priorität im MFS besitzt, anschließend folgen Einlager- und Rücklagerfluss. Daher zielen die

meisten Optimierungen eines MFS in erster Linie auf die Optimierung des Auslagerflusses ab, dieser zieht die Optimierungen der anderen Flüsse nach sich. Ferner wird die hohe Komplexität deutlich, die mit der Optimierung des Materialflusses einhergeht. So sind bspw. zur Beantwortung der Frage, ob eine FE von RBG 1 das Loop 1 betreten kann, die momentanen Zustände des Loop 1, Loop 3 und jedes einzelnen K-Platzes notwendig.

4.4 Lösungsansatz

Zur Lösung des soeben beschriebenen Optimierungsproblems soll ein System erstellt werden, das mit der hohen Dimension umgehen kann und zusätzlich in der Lage ist, dynamisch auf Schwankungen im Materialfluss zu reagieren. Ferner soll das System die Zusammenhänge innerhalb des MFS selbstständig erlernen und so den Materialfluss optimal gestalten.
Konkret soll das System die folgenden primären Optimierungsziele erfüllen:

- Selbstständige Optimierung der Leistung von Quellen und Senken mit externem Bediener
- Selbstständige Optimierung der internen Quellen und Senken in Abhängigkeit von den externen Quellen und Senken

Aus diesen beiden primären Optimierungszielen lassen sich folgende sekundäre Ziele ableiten:

- Einhaltung der vorgegebenen Geräteleistung
- Um die Auftragszeiten zu minimieren, darf keine FE übermäßig lange in den MFEs verweilen
- Keine Blockade der MFEs durch sog. Deadlocks[3]

Zur Lösung der beschriebenen Anforderungen wurden verschiedene Ansätze aus dem Bereich der künstlichen Intelligenz auf ihre Anwendbarkeit untersucht. Da im Regelfall keine Betriebsdaten über die Anlage vorliegen, scheiden die meisten Ansätze, wie etwa das in 3.4 beschriebene maschinelle Lernen oder auch der starre Einsatz von neuronalen Netzen wie von [Heu97] beschrieben, zur Optimierung der Anlage im Live-Betrieb aus.

Um kontinuierlich aus den Live-Daten lernen zu können, soll das System mittels des Reinforcement-Learning-Ansatzes in der Q-Learning-Variante umgesetzt werden. Der Reinforcement-Learning-Ansatz, seine Komponenten und dessen Umsetzung werden in den kommenden Abschnitten beschrieben. Abschließend wird ein Ansatz zur Integration des Reinforcement-Learning-Ansatzes in die Software viadat vorgestellt.

[3] Die klassische Deadlock-Situation in einem MFS entsteht, wenn alle Senken durch FEs blockiert sind und zusätzlich alle MFEs ausgelastet sind.

5 BESCHREIBUNG DES LÖSUNGSANSATZES

Wie schon in 4.4 erläutert, soll das MFS durch den Einsatz von Reinforcement-Learning in der Q-Variante optimiert werden. Der Hintergrund des Lösungsansatzes sowie die einzelnen Bausteine des Lösungsansatzes werden im Kommenden näher erläutert.

5.1 Reinforcement-Learning

Das Lernen durch Verstärkung[4] stammt ursprünglich aus der Robotik. Das Ziel ist es, den Roboter ohne Überwachung von einem Ausgangszustand zu einem Endzustand, z. B. das Greifen eines Gegenstands, zu bringen. Hierzu erhält der Roboter eine Belohnung für eine gute Aktion und eine Bestrafung für eine schlechte Aktion. [Ert13]

Die genaue Funktionalität des Reinforcement-Learning wird im Folgenden erläutert.

5.1.1 Allgemeine Funktionsweise

Beim Reinforcement-Learning wird zwischen dem Agenten und der Umwelt unterschieden. Abbildung 5-1 veranschaulicht die Funktionsweise des Reinforcement-Learning.

Abbildung 5-1: Der Agent und seine Interaktion mit der Umwelt [Ert13]

Der Agent führt im aktuellen Zustand s_t eine Aktion a_t aus, die ihn die den Folgezustand s_{t+1} bringt. Für das Ausführen der Aktion a_t erhält der Agent eine Belohnung $r_t = (s_t, a_t)$. Die Belohnung kann positiv für „gute“ Aktionen und negativ für „schlechte“ Aktionen sein. Hierbei ist auch der Wert Null eine gültige Belohnung, der lediglich besagt, dass die aktuelle Aktion nicht sofort belohnt wird. In diesem Fall spricht man von einer verzögerten Belohnung. Durch die Belohnungen soll der Agent eine optimale Strategie zur Lösung des vorgegebenen Problems erlernen. Eine Strategie gilt als optimal, wenn sie langfristig über viele Schritte hinweg die Belohnung maximiert. [Ert13]

Die Modellierungen der Umwelt des Agenten basieren meist auf der sog. Markov-Eigenschaft. Die Markov-Eigenschaft sagt aus, dass der Folgezustand des Agenten ausschließlich vom aktuellen Zustand und der momentan ausgeführten Aktion abhängt. [Kra09]

Q-Learning-Variante

Bei der Q-Learning-Variante liegt dem Agenten kein Modell seiner Umwelt vor, der Agent lernt seine Umwelt durch die von ihm ausgeführten Aktionen kennen. [Kra09] Hierzu werden mittels der sog. Q-Funktion $Q(s_t, a_t)$ die aktuelle Aktion sowie der Zustand bewertet. Auf Basis dieser Bewertung erfolgt die Auswahl der optimalen Strategie nach der rekursiven Vorschrift: [Sut15]

$$Q(s_t, a_t) = Q_t(s_t, a_t) + \alpha * \left(r_{t+1} + \gamma * \max_a Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right) \tag{5.1}$$

Die Vorschrift zur Bewertung des aktuellen Zustands $Q(s_t, a_t)$ gliedert sich in die folgenden sechs Bestandteile:

- $Q(s_t, a_t)$ Bewertung des aktuellen Zustands

[4] Engl. Reinforcement-Learning

- α $(0 \leq \alpha \leq 1)$ repräsentiert die Lernrate (Einflussfaktor des zukünftig Gelernten auf das jetzt Gelernte)
- r_{t+1} Belohnung für die aktuell ausgeführte Situation
- γ $(0 \leq \gamma \leq 1)$ ist der Einflussfaktor der zukünftigen Belohnungen
- $\max_a Q_t(s_{t+1}, a_{t+1})$ vorausgesagte zukünftige Bewertung [Sut15]

I. d. R.wird mit einer konstanten Lernrate α und einem konstanten Einflussfaktor γ gearbeitet. [Ert13]

Varianten zur Berechnung der Q-Funktion

In der klassischen Variante des Q-Learning werden zwei Tabellen zur Repräsentation der Q-Funktion und den daraus resultierenden Belohnungen erstellt und bei jedem Besuch der entsprechenden State-Action-Paare aktualisiert, bis der Agent den Endzustand erreicht. [Kra09]

Neuere Ansätze des Q-Learning benutzen neuronale Netze zur Approximation der Q-Funktion, da diese sich aufgrund ihrer Effizienz sehr gut für diese Problemstellung eignen. Auch stellen neuronale Netze eine Art Gedächtnis des Agenten dar. Bei der Approximation durch neuronale Netze werden die Zustände und die aktuelle Aktion als Eingänge an das Netz angelegt und das neuronale Netz gibt als Ergebnis die approximierte Q-Funktion aus. [Ert13]

Auswahl einer Aktion

Der Agent führt verschiedene Aktionen innerhalb des Systems aus. Hierzu hat er zwei Möglichkeiten: Er kann Teile seiner – ihm unbekannten – Umwelt entdecken (Engl. Exploration) oder auf Grundlage des bereits Gelernten (Engl. Exploitation) Entscheidungen treffen. [Ert13]
Die Idee hinter diesen beiden Konzepten wird im Folgenden kurz erläutert.

Entdeckung der Umwelt (Exploration)

Um seine Umwelt kennenzulernen, muss der Agent zu Beginn die Umwelt erforschen, da noch nicht ausreichend Informationen über die Umwelt und ihre Abhängigkeiten vorhanden sind. Hierzu muss der Agent neue Verhaltensweisen ausprobieren, um sich einen Eindruck über das optimale Verhalten zu verschaffen (Engl. Exploration). Dazu führt der Agent zu Beginn mit einer gewissen Wahrscheinlichkeit eine zufällige Aktion aus, um die Abhängigkeiten seiner Umwelt zu verstehen. [Ert13]

Nutzung des bereits Gelernten (Exploitation)

Je mehr der Agent seine Umwelt kennenlernt, desto geringer wird die Wahrscheinlichkeit für die Erforschung der Umwelt und er entscheidet auf Grundlage der gelernten Zustände (Engl. Exploitation). Zur Entscheidungsfindung legt der Agent jedes mögliche Zustands-Aktions-Paar an das neuronale Netz an, approximiert den Q-Wert und die Aktion mit dem höchsten Q-Wert aus. In der Literatur wird eine Kombination aus Exploration und Exploitation empfohlen, die zu Beginn mit einer hohen Wahrscheinlichkeit das System erforscht und im späteren Verlauf die Wahrscheinlichkeit für die Exploration immer weiter senkt. Durch diese Strategie soll die Approximation einer optimalen Strategie gewährleistet werden. [Ert13]

Das Konzept hinter neuronalen Netzen, die Funktionsweise des verwendeten Netzes sowie das Umsetzungswerkzeug werden im kommenden Abschnitt erläutert. Im Anschluss wird die Q-Learning-Variante mit dem Einsatz von neuronalen Netzen näher beschrieben.

5.2 Künstliche neuronale Netze

Neuronale Netze sind ein Teilgebiet der künstlichen Intelligenz. [Kru11] beschreibt ein neuronales Netz als ein *„informationsverarbeitendes System, dessen Aufbau und Funktionsweise dem Nervensystem und vor allem dem Gehirn eines Menschen nachempfunden ist“* [Kru11].

Ein neuronales Netz besteht aus einer endlichen Menge an einfachen, parallel arbeitenden Prozessoren, den Neuronen. Die Neuronen eines neuronalen Netzes *„senden sich Informationen in Form von Aktivierungssignalen über gerichtete Verbindungen zu“* [Kru11]. [Ert13]
Die Grundeigenschaft eines neuronalen Netzes ist die Fähigkeit zu lernen. So finden neuronale Netze überall dort Einsatz, wo kein konkretes mathematisches Modell zur Beschreibung eines Problems vorhanden ist, wie z. B. zur Erkennung von Fingerabdrücken, zur Risikobewertung von Bankkunden und sogar schon zur Voraussage von Lottozahlen. [Liv08]

Im Anschluss wird das Konzept hinter neuronalen Netzen beschrieben und die Funktionsweise des in dieser Arbeit eingesetzten neuronalen Netzes beschrieben.

5.2.1 Bestandteile eines neuronalen Netzes

Ein neuronales Netz ist ein gerichteter Graph $G = (U, C)$, dessen Knoten $u \in U$ als Neuronen und dessen Kanten $c \in C$ als Verbindungen bezeichnet werden. [Kru11]
Grundsätzlich besteht jedes neuronale Netz aus drei Komponenten:

- **Neuronen:** Die Eigenschaften eines Knotens bzw. Neurons bestimmen die Art der Verarbeitung der Signale, wie etwa die Anzahl an Ein- und Ausgängen, die Gewichtsfunktion der Ein- und Ausgänge und die Aktivierungsfunktion.
- **Netzwerktopologie:** Die Netzwerktopologie beschreibt die Art der Kanten zwischen den Knoten.
- **Lernregeln:** Die Lernregeln legen fest, wie die Gewichte initialisiert und ausgerichtet werden. [Liv08]

Die folgenden Abschnitte erläutern die soeben eingeführten Komponenten.

Neuronen

Wie bereits im vorherigen Abschnitt beschrieben, kann ein Neuron auch als Knoten eines gerichteten Graphen verstanden werden. Der grundsätzliche Aufbau eines Neurons ist in Abbildung 5-2 dargestellt.

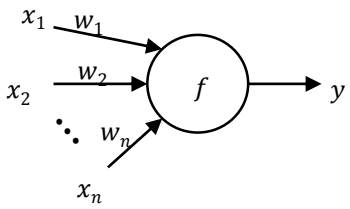

Abbildung 5-2: Aufbau eines Neurons nach [Liv08]

An einem Neuron liegen die Eingänge x_1 bis x_n mit ihren dazugehörigen Gewichten w_1 bis w_n an, jeder Eingang ist der Ausgang eines vorhergehenden Neurons. In den Gewichten der ausgehenden Kante wird das erlernte Wissen des Neurons gespeichert. [Liv08]

Aktivierungsfunktion

Zur Aktivierung eines Neurons werden die gewichteten Eingangswerte $x_1, x_2, \dots, x_n$ [Ert13]

$$\sum_{j=1}^{n} w_{ij} x_j \tag{5.2}$$

aufsummiert und im Anschluss wird die Aktivierungsfunktion f auf das Ergebnis angewendet und das Ergebnis an die nachfolgenden Neuronen über die Kantengewichte weitergegeben. [Ert13]
Zur Berechnung der Aktivierungsfunktion f bieten sich eine Reihe von Funktionen an, [Ert13] nennt hier die Identitätsfunktion

$$f(x) = x \tag{5.3}$$

oder auch eine sigmoide Funktion wie z. B.

$$f(x) = \frac{1}{1 + e^{x}} \tag{5.4}$$

Um Konvergenzprobleme und Unstetigkeiten in der Aktivierungsfunktion zu umgehen, empfiehlt [Ert13] die Verwendung einer sigmoiden Aktivierungsfunktion.

Netzwerktopologie

Die Netzwerktopologie beschreibt die Art der Verbindung zwischen den Neuronen. Die Neuronen sind in linearen Vektoren, den sog. Schichten, organisiert. Die Schichten werden in Eingabe-, Ausgabe- und versteckten Schichten unterteilt. Entsprechend werden die Neuronen in Eingabe-, Ausgabe- bzw. versteckte Neuronen, die nur Kontakt mit anderen Neuronen im Netz haben, unterschieden. [Liv08]

[Kru11] klassifiziert die neuronalen Netze anhand der Kanteneigenschaften des Graphen in Vorwärtsbetriebe und rekurrente Netze.
Abbildung 5-3 zeigt exemplarisch den Aufbau der Netzwerktopologien neuronaler Netze.

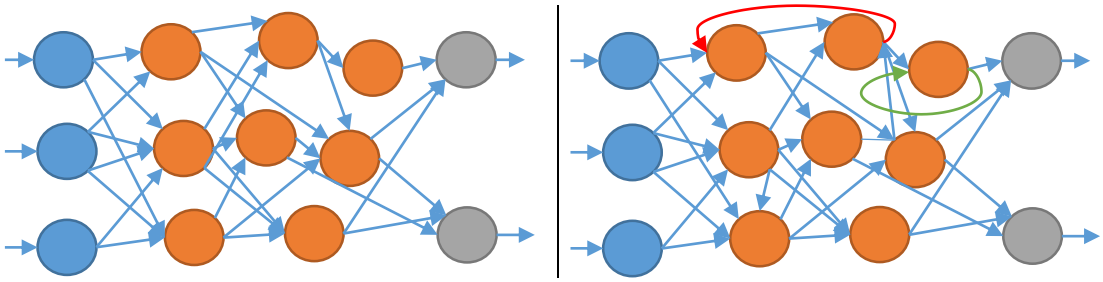

Abbildung 5-3: Vorwärtsbetriebenes (links) und rekurrentes (rechts) neuronales Netz

In vorwärtsbetriebenen Netzen (links) werden die Informationen immer in einer festen Richtung vom Eingang zum Ausgang weitergeleitet.

In rekurrenten Netzen hingegen sind Schleifen (grün) oder auch Zyklen (rot) erlaubt. Die Funktionsweise beider Netztopologien ist jedoch dieselbe.

Funktionsweise

An den Eingabeneuronen (blau) in Abbildung 5-3 liegen die Eingangssignale für das Netzwerk an. Hier werden die entsprechenden Ausgangsfunktionen der Neuronen, nach Erreichen des Schwellwertes, berechnet. Sobald alle Eingabeneuronen ihr Ergebnis ermittelt haben, wird das Ergebnis an die versteckten Neuronen (orange) weitergegeben. Die versteckten Neuronen berechnen wiederum ihre Funktion und geben das Ergebnis nach Abschluss aller Berechnungen an die Ausgabeneuronen (grau) weiter. Diese geben das entsprechende Ergebnis an die Umwelt ab. Die Weitergabe der Ergebnisse erfolgt immer schichtweise. [Kru11]

Lernregeln

Der Kernpunkt der neuronalen Netze sind die Lernregeln. Sie bestimmen die Art der Wissensgenerierung. Der Lernprozess kann in überwachtes und unüberwachtes Lernen unterteilt werden.

Überwachtes Lernen

Beim überwachten Lernen wird dem System eine Menge an Trainingsdaten $T = \{(x_i, d_i) | i = 1,2, \dots n\}$, als Menge an Eingabevektoren x_t und Ausgabevektoren d_t bereitgestellt,

auf deren Basis das System trainiert wird. Die einzelnen Gewichte der Kanten des neuronalen Netzes werden so ausgerichtet, dass der Fehler zwischen der Ausgabe des neuronalen Netzes und der tatsächlichen Ausgabe minimal wird. Zur Erstellung der Trainingsdaten wird i. d. R. spezielles Expertenwissen benötigt. Ferner müssen die Trainingsdaten repräsentativ für das vorliegende Modell sein. Mit den vorliegenden Trainingsdaten wird das System zuerst trainiert. Sobald das neuronale Netz die gewünschten Ausgabeparameter produziert, werden die Kantengewichte fest eingestellt und das System geht in den Produktivmodus. [Liv08]

Unüberwachtes Lernen

Im Gegensatz zum überwachten Lernen liegen dem Netz beim unüberwachten Lernen keine Zielparameter vor. Das neuronale Netz versucht das Muster in den Trainingsdaten selbstständig zu erkennen. [Liv08]

5.2.2 Resilient Backpropagation

Der Resilient-Backpropagation-Algorithmus (RProp) basiert auf dem Standard-Backpropagation-Algorithmus und gehört zur Klasse des überwachten Lernens. Der RProp besteht aus zwei Durchläufen: dem Vorwärts- und Rückwärtsdurchlauf. [Rie93]
Auf den genauen Ablauf sowie die Eigenschaften des RProp wird im Folgenden eingegangen.

Trainingsablauf

Zu Beginn erhält der RProp eine Menge T an Trainingsdaten. Danach werden die Gewichte mit zufälligen, möglichst niedrigen Werten initialisiert.
Abbildung 5-4 illustriert den Ablauf des RProp mit seinem Vorwärtsdurchlauf (links) und Rückwärtsdurchlauf (rechts).

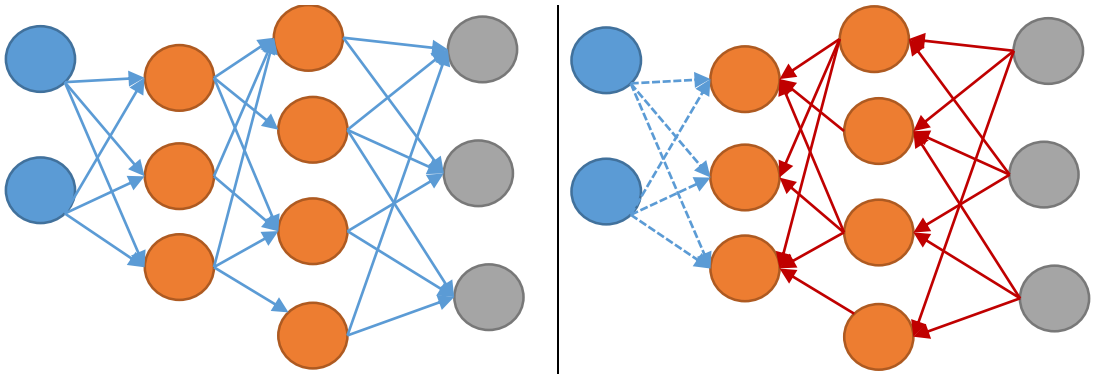

Abbildung 5-4: Vorwärtsdurchlauf im RProp (links) und Rückwärtsdurchlauf im RProp (rechts)

Vorwärtsdurchlauf

Im ersten Durchlauf wird für ein Trainingsbeispiel $\{(x_t, d_t) | \, t \in T\}$ das Netz von den Eingabeneuronen (blau) über die versteckten Neuronen (orange) zu den Ausgabeneuronen (grau) durchlaufen. Hier wird, ab der ersten versteckten Schicht, für jedes Neuron in der aktuellen Schicht die Aktivierungsfunktion berechnet. Nach dem Durchlauf aller versteckten Schichten wird die Netzausgabe sowie der Approximationsfehler für das gewählte Trainingsbeispiel $\{(x_t, d_t) | \, t \in T\}$ berechnet. [Ert13]

Rückwärtsdurchlauf

Im Anschluss wird das Netz von hinten nach vorne (roter Pfeil), in entgegengesetzter Reihenfolge des Vorwärtsdurchlaufs, für den Satz $\{(x_t, d_t) | \, t \in T\}$ bis zur letzten versteckten Schicht durchlaufen. Im Gegensatz zum Vorwärtsdurchlauf wird hier jedoch der Approximationsfehler durch das Netz propagiert. Unter dessen Verwendung dann Schicht für Schicht die Kantenge-

wichte neu berechnet werden. Der hier beschriebene Ablauf wird nach dem Durchlauf des Satzes $\{(x_t, d_t) | \, t \in T\}$ auf die gesamte Menge der vorhandenen Trainingsdaten angewandt. Der Algorithmus terminiert, sobald sich die Gewichte nicht mehr ändern oder eine voreingestellte Zeit erreicht ist. Der komplette Durchlauf aller Trainingsdaten wird Epoche genannt. [Ert13]

Besonderheiten

Im Gegensatz zum Standard-Backpropagation-Algorithmus werden beim RProp-Ansatz die einzelnen Kantengewichte nicht durch das Netzwerk propagiert, sondern jedes Gewicht besitzt seinen individuellen sich entfaltenden Aktualisierungswert. Die Gewichtung der Kante wird ausschließlich durch ihren Aktualisierungswert und das Vorzeichen des Gradienten bestimmt. [Rie93]

5.2.3 Auswahl des Umsetzungswerkzeugs

Zur Umsetzung von neuronalen Netzen steht eine große Bandbreite an Werkzeugen und Bibliotheken bereit. In diesem Abschnitt werden die für diese Arbeit näher betrachteten Umsetzungsmöglichkeiten kurz beschrieben, bewertet und ein Werkzeug auf Basis der Bewertung ausgewählt.

Stuttgart Neural Network Simulator

Der Stuttgart Neural Network Simulator (SNNS) ist ein Kooperationsprojekt der Universitäten Stuttgart und Tübingen. Entwickelt wurde der SNNS in Stuttgart, gewartet wird er in Tübingen. Mittlerweile ist die Weiterentwicklung jedoch eingestellt worden. Der SNNS besitzt eine GUI und ist in C programmiert. Der Kern des Simulators kann auch in externe Programme eingebettet und durch eigene Funktionen erweitert werden. Der SNNS bietet im Standard die folgenden Netzwerktopologien und Lernregeln[5] an:

- Backpropagation
- Counterpropagation
- Quickprop
- Backpercolation 1
- Resilient Backpropagation (RProp)
- Generalisierte radiale Basisfunktionen (RBF)
- ART1 und ART2
- ARTMAP
- Cascade Correlation
- Recurrent Cascade Correlation
- Dynamic LVQ
- Backpropagation durch die Zeit
- Quickprop durch die Zeit
- Selbstorganisierende Karten
- Zeitverzögerte Netzwerke mit Backpropagation
- Jordan und Elman Netzwerke
- Assoziative Speicher

Zusätzlich können über die GUI die Netze analysiert werden. Zur Ausführung der GUI des SNNS wird eine X-GUI für Windows oder ein Linux-System benötigt. Zusätzlich können die erstellten Netze durch ein mitgeliefertes Werkzeug in C-Quellcode übersetzt werden. Der SNNS ist unter der GNU GPL[6] lizensiert. [Zel151]

[5] Eine genaue Erläuterung der hier aufgeführten Algorithmen kann u. a. [Alp08] entnommen werden..
[6] Vgl. [GNU151] für eine genaue Beschreibung der GNU GPL.

Java Neural Network Simulator

Der Java Neural Network Simulator (Java NNS) ist der Nachfolge des SNNS. Er basiert auf dem Kern des SNNS und wurde mit einer neuen GUI in Java erweitert. Er bietet die gleichen Lernregeln und Netzwerktopologien wie sein Vorgänger an, jedoch kann der JavaNNS keinen C-Quellcode erstellen. Er kann ohne eine zusätzliche Installation von X-GUI betrieben werden. Zur Ausführung wird jedoch eine aktuelle Version der Java-Bibliothek benötigt. Des Weiteren bietet Java NNS mehr Möglichkeiten zur Analyse und Darstellung der neuronalen Netze. Das Java NNS darf verändert werden, unterliegt jedoch keiner speziellen GNU Lizenz. [Zel15]

Open Neural Networks Library

Das Open Neural Networks Library (Open NN) ist eine offene Bibliothek erstellt in C++ zur Implementierung von neuronalen Netzen. Open NN bietet im Standard die folgenden Netzwerktopologien und Lernregeln[7] an:

- 1-D Optimierung
- Multidimensionale Optimierung
- Gradientenabstiegsverfahren
- Newton-Methode
- Konjugierte Gradienten
- Quasi-Newton-Methode
- Zufallssuche
- Evolutionäre Algorithmen

Die Bibliothek ist in C++ erstellt und kann in C++-Quellcode implementiert werden. Ferner wird die Bibliothek ständig weiterentwickelt. Das Open NN ist unter der GNU LGPL[8] lizenziert und kann kommerziell verwendet werden. [Rob15]

MemBrain NN

Das MemBrain NN ist ein Simulator für neuronale Netze, der zusätzlich die Möglichkeit bietet, Netze selbst zu erstellen und diese nach C zu exportieren. Darüber hinaus kann in MemBrain mithilfe von selbst erstellen Skripten das Verhalten des neuronalen Netzes komplett vom Bediener gesteuert werden. Zusätzlich können die Funktionen von MemBrain mittels einer DLL in alle gängigen sowie in selbst erstellte Programme integriert werden. MemBrain bietet im Standard die folgenden Lernregeln an:

- Backpropagation
- Backpropagation mit Rückkopplung
- Resilient Backpropagation (RProp)
- Cascade Correlation
- Trial and Error
- Unterstützung von Self Organizing Maps mittels „Winner takes it all“-Ansatz

MemBrain kann für Forschungszwecke kostenlos eingesetzt werden. Für die kommerzielle Nutzung fallen je nach Nutzungsgrad unterschiedliche Lizenzgebühren an. [Tho16]

[7] Die genaue Funktionsweise der aufgelisteten Algorithmen kann [Rob15] entnommen werden.
[8] Vgl. [GNU15] für eine genau Beschreibung der GNU LGPL.

Auswahl des Umsetzungswerkzeugs

Zur Auswahl des geeigneten Werkzeugs werden die vorgestellten Werkzeuge in die Bewertungsmatrix in Tabelle 5-1 eingetragen und anhand der Kriterien verglichen. Hierzu werden den einzelnen Bewertungskriterien unterschiedliche Gewichtungen (g) entsprechend der Wichtigkeit zugeordnet. Des Weiteren erhält jedes Werkzeug eine Punktzahl (n) zwischen 0 und 9 entsprechend seiner Bedeutung. Die Summe aller gewichteten Teilkriterien (n*g) ergibt den Rang des jeweiligen Werkzeugs.

Tabelle 5-1: Bewertungsmatrix der Werkzeuge

Kriterium	**g**	**SNNS-Kern**		**Java NNS**		**Open NN**		**MemBrain**	
		n	**n*g**	**n**	**n*g**	**n**	**n*g**	**n**	**n*g**
Lizenzierung	**5**	7	35	5	25	7	35	3	15
Verfügbare Netze	**10**	9	90	9	90	3	30	9	90
Implementierbarkeit	**30**	9	270	0	0	9	270	9	270
Weiterentwicklung	**20**	0	0	0	0	9	180	9	180
Plattformunabhängigkeit	**10**	9	90	9	90	9	90	9	90
Dokumentation	**5**	9	45	9	45	9	45	9	45
Grafischer Editor	**5**	6	30	8	40	8	40	9	45
Testmöglichkeiten	**5**	8	40	8	40	5	25	9	45
Kombinierbarkeit mehrerer Netze	**5**	9	45	0	0	9	45	9	45
Summe	**95**	645		330		760		**825**	
Rang		**3**		**4**		**2**		**1**	

Aufgrund des in Tabelle 5-1 festgelegten Ranges der Werkzeuge wird zur Realisierung des neuronalen Netzes das MemBrain-Werkzeug ausgewählt.

5.3 Q-Learning mit neuronalen Netzen

Wie in 5.1.1 beschrieben, eignen sich neuronale Netze zur Approximation der Q-Funktion. Abbildung 5-5 illustriert die Verwendung von neuronalen Netzen zur Approximation der Q-Funktion innerhalb eines Agenten.

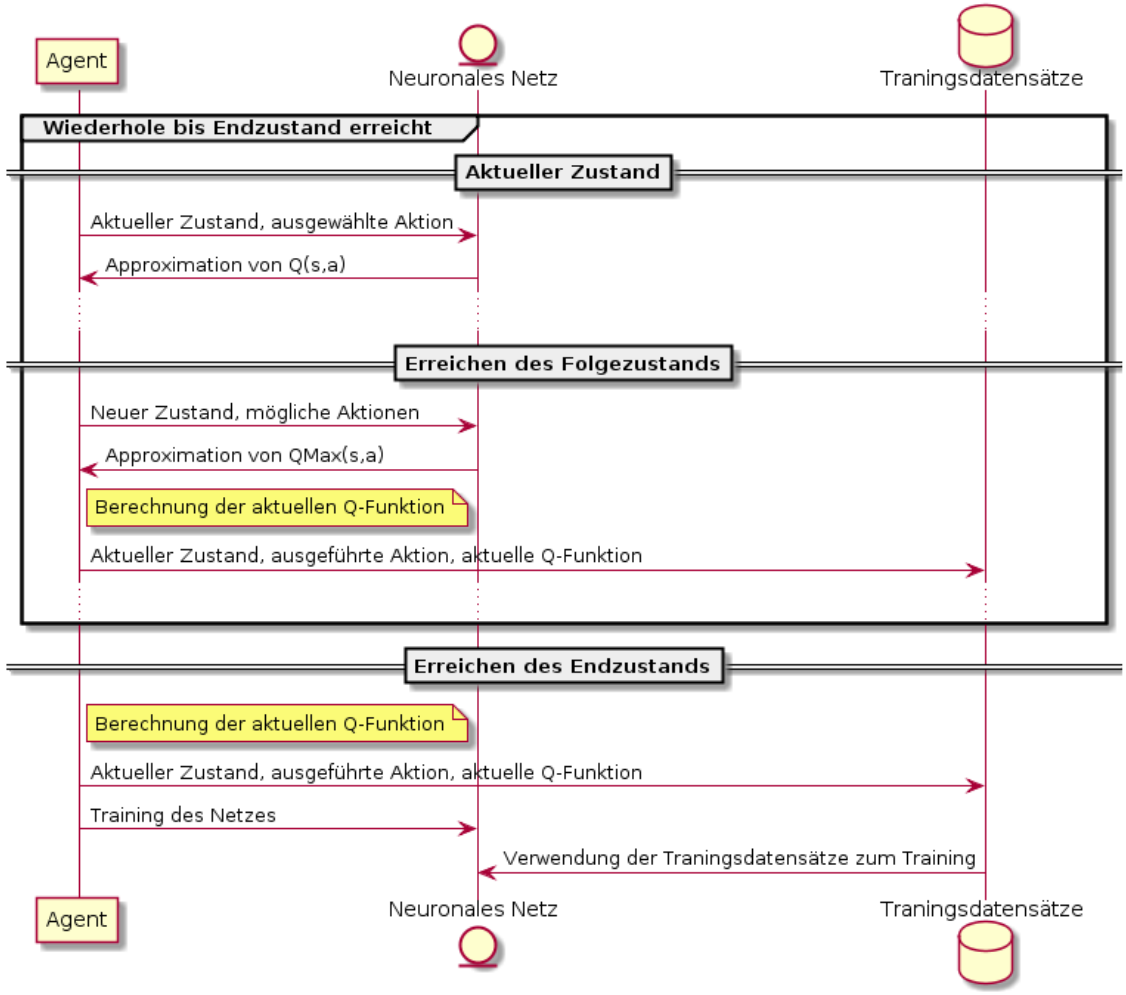

Abbildung 5-5: Ablauf des Q-Learning mit neuronalen Netzen

Hierzu werden die aktuelle Aktion und die Zustände an das neuronale Netz als Eingänge angelegt und die daraus resultierende Q-Funktion approximiert. Im Anschluss wird durch das gleiche Vorgehen die Bewertung des Folgezustandes $\max_a Q_t(s_{t+1}, a_{t+1})$ durch das neuronale Netz approximiert. Die Approximation der Bewertung des jetzigen und des Folgezustands wird solange wiederholt, bis der Endzustand erreicht ist. Nach Erreichen des Endzustands werden die Trainingsdaten den bereits vorhandenen Trainingsdaten hinzugefügt und das neuronale Netz wird mit den vorhandenen Trainingsdaten trainiert. Um die Berechnungszeit der Approximation der Q-Funktion zu reduzieren, wird die Anzahl der Trainingsdatensätze des neuronalen Netzes auf z. B. 10000 begrenzt. [Kra09]

Das genaue Vorgehen zur Auswahl einer Aktion, Bewertung der Zustände und dem Training des Agenten wird im Folgenden näher erläutert.

5.3.1 Auswahl einer Aktion

Der Agent wählt, wie in 5.1.1 beschrieben, seine auszuführende Aktion entweder zufällig (Engl. Exploration) oder auf Grundlage des bereits Gelernten (Engl. Exploitation) aus. [Kru11]

Entdeckung der Umwelt

In der Exploration-Phase wird die Q-Funktion für das Zustands-Aktions-Paar einer zufällig gewählten Aktion approximiert. In beiden Fällen werden die Werte der Zustände, die Aktion und der approximierte Wert der Q-Funktion für das spätere Training des Netzes durch den Agent in der Menge der Trainingsdatensätze abgelegt. [Ert13]

Nutzung des bereits Gelernten

Um eine Aktion in der Exploitation-Phase auszuwählen, legt der Agent für jede Aktion das Zustands-Aktions-Paar an das neuronale Netz als Eingangsvariablen an und approximiert die Q-Funktion. Nach der Approximation aller möglichen Zustands-Aktions-Paare wählt der Agent das Zustands-Aktions-Paar mit der höchsten Bewertung der Q-Funktion aus und speichert das Zustands-Aktions-Paar zusammen mit dem approximierten Q-Wert in den Trainingsdaten des neuronalen Netzes für das spätere Training ab. [Ert13]

In beiden Fällen wird der vorausgesagte Gewinn $\max_a Q_t(s_{t+1}, a_{t+1})$ aus (5.1) nach Erreichen des Folgezustands durch das neuronale Netz des Agenten approximiert und das Ergebnis wird dem aktuellen Trainingsdatensatz hinzugefügt. [Kru11]

Zusätzlich ist darauf zu achten, dass die Exploration-Phase nicht zu früh verlassen wird, da es sonst zu einer nicht optimalen oder schlechten Approximation einer optimalen Strategie kommen kann. [Ert13]

5.3.2 Training des neuronalen Netzes des Agenten

Erreicht der Agent einen Endzustand, werden die entsprechenden Werte der Q-Funktion nach der Vorschrift aus (5.1) berechnet. Anschließend wird das neuronale Netz mit den aktualisierten Trainingsdatensätzen trainiert. [Ert13]

5.4 Das Agentensystem

Vor der Umsetzung des Lösungsansatzes werden an dieser Stellte noch einige Vorüberlegungen bzgl. des Agentensystems angestellt. Aufgrund der hohen Komplexität des vorliegenden Problems empfiehlt sich zur Lösung des Problems der Einsatz eines Multiagentensystems, da im Betrieb eines MFS die jeweiligen Entscheidungen in Sekundenbruchteilen getroffen werden müssen. Durch den Einsatz von mehreren Agenten kann die Rechenzeit minimiert werden, da mehrere Agenten auch parallel trainiert und befragt werden können. In einem Multiagentensystem ist jeder Agent für seinen Bereich zuständig und agiert unabhängig von den anderen Agenten. [Ert13]

5.4.1 Einsatz des Agentensystems

Wie in Kapitel 4 beschrieben, existieren in einem MFS unterschiedliche Flüsse. Eine sich im MFS bewegende FE folgt immer einem dieser Flüsse. Abhängig von Quelle, Ziel und Fluss werden unterschiedliche MFEs besucht. Um die MFEs auf der Route der FE nicht zu überlasten und den Materialfluss konstant zu halten, muss möglichst früh die Entscheidung getroffen werden, ob die aktuelle FE das MFE wechseln kann. Ausgehend von den möglichen Aktionen des Agenten ist der Einsatz von Agenten überall da sinnvoll, wo die Frage beantwortet werden muss, ob das nächste MFE betreten werden kann. Somit ergeben sich die ein- und ausgehenden Verbindungen der einzelnen MFEs als mögliche Agentenposition.

Abbildung 5-6 zeigt am Beispiel eines fiktiven MFS, bestehend aus einem RBG, Loop, zwei K-Plätzen, einem Wareneingangsplatz und einem Versandplatz die Position der zugehörigen Agenten, markiert durch einen blauen gestrichelten Pfeil.

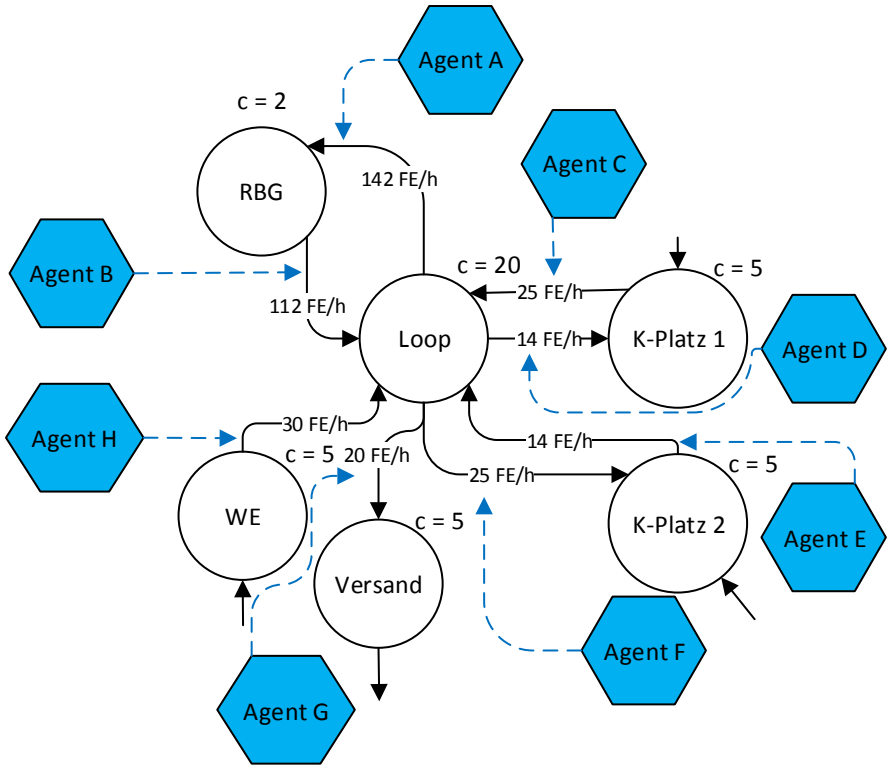

Abbildung 5-6: Agentenposition in einem fiktiven Modell

Zur besseren Übersicht wurden in Abbildung 5-6 die Agenten durch ein blaues Sechseck dargestellt. Das hier abgebildete Modell eines fiktiven MFS besteht aus insgesamt sechs MFEs und acht zur Steuerung des Materialflusses benötigten Agenten.

Die einzelnen MFEs des Modells besitzen alle eine beschränkte Aufnahmekapazität c und die ein- und ausgehenden Verbindungen der MFEs sind durch eine maximale Förderkapazität pro Stunde beschränkt. Der Wareneingang ist das FEs-erzeugende und der Versand das FEs-verbrauchende MFE des Modells.

Die FEs fließen nach Erzeugung vom Wareneingang über das Loop zum RBG, dort verbleiben sie bis zum Abruf durch einen der K-Plätze. Nach Abruf einer FE fließt diese vom RBG über das Loop zum anfordernden K-Platz. Dort wird durch den Benutzer der Kommissioniervorgang gestartet und nach Abschluss des Vorgangs fließt die FE über das Loop zurück in das RBG, zusätzlich kann eine neue FE auch über das Loop zum Versand fließen.

Im hier dargestellten Modell steuern die Agenten (blau) die Übergänge der folgenden MFEs:

- **Agent A:** Übergang von Loop zu RBG
- **Agent B:** Übergang von RBG zu Loop
- **Agent C:** Übergang von K-Platz 1 zu Loop
- **Agent D:** Übergang von Loop zu K-Platz 1
- **Agent E:** Übergang von K-Platz 2 zu Loop
- **Agent F:** Übergang von Loop zu K-Platz 2
- **Agent G:** Übergang von Loop zum Versandplatz
- **Agent H:** Übergang von WE zu Loop

Ausgehend vom hier vorgestellten Lösungsansatz wird im folgenden Kapitel die konkrete Umsetzung des vorgestellten Lösungsansatzes in einem ereignisbasierten Simulator beschrieben. Anschließend wird der Lösungsansatz auf seine praktische Anwendbarkeit untersucht und es wird ein Ansatz zur Integration des Lösungsvorschlags in die Software viadat vorgestellt.

6 UMSETZUNG DES LÖSUNGSANSATZES

Zur Analyse der Umsetzbarkeit des in Kapitel 5 vorgestellten Lösungsansatzes wurde ein Simulator in C++ auf Basis der Modellierung der Referenzanlage erstellt. Das Vorgehen zur Erstellung des Simulators und die Integration des Lösungsansatzes in einen selbst erstellten Simulator werden im Folgenden näher erläutert.

6.1 Der ereignisorientierte Simulator

Zur Simulation des Verhaltens der Referenzanlage und zur Analyse des Lösungsansatzes wurde in C++ ein ereignisorientierter Simulator erstellt, der auf einem von viastore SOFTWARE GmbH erstellten Framework basiert und über die Kommandozeile gestartet werden kann. Eine grafische Repräsentation des Materialflusses ist in der momentanen Version des Simulators nicht vorgesehen.

Die Transporte von FEs werden als diskrete Ereignisse abgebildet, wobei das Auslösen eines Ereignisses als Ankunft an einem Fördertechnikplatz innerhalb des MFS gewertet wird.

6.1.1 Besonderheiten des Simulators

Der Simulator wurde so konzipiert, dass einzelne Bereiche durch eine externe Konfiguration vor Beginn der Simulation aktiviert bzw. deaktiviert werden können. So können ohne Änderungen des Quellcodes des Simulators mehrere Simulationen parallel betrieben werden.

Da physikalische Gegebenheiten im erstellten Simulator nicht von Belang sind, bietet der Simulator die Möglichkeit, einen Zeitraffer zu konfigurieren. Somit kann das zeitliche Verhalten einer realen Anlage in einem Bruchteil der Echtzeit, sechs Simulationsminuten entsprechen hier einer Stunde in Echtzeit, nachempfunden werden.

Zusätzlich können alle Parameter, die für die Implementierung des Reinforcement-Learning-Ansatzes benötigt werden, während der Laufzeit der Simulation angepasst werden. Eine Änderung der Variablen der Q-Funktion kann bspw. einfach per IPC[9]-Nachricht an den Simulator geschickt werden.

Zusätzlich bietet der Simulator die Möglichkeit, die trainierten neuronalen Netze und die damit verbundenen Trainingsdaten als MemBrain-Dateien auszugeben. Ferner wird in regelmäßigen Abständen der aktuelle Zustand der Simulation als CSV-Dateien, bestehend aus:

- Liste der aktiven Transporte
- Liste der durchschnittlichen Transportzeiten der letzten zehn Minuten von allen Quellen zu den Senken
- Liste der durchschnittlichen Transportzeiten der letzten Stunde von allen Quellen zu den Senken
- Liste der Durchsätze der letzten zehn Minuten an allen MFEs, Quellen und Senken
- Liste der Durchsätze der letzten Stunde an allen MFEs, Quellen und Senken
- Trainingsdaten der einzelnen Agenten

für die Verarbeitung in Excel ausgegeben. Zusätzlich können die soeben beschriebenen Daten auch auf Anforderung per IPC-Nachricht ausgegeben werden.

Darüber hinaus können während der Simulation im Zeitraum von 10 Simulationsminuten die gemessenen Durchsätze als Diagramme durch den Einsatz des Werkzeugs Gnu Plot [Gnu16] ausgegeben werden. Durch die regelmäßige Ausgabe der Daten als CSV und grafische Darstellung konnte in den einzelnen Erstellungsschritten das Verhalten des Simulators überprüft und die korrekte Funktion validiert werden.

[9] Engl. Interprocesscommunication (IPC): Kommunikation von verschiedenen Prozessen mit getrenntem Speicherbereich. [Fis11]

6.1.2 Abbildung der Transportzeiten

Zur Abbildung der Transportzeiten innerhalb des Simulators wurden die Transportzeiten der Referenzanlage im laufenden Betrieb erfasst und durch eine geeignete Abbildung als diskrete Ereignisse in den Simulator übertragen. Im Folgenden wird die detaillierte Vorgehensweise zur Abbildung der Transportzeiten innerhalb des Simulators beschrieben.

Erfassung der Transportzeiten

Für eine möglichst exakte Abbildung der Transportzeiten innerhalb des Simulators wurden durch eine Auswertung der Transportzeiten der Referenzanlage im laufenden Betrieb die Transportzeiten ermittelt. Hierzu wurde aus den vorhandenen Transportzeiten der jeweiligen Strecke ein Mittelwert über jeweils 100 erfasste Daten genommen. Ein Auszug der ermittelten Transportzeiten der Referenzanlage kann Tabelle 6-1 entnommen werden.

Tabelle 6-1: Ermittelte Transportzeiten der Referenzanlage

Strecke		**Transportdauer in ms**
Von	**Nach**	
RBG 1	Eingang RBG 1 Loop 1	5000
RBG 2	Eingang RBG 2 Loop 1	5000
Eingang Loop 1	Übergang Loop 1 zu Loop 2	54000
Eingang Loop 2	Eingang K-Platz 1	38000
Eingang Loop 2	Eingang K-Platz 2	39000
Eingang Loop 2	Eingang K-Platz 3	40000
Eingang Loop 2	Eingang K-Platz 4	41000
Eingang K-Platz 1	K-Platz 1	1000
Eingang K-Platz 2	K-Platz 2	1000
Eingang K-Platz 3	K-Platz 3	1000
Eingang K-Platz 4	K-Platz 4	1000
Übergang Loop 1 zu Loop 2	Übergang Loop 1 zu Loop 2	118000
Eingang Loop 2	Eingang Loop 2	115000
Ausgang K-Platz 1	Übergang Loop 2 zu Loop 1	148000
Ausgang K-Platz 2	Übergang Loop 2 zu Loop 1	147000
Ausgang K-Platz 3	Übergang Loop 2 zu Loop 1	146000
Ausgang K-Platz 4	Übergang Loop 2 zu Loop 1	145000
Eingang Loop 1	Eingang RBG 1 Loop 1	46000
Eingang Loop 1	Eingang RBG 2 Loop 1	49000
Eingang RBG 1 Loop 1	RBG 1	5000
Eingang RBG 2 Loop 1	RBG 2	5000

Die Zeiten von *Übergang Loop 1 zu Loop 2* nach *Übergang Loop 1 zu Loop 2* und von *Eingang Loop 2* nach *Eingang Loop 2* repräsentieren eine vollständige Umkreisung des entsprechenden

Loops. Die hier ermittelten Zeiten der Strecken können so dem Gesamtmodell aus Abbildung 4-12 zugeordnet werden.

Transportzeiten in befördernden Materialflusselementen

Beim Betreten eines MFE wird der Timer mit der entsprechenden Transportdauer gestartet, ist der Timer abgelaufen, gilt die FE als am Ende des aktuellen MFE angekommen. Vor dem Betreten des nächsten MFE wird dessen Kapazität geprüft, ist noch Kapazität vorhanden, betritt die FE das nachfolgende MFE. Ist keine Restkapazität vorhanden, verweilt die FE für eine zusätzliche Runde im aktuellen MFE.

Die Motoren innerhalb der Fördertechnik zeichnen sich durch einen konstanten Vortrieb der FEs aus. Mit der Zunahme von FEs innerhalb einer Strecke sinkt die Fördergeschwindigkeit innerhalb des MFE. Um ein möglichst reales Verhalten der Transportzeiten zu erreichen, wird die Abnahme der Fördergeschwindigkeit durch die Zunahme von FEs in den Loops durch eine zunehmende Transportzeit abgebildet. Hierbei soll durch einen Sprung in den Transportzeiten, bedingt durch eine hohe Anzahl an FEs innerhalb eines MFE, ein Staueffekt simuliert werden. Hierzu wurde die Transportzeit t_{tr} um einen Faktor $t_{tr+} = 5$ und eine Stufenfunktion

$$t_{tr} = \begin{cases} \dfrac{Anzahl\ FE\ im\ FME}{maximal\ erlaubte\ MFE} * t_{tr+} & \text{Falls Anzahl FE < x} \\ 5 * t_{tr+} & \text{sonst} \end{cases} \qquad (6.1)$$

erweitert. Mit der Annahme $x = 20$ ergibt sich mit (6.1) die in Abbildung 6-1 dargestellte Kurve für füllstandsabhängige Transportzeit in den MFE Loop 1 und Loop 3.

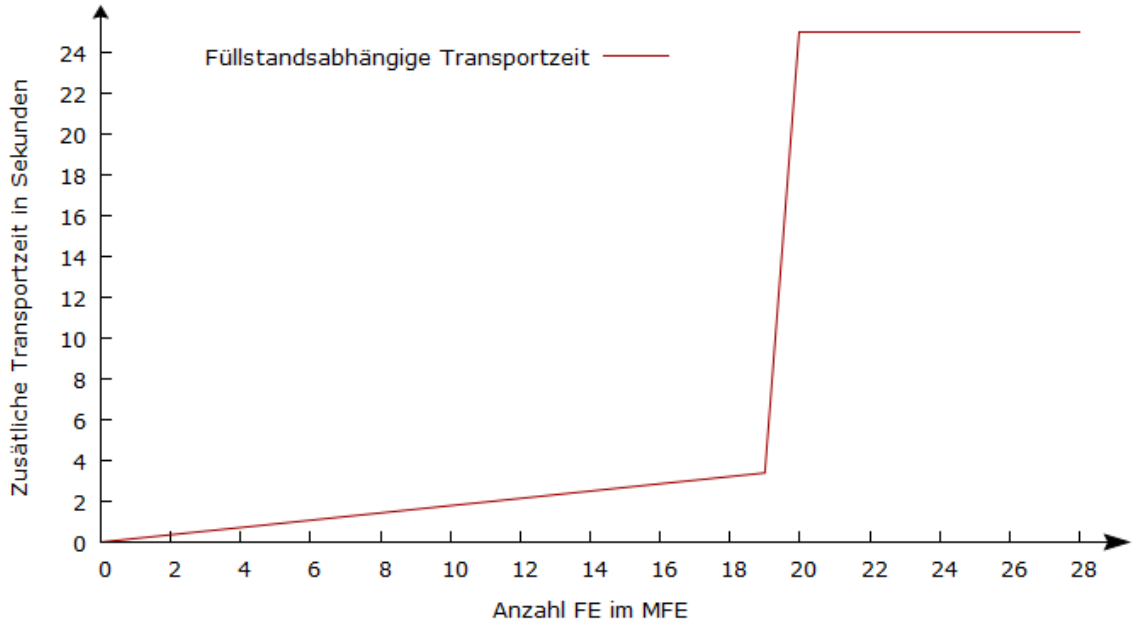

Abbildung 6-1: Verlauf der Transportzeit für die MFE Loop 1 und Loop 3

In Abbildung 6-1 ist der stufenartige Verlauf der füllstandsabhängigen Transportzeit in den Loop 1 und Loop 3 zu erkennen. Bis zu einer Füllmenge von 19 FEs im Loop steigt die zusätzliche Transportdauer linear an, danach bleibt sie konstant bei 25 Sekunden.

Die Wahl der Stufenfunktion als füllstandsabhängige Transportzeit soll im späteren Verlauf der vorliegenden Arbeit zur Beurteilung der Qualität der Agentenentscheidungen dienen. Hier soll untersucht werden, inwiefern sich die Transportzeit auf die Füllstände der einzelnen Loops im Agentensystem auswirken kann.

Bearbeitungszeiten in den K-Plätzen

Zusätzlich zu den Transportzeiten zu den Scannern wird in den K-Plätzen eine fiktive Bearbeitungszeit der FE, ausgelöst durch den Kommissioniervorgangs des Bedieners von 30 Sekunden pro FE im K-Platz angenommen. Als weitere Beschränkung wird im Simulator maximal eine zu bearbeitende FE pro K-Platzangenommen.

6.1.3 Auslagerfluss des Simulators

Im optimierten Fall transportieren die RBG 1 und RBG 2 im Betrieb der vorliegenden Referenzanlage 2 FEs gleichzeitig. Darüber hinaus kann für den optimierten Fall der maximal erreichbare Durchsatz der einzelnen MFEs auf 120 % erhöht werden, da hier der technisch maximal erreichbare Durchsatz angenommen werden kann.

Dazu werden RBG 1 und RBG 2 der Referenzanlage als diskrete Ereignisse abgebildet. Um den optimierten Fall zu simulieren, erzeugen die RBG 1 und RBG 2 jeweils alle 30 Sekunden maximal zwei FEs mit dem gleichen Ziel und schleusen diese auf die entsprechende Strecke von RBG zu Loop 1 ein.

Ist die maximale Kapazität der entsprechenden Strecke erreicht, können keine neuen FEs mehr erzeugt werden, bis die Strecke wieder freie Kapazitäten hat.

6.1.4 Einlagerflüsse des Simulators

Der Fluss des Wareneingangs an Loop 3 unterliegt uhrzeitabhängigen Schwankungen. Zur möglichst realitätsgetreuen Abbildung wurden hierzu die Wareneingänge von insgesamt 434 Stunden erfasst und in den Simulator übertragen. Tabelle 6-2 zeigt einen Auszug aus den Wareneingangsdaten.

Tabelle 6-2: Auszug aus den Wareneingängen

Stunde	Anzahl erfasste FE
0	57
1	81
2	28
3	60

Wie bei den RBGs kann auch bei den Wareneingangsplätzen der technisch erreichbare Durchsatz von 120 % der Normalleistung als Optimum angenommen werden. Daher erzeugen die Wareneingangsplätze abhängig von der Streckenkapazität jeweils alle 60 Sekunden und 112,5 Sekunden jeweils eine FE und schleusen diese auf die Strecke von WE-Platz zu Loop 3.

Zusätzlich erzeugt die Produktion an Loop 2 jeweils alle 29,5 Sekunden eine FE und das RBG „Produktion“ alle 59 Sekunden jeweils 2 FEs mit dem gleichen Ziel.

6.1.5 Rücklagerfluss des Simulators

Nach der Ankunft im K-Platz wird die ankommende FE wieder zurück zu ihrer Quelle transportiert. Hierbei können sich aufgrund der maximalen Streckenkapazität maximal sieben FEs gleichzeitig in der jeweiligen Strecke von K-Platz zu Loop 3 befinden. Zusätzlich können an den K-Plätzen nur FEs aus Loop 3 in die K-Plätze ausgeschleust werden, wenn auch FEs aus den K-Plätzen in Loop 3 eingeschleust werden.

6.1.6 Testdaten des Simulators

Zur Simulation von Materialflüssen wurden für die Quellen Testdaten in Abhängigkeit der Funktion der Quellen erzeugt. So wurden für die RBG jeweils immer zwei LEs mit der gleichen Zielsenke erstellt, für die WE-Plätze wurden fortlaufend FE-Nummern mit jeweils wechselnden Zielsenken erzeugt. Tabelle 6-3 zeigt einen Auszug der erzeugten Testdaten.

Tabelle 6-3: Auszug der Testdaten

FE-Nummer	Quelle	Ziel
FE1	RBG 1	K-Platz 1
FE2	RBG 1	K-Platz 1
FE3	RBG 2	K-Platz 2
FE4	RBG 2	K-Platz 2
FE5	RBG 1	K-Platz 3
FE6	RBG 1	K-Platz 4
FE7	WE-Platz 1	RBG 1
FE8	WE-Platz 2	RBG 2

Insgesamt wurden für einen Langzeittest des Simulators 50000 Testdatensätze erzeugt. Der Simulator liest zu Beginn alle Testdatensätze ein, speichert diese im Arbeitsspeicher und arbeitet die Daten sequenziell ab.

6.1.7 Validierung des Simulators

Zur Validierung des korrekten Verhaltens des Simulators sowie seiner Komponenten wurden in jedem Schritt der Erweiterung des Simulators die Ausgaben in den Log-Dateien des Simulators sowie die übergebenen Trainingsdaten an die jeweiligen neuronalen Netze nach Ende eines Simulationslaufs überprüft. Im Fehlerfall wurde der Quellcode des Simulators angepasst und der Simulationsprozess erneut gestartet.

6.2 Definition des Agenten

Vor der Umsetzung des Simulators wurden die Zustände, Aktionen und die Belohnung des Agenten definiert. Das genaue Vorgehen zur Definition des Verhaltens des Agenten wird im Kommenden näher erläutert.

6.2.1 Position der Agenten

Da die in 5.4 eingeführten Agenten oft redundante Zustandsräume abbilden und die neuronalen Netze der entsprechenden Agenten somit oft gleich sind, werden in der konkreten Umsetzung des Agentensystems in der Simulation Agenten zusammengefasst. Des Weiteren werden in der vorliegenden Umsetzung des Agentensystems die Agenten für das Ausschleusen von FEs von den Loops in die K-Plätze nicht umgesetzt. Abbildung 6-2 illustriert die umzusetzenden Agentenpositionen im Simulator.

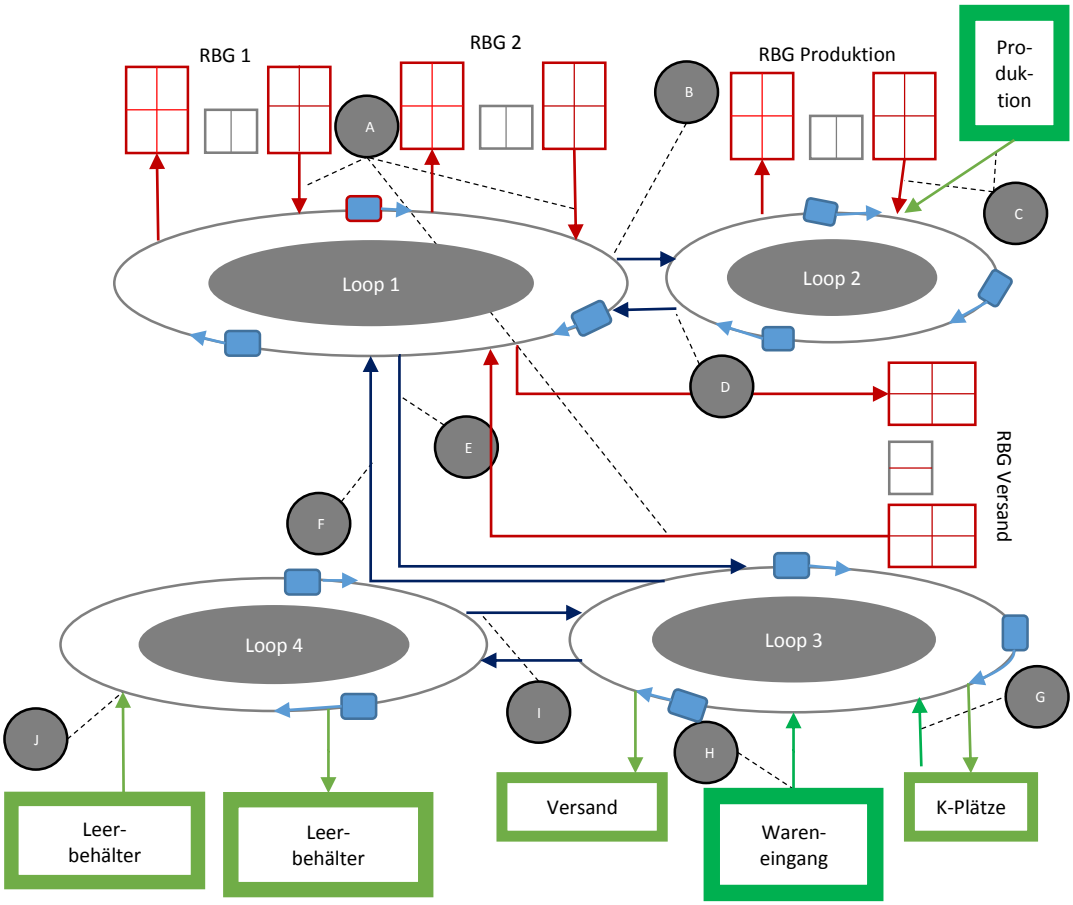

Abbildung 6-2: Agentenpositionen im Simulator

Es ergeben sich somit die folgenden Agenten im Simulator:

- **Agent A:** Einfahrt von den RBG 1, RBG 2 und RBG „Versand“ zu Loop 1
- **Agent B:** Überfahrt von Loop 1 zu Loop 2
- **Agent C:** Einfahrt von Loop 2 vom Aufsetzpunkt in der Produktion und RBG „Produktion“
- **Agent D:** Überfahrt von Loop 2 zu Loop 1
- **Agent E:** Überfahrt von Loop 1 zu Loop 3
- **Agent F:** Überfahrt von Loop 3 zu Loop 1
- **Agent G:** Einfahrt von den K-Plätzen zu Loop 3
- **Agent H:** Einfahrt von den Wareneingangsplätzen zu Loop 3
- **Agent I:** Überfahrt von Loop 4 zu Loop 3
- **Agent J:** Einfahrt von den Leerbehälteraufsetzpunkten zu Loop 4

6.2.2 Zustände und Aktionen

Zur Ermittlung der Eingangsvariablen des Agenten wurden die möglichen Aktionen und Zustände auf Basis der Modelle aus 4.3 bestimmt.

Aktionen des Agenten

Die möglichen Aktionen des Agenten beschränken sich auf zwei Möglichkeiten: „fahren“ und „nicht fahren“.

Zustände des Auslager- und Rücklagerflusses

Aus der Modellierung der Rücklager- und Auslagerflüsse in Abbildung 4-11 lassen sich die unten aufgeführten Zustände ableiten:

- Anzahl der FEs in RBG 1
- Anzahl der FEs in RBG 2
- Anzahl der FEs in RBG 3
- Anzahl der FEs in Loop 1
- Anzahl der FEs in Loop 2
- Anzahl der FEs in Loop 3
- Anzahl der FEs in K-Platz 1 eingehend
- Anzahl der FEs in K-Platz 2 eingehend
- Anzahl der FEs in K-Platz 3 eingehend
- Anzahl der FEs in K-Platz 4 eingehend
- Anzahl der FEs in K-Platz 1 ausgehend
- Anzahl der FEs in K-Platz 2 ausgehend
- Anzahl der FEs in K-Platz 3 ausgehend
- Anzahl der FEs in K-Platz 4 ausgehend
- Anzahl der ausgehenden FEs aus dem Wareneingangsplatz 1
- Anzahl der ausgehenden FEs aus dem Wareneingangsplatz 2
- Anzahl aller FEs im MFS mit dem aktuellen Ziel

Hinzu kommt die längste Verweildauer aus allen im aktuellen MFE befindlichen FEs. Dieser Zustand soll dem Agenten eine gewisse Zielpriorität vermitteln und ihn zusätzlich für unnötig lange Aufenthalte bestrafen.

Um die Dimension des Problems und die damit verbundenen Eingänge des Agenten zu verringern, können die möglichen Zustände zu

- Summe der FEs in dem RBG 1 und 2
- Anzahl der FEs in RBG 3
- Anzahl der FEs in Loop 1
- Anzahl der FEs in Loop 2
- Anzahl der FEs in Loop 3
- Summe der ausgehenden FEs in allen K-Plätzen
- Summe der eingehenden FEs in allen K-Plätzen
- Summer der ausgehenden FEs aus dem Wareneingang
- Anzahl aller FEs im MFS mit dem aktuellen Ziel
- Längste Verweildauer aus allen im aktuellen MFE befindlichen FEs

zusammengefasst werden. Die Zustände der RBG 1 und RBG 2 lassen sich zusammenfassen, da sie am gleichen Folge-MFE angeschlossen sind. Ferner sind sie aufgrund der gleichen Taktung gleichberechtigt. Die Zustände der K-Plätze können zusammengefasst werden, da durch die physische Verbindung der K-Plätze mit Loop 3 lediglich der Zustand aller K-Plätze von Interesse ist.

Zustände des Einlagerflusses

Der Einlagerfluss verhält sich ähnlich wie der Rücklagerfluss, er weicht nur in der Startposition ab. Daher können die meisten Zustände übernommen werden. Es kommt lediglich der Zustand des Wareneingangs hinzu. Und es ergeben sich die folgenden Zustände:

- Summe der FEs in dem RBG 1 und 2
- Anzahl der FEs in RBG 3
- Anzahl der FEs in Loop 1
- Anzahl der FEs in Loop 2

- Anzahl der FEs in Loop 3
- Summe der ausgehenden FEs aus dem Wareneingang
- Summe der ausgehenden FEs in allen K-Plätzen
- Summe der eingehenden FEs in allen K-Plätzen
- Anzahl aller FEs im MFS mit dem aktuellen Ziel
- Längste Verweildauer aus allen im aktuellen MFE befindlichen FEs

Auch die Zustände des Einlagerflusses können zusammengefasst werden, da hier ebenso der Gesamtzustand aller Ein- bzw. Ausgänge für die Bewertung des aktuellen Zustands ausreichend ist.

6.2.3 Eingangsvariablen der Agenten

Auf Grundlage der in 6.2.1 eingeführten Zustände der einzelnen Materialflüsse lassen sich für die einzelnen Agenten aus 5.4.1 die Eingangsvariablen zur Approximation der Q-Funktion ableiten. Alle Agenten erhalten als Standardeingangsvariablen

- die auszuführende Aktion
- die längste Verweildauer aus allen im aktuellen MFE befindlichen FEs
- die Anzahl aller FEs im MFS mit dem aktuellen Ziel und
- die Anzahl der FEs im aktuellen MFE.

Nachfolgend werden die zusätzlich benötigten Eingangsvariablen der einzelnen Agenten aufgeführt.

- **Agent A: Einfahrt von den RBG 1, RBG 2 und RBG „Versand“ zu Loop 1**
  - Anzahl der FEs in Loop 1
  - Anzahl der FEs in Loop 2
  - Anzahl der FEs in Loop 3
  - Summe der ausgehenden FEs in allen K-Plätzen
  - Summe der eingehenden FEs in allen K-Plätzen
  - Summe der ausgehenden FEs aus dem Wareneingang

- **Agent B: Überfahrt von Loop 1 zu Loop 2**
  - Anzahl der FEs in Loop 1
  - Anzahl der FEs in Loop 2
  - Summe der ausgehenden FEs aus der Produktion
  - Summe der eingehenden FEs in den RBG 1 und RBG 2

- **Agent C: Einfahrt von Loop 2 vom Aufsetzpunkt in der Produktion und RBG „Produktion“**
  - Anzahl der FEs in Loop 1
  - Anzahl der FEs in Loop 2
  - Summe der eingehenden FEs in die Produktion
  - Summe der ausgehenden FEs aus der Produktion
  - Summe der eingehenden FEs in den RBG 1 und RBG 2

- **Agent D: Überfahrt von Loop 2 zu Loop 1**
  - Anzahl der FEs in Loop 1
  - Summe der eingehenden FEs in den RBG 1 und RBG 2

- **Agent E: Überfahrt von Loop 1 zu Loop 3**
  - Anzahl der FEs in Loop 3
  - Summe der ausgehenden FEs in allen K-Plätzen
  - Summe der eingehenden FEs in allen K-Plätzen
  - Summe der ausgehenden FEs aus dem Wareneingang

- **Agent F: Überfahrt von Loop 3 zu Loop 1**
  - Anzahl der FEs in Loop 1
  - Anzahl der FEs in Loop 2
  - Summe der ausgehenden FEs aus der Produktion
  - Summe der eingehenden FEs in den RBG 1 und RBG 2
  - Summe der ausgehenden FEs aus den RBG 1 und RBG 2

- **Agent G: Einfahrt von den K-Plätzen zu Loop 3**
  - Anzahl der FEs in Loop 1
  - Anzahl der FEs in Loop 2
  - Anzahl der FEs in Loop 3
  - Summe der ausgehenden FEs in allen K-Plätzen
  - Summe der eingehenden FEs in allen K-Plätzen
  - Summe der ausgehenden FEs aus dem Wareneingang
  - Summe der eingehenden FEs in den RBG 1 und RBG 2

- **Agent H: Einfahrt von den Wareneingangsplätzen zu Loop 3**
  - Anzahl der FEs in Loop 1
  - Anzahl der FEs in Loop 2
  - Anzahl der FEs in Loop 3
  - Summe der ausgehenden FEs in allen K-Plätzen
  - Summe der eingehenden FEs in allen K-Plätzen
  - Summe der ausgehenden FEs aus dem Wareneingang
  - Summe der eingehenden FEs in den RBG 1 und RBG 2

- **Agent I: Überfahrt von Loop 4 zu Loop 3**
  - Anzahl der FEs in Loop 1
  - Anzahl der FEs in Loop 3
  - Summe der eingehenden FEs in den RBG 1 und RBG 2

- **Agent J: Einfahrt von den Leerbehälteraufsetzpunkten zu Loop 4**
  - Anzahl der FEs in Loop 4
  - Anzahl der FEs in Loop 3
  - Anzahl der FEs in Loop 1
  - Summe der eingehenden FEs in den RBG 1 und RBG 2

6.2.4 Belohnung des Agenten

Der Agent soll nach Ankunft der FE in der Zielsenke für seine mit der FE verbundenen Entscheidungen und Aktionen belohnt bzw. bestraft werden. Hierzu wurden verschiedene Ansätze entworfen, die an dieser Stelle diskutiert werden sollen. Die Transportzeiten sind, falls nicht anders angegeben, in Sekunden zu verstehen.

- **Belohnung auf Basis des Durchsatzes der letzten Stunde**

  Der Agent wird nach Ankunft einer FE auf Basis des Durchsatzes der letzten Stunde belohnt. Eine Belohnung des Agenten auf Basis des Durchsatzes an den Senken ist als Belohnung nicht ausreichend. Der Agent wird zwar für die Ankunft von möglichst vielen FEs pro Stunde belohnt, jedoch wird die eigentliche Transportzeit einer FE nicht in Betracht gezogen. Der Agent kann zwar viel Durchsatz produzieren, jedoch können sich einzelne FEs unendlich lange in den MFEs aufhalten.

- **Belohnung auf Basis der Transportdauer der FE und der theoretisch erreichbaren Transportzeit**

  Der Agent wird nach Ankunft einer FE in der Zielsenke mittels der Belohnung

  $$r = \frac{1}{t_{tr}} \text{ mit } t_{tr} = \frac{aktuelle\ Transportdauer}{theoretisch\ erreichbare\ Zeit} \tag{6.2}$$

  belohnt. Diese Art der Belohnung setzt einen gewissen Kenntnisstand des Agenten über seine Umwelt voraus. Hierzu müssten dem Agenten spezielle Kontextinformationen mitgeteilt werden, die vorab experimentell bestimmt werden müssen.

- **Belohnung auf Basis der Transportdauer der FE und des Durchsatzes der letzten Stunde an der Zielsenke**

  Der Agent wird nach Ankunft einer FE in der Zielsenke durch die Belohnung

  $$r = \frac{1}{Transportdauer} * \text{Durchsatz der letzten Stunde an der Zielsenke} \tag{6.3}$$

  belohnt. Diese Belohnung verspricht eine bessere Art der Belohnung des Agenten, da er hier in Abhängigkeit der Transportdauer und des Durchsatzes an der entsprechenden Zielsenke belohnt wird.

  Durch erste Experimente konnte gezeigt werden, dass der Agent für das Verlassen von Loop 1 und Loop 3 eine zusätzliche Belohnung r_{in} erhalten soll. Hierzu wurden die nachfolgenden vier Experimente mit unterschiedlichen Werten für r_{in} und (6.3) erweitert um r_{in}zu

  $$r = \frac{1}{Transportdauer} * \text{Durchsatz der letzten Stunde an der Zielsenke} + r_{in} \tag{6.4}$$

  durchgeführt.

Zur Beurteilung der unterschiedlichen Faktoren für r_{in} aus (6.4) wurde die zusätzliche Belohnung r_{in} für:

- Testlauf 1: $r_{in} \coloneqq 0$
- Testlauf 2: $r_{in} \coloneqq 1$
- Testlauf 3: $r_{in} \coloneqq 5$
- Testlauf 4: $r_{in} \coloneqq 10$

festgelegt. Abbildung 6-3 zeigt einen Vergleich der soeben beschriebenen zusätzlichen Belohnungen r_{in} für die in 6.2.1eingeführten Agenten *A „Einfahrt von den RBG 1 und RBG 2 in Loop1“, E „Überfahrt von Loop 1 nach Loop 3“ , F „Überfahrt von Loop 3 nach Loop 1“* und *G „Einfahrt von den K-Plätzen zu Loop 3“*.

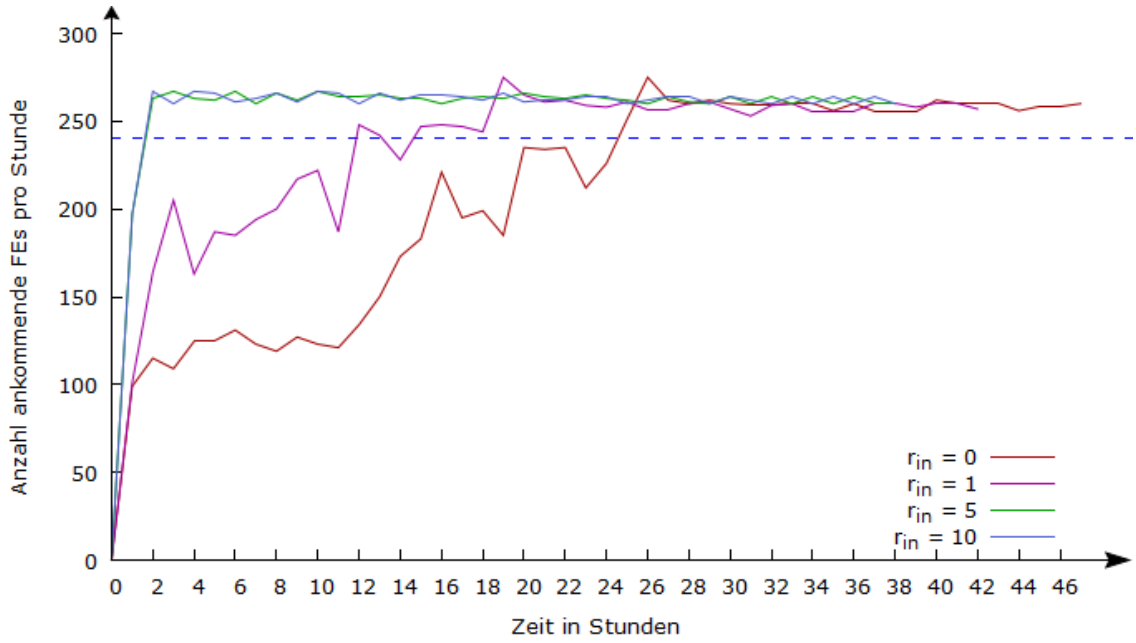

Abbildung 6-3: Vergleich der Auswirkung von unterschiedlichen Belohnungen auf den Gesamtdurchsatz an den K-Plätzen

Hierbei wurden bei allen Testläufen insgesamt 10000 FEs an den K-Plätzen abgenommen. Die Exploration-Wahrscheinlichkeit wurde initial auf 50 %, der Parameter $\alpha = 0{,}1$ und der Parameter $\gamma = 0{,}6$ gesetzt. Die Exploration-Wahrscheinlichkeit wurde bei allen Testfällen ab 1000 an den K-Plätzen ankommenden FEs nach jeweils 100 weiter ankommenden FEs um 1 % gesenkt. Alle Testläufe erreichen den Zieldurchsatz an den K-Plätzen von 240 FEs pro Stunde (vgl. Tabelle 4-2).

Jedoch kann, wie in den Testläufen 3 und 4 gezeigt, durch eine Definition einer zusätzlichen Belohnung r_{in} der Agent schneller in für ihn interessante Zustände geführt und der gewünschte Zieldurchsatz schneller erreicht werden. Da die Durchsätze der Testläufe 3 und 4 ein ähnliches Verhalten zeigen, wird für das weitere Vorgehen in dieser Arbeit die zusätzliche Belohnung r_{in} auf den Wert 5 festgelegt.

6.2.5 Neuronales Netz des Agenten

Wie in Kapitel 5 beschrieben, sollen neuronale Netze zur Approximation der Q-Funktion eingesetzt werden. Die neuronalen Netze wurden in der aktuellen Umsetzung alle manuell mit dem grafischen Editor des MemBrain-Werkzeugs erstellt. Abbildung 6-4 zeigt exemplarisch das neuronale Netz zur Bewertung des *Agenten A*.

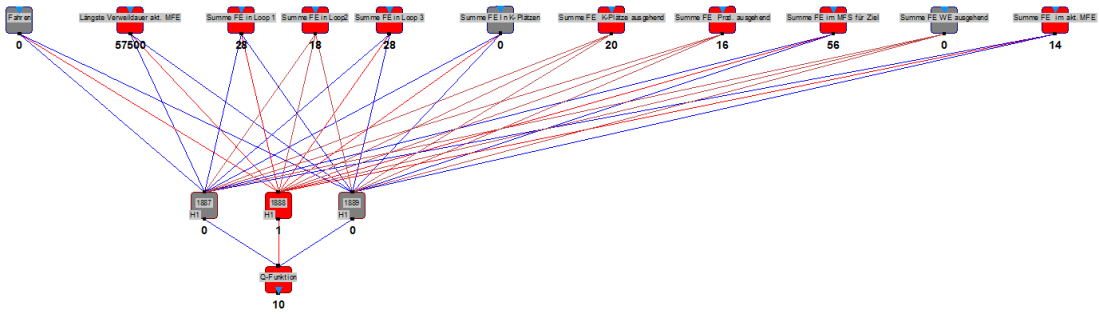

Abbildung 6-4: Beispiel für ein neuronales Netz eines Agenten

Die Neuronen sind als Vierecke und die Verbindungen als Linien zwischen den Neuronen dargestellt. Die schwarzen Zahlen unterhalb der Neuronen stellen die Normalisierungsgrenzen dar. Bei der Erstellung der Neuronen und Verbindungen wurden die vom MemBrain-Werkzeug vorgeschlagenen Standardwerte[10] verwendet. Die neuronalen Netze der anderen Agenten besitzen alle eine ähnliche Topologie wie das Netz aus Abbildung 6-4, daher wird an dieser Stelle auf eine Abbildung der restlichen Netze verzichtet.
Das neuronale Netz aus Abbildung 6-4 besteht aus einer Eingabeschicht mit elf Eingängen, einer versteckten Schicht mit drei Neuronen und einer Ausgabeschicht mit einem Ausgabeneuron für die Ausgabe des Q-Wertes.

Die Anzahl der Neuronen in der Eingabeschicht resultiert aus der Anzahl der Eingabevariablen des Agenten.

Für die Anzahl der Neuronen in der versteckten Schicht wurden drei Neuronen gewählt, da hier ein Kompromiss zwischen dem Over Fitting, bei dem das neuronale Netz die übergebenen Werte aufgrund von zu vielen versteckten Neuronen einfach auswendig lernt, und dem Under Fitting, bei dem das Netz die gewünschte Funktion aufgrund von zu wenig vorhandenen Informationen innerhalb des Netzes nicht ausreichend approximieren kann, eines Netzes gefunden werden muss.

6.2.6 Normalisierung der Neuronen des neuronalen Netzes

Um Fehler in der Ein- und Ausgabe des neuronalen Netzes zu vermeiden, müssen die Eingangs- und Ausgangswerte vor der Übergabe an das neuronale Netz normalisiert werden. Hierzu können im grafischen Netzeditor von MemBrain die Normalisierungsgrenzen vergeben werden.

Normalisierungsgrenzen der Eingabeneuronen

Die Werte der Eingangsneuronen werden vor der Übergabe an das neuronale Netz auf den Wertebereich der entsprechen Aktvierungsfunktion normalisiert. Tabelle 6-4 zeigt die Eingangsneuronen und ihre Wertebereiche für die Normalisierung.

[10] Vgl. [Tho16] für Standardeinstellungen der einzelnen Neuronen.

Tabelle 6-4: Eingangsneuronen und ihre Minima und Maxima

Eingangsneuron	Minimum	Maximum
Fahren	0	1
Längste Verweildauer im aktuellen MFE	0	$5 * Transportdauer\ aktuelles\ MFE$
Summe FEs in Loop 1	0	28
Summe FEs in Loop 2	0	18
Summe FEs in Loop 3	0	28
Summe FEs in den RBGs eingehend	0	20
Summe FEs in den RBGs ausgehend	0	20
Summe FEs in den K-Plätzen eingehend	0	28
Summe FEs in den K-Plätzen ausgehend	0	28
Summe FEs im MFS für Ziel	0	28
Summe FEs im aktuellen MFE	0	56

Mit Ausnahme der längsten Verweildauer können alle Maxima statisch, durch eine Ableitung der Maxima aus den technischen Vorgaben des MFS, bestimmt werden. Das Maximum der Transportdauer im aktuellen MFE muss dynamisch während der Laufzeit angepasst werden, hierzu bietet die MemBrain-DLL-Schnittstelle eine einfache Möglichkeit.

Normalisierungsgrenze des Ausgabeneurons

Das zur Approximation der Q-Funktion verwendete Netz besteht lediglich aus einem einzigen Ausgabeneuron. Erste Versuche haben gezeigt, dass auch hier die Grenzen dynamisch zur Laufzeit angepasst werden müssen, da es sonst zu fehlerhaften Approximationen der Q-Funktion kommen kann.
Zur ersten Abschätzung der Normalisierungsgrenzen wurde, für das im Folgenden dargestellte Experiment, für das Maximum der Normalisierungsgrenze des Ausgabeneurons der Wert 10 gewählt. Abbildung 6-5 illustriert den Unterschied im Verlauf der Q-Funktion zwischen dem tatsächlichen und dem durch das neuronale Netz approximierten Ergebnis der Q-Funktion über insgesamt 2524 Trainingsdatensätze des *Agenten A*.

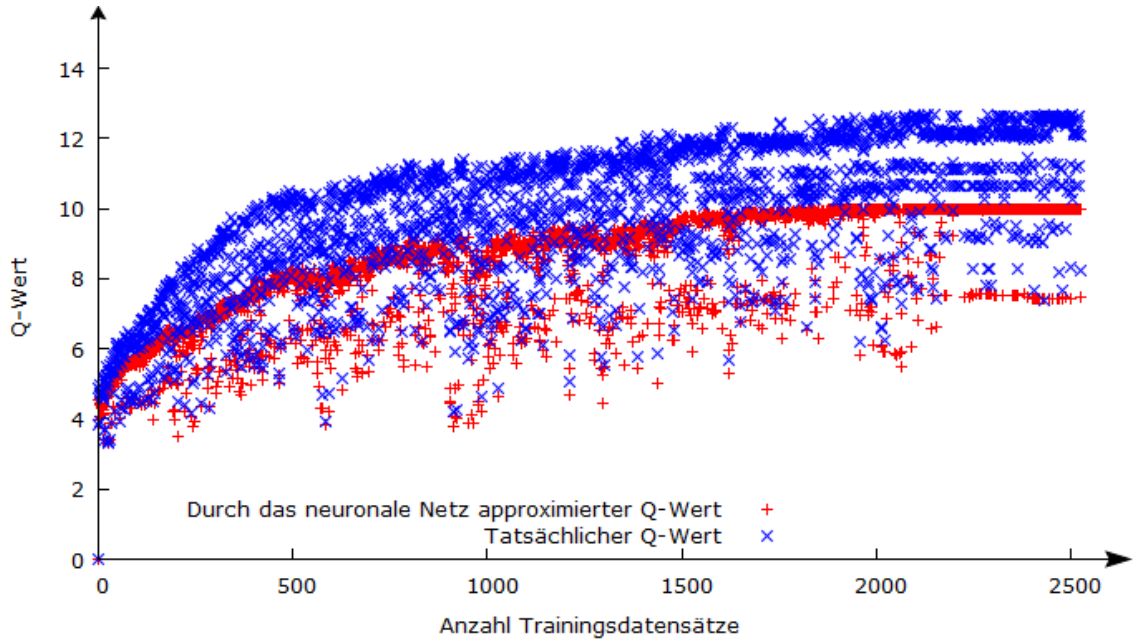

Abbildung 6-5: Verlauf der tatsächlichen (blau) und approximierten Q-Funktion (rot)

Aus Abbildung 6-5 ist eine Konvergenz des approximierten Q-Werts des Ausgabeneurons um die obere Grenze von zehn zu erkennen. Bei einem Vergleich der tatsächlich berechneten Werte mit den approximierten Werten fällt jedoch die starke Streuung der tatsächlichen Werte auf. Da die approximierten Werte sehr häufig um das Maximum der Normalisierungsgrenze konvergieren, kann es im Verlauf der Simulation zu Fehlbewertungen des Zustand-Aktions-Paares durch das neuronale Netz und zu Fehlentscheidungen des Agenten kommen. Auf Grundlage des hier dargestellten Experiments werden die Normalisierungsgrenzen des Ausgabeneurons initial auf

- **Minimum:** 0
- **Maximum:** 30

festgelegt. Durch das hohe Maximum von 30 soll eine zu häufige Neujustierung der oberen Normalisierungsgrenze verhindert werden. Durch eine Normalisierung der entsprechenden Werte direkt vor der Übergabe an das neuronale Netz bleiben diese unberührt und können im Falle einer Anpassung einfach erneut normalisiert werden. Um den neuen Definitionsbereich anzulernen, kann das neuronale Netz mit den vorhandenen Trainingsdaten im neuen Wertebereich trainiert werden. [Tho16]

6.2.7 Training des Agenten

Der Agent wird immer nach Ankunft der FE in der Zielsenke mit den vorhandenen Trainingsdatensätzen trainiert. Zusätzlich werden die Trainingsdatensätze des Agenten nach dem FIFO-Prinzip aktualisiert. Hierzu wurde zu Beginn die maximale Anzahl an Trainingsdatensätzen im Fundus des Agenten auf maximal 50000 festgelegt, nach Erreichen der Obergrenze wird jeweils der älteste Trainingsdatensatz gelöscht.

6.3 Beschreibung der MemBrain-DLL

Um mit den Funktionen der MemBrain-DLL zu arbeiten, muss diese zuerst in den Quellcode integriert werden. Hierzu kann die MemBrain-DLL in den Quellcode des Simulators eingebunden werden. Danach können alle Funktionen der MemBrain-DLL verwendet werden. Sowohl die in der vorliegenden Arbeit verwendeten Befehle zum Ausführen der MemBrain-DLL als auch der grundsätzliche Aufbau der MemBrain-DLL werden im kommenden Abschnitt Schritt für Schritt kurz erläutert. Genauere Beschreibungen aller Befehle und der MemBrain-DLL können [Tho16] entnommen werden.

6.3.1 Aufbau der MemBrain-DLL

Mit der MemBrain-DLL können beliebig viele Netze, Trainingsdatensätze und die von MemBrain unterstützen Lernregeln verwaltet und benutzt werden. MemBrain unterscheidet hierzu die Netzverwaltung, Trainingsdatensatzverwaltung und die globale Verwaltung der Lernregeln.

Die von der MemBrain-DLL verwalteten neuronalen Netze werden über ihre eindeutige Nummer durch die MemBrain-DLL angesprochen.

Die Trainingsdatensätze werden zu sog. Lessons gruppiert und können durch ihre eindeutige Kennung einem passenden Netz zum Training vorgelegt werden. Die Sammlung der verfügbaren Lernregeln wird zu Beginn des Programms einmalig geladen, danach kann die gewünschte Lernregel geladen werden.

6.3.2 Verwendung der MemBrain-DLL

Die MemBrain-DLL kann als externe DLL in ein Programm eingebunden werden. Zur Bedienung und Administration von neuronalen Netzen bietet die MemBrain-DLL u. a. die folgenden Funktionen:

Tabelle 6-5: Auszug der Befehle der MemBrain-DLL

Funktion	Befehl
Laden eines neuronalen Netzes	*_MB_LoadNet(“myNet.mbn“);*
Hinzufügen eines Netzes	*_MB_AddNet();*
Randomisieren eines Netzes	*_MB_RandomizeNet();*
Zurücksetzen eines Netzes	*_MB_ResetNet();*
Laden der Lernregeln	*_MB_LoadTeacherFile(“myTeacher.mbn“);*

Eine detaillierte Beschreibung der Befehle zur Verwendung von neuronalen Netzen durch Einsatz der MemBrain-DLL kann Anhang A-1 entnommen werden.

Auf Basis der hier beschriebenen Umsetzung des Lösungsansatzes werden im folgenden Kapitel die einzelnen Simulations- und Analyseschritte näher erläutert. Abschließend wird ein Integrationsvorschlag des Lösungsansatzes in die Software viadat diskutiert.

7 ANALYSE DES LÖSUNGSANSATZES

Für eine erste Bewertung der Anwendung des Agentensystems werden in den folgenden Abschnitten verschiedene Materialflüsse und die damit verbundenen Agenten genauer betrachtet. In einem ersten Schritt soll die generelle Fähigkeit des Agentensystems zur Steuerung des Auslagerflusses betrachtet werden.

Im nächsten Schritt werden der Auslager- und Rücklagerfluss näher betrachtet. Hierbei soll in jedem Schritt die generelle Adaption des Lösungsvorschlags an die gegebene Problemstellung näher untersucht werden.

Im ersten Schritt wurde die Anwendbarkeit des Reinforcement-Learning-Ansatzes auf die Optimierung des Materialflusses untersucht. Im Anschluss wurde die Simulation iterativ um weitere Bestandteile und ggf. Agenten erweitert.

Zur Beurteilung der Leistung des Agentensystems wurden zum einen der Durchsatz und zum anderen die Transportzeiten analysiert.

7.1 Risiken des Lösungsansatzes

Durch die Verwendung von zufälligen Entscheidungen in allen Agenten kann es zu einer Blockade des kompletten MFS, dem sog. Deadlock, kommen. Ein Beispiel für einen Deadlock ist die Belegung von Loop 1 und Loop 3 mit FEs, die als Ziel die K-Plätze 1 bis 4 haben. Zusätzlich sind alle K-Plätze mit ein- und ausgehenden FEs belegt und können somit keine neuen FEs mehr abnehmen und auch keine FEs mehr in Loop 3 einschleusen. Im bisherigen Stand der Simulation kann dieser Situation nur durch einen Neustart der Simulation entgegengewirkt werden. Ein besserer Ansatz wäre hier, direkt den entsprechenden Agenten für die zur Blockade führenden Entscheidungen über eine negative Belohnung p zu bestrafen. Somit sollte der Agent lernen die entsprechenden Zustände zu vermeiden.

7.1.1 Verfrühte Anpassung der Exploration-Wahrscheinlichkeit

Durch eine zu frühe Anpassung der Exploration-Wahrscheinlichkeit kann es zu einer Approximation einer schlechten Strategie im Hinblick auf Optimierung des Materialflusses kommen. Abbildung 7-1 zeigt exemplarisch den Verlauf des Gesamtdurchsatzes an allen vier K-Plätzen bei einer Senkung der Exploration-Wahrscheinlichkeit um 1 % ab 20 an den K-Plätzen ankommenden FEs in Schritten von jeweils 10 FEs. In dem in Abbildung 7-1 gezeigten Experiment wurde die Anfangswahrscheinlichkeit für die Exploration auf 50 % gesetzt und die Parameter $\alpha = 0{,}1$ und $\gamma = 0{,}6$ gewählt, insgesamt wurden 2412 FEs an den RBG 1 und RBG 2 erzeugt, durch die K-Plätze abgenommen und zurück an das entsprechende Quell-RBG geschickt.

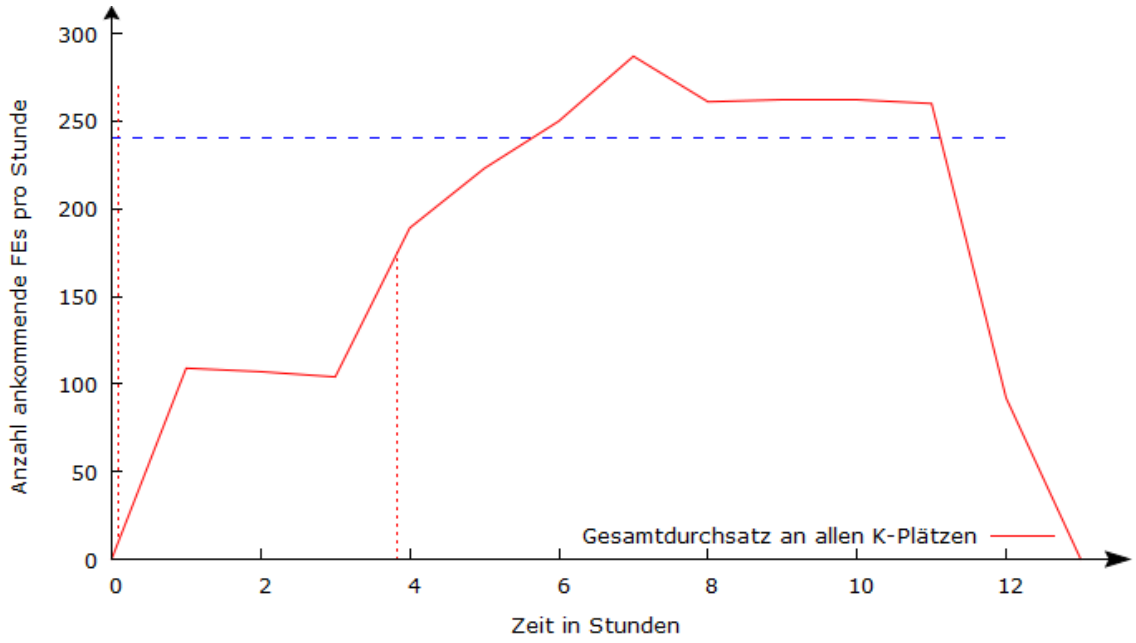

Abbildung 7-1: Verlauf des Durchsatzes bei zu früher Anpassung der Exploration-Wahrscheinlichkeit

Die erste rot gestrichelte Linie markiert den ersten Zeitpunkt der Reduzierung der Exploration-Wahrscheinlichkeit, die zweite das Erreichen von 0 % Exploration-Wahrscheinlichkeit, die blau gestrichelte Linie den Zieldurchsatz von 240 FE/h an allen K-Plätzen (vgl. Tabelle 4-2).

Aus dem Graphen der hier gezeigten Simulation ist zu erkennen, dass das zu frühe Herabsetzen der Exploration-Wahrscheinlichkeit zu einem kompletten Einbruch des Durchsatzes an den K-Plätzen führen kann. Die Analyse des hier dargestellten Testfalls ergab, dass der für das Einschleusen von FEs in Loop 1 verantwortliche *Agent A* (vgl. 5.4.1) die zu bewertenden Zustände und Aktionen falsch approximierte. Aufgrund dessen führte der Agent einfach immer die Aktionen „fahren“ und „nicht fahren“ nacheinander aus. Dieses Vorgehen des Agenten führte zu einem kompletten Durchsatzeinbruch an den Senken.

7.2 Trainingszeiten des neuronalen Netzes

Für die Verwendung des Agentensystems im Live-Betrieb eines MFS ist die effiziente Funktion des MemBrain-Werkzeugs unabdingbar. Hierzu wurden die Trainingszeiten für jeweils eine Epoche[11] durch Verwendung der MemBrain-DLL im Simulator mit verschiedener Anzahl an Trainingsdatensätzen gemessen. Das Ergebnis der Analyse kann Tabelle 7-1 entnommen werden.

Tabelle 7-1: Trainingszeiten des MemBrain-Werkzeugs

Anzahl Trainingsdatensätze	Trainingszeit pro Epoche
50000	1 ms ~ 10 ms
75000	14 ms ~ 544 ms
100000	14 ms ~ 600 ms

[11]Epoche: Kompletter Durchlauf aller Trainingsdaten (vgl. 5.2.2)

Als Ergebnis der Trainingszeitenanalyse werden für den folgenden Verlauf der vorliegenden Arbeit 50000 Trainingsdatensätze als Obergrenze für die maximale Anzahl an für den Agenten verfügbaren Trainingsdaten festgelegt, da die Trainingszeiten von maximal 10 ms für eine Epoche in den geforderten Reaktionszeiten von maximal 50 ms liegen.

7.3 Analyse des Auslagerflusses

Im ersten Schritt wurden zur Überprüfung der Funktionalität des Lösungsansatzes die für die Einfahrt in Loop 1 und die Überfahrt von Loop 1 zur Loop 3 zuständigen Agenten A und E implementiert. Zusätzlich wurde für die zwei implementierten Agenten die erweiterte Belohnung

$$r = \frac{1}{Transportdauer} * \text{Durchsatz der letzten Stunde an der Zielsenke} + r_{in}$$

aus (6.4) verwendet. Zur Beurteilung der Qualität der beiden Agenten werden in den kommenden Abschnitten

- der Durchsatz pro K-Platz
- die Transportzeiten sowie
- die Auslastung der Loop 1 und Loop 3

untersucht.

In allen Simulationen markiert die blau gestrichelte Linie den Zieldurchsatz von 60 FE/h (vgl. Abschnitt 6.1.3) pro K-Platz.

7.3.1 Anpassung der Exploration-Wahrscheinlichkeit

Da sich ein konkreter Zeitwert für die Anpassung der Exploration-Wahrscheinlichkeit, wie von [Ert13] vorgeschlagen, nicht ohne Weiteres festlegen lässt, wird in dem vorgeschlagenen Lösungsansatz die Ankunft einer noch genauer zu bestimmenden Anzahl an FEs an den K-Plätzen als zeitliches Ereignis für die Herabsenkung der Exploration-Wahrscheinlichkeit verwendet. Zur Definition eines Startwertes für das Senken der Exploration-Wahrscheinlichkeit wurden verschiedene Startparameter in der Simulation getestet.

Bei der Simulation des Auslagerflusses wurde ab 500 an den K-Plätzen ankommenden FEs in Schritten von jeweils 100 zusätzlich an den K-Plätzen ankommenden FEs die Exploration-Wahrscheinlichkeit um 1 % gesenkt. Der Anfang und das Ende der Änderungen der Exploration-Wahrscheinlichkeit sind in den entsprechenden Abbildungen durch senkrechte rot gepunktete Linien markiert.

7.3.2 Durchsatzanalyse

Zur Beurteilung der Qualität des Agentensystems wurden zwei Simulationen durchgeführt. Simulation 1 wurde als idealisierte Referenzsimulation ohne das Agentensystem durchgeführt. Die idealisierte Referenzsimulation dient zum einen als Referenz für die spätere Bewertung der Qualität des Agentensystems, zum anderen gibt sie Anhaltspunkte für das allgemeine Verhalten des Simulators. Die Referenzsimulation lässt, da sie ohne andere Materialflüsse ausgeführt wird, schon Rückschlüsse auf das gewünschte Zielverhalten des Lösungsansatzes zu.

Die Referenzsimulation ohne Agentensystem wurde mit 10000 FEs und über einen Zeitraum von insgesamt 38 Simulationsstunden ausgeführt.

Die Simulationen mit Agentensystem wurden mit den in 6.3 eingeführten Startparametern und schrittweiser Anpassung der Exploration-Wahrscheinlichkeit ausgeführt. Beide Simulationen mit Agentensystem wurden jeweils mit und ohne füllstandsabhängiger Transportzeit erstellt und erstrecken sich über einen Zeitraum von jeweils 38 Stunden bzw. 10000 beförderte FEs.

Abbildung 7-2 zeigt den Verlauf des Durchsatzes an den K-Plätzen ohne Einsatz des Agentensystems.

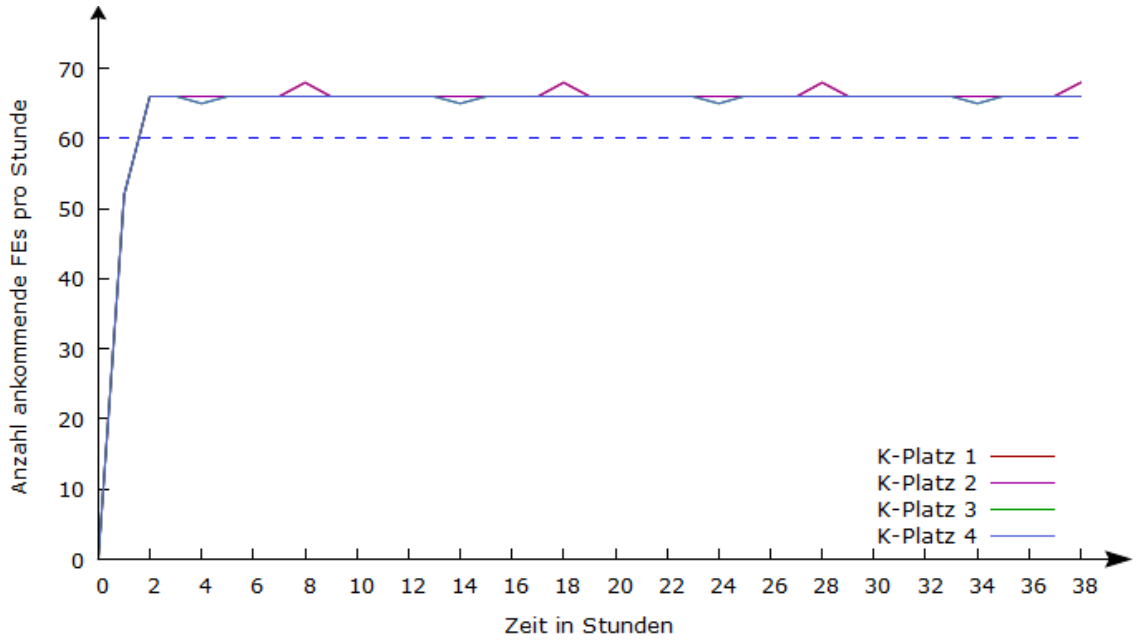

Abbildung 7-2: Verlauf des Durchsatzes an den K-Plätzen ohne Einsatz des Agentensystems

Der weitestgehend konstante Verlauf des Durchsatzes an allen vier K-Plätzen ist aus Abbildung 7-2 ersichtlich. Die minimalen Schwankungen des Durchsatzes an den K-Plätzen in Abbildung 7-2 sind durch ein nicht deterministisches Verhalten innerhalb des Simulators zu erklären. Abbildung 7-3 zeigt den Verlauf des Durchsatzes an den K-Plätzen mit Einsatz des Agentensystems.

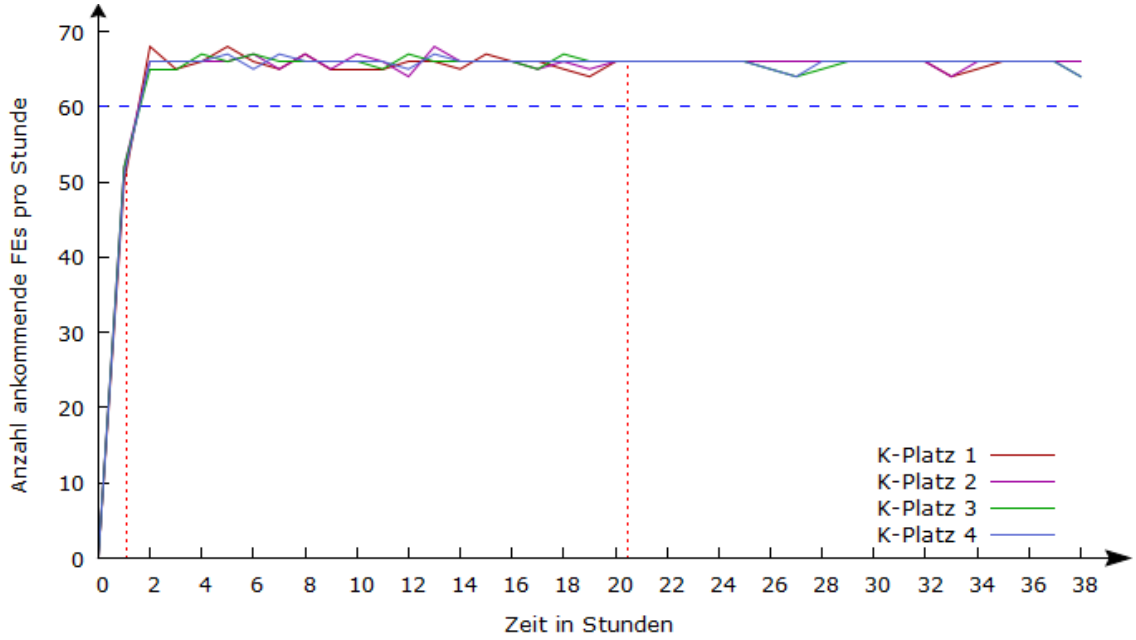

Abbildung 7-3: Verlauf des Durchsatzes mit Einsatz des Agentensystems ohne füllstandsabhängige Transportzeit

Aus Abbildung 7-3 ist die Lernkurve des Agenten ersichtlich. In der Exploration-Phase der Simulation oszillieren die einzelnen Durchsätze noch um den maximal erreichbaren Durchsatz. Nach Erreichen der Exploitation-Phase sind die Schwankungen jedoch konstant und lassen sich ebenfalls durch Varianzen in der Abbildung der einzelnen Ereignisse im Simulator erklären. Zum Vergleich zeigt Abbildung 7-4 den Verlauf des Durchsatzes an den K-Plätzen mit füllstandsabhängiger Transportzeit.

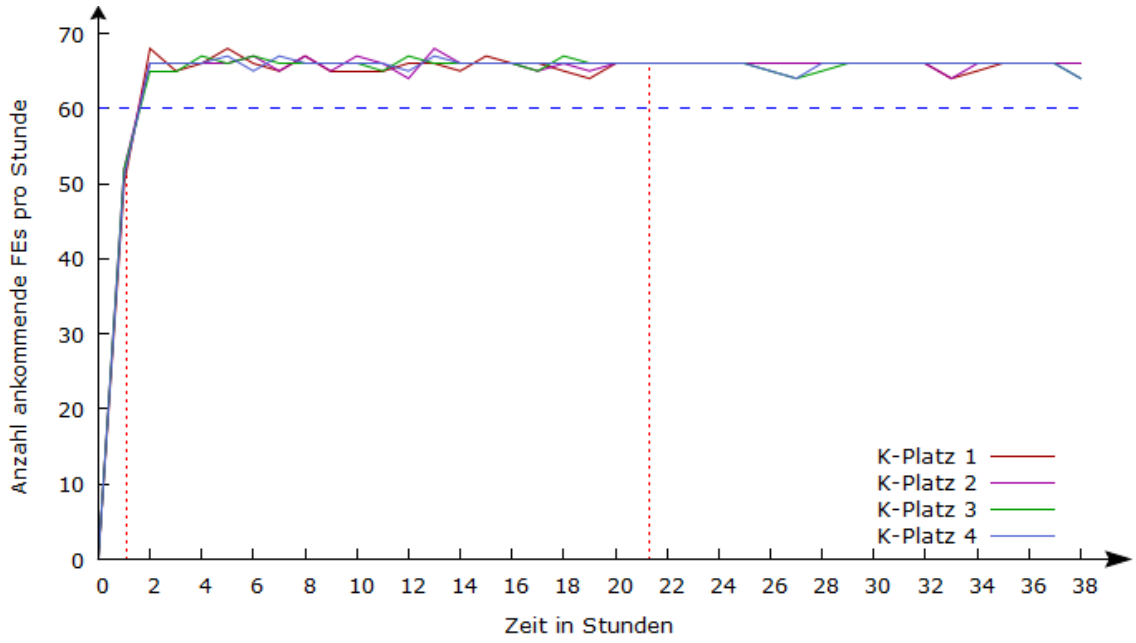

Abbildung 7-4: Verlauf des Durchsatzes mit Einsatz des Agentensystems mit füllstandsabhängiger Transportzeit

Ein Vergleich der Durchsätze aus Abbildung 7-3 und Abbildung 7-4 zeigt keine signifikanten Unterschiede in den Durchsätzen an den K-Plätzen. In beiden Versuchen oszilliert der Durchsatz in den K-Plätzen während der Exploration-Phase um den maximal möglichen Durchsatz.

7.3.3 Analyse der Transportzeiten

Zusätzlich zur Durchsatzanalyse sollen an dieser Stelle die Transportzeiten von den RBG 1 und RBG 2 zu den K-Plätzen betrachtet werden. Zur Analyse der Transportzeiten werden die durchschnittlichen Transportzeiten der ersten Simulation mit denen der zweiten Simulation verglichen. Abbildung 7-5 zeigt den Verlauf der durchschnittlichen Transportzeiten ohne Agentensystem.

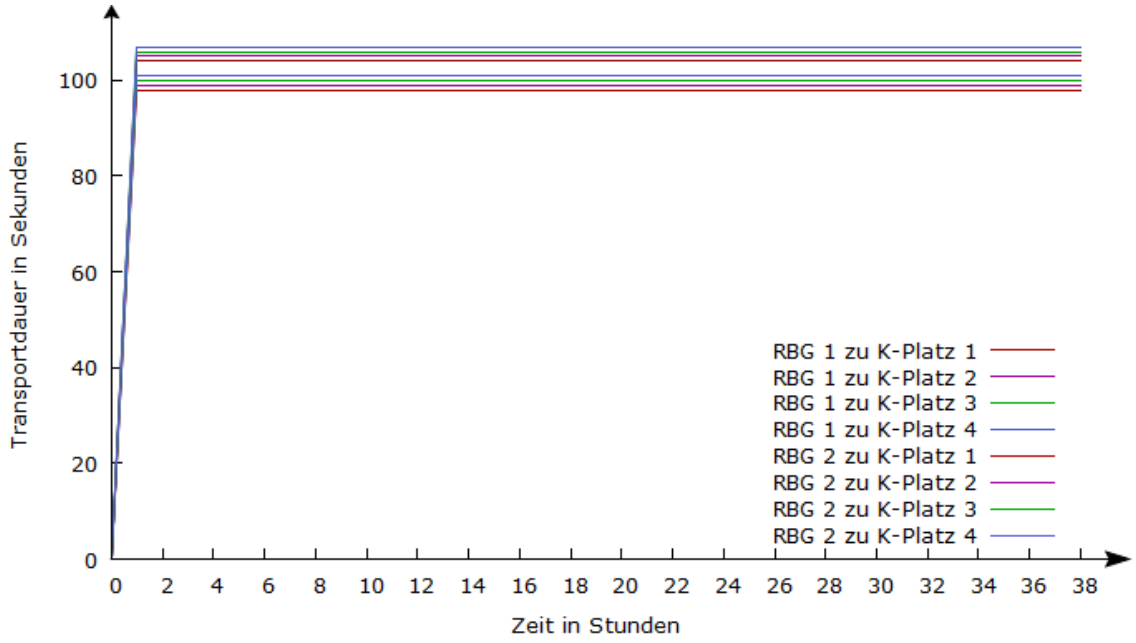

Abbildung 7-5: Verlauf der durchschnittlichen Transportzeiten an den K-Plätzen ohne Agentensystem

Die Transportzeiten in Abbildung 7-5 bleiben konstant, da noch kein anderer Fluss die im MFS befindlichen FEs behindern können und die FEs ungehindert von den RBGs zu den K-Plätzen fließen können. Abbildung 7-6 zeigt zum Vergleich die durchschnittlichen Transportzeiten mit Agentensystem. Die erste rot gestrichelte Linie markiert den Beginn und die zweite rot gestrichelte Linie das Ende der Reduzierung der Exploration-Wahrscheinlichkeit, die blau gestrichelte waagrechte Linie markiert die durchschnittliche Transportzeit von 102,5 Sekunden aus der Simulation ohne Agentensystem.

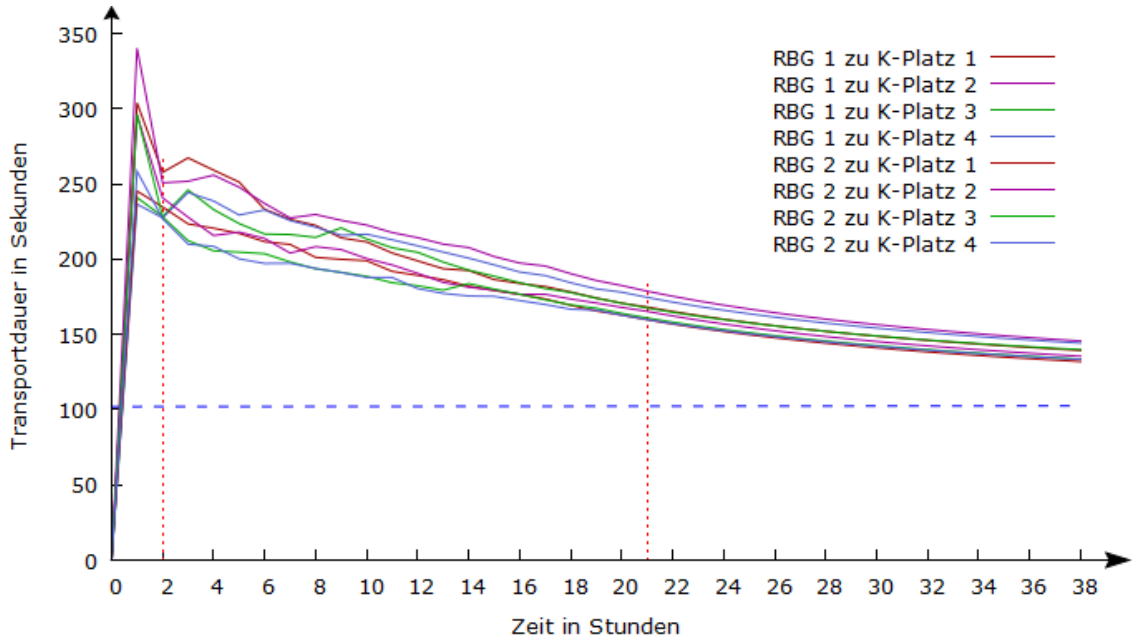

Abbildung 7-6: Verlauf der durchschnittlichen Transportzeiten an den K-Plätzen mit Agentensystem ohne füllstandsabhängige Transportzeit

Die Konvergenz der Transportzeiten in der Simulation mit Agentensystem aus Abbildung 7-6 gegen die Transportzeiten der Simulation ohne Agentensystem aus Abbildung 7-5 ist ersichtlich. Zum Vergleich zeigt Abbildung 7-7 das Ergebnis der Simulation mit Agentensystem und füllstandsabhängiger Transportzeit.

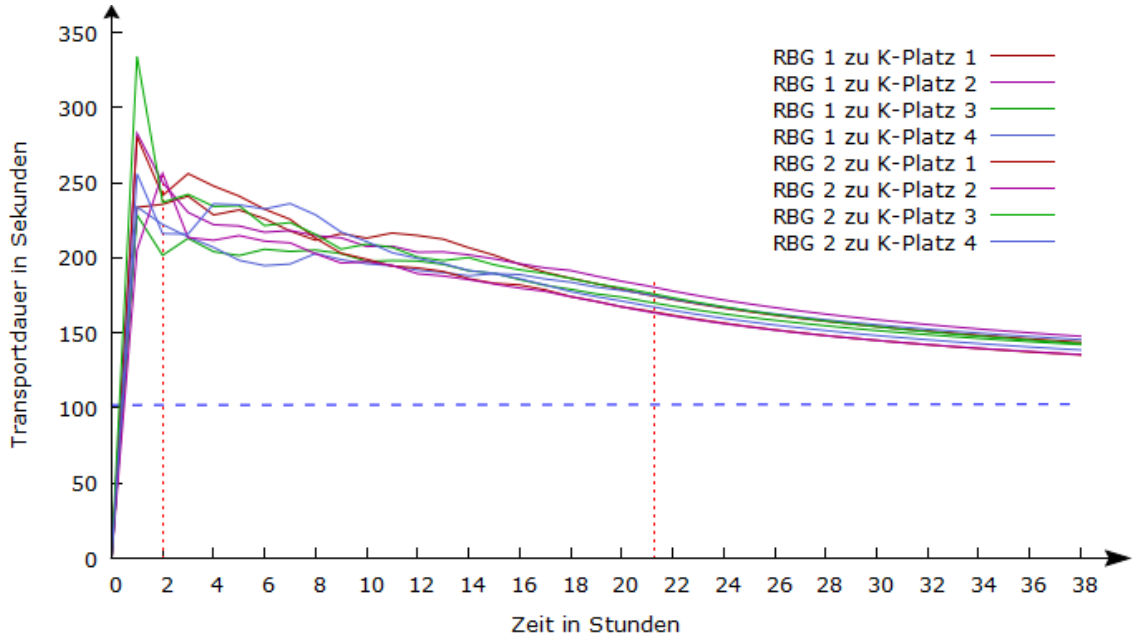

Abbildung 7-7: Verlauf der durchschnittlichen Transportzeiten an den K-Plätzen mit Agentensystem mit füllstandsabhängiger Transportzeit

Der Vergleich der Simulationen mit füllstandsabhängiger und füllstandsunabhängiger Transportzeit ist hier noch nicht relevant, da der Auslastungsgrad der Loop 1und Loop 3 durch den Einsatz eines Materialflusses noch zu gering ist.

Die durchschnittlichen Transportzeiten beider Simulationen konvergieren gegen einen Wert von ca. 150 s. Es ist jedoch auch zu erkennen, dass mit der Laufzeit der Simulation die durchschnittlichen Transportzeiten weiter abnehmen.

Für eine genauere Beurteilung der Leistungsfähigkeit des Lösungsansatzes müssen jedoch Langzeitsimulationen durchgeführt werden.

7.3.4 Analyse des Auslastungsgrads der Loop 1 und Loop 3

Von Interesse ist für eine erste Beurteilung des Agentensystems die Auslastung der beteiligten Loop 1 und Loop 3. Auch hier wurden der Beginn und das Ende der Reduzierung der Exploration-Wahrscheinlichkeit durch rot gestrichelte Linien markiert. Die schwarz gestrichelte Linie markiert den Auslastungsgrad für den Sprung in der füllstandsabhängigen Transportzeit.

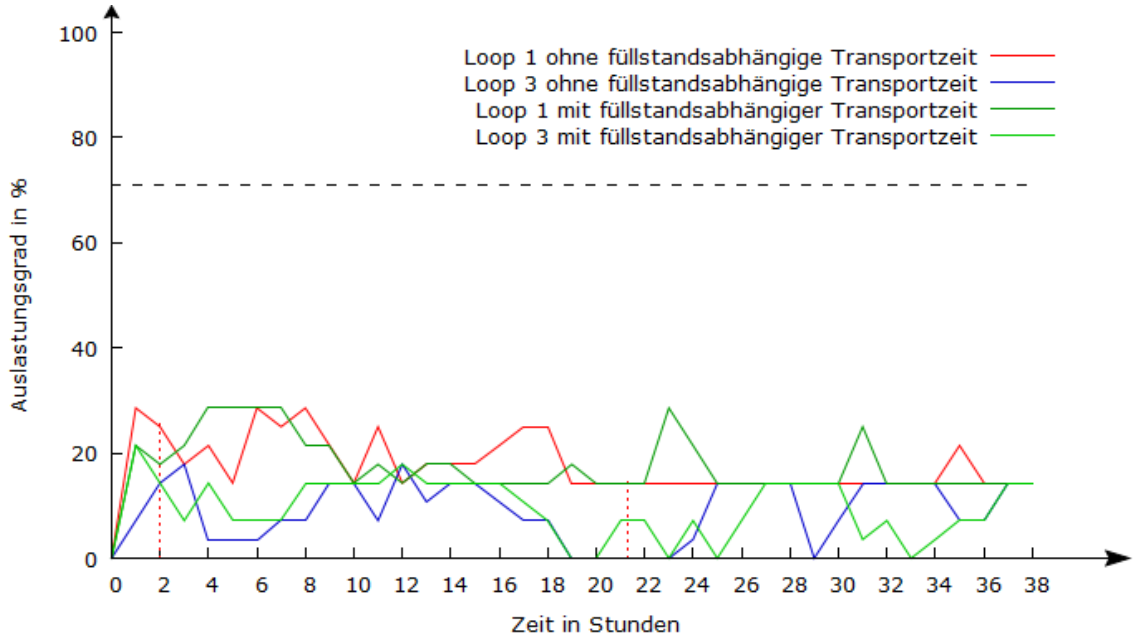

Abbildung 7-8: Verlauf des Auslastungsgrads der Loop 1 und Loop 3 mit und ohne füllstandsabhängiger Transportzeit

Aus Abbildung 7-8 geht hervor, dass sich die beiden Simulationen ähnlich verhalten. Das Agentensystem ist bemüht, die Auslastung der Loop 1 und Loop 3 konstant zu halten. Eine Entlastung von Loop 1 führt in beiden Fällen zu einer Belastung von Loop 3. Der Füllstand von beiden Simulationen bewegt sich nach der Absenkung der Exploration-Wahrscheinlichkeit auf 0 % im Bereich von 0 % bis 35 %.

Als Referenz zeigt Abbildung 7-9 die Füllstände der Loop 1 und Loop 3 über einen Zeitraum von ebenfalls 38 Stunden ohne Einsatz eines Agentensystems.

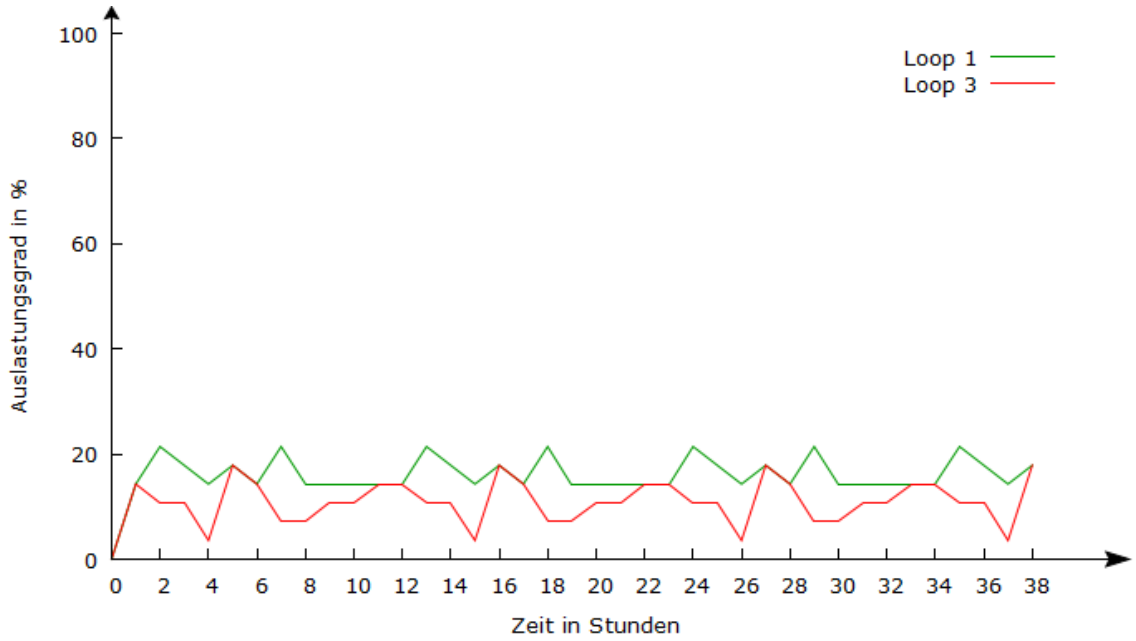

Abbildung 7-9: Verlauf des Auslastungsgrads der Loop 1 (grün) und Loop 3 (rot) ohne Agentensystem

Beim Vergleich der Simulationen mit Agentensystem und ohne Agentensystem wird deutlich, dass die Simulation mit Agentensystem eine ähnliche Auslastung der Loop 1 und Loop 3 wie die Simulation ohne Agentensystem erreicht.

Falls nicht anders angegeben, wird in den nachfolgenden Simulationen immer die füllstandsabhängige Transportzeit verwendet.

7.4 Analyse des Auslager- und Rücklagerflusses

Im nächsten Schritt wurde die Simulation um die Rückflüsse aus den K-Plätzen zu den RBGs und die entsprechenden Agenten erweitert. Zur Beurteilung der Qualität des Agentensystems im Umgang mit einem weiteren Materialfluss werden in diesem Abschnitt die gleichen Analysen wie in Abschnitt 7.3 durchgeführt. Zusätzlich wurden die in diesem Abschnitt beschriebenen Simulationen alle mit der in 6.1.2 beschriebenen füllstandsabhängigen Transportzeit t_{tr} durchgeführt. Für die in diesem Abschnitt dargestellten Testfälle gelten die folgenden Startparameter:

- $\alpha = 0{,}1$
- $\gamma = 0{,}6$
- $r_{in} = 5$
- Exploration-Wahrscheinlichkeit: 50 %

Darüber hinaus wurde für die Agenten *A „Einfahrt von RBG 1 und RBG 2 zu Loop 3“*, *E „Überfahrt von Loop 1 zu Loop 3“*, *F „Überfahrt von Loop 3 zu Loop 1“* und G *„Einfahrt von den K-Plätzen in Loop 3“* die erweiterte Belohnung

$$r = \frac{1}{Transportdauer} * \text{Durchsatz der letzten Stunde an der Zielsenke} + r_{in}$$

aus (6.4) verwendet.

7.4.1 Anpassung der Exploration-Wahrscheinlichkeit

Zur Ermittlung eines Startwertes für den Beginn der Absenkung der Exploration-Wahrscheinlichkeit wurden unterschiedliche Simulationen mit verschiedenen Startwerten für das Absenken der Exploration-Wahrscheinlichkeit durchgeführt. Im Folgenden wird das Ergebnis von drei exemplarischen Testläufen vorgestellt.

Um eine bessere Einschätzung für einen optimalen Startwert für das Absenken der Exploration-Wahrscheinlichkeit zu bekommen, wurde die zusätzliche Belohnung r_{in} für die Testläufe 1 und 2 auf 0 festgelegt. Für Testlauf 3 wurde die zusätzliche Belohnung r_{in} auf den Wert 5 festgelegt.

Hierzu wurde für Testlauf 1 als Startwert für das Heruntersetzen der Exploration-Wahrscheinlichkeit 1000 an den K-Plätzen ankommende FEs und für Testlauf 2 und Testlauf 3 als Startwert für das Reduzieren der Exploration-Wahrscheinlichkeit 500 an den K-Plätzen ankommende FEs gewählt. In allen Testfällen wurde die Exploration-Wahrscheinlichkeit in Schritten von 100 an den K-Plätzen ankommenden FE um 1 % gesenkt. Abbildung 7-10 zeigt den Verlauf des Gesamtdurchsatzes an allen K-Plätzen. Die blau gestrichelte Linie markiert den Zieldurchsatz von 240 FEs pro Stunde an den K-Plätzen (vgl. 6.1.3). Die rot gestrichelten Linien markieren den Beginn und das Ende der Änderung der Exploration-Wahrscheinlichkeit bei Testlauf 1. Die grün gestrichelten Linien markieren zum einen den Beginn der Änderung der Exploration-Wahrscheinlichkeit und zum anderen das letzte Herabsetzen der Exploration-Wahrscheinlichkeit auf 30 %, bevor der Durchsatz bei Testlauf 2 komplett einbricht. Auch bei Testlauf 3 bricht der Durchsatz nach dem letztmaligen Absenken der Exploration-Wahrscheinlichkeit auf 1 % komplett ein.

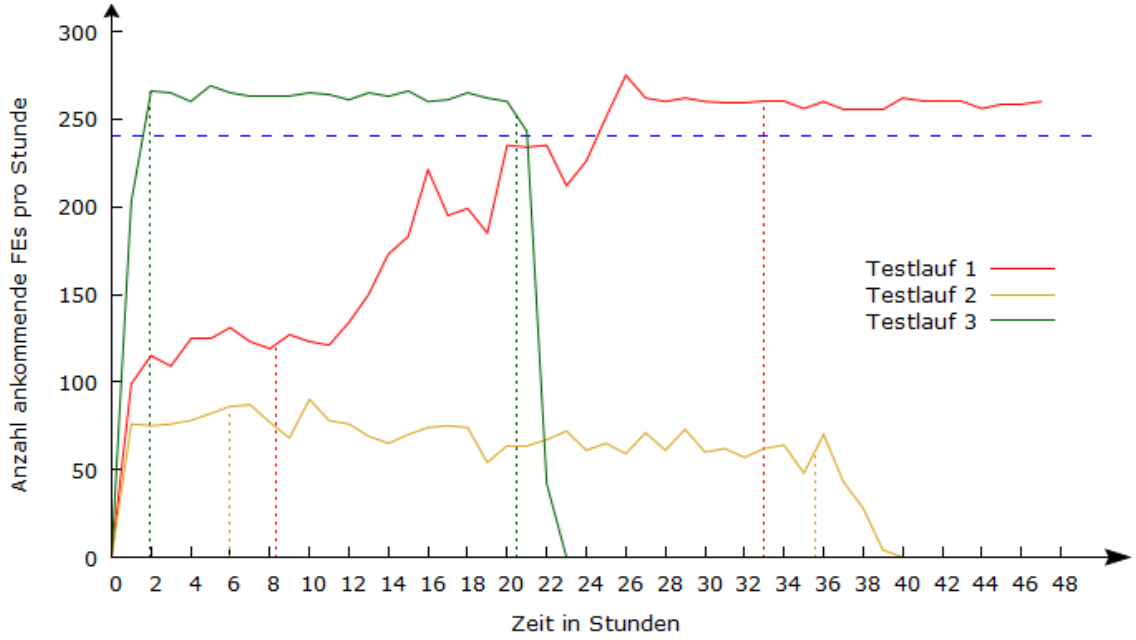

Abbildung 7-10: Verlauf des Gesamtdurchsatzes an den K-Plätzen mit den Startwerten 500 FEs und 1000 FEs für das Herabsetzen der Exploration-Wahrscheinlichkeit

Aus Abbildung 7-10 ist ersichtlich, dass 1000 an den K-Plätzen ankommende FEs als guter Startwert für das Herabsetzen der Exploration-Wahrscheinlichkeit angenommen werden können, da der Agent bei einer größeren Anzahl an ankommenden FEs eine bessere Möglichkeit hat, seine Umwelt und deren Bedingungen besser kennenzulernen.

Testlauf 2 zeigt keine Konvergenz gegen den Zieldurchsatz, hier wird sogar eine schlechte Strategie erreicht. Testlauf 1 hingegen zeigt sogar ohne Verwendung der zusätzlichen Belohnung aus 6.2.4 das Finden einer guten Strategie zur Maximierung des Durchsatzes.

Testlauf 3 ist schnell auf dem gewünschten Niveau über dem Zieldurchsatz. Jedoch werden auch hier nicht ausreichend zufällige Zustände für das optimale Training des Agenten erzeugt und der Durchsatz konvergiert nach der letzten Änderung der Exploration-Wahrscheinlichkeit gegen 0.

Auf Grundlage der Ergebnisse von Testlauf 1 werden 1000 an den K-Plätzen ankommende FEs als Startwert für das Herabsetzen der Exploration-Wahrscheinlichkeit mit einer Schrittweite von jeweils 100 an den K-Plätzen ankommenden FEs gewählt.

7.4.2 Durchsatzanalyse

Abbildung 7-11 zeigt den Verlauf der Durchsätze an den K-Plätzen über einen Zeitraum von 38 Stunden.

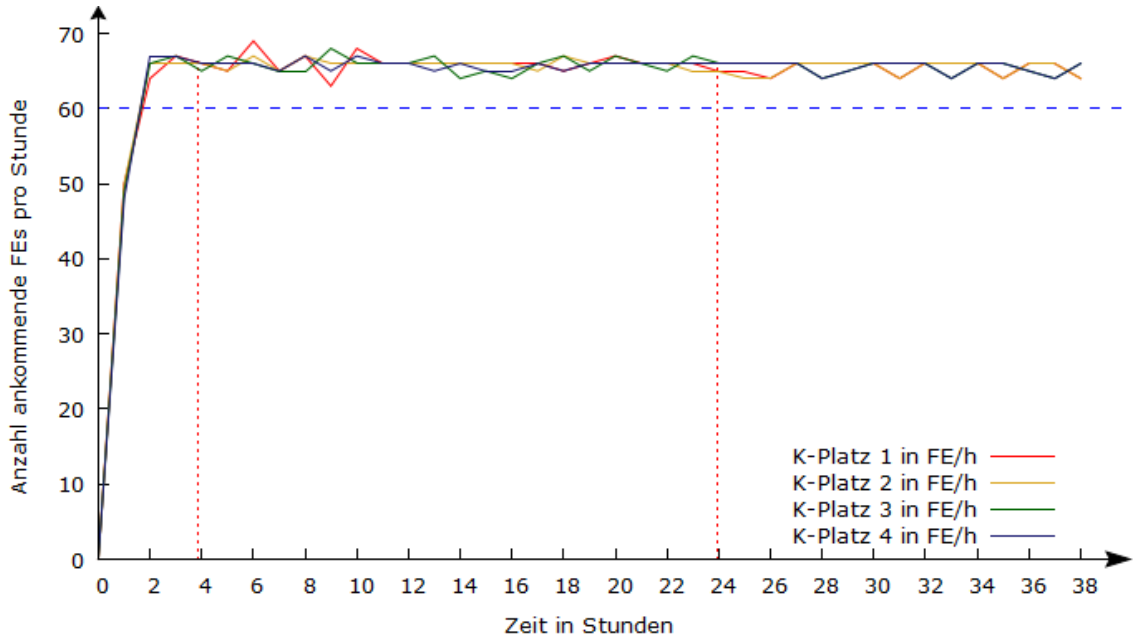

Abbildung 7-11: Verlauf des Durchsatzes an den K-Plätzen mit Rücklagerfluss

In der Exploration-Phase des in Abbildung 7-11 dargestellten Testlaufs ist eine höhere Oszillation der Kurven der Durchsätze zu beobachten. Nach Absenkung der Exploration-Wahrscheinlichkeit auf 0 % ist eine nahezu konstante Kurve der Durchsätze in Abhängigkeit der Einschleusungen der RBG 1 und RBG 2 zu beobachten (vgl. 6.1.3). Auch hier sind die Schwankungen in den Durchsätzen durch die Abbildung der Transportzeiten als Ereignisse innerhalb des Simulators zu erklären.

7.4.3 Analyse der Transportzeiten

Zusätzlich zu den Durchsätzen an den K-Plätzen wurde der Verlauf der durchschnittlichen Transportzeiten von den RBG 1 und RBG 2 zu den K-Plätzen näher betrachtet. Abbildung 7-12 zeigt den Verlauf der durchschnittlichen Transportzeiten von den RBG zu den K-Plätzen ohne Wareneingang über einen Zeitraum von 38 Stunden. Die rot gestrichelten Linien markieren den Beginn und das Ende der Reduzierung der Exploration-Wahrscheinlichkeit.

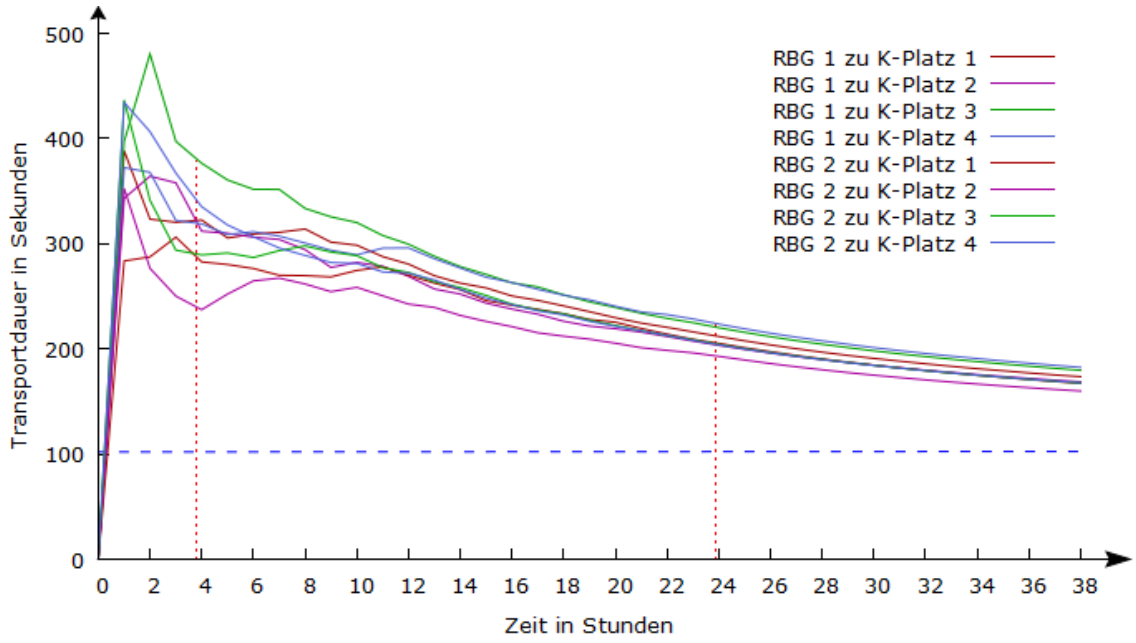

Abbildung 7-12: Verlauf der durchschnittlichen Transportzeiten von den RBGs zu den K-Plätzen

Zu Beginn der Simulation oszillieren die Transportzeiten noch stark, jedoch nehmen sie mit sinkender Exploration-Wahrscheinlichkeit weiter ab.

Der weitere Verlauf der durchschnittlichen Transportzeiten kann jedoch auch hier nur durch Langzeitexperimente gezeigt werden.

7.4.4 Analyse des Auslastungsgrads der Loop 1 und Loop 3

Um die Qualität der für die Loop 1 und Loop 3 zuständigen Agenten zu beurteilen, wird auch an dieser Stelle wieder der Auslastungsgrad der Loop 1 und Loop 3 betrachtet. Abbildung 7-13 zeigt den Verlauf des Füllstandes der Loop 1 und Loop 3.

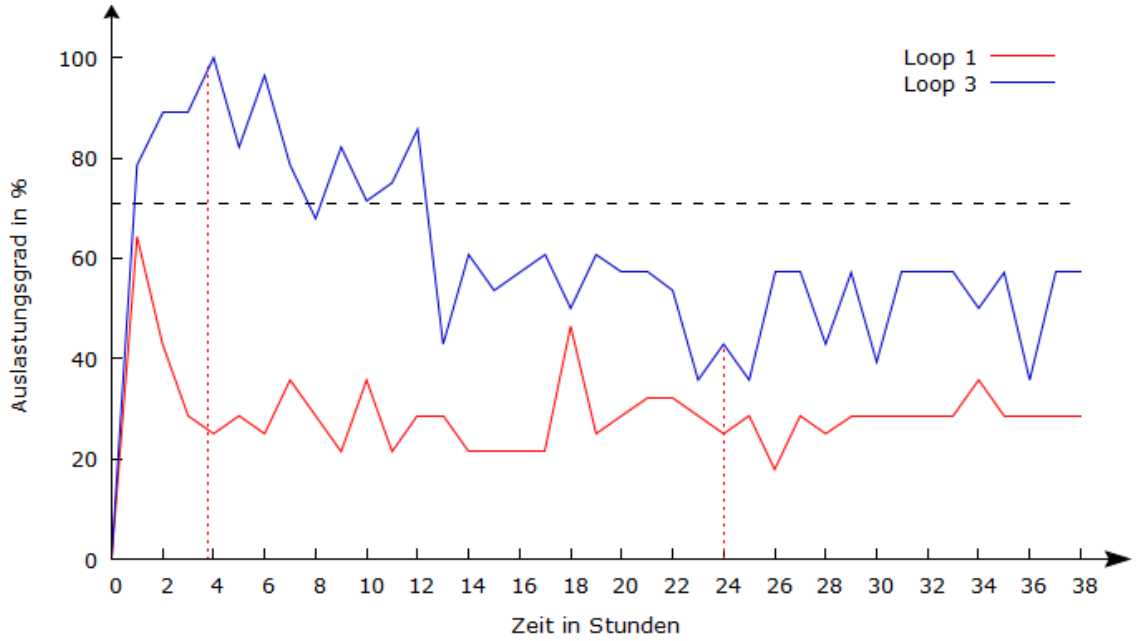

Abbildung 7-13: Verlauf der Füllstände der Loop 1 und Loop 3

Aus dem in Abbildung 7-13 dargestellten Verlauf der Füllstände der Loop 1 und Loop 3 ist zu erkennen, dass die für die Loop 1 und Loop 3 zuständigen Agenten in der Lage sind, die Loop 1 und Loop 3 kontinuierlich mit FEs zu versorgen. Der Auslastungsgrad von Loop 3 oszilliert zu

Beginn der Simulation, bedingt durch die hohe Exploration-Wahrscheinlichkeit, sogar im Bereich der füllstandsabhängigen Transportzeit.

Durch die spätere Oszillation des Auslastungsgrads unterhalb der Grenze der füllstandsabhängigen Transportzeit lässt sich die Annahme treffen, dass das Agentensystem versucht, die Zustände mit hohen Transportzeiten und einer dadurch sinkenden Belohnung zu vermeiden.

Des Weiteren ist erkennbar, dass die Füllstände der Loop 1 und Loop 3 in der Exploitation-Phase in einem konstanten Intervall oszillieren, zeitweise sind sie sogar konstant.

Durch das hier dargestellte Experiment konnte gezeigt werden, dass das Agentensystem grundsätzlich in der Lage ist, mehrere Materialflüsse innerhalb eines MFS zu optimieren.

7.5 Abschließende Bewertung des Lösungsansatzes

Um qualitative Aussagen über das Verhalten eines, durch den in dieser Arbeit vorgestellten Lösungsansatzes, zu optimierenden MFS treffen zu können, muss das Verhalten des MFS über einen längeren Zeitraum untersucht werden. Abbildung 7-14 zeigt exemplarisch den Verlauf des Gesamtdurchsatzes über einen Zeitraum von insgesamt 38 Simulationsstunden bzw. 6 Echtzeitstunden. Die gepunktete grüne Linie markiert hier das Erreichen der Exploitation-Phase.

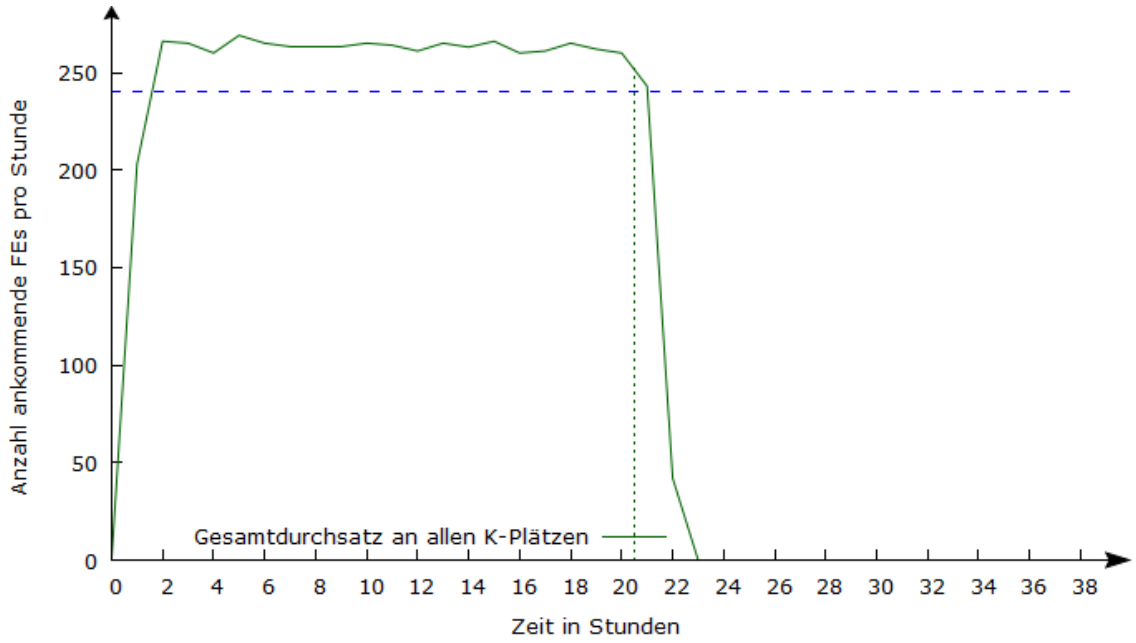

Abbildung 7-14: Verlauf des Gesamtdurchsatzes an allen K-Plätzen

Aus Abbildung 7-14 ist der zu Beginn der Simulation gute Durchsatzverlauf ersichtlich. Jedoch wurde die Exploration-Phase zu früh verlassen und das Agentensystem konnte die MFS-Zustände nicht mehr ausreichend bewerten. Deshalb ist es i. A. nicht ausreichend, das System direkt nach Erreichen der Exploitation-Phase und Erreichen des Zieldurchsatzes als optimiert zu betrachten. Im Verlauf der Exploitation-Phase der Simulation kann es zu noch nicht erreichten Zuständen des Agentensystems kommen, was zu einem Fehlverhalten des Agentensystems führen kann.

Wie in den hier gezeigten Ergebnisses der Experimente mit Agentensystem muss die Simulation über möglichst lange Zeiträume, z. B. 38 Simulationsstunden ausgeführt werden, um eine Optimierung des MFS beobachten zu können.

Durch den Einsatz eines Zeitraffers konnte die tatsächliche Simulationszeit jedoch auf ca. 6 Echtzeitstunden reduziert werden. Jedoch kann auch hier eine Analyse des simulierten MFS erst nach Abschluss der Simulation getätigt werden. Bei der Verwendung eines Zeitraffers in einer Simulation ist zu beachten, dass dieser nicht beliebig klein eingestellt werden kann, da das Training eines neuronalen Netzes nicht im Zeitraffer ausgeführt werden kann, da die Trainingsoperationen in Echtzeit ausgeführt werden.

8 ZUSAMMENFASSUNG UND AUSBLICK

Die Steuerung von Materialflusssystemen existiert momentan als statisches Element in einer zentralen Steuerungseinheit. Die Optimierung einer solchen Steuerung ist ein sehr zeitintensiver Prozess, der auf dem Wissen und den Erfahrungen des Materialflussingenieurs beruht. Zu Beginn der vorliegenden Arbeit wurde eine Auswahl an vorhandenen Methoden zur Analyse und Steuerung von Materialflusssystemen vorgestellt. Die vorgestellten Wege zur Analyse von intralogistischen Materialflüssen ermöglichen jedoch i. d. R. nur eine statische Sicht auf den Materialfluss.

Durch den Einsatz von Algorithmen aus dem Bereich der künstlichen Intelligenz kann der Optimierungsprozess jedoch automatisiert werden. Hierzu wurde in der vorliegenden Arbeit die Machbarkeit einer automatisierten Materialflusssteuerung durch den Einsatz eines Reinforcement-Learning-Ansatzes in der Q-Variante unter Verwendung von neuronalen Netzen zur Approximation der Q-Funktion untersucht. Es wurde ein Multiagentensystem entworfen, das grundsätzlich in der Lage ist, hochkomplexe Materialflusssysteme im Betrieb auf Grundlage der vorhandenen Informationen zu optimieren. Das Multiagentensystem wurde durch eine selbst erstellte Simulation einer vorhandenen Referenzanlage auf seine Qualität und Leistungsfähigkeit untersucht. Hierzu wurde die Simulation der Referenzanlage in drei Schritten um jeweils einen weiteren Bereich der Referenzanlage und die benötigten Agenten erweitert. In jedem Erweiterungsschritt wurden verschiedene Simulationen mit unterschiedlichen Start- und Zeitparametern des Reinforcement-Learning-Ansatzes ausgeführt und analysiert. Das Resultat der verschiedenen Analysen ist die Bestätigung einer Möglichkeit zur Optimierung von Materialflusssystemen durch den Einsatz von Reinforcement-Learning und Multiagentensystemen.

8.1 Ausblick

In den folgenden Abschnitten soll ein Ausblick zur Integration des Lösungsansatzes in die Software viadat gegeben werden. Im Vordergrund stehen hierbei die Topologie des Agentensystems sowie die Kommunikation zwischen dem Agentensystem und der Software viadat.

8.1.1 Weiterentwicklung des Simulators

In den in Kapitel 7 dargestellten Experimenten konnte die grundsätzliche Eignung des Lösungsansatzes zur Optimierung mehrerer konkurrierender Materialflüsse gezeigt werden.

Zur besseren Bewertung des Lösungsansatzes sollten noch zusätzliche MFE in die Simulation integriert werden. Hierbei sollte der Fokus auf MFE mit unterschiedlichen Funktionalitäten liegen, da so das Verhalten von Agenten für unterschiedliche MFE untersucht werden kann. So ist es bspw. nicht mehr ratsam, den Simulator nochmals um ein RBG-MFE zu erweitern. An dieser Stelle wäre die Erweiterung des Simulators um bspw. einen Wareneingangsplatz und mehrere Versandplätze sinnvoller.

Durch die Hinzunahme zusätzlicher MFEs steigt jedoch die Komplexität innerhalb der Simulation. Dies erhöht die Anforderungen an den Ersteller der Simulation, da in jedem Schritt der Erweiterung des hier vorgestellten Simulators die einzelnen Abhängigkeiten eines echten MFS auf den Simulator übertragen werden müssen.

Des Weiteren sollte in einem nächsten Schritt eine automatisierte Deadlock-Erkennung innerhalb des Simulators zur Bestrafung des Agentensystems implementiert werden. So muss im Fehlerfall die Simulation nicht neu gestartet werden und das Agentensystem kann besser aus den Daten des Simulators lernen.

Zur vollständigen Validierung des vorgestellten Lösungsansatzes sollte dieser in einer weiteren Arbeit auf eine bestehende Anlage transferiert werden. Hierzu muss allerdings ein allgemeingültigerer Ansatz zur Anpassung der Exploration-Wahrscheinlichkeit als der im Verlauf dieser Arbeit vorgeschlagene gefunden werden. Der in der vorliegenden Arbeit vorgeschlagene Ansatz zu Anpassung der Exploration-Wahrscheinlichkeit basiert in erster Linie auf einer Be-

obachtung des Durchsatzes an den K-Plätzen, was sehr stark an den momentanen Konfigurationsprozess eines MFS erinnert. Ein besserer Ansatz für die Wahl des Zeitpunkts der Anpassung der Exploration-Wahrscheinlichkeit könnte die Festlegung einer Zeitspanne sein, in der sich der Durchsatz an den Zielsenken nicht mehr ändert, bzw. die Oszillation konstant bleibt. Somit könnte der Agent die Optimierungen selbständig ohne Eingriffe durch einen MF-Ingenieur durchführen.

8.1.2 Integration des Lösungsansatzes in die Software viadat

In diesem Abschnitt werden die nötigen Schritte zur Umsetzung des Lösungsansatzes in der Software viadat kurz skizziert. Hierzu wird in einem ersten Schritt die Idee der Kommunikation des Agentensystems mit der SPS diskutiert. Im Anschluss werden die Möglichkeiten zur Training des Agentensystems in einer virtuellen Umgebung erörtert.

Das Agentensystem

Das Agentensystem kann zum einen als zentrale Einheit, in der jeder Agent als eigenständiger Prozess behandelt wird und seine eigene ausgehende Verbindung zur SPS besitzt erstellt werden. Zum anderen kann jeder Agent als eine eigene Einheit mit eigenen Verbindungen zur SPS implementiert werden. Abbildung 8-1 zeigt exemplarisch die beiden Möglichkeiten zur Implementierung des Agentensystems.

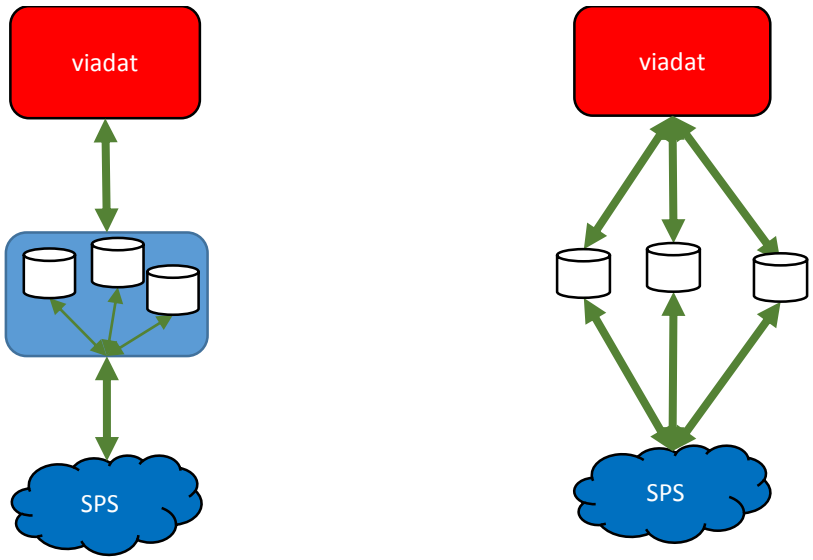

Abbildung 8-1: Zentrales Agentensystem (links) und dezentrales Agentensystem (rechts)

Die Zylinder symbolisieren jeweils einen Agenten, die grünen Pfeile stehen für Verbindungen, das rote Viereck stellt die Software viadat und die blaue Wolke eine SPS dar.

Zentrales Agentensystem

In einem zentralen Agentensystem (links) muss nur eine bidirektionale Verbindung von SPS zu Agentensystem aufgebaut werden. Zusätzlich wird nur eine Verbindung von Agentensystem zur Software viadat benötigt. Hier können die Agenten in eigenständigen Prozessen trainiert werden und jeder Agent kann über einen eigenen ausgehenden Kanal mit der SPS kommunizieren.

Dezentrales Agentensystem

Bei einem dezentralen Agentensystem (rechts) hat jeder Agent eine eigene bidirektionale Verbindung zur SPS. Hier kann die SPS entweder eine Art Broadcast [12] an alle Agenten schicken

[12] Broadcast (Engl.) : Nachricht an alle Teilnehmer eines Netzwerks. [Fis11]

um einen einzelnen Agenten anzusprechen oder es muss die Logik zum Ansprechen eines gezielten Agenten in die SPS integriert werden. Ferner verfügt jeder Agent über eine eigene Verbindung zur Software viadat. Auch hier kann jeder Agent eigenständig trainiert und befragt werden. Darüber hinaus können die Agenten sogar verteilt auf mehreren PCs eingesetzt werden, was die Antwortzeit eines Agenten erheblich reduzieren kann.

Um den Datenverkehr minimal zu halten und aus Übersichtsgründen, wird in der vorliegenden Arbeit das zentrale Agentensystem als Implementierungsansatz vorgeschlagen. Daher beziehen sich in den kommenden Abschnitten alle weiteren Vorschläge auf das hier vorgestellte zentrale Agentensystem.

Kommunikation mit der SPS

Zur Kommunikation des Agentensystems mit der SPS gibt es zwei Lösungsansätze. Zum einen kann das Agentensystem direkt mit der SPS kommunizieren. Zum anderen kann das Agentensystem aber auch in die Umgebung der SPS integriert werden.

Direkte Kommunikation mit der SPS

Damit die Entscheidungen des Agentensystems so früh wie möglich einen Einfluss auf den Materialfluss haben und um die Kosten in der Anlagenplanung auf einem akzeptablen Niveau zu halten, müssen die Agenten so früh wie möglich Zugriff auf einzelne Komponenten der Fördertechnik besitzen.

So sollten die für das Verlassen eines MFE zuständigen Agenten schon vor dem Verlassen des aktuellen MFE der SPS die mögliche Aktion mitteilen. Abbildung 8-2 veranschaulicht am Beispiel eines Loops mit verbundenem RBG die Position der Signale an die SPS.

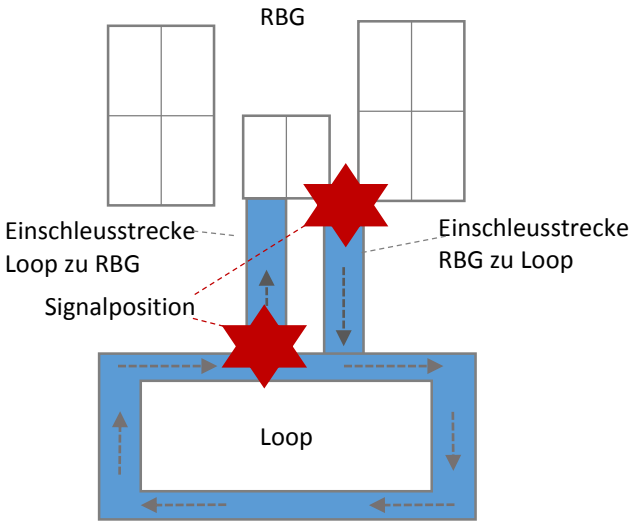

Abbildung 8-2: Beispiel für die Position der Signale an die SPS

Der rote Stern zeigt zum einen die Position des Einschleusstoppsignals am RBG-MFE vor Betreten der Verbindung zum Loop und zum anderen die Position des Einschleusstoppsignals vor dem Verlassen des Loops an. Die grau gestrichelten Linien veranschaulichen die Richtung des Materialflusses. Für das RBG bedeutet dies, dass die Staustrecke von Abgabeplatz zu Loop bereits zur maximalen Kapazität des Loops gezählt werden muss. Analog gilt für das Verlassen eines Loops, dass die Kapazität Einschleusstrecke von Loop zu RBG bereits zur Kapazität des RBG gehört.

Der hier vorgestellte Ansatz zur Positionierung der Signale an die SPS kann ohne Weiteres auf alle MFEs übertragen werden.

Wird das Einschleusstoppsignal erst an der Kreuzung von Staustrecke zu Loop positioniert, müssten an den entsprechenden Kreuzungen zusätzliche Scanner zur Erfassung der FE an den Fördertechnikplätzen installiert werden.

Direkte Einbettung des Agentensystems in die SPS

Um im Hinblick auf Hochleistungsmateriaflusssysteme mit Fördergeschwindigkeiten von mehr als 2 m/s eine effiziente Kommunikation der SPS mit dem Agentensystem zu gewährleisten könnte das Agentensystem direkt in die SPS-Umgebung der Fördertechnik integriert werden. Hierzu bietet der Hersteller der SPS-Steuerung Siemens einen Software PLC, den sog. Simatic WinAC RTX, mit dem die Steuerung der SPS auf einem normalen Industrie-PC mit höherer Leistung und Speicherkapazität in Echtzeit ausgeführt werden kann. Der WinAC RTX kann auf allen gängigen Windows-PC-Systemen verwendet werden. Zusätzlich unterstützt er die Implementierung selbst erstellter Sofwaremodule in den momentan gängigen Programmiersprachen, wie z.B. C++ oder C#. [Sie161]

Das genaue Vorgehen zur Implementierung des Agentensystems in einem Software-PLC ist nicht Gegenstand der vorliegenden Arbeit und kann in einer Folgearbeit näher untersucht werden.

Vorabtraining des Agentensystems

Um dem Agentensystem vorab schon Grundkenntnisse über die physikalische Topologie eines Zielmaterialflusssystems zu vermitteln, kann das Agentensystem schon vor der Inbetriebnahme trainiert werden. Da vorab die genauen Anforderungen, wie z. b. die genaue Auftragsstruktur des jeweiligen Kunden an das MFS noch nicht bekannt sind, muss das Agentensystem im Anschluss lediglich an die speziellen Kundenanforderungen wie z. B. Auftragsreihenfolgen im laufenden Betrieb des MFS angepasst werden.

Zum Vorabtraining des Agentensystems werden an dieser Stelle die virtuelle Inbetriebnahme inklusive einer Einbettung des Agentensystems und die Verwendung eines selbst entwickelten Simulators diskutiert.

Virtuelle Inbetriebnahme

Reale Materialflusssysteme können mit einem Zeitaufwand von ca. zwei Tagen bis hin zu drei Wochen in der Simulations- und Emulationssoftware Demo 3D vollständig, inklusive aller physischen Abhängigkeiten, modelliert werden.

Zusätzlich kann das Agentensystem per TCP/IP Verbindung an die Software Demo 3D angebunden werden. So könnte das Agentensystem schon vorab, virtuell, die physikalischen Abhängigkeiten des realen Materialflusssystems kennenlernen.

Durch eine Simulation eines MFS in der Software Demo 3D kann momentan ausschließlich das Echtzeitverhalten simuliert werden, da der Einsatz eines Zeitraffers die interne Physik-Engine der Software Demo 3D überfordert. In der Anfangsphase einer Simulation muss zusätzlich immer ein Materialflussingenieur anwesend sein, um im Fehlerfall die Simulation anpassen und neu starten zu können, da wie bei einem realen System Fehler innerhalb der einzelnen Fördertechnikelemente auftreten können.

Somit muss vor realer Inbetriebnahme einer Anlage ein relativ großer Aufwand zur Vorkonfiguration des Agentensystems und virtuellen Modellierung eines Materialflusssystems betrieben werden.

Jedoch kann durch eine virtuelle Inbetriebnahme und einer Vorkonfiguration des Agentensystems der Aufwand einer realen Inbetriebnahme eines Materialflusssystems minimiert werden.

Selbst entwickelter Simulator

Jedem realen Materialflusssystem geht eine Planung des MFS voraus. Aus dieser Planung können die Transportzeiten, Leistungen der Quellen und Senken sowie die Beschränkungen der MFE abgeleitet werden.

Somit kann auf Grundlage des in dieser Arbeit entwickelten Simulators ein rudimentärer Simulator erstellt werden, der als Eingabewerte die abgeleiteten Werte des Zielmaterialflusssystems erhält. Hierzu muss die Simulation vorab so erstellt werden, dass die Agenten und neuronalen Netze aus den Eingabewerten abgeleitet werden können.

Durch den Einsatz eines einfachen Simulators kann das Zielagentensystem im Zeitraffer trainiert werden, da physische Gegebenheiten ignoriert werden können.

Hierzu können die bereits vorhandenen Simulatoren in der Software viadat an das Agentensystem angepasst werden. So kann im Gegensatz zu einer Modellierung in der Software Demo 3D das Zielmaterialflusssystem mit relativ geringem Aufwand abgebildet werden.

8.1.3 Weitere Anwendungsgebiete des Lösungsansatzes und neuronaler Netze

Der in der vorliegenden Arbeit vorgeschlagene Lösungsansatz kann durch eine geeignete Modellierung auch ohne Weiteres auf die Leistungsoptimierung von Regalbediengeräten übertragen werden. Hier könnten bspw. die Ein- und Auslagerungen eines RBG durch einen geeigneten Ansatz optimiert werden. Somit könnte die momentan starre Konfiguration der RBG dynamischer gestaltet werden. Die genaue Anwendung von Reinforcement-Learning-Systemen zur Optimierung der Leistung von Regalbediengeräten kann in Folgearbeiten weiter untersucht werden und ist nicht Gegenstand der vorliegenden Arbeit.

Zukünftige Arbeiten können die Anwendung von neuronalen Netzen in puncto Predictive Maintenance näher betrachten. Hierzu haben bspw. [Jav01] gezeigt, dass durch den Einsatz von geeigneten neuronalen Netzen die Symptome, die zum Ausfall einer Komponente führen, erkannt werden können.

[Sze07] gehen hier sogar noch einen Schritt weiter und entwerfen in ihrer Arbeit ein System zur Entscheidungsunterstützung im Hinblick auf das Erstellen einer optimalen Ersetzungsrichtlinie von Komponenten innerhalb eines Systems. Der Fokus der Arbeit von [Sze07] liegt auf der Optimierung der Kosten, die durch den Ausfall und das Ersetzen einer Komponente entstehen können.

9 ABKÜRZUNGSVERZEICHNIS

2D	zweidimensional
3D	dreidimensional
AKL	Automatisches Kleinteilelager
bspw.	beispielsweise
bzw.	beziehungsweise
CEP	Complex Event Processing
COM	Communication Module
DG	DropGroup
ERP	Enterprise Ressource Planning
ETH	Ethernet Interface
FE	Fördereinheit
FEs	Fördereinheiten
FIFO	First in First out
ggf.	gegebenenfalls
GmbH	Gesellschaft mit beschränkter Haftung
GNU	GNU's Not Unix
GPL	General Public License
GUI	Graphical User Interface
HMI	Human Machine Interface
i. A.	im Allgemeinen
ID	Identifikation
IP	Internet Protocol
I-Punkt	Identifikationspunkt
IRIS	Institut für Rechnergestützte Ingenieursysteme
Java NNS	Java Neural Network Simulator
KNN	Künstliches neuronales Netz
K-Platz	Kommissionierplatz
K-Plätze	Kommissionierplätze
LAM	Lastaufnahmemittel
LAN	Local Area Network
LE	Lagereinheit
LVS	Lagerverwaltungssystem
MF	Materialfluss
MFE	Materialflusseinheit
MFEs	Materialflusseinheiten
MFS	Materialflusssystem
NC	Numerical Conrol
PG	PickGroup
PLC	Programmable Logic Controller
RBF	Radiale Basisfunktion
RBG	Regalbediengerät
RBGs	Regalbediengeräte
RProp	Resilient Backpropagation
SADT	Structured Analysis and Design Technique
SNNS	Stuttgart Neural Network Simulator
sog.	sogenannt
SPS	Speicherprogrammierbare Steuerung
SRM	Engl. Storage Retrieval Manager für Regalbediengerät
TCP	Transmission Control Protocol
VZ	Verzweigung
WAN	Wide Area Network
WE	Wareneingang
z. B.	zum Beispiel
ZE	Zeiteinheit
ZF	Zusammenführung

10 ABBILDUNGSVERZEICHNIS

11 TABELLENVERZEICHNIS

12 LITERATURVERZEICHNIS

[Sie16] Siemens AG. Totally Integrated Automation - Siemens. [Online], http://www.industry.siemens.com/topics/global/de/tia/Seiten/Default.aspx, Besucht am: 05.03.2016

[Alp08] Ethem Alpaydin. *Maschinelles Lernen*. München: Oldenburg Wissenschaftsverlag GmbH, 2008, S.:1-11, 270-273, 284-287.

[Alz15] Nada Alzahrani, Rasha Alsulim , Nourah Alaseem. "Data Analysis for Courses Registration" aus *Machine Learning and Data Mining in Pattern Recognition*, Hrsg. Petra Perner. Hamburg: Springer, 2015, S.:358-360.

[Arn09] Dieter Arnold , Kai Furmans. *Materialfluss in Logistiksystemen*, 6. Ausgabe. Karlsruhe, Deutschland: Springer, 2009, S.:11-55, 65-81,253-257.

[Bau14] Thomas Bauernhansel, Michael ten Hompel , Birgit Vogel-Heuser. *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Wiesbaden: Springer, 2014, S.:57-62, 238-242, 543-554.

[Ben12] Benjamin Behrens. "Entwicklung von Layout-Konzepten und Algorithmen zur komponentenneutralen Beschreibung von Materialflusssystemen," Institut für Rechnergestützte Systeme, Stuttgart, Diplomarbeit 2012, S.:17.

[Clo15] CloudComputingPatterns. [Online], http://www.cloudcomputingpatterns.org/Software_as_a_Service_(SaaS), Besucht am: 02.11.2015

[Dem15] demo3d.com. [Online], http://www.demo3d.com/show/Demo3D_OV, Besucht am: 23.10.2015

[Dic09] Philipp Dickmann. *Schlanker Materialfluss*, 2. Ausgabe. München, Deutschland: Springer, 2009, Kapitel 2, S.:140-148.

[Ert13] Wolfgang Ertel. *Grundkurs Künstliche Intelligenz*. Weingarten: Springer, 2013, S.:247-284, 287-308.

[Fis11] Peter Fischer , Peter Hofer. *Lexikon der Informatik*. Luzern, Schweiz: Springer, 2011, S.:129, 447.

[Geh07] Dirk Gehlich. "Analyse der Adaptierbarkeit moderner Internet-Routingverfahren an heutige Materialflusssysteme," Berufsakademie Stuttgart, Stuttgart, Dipomarbeit 2007, S.:10-16.

[GNU15] GNU Betriessystem. [Online], http://www.gnu.de/documents/lgpl-3.0.de.html, Besucht am: 01.12.2015

[GNU151] GNU Gernel Public License. [Online], http://www.gnu.de/documents/gpl.de.html, Besucht am: 01.12.2015

[Gnu16] Gnuplot homepage. [Online], http://www.gnuplot.info/, Besucht am: 26.02.2016

[Göh13] Peter Göhner. *Agentensysteme in der Automatisierungstechnik*. Stuttgart: Springer, 2013, S.:113-128.

[Gud121] Timm Gudehus. *Logistik 1*, 4. Ausgabe. Hamburg, Deutschland: Springer, 2012, S.:472-514.

[Gün10] Willibald Günter , Michael ten Hompel. *Internet der Dinge in der Intralogistik*. Heidelberg: Springer, 2010, S.:7, 53-167.

[Hed13] Ulrich Hedstück. *Simulation diskreter Prozesse*. Berlin, Deuschland: Springer Vieweg, 2013, S.:21-26, 176-180.

[Her05] Ekbert Hering, Klaus Bressler , Jürgen Gutekunst. *Elektronik für Ingenieure und Naturwissenschaftler*. Aalen, Deutschland: Springer, 2005, S.:552-567.

[Heu97] Jürgen Heuer. *Neuronale Netze in der Industrie*. Wiesbaden, Deutschland: Deutscher Universitäts Verlag, 1997, S.:207-237.

[Tho16] Thomas Jetter. MemBrain NN. [Online], http://membrain-nn.de/, Besucht am: 20.01.2016

[Kra09] Dr. rer. nat Oliver Kramer. *Computational Intellginence Eine Einführung*, Hrsg. Prof. Dr. O. Günther et al. Dortumund: Springer, 2009, S.:101-147.

[Kru11] Rudolf Kruse et al. *Computational Intelligence*. Magdeburg: Vieweg+Teubner, 2011, S.:7-151.

[Chr13] Christian Lauer. "Integriertes Modell zur Materialflusssimulation und zur Visualisierung in der virtuellen Realität," TU Kaiserslautern, Kaiserslautern, Dissertation 2013, S.:15-43.

[Liv08] David J. Livingstone. *Artifical Neural Networks - Methods and application*. Sandown, Isle of Wight: Humana Press, 2008, S.:3, 15-23.

[Lum15] Lumina. Lumina Analytica. [Online], http://www.lumina.com/why-analytica, Besucht am: 29.10.2015

[Mar03] Ulf Markwardt. "Modellierung modularer Materialfluss-Systeme mit Hilfe von künstlichen neuronalen Netzen," TU Dresden, Dresden, Dissertation 2003, S.:6-10, 14-21.

[Mat15] MathWorks. MATLAB. [Online], http://de.mathworks.com/products/matlab/, Besucht am: 29.10.2015

[Nyh12] Peter Nyhius , Hans-Peter Wiendahl. *Logistische Kennlinien*, 3. Ausgabe. Hannover, Deutschland: Springer, 2012, S.:17 ff.

[Jav01] Javier Ropero Pelhez et al. "PREDICTIVE MAINTENANCE ORIENTED NEURAL NETWORK SYSTEM - PREMON," in *IECON'O1: The 27th Annual Conference of the IEEE Industrial Electronics Society*, Sao Paulo, 2001.

[Rie93] Martin Riedmiller , Heinrich Braun. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," IEEE, IEEE International Conference on Neural Networks 1993, S.:586-591.

[Rob15] López González Roberto. Open NN. [Online], http://www.cimne.com/flood/default.asp, Besucht am: 01.12.2015

[Sie161] Siemens. PLC-Software von Siemens. [Online], http://w3.siemens.com/mcms/programmable-logic-controller/de/software-controller/software-plc-simatic-winac/simatic-winac-rtx-f/Seiten/Default.aspx, Besucht am: 30.03.2016

[via15] viastore SOFTWARE GmbH. viastoresoftware.de. [Online], http://www.viastoresoftware.de/viadat/

[Sut15] Richard S. Sutton , Andrew G. Barto. (2015, Februar) Reinforcement Learning: An Introduction. PDF. [Online]. https://webdocs.cs.ualberta.ca/~sutton/book/ebook/

[via151] viastore SYSTEMS GmbH. 3D-Modell einer Referenzanlage, 2015, Internes Dokument.

[via152] viastore SYSTEMS GmbH. Gesamtschaubild der Referenzanlage, 2015, Internes Dokument.

[via155] viastore SYSTEMS GmbH. Konfigurationsauszug aus der Referenzanlage, 2015, Internes Dokument.

[via154] viastore SYSTEMS GmbH. Leistungsbeschreibung der Referenzanlage, 2015, Internes Dokument.

[ten11] Michael ten Hompel , Volker Heidenblut. *Taschenlexikon Logistik*, 3. Ausgabe. Dortmund, Deutschland: Springer, 2011, S.:19, 179.

[Sze07] Sze-jung Wu et al. "A Neural Network Integrated Decision Support System for Condition-Based Optimal Predictive Maintenance Policy," in *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 37, NO. 2, MARCH*, 2007.

[Zel15] Prof. Dr. Andreas Zell. JavaNNS. [Online], http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/welcome_e.html, Besucht am: 01.12.2015

[Zel151] Prof. Dr. Andreas Zell , Hannes Planatscher. Stuttgart Neural Network Simulator. [Online], http://www.ra.cs.uni-tuebingen.de/SNNS/, Besucht am: 01.12.2015

13 INDEX

14 ANHANG

A-1 Verwendung der MemBrain-DLL

Die MemBrain-DLL kann u. a. als externe in C++ Bibliothek verwendet werden. Im folgenden Abschnitt wird eine kurze Einführung in die wichtigsten Befehle gegeben.

Laden eines neuronalen Netzes

In der vorliegenden Arbeit wurden statische Netze, die vorab unter Zuhilfenahme des grafischen Editors des MemBrain-Werkzeugs erzeugt wurden, verwendet. Diese können mit

```
_MB_LoadNet(netLocation_.c_str());
```

geladen werden. Wichtig ist hier, dass bei der Verwendung von mehr als einem Netz dynamisch Platz für neue Netze im Speicher mit

```
_MB_AddNet();
```

vor dem Hinzufügen von neuen Netzen allokiert werden muss. Nach dem Laden des Netzes müssen der Speicherplatz für die Trainingsdaten sowie die Eigenschaften der mit dem Netz verknüpften Trainingsdaten definiert werden. Im Anschluss müssen das Netz beim erstmaligen Laden zurückgesetzt und die Gewichte der Kanten randomisiert werden.

```
int lessonSize = _MB_GetInputCount();
if (_MB_SetLessonCount(trainingLessonPos_ + 1) == MEMBRAIN_ERR)
	return false;

if (_MB_SetLessonInputCount(lessonSize) == MEMBRAIN_ERR)
	return false;

if (_MB_NamesFromNet() == MEMBRAIN_ERR)
	return false;

if (_MB_SelectLesson(trainingLessonPos_) == MEMBRAIN_ERR)
	return false;

if (_MB_SetLessonInputCount(lessonSize) == MEMBRAIN_ERR)
	return false;

_MB_ResetNet();

_MB_RandomizeNet();
```

Laden der Lernregeln

Sind alle Netze und Trainingsdaten geladen, müssen die vorhandenen Lernregeln geladen werden. Wichtig ist hier, dass die Lernregeln nur einmal geladen werden müssen. Die Datei mit den

Lernregeln beinhaltet alle verfügbaren Lernregeln, die spezielle Lernregel kann über die genaue Spezifizierung des Namens geladen werden.

```
_MB_LoadTeacherFile(teacherFile_.c_str());
if (_MB_SelectTeacher(teacherName_.c_str()) == MEMBRAIN_ERR)
	return false;

return true;
```

Eingangswerte direkt an das neuronale Netz anlegen

Mit der MemBrain-DLL können dynamisch einzelne Eingangswerte an das neuronale Netz angelegt und der dazugehörige Ausgabewert erzeugt werden. Hierzu müssen das aktive Netz ausgewählt und die Eingangswerte an das Netz übergeben werden. Im Anschluss kann die Ausgabe des Netzes erzeugt werden.

```
_MB_SelectNet(netNumber_);
for (int i = 0; i < currentSet.size(); i++)
{
	if (_MB_ApplyInputAct(i, currentSet[i]) == MEMBRAIN_ERR)
	{
		success = false;
		break;
	}
}

double retVal = 0.0;
_MB_ThinkStep();

_MB_GetOutputAct(0, &retVal);
```

Das Anlegen von neuen Datensätzen an das neuronale Netz, ohne sie einer Lesson hinzuzufügen, ist bei der Beurteilung von zufällig ausgewählten Aktionen und ihren zugehörigen Zuständen von immenser Bedeutung, da so nicht immer neue Trainingsmuster teuer hinzugefügt und gelöscht werden müssen.

Ablegen von Trainingsdatensätzen

Um das neuronale Netz zu trainieren, müssen die Trainingsdatensätze in Lessons gespeichert werden. Zum Hinzufügen eines neuen Trainingsdatensatzes zu einer speziellen Lesson muss diese ausgewählt und die neuen Datensätze müssen zu den bereits existierenden hinzugefügt werden.

```
if (_MB_SelectLesson(trainingLessonPos_) == MEMBRAIN_ERR)
	return false;

_MB_AddPattern();
if (_MB_SelectPattern(actPos) == MEMBRAIN_ERR)
	return false;

for (int i = 0; i < currentSet.size(); i++)
{
	if (_MB_SetPatternInput(i, currentSet[i]) == MEMBRAIN_ERR)
	{
		success = false;
		break;
	}
}

if (_MB_SetPatternOutput(0, mylesson.getNormalizedResult()) == MEM-
BRAIN_ERR)
	success = false;
```

Die entsprechenden Werte des neuen Trainingsdatensatzes müssen einzeln in die ausgewählte Lesson eingefügt werden. Ebenso muss der Zielausgabewert an die gewünschte Position der Ausgabe geschrieben werden.

Trainieren eines neuronalen Netzes

Für das Training eines neuronalen Netzes müssen das gewünschte Netz ausgewählt und die zum Netz gehörenden Trainingsdatensätze geladen werden. Im Anschluss muss das Netz mit den Trainingsdatensätzen synchronisiert werden.

```
_MB_SelectNet(netNumber_);
_MB_EnableLessonOutData(1);
_MB_SetLessonOutputCount(1);

if (_MB_SelectLesson(trainingLessonPos_) == MEMBRAIN_ERR)
	return false;

lessonSize = _MB_GetLessonSize();
_MB_NamesFromNet();
int teachResult = _MB_TeachStep();
if (teachResult != MB_TR_OK)
	{
	_MB_StopTeaching();
	throw ViaError(VIAERROR_LOCATION, ViaErrId::HardError, agentId_ + "
COULD NOT TRAIN NET error code[" + to_string(teachResult) + "]");
	}
_MB_StopTeaching();
```

Nach Ende eines Trainingsschritts muss das Training durch `_MB_StopTeaching()` beendet werden. Ferner sollte das Training eines Netzes durch einen kontrollierten Programmabbruch beendet werden, um Fehler durch ein nicht trainiertes Netz zu vermeiden.

Änderung der Normalisierungsgrenzen von Eingangsneuronen

Um die Normalisierungsgrenzen eines Eingangsneurons zu ändern, muss dieses ausgewählt werden. Im Anschluss werden die Eigenschaften des Neurons abgefragt, die gewünschten Werte neu gesetzt und der MemBrain-DLL wieder mitgeteilt.

```
if (_MB_SelectInput(pos, FALSE) == MEMBRAIN_ERR)
	throw ViaError(VIAERROR_LOCATION, ViaErrId::HardError, "COULD NOT
SELECT NEURON FOR MAX AGE!!");

SMBNeuronProp* prop = new SMBNeuronProp();
_MB_GetSelectedNeuronProp(prop);
prop->normRangeHigh = maxTimeValue_;
if (_MB_SetSelectedNeuronProp(prop) == MEMBRAIN_ERR)
	throw ViaError(VIAERROR_LOCATION, ViaErrId::HardError, "COULD NOT
SET NEW MAX AGE!");

_MB_ClearSelection();
```

Um Fehler im Programmfluss zu vermeiden, empfiehlt es sich, das Programm durch einen kontrollierten Abbruch durch Auslösen eines definierten Fehlers (ViaError) zu beenden.

Änderung der Normalisierungsgrenzen des Ausgabeneurons

Die Auswahl eines Ausgabeneurons erfolgt analog zur Auswahl eines Eingabeneurons über die entsprechenden Anweisungen der MemBrain-DLL.

```
if (_MB_SelectOutput(0, FALSE) == MEMBRAIN_ERR)
	throw ViaError(VIAERROR_LOCATION, ViaErrId::HardError, "COULD NOT
SELECT OUTPUT NEURON!");

SMBNeuronProp* prop = new SMBNeuronProp();
_MB_GetSelectedNeuronProp(prop);
prop->normRangeHigh = maxOutValue_;
if (_MB_SetSelectedNeuronProp(prop) == MEMBRAIN_ERR)
	throw ViaError(VIAERROR_LOCATION, ViaErrId::HardError, "COULD NOT
SET NEW OUTPUT VALUE!");

_MB_ClearSelection();
```

Im obigen Beispiel existiert lediglich ein Ausgabeneuron, daher wird hier das Ausgabeneuron an der Position null ausgewählt. Auch hier ist der kontrollierte Programmabbruch durch das Auslösen eines definierten Fehlers empfehlenswert, da falsche Normalisierungsgrenzen des Ausgabeneurons zu einem kompletten Fehlverhalten des Agenten führen können.

Erklärung

Ich versichere, diese Arbeit selbständig verfasst zu haben.
Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.
Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens.
Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.
Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

(Benjamin Schehrer)

Declaration

I hereby declare that the work presented in this thesis is entirely my own.
I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations.
Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before.
The electronic copy is consistent with all submitted copies.

(Benjamin Schehrer)