technische universität dortmund

IT in Produktion und Logistik

FAKULTÄT MASCHINENBAU

Fachgebiet für IT in Produktion und Logistik
Univ.-Prof. Dr.-Ing. Markus Rabe

Bachelorarbeit

Multiagenten-Tabu-Suche
zur Variation modular modellierter Produktionssysteme

Jonas Müller
Matrikelnummer: 168532
Studiengang Wirtschaftsingenieurwesen

Eingereicht am 27.02.2017

Prüfer: Univ.-Prof. Dr.-Ing. Markus Rabe
Betreuer: Dipl.-Geoinf. Maik Deininger

Abstract

Die Produktionsplanung ist eine fortwährende Herausforderung. Insbesondere die Einplanung neuer Aufträge erfordert eine Veränderung des Produktionssystems. Um die Simulationslaufzeit im Rahmen einer simulationsgestützten Optimierung möglichst gering zu halten, bietet sich der Einsatz von Heuristiken an, die im Rahmen dieser Arbeit klassifiziert und charakterisiert werden. Mit Fokus auf modular modellierte Produktionssysteme, wird eine hybride Metaheuristik entwickelt, die sich beliebig zur Variation der Anzahl verschiedener Systemkomponenten einsetzen lässt. Eine exemplarische Umsetzung zeigt, dass die Suche in lokalen Stellen ein exploratives Verhalten annehmen kann und somit ein gutes Gleichgewicht zwischen Intensivierung und Diversifikation aufweist.

Inhaltsverzeichnis

1 Einleitung

Produzierende Unternehmen sind in der heutigen Zeit aus Wettbewerbsgründen dazu gezwungen, ihre Produktionssysteme an die dynamischen Entwicklungen in den Märkten und Technologien anzupassen. Einen typischen Faktor stellt dabei die Einplanung neuer Aufträge dar. Um eine möglichst zeiteffiziente Anpassung der Produktionsabläufe zu erzielen, müssen Produktionssysteme in der Lage sein, jeder Zeit neu konfiguriert zu werden. (Westkämper und Zahn 2009, S. 2-3) Es bietet sich an, alternative Konfigurationen eines Produktionssystems mithilfe einer Simulation zu bewerten. Bereits seit über 30 Jahren werden Simulationsverfahren zur Planung von Produktions- und Logistiksystemen unterstützend eingesetzt. (März et al. 2011, Vorwort) Um ein modelliertes Produktionssystem mittels einer simulationsgestützten Optimierung zu konfigurieren, muss die Simulationslaufzeit möglichst gering gehalten werden. Dies kann unter anderem mit der Verwendung von geeigneten Heuristiken erreicht werden. (März et al. 2011, S. 62)

Doch welche Heuristiken bieten sich an und welche sind dazu geeignet? Welche Bedeutung kommt in diesem Zusammenhang der Tabu-Suche als Metaheuristik zu? Und inwiefern können einzelne Verfahren miteinander kombiniert werden, sodass die Suche nach einer optimalen Lösung möglichst effizient verläuft?

Das Ziel dieser Arbeit ist daher die Entwicklung einer Multiagenten-Tabu-Suche, die als Heuristik die Simulationslaufzeit beschleunigen soll. Insbesondere soll verhindert werden, dass einzelne Lösungen mehrfach evaluiert werden. Im Fokus stehen zu variierende Produktionssysteme, die Maschinen oder Transportmittel unterschiedlichen Typs oder unterschiedlicher Menge verwenden. Das entwickelte Konzept soll somit in der Lage sein, unter entsprechender Parametrisierung auf vorgegebene Problemstellungen übertragen werden zu können.

Zu Beginn werden die Modellierung und Simulation von Produktionssystemen aufbereitet, die von der zu entwickelnden Multiagenten-Tabu-Suche als gegeben vorausgesetzt werden. Die Darstellung des Prozesses einer simulationsgestützten Optimierung wird den Übergang zur Verwendung von Optimierungsverfahren bilden. Da diese Verfahren eine Art Baukasten für die Entwicklung repräsentieren, wird eine Klassifizierung nach relevanten Kriterien vorgenommen. Mit der Erarbeitung verschiedener Ansätze zur Gestaltung von Heuristiken, wird sodann die Verwendung einer Multiagenten-Tabu-Suche begründet. Nach der Klassifizierung von Optimierungsverfahren und mit Blick auf diverse Entwicklungskonzepte, bei denen sowohl die Tabu-Suche als auch der Aspekt eines multiagentenbasierten Ansatzes ihre Bedeutung finden, kann die Entwicklung der Multiagenten-Tabu-Suche gestartet werden.

Im Zuge der Entwicklung werden einzelne Abschnitte und schließlich das Gesamtkonzept mithilfe der Modellierungssprache UML (Unified Modeling Language) formalisiert. Die

Entwicklung wird für die Variation einer Anzahl von Systemkomponenten wie Maschinen oder Transportmitteln vorgenommen.

Aufbauend auf die Entwicklungen wird die Multiagenten-Tabu-Suche für die Variation eines exemplarischen Produktionssystems angewendet. Dazu wird ein Systemüberblick verschafft, der veranschaulicht, welche Komponenten das System enthält und welche explizit variiert werden sollen. Die Umsetzung erfolgt in einer Entwicklungsumgebung unter Verwendung der Programmiersprache C++. Als Basis werden die in der Entwicklung formalisierten Modelle herangezogen. Die Ergebnisse einer Anzahl von Experimenten werden anschließend diskutiert. Zudem wird eine Bewertung des entwickelten Konzepts vorgenommen. Abschließend wird das Ergebnis dieser Arbeit zusammengefasst und mit einem Ausblick versehen.

2 Modellierung und Simulation modularer Produktionssysteme mittels Petri-Netzen

Im Rahmen dieser Arbeit wird eine Multiagenten-Tabu-Suche auf der Basis eines modular modellierten Produktionssystems entwickelt. Dieses Kapitel dient der Aufbereitung von Grundlagen, die in der späteren Betrachtung von Optimierungsansätzen und der anschließenden Entwicklung eines geeigneten Lösungsansatzes von Bedeutung sind.

2.1 Definition und Einordnung von Produktionssystemen

Produktionssysteme können abstrakt als ein *Input-Output-Prozess* betrachtet werden. Der Input umfasst einsatzbezogene Kriterien, die durch einen Transformationsprozess zum Output werden. Dieser wird durch ausbringungsbezogene Kriterien definiert. Des Weiteren können prozessbezogene Kriterien bestimmt werden, welche als Throughout bezeichnet werden. (Dyckhoff und Spengler 2007, S. 13) Produktionssysteme haben die Organisation der Produktion zur Aufgabe. Um ein komplexes System übersichtlicher darstellen zu können, erfolgt eine Zerlegung in Subsysteme. Auf verschiedenen Ebenen werden somit z.B. Werke, Fertigungsbereiche oder Arbeitsplätze abgebildet. (Schuh und Schmidt 2014, S. 3)

Eingehende Aufträge müssen mehrere Arbeitssysteme durchlaufen. Abbildung 2-1 zeigt beispielhaft einen vereinfachten Materialfluss. Ein eingehender Auftrag gelangt über vier Arbeitssysteme zum Kunden (K). Werden in kurzer Zeit viele Aufträge auf einmal ins System gebracht, können Arbeitssysteme an ihre Kapazitätsgrenzen gelangen. (Lödding 2010, S. 56)

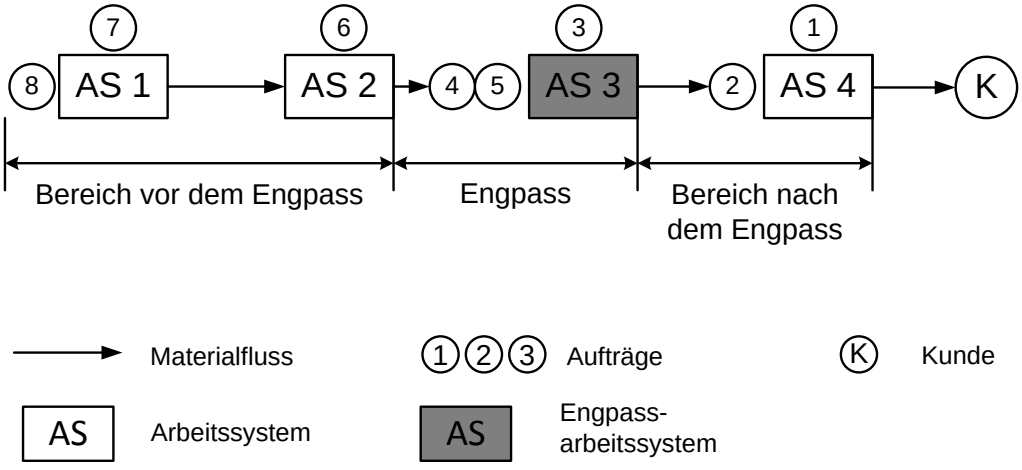

Abbildung 2-1: Engpassarbeitssystem (nach Lödding 2010, S. 56)

Die Folge ist eine Staubildung, die durch ein kritisches Arbeitssystem verursacht wird. Das als kritisch identifizierte Arbeitssystem wird auch als *Engpass* bezeichnet. Abbildung 2-1 zeigt, dass das dritte Arbeitssystem (AS 3) zum Engpass wird. Vor diesem bildet sich ein Bestandsaufbau, der durch die Warteschlange der Aufträge (4) und (5) kenntlich gemacht

ist. Um diesen möglichst gering zu halten, dürfen die Arbeitssysteme nicht mit voller Kapazitätsauslastung betrieben werden. Für eine reibungslose Produktion ist es daher angebracht, die Versorgung des Engpasses sicherzustellen. Eine sichere Versorgung ist dann gewährleistet, wenn im Bereich vor dem Engpass ausreichend Kapazitäten vorhanden sind. Für den identifizierten Engpass ist es sinnvoll, die maximale Kapazität zu wählen. Die gewählte Kapazität des vierten Arbeitssystems (AS 4) im Bereich nach dem Engpass ist ausschlaggebend für die Termintreue beim Kunden (K). (Lödding 2010, S. 56-57)

Ein Maß für die ideale Bearbeitungszeit eines Auftrages inklusive der Be- und Entladezeiten einzelner Arbeitsstationen ist die physikalische Durchlaufzeit (*Raw Prozess Time*). Werden alle entstehenden Wartezeiten mit einbezogen, so ergibt sich die Durchlaufzeit (*Cycle Time*) eines Auftrags. Verschiedenartige Aufträge besitzen eine unterschiedliche Raw Prozess Time. In einer Fertigung mit hoher Variantenzahl bietet sich daher eine Normierung an, der *Flussfaktor*. (Winz 2012, S. 28-29)

$$\text{Flussfaktor} = \frac{\text{Cycle Time}}{\text{Raw Prozess Time}} \tag{1}$$

Aufgrund zunehmender Anforderungen durch die Auftraggeber, müssen Unternehmen ihre Produktionssysteme ständig anpassen. Zumindest sollten sie auf Veränderungen reagieren. Erfolgen beim Auftraggeber Produktveränderungen, ist die Anpassung des Produktionssystems mit der Vielzahl an integrierten Prozessen zwingend erforderlich. Die erwähnten Transformationsprozesse einzelner Aufträge werden zunehmend individueller. Ein weiterer Faktor sind Nachfrageveränderungen, auf die reagiert werden sollte. Steigt beispielsweise die Nachfrage für ein bestimmtes Produkt, könnten zusätzliche Maschinen erforderlich sein. (Westkämper und Zahn 2009, S. 1-2)

Wiendahl et al. (2007, S. 786) gliedert die *factory changebilty* (Veränderbarkeit von Produktionsstätten) in fünf Klassen (siehe Abbildung 2-2). Die vorgenommene Klassifizierung werden auch von ElMaraghy (2009, S. 12-13) sowie Schuh und Schmidt (2014, S. 18-20) aufgegriffen und nehmen damit einen bedeutenden Stellenwert in der Literatur ein. In Abbildung 2-2 sind fünf Produktebenen über fünf Produktionsstufen aufgetragen. Beginnend von unten nach oben werden die Klassen *Umrüstbarkeit*, *Rekonfigurierbarkeit*, *Flexibilität*, *Wandlungsfähigkeit* und *Agilität* eingeordnet. Die Zuordnungen von Produktebenen und Produktionsstufen werden im Folgenden begründet dargelegt. *Umrüstbarkeit* gibt an, dass eine einzelne Maschine bzw. eine einzelne Arbeitsstation unmittelbar in der Lage ist, ein bereits bekanntes Teil mit einem Minimum an Verzögerungen und Aufwand zu bearbeiten. Können dagegen ähnliche Teile ohne die physische Veränderung des Produktionssystems (d.h. lediglich durch Umprogrammieren, Neu- oder Umplanung von Aufträgen) bearbeitet werden, liegt die *Rekonfigurierbarkeit* vor. Wird das Produktionssystem jedoch durch neue Produktionsprozesse, Materialflüssen oder Logistikfunktionen verändert, um

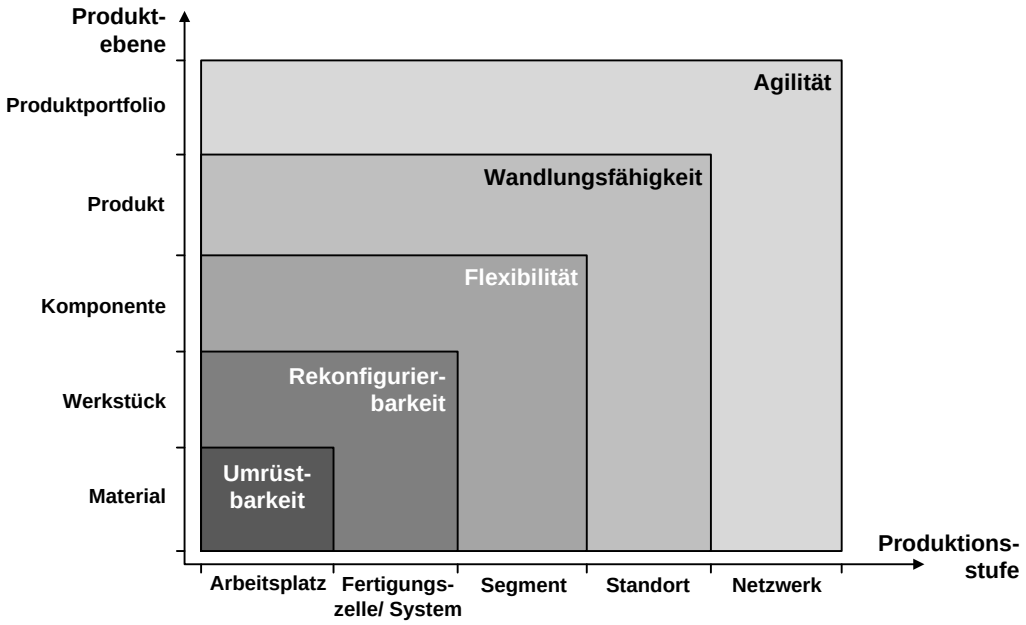

Abbildung 2-2: Klassen der Wandelbarkeit von produzierenden Unternehmen (nach Schuh und Schmidt 2014, S. 19)

damit neue ähnliche Teilefamilien zu bearbeiten, weist dies auf die Klasse der *Flexibilität* hin. Flexibilität kann außerdem durch das Hinzufügen oder Entfernen einer ganzen Komponente des Produktionssystems gekennzeichnet sein. *Wandlungsfähigkeit* liegt vor, wenn ein Produktionsstandort in der Lage ist, seine Fabrikstruktur an neue Produktgruppen anzupassen. Die strukturellen Veränderungen beziehen sich neben veränderten Produktions- und Logistikprozessen auch auf eine Neuausrichtung der Organisationsstruktur sowie der Berücksichtigung von personellen Veränderungen. *Agilität* bildet die fünfte Klasse und ist durch die Erweiterung des Produktportfolios, insbesondere durch die Eröffnung neuer Märkte und die Schaffung der dafür notwendigen Fertigungskapazitäten, charakterisiert. (Wiendahl et al. 2007, S. 786; ElMaraghy 2009, S. 12-13; Schuh und Schmidt 2014, S. 19-20)

2.2 Modellierung und Modularität ereignisdiskreter Systeme

Um eine Variation modular modellierter Produktionssysteme vornehmen zu können, sollte zunächst auf die modulare Modellierung eingegangen werden. An dieser Stelle soll darauf hingewiesen sein, dass es sich bei den modellierten Modulen nicht ausschließlich um flexible Fertigungszellen handelt. Der Begriff *modular* sagt lediglich aus, dass das Produktionssystem in Module zerlegt wurde. Es kann sich hier beispielsweise um Abschnitte einer Produktionslinie handeln. Dennoch ist es möglich, dass ein modelliertes Modul eine vollständige Produktionslinie abbildet. Nach Westkämper und Zahn (2009, S. 80) erleichtern Komponenten, die durch Modularität, Kompatibilität und eine Entspezialisierung gekennzeichnet sind, bei einer kurzfristigen Anpassungsnotwendigkeit des Systems, die Rekonfi-

guration. Dies bekräftigt die Relevanz von Modularität für ein zu optimierendes Produktionssystem.

Um komplexe Systeme zu modellieren, werden sie in Teilsysteme zerlegt. Wird innerhalb eines Teilsystems ein anderes Teilsystem aufgerufen, handelt es sich dabei um ein Ereignis. Das aufrufende Teilsystem begibt sich für die Dauer des Ereignisses in einen diskreten Zustand, der z.B. die Bearbeitung eines Auftrags sein könnte. Interaktionen im übergeordneten Gesamtsystem lassen sich durch die Anzahl an Ereignissen und die Dauer der Zustände beschreiben. In der Folge repräsentiert die Modellierung des Gesamtsystems ein ereignisdiskretes System. Die Summe der Zustände aller Teilsysteme (lokale Zustände) bildet den Zustand des Gesamtsystems (globaler Zustand). Um die Übergänge zwischen den verschiedenen Zuständen abzubilden, bieten sich verschiedene Möglichkeiten an. Die Darstellung ereignisdiskreter Systeme kann beispielsweise mithilfe von Zustandsübergangsgraphen vorgenommen werden. In diesem Graph werden alle Übergänge von einem Zustand zu einem Folgezustand grafisch modelliert. Diese Art der Modellierung ist für ereignisdiskrete Systeme deutlich komplexer als eine Modellierung mittels Petri-Netz-Graphen. Da Petri-Netze sich besonders zur Modellierung ereignisdiskreter Systeme eignen, werden im Folgenden kurz deren Grundlagen aufbereitet. (Puente León und Kiencke 2013)

2.3 Grundlagen Petri-Netze

Ereignisdiskrete Systeme beinhalten Abhängigkeiten der Teilsysteme untereinander sowie zu berücksichtigende Nebenläufigkeiten. Petri-Netze sind in der Lage dies umzusetzen und erzielen dabei eine geringere Komplexität als die erwähnten Zustandsübergangsgraphen. (Puente León und Kiencke 2013, S. 333) Aufgrund ihrer Vorteile bilden sie die Grundlage der Modellierung des in dieser Arbeit betrachteten Produktionssystems. Im Folgenden wird kurz der Hintergrund dieses Konzepts erklärt und die Verwendung für den vorliegenden Anwendungsfall begründet.

Das von Carl Adam Petri entwickelte Modellierungskonzept wurde mit seiner Dissertation über Kommunikation mit Automaten im Jahr 1962 erstmals bekannt. Ihren Namen haben die Petri-Netze durch ihren Begründer erhalten. Die Verwendung von Petri-Netzen hat sich jedoch in den letzten Jahrzenten verändert. Ursprünglich hatte Petri die Netze zur Darstellung von physikalischen Informationsflüssen angedacht. Neben einer anschaulichen Darstellung bieten Petri-Netze umfangreiche Funktionen, insbesondere auch für die Modellierung von Automatisierungssystemen. Petri-Netze unterstützen beispielsweise die Analyse betrieblicher Abläufe. Zudem werden sie verwendet, um Verkehrssysteme zu planen oder industrielle Fertigungs- und Steuerungsprozesse zu modellieren. (Puente León und Kiencke 2013, S. 333; Schöf 1997, S. 5)

Die wichtigsten Komponenten eines klassischen Petri-Netzes bilden die *Plätze* und die *Transitionen*, welche durch gerichtete *Kanten* miteinander verknüpft werden. Dabei ist zu beachten, dass immer nur ein Platz mit einer Transition verbunden werden darf. Die Kombinationen Platz-Platz und Transition-Transition sind nicht erlaubt. Plätze bilden *passive* Komponenten des Netzes und werden als Kreis oder Ellipse visualisiert. Ein Platz besitzt die Fähigkeit Dinge zu lagern, zu speichern oder sie sichtbar zu machen. Zudem kann er einen Zustand annehmen. Demgegenüber stehen die Transitionen als *aktive* Komponenten eines Petri-Netzes. Zur Darstellung wird ein Quadrat oder Rechteck verwendet. Als aktives Element ist eine Transition in der Lage Dinge zu erzeugen, zu verbrauchen, zu transportieren oder auch zu verändern. Die bereits erwähnten Kanten werden durch Pfeile visualisiert und stellen Systemkomponenten miteinander in Beziehung. (Reisig 2010, S. 22-23) Um Systemzustände im Petri-Netz darstellen zu können, werden noch die sogenannten *Marken* (*tokens*) benötigt, die als kleine schwarze Punkte visualisiert werden. Während die Plätze einzelne Zustandsparameter darstellen, dienen die Marken dazu, diese Zustandsparameter im Netz zu markieren. Dabei kann ein Platz auch von mehreren Marken gleichzeitig belegt werden. (Puente León und Kiencke 2013, S. 333-334) Die genannten Elemente sind in Abbildung 2-3 miteinander in Verbindung gebracht. Der linke Platz ist dabei von einer Marke belegt.

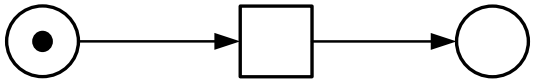

Abbildung 2-3: Komponenten in einem Petri-Netz (nach Reisig 2010, S. 14)

Die Verteilung der Marken im Petri-Netz ergibt eine *Markierung*, die wiederum einen globalen Zustand modelliert. Auf Modellebene wird eine Zustandsänderung des Systems folglich durch die Veränderung einer Markierung zu einer neuen Markierung erreicht. (Reisig 2010, S. 18) Petri-Netze sind somit in der Lage, das System mit einer überschaubaren Komplexität zu modellieren. (siehe Abschnitt 2.2)

Der Nutzen von Petri-Netzen und deren Eignung – sowohl für die Modellierung von Produktionssystemen als auch für die Simulation – wird bei zunehmender Größe problematisch, da die übersichtliche Struktur verloren geht. Daher bietet es sich an, ein System modular zu modellieren. Ein einfaches Petri-Netz bildet zunächst nur eine Ebene. Abhilfe schafft die Verwendung höherer bzw. hierarchischer Petri-Netz-Modelle. Diese bieten die Möglichkeit einer Abstraktion und schaffen damit ein besseres Verständnis. Bei der Verwendung hierarchischer Petri-Netze erfolgt die Modellierung in mehreren Ebenen, sodass bei Bedarf stärker ins Detail gegangen werden kann. (Righini 1993, S. 2463-2466; Schöf 1997, S. 19; Puente León und Kiencke 2013, S. 382)

In einem hierarchischen Petri-Netz-Modell wird die Detaillierung durch die Verfeinerung eines *Vaterknotens* (Platz oder Transition) zu einem neuen *Unternetz* (Teilnetz) erreicht. Die Abstraktion wird auch als Vergröberung bezeichnet und bildet den umgekehrten Fall ab (siehe Abbildung 2-4). Vaterknoten und Unternetze bilden somit die hierarchischen Strukturen in dieser Art von Petri-Netzen. (Puente León und Kiencke 2013, S. 334 und S . 382)

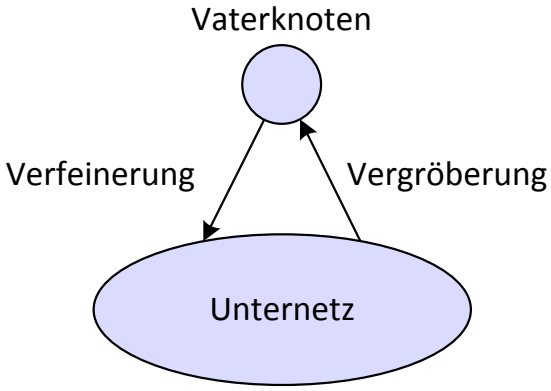

Abbildung 2-4: Hierarchische Struktur in Petri-Netzen (nach Puente León und Kiencke 2013, S. 382)

Speziell für die Modellierung eines Produktionssystems haben hierarchische Netze den Vorteil, dass einzelne Module separat modelliert und nach Belieben miteinander verknüpft werden können. Eine Klasse höherer Petri-Netze bilden die Zeitbehafteten Hierarchisch Objektorientieren Netze (THORNs). Zeitbehaftet und objektorientiert müssen die Netze sein, damit kausale Auftragsstrukturen abgebildet werden können. Kausale Strukturen sind z.B. Sequenzen, Verzweigungen oder Synchronisationen. Ein Auftrag muss als Objekt erfasst werden, um auf diese Weise einzelne Teilnetze zur Weiterverarbeitung durchlaufen zu können. Dabei fallen Bearbeitungs- und Wartezeiten an, die durch den zeitbehafteten Aspekt berücksichtigt werden. (Schöf. 1997, S. 43; Puente León und Kiencke 2013, S. 478-479).

2.4 Simulationsgestützte Optimierung

Es bieten sich verschiedene Vorgehensweisen an, Produktionssysteme zu analysieren und mithilfe der gewonnen Kenntnisse anschließend zu optimieren. Zum einen besteht die Möglichkeit, reale Experimente durchzuführen. Die empirisch ermittelten Werte würden eine Optimierung zulassen, jedoch kann es bei einer gewissen Komplexität des Systems – z.B. bei Industrieanlagen oder Verkehrssystemen – kostengünstiger und zeitsparender sein, wenn das zu untersuchende System zunächst mittels einer Modellierungssprache abgebildet wird. Auf dieses Modell können dann geeignete Verfahren angewendet werden. An dieser Stelle kommt eine Simulation zum Einsatz, die das Systemverhalten nachbildet. Die Erkenntnisse aus der Simulation können folglich auf das reale System übertragen werden

(Schöf 1997, S. 1-3). Dazu berücksichtigt die Simulation stochastische Verteilungen für verschiedene Prozesse. Denn fast alle Systeme weisen eine oder mehrere Quellen auf, die vom Zufall abhängig sind. In der Fertigung sind dies beispielsweise Bearbeitungszeiten, aber auch Maschinenreparatur- oder Ausfallzeiten. (Law 2007, S. 275-276) Da Produktionssysteme sich aus vielen Ereignissen zusammensetzen, bilden sie ein ereignisdiskretes System. Um dieses getreu abbilden zu können, eignet sich die *ereignisdiskrete Simulation*. Sie verwendet eine Ereignisliste die unter Verwendung von Zeitstempeln abgearbeitet werden kann. (Bracht et al. 2011, S. 122-123)

Jede Kombinationsmöglichkeit eines zu variierenden Systems mit der Simulation zu bewerten und letztlich einfach die beste Lösung auszuwählen, ist aufgrund einer sehr langen Simulationslaufzeit kritisch. Abhilfe können hier Algorithmen liefern, die auch als *Heuristiken* bekannt sind. Diese garantieren keine optimale Lösung, aber liefern – trotz sehr komplexer Problemstellungen – gute Lösungen in kurzer Zeit. Eine spezielle Form der Heuristik ist die *Metaheuristik*. Im Allgemeinen weisen Metaheuristiken Methoden auf, die Forscher aus der Natur abgeschaut haben. Sie bieten ein allgemeines Rahmenwerk und müssen an spezifische Probleme angepasst werden. (Oster 2007, S.81-82)

Abbildung 2-5 setzt den Fokus erneut auf den Prozess einer simulationsgestützten Optimierung.

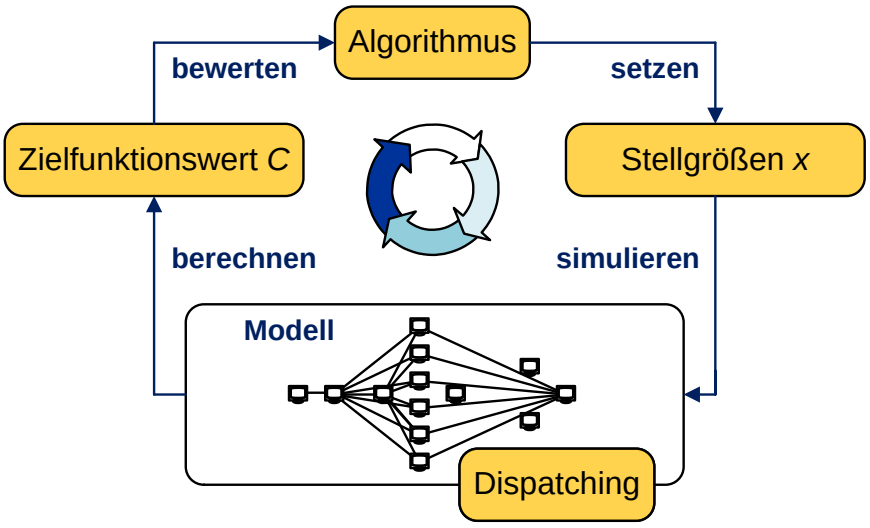

Abbildung 2-5: Prinzip der simulationsgestützten Optimierung (nach Klemmt et al. 2011, S. 54)

Stellgrößen werden durch einen entworfenen Suchalgorithmus bestimmt. Es entsteht ein Stellgrößenvektor mit allen notwendigen Stellgrößen, der das Modell entsprechend vorgegebener Parameter anpassen wird. Mit dem *Dispatching* ist genau diese Modellanpassung gemeint. Nach einem Simulationsdurchlauf kann ein Zielfunktionswert bestimmt und durch den Vergleich mit anderen Zielwerten bewertet werden. Daraufhin optimiert der Algorithmus die Stellgrößen. Durch mehrfache Anpassung der Stellgrößen, entsteht ein Kreislauf, der nach mehreren Iterationen beendet wird. (Klemmt et al. 2011, S. 54-55)

3 Klassifikation von Optimierungsverfahren

Simulationsgestützte Optimierung benötigt einen geeigneten Suchalgorithmus, bezüglich dessen bereits der Einsatz von Heuristiken aufgegriffen wurde. Innerhalb dieses Kapitels wird anhand gewählter Klassifizierungsmerkmalen aufgezeigt, welche Unterschiede zwischen einzelnen Suchverfahren bestehen. Zunächst werden die charakteristischen Eigenschaften allgemein erklärt. Darauf aufbauend können Optimierungsverfahren klassifiziert werden. Abschließend wird mit einer detaillierten Darstellung einzelner Optimierungsverfahren aufgezeigt, welche Möglichkeiten sich zur Gestaltung einer Heuristik anbieten.

3.1 Abgrenzung heuristischer gegenüber analytischer Verfahren

Grundlegend werden Lösungsverfahren in *explorative* und *analytische* Verfahren unterschieden. Heuristische Verfahren sind dabei der Klasse explorativer Verfahren zuzuordnen. Beide Verfahrenstypen können zur Lösung von Planungsproblemen verwendet werden und unterscheiden sich insbesondere in ihrer Vorgehensweise. (Fritzsche 2009, S. 39)

Analytische Verfahren durchsuchen nicht den Lösungsraum, sondern verwenden den Wertebereich einer gegebenen Zielfunktion. Das Auflösen linearer oder quadratischer Gleichungssysteme sowie die Berechnung von Optima mittels Ableitung differenzierbarer Funktionen zählen beispielsweise zu den exakten analytischen Verfahren. Wird die exakte Analytik durch eine nicht differenzierbare Funktion eingeschränkt, kann jedoch auf Approximation zurückgegriffen werden. Die Näherung an gute Lösungen kann somit erreicht werden. (Fritzsche 2009, S. 39-40)

Explorative Verfahren bilden den Gegensatz zu den analytischen Verfahren. Sie beschränken sich nicht auf den Wertebereich einer Zielfunktion, sondern nutzen die Zielfunktion lediglich um die Güte einzelner Lösungen zu bewerten. Die Suche nach einer optimalen Lösung für ein zugrundeliegendes Problem wird ohne Wissen gestartet. Möglichkeiten, Wissen über den Lösungsraum zu generieren und Lösungen hoher Güte zu ermitteln, bieten sowohl Simulationen als auch Heuristiken. Heuristiken sind durch einen iterativen Suchprozess im Lösungsraum gekennzeichnet. Innerhalb des Lösungsraums betrachten sie mehrere Teilbereiche, die sich durch iterative Veränderungen der bereits untersuchten Lösungen erschließen. Ziel über mehrere Iterationen hinweg ist das Auffinden eines Optimums. Aufgrund der Größe des Lösungsraums bei Planungsproblemen wird die Nutzung analytischer Verfahren eingeschränkt oder unmöglich. Hier erweist sich der Vorteil explorativer Verfahren. Sie erschließen trotz eines komplexen Lösungsraums gute Lösungen. (Fritzsche 2009, S. 40)

3.2 Lokale Suche

Die Lokale Suche bildet ein iteratives Suchverfahren (Zäpfel und Braune 2005, S. 3), welches sich durch die Bildung einer Nachbarschaft kennzeichnet. Das Prinzip wird im folgenden Abschnitt erläutert. Anschließend wird eine Vorstellung mehrdimensionale Lösungsräume vermittelt, die das Prinzip der lokalen Suche greifbarer machen.

3.2.1 Prinzip der Nachbarschaftsgenerierung

Die lokale Nachbarschaftssuche stellt meist die Basis für andere heuristische Verfahren dar. Denn sie eignet sich zur Durchsuchung eines gegebenen Lösungsraums. Lokale Suchverfahren sind durch die Bildung von Nachbarschaften gekennzeichnet. Eine Nachbarschaft N(s) einer Lösung s im Lösungsraum besteht aus allen Lösungen, die eine direkte Umgebung der Lösung s bilden. Wird eine Lösung in einem Schritt minimal modifiziert, handelt es sich um einen *Zug*. In der Literatur wird die Modifikation auch *Move* genannt (Biethahn et al. 2004, S. 47). Für ein gegebenes Optimierungsproblem wird eine Nachbarschaftsfunktion bestimmt, die je nach Anwendungsfall die Kriterien für die Modifikationsmöglichkeiten einer Lösung festlegt. Daher kann die Anzahl der Nachbarlösungen auch je Anwendungsfall variieren. (Zäpfel und Braune 2005, S.27 und S. 89)

Um das Prinzip der lokalen Nachbarschaftssuche besser verstehen zu können, zeigt Abbildung 3-1 einen fiktiven Lösungsraum. Ausgehend von einer Startlösung s_1 wurde im ersten Schritt eine Nachbarschaft bestimmt.

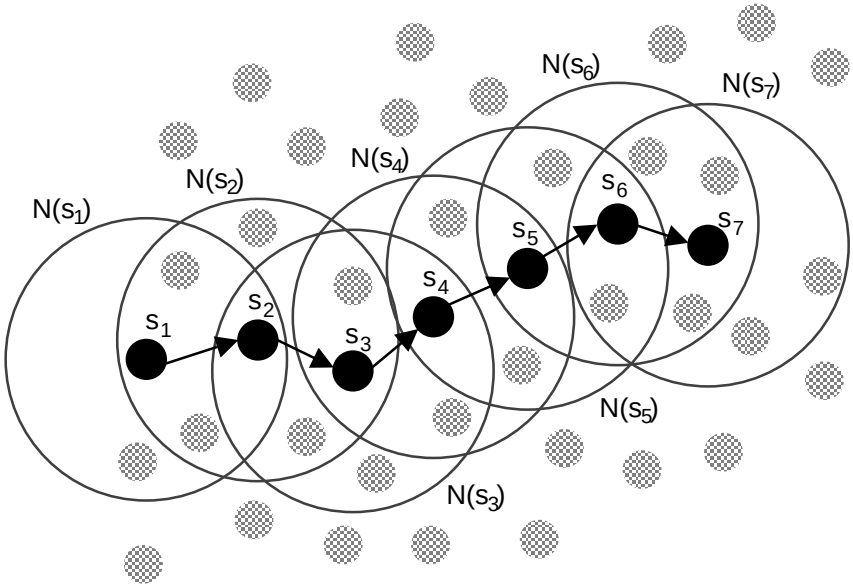

Abbildung 3-1: Allgemeines Prinzip der lokalen Nachbarschaftssuche (nach Zäpfel und Braune 2005, S. 28)

Die Nachbarschaft ist durch den großen Ring um s_1 visualisiert. Alle Lösungskandidaten (graue Punkte), die sich innerhalb dieses Rings befinden, stellen direkte Nachbarn der

Lösung s_1 dar. Jeder durch die Nachbarschaftssuche erreichte Lösungskandidat wird in Abbildung 3-1 als schwarzer Punkt gekennzeichnet. Im nächsten Iterationsschritt wird die Nachbarschaftssuche erneut von Lösung s_2 durchgeführt. In der Nachbarschaft $N(s_2)$ ergibt sich s_3 als bester Kandidat. (Zäpfel und Braune 2005, S. 28)

Die Güte einer Nachbarlösung wird mithilfe einer Zielfunktion bestimmt. Da die Suche über mehrere Iterationen hinweg zu besseren Lösungen führt, wird dieses Prinzip auch schrittweise Verbesserung genannt. Die Gefahr, die bei der lokalen Suche besteht, ist das Steckenbleiben in einem lokalen Optimum. Dadurch findet die beste Lösung der Nachbarschaft spätestens nach ein paar Iterationen immer wieder zum gleichen Kandidat zurück. Es entstehen *Zyklen*, die das Erreichen des globalen Optimums erschweren oder sogar verhindern. Um den Suchprozess bei einer lokalen Suche zu steuern, wird sie daher in Algorithmus integriert, der dann als *Metaheuristik* bezeichnet wird. (Zäpfel und Braune 2005, S. 28 und S.89; Biethahn et al. 2004, S. 45)

3.2.2 n-dimensionaler Lösungsraum

Für die Entwicklung einer Nachbarschaft sollte die Vorstellung über den gegebenen Lösungsraum existieren. Daher soll in diesem Abschnitt kurz auf die Grundlagen von Vektoren und Vektorräumen, wie bei Dietmaier (2014, S. 167-204) detaillierter beschrieben, eingegangen werden.

Ein Vektorraum kann als Menge von Vektoren verstanden werden. Meist werden Vektoren im Dreidimensionalen betrachtet. Ein Zahlentripel würde einen dreidimensionalen Vektor definieren. Für höherdimensionale Vektoren ist hier eine gleiche Betrachtungsweise möglich. Es wird dann nicht mehr von einem Zahlentripel, sondern von einem n-Tupel gesprochen. Abbildung 3-2 zeigt einen Vektor a, der sich in einem Vektorraum befindet, deren Grenzen auf jeder Achse durch den Ursprung und den Parameter a_i mit i ={x, y, z} definiert ist. Jeder Vektor, der in diesen Grenzen definiert werden kann, ist ein Element des Vektorraums. (Dietmaier 2014, S. 170-182)

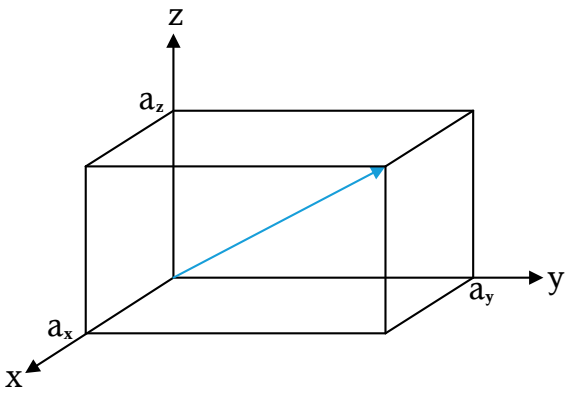

Abbildung 3-2: Dreidimensionaler Vektor (nach Dietmaier 2014, S. 170)

3.3 Differenzierungsmerkmale von Metaheuristiken

Metaheuristiken sind in den 80'er und 90'er Jahren aufgekommen (Biethahn et al. 2004, S. 43). Nach Glover und Laguna (1997, S. 17) ist die Metaheuristik insbesondere mit der Veröffentlichung des Themas *Tabu-Suche* von Glover im Jahr 1986 bekannt geworden. Die Optimierung von Systemen wird heute bereits zum großen Teil durch die Metaheuristiken unterstützt. Ein einfaches Beispiel aus dem Alltag stellt das Navigationssystem dar. Ein Ziel der Metaheuristik ist die Laufzeitverkürzung einer Optimierung. (Bogon 2013, S. 25) Diese wird beispielsweise durch die Steuerung einer lokalen Suche erreicht. (siehe Abschnitt 3.2)

Normale Heuristiken weisen den Nachteil auf, dass sie in einem komplexen Lösungsraum nach genau einem Optimum suchen. Jedoch stellt gerade bei mehrdimensionalen Lösungsräumen das Auftreten mehrerer lokalen Optima ein Hindernis für die normale Heuristik dar. An dieser Stelle wird die Überlegenheit der Metaheuristik erkennbar. (Biethahn et al. 2004, S. 43) Als Metaheuristik kommen mögliche Kombinationen diverser allgemeinerer Ansätze zum Einsatz, die größtenteils selbst eine Heuristik darstellen (Oster 2007, S. 81-82). Durch die Verwendung verschiedener Ansätze in einer Metaheuristik ist der dahinterstehende Algorithmus sogar in der Lage, lokale Optima wieder zu verlassen und die Suche nach dem globalen Optimum fortzusetzen. Durch den Einsatz einer Metaheuristik ist allerdings nicht sichergestellt, dass das tatsächliche globale Optimum im Lösungsraum gefunden wird. (Biethahn et al. 2004, S. 43)

Um Metaheuristiken charakterisieren zu können, werden in den folgenden Abschnitten Differenzierungsmerkmale zur Kategorisierung beschrieben.

3.3.1 Exploration und Fokussierung

Eine Metaheuristik hat unter anderem die Aufgabe, die Suche im Lösungsraum zeitlich effizienter zu gestalten (siehe Abschnitt 3.3). Für den Suchprozess muss entschieden werden, ob detaillierter oder weiträumig gesucht werden soll (Bogon 2013, S. 27). Die Vorgehensweisen werden in der Literatur *Exploration* und *Fokussierung* genannt. (Eiben und Schippers 1998) Synonym wird in *Diversifikation* und *Intensivierung* (Zäpfel und Braune 2005, S. 91) oder *Breiten-* und *Tiefensuche* (Zäpfel und Braune 2005, S. 129-130) unterschieden. Nach Blum und Roli (2003, S. 271) ist es entscheidend das richtige Gleichgewicht von Diversifikation und Intensivierung als Suchstrategien in eine Metaheuristik einzubringen. In der Folge werden einerseits in kurzer Zeit Regionen mit guten Lösungen entdeckt und andererseits keine Zeit in Regionen verschwendet, die bereits untersucht wurden oder die keine guten Lösungen versprechen.

Exploration meint das Erreichen neuer Regionen im Lösungsraum. Die Regionen weisen nach mehreren Iterationen eine bestimmte Distanz zur Startlösung auf. Mit der sogenann-

ten *Hamming-Distanz* als Distanzmaß für binäre Zahlen ist die Anzahl der sich unterscheidenden Stellen gemeint. Als Beispiel ist die Hamming-Distanz zwischen den Zahlen 01001 und 10111 gleich vier, da sie sich an vier Stellen unterscheiden. Die erste Möglichkeit zur Exploration ist bereits sichergestellt, wenn in zwei aufeinanderfolgenden Iterationen eine Hamming-Distanz größer eins vorliegt. Übertragen auf das Beispiel mit den Binärzahlen bedeutet dies, dass von einer Iteration zur nächsten nicht nur eine Stelle, sondern mehrere Stellen geändert werden. Eine zweite Möglichkeit zur Umsetzung der Exploration sieht vor, die Hamming-Distanz von eins beizubehalten, jedoch die Nachbarschaftssuche in folgenden Iterationen nicht mit der besten, sondern einer anderen Lösung fortzusetzen. Die Verwendung einer schlechteren Lösung für die Bildung weiterer Nachbarschaften ermöglicht es, einem lokalen Optimum zu entkommen. Durch die Exploration wird die lokale Suche in neue Regionen verlagert und eröffnet neue Möglichkeiten zur Entdeckung des globalen Optimums. (Zäpfel und Braune 2005, S. 126-129)

Fokussierung meint den lokalen Suchprozess zu intensivieren. Es ist nicht mehr das Ziel, möglichst viele Regionen im Lösungsraum zu erreichen, sondern in einer gegebenen Region ein lokales Minimum aufzusuchen. Die Fokussierung bedient sich dabei dem typischen Vorgehen einer lokalen Nachbarschaftssuche. In jeder Iteration wird die beste Lösung aus den Nachbarschafskandidaten als neue Startlösung verwendet. Die Hamming-Distanz wird für alle Iteration gleich eins gewählt. Dadurch kommen nur Kandidaten infrage, die auch in nächster Nähe liegen. (Zäpfel und Braune 2005, S. 129-130)

3.3.2 Populationsbasierte vs. Einzellösungsverfahren

Die Population stellt ein weiteres Unterscheidungsmerkmal gängiger Optimierungsverfahren dar. Je nach Gestaltung des Berechnungsvorgangs einer weiteren Lösung bzw. weiterer Lösungen kann zwischen populationsbasierten und Einzellösungsverfahren differenziert werden. (Bogon 2013, S.27)

Einzellösungsverfahren setzen den Fokus während des Optimierungsprozesses jeder Zeit auf eine Lösung. Zu Beginn wird eine Initiallösung bestimmt. Von dieser Startlösung werden mithilfe von Nachbarschaftsverfahren weitere Kandidaten im Lösungsraum bestimmt. In jeder Iteration wird die beste Lösung der Nachbarschaft als neuer Startwert verwendet. Dadurch soll sich die Lösung einem Optimum im Lösungsraum nähern. Im Allgemeinen weisen Einzellösungsverfahren einen geringen Speicheraufwand auf, da immer nur der aktuelle Lösungskandidat betrachtet und gespeichert werden muss. Jedoch können im Rahmen Nachbarschaftsbildung nötige Operatoren so viel Speicher in Anspruch nehmen, dass dessen Aufwand nicht zwingend geringer ausfällt als bei anderen Verfahren. Ein Konflikt, den Einzellösungsverfahren aufgrund ihrer Nachbarschaftsbildung mit sich bringen, ist die Terminierung. Durch mehrere Iterationen kann der Fall eintreten, dass

bereits gefundene Lösungen immer wieder betrachtet werden. Die Entstehung solcher Zyklen kann jedoch behoben werden. (Bogon 2013, S. 31)

Populationsbasierte Verfahren bilden den Gegensatz zu Einzellösungsverfahren. Kennzeichen der populationsbasierten Verfahren ist die Generierung mehrerer Lösungen in jeder Iteration. Diese Lösungen werden gespeichert und bilden gemeinsam eine Population. Allgemein können Erkenntnisse aus der berechneten Population zur Bildung einer neuen Generation genutzt werden. Die populationsbasierten Verfahren sind den Einzellösungsverfahren in Bezug auf die Exploration im Lösungsraum überlegen, da sie mehrere Lösungen simultan und nicht nacheinander auswerten. (Bogon 2013, S. 32) Die Differenzierung zwischen Populationsbasierten und Einzellösungsverfahren sowie zwischen den Suchstrategien Exploration und Fokussierung lassen bereits eine Kategorisierung gängiger Metaheuristiken zu. (Talbi 2009, S. 24)

3.4 Einordnung gängiger Metaheuristiken

Talbi (2009, S. 24-25) hat die beiden Kriterien in einen Zusammenhang gestellt. Mit der Einordnung grundlegender Optimierungsansätze auf einer Explorationsskala (siehe Abbildung 3-3) werden die Tendenzen zur Exploration bzw. zur Fokussierung für jeden Verfahrenstyp deutlich.

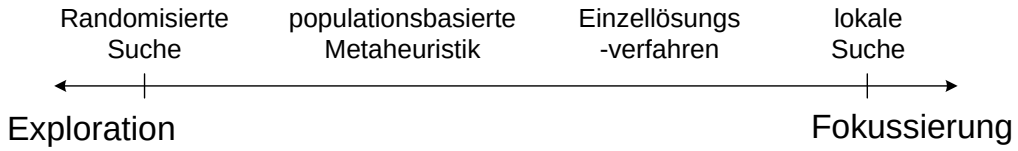

Abbildung 3-3: Explorationsskala (nach Talbi 2009, S. 24)

Abbildung 3-3 veranschaulicht, dass lokale Suchverfahren ihre Priorität auf eine Fokussierung legen und randomisierte Verfahren die stärkste Exploration mit sich bringen. Randomisiertes Verhalten liegt vor, wenn die Exploration im Lösungsraum willkürlich vorgenommen wird (Bogon 2013, S. 37). Aufbauend auf der vorgenommenen Einordnung grundlegender Optimierungsverfahren, sowie dem Wissen über deren Charakteristika (siehe Abschnitt 3.3) werden nun gängige Metaheuristiken vorgestellt.

3.4.1 Simulated Annealing

Der Ursprung des *Simulated Annealings* liegt in der Thermodynamik. Dort wurden bei Abkühlungsprozessen verschiedener Werkstoffe die thermischen Gleichgewichte auf optimale Energiezustände hin untersucht. Als Metaheuristik für kombinatorische Optimierungsprobleme wurde Simulated Annealing erst in den 80’er Jahren zum einen von Kirk-

patrick, Gelatt und Vecchi und zum anderen von Cerny unabhängig voneinander entwickelt. (Zäpfel und Braune 2005, S. 71; Biethahn et al. 2004, S. 95-96)

Simulated Annealing stellt ein typisches Einzellösungsverfahren dar, wie es in Abschnitt 3.3.2 beschrieben wurde. (Bogon 2013, S. 31) Der Ablauf des Simulated Annealings gleicht dem einer lokalen Suche. Beim ursprünglichen Verfahren startet die Suche mit einer hohen Starttemperatur T. Die Variable T wird nun als Steuerungsparameter betrachtet. Dieser steuert die Wahrscheinlichkeit, schlechtere Lösungen für weitere Iterationen anzunehmen. Die Metaheuristik beginnt mit einer zufälligen Lösung, die unter Berücksichtigung des Steuerparameters T initialisiert wird. Die Wahrscheinlichkeit zur Annahme einer schlechteren Lösung ist zu Beginn noch sehr hoch und nimmt mit der Anzahl durchgeführter Iterationen ab. Die Analogie zur lokalen Suche ist an der Generierung von Nachbarschaftslösungen zu erkennen. Denn auch beim Simulated Annealing werden nach jeder Iteration Nachbarschaftslösungen gebildet und einer Bewertung unterzogen. Hierzu wird ein Zielfunktionswert Δf bestimmt, der problemspezifisch definiert ist. Führt die Variation der Lösung zu einer Verbesserung des Zielfunktionswerts, so wird er für die nächste Iteration verwendet. Ist dies nicht der Fall, wird zufällig die Wahrscheinlichkeit zur Akzeptanz der schlechten Lösung bestimmt. Der erste Schritt dazu ist die Generierung einer gleichverteilten Zufallszahl im Intervall [0,1). Anschließend wird der erzeugte Zufallswert mit dem zugehörigen Wert $P(\Delta f)$ der folgenden Akzeptanzfunktion verglichen:

$$P(f\Delta) = e^{\frac{-\Delta f}{T}} \tag{2}$$

Es liegt eine Fallunterscheidung vor. Im ersten Fall ist der erzeugte Zufallswert aus dem Intervall [0;1) kleiner als $P(\Delta f)$. Daraus folgt, dass die Lösung für die nächste Iteration verwendet wird, auch wenn sie keine Verbesserung des Zielfunktionswerts mit sich gebracht hat. Fall zwei kommt zum Zug, wenn der Zufallswert größer als $P(\Delta f)$ ist. Für die folgende Iteration wird der zufällige Wert nicht angenommen und die bisherige Lösung muss erneut verändert und bewertet werden. Die Verwendung schlechterer Werte für weitere Iterationen stellt den großen Vorteil des Simulated Annealings gegenüber einer lokalen Suche dar. Lokale Optima können wieder verlassen werden, da durch den Steuerungsparameter T zufällig schlechtere Lösungen akzeptiert werden. Um den Lösungsraum über die Laufzeit nicht konstant zufällig zu erweitern, wird der Steuerungsparameter über die Laufzeit schrittweise verringert. Dadurch sinkt auch $P(\Delta f)$ und somit die Wahrscheinlichkeit für die Akzeptanz schlechter Lösungen. Der Lösungsraum wird auf Teilgebiete eingegrenzt, die sich durch die Zielfunktionswerte als günstig erwiesen haben. In den Teilgebieten findet zum Ende des Suchprozesses überwiegend eine lokale Suche statt. (Biethahn et al. 2004, S. 97-98)

Der Suchprozess kann unter den Aspekten *Exploration* und *Fokussierung* analysiert werden. Zu Beginn der Suche wird überwiegend exploriert. Die Fokussierung bleibt vernach-

lässigt. Bei steigender Anzahl an Iterationen verringert sich der Steuerungsparameter. Dies führt im Zeitverlauf dazu, dass die Wahrscheinlichkeit für die Fokussierung gegenüber der Exploration zunimmt. (Zäpfel und Braune 2005, S. 136-138)

Um diese Metaheuristik für kombinatorische Optimierungsprobleme nutzen zu können, müssen einige Voraussetzungen erfüllt sein. Zunächst müssen alle Konfigurationsmöglichkeiten eindeutig dargestellt werden können. Des Weiteren muss durch minimale Veränderung einer Konfiguration die Erzeugung einer neuen Konfiguration möglich sein. Zur Generierung neuer Konfigurationen kann die in Abschnitt 3.2 beschriebene Nachbarschaftssuche verwendet werden. Außerdem wird vorausgesetzt, dass ein zur Energie analoger Wert vorliegt, der minimiert bzw. maximiert werden soll. Die letzte Voraussetzung bildet das zwingende Vorliegen eines Kontrollparameters, der die Aufgabe des Steuerungsparameters T übernimmt. (Zäpfel und Braune 2005, S. 72-73)

Neben den Voraussetzungen zur Verwendung des Simulated Annealings sind einige Parameter von Bedeutung, die Einfluss auf die Performance der Anwendung dieser Metaheuristik nehmen. Zu den wichtigen Parametern und Faktoren bei der Implementierung von Simulated Annealing zählen neben dem Initialisierungswert des Kontrollparameters auch die Art und Weise wie der Kontrollparameter verringert werden soll. Des Weiteren nehmen die zu definierenden Abbruchkriterien einen besonderen Stellenwert für das Simulated Annealing ein. (Zäpfel und Braune 2005, S. 76-77)

Abschließend kann festgestellt werden, dass jede lokale Heuristik durch die Implementierung von Simulated Annealing unter Einhaltung aller Voraussetzungen verbessert werden kann. (Biethahn et al. 2004, S. 99). Das wesentliche Alleinstellungmerkmal gegenüber anderen Verfahren ist der kontinuierliche Übergang von einer erst explorativen zu einer zunehmend intensiveren Suche. (Zäpfel und Braune 2005, S. 128)

3.4.2 Genetischer Algorithmus

Der *Genetische Algorithmus* ist den evolutionären Algorithmen zuzuordnen (Bogon 2013, S. 33). Das Ziel des Genetischen Algorithmus ist es, das Prinzip der biologischen Evolution auf Optimierungsprobleme zu übertragen (Zäpfel und Braune 2005, S. 43). Die Basis der Metaheuristik lieferte die Arbeit von J.H. Holland im Jahr 1975, in der er mithilfe der biologischen Evolution als Vorbild versuchte, die Funktionsweise adaptiver Systeme abzubilden. (Nissen 1997, S. 33).

Genetische Algorithmen sind den populationsbasierten Verfahren zugeordnet. Ein bedeutendes Merkmal bei der Generierung von Populationen ist, dass die Größer der Population in jeder Iteration konstant bleibt. Jede Generation besitzt somit die gleiche Anzahl an Lösungskandidaten, die durch eine Fitnessfunktion bezüglich ihrer Qualität bewertet werden. (Biethahn et al. 2004, S. 55-56) Für den Genetischen Algorithmus ist die binäre Dar-

stellung einzelner Lösungen charakteristisch. Mithilfe einer *Kodierungsfunktion* werden Phänotyp und Genotyp einer Lösung ineinander umgewandelt. Die Begrifflichkeiten stammen aus der Biologie. Ein Phänotyp steht in diesem Kontext für eine dekodierte Lösung. Der Genotyp stellt eine kodierte Lösung dar. Abbildung 3-4 zeigt eine beispielhafte Lösung für einen unbestimmten Kontext.

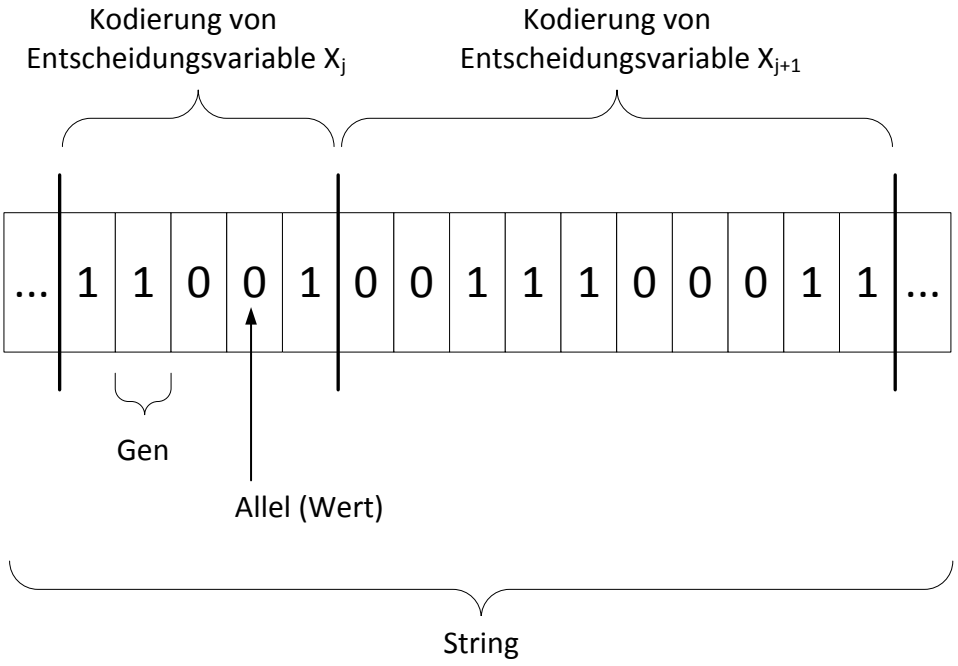

Abbildung 3-4: Binäre Lösungskodierung (nach Nissen 1997, S. 35)

Der Binärstring stellt durch einzelne Bits die einzelnen Gene einer Lösung dar. Ihr angenommener Wert nennt sich Allel und ist bei einer binären Kodierung aus der Menge {0,1} gewählt. Bitfolgen können zu Teilbereichen zusammengefasst werden. Die Kombination aus Allelen, die einen konkreten Teilbereich identifizieren, ergeben *Entscheidungsvariablen* für ein Optimierungsproblem. Die Aneinanderreihung von Entscheidungsvariablen wie X_j und X_{j+1} bildet folglich eine Lösung. (Biethahn et al. 2004, S. 54-55, Nissen 1997, S. 34-35)

Bevor einzelne Lösungen bewertet werden, muss die Anzahl an Lösungen innerhalb einer Population festgelegt werden. Die einfachste Möglichkeit dazu ist die Verwendung einer Zufallszahlengenerierung, welche die Reihenfolge der Nullen und Einsen bestimmt. Einzelne Lösungen werden auch Individuen genannt und liegen in Binärkodierung vor. Um sie mit einer Fitnessfunktion bezüglich ihrer Qualität bewerten zu können, müssen sie erst dekodiert werden. (Biethahn et al. 2004, S. 56-57)

Im Gegensatz zu den Einzellösungsverfahren betrachtet der populationsbasierte Genetische Algorithmus mehrere Lösungen gleichzeitig. Die Berechnung der Lösungsqualität wird somit für jeden gültigen Lösungskandidaten vorgenommen. Aus den besten Lösungen wird

im Anschluss eine neue Population gebildet, die insgesamt bessere Werte liefern soll als die vorherige Population. Um eine neue Population zu generieren, bieten sich verschiedene Operatoren an, die die Gene einer Lösung verändern können. Zu den populärsten Operatoren gehören *crossover*, *mutation* und *adaptation*. (Bogon 2013, S. 33) Sie bilden die Grundmechanismen, die zur biologischen Evolution notwendig sind. In deutscher Literatur werden auch die Bezeichnungen *Rekombination*, *Mutation* und *Selektion* verwendet. (Zäpfel und Braune 2005, S. 43) Um die Population in jeder Iteration zielsicher zu verbessern, müssen positive Eigenschaften der Ausgangspopulation erhalten bleiben. Gute Lösungen sollten in den weiteren Iterationen nur minimal verändert werden. Außerdem muss versucht werden, die guten Eigenschaften zu erkennen und bei der Generierung neuer Lösungen zu berücksichtigen. Neben den bereits genannten Mechanismen bieten sich zudem die Operatoren s*hift* (Verschiebung) und *inversion* (Umkehrung) an. (Fritzsche 2009, S. 42) Der Operator *inversion* dreht die Bestandteile eines gegeben Vektors vollständig um. Beim s*hift* kann in einen *forward shift* und einen *backward shift* unterschieden werden. Im ersten Fall wird eine beliebige Stelle π_i hinter die Stelle π_j verschoben. Dabei muss beachtet werden, dass die Stelle π_i vor π_j liegt. Der Operator Backward Shift bildet den Gegensatz und verschiebt damit eine Stelle nicht hinter, sondern vor eine andere. (Reeves 1999, S. 478-479)

Ein Grund zur Verwendung der aufgeführten Operatoren wurde bereits genannt. Positive Merkmale sollen für die Generierung einer neuen Population erhalten bleiben. Die Verwendung der Operatoren bringt sowohl Vorteile als auch Nachteile mit sich. Nennenswert ist hier die Bedeutung der *Rekombination*. Ziel der Rekombination ist es, vorteilhafte Teilbereiche des Erbmaterials zweier Elternindividuen zu kombinieren, dass die Nachkommen im besten Fall sogar eine höhere Lösungsqualität aufweisen. Durch eine Rekombination werden in jeder Iteration neue Bereiche des Lösungsraums erreicht. Durch diese Art von Exploration wird es ermöglicht, lokale Optima zu verlassen. Wiederholte Selektion und Rekombination würde jedoch die Ausdünnung des Erbmaterials hervorrufen. Dieser Effekt wird auch *verfrühte Konvergenz* genannt. Teilbereiche, die nur im Erbmaterial von Lösungen mit geringerer Güte eingebaut sind, werden so nicht für neue Populationen berücksichtigt. Das Auffinden des globalen Optimums wird nicht mehr möglich. (Zäpfel und Braune 2005, S. 45) Um dieses Phänomen zu verhindern, wird die *Mutation* verwendet. Teilbereiche des Strings sollen bei den Lösungen einer Population möglichst verschieden sein. Andernfalls führt die identische Bitfolge dazu, dass sich die Individuen sich nicht im Lösungsraum ausbreiten können. (Biethahn et al. 2004, S. 59)

Der Algorithmus setzt die Generierung neuer Populationen solange fort, bis ein definiertes Abbruchkriterium erreicht wird. Abbruchkriterien können beispielsweise eine vorgegebene Anzahl an Iterationen oder eine vorgegebene Rechenzeit sein. Ebenso ist es möglich eine Lösungsqualität festzulegen, bei deren Erreichen das Abbruchkriterium greift. Letztlich

kann der Algorithmus auch abgebrochen werden, wenn sich der Zielfunktionswert bzw. die Qualität der Lösung über eine bestimmte Anzahl an Iterationen nicht mehr verbessert. (Biethahn et al. 2004, S. 60)

3.4.3 Tabu-Suche

Die Wurzeln der *Tabu-Suche* liegen bereits in den 70'er Jahren. (Zäpfel und Braune 2005, S. 89) Populär wurde sie allerdings erst mit der Veröffentlichung von Fred Glover im Jahr 1986. (Glover und Laguna 1997, S. 17) Die Tabu-Suche startet auf die gleiche Weise wie die typische lokale Nachbarschaftssuche wie sie in Abschnitt 3.2 beschrieben ist. Von einer beliebigen Startlösung arbeitet sie sich iterativ durch den Lösungsraum. (Glover und Laguna 1997, S. 27) Damit stellt die Tabu-Suche ein typisches Einzellösungsverfahren dar. Die Besonderheit ist, dass sie Zyklen und damit das Steckenbleiben in einem lokalen Optimum verhindern kann. (Bogon 2013, S. 31) Während des Suchprozesses werden einzelne Lösungen vorrübergehend in einer *Tabuliste* gespeichert und sind damit für eine definierte Anzahl an Iterationen *tabu* gesetzt. (Glover und Laguna 1997, S. 29-30) Rückschritte werden verhindert und die Suche in eine neue Richtung gelenkt. Die Tabuliste bildet damit den Hauptmechanismus, um das Steckenbleiben in lokalen Optima zu vermeiden. Es ist durchaus möglich mehrere Tabulisten gleichzeitig zu führen. Durch diese Listen verringert sich die Anzahl möglicher Nachbarschaftslösungen im Lösungsraum. (Biethahn et al. 2004, S. 45; Zäpfel und Braune 2005, S. 89)

Mit der Nachbarschaftssuche und einer Tabuliste ist die Basis für eine Tabu-Suche geschaffen. Zudem sind weitere Elemente kennzeichnend für eine erfolgreiche Anwendung. Die Nachbarschaftssuche dient als Generierungsstrategie neuer Lösungen. Für folgende Iterationen muss entschieden werden, von welcher dieser Lösung eine neue Nachbarschaftssuche gestartet werden soll. Dazu bieten sich zwei grundlegende Selektionsstrategien an. (Biethahn et al. 2004, S. 47-48) Auf der einen Seite wählt die *Best-Acceptance-Strategie* immer die beste Lösung einer gebildeten Nachbarschaft als neue Startlösung aus. Auf der anderen Seite sorgt die *First-Acceptance-Strategie* für eine zeitliche Begrenzung der Nachbarschaftssuche. Sobald in der Nachbarschaft eine bessere Lösung gefunden wurde, die nicht in der Tabuliste aufgeführt ist, wird sie unmittelbar als neue Startlösung gewählt. Weitere Nachbarlösungen werden nicht abgewartet. In der Literatur ist ein Vergleich beider Strategien vorgenommen worden. Dort werden für die genannten Strategien die Bezeichnungen *Best-Improvement-Tabu-Search* (BITS) und *First-Improvement-Tabu-Search* (FITS) verwendet. Zunächst wurde festgestellt, dass die BITS eine intensivere Suche vornimmt und die FITS zu einer besseren Exploration im Lösungsraum führt. Das Ergebnis zeigt, dass die FITS am Ende der Suche eine höhere Lösungsqualität aufweisen kann. (Biethahn et al. 2004, S. 48; Marett und Wright 1996, S. 470-472)

Neben der Selektionsstrategie spielt das *Tabulistenmanagement* für die Tabu-Suche eine wichtige Rolle. In einer Tabuliste können verschiedene Informationen gespeichert werden. Die Möglichkeit zur Verwaltung ganzer Lösungskandidaten wurde bereits erwähnt. In Anbetracht nötiger Speicher- und Rechenkapazitäten, bieten sich Alternativen an, die weniger Kapazitäten in Anspruch nehmen. Es können Lösungsstrukturen, die auf gute Eigenschaften von Lösungen deuten, verwaltet werden. Andererseits ist es denkbar, lediglich Eigenschaften vorteilhafter *Moves* zu speichern. In beiden Fällen wird der Speicherbedarf reduziert. Jedoch kann bei diesem Vorgehen eine noch nicht bewertete Lösung aufgrund ihrer Lösungsstruktur unbeabsichtigt *tabu* gesetzt werden und damit aus der Betrachtung fallen. Um diesen Fall auszuschließen, ist die Speicherung der ganzen Lösung notwendig. (Biethahn et al. 2004, S. 48-50)

Eine ähnliche Abwägung kann für die Tabulistenlänge getroffen werden. Während das *statische* Tabulistenmanagement begrenzte Lösungen speichert und nach einer festgelegten Anzahl an Iterationen wieder freigibt, kann durch ein *dynamisches* Tabulistenmanagement die Bildung von Zyklen vollständig ausgeschlossen werden. Dynamisches Taulistenmanagement meint dabei die fortlaufende Speicherung aller bewerteten Lösungen. (Biethahn et al. 2004, S. 48-50)

Neben den bereits aufgezeigten Charakteristiken der Tabu-Suche sind über die Zeit noch einige Erweiterungen hinzugekommen. *Aspirationskriterien* bilden dabei einen relevanten Aspekt. Allgemein kommt ein Aspirationskriterium zum Tragen, wenn die Durchführung eines sich in der Tabuliste gespeicherten Moves zu der besten bisher gefundenen Lösung führen würde. Mit einem Aspirationskriterium ist es der Suche möglich, das Tabu zu übergehen und trotz des Verbots zur besseren Lösung zu gelangen. (Zäpfel und Braune 2005, S. 90) Nennenswerte Aspirationskriterien, die bei einer Tabu-Suche eingesetzt werden können, sind *Aspiration by Objective*, *Aspiration by Default* und *Aspiration by Search Direction*. Ist eine tabuisierte Lösung aufgrund ihres Zielfunktionswertes besser als alle bisherigen Lösungen, wie im obigen Fall beschrieben, kann das Tabu durch ein Aspiration by Objective Kriterium gebrochen werden. Das Aspiration by Default Kriterium berücksichtigt den Fall, dass alle Nachbarlösungen bereits tabuisiert sind. Um die Suche fortzusetzen, wird die beste tabuisierte Lösung für die weitere Suche verwendet. Neben den Aspirationskriterien gibt es weitere Techniken, die jede für sich eine übergeordnete Steuerung für die Tabu-Suche bilden. Zu diesen zählen insbesondere auch Intensivierung und Diversifikation wie sie in Abschnitt 3.3.1 beschrieben wurden. (Glover und Laguna 1995, S. 98-100; Biethahn et al. 2004, S. 51-52)

Ein weiterer Aspekt, der eine Tabu-Suche ausmacht, ist das Vorliegen von Gedächtnisfunktionen. Dabei wird zwischen einem Kurzzeitgedächtnis (*short term memory*) und einem Langzeitgedächtnis (*long term memory*) unterschieden. Das am häufigsten verwendete und bereits erwähnte Kurzzeitgedächtnis ist die Verwendung einer *Tabuliste*. Da

lediglich die kürzlich gefundenen Lösungen gespeichert werden, wird diese Art von Gedächtnis auch als *recency-based-memory* bezeichnet. Das *frequency-based-memory* bildet das Gegenstück zum recency-based-memory. Hier werden die Häufigkeiten aller Moves gespeichert. Mit einem Blick auf die Verteilung der Häufigkeiten, werden Moves identifiziert, die bisher weniger betrachtet wurden. Die Suche kann so in neue Richtungen gelenkt werden. Sichere Aussagen können allerdings erst nach vielen Iterationen gemacht werden. Daher wird dieser Ansatz auch Langzeitgedächtnis genannt. (Glover und Laguna 1995, S. 94-106; Glover und Laguna 1997, S. 29-31; Biethahn et al. 2004, S. 51-53)

Für viele Optimierungsprobleme reicht die Verwendung eines Kurzeitgedächtnisses aus, um bereits sehr hohe Lösungsqualitäten zu erhalten. Wird jedoch das Langzeitgedächtnis mit in die Tabu-Suche einbezogen, besteht die Chance noch bessere Lösungen zu finden. Die Vorteile des Langzeitgedächtnisses werden meist schneller ersichtlich als erwartet und können eine Tabu-Suche im Allgemeinen deutlich erfolgreicher machen. (Glover und Laguna 1997, S. 93)

3.5 Verfahren mit Schwarmintelligenz

Eine Schwarmoptimierung erfolgt populationsbasiert. Dennoch grenzt sie sich von dem populationsbasierten Vorgehen evolutionärer Verfahren wie dem des Genetischen Algorithmus ab. Der Unterschied zwischen den beiden Verfahren ist, dass Schwärme bei der Entwicklung einer neuen Population lediglich die alten Lösungen verändern, um neue Lösungen zu generieren. Sie halten damit an ihrer bestehenden Population fest. Evolutionäre Verfahren hingegen ermöglichen die Erzeugung wirklich neuer Lösungen. Während beispielsweise der Genetische Algorithmus durch die Verwendung der in Abschnitt 3.4.2 genannten Operatoren neue Regionen im Lösungsraum erreichen kann, orientieren sich Schwärme ausschließlich an bereits evaluierten Lösungen. Deshalb spielt die *Initialisierung* bei der Schwarmoptimierung für eine erfolgreiche Suche, insbesondere unter dem Aspekt der Exploration, eine wichtige Rolle. (Bogon 2013, S. 34-35)

Zu den bekanntesten Verfahren, die zur Lösung von Optimierungsproblemen auf die Schwarmintelligenz setzen, sind die naturanalogen Verfahren wie der Ameisenalgorithmus und die Partikelschwarmoptimierung. (Bogon 2013, S. 34-35)

3.5.1 Ant Colony Optimization

Ant Colony Optimization (ACO-Metaheuristik) hat ihren Ursprung im einfachen Ameisenalgorithmus. Dieser wurde Anfang der 90’er von Dorigo, Maniezzo und Colorino entwickelt. Seine Besonderheit liegt darin, dass er einen Multiagenten-Ansatz verfolgt. Dieser beruht auf der Beobachtung von Ameisen. Diese sind in der Lage als Kolonie größere

Probleme zu bewältigen. Dazu gehört beispielsweise die Nahrungssuche. Ameisen sind in der Lage den kürzesten Weg von ihrem Nest zur Nahrungsquelle zu finden. Im ersten Anlauf werden mehrere Ameisen den Weg zur Nahrungsquelle aufnehmen. Dabei hinterlässt jede einzelne Ameise eine Pheromonspur, deren Konzentration andere Ameisen wahrnehmen können. Die Ameisen, die den kürzesten Weg gefunden haben, werden als erste wieder ins Nest zurückkehren. Dabei erhöhen sie erneut die Konzentration des Pheromons. Die nächsten Ameisen werden den Weg mit der höchsten Pheromonspur bevorzugen. So wird der kürzeste Weg durch die Pheromonspur markiert. (Biethahn et al. 2004, S. 108-109) Im Jahr 1996 erreicht Dorigo et al. eine Formalisierung des Ameisenverhaltens, welche nun die ACO-Metaheuristik ausmacht. (Bogon 2013, S. 36) Die Modifikationen des Ameisenalgorithmus weisen keinen Bezug mehr zur Natur auf. Die Kernprozesse bleiben erhalten, jedoch werden künstliche Ameisen zur Erforschung eines Lösungsraums eingesetzt. Es handelt sich bei der ACO-Metaheuristik um ein populationsbasiertes Optimierungsverfahren, bei dem einzelne Agenten optimale Lösungen aufsuchen und erst durch ihr Zusammenwirken ein globales Optimum gefunden werden kann. Die beim Ameisenalgorithmus verwendete Pheromonspur wird im Rahmen der ACO-Metaheuristik durch eine lokale Variable berücksichtigt. Da die Intensität der Pheromonspur in der Natur über die Zeit abnimmt und auch in der Heuristik ein frühzeitiges Konvergieren vermieden werden soll, bedarf es einer Art Verdunstungsfunktion. Durch diese wird gewährleistet, dass die Suche nicht nur fokussiert, sondern die Exploration in neue Bereiche des Lösungsraums fortschreiten kann. Es ist zu berücksichtigen, dass Sprünge im Lösungsraum ausgeschlossen sind, da die Agenten ihre Suche immer nur in benachbarten Punkten fortsetzen. Anhand der Pheromonspur entscheidet sich ein Agent in welche Richtung er geht. Die Bewegung der künstlichen Ameisen zu einer neuen Nachbarlösung erfolgt über sogenannte Graphen, die für ein gegebenes Problem bereits modelliert sind. Zur Vereinfachung wird angenommen, dass sich die künstlichen Ameisen nur in diskreten Schritten fortbewegen. In jedem dieser Schritte verteilen die Agenten eine bestimmte Menge ihrer Pheromone. Im Gegensatz zur Natur, kann für die künstliche Ameise individuell eine bestimmte Duftmenge definiert werden. Es ist durchaus sinnvoll, diese Menge von der erreichten Lösungsqualität abhängig zu machen. Die besten Lösungen werden so schneller von anderen Ameisen erkannt. (Biethahn et al. 2004, S. 112-117)

Die ACO-Metaheuristik besitzt gegenüber den natürlichen Ameisen einen sehr entscheidenden Vorteil. Denn die Heuristik kann erweitert und damit effizienter gestaltet werden. Wesentliches Augenmerk sollte auf die Hybridisierung mit weiteren Optimierungsverfahren gelegt werden. Die Kombination von Ant Colony Optimization mit einer Tabu-Suche ist durchaus denkbar und wurde explizit von De la Cruz et al. (2011, S. 234-235) für ein definiertes Tourenplanungsproblem genannt. Die Kombination aus lokalem Optimierungsverfahren und der ACO-Metaheuristik als populationsbasiertes Verfahren verspricht eine Steigerung der Effizienz. (Biethahn et al. 2004, S. 114)

Die Verwendungsmöglichkeit einer ACO-Metaheuristik zur Lösung eines definierten Optimierungsproblems muss für jeden individuellen Fall überprüft werden. Wird aufgrund komplexer Nachbarschaftsstrukturen eine kritische Problemgröße überschritten, führt dies zu einem extremen Rechenaufwand. Trotz der aufgezeigten Grenzen der ACO-Metaheuristik gilt sie als leistungsstarkes Verfahren und stellt eines der erfolgreichsten Systeme mit Schwarmintelligenz dar. (Biethahn et al. 2004, S. 108 und S. 118-119)

3.5.2 Partikelschwarmoptimierung

Schwarmintelligenz wurde durch den Ameisenalgorithmus erstmals populär. Russell C. Eberhart und James Kennedy suchten in anderen Schwärmen nach neuen Konzepten. Durch die Betrachtung diverser Schwärme wie den Fischen und Vögeln, entwickelten sie im Jahre 1995 die *Partikelschwarmoptimierung* als neue Rubrik der Schwarmoptimierung. (Bogon 2013, S. 35)

Die Basis der Partikelschwarmoptimierung bilden die drei Grundprinzipien *Evaluation*, *Vergleich* und *Imitation*. Die Position eines Partikels im Lösungsraum wird durch eine Variablen-Belegung p angegeben. Mittels der Evaluation kann die Position p bewertet werden. Um die Positionen einzelner Partikel vergleichen zu können, wird die Bewertung als Fitnesswert gespeichert. Durch den relativen Vergleich dieser Fitnesswerte wird die Lösungsqualität einzelner Partikelpositionen erkennbar. Das dritte Prinzip, die Imitation, kann als eine Lernphase verstanden werden. Damit ein Partikel lernfähig wird, muss es mit einem Geschwindigkeitsvektor v ausgestattet werden. Dieser wird beim Prinzip der Imitation durch folgende Formel berücksichtigt.

$$v_t = v_{t-1} + r_1 * c_1(p_{aktuell} - p_{best}) + r_2 * c_2(p_{aktuell} - g_{best}) \quad (3)$$

In jedem Evolutionsschritt (auch Epoche genannt) erhält der Geschwindigkeitsvektor v_t eines jeden Partikels eine neue Richtung. Dabei berücksichtigt er seine bisher beste gefundene Position p_{best} und die globale beste Position g_{best}, die der gesamte Schwarm bisher auffinden konnte. Ein neuer Geschwindigkeitsvektor v_t berechnet sich aus der vorherigen Richtung v_{t-1} summiert mit den Richtungen, die sich aus den Differenzen der aktuellen Position zu den jeweils besten Positionen ergibt. Mithilfe der Konstanten c_1 und c_2 kann eine Gewichtung vorgenommen werden, die den Einfluss von p_{best} und g_{best} auf den neuen Geschwindigkeitsvektor reguliert. Durch zufällige Vektoren r_1 und r_2, die Werte zwischen Null und Eins annehmen können, wird verhindert, dass sich die Summanden aufheben und der Geschwindigkeitsvektor keinen neuen Wert erhält. Nach einigen Iterationen kann der Fall eintreten, dass ein Geschwindigkeitsvektor zu groß wird. Um diesen Fall zu berücksichtigen, wird eine maximale Geschwindigkeit definiert. Sobald diese erreicht wird, reduziert sich der Vektor auf die Startgeschwindigkeit. (Bogon 2013, S. 39-41 und S. 65)

Das grundlegende Prinzip der Partikelschwarmoptimierung kann noch erweitert werden. Es besteht die Möglichkeit eine parallele Optimierung durchzuführen. Dabei werden mehrere Schwärme gleichzeitig im Lösungsraum initialisiert. Eine Art der Partikelschwarmoptimierung, die eine parallele Optimierung durchführt, ist das *Niching*. Es fokussiert während der Suche nicht nur auf eine Lösung, sondern teilt die Population an Partikeln in mehrere Teile. Im Rahmen agentenbasierter Ansätze ist eine Problemzerlegung in mehrere Teilprobleme üblich. Diese Teilschwärme, oder auch Subschwärme, sind sowohl in der Lage untereinander zu kommunizieren als auch mit anderen Subschwärmen zu kooperieren. Die Kooperation erfolgt über den Austausch von Daten über ihre besten Partikel. Die Vernetzung der besten Partikel wird auch als *Topologie* bezeichnet. Die Topologie eines Schwarms kann unterschiedlich sein. Zwei typische Formen der Vernetzung sind in Abbildung 3-5 zu sehen. In der linken Topologie sind immer nur zwei Partikel von zwei Subschwärmen miteinander im Kontakt. In der rechten Topologie sind alle besten Partikel miteinander vernetzt. (Bogon 2013, S. 42-45)

Ein Wissensaustausch zwischen den Schwärmen beeinflusst das Suchverhalten positiv. Jedoch führt die Kommunikation zu einem erhöhten Rechenaufwand. (Bogon 2013, S.73)

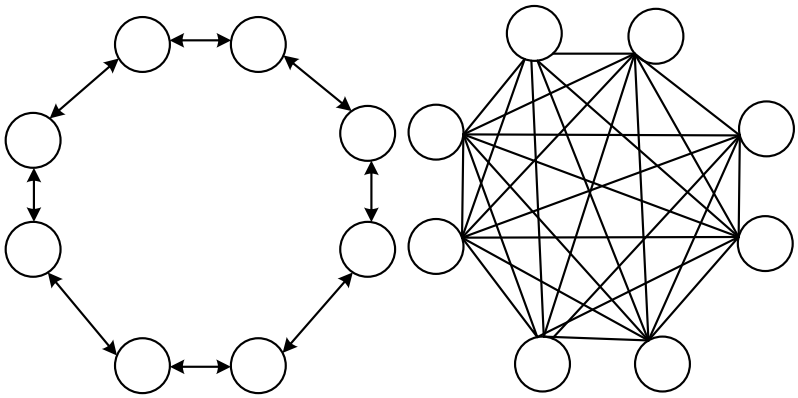

Abbildung 3-5: Topologien von Schwärmen (nach Bogon 2013, S. 43)

Die Kommunikation zwischen den Partikeln kann durch verschiedene Techniken erreicht werden. Eine Möglichkeit dazu bildet die Unterscheidung von Schwarmagenten und einem Koordinationsagenten. Die Schwarmagenten stellen typische Partikel dar und führen die oben beschriebene Partikelschwarmoptimierung durch. Der Koordinationsagent erhält nach einer festgelegten Anzahl von Iterationen relevantes Wissen der eingesetzten Schwarmagenten. Auf der Grundlage berechneter Fitnesswerte kann entschieden werden, welcher Schwarm mehr oder weniger Partikel erhalten soll. Wird einem Schwarm ein Partikel hinzugefügt, ändert sich die Suchrichtung des gesamten Schwarms. Dies nimmt positiven Einfluss auf die Exploration im Lösungsraum. Es sollte berücksichtigt werden, dass sich die einzelnen Schwärme in ihrer Anzahl an Partikeln nicht zu sehr unterscheiden, damit nach einer definierten Anzahl an Iterationen keine Wartezeiten zum Wissensaustausch entstehen. (Bogon 2013, S. 74-77)

Die Partikelschwarmoptimierung bietet die Möglichkeit, die Optimierung mit mehreren Agenten gleichzeitig durchzuführen. Wie auch der Ant Colony Optimization ist es der Partikelschwarmoptimierung möglich, den Suchprozess mit einer Tabu-Suche zu kombinieren. (Bogon 2013, S. 54) Bekrar et al. (2011, S. 4) betonen dabei, dass es wenige Veröffentlichungen gibt, bei denen die Partikelschwarmoptimierung mit der Tabu-Suche als Hybride Metaheuristik eingesetzt wird. Als Grund nennen Bekrar et al., dass die Tabu-Suche meist bei diskreten Problemen verwendet wird.

3.5.3 Diversifikationsstrategien zur Initialisierung von Schwärmen

Da der Erfolg einer Schwarmoptimierung stark von der Initialisierung abhängig ist (siehe Abschnitt 3.5), werden im Folgenden mögliche Strategien aufgezeigt, mit denen ein Lösungsraum weiträumig abgedeckt werden kann.

Die Initialisierung einer Startpopulation erfolgt vor der Evaluierung einzelner Lösungen. Im Rahmen der Generierung von Startlösungen, ist ein besonderes Augenmerk auf den Aspekt der Diversifikation zu legen. (Talbi 2009, S. 193) Eine einfache und häufig verwendete Möglichkeit ist Generierung einer Zufallsgeneration. Dabei wird jede Lösung zufällig und unabhängig von bereits generierten Lösungen erzeugt. Im direkten Vergleich zu diesem Vorgehen bieten sich weitere Diversifikationsstrategien an. Eine davon bildet die *Sequenzielle Diversifikation*. Ausgehend von einer zufälligen Startlösung, werden weitere Kandidaten zufällig im Lösungsraum generiert. Diese Lösungen sind allerdings nur gültig, wenn sie einen bestimmten Abstand zu den bereits generierten Lösungen aufweisen. Nachteilig an dieser Strategie ist, dass die Rechenzeit höher ausfällt als bei vollständig zufällig generierten Initiallösungen. Es wird zwar eine Populationsgröße definiert, jedoch ist im Vorhinein nicht bekannt, wie viele Zufallsgenerationen notwendig sind, um die Anzahl an Lösungen zu erreichen. Schließlich muss die vorgegebene Distanz zwischen den Lösungen eingehalten werden. Die Parametrisierung der Distanz kann einen erheblichen Einfluss auf die Rechenzeit nehmen. Davon abgesehen, führt diese Strategie in jedem Fall zu einer sehr guten Verteilung im Lösungsraum. (Talbi 2009, S. 194-195)

Neben dieser Diversifikationsstrategie gibt es eine weitere, die Vorteile gegenüber einer zufälligen Suche aufweist. Im Rahmen dieser Strategie, wird der Lösungsraum in gleiche Teile zerlegt. Es entsteht ein Raster, welches den Raum in gleich große Bereiche teilt. In jedem dieser Teile wird zufällig eine Lösung gewählt. Dies führt ebenso wie die sequentielle Diversifikation zu einer guten Verteilung der Startlösungen im Lösungsraum. Abbildung 3-6 zeigt den direkten Vergleich zufallsgenerierter Lösungen (rechts) und der Generierung mithilfe der Rasterbildung (links). Beide Vorgehensweisen generieren 25 Lösungen. Und es wird deutlich, dass die Verteilung durch eine Rasterbildung gleichmäßiger vorgenommen wird. (Talbi 2009, S. 195- 196)

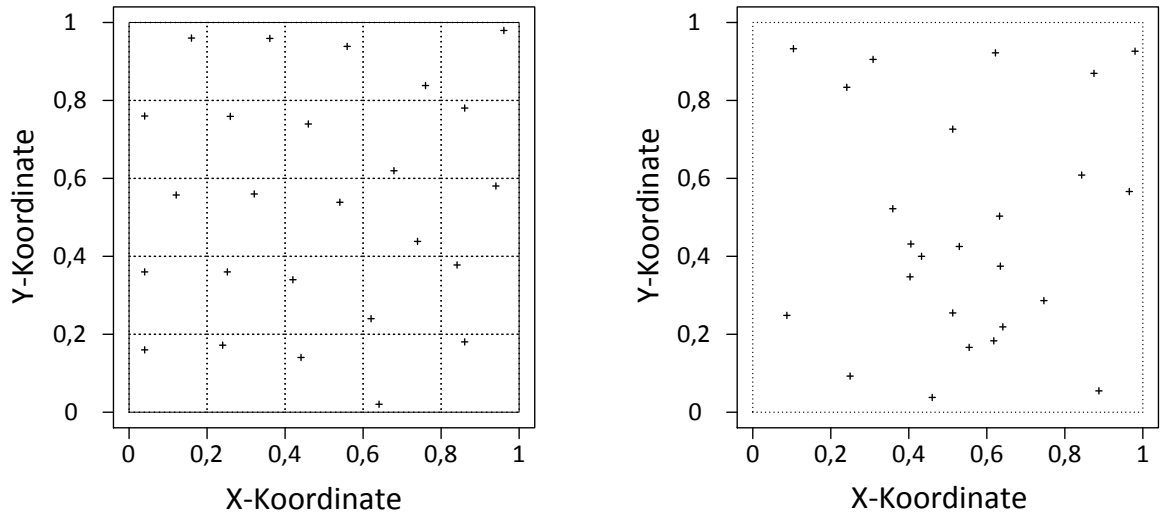

Abbildung 3-6: Rasterbildung gegenüber Zufallsgenerierung (nach Talbi 2009, S. 196-197)

Ein letztes Beispiel zur Initialisierung, unter dem Aspekt der Diversifikation, wird ein quadratisches Zuweisungsproblem vorgestellt. Die Permutation π von m Zahlenwerten {1, 2, …, m} bildet dabei eine potentielle Lösung. Als Beispiel soll nun $m = 6$ angenommen werden. Zur Diversifikation werden i Lösungen X_i (mit i = m) gebildet wie in Abbildung 3-7 dargestellt. (Talbi 2009, S. 196-197)

X_1	5	3	2	1	6	4
X_2	6	2	3	5	4	1
X_3	4	5	6	2	1	3
X_4	3	1	4	6	5	2
X_5	2	6	1	4	3	5
X_6	1	4	5	3	2	6

Abbildung 3-7: Diversifikation bei Permutationsproblemen (nach Talbi 2009, S. 197)

Das Besondere an den Lösungen ist, dass die Permutation π nicht mehr nur in den Zeilen (blau), sondern auch in jeder Spalten (grau) vorhanden sind. Die Distanz zwischen je zwei Lösungen ist dabei gleich $\sqrt{2 * 6}$. (Talbi 2009, S. 196- 197)

3.6 Begründung einer Multiagenten-Tabu-Suche

Aus der Betrachtung verschiedener Optimierungsverfahren, gehen verschiedene Vor- und Nachteile der einzelnen Techniken hervor. Innerhalb dieses Abschnitts werden einzelne Aspekte der Metaheuristiken aufgegriffen und mit der Betrachtung von Agentensystemen die Verwendung einer Multiagenten-Tabu-Suche begründet.

In den letzten Jahren hat sich gezeigt, dass die Verwendung einer einzelnen Metaheuristik bei komplexen kombinatorischen Optimierungsproblemen weniger effizient ist, als wenn sie mit weiteren Optimierungstechniken fusioniert wird. Die sogenannten *Hybrid-Metaheuristiken* ermöglichen ein effizienteres Verhalten und eine höhere Flexibilität. Der Grund dafür ist, dass hybride Konzepte verschiedene Optimierungstechniken in einem Verfahren nutzen. Es bieten sich Kombinationen aus mehreren verschiedenen Metaheuristiken an. Aber auch die Integration künstlicher Intelligenz und ähnlicher Verfahren ist denkbar. (De la Cruz et al. 2011, S. 234)

Im Rahmen der Einordnung gängiger Metaheuristiken wurde speziell auf die einfache *Tabu-Suche* eingegangen (siehe Abschnitt 3.4.3). *Multiagenten-Verfahren* sind besonders mit der ACO-Metaheuristik (siehe Abschnitt 3.5.1) und der Partikelschwarmoptimierung (siehe Abschnitt 3.5.2) thematisiert worden. Im Folgenden werden einzelne Klassifizierungsmerkmale hervorgehoben und Möglichkeiten zur Kombination einer Tabu-Suche mit populationsbasierten Verfahren aufgezeigt. Dabei wird auf Verweise einzelner Abschnitte, die bereits in Kapitel drei herausgearbeitet wurden, verzichtet.

Hybride Metaheuristiken können nicht mehr eindeutig klassifiziert werden. Die Tabu-Suche hingegen stellt ein klassisches Einzellösungsverfahren dar und forciert als Suchstrategie die Fokussierung. Naturanaloge Verfahren wie die ACO-Metaheuristik und die Partikelschwarmoptimierung weisen als populationsbasierte Verfahren eher eine explorative Suchstrategie auf. Das richtige Gleichgewicht von Exploration und Fokussierung kann durch die optimale Verknüpfung der verschiedenen Verfahren erreicht werden.

Als Einzellösungsverfahren ist die *Tabu-Suche* mit dem Nachbarschaftssuchprozess sehr gut dazu geeignet, eine starke Fokussierung an einer bestimmen Stelle im Lösungsraum vorzunehmen. Mit der Tabuliste ist die Suche in der Lage, das Steckenbleiben in lokalen Minima zu vermeiden und weitere Regionen eines Lösungsraums zu erkunden. Es bleibt allerdings die Frage danach, wie der Lösungsraum weiträumig abgedeckt werden kann. Eine einzelne Suche würde einem einzelnen Partikel entsprechen. Dieser könnte einen komplexen Lösungsraum nicht allein bewerkstelligen. Die Suche wird bei der Verwendung von *Simulated Annealing* zunächst explorativ gestaltet und erst nach einigen Iterationen, ausgehend von den bisher besten Lösungen, fokussiert. Um der Suche einen explorativen Charakter zu verleihen, ist es daher sinnvoll, ein multiagentenbasiertes Verfahren zu verwenden. Zur Implementierung populationsbasierter Verfahren besteht die Option, den

Genetischen Algorithmus mit den genannten Operatoren wie Rekombination, Mutation und Selektion, zu nutzen. Das Verfahren kennzeichnet sich durch eine binäre Kodierung und führt bei Anwendung der verschiedenen Operatoren zu einer sprunghaften und zufälligen Durchsuchung des Lösungsraums. Der Vorteil dieses Prinzips ist die Vermeidung von Zyklen. Jedoch bietet auch die Tabuliste diese Funktion und macht damit die Verwendung zufälliger Lösungen hinfällig. Die Bildung neuer Generationen, die bessere Lösungseigenschaften aufweisen als die vorherige Population, ist als Prinzip durchaus positiv zu bewerten. Jedoch wird im Rahmen dieser Arbeit der Fokus auf die Tabu-Suche gesetzt.

Für die Kommunikation zwischen einzelnen Partikeln oder Schwärmen wurden bereits Schwarmagenten und Koordinationsagenten erwähnt. Agent und Metaheuristik zu vereinen, ist ein guter Kerngedanke. Während die Heuristik nach einer optimalen Lösung sucht, verwendet der Agent dieses Wissen, um sich im Lösungsraum zu orientieren. Mit dieser Orientierung ist er in der Lage einen Schwarm durch den Lösungsraum zu lenken. Hinzu kommt die Möglichkeit, Informationen zwischen einzelnen Agenten auszutauschen. (Bogon 2013, S. 55-56) Bogon führt einen weiteren Ansatz an, bei dem Agenten je die Steuerung eines Subschwarms übernehmen. Agenten besitzen Nachbaragenten, denen sie jeweils nach drei Epochen ihre besten Partikeldaten zukommen lassen. Jeder Agent ersetzt dann seine Partikel mit den schlechtesten Fitnesswerten durch die besten Partikel seiner direkten Nachbaragenten. Insgesamt weisen die Partikel dann zwar bessere Fitnesswerte auf, jedoch kann diese Verteilung zu Überschneidungen führen, da die verschiedenen Schwärme teilweise die gleichen Partikel besitzen. Im schlechtesten Fall entwickeln mehrere Subschwärme ein gleiches Verhalten. (Bogon 2013, S. 49). Unter der Annahme, dass dieser Fall nicht zu stark in Gewicht fällt, kann das Prinzip von Bedeutung werden. Die Partikel eines Schwarms erhalten durch die Imitation sowohl einen neuen Geschwindigkeitsvektor als auch eine neue Position. Daher geeignet sich dieses Vorgehen, wenn die Agenten gemeinsam auf der Suche nach genau einem Optimum sind. (Bogon 2013, S. 55)

In Anlehnung an die vorgestellten Schwarmoptimierungen (ACO-Metaheuristik und Partikelschwarmoptimierung) kann ein Agentensuchverfahren entwickelt werden, welches in Kombination mit der Tabu-Suche eine effiziente Hybrid-Metaheuristik darstellt. Da die ACO-Metaheuristik in ihrem Grundprinzip eher auf der Suche nach dem kürzesten Weg ist, kommt diese Art des multiagentenbasierten Ansatzes für die Variation der Anzahl von Systemkomponenten nicht infrage.

Im Rahmen einer Schwarmoptimierung kommt der Initialisierungsphase, insbesondere in Bezug auf den Aspekt der Exploration, eine große Bedeutung zu. Diese ist daher an eine gegebene Problemstellung anzupassen. Wichtige Aspekte, die in der Initialisierungsphase zu berücksichtigen sind, werden von Bogon (2013, S. 65) genannt. Dazu zählt die Verwendung einer optimalen Anzahl von Agenten sowie Partikeln in einem Schwarm. Ebenso müssen die Startpositionen gut im Lösungsraum verteilt sein. Die Betrachtung eines mehr-

dimensionalen Lösungsraums (siehe Abschnitt 3.2.2) und die Verwendung von Diversifikationsstrategien (siehe Abschnitt 3.5.3) bilden die Säulen zur Entwicklung einer geeigneten Initialisierungspopulation. Ausgehend von den definierten Startpositionen arbeiten sich die Partikel durch den Lösungsraum. Dabei sollten für jeden einzelnen stets die Fitnesswerte der anderen Partikel berücksichtigt werden, um den Gesamtüberblick nicht zu verlieren. Mithilfe der Kommunikation zwischen einzelnen Agenten wird dieses Prinzip realisiert. (Bogon 2013, S. 65) Es besteht zudem die Möglichkeit, Lösungen mit besonders guten Fitnesswerten in einer neuen Liste als Elitelösungen zu speichern, damit diese nach einer Iteration nicht verloren gehen. Die Eliteliste kann entweder aus mehreren Lösungen bestehen, oder lediglich einen einzelnen Bestwert speichern.

Die Kombination eines Agentensystems mit der Tabu-Suche als Metaheuristik wird in der Lage sein, eine gute Exploration vorzunehmen und an den besten Positionen mittels der lokalen Suche zu fokussieren. Die Tabuliste wird mit jeder Iteration wachsen und verhindern, dass bereits evaluierte Lösungen nicht erneut betrachtet werden.

Um konkrete Möglichkeiten der Kombination von Metaheuristiken aufzuzeigen, werden im nächsten Kapitel exemplarische Umsetzungen hybrider Metaheuristiken betrachtet.

4 Hybridisierung von Metaheuristiken

Die Betrachtung der Metaheuristiken und der Einsatz von Schwarmintelligenz haben gezeigt, dass hybride Metaheuristiken eine geeignete Wahl zur Entwicklung des Multiagentensystems bilden. Dieses Kapitel wird daher Kombinationen der Optimierungstechniken vorstellen. Insbesondere die allgemeine Betrachtung von PSO-Systemen zeigt Möglichkeiten auf, Zusammenhänge eines Multiagentensystems zu verwirklichen.

4.1 Hybride PSO-Tabu-Suche

Ein Ansatz zur Hybridisierung von Tabu-Suche und Partikelschwarmoptimierung wurde von Bekrar et al. im Jahr 2011 veröffentlicht. Abbildung 4-1 zeigt den Suchprozess in strukturierter Form.

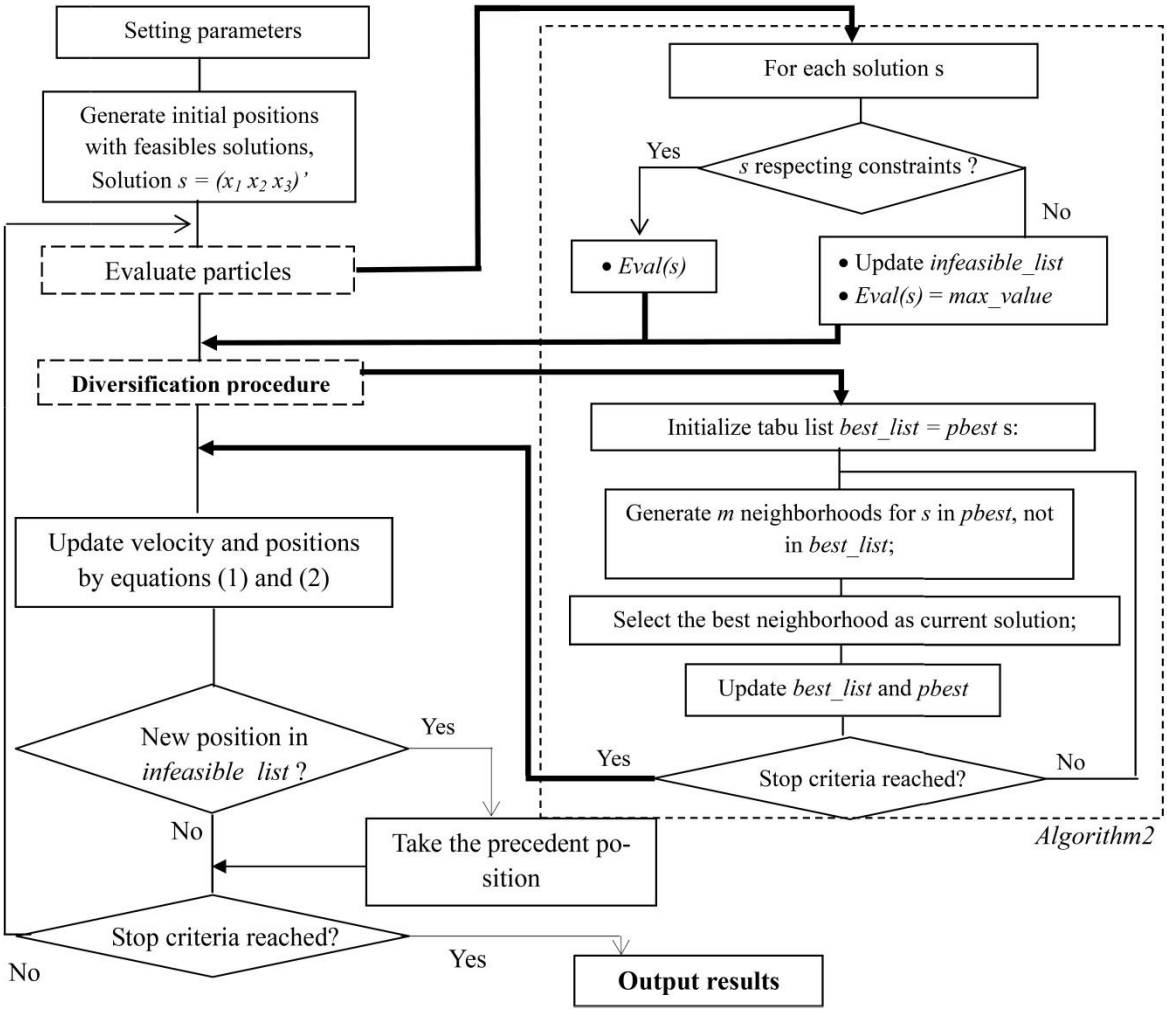

Abbildung 4-1: Suchprozess PSO-Tabu-Suche im Flussdiagramm (Bekrar et al. 2011, S. 7)

Bekrar et al. verwenden zwei Tabulisten, um die Partikelschwarmoptimierung zu unterstützen. Die erste Liste speichert nicht durchführbare Lösungen, die aufgrund definierter Bedingungen bei der Evaluation nicht berücksichtigt werden können. Der Diversifikati-

onsprozess im Ansatz von Bekrar et al. kennzeichnet sich nicht ausschließlich durch die Berechnung neuer Geschwindigkeitsvektoren für einzelne Partikel, sondern verwendet zudem eine lokale Nachbarschaftssuche, die von den besten Partikeln der PSO gestartet wird. Der beste Nachbarkandidat wird in eine zweite Tabuliste aufgenommen. Diese Liste speichert über den gesamten Suchprozess die besten Lösungen, die sich aus der Tabu-Suche ergeben haben. Am Ende des Suchprozesses wird die beste gefundene Lösung ermittelt und ausgegeben. (Bekrar 2011, S. 4 und S. 7; Bogon 2013, S. 54).

Bevor die Suche starten kann, muss die Anzahl an Partikeln für eine bestimmte Population festgelegt werden und die Initialisierungsphase abgeschlossen sein. Diese startet mit der Generierung zufälliger Lösungen, die auf Einhaltung vorgegebener Bedingungen überprüft werden. Ist eine Lösung durchführbar, wird sie der Population hinzugefügt. Dieser Schritt wird so oft wiederholt, bis die Anzahl an Partikeln für die Population erreicht ist. (Bekrar et al. 2011, S. 3) Die einzelnen Partikel der gleichen Population sind nun zufällig im ganzen Lösungsraum verteilt. Der Suchablauf entspricht insofern einer typischen Partikelschwarmoptimierung. Die Tabu-Suche wird mit ihrer lokalen Nachbarschaftssuche speziell auf gut bewertete Lösungen der PSO angewendet. Durch Abbildung 4-1 wird die Konstruktion der Hybriden PSO-Tabu-Suche sehr gut veranschaulicht. Es ist gut zu erkennen, dass die Tabu-Suche parallel zur PSO durchgeführt wird und die Tabu-Suche daher keine zufälligen Startwerte, sondern die bisher besten gefundenen Lösungen der PSO zur lokalen suche verwendet.

4.2 TRIBES

TRIBES stellen ein parameterfreies Partikelschwarmoptimierungssystem dar. (Clerc 2006, S. 18) Durch Maurice Clerc wurden TRIBES im Jahr 2006 erstmals bekannt. Kurz zusammengefasst, bilden TRIBES (Teilschwärme; wörtlich: Stämme) eine effiziente Möglichkeit zur Suche nach guten Regionen in einem Lösungsraum. Jedoch sind sie weniger dazu geeignet, in den lokalen Stellen die Suche zu verfeinern. (Du und Swamy 2016 ,S. 168-169)

Larabi Marie-Sainte et al. (2010, S. 64) beschreiben TRIBES als eine hybride Partikelschwarmoptimierung und nutzen sie im Bereich der Statistik zur Optimierung von Projektionsverfahren. Eine ausführliche Darstellung des Konzepts bietet sowohl Clerc (2006, S.139-150) als auch Cooren et al. (2009, S. 149-178).

Das Konzept TRIBES beinhaltet in seiner Umsetzung mehrere Tribes (auch Sub- oder Teilschwärme genannt), die eine unterschiedliche und variable Größe aufweisen. Es handelt sich um ein Multi-Schwarm-System, welches zum einen eine *Inter-Tribes Communication* und zum anderen eine *Intra-Tribes Communication* verwendet. Inter-Tribes deutet auf die Kommunikation zwischen einzelnen Subschwärmen hin. Somit können die jeweils

besten Partikel im Subschwarm ihre Informationen austauschen. Im Gegensatz dazu, weist die Bezeichnung Intra-Tribes auf die Kommunikation innerhalb eines einzelnen Subschwarms. Daher können die besten und die schlechtesten Partikel eines Subschwarms identifiziert werden. Abbildung 4-2 visualisiert die *Topologie* der Subschwärme. (Cooren et al. 2009, S.153)

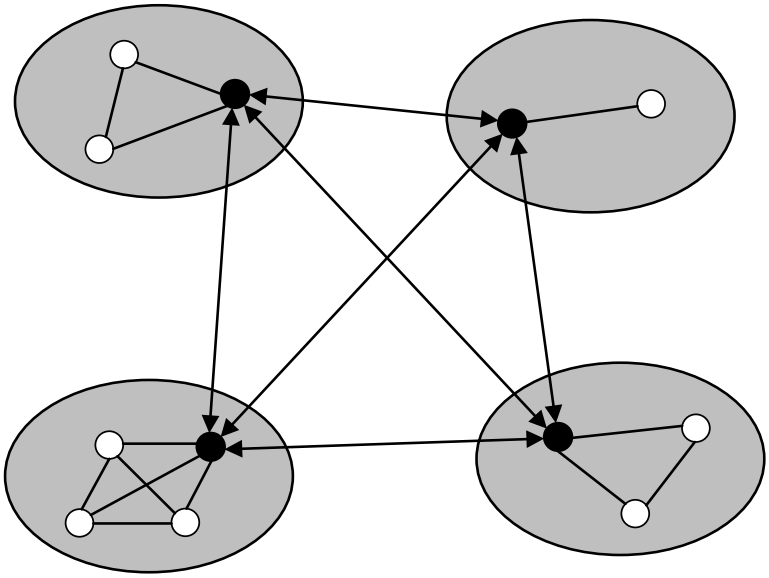

Abbildung 4-2: Topologie der Tribes (nach Cooren et al. 2009, S.154)

Die Pfeile kennzeichnen die Inter-Tribes-Kommunikation, die zwischen den besten Partikeln der verschiedenen Subschwärme stattfindet. Die Intra-Tribes-Kommunikation ist in Abbildung 4-2 durch einfache Linien dargestellt und visualisiert den Informationsaustausch innerhalb eines einzelnen Subschwarms. (Cooren et al. 2009, S.153)

Die *Topologie* des Schwarms wird automatisch erzeugt und verändert. Dafür sind das Hinzufügen (*creation*), die Evolution (*evolution*) und das Entfernen (*removal*) von Partikeln notwendig. Der zeitaufwendigste Teil der PSO ist die Auswertung der Zielfunktion. Daher sollte der Schwarm insgesamt möglichst klein gehalten werden. Eine einfache Möglichkeit bietet das Entfernen (removal) des schlechtesten Partikels aus dem besten Schwarm. Das Ergebnis sollte dadurch nicht negativ beeinträchtig werden. Allerdings kann dieses Vorgehen dazu führen, dass in einem Tribe T1 nur noch ein Partikel P übrig bleibt. Wenn dieser schlechter ist als ein verbundener Partikel Mp eines anderen Subschwarms, so wird der gesamte Tribe T1 entfernt. Abbildung 4-3 veranschaulicht den Sachverhalt. (Cooren et al. 2009, S.154)

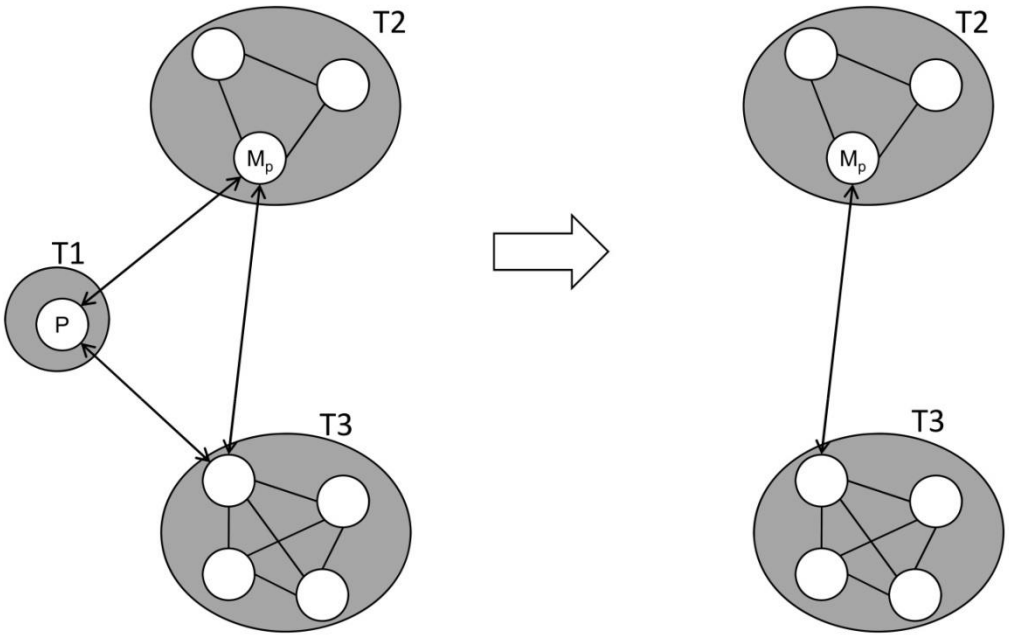

Abbildung 4-3: Removal eines schlechten Tribes (Cooren et al. 2009, S.154)

Die Evolution kennzeichnet in jeder Iteration sowohl die besten als auch die schlechtesten Partikel innerhalb eines Schwarms, aber auch ganze Tribes werden als gut oder schlecht deklariert, je nachdem, ob die einzelnen Partikel des Schwarms eine Verbesserung erreicht haben. Schlecht deklarierte Schwärme generieren neue Partikel (*creation*), die einen neuen Tribe bilden und in direkter Beziehung zum schlechten Tribe stehen. Aus diesem Grund beeinflusst die *Inter-Tribes-Kommunikation* den Optimierungsprozess positiv. (Cooren et al. 2009, S. 154-155)

Bevor Tribes miteinander kommunizieren können, müssen sie initialisiert werden. Daher beginnt die Schwarmoptimierung zunächst mit einem einzelnen Partikel, der einen einzigen Tribe darstellt. Wenn sich dieser in der nächsten Iteration nicht verbessert, wird ein zweiter Schwarm gebildet, wie es im vorherigen Abschnitt geschildert wurde. Auf diese Weise entsteht die Topologie des Schwarms. Die Schwarmgröße wird über mehrere Iterationen hinweg wachsen, womit eine Exploration im Lösungsraum einhergeht. Es ergeben sich zunehmend mehr vielversprechende Regionen mit guten Lösungen. Für die besten Partikel wird dann eine Verschiebungsstrategie gewählt, die einer lokalen Suche ähnelt. (Cooren et al. 2009, S. 156-157)

Der Suchprozess kann durch zwei entscheidende Aspekte konvergieren. Da die Anzahl der Partikel begrenzt ist, werden erstens einzelne schlechte Partikel entfernt. Zweitens werden neue Tribes mit neuen Partikeln hinzugefügt. Der optimale Fall wird erreicht, wenn am Ende des Suchprozesses jeder Tribe genau aus einem Partikel besteht. (Cooren et al. 2009, S.157)

5 Entwicklung einer Multiagenten-Tabu-Suche

Hybride Metaheuristiken stellen eine effiziente Form der Heuristik dar. Die in Kapitel drei erarbeiteten Grundlagen einzelner Suchverfahren werden nun zur Entwicklung einer Multiagenten-Tabu-Suche genutzt. Zudem werden die vorangestellten Konzepte, PSO-Tabu-Suche und TRIBES aus Kapitel vier, die Verknüpfung von Schwarmagenten mit einer Tabu-Suche inspirieren. Insbesondere unter dem Aspekt des Gleichgewichts von Exploration und Fokussierung, müssen beide Verfahrenstypen zielgerichtet eingesetzt werden.

Im Fokus stehen zu variierende Produktionssysteme, die Maschinen oder Transportmittel unterschiedlichen Typs oder unterschiedlicher Menge verwenden. Für die Entwicklung sind die Details einzelner Entscheidungsvariablen zunächst nicht relevant. Das entwickelte Konzept soll auf jede Problemstellung übertragen werden können, die eine Variation der Anzahl einzelner Systemkomponenten betrifft.

5.1 Kodierung für einen n-dimensionalen Lösungsraum

Die in Abschnitt 3.4.2 binäre Kodierung für einen Genetischen Algorithmus soll nicht die Grundlage für eine zu variierende Anzahl von Systemkomponenten sein. Mit der Vorstellung über einen n-dimensionalen Lösungsraum (siehe Abschnitt 3.2.2) ist es sinnvoll die Kodierung in einem n-Tupel vorzunehmen. Einzelne Entscheidungsvariablen werden so genau durch einen Wert des Typ *Integer* dargestellt. Das Ergebnis ist ein Vektor der Länge n, der alle nötigen Parameter enthält. Der Lösungsraum wird entsprechend n Achsen besitzen, auf denen festzulegende Grenzen definiert werden müssen. Um eine Vorstellung über den n-dimensionalen Lösungsraum zu erhalten, wird die Visualisierung eines Vektorraums auf drei Dimensionen reduziert (siehe Abbildung 5-1). Die Variation der Anzahl unterschiedlicher Maschinentypen zeigt einen bespielhaften Lösungsraum. Es sei darauf hinge-

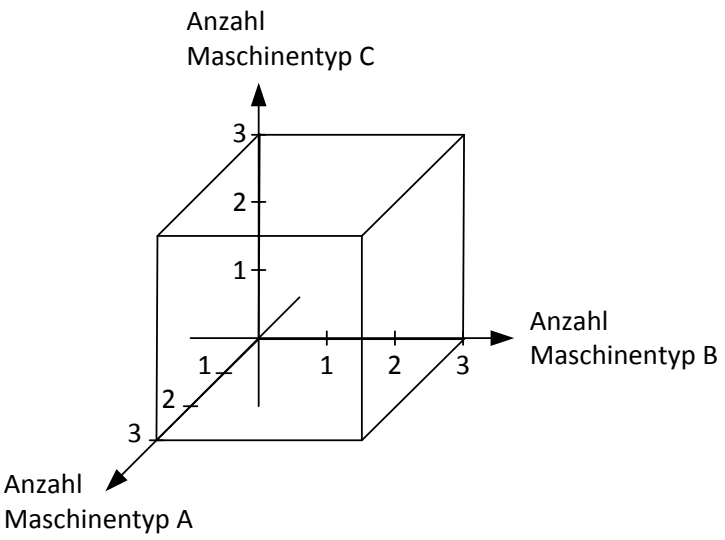

Abbildung 5-1: Lösungsraum zur Variation der Anzahl von Maschinentypen (in Anlehnung an Dietmaier 2014, S. 170)

wiesen, dass die Würfelform die Grenzen des Lösungsraumes visualisiert. Es können lediglich diskrete, hier ganzzahlige, Werte auf einzelnen Achsen angenommen werden.

Aus Abbildung 5-1 wird ersichtlich, welche Lösungen Elemente des Lösungsraums sind. Die Grenzen für jede Achse bilden je der Ursprung und eine Anzahl von drei Maschinen des jeweiligen Typs. Für jedes System muss entschieden werden, welche Grenzen sinnvoll sind. Sind für die Transformationsprozesse verschiedener Produkte oder Produkttypen alle Maschinentypen zwingend notwendig, wird ein Vektor mit einer Anzahl von Null keine sinnvolle Lösung sein. Daher werden ggf. die Grenzen der verschiedenen Maschinentypen mindestens auf Eins gesetzt.

Besteht der Fall, dass die Modellierung des ereignisdiskreten Systems nicht auf Grundlage von Integer-Werten erfolgt, muss eine Funktion zur Kodierung bzw. Dekodierung individuell eingebracht werden, damit das *Dispatching* (siehe Abschnitt 2.4) durchgeführt werden kann. Der angepasst Ablauf einer simulationsgestützten Optimierung ist in Abbildung 5-2 veranschaulicht.

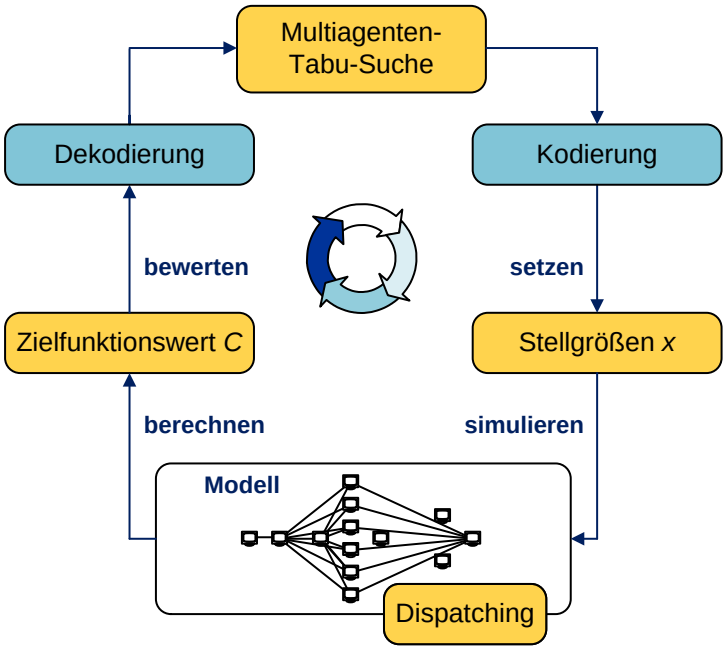

Abbildung 5-2: Angepasster Ablauf simulationsgestützter Optimierung (in Anlehnung an Klemmt et al. 2011, S. 54)

Mit einer Integer-Kodierung können die Stellgrößen, welche den Entscheidungsvariablen entsprechen, in jeder Iteration angepasst werden. Nach der Evaluation wird der Zielfunktionswert für die Multiagenten-Tabu-Suche dekodiert.

5.2 Nachbarschaftsbildung

Die lokale Nachbarschaftssuche (siehe Abschnitt 3.4.3) ist ein wesentliches Merkmal einer Tabu-Suche. Daher soll zunächst das Prinzip am Beispiel zu variierender Systemkompo-

nenten gezeigt werden. Darauf folgt eine Formalisierung in Form eines Aktivitätsdiagramms mittels der Modellierungssprache UML. Weiterführende Literatur zu Aktivitätsdiagrammen bilden Czuchra (2010, S. 117-135) und Balzert (2013, S. 86-92).

5.2.1 Nachbarschaft eines Vektors der Länge n

Für eine lokale Nachbarschaftssuche wird für binäre Kodierungen typischerweise eine Hamming-Distanz von Eins gewählt. (siehe Abschnitt 3.3.1) Die verwendete Integer-Kodierung lässt für einzelne Werte, abhängig von den definierten Grenzen, sowohl eine Addition als auch eine Subtraktion von Eins zu. Damit ergeben sich für die Variation der Anzahl der drei Maschinentypen aus dem Beispiel (siehe Abbildung 5-1) mit einer zufälligen Startlösung folgende Nachbarkandidaten (siehe Abbildung 5-3).

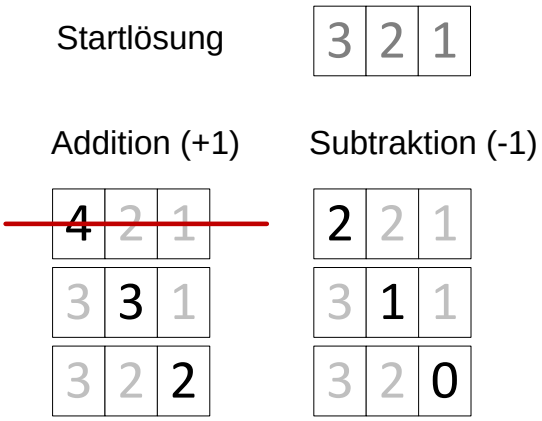

Abbildung 5-3: Bildung einer Nachbarschaft (Beispiel)

Bei der Generierung einer Nachbarschaft wurde zunächst für alle Werte eine Addition bzw. Subtraktion mit Eins durchgeführt und als neue Kombination ausgegeben. Die Lösung (4, 2, 1) stellt keine gültige Lösung dar, weil die obere Grenze von „3“ überschritten wurde.

Das beschriebene Vorgehen kann auf jeden Vektor mit einer beliebigen Länge an Systemkomponenten übertragen werden. Dabei können sich die Grenzen einzelner Entscheidungsvariablen sogar unterscheiden. Es entsteht ein n-dimensionaler Lösungsraum, der durch n obere und n untere Grenzen definiert ist.

5.2.2 Formalisierung mittels UML

Die Eingabeparameter der lokalen Suche stellen die *Startlösung* und die *Distanz* dar. Abbildung 5-4 zeigt den modellierten Ablauf der Nachbarschaftsbildung. Um eine Nachbarschaft einer Startlösung zu erzeugen, muss eine Nachbarschaftsliste angelegt werden. Da

nicht die Startlösung manipuliert werden soll, wird eine Vektorkopie erstellt, die eine Manipulation für mehrere Iterationen übernimmt.

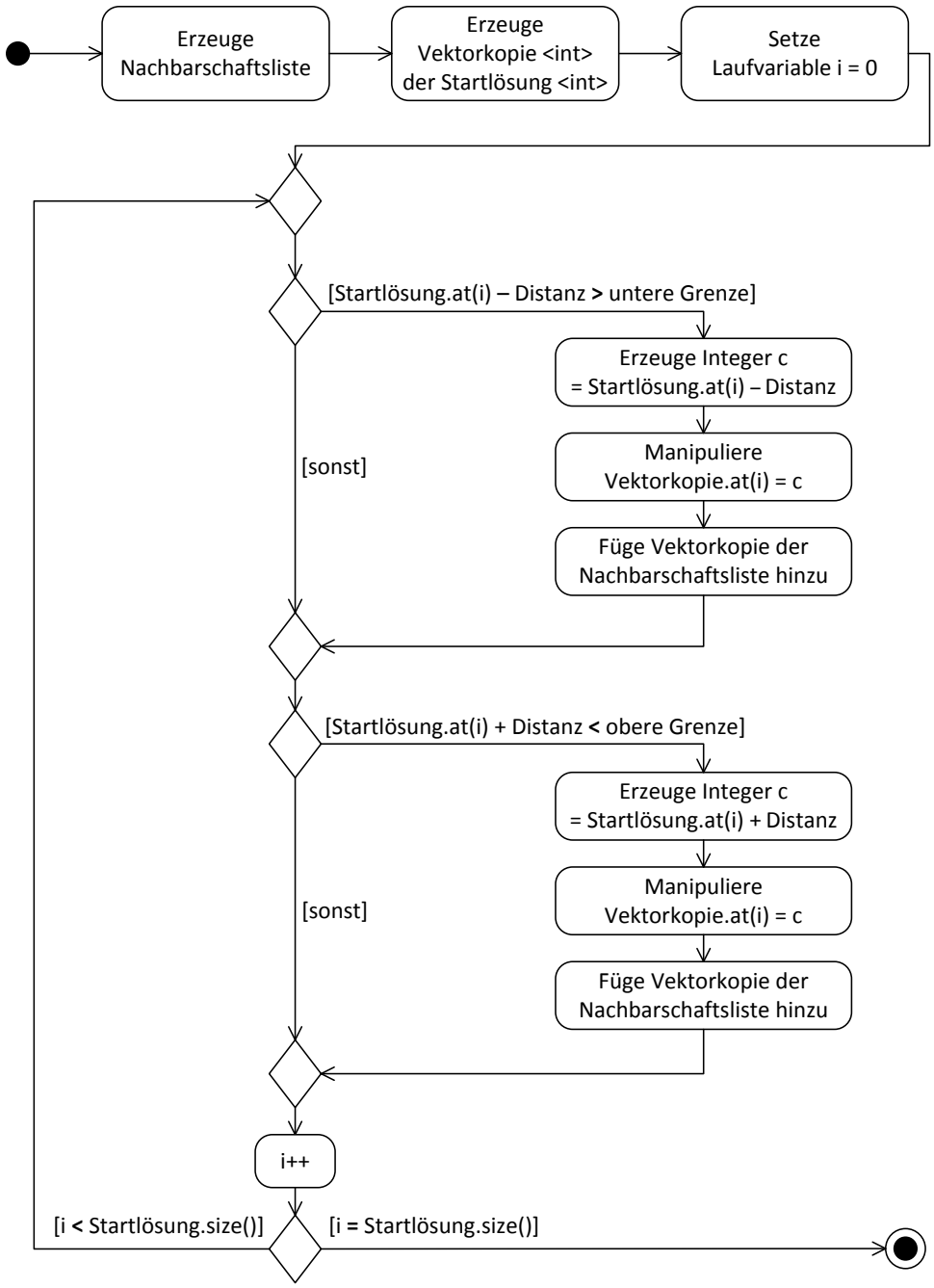

Abbildung 5-4: Aktivitätsdiagramm der Nachbarschaftsbildung

Für jede Stelle Vektorkopie.at(i) eines des Vektors, werden die gleichen Schritte durchgeführt. Falls durch die Subtraktion der Distanz die untere Grenze nicht unterschritten wird, kann die Manipulation erfolgen und der Vektor der Nachbarschaftsliste hinzugefügt werden. Für die Addition der Distanz muss die obere Grenze überprüft werden. Die Ausgabe der Aktivität ergibt eine *Nachbarschaftsliste* mit allen Nachbarn der Startlösung.

5.3 Initialisierung

In einem n-dimensionalen Lösungsraum sollen parallel mehrere Startpositionen initialisiert werden. Besonders für Schwarmoptimierungen muss unter dem Aspekt der Exploration eine gute Verteilung der Startlösungen gewährleistet sein. (siehe Abschnitt 3.5) Dazu soll mithilfe der Diversifikationsstrategien aus Abschnitt 3.5.3 eine geeignete Initialisierungstechnik entworfen werden.

5.3.1 Verwendung einer parallelen Diversifikationsstrategie

Die parallele Diversifikation kann in einem zweidimensionalen Raum erreicht werden, indem eine Rasterbildung vorgenommen wird. Für die Variation einzelner Systemkomponenten in einem Produktionssystem scheint die Bildung eines solchen Rasters nicht trivial. Dennoch kann auch ein n-dimensionaler Lösungsraum in gleicherweise mithilfe eines agentenbasierten Ansatzes zerlegt werden. Um den Lösungsraum gleichmäßig zu zerlegen, müssen m Entscheidungsvariablen zur Bildung von Teilschwärmen verwendet werden. Der Lösungsraum jedes einzelnen Schwarms besitzt folglich eine Dimension von n − m.

Im Beispiel aus Abschnitt 5.1 kann der dreidimensionale Lösungsraum auf zwei Dimensionen reduziert werden, indem z.B. die Entscheidungsvariable für die Anzahl Maschinentyp A genutzt wird, um Teilschwärme zu generieren. Aus der Menge aller Entscheidungsvariablen M = {0, 1, 2, 3} ergeben sich vier zu initialisierende Schwärme. Abbildung 5-5 visualisiert die Zerlegung des dreidimensionalen Lösungsraums in vier Teilräume mit einer Dimension von zwei. Die sich ergebenen Flächen sind in der Abbildung farbig markiert.

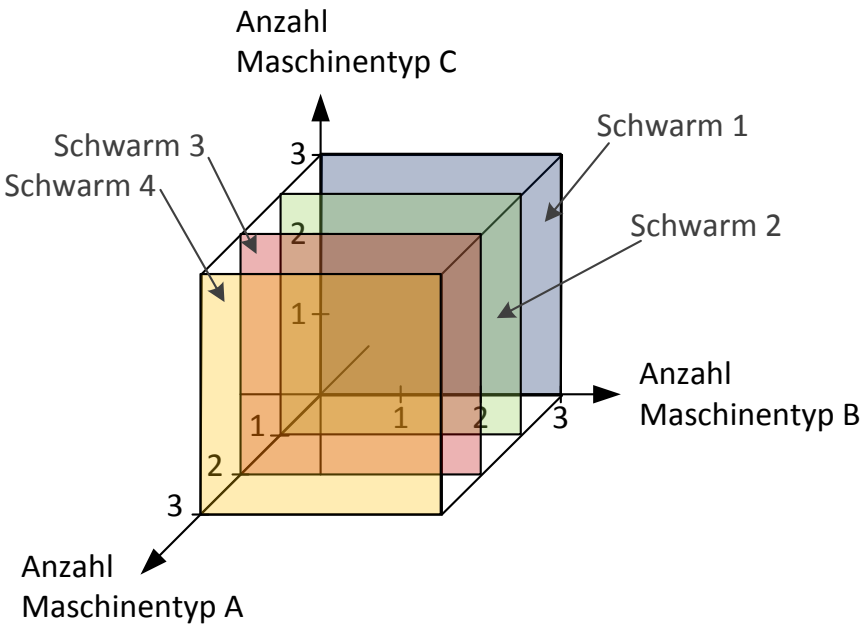

Abbildung 5-5: Lösungsraumzerlegung in vier Schwärme

Da nur diskrete, hier ganzzahlige Werte, angenommen werden können, bildet die Fläche lediglich die Grenzen des Teilraums ab. Die Schwärme müssen innerhalb des (n–1)-dimensionalen Lösungsraums wiederum gleichmäßig initialisiert werden. Dazu bieten sich zwei Möglichkeiten. Entweder kann die Dimension der Teilräume erneut reduziert werden oder die Schwärme werden in den vorhandenen Teilräumen mithilfe einer weiteren Diversifikationsstrategie direkt initialisiert.

5.3.2 Verwendung einer sequentiellen Diversifikationsstrategie

Die sequentielle Diversifizierungsstrategie wurde bereits in Abschnitt 3.5.3 eingeführt. Sie bietet über einen festgelegten Distanzparameter die Möglichkeit, bei der Zufallsgenerierung zufälliger Werte, Lösungen zu verwerfen, die diesen Distanzparameter nicht zu allen bestehenden Lösungen einhalten. Dabei wurde der Nachteil genannt, dass im Vorfeld nicht bekannt ist, wie häufig die Zufallsgenerierung durchgeführt werden muss. Die Rechenzeit kann somit kritisch ausfallen. Der Distanzparameter bewirkt jedoch eine gleichmäßige Verteilung im Lösungsraum und soll daher Bedeutung für den n-dimensionalen Lösungsraum finden. Mit seinen n Achsen, können die Distanzen einzelner Lösungen sehr gut bewertet werden. So kann beispielsweise festgestellt werden, dass die Lösungen des ersten Schwarms in Abbildung 5-5 eine Mindestdistanz von „1“ zum zweiten Schwarm aufweisen, da ihre Lösungsräume auf der Achse für Anzahl Maschinentyp A genau um eine Einheit verschoben ist. Gleiche Distanzen gelten innerhalb der einzelnen Schwärme. Da der Lösungsvektor nur diskrete Zustände annehmen kann, entspricht die Distanz zwischen einzelnen Lösungen mindestens „1“.

Um Startlösungen innerhalb der gebildeten Schwärme gleichmäßig zu verteilen, wird erneut eine Distanz zwischen diesen Lösungen benötigt. Damit die Populationsgröße überschaubar bleibt, muss die Distanz möglichst groß gewählt werden. Für die richtige Parametrisierung der Distanz ist nicht die Dimension des Lösungsraums entscheidend, sondern die gewählten Grenzen der verschiedenen Achsen. Werden die Grenzen auf einer Achse, wie im Beispiel der Maschinentypen, durch das Intervall [0; 3] definiert, so würden sich Distanzparameter von zwei oder drei anbieten. Die Erzeugung der Initiallösungen würde einer Nachbarschaftsgenerierung ähneln, die allerdings eine Distanz von zwei oder mehr verwendet. Ausgehend von einem beliebigen Startvektor werden neue Lösungen generiert, auf die das Prinzip erneut angewendet werden kann. Dopplungen können durch die Verwendung einer *Tabuliste* ausgeschlossen werden. Bevor neu generierte Lösungen der Initialpopulation hinzugefügt werden, wird geprüft, ob diese Lösung bereits vorhanden ist.

Die Formalisierung erfolgt erneut in einem Aktivitätsdiagramm (siehe Abbildung 5-6). Die Aktivität der Nachbarschaftsbildung aus Abschnitt 0 ist grau hervorgehoben und wird mit entsprechenden Eingabeparametern integriert.

Die Initialisierungsliste benötigt für den ersten Durchgang einen Startvektor. Ausgehend von diesem wird die Aktivität in Abbildung 5-6 so oft wiederholt, bis in einer Iteration keine neuen Lösungen dazukommen. Um die Modellierung übersichtlich zu halten, wurde von der Modellierung einer weiteren Schleife abgesehen.

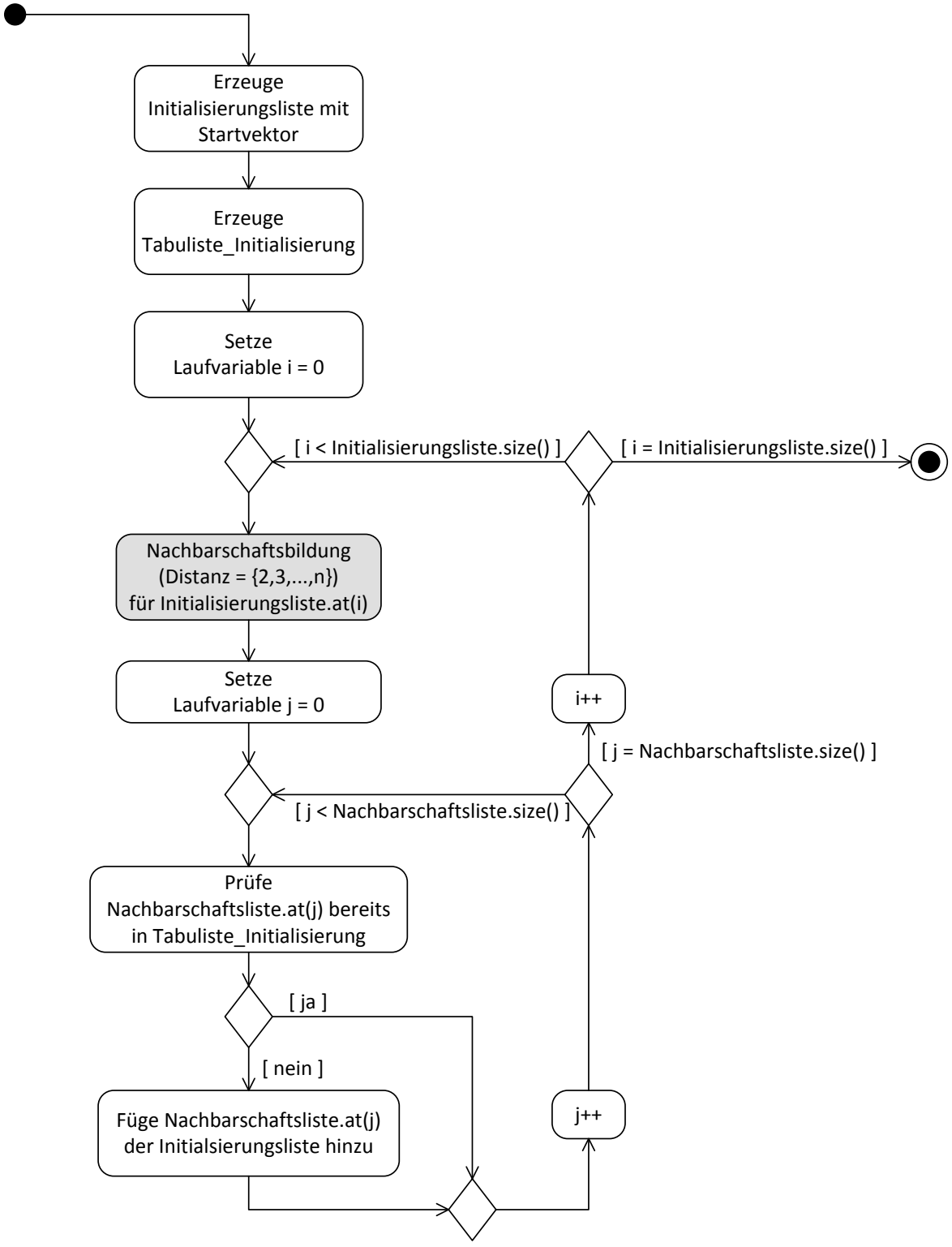

Abbildung 5-6: Aktivitätsdiagramm zur sequentielle Initialisierung

Der Startvektor muss abhängig vom Anwendungsfall bestimmt werden. Um möglichst viele Variationen zu erhalten, sollten die Werte möglichst verschieden sein. Wenn möglich bietet sich eine Permutation der jeweiligen Entscheidungsvariablen an. Im Beispiel der Maschinentypen (siehe Abbildung 5-1) sind die Grenzen in allen Achsen gleich. Eine Permutation ist in diesem Fall nicht möglich, da die Wertemenge {0, 1, 2, 3} für drei Dimensionen einen Wert zu viel beinhaltet. Die Stellen des Startvektors könnten mit Werten der Menge {0, 1, 2, 3} zufällig gefüllt werden. Eine doppelte Verwendung einzelner Werte sollte dabei ausgeschlossen werden.

5.4 Lokale Explorationsstrategie unter Verwendung einer Tabuliste

Das Konzept der Initialisierung unter Berücksichtigung der parallelen und sequentiellen Diversifikation führt zu einer gleichmäßigen Verteilung einzelner Partikel im Lösungsraum. Mit der Evaluation dieser Partikel kann für jeden einzelnen Schwarm die beste Lösung ermittelt werden. Beim *Niching*, als Konzept der Partikelschwarmoptimierung, wird eine Zerlegung in mehrere Subschwärme vorgenommen. Jeder Schwarm konzentriert sich dabei auf genau eine Lösung und vernetzt die Partikel über eine *Topolgie*. (siehe Abschnitt 3.5.2) Die Tabu-Suche fokussiert als typisches Einzellösungsverfahren die beste Lösung einer Nachbarschaft, um die lokale Suche fortzusetzen. (siehe Best-Acceptance-Strategie Abschnitt 3.4.3) Allerdings berücksichtigt eine Simulation den Einfluss stochastischer Ereignisse. (siehe Abschnitt 2.4) Es kann daher der Fall eintreten, dass die in einer Iteration beste gefundene Lösung, nicht generell die beste Lösung bildet. Statt nur den besten Wert für eine Fortführung der Suche zu verwenden, soll für die Multiagenten-Tabu-Suche eine Anzahl von besten Lösungen zur Fortführung der Suche in Betracht gezogen werden. Die Anzahl zu evaluierender Lösungen ist dann zwar höher als bei einem Einzellösungsverfahren und wird eine gewisse Rechenzeit beanspruchen, jedoch gewinnt die Suche an Sicherheit, eine gute Lösung zu finden.

Auf Grundlage des dritten und vierten Kapitels, insbesondere mit den Kenntnissen über die Tabu-Suche, der Partikelschwarmoptimierung, deren Verknüpfung sowie dem vorgestellten Konzept TRIBES, wird nun eine lokale Explorationsstrategie entwickelt. Dabei erfolgt die Initialisierung der Partikel wie in Abschnitt 5.3 beschrieben.

Abbildung 5-7 veranschaulicht für eine grau markierte Startlösung einen Suchprozess, bei dem eine Nachbarschaftsgröße von Neun angenommen wird. Die drei besten Werte der erzeugten Nachbarschaft werden schwarz und größer hervorgehoben. Das Vorhandensein eines n-dimensionalen Lösungsraums muss bedacht werden, aber würde die Visualisierung unübersichtlich erscheinen lassen. Aus diesem Grund wurde eine hierarchische Darstellung gewählt, in der jede Iteration mit den jeweils neuen Nachbarschaften eine Ebene bildet.

Die Fortführung der lokalen Suche über zwei Iterationen zeigt, dass sich die Suche im Lösungsraum ausbreitet.

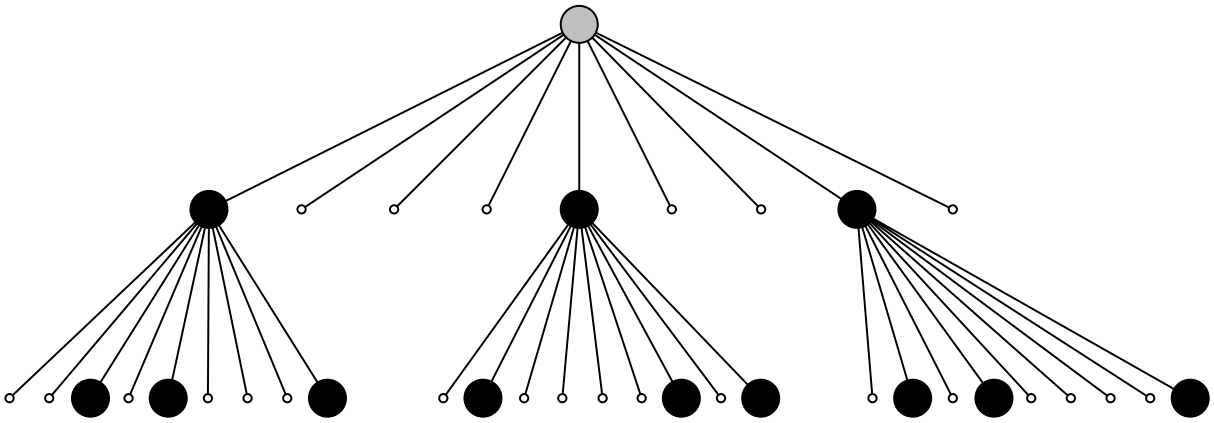

Abbildung 5-7: Lokale Exploration

Das Besondere an diesem Vorgehen ist, dass die Suche nicht nur in eine Richtung verläuft, sondern mehrere Suchrichtungen angestrebt werden. Im obigen Beispiel beläuft sich die Anzahl der Suchrichtungen auf drei. Da die drei Partikel räumlich nicht weit voneinander entfernt sind, liegt es nahe, dass Redundanzen entstehen können. Jedoch kann dieses Problem unter dem Einsatz einer *Tabuliste* vollständig vermieden werden. Die Evaluierung erfolgt somit für jede mögliche Kombination nur einmal. Die Suche ist sehr intensiv und besitzt dennoch einen explorierenden Charakter. Daher bietet das Konzept ein gutes Gleichgewicht zwischen den beiden Suchstrategien. Trotz dieses ausgewogenen Verhältnisses, würde die Einbindung der Simulation mit jedem zusätzlichen Partikel zu einer steigenden Rechenzeit führen. In der Folge muss ein Schwarm an Partikeln möglichst klein bleiben (siehe Abschnitt 4.2). Mit dem Populationswachstum in Abbildung 5-7 lässt sich eine zunehmende Anzahl an Nachbarkandidaten feststellen. Durch eine Art *Evolution* wie im Konzept TRIBES (siehe Abschnitt 4.2) kann nach einer festgelegten Anzahl an Iterationen, die Anzahl der Partikel wieder reduziert werden. Die schlechteren Partikel können durch ein *Removal* einfach entfernt werden und fallen damit für weitere Iteration aus der Betrachtung. Ihre Aufnahme in eine Tabuliste bewirkt zudem, dass sie nicht erneut evaluiert werden können.

Um im Rahmen der Evolution und des Removals einzelne Partikel als beste Partikel zu deklarieren oder sie zu entfernen, muss ein Informationsaustausch über ihre Fitnesswerte stattfinden. Die Schwarm- und Kommunikationsagenten (siehe Abschnitt 3.5.2) und die Inter-Tribes- sowie Intra-Tribes-Kommunikation (siehe Abschnitt 4.2) bilden mögliche Grundlagen für eine *Topologie* der Partikel innerhalb des gesamten Schwarms.

Die in Abbildung 5.6 grau markierte Startlösung stellt eine Art Tribe dar, der zu Beginn der Suche nur aus einem Partikel besteht. In der nächsten Iteration, kommen sehr viele Partikel durch die lokale Suche der Multiagenten-Tabu-Suche hinzu. Die Kommunikation

erfolgt allerdings erst nach einer festgelegten Anzahl an Iterationen. Beispielsweise könnte nach zwei Iterationen eine *Evolution* mithilfe einer Intra-Tribes-Kommunikation stattfinden. Der Tribe wird auf seine besten drei Partikel reduziert (*removal*) und startet unter Berücksichtigung der Tabuliste erneute das Populationswachstum, wie es in Abbildung 5.6 visualisiert ist. Die Inter-Tribes-Kommunikation ermöglicht eine Kommunikation der zu Beginn initialisierten Teilschwärme. Jeder Schwarm stellt so einen Tribe dar, der wiederum mit anderen in Konkurrenz steht. Auch Subschwärme können aufgrund schlechterer Werte vollständig entfernt werden. Wenn beispielsweise nach einer Menge von Iterationen ein Vergleich zwischen den Schwärmen stattfindet und dabei die Partikel eines Schwarms alle schlechtere Fitnesswerte aufweisen als die drei besten Partikel aller konkurrierenden Schwärme, kann die Suche des schlechten Schwarms vorzeitig beendet werden. Die übrigen Teilschwärme stellen schließlich vielversprechendere Regionen im Lösungsraum dar.

Die Suche wird mit einer festgelegten Anzahl durchzuführender Iterationen in seiner Laufzeit begrenzt. Mit einem *Aspirationskriterium* kann die Suche beendet werden, sobald für eine definierte Anzahl an Iterationen keine Verbesserung eingetreten ist. Dieser Parameter muss abhängig vom Anwendungsfall bestimmt werden. Von der Verwendung weiterer Aspirationskriterien (siehe Abschnitt 3.4.3) kann für die Entwicklung abgesehen werden. Ein Tabu soll weder gebrochen werden (Aspiration by Objective Kriterium) noch muss der Fall berücksichtigt werden, dass bereits alle Nachbarkandidaten tabu gesetzt sind (Aspiration by Default), weil die Verwendung mehrerer paralleler Suchrichtungen voraussetzen würde, dass für mehrere Nachbarschaften bereits alle Kandidaten evaluiert sind.

5.5 Tabulistenmanagement

Im Rahmen des Tabulistenmanagements sollen alle nötigen Tabulisten verwaltet werden. Die erste Tabuliste ist bereits bei der Initialisierung erwähnt worden. Als *Tabuliste_Initialsierung* soll sie im Initialisierungsprozess verhindern, dass eine Kombination mehrfach in die Initialisierungsliste aufgenommen wird. Diese Liste wird jedoch nur für den Prozess der Initialisierung verwendet und wird für die weitere Suche nicht mehr benötigt.

Jeder eingesetzte Teilschwarm wird mit den Kombinationen aus der Initialisierungsliste versorgt und evaluiert werden. Die Evaluierung kann bei der Verwendung mehrerer Schwärme nicht zu Redundanzen führen, da sich die Schwärme mindestens durch eine Entscheidungsvariable unterscheiden. Im Bespiel der Maschinentypen, bildet die Anzahl der Maschinen von Typ A bereits das Differenzierungsmerkmal. Es ist nicht nötig, die evaluierten Lösungen aller Schwärme in einer Tabuliste zu verwalten. In der Folge wird für jeden Schwarm eine eigene Tabuliste angelegt, die in jeder Iteration zur Vermeidung

von Redundanzen herangezogen wird. Resultierend wird mit der Tabuliste für jeden Tribe (siehe Abbildung 5-7) verhindert, dass sich die Suchrichtungen überschneiden.

Mit der Tabu-Suche ist der Suchalgorithmus in der Lage, lokale Optima zu verlassen. (siehe Abschnitt 3.4.3) Daher sollten die besten Lösungen während des gesamten Suchprozesses gespeichert werden. Ähnlich wie in der Hybriden PSO-Tabu-Suche (siehe Abschnitt 4.1) wird dazu eine weitere Tabuliste *Bestenliste* eingeführt, die in jeder Iteration die besten Lösungen sichert. Am Ende des Suchprozesses werden über die Bestenliste alle Partikel mit ihrem Fitnesswert ausgegeben.

Im Rahmen des Tabulistenmanagements kann zwischen der Verwendung einer statischen und einer dynamischen Tabuliste abgewogen werden. (siehe Abschnitt 3.4.3) Für die Tabulisten der einzelnen Schwärme ist die dynamische Tabuliste zu bevorzugen, um das Steckenbleiben in einem lokalen Optimum absolut zu vermeiden. Die Länge der Bestenliste ist auf die vorgegebene Anzahl durchzuführender Iterationen begrenzt und kann daher auch als dynamisch angenommen werden.

5.6 Formalisierung des Gesamtkonzepts

Die Entwicklungen sollen in diesem Abschnitt zusammengetragen werden. Mithilfe der Modellierungssprache UML sind die einzelnen Entwicklungsabschnitte in Abbildung 5-8 zusammengeführt worden. Zur Vereinfachung wurden die Prozesse *Evolution* und *Removal* in einer Aktion zusammengeführt, indem eine Schwarmliste auf die drei besten Partikel reduziert wird. Dieser Schritt erfolgt somit nach jeder Iteration. Für eine Umsetzung sind Alternativen wie jede zweite oder dritte Iteration denkbar. Zudem wurde zur Übersichtlichkeit die Schleife für die Laufvariable w nicht modelliert. Das Aktivitätsdiagramm soll lediglich zeigen, dass jeder initialisierte Schwarm den gleichen Evaluierungsprozess durchläuft. Die Aktivität der *sequentiellen Initialisierung* wurde grau markiert, da sie eine Abstraktion der in Abbildung 5-6 modellierten Aktivität bildet. Die Nachbarschaftsbildung ist auch grau markiert und stellt die Vergröberung der in Abbildung 5-4 modellierten Aktivität dar. Als Eingabeparameter für den Suchprozess ist ein Startvektor notwendig, der in der Modellierung als bekannt vorausgesetzt wird. Ebenso muss die Distanz für die sequentielle Initialisierung festgelegt sein. Des Weiteren werden die Anzahl der initialisierten Schwärme sowie die Bestimmung der Aspirationskriterien vorausgesetzt.

Wird ein Aspirationskriterium erreicht, so endet der Suchprozess mit der Ausgabe der *Bestenliste*. Im einfachsten Fall wird dies durch eine definierte Anzahl von Iterationen erreicht.

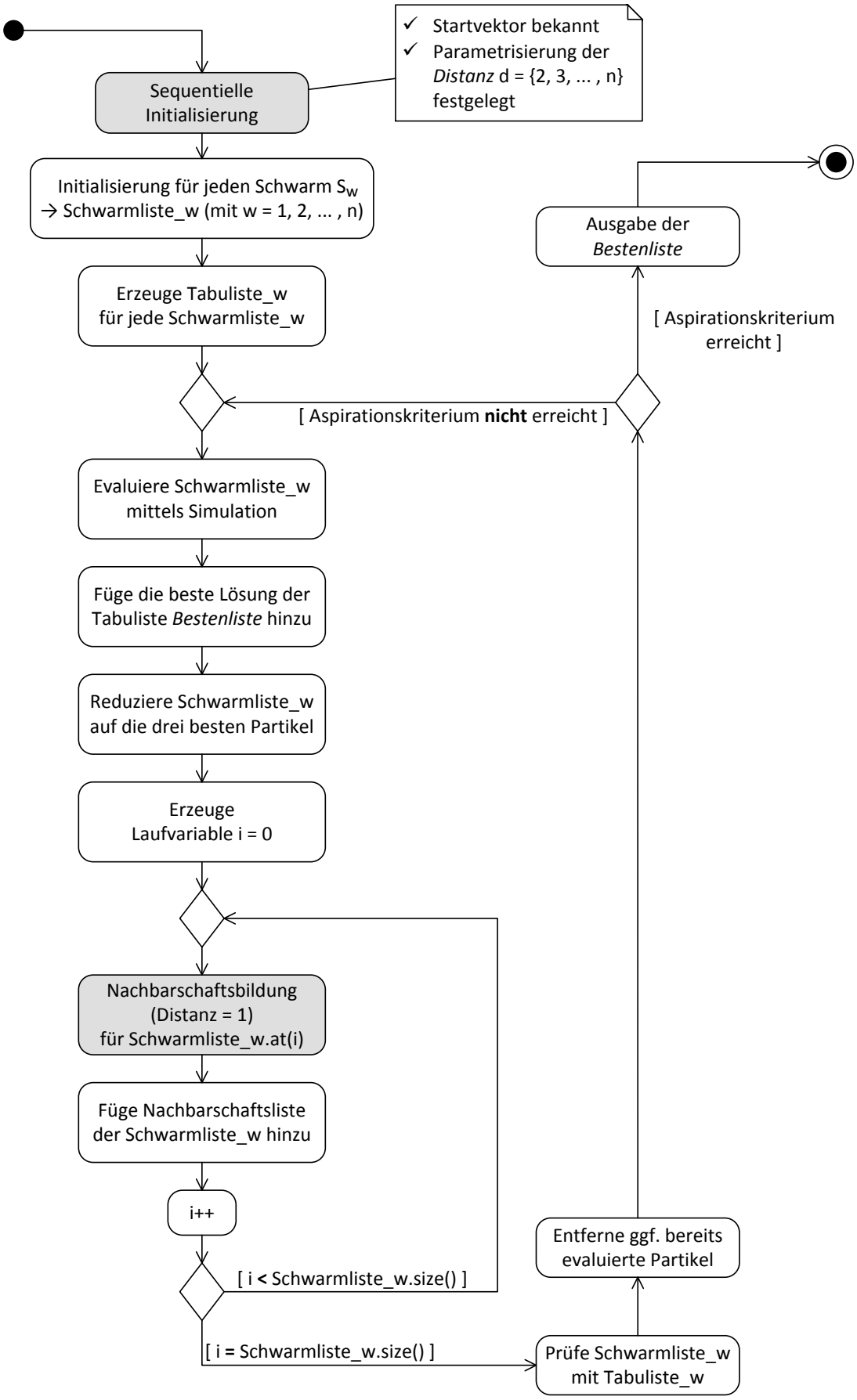

Abbildung 5-8: Aktivitätsdiagramm des Gesamtkonzepts

6 Exemplarische Umsetzung der Entwicklungen

Das Konzept der in Kapitel fünf entwickelten Multiagenten-Tabu-Suche wird im Folgenden exemplarisch auf ein zu variierendes Produktionssystem angewendet. Dazu erfolgt erst die Vorstellung des beispielhaften Systems. Darauf aufbauend werden Anpassungen des allgemeinen Vorgehens aus Kapitel fünf für den konkreten Anwendungsfall vorgenommen. Mit der Implementierung des Suchalgorithmus in eine Entwicklungsumgebung unter Verwendung der Programmiersprache C++ wird die Simulationsgestützte Optimierung wie in Abschnitt 2.4 durchgeführt.

Das Ziel der Optimierung ist es, für ein geplantes Produktionssystem die optimale Anzahl und Verteilung von Systemkomponenten zu bestimmen. Die optimale Anzahl und Verteilung wird durch zwei Kriterien beeinflusst. Zunächst soll die Anzahl eingesetzter Systemkomponenten in Summe minimal ausfallen. Außerdem soll der durchschnittliche Flussfaktor aller Aufträge nahe „3“ sein. Die Evaluierungsfunktion wird daher als Multiplikation beider Kriterien angenommen und im Rahmen des Suchprozesses als Fitnesswert für einzelne Partikel ausgegeben.

6.1 Aufbau des zu variierenden Produktionssystems

Das Produktionssystem, welches zur exemplarischen Umsetzung der entwickelten Multiagenten-Tabu-Suche genutzt wird, findet sich bei Law (2007, S. 694-704). Es handelt sich um ein Fertigungssystem (siehe Abbildung 6-1), welches aus insgesamt fünf Arbeitsstationen besteht und die relevanten Systemkomponenten Maschinen (schwarze Rechtecke) und Gabelstapler aufweist. Des Weiteren werden Aufträge (graue Kreise) über eine Quelle ins System gebracht. Nachdem der Auftrag seinen Transformationsprozess durchlaufen hat, verlässt er das System wieder über die Senke.

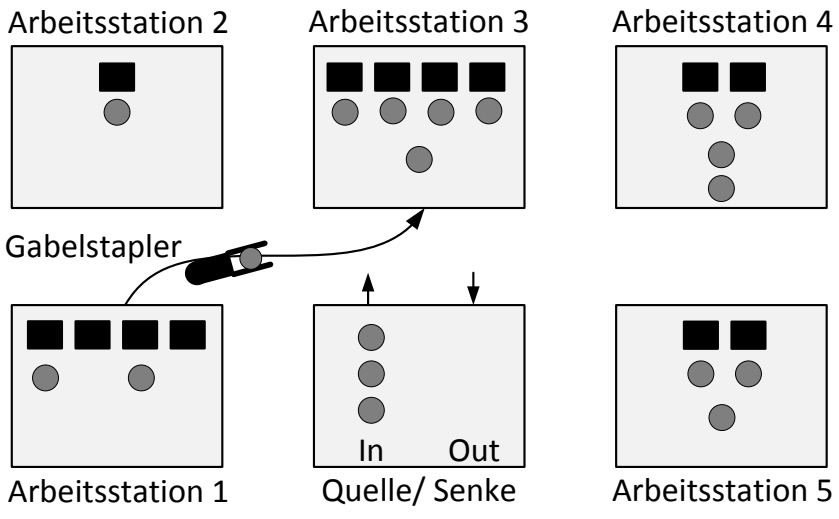

Abbildung 6-1: Aufbau des zu variierenden Produktionssystems (nach Law 2007, S. 694)

Innerhalb der Arbeitsstationen kann die Anzahl eingesetzter Maschinen variieren. Die Anzahl der Maschinen in einer Arbeitsstation stellt somit eine Entscheidungsvariable dar, welche die Werte {1, 2, 3, 4, 5} annehmen kann. Für die Anzahl verwendeter Gabelstapler stehen die Werte {1, 2, 3} zur Verfügung.

Die sich im System befindenden Aufträge können einen von drei verschiedenen Typen annehmen und haben jeweils andere Auftrittswahrscheinlichkeiten. Außerdem sind jedem Arbeitsschritt an den Maschinen für jeden Auftragstyp verschiedene Bearbeitungszeiten vorgegeben, welche durch die ablaufende Simulation berücksichtigt werden. Zudem werden neben den Entfernungen zwischen einzelnen Arbeitsstationen und der Geschwindigkeit der Gabelstapler noch weitere Kennzahlen verarbeitet. Da diese Werte für den Suchalgorithmus keine Bedeutung haben, soll der simulierte Produktionsablauf als Blackbox betrachtet werden.

Das beschriebene System eignet sich zur exemplarischen Anwendung, weil es ein typisches Produktionssystem darstellt. Es kann als typisch bezeichnet werden, da verschiedenartige Aufträge den ihnen zugeordneten Transformationsprozess innerhalb des Systems durchlaufen. Da die Variation nicht für ganze Arbeitssysteme, sondern nur für die Anzahl der eingesetzten Maschinen vorgenommen wird, kann die Veränderbarkeit in Bezug auf die *factory changebilty* der Klasse Rekonfigurierbarkeit mit Tendenz zur Flexibilität zugeordnet werden. Flexibilität liegt noch nicht vollständig vor, da keine grundlegende Veränderung von Materialfluss- oder Logistikveränderung stattfindet. (siehe Abschnitt 2.1)

6.2 Kodierungen innerhalb der Simulationsgestützten Optimierung

Um das beschriebene System variieren zu können, muss zunächst eine Kodierung festgelegt werden, mit der entschieden werden kann, welche Möglichkeiten zur Nachbarschaftsbildung bestehen. Es wird einerseits die Kodierung vorgestellt, die bereits für die Simulation verwendet wird. Andererseits wird eine neue Kodierung entworfen, die den Kriterien einer Nachbarschaft in Abschnitt 3.2 entspricht und den Sachverhalt insgesamt besser darstellt. (siehe Abschnitte 5.1 und 5.2)

6.2.1 Redundanzfreie Kodierung für Entscheidungsvariablen

Für den beschriebenen Anwendungsfall ist eine Kodierung der Entscheidungsvariablen vom Typ *Integer* vorzunehmen. (siehe Abschnitt 5.1) Im Rahmen des *Dispatching*, mit dem die modellierten Petri-Netze für die Simulation vorbereitet werden, kommt eine Kodierung vom Typ *Boolean* zum Einsatz. Diese Kodierung setzt sich aus einem String mit 31 Bits zusammen (siehe Abbildung 6-2). Dabei werden die Ausprägungen *true* und *false* als „1“ und „0“ veranschaulicht.

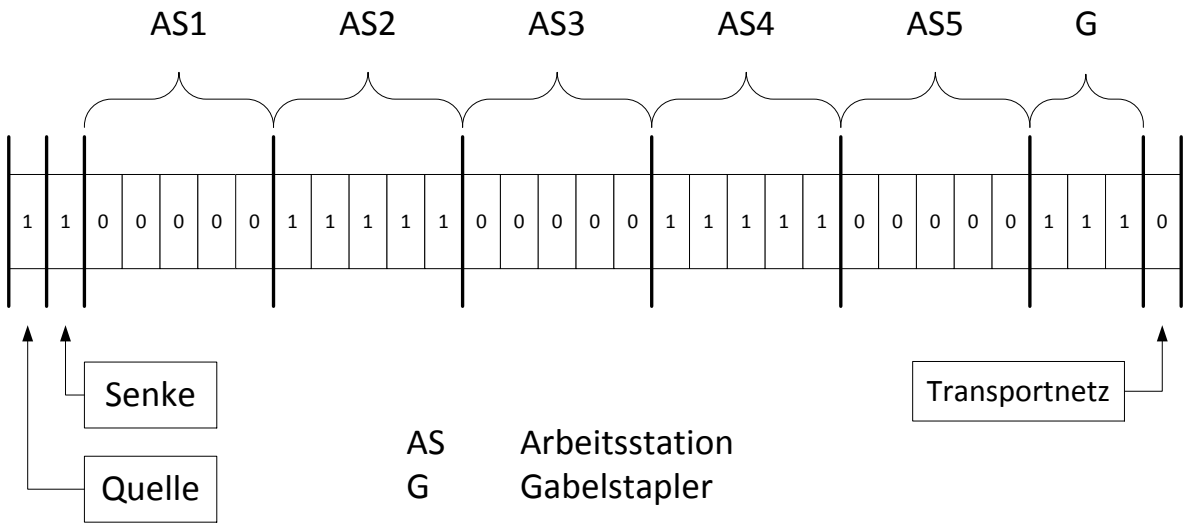

Abbildung 6-2: Binäre Lösungskodierung des zu variierenden Produktionssystems (in Anlehnung an Nissen 1997, S.35)

Die Entscheidungsvariablen sind nach ihrer Arbeitsstation (AS) benannt. Die Kodierung der Entscheidungsvariable AS[j] (mit j = {1, 2, 3, 4, 5}) nimmt durch die Summe verwendeter Einsen eine Ausprägung für die Anzahl eingesetzter Maschinen an. In Abbildung 6-2 nimmt die Entscheidungsvariable AS1 mit (0 0 0 0 0) die Kombination für den Wert von Null Maschinen in Arbeitsstation AS1 an. In Arbeitsstation AS2 sind aufgrund der Entscheidungsvariable AS2 in Summe fünf Maschinen eingesetzt. Die Variable G gibt die Anzahl eingesetzter Gabelstapler an und beträgt im obigen Beispiel drei. Die Entscheidungsvariablen Senke, Quelle und Transportnetz bestehen lediglich aus einem Bit. Das Transportnetz ist eine notwendige Variable, die den Transport innerhalb des modular modellierten Produktionssystems ermöglicht.

Im vorliegenden Fall streckt sich der Binärstring über 31 Bits und könnte aufgrund der Wertemenge {0, 1} für jedes Bit genau einen anderen Wert annehmen. Infolgedessen entsteht eine Nachbarschaft mit 31 Lösungskandidaten. Der erste denkbare Schritt zur Verminderung der Nachbarschaftsgröße ist eine Prüfung auf nicht durchführbare Lösungen. Dazu könnten Bedingungen in eine Tabuliste aufgenommen werden, mit der jede Kombination in einer Nachbarschaft vor einem Simulationsdurchlauf abgeglichen wird. Mit diesem Vorgehen erreichen Bekrar et al. in ihrer Hybriden PSO-Tabu-Suche, dass nicht-durchführbare Kandidaten noch vor einem Diversifikationsprozess tabu gesetzt werden. (siehe Abschnitt 4.1)

Statt nichtdurchführbare Kombinationen zu erkennen, sollte eine erste Bedingung sein, dass die Variablen Senke, Quelle und Transportnetz je mit der Ausprägung *true* angenommen werden. Im Produktionssystem können diese Werte nicht variiert werden, da die Simulation sonst fehlschlägt. Die Bits aus dem String zu entfernen, würde zwei Vorteile

bringen. Die Nachbarschaft würde an Größe verlieren und die Menge nicht-durchführbarer Kandidaten wird massiv reduziert. Die Anzahl möglicher Kombinationen hat sich damit von 2^{31} (≈ 2,1 Milliarden) auf 2^{28} (≈ 2,7 Millionen) reduziert. Es sind jedoch noch weitere Redundanzen zu verzeichnen, die sich durch die Kodierung ergeben. Da für die Anzahl eingesetzter Maschinen ein Vektor vom Typ Boolean verwendet wird, ist es für die Ausprägung „Anzahl Maschinen in AS[j] = 1“ letztlich gleichgültig, ob z.B. die erste oder die fünfte mögliche Maschine in der Arbeitsstation verwendet wird. Dies bekräftigt die Verwendung der in Abschnitt 5.1 entwickelten *Integer*-Kodierung. Auf den konkreten Anwendungsfall übertragen, ergibt sich eine Vektorkodierung vom Typ Integer wie in Abbildung 6-3. Die Arbeitsstationen können dabei Ausprägungen aus der Menge {1, 2, 3, 4, 5} annehmen, die Anzahl der Gabelstapler lediglich aus der Menge {1, 2, 3}.

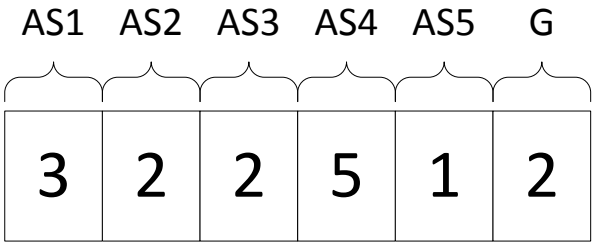

Abbildung 6-3: Vektorkodierung vom Typ Integer

Die Anzahl möglicher Kombinationen entspricht nun der Anzahl durchführbarer Kombinationen. Es bleiben $5^5 * 3 = 9.375$ Kombinationen, die für die Variation der Entscheidungsvariablen für das exemplarische Produktionssystem infrage kommen. Aufgrund der neuen Kodierung sind die Entscheidungsvariablen eindeutig und redundanzfrei.

6.2.2 Kodierungsfunktion

Da nun zwei verschiedene Kodierungen in der Simulationsgestützten Optimierung verwendet werden, ist es notwendig eine (De-)Kodierungsfunktion zu entwerfen, wie sie im Rahmen der Entwicklung und dem damit angepassten Prozess der Simulationsgestützten Optimierung vorgesehen ist. (siehe Abschnitt 5.1) Dazu werden Funktionen benötigt, die eine Kombination für das gegebene Beispiel sowohl vom Typ *Integer* in Typ *Boolean* als auch umgekehrt kodieren können.

Zunächst wird ausgehend von einem Typ Integer in den für das Dispatching benötigten Typ Boolean kodiert. Da die Quelle und die Senke immer *true* sind, werden sie zu Beginn einfach dem auszugebenden Boolean-Vektor hinzugefügt. Gleiches gilt für das Transportnetz, welches den letzten Bit der Boolean-Kodierung definiert. Für die Anzahl der Maschinen und Gabelstapler werden Laufvariablen in einer Schleife erhöht, die je mit der gleichen Stelle des Integer-Vektors verglichen werden. Solange nicht die Anzahl X erreicht ist, wird dem VektorBool ein *true* hinzugefügt. Sobald die korrekte Anzahl des VektorInt erreicht

ist, wird der Rest der Entscheidungsvariable im VektorBool mit *false* aufgefüllt (siehe Abbildung 6-4).

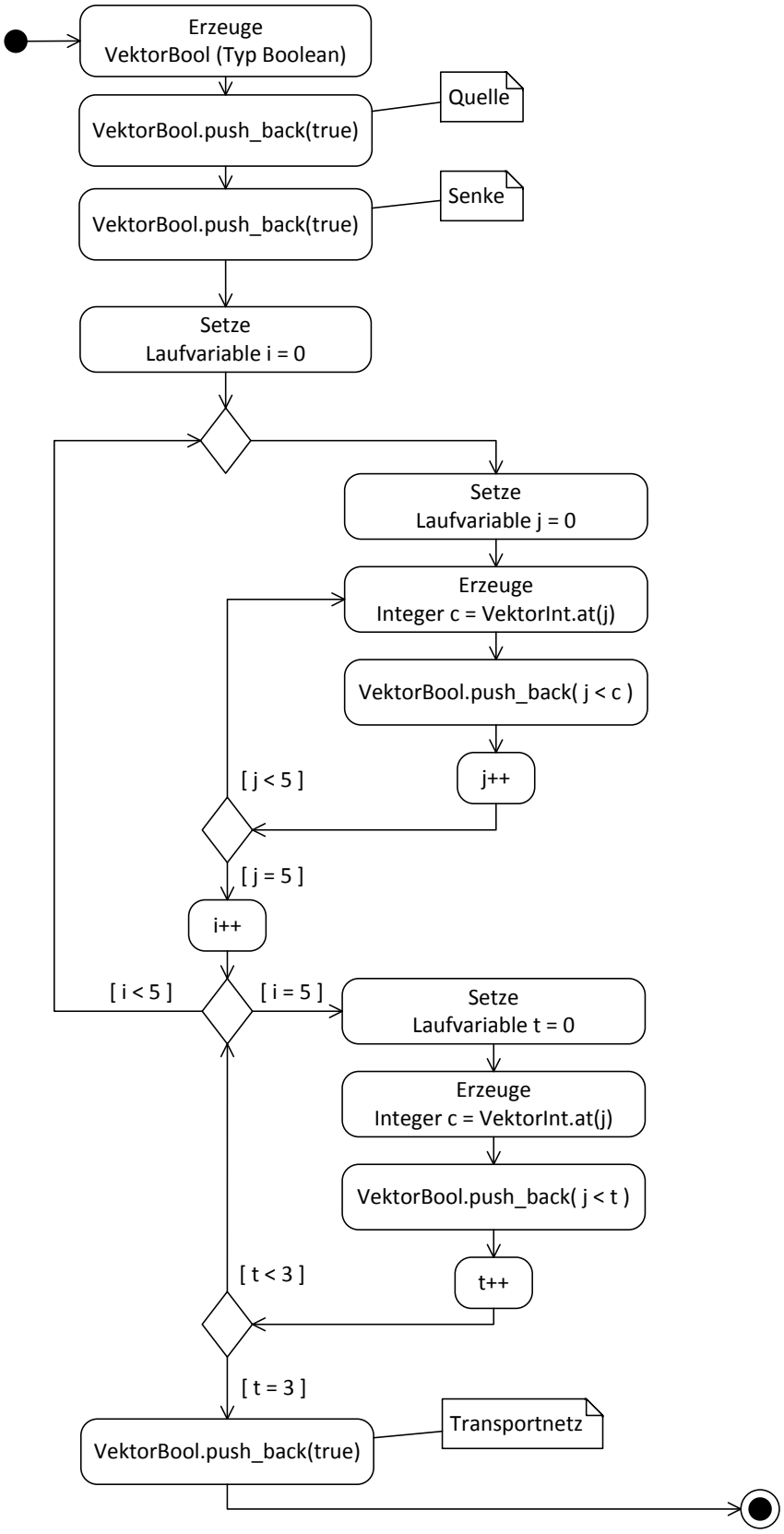

Abbildung 6-4: Aktivitätsdiagramm der Kodierungsfunktion Integer in Boolean

Für die Dekodierungsfunktion werden die Einsen der jeweiligen Entscheidungsvariable addiert und dem Vektor vom Typ Integer hinzugefügt. Auf die Modellierung dieser Funktion wird an dieser Stelle verzichtet, da das Prinzip bereits verständlich ist.

6.3 Lösungsraum der entwickelten Nachbarschaft

Der Lösungsraum wird durch die Anzahl seiner Achsen definiert. (siehe Abschnitt 5.1) Für das exemplarische Produktionssystem definiert der Vektor vom Typ Integer aus Abbildung 6-3 die festzulegenden Entscheidungsvariablen auf sechs Achsen (6-Tupel). Der vorliegende Raum ist somit 6-dimensional. Für jede dieser Achsen sind Grenzen festzulegen, die im Rahmen der Nachbarschaftsbildung berücksichtigt werden müssen. Außerdem kennzeichnen die Grenzen die äußersten Punkte im Lösungsraum. Die Grenzen einzelner Achsen können den Ausprägungsmengen der einzelnen Entscheidungsvariablen entnommen werden. (siehe Abschnitt 6.1) Für die Entscheidungsvariablen AS[j] werden die Grenzen durch das Intervall [1; 5] festgelegt. Für die Anzahl der Gabelstapler werden die Grenzen [1; 3] verwendet.

Mit der vorliegenden Kodierung sind noch 9.375 durchführbare Kombinationen zu durchsuchen. (siehe Abschnitt 6.2.1) Zur vereinfachten Vorstellung des Lösungsraums, wird in Abbildung 6-5 ein Vektorraum visualisiert, der lediglich aus drei Achsen besteht. In diesem Raum befinden sich alle möglichen diskreten Zustände, die als Zahlentripel mit zwei Arbeitsstationen und der Gabelstaplermenge {1, 2, 3} entstehen können. Ein 6-Tupel lässt sich nicht mehr räumlich darstellen. Daher werden die Arbeitsstationen AS3, AS4 und AS5 für die Visualisierung vernachlässigt.

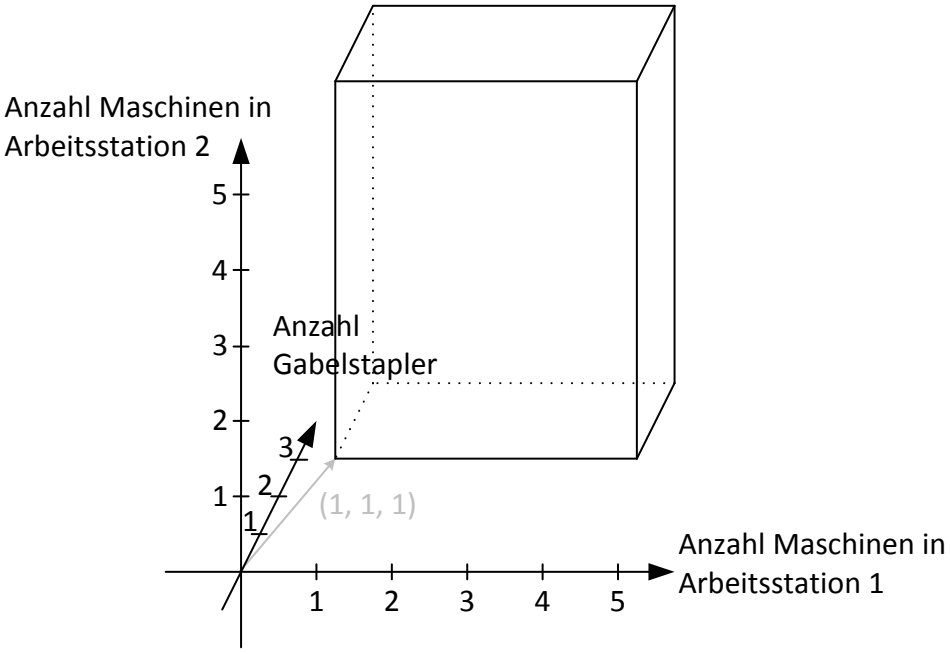

Abbildung 6-5: Vereinfachter Lösungsraum des zu variierenden Produktionssystems

Mit der entwickelten Nachbarschaftsfunktion (siehe Abschnitt 5.2) soll nun exemplarisch eine Nachbarschaft für den vorliegenden Anwendungsfall erzeugt werden. Die Vektorkodierung (3 2 2 5 1 2) (siehe Abbildung 6-3) bildet dabei eine beispielhafte Lösung. Des Weiteren wird als Eingangsparameter eine Distanz d = 1 angenommen.

Die Bildung der vollständigen Nachbarschaft zeigt Abbildung 6-6. Ausgehend vom Vektor (3 2 2 5 1 2) wird jede Stelle um Eins verändert. Auf der linken Seite sind alle Kombinationen gelistet, bei denen sich die Werte je um Eins erhöht haben. Der Vektor mit Wert „6“ ist ungültig, da die obere Grenze von „5“ überschritten wurde. Die rechte Seite zeigt die Kombinationen mit einer Reduzierung um Eins aller Werte. Die Lösung mit „0“ wird ebenso wie die mit „6“ als nicht durchführbar identifiziert und fällt aus der weiteren Betrachtung.

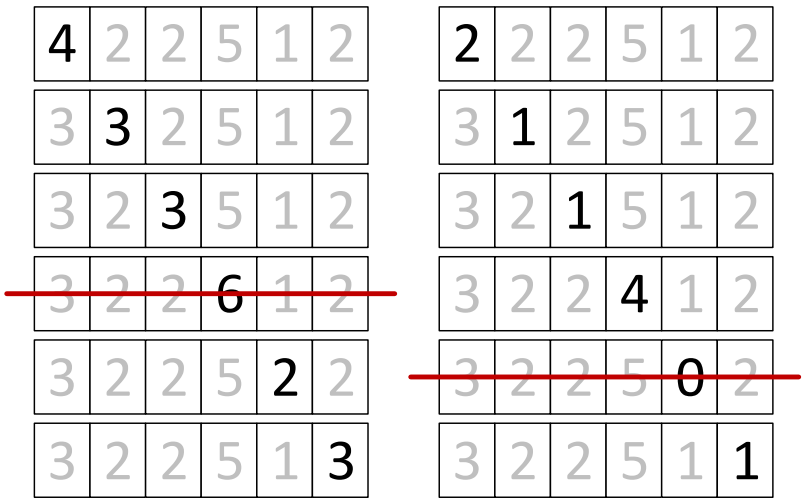

Abbildung 6-6: Beispiel einer Nachbarschaft

Die Ausgabe der Nachbarschaftsliste (siehe Abbildung 6-6) zeigt im Ergebnis zehn gültige Nachbarlösungen. Im Allgemeinen kann die Anzahl der Nachbarschaftskandidaten schwanken. Je nach Anzahl der Einsen und Fünfen in der Startlösung ergeben sich zwischen sechs und zwölf Lösungen.

6.4 Lösungsraumzerlegung mit Agentenbasiertem Ansatz

Aus der Problemstellung für das zu variierende Produktionssystem aus der Kapiteleinführung geht hervor, dass die Anzahl der Maschinen und Gabelstapler in Summe minimal sein soll. Die Ausprägungsmenge für die Anzahl der Gabelstapler ist dabei geringer als die anzunehmende Menge für die Anzahl der Maschinen in den fünf Arbeitsstationen. Es ist zu Beginn unklar, ob die Anzahl der Gabelstapler ein größerer Stellhebel ist, als bisher angenommen wurde. Daher besteht die Option eine getrennte Betrachtung vorzunehmen.

Unter Verwendung der parallelen Diversifikationsstrategie (siehe Abschnitt 5.3.1) soll das System mit seinen sechs Achsen auf fünf Dimensionen reduziert werden. Durch einen

agentenbasierten Ansatz werden drei Schwärme gebildet, die ihren eigenen Lösungsraum besitzen. Jeder Schwarm erhält eine konstante Anzahl an Gabelstaplern und variiert lediglich die Anzahl der Maschinen in den Arbeitsstationen. Durch eine geeignete Initialisierung sollte sich schnell herauskristallisieren, welche Gabelstapleranzahl optimal ist. Das Vorgehen der Dimensionsreduzierung erfolgt nach dem in Abschnitt 5.3.1 entwickelten Konzept. Abbildung 6-7 visualisiert die Zerlegung des Lösungsraums zur Vereinfachung im Dreidimensionalen. Dabei sind die zerlegten Teilräume als Flächen farbig markiert.

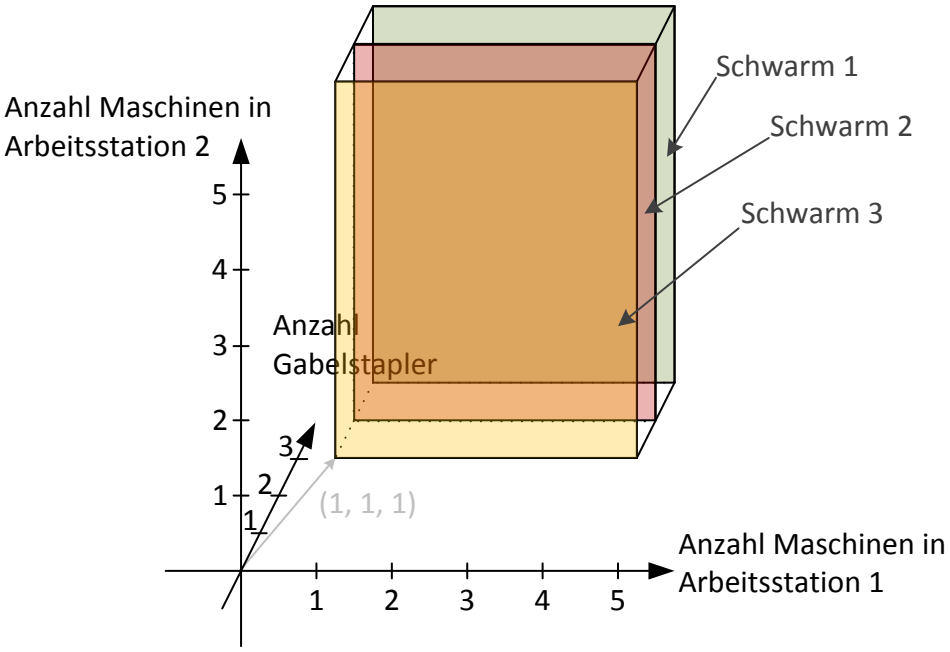

Abbildung 6-7: Vereinfachte Lösungsraumzerlegung

Da nur ganzzahlige Werte, angenommen werden können, bildet die Fläche lediglich die Grenzen des Teilraums ab. Die Schwärme müssen innerhalb des (6 – 1)-dimensionalen Lösungsraums wiederum gleichmäßig initialisiert werden. (siehe Abschnitt 5.3.1)

Um gute Initiallösungen für die Multiagenten-Tabu-Suche sicherzustellen, wird der folgende Abschnitt 6.5 unter Verwendung der sequentiellen Diversifikation (siehe Abschnitt 5.3.2) die entwickelte Initialisierungstechnik auf den Anwendungsfall übertragen.

6.5 Initialisierung einer Startpopulation

Für die Initialisierung einer ganzen Startpopulation sind Diversifikationsstrategien zu berücksichtigen. (siehe Abschnitt 3.5.3) Die in Abschnitt 5.3.2 entwickelte Initialisierungstechnik wird nun ihre Anwendung für das vorliegende Optimierungsproblem finden. Da aufgrund des agentenbasierten Ansatzes (siehe Abschnitt 6.4) die Dimension des Lösungsraums um Eins verringert wurde, erfolgt die Initialisierung lediglich für den fünf-

dimensionalen Lösungsraum und wird anschließend jedem Schwarm bereitgestellt. Als Eingangsparameter für die Aktivität der Initialisierung (siehe Abschnitt 5.3.2) müssen *Startvektor* und *Distanz* optimal bestimmt werden.

Für die Initialisierung der Startpopulation werden nun in Anlehnung an die Strategie der sequenziellen Diversifikation entfernte Regionen im Lösungsraum gebildet. Da die Wertemengen für die Entscheidungsvariablen der Arbeitsstationen auf {1, 2, 3, 4, 5} beschränkt sind, bieten sich Distanzen von zwei oder drei an. Da die Populationsgröße möglichst klein gewählt werden soll, muss die Distanz möglichst groß sein und wird für die folgende Anwendung zunächst mit d=3 angenommen. Zudem muss eine Startlösung festgelegt werden. In Abschnitt 5.3.2 wurde die Verwendung einer Permutation mit der erhöhten Variationsmöglichkeit der Entscheidungsvariablen begründet. Für das zu variierende Produktionssystem wird die Permutation von fünf nicht zwingend eine optimale Lösung für die Anzahl von Maschinen in den Arbeitsstationen darstellen, aber von ihr könnten Startlösungen mit einer Distanz von drei erzeugt werden. Da von dem Wert drei keine neue gültige Lösung mit der Distanz d=3 gebildet werden kann, wird die Menge initialisierter Lösungen geringer gehalten. Die Generierung der Initiallösungen nach dem entwickelten Prinzip in Abschnitt 5.3.2 wird für zwei erste Iterationen demonstriert (siehe Abbildung 6-8). Als Startpartikel wird in diesem Beispiel eine Permutation von fünf gewählt.

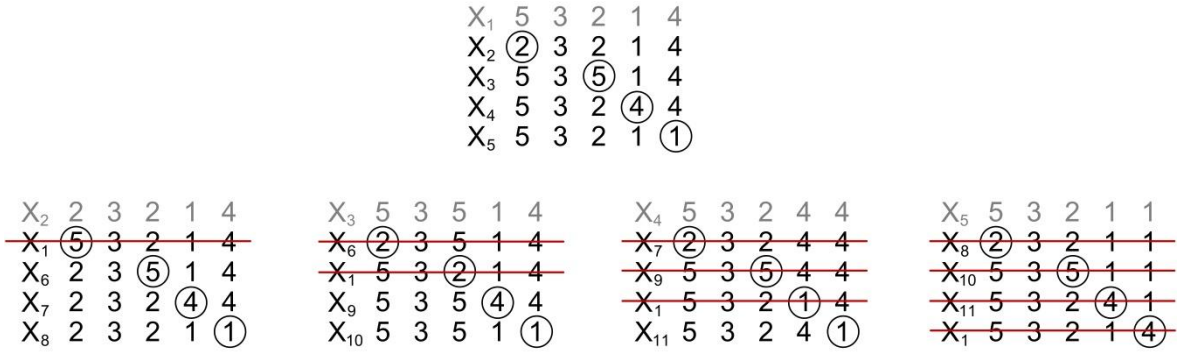

Abbildung 6-8: Sequentielle Diversifikation über zwei Iterationen

Abbildung 6-8 zeigt, dass mit einer Ausgangslösung X_1 in der ersten Iteration vier neue Lösungen mit einer Distanz von drei entstehen. In der zweiten Iteration fallen Redundanzen an, die nicht berücksichtigt werden sollen und daher in Rot durchgestrichen sind. Im Rahmen dieser Arbeit wurde experimentell ermittelt, dass vier Iterationen notwendig sind, um insgesamt 16 Startlösungen zu erzeugen, die jeweils mindestens eine Distanz von drei aufweisen. Die Anzahl gültiger Lösungen, die die vorgegebene Distanz einhalten, wird durch weitere Iterationen nicht mehr erhöht. Es bleiben 16 Startlösungen, die folglich für alle drei Schwärme bereitgestellt werden müssen.

6.6 Experimentelle Durchführung und Auswertung

In den vorangegangenen Abschnitten ist eine Spezifizierung der Entwicklungen aus Kapitel fünf in Bezug auf das exemplarische Produktionssystem vorgenommen worden. Mithilfe einer Anzahl von Simulationsdurchläufen wird das Konzept in den folgenden Abschnitten auf seine Funktionalität hin überprüft.

6.6.1 Relevante Kennzahlen und Parameter

Der erste Schritt vor der Durchführung ist die Festlegung geeigneter Kennzahlen, die als Bewertungskriterium für die Funktionsfähigkeit des Konzepts herangezogen werden können. Dazu zählt insbesondere der *Fitnesswert*, da dieser als Zielvorgabe minimiert werden muss. Zudem kennzeichnet sich eine effiziente Schwarmoptimierung durch eine geringe *Populationsgröße*, da diese mit einer insgesamt geringeren Simulationslaufzeit einhergeht. (siehe Abschnitt 4.2) Um die Ergebnisse übersichtlich zu halten, wird nicht der Fitnesswert aller Lösungen angegeben. Stattdessen wird für jede Iteration das Minimum und Maximum sowie der Durchschnittswert innerhalb der Population betrachtet. Des Weiteren werden für jedes Experiment die für den Initialisierungsprozess definierten Parameter *Startvektor* und *Distanz* angegeben. Der Startvektor zur Erzeugung einer Startpopulation wird dabei zufällig gewählt. (siehe Abschnitt 6.5) Für die Distanz wird zunächst eine Parametrisierung von d=3 angenommen. Der Suchprozess wird ab Iteration i=1 mit der entwickelten lokalen Explorationsstrategie (siehe Abschnitt 5.4) fortgesetzt und endet für einen besseren Vergleich der Ergebnisse immer bei i=10.

6.6.2 Vergleich der Ergebnisse eingesetzter Teilschwärme

Aufgrund der Lösungsraumzerlegung findet die Betrachtung der Gabelstapleranzahl mit der Wertemenge {1, 2, 3} zunächst getrennt mithilfe von Teilschwärmen statt. Der *Schwarm_w* gibt dabei die Anzahl *w* eingesetzter Gabelstapler an. (siehe Abschnitt 5.6) In Tabelle 6-1, Tabelle 6-2 und Tabelle 6-3 sind die Ergebnisse der eingesetzten Schwärme aufgeführt. Für i=0 wurde die Initialisierungspopulation evaluiert. Ausgehend von den besten drei Lösungen ist die lokale Nachbarschaftssuche fortgesetzt worden. Für jede Iteration sind Populationsgröße und Fitnesswerte angegeben. Der minimalste Fitnesswert, der im Verlauf der Suche gefunden wurde, ist grün markiert.

Die Ergebnisse von Schwarm_1 (siehe Tabelle 6-1) zeigen, dass bereits in Iteration i=2 ein Minimum zu verzeichnen ist. Die folgenden Iterationen erreichen keine besseren Fitnesswerte. Da die Ergebnisse auf eine Nachkommastelle gerundet sind, scheint der Fitnesswert sich von i=2 zu i=3 nicht zu ändern. Dies ist jedoch nicht der Fall, da sich der Fitnesswert in i=3 minimal schlechter ausfällt.

Tabelle 6-1: Ergebnisse Schwarm_1

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 3**				
	5	**3**	**2**	**1**	**4**	**1**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	16	22	20	17	16	12	13	14	17	16	15
Fitnesswert MAX	653,5	361,8	276,2	434,9	274,2	305,4	273,2	233,7	462,3	267,9	399,7
Fitnesswert ø	451,0	209,1	95,7	142,8	124,8	151,1	135,7	134,7	141,7	144,2	162,2
Fitnesswert MIN	157,4	22,7	22,2	22,2	26,0	38,6	23,0	27,7	27,4	58,1	31,6

Insgesamt liefert dieser Schwarm schlechtere Fitnesswerte als die Schwärme zwei und drei. In Schwarm_2 (siehe Tabelle 6-2) liegen ab i=2 alle Fitnesswerte unter den Werten, die in Schwarm_1 gefunden wurden. Ähnliches gilt für Schwarm_3. (siehe Tabelle 6-3) Werden die durchschnittlichen Fitnesswerte des zweiten und dritten Schwarms je Iteration miteinander verglichen, so stellt sich heraus, dass die Verläufe nahezu identisch sind. Da die Ergebnisse auch bezüglich der maximalen und minimalen Fitnesswerte keine signifikanten Unterschiede aufweisen, wird die Auswertung ausführlich für einen der beiden Schwärme vorgenommen.

Tabelle 6-2: Ergebnisse Schwarm_2

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 3**				
	5	**3**	**2**	**1**	**4**	**2**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	16	22	20	16	18	18	17	17	15	18	17
Fitnesswert MAX	628,5	317,4	333,3	255,9	294,5	283,1	288,4	300,1	240,1	181,2	315,4
Fitnesswert ø	417,2	137,2	62,4	46,3	74,9	95,7	117,5	100,6	109,5	72,0	130,8
Fitnesswert MIN	36,9	21,5	14,6	4,2	3,0	0,1	12,0	15,4	19,1	19,7	19,4

Tabelle 6-3: Ergebnisse Schwarm_3

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 3**				
	5	**3**	**2**	**1**	**4**	**3**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	16	22	20	15	16	15	16	18	15	16	12
Fitnesswert MAX	651,5	325,7	243,7	89,4	327,7	266,4	290,7	274,4	272,0	121,1	218,3
Fitnesswert ø	433,0	137,9	52,9	36,6	89,7	97,5	94,8	106,4	73,5	49,6	73,9
Fitnesswert MIN	40,6	29,1	20,6	21,5	7,3	6,6	8,6	8,3	2,3	0,8	9,5

6.6.3 Beobachtung und Auswertung des iterativen Suchprozesses

Da drei Gabelstapler keine besseren Fitnesswerte hervorrufen als bei der Verwendung von zwei Gabelstaplern, wird der Fokus auf Schwarm_2 gesetzt. Werden die Fitnesswerte aus Tabelle 6-2 für eine graphische Auswertung herangezogen, ergibt sich für die Entwicklung der Fitnesswerte über die Iterationen folgender Verlauf (siehe Abbildung 6-9).

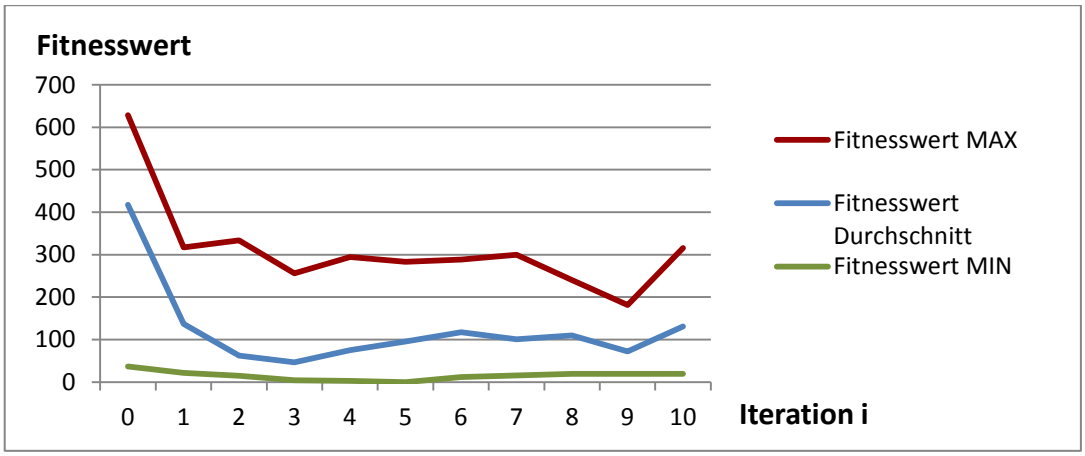

Abbildung 6-9: Graphische Auswertung der Fitnesswerte von Schwarm_2

Die Evaluation der Initialisierungspopulation zeigt in Iteration i=0, dass das Spektrum erzielter Fitnesswerte sehr breit gefächert ist. Die Partikel sind im Lösungsraum sehr weit voneinander entfernt und weisen entsprechend sehr unterschiedliche Fitnesswerte auf. Für einen Großteil dieser Werte fallen die Fitnesswerte schlecht aus. Dies ist daran zu erkennen, dass der durchschnittliche Fitnesswert in Iteration i=0 nicht mittig zwischen dem besten und dem schlechtesten Wert liegt, sondern insgesamt eher zum schlechtesten Wert tendiert. Dennoch sind bereits in Iteration i=0 gute Fitnesswerte vorhanden. Der grüne Graph zeigt bereits für Iteration i=0 einen kleinen Fitnesswert. Über die lokale Nachbarschaftssuche für die guten Werte verbessern sich die Fitnesswerte der Folgeiterationen enorm. Bereits in Iteration i=3 nimmt der durchschnittliche Fitnesswert der variierenden Population (blau) einen Wert an, der nur noch einem Achtel des durchschnittlichen Fitnesswerts in Iteration i=0 entspricht. Abgesehen von leichten Schwankungen nehmen die Graphen des maximalen Fitnesswerts (rot) und des durchschnittlichen Fitnesswerts (blau) über die weiteren Iterationen konstante Werte an. Zum Ende der Suche ist ein leichter Anstieg zu erkennen. Dieser lässt darauf schließen, dass nicht evaluierte Lösungen in der vielversprechenden Region knapp werden und einzelne Partikel neue Suchrichtungen anstreben.

Die Verteilung der Partikel im Lösungsraum wird besonders in den ersten Iterationen von den evaluierten Lösungen beeinflusst. Da mithilfe der lokalen Nachbarschaftssuche nur Lösungskandidaten infrage kommen, die sich bereits in der Umgebung einer guten Lösung befinden, ballen sich die Suchpartikel einer Population in den guten Regionen des Lösungsraums. Dabei wird mithilfe der *Tabuliste* eine mehrfache Evaluation einzelner Lö-

sungen verhindert. Jede neue Lösung steht in direkter Konkurrenz zu den bisherigen und wird bei vorteilhaften Eigenschaften zur Startlösung für die weitere Suche.

Der minimale Fitnesswert von 0,1 (siehe Tabelle 6-2) wird in Iteration i=5 gefunden. Um den Fokus auf die Entwicklung des minimalen Fitnesswerts zu legen, wird der Verlauf separat in Abbildung 6-10 veranschaulicht.

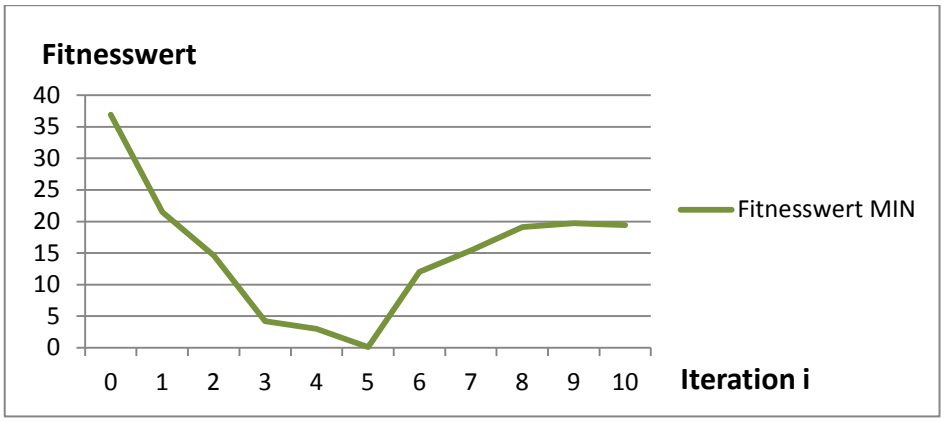

Abbildung 6-10: Fokus auf die Entwicklung des minimalen Fitnesswerts

In diesem Fall zeigt sich, dass zielstrebig auf ein Minimum zugearbeitet wurde. Im Verlauf dieses Experiments ist kein weiteres lokales Minimum des Fitnesswerts zu verzeichnen.

Die minimalen Fitnesswerte, die in diesem Experiment gefunden wurden, sind im Suchprozess einer Vektorkodierung zugeordnet. Um den Effekt der entwickelten lokalen Explorationsstrategie (siehe Abschnitt 5.4) zu veranschaulichen, werden die besten Kodierungen B je Iteration i in der Form B_i für den betrachteten Suchverlauf zusammengetragen (siehe Abbildung 6-11).

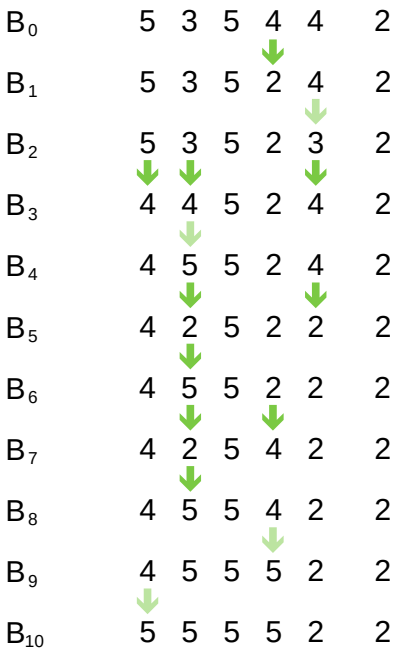

Abbildung 6-11: Effekt des multiagentenbasierten Ansatzes

Mit grünen Pfeilen sind alle Veränderungen von einer besten Lösung B_i zu der nächsten besten Lösung B_{i+1} visualisiert. Trotz einer lokalen Nachbarschaftssuche, bei der sich die Distanz zwischen einzelnen Lösungen lediglich um den Wert „1“ unterscheidet (wie bei den transparenten Pfeilen), wird für diese Liste an besten Lösungen deutlich, dass der überwiegende Teil der Veränderungen größere Distanzen aufweist oder sogar mehrere Veränderungen von einer Iteration zur nächsten vorgenommen werden (wie im Übergang von B_2 zu B_3). Diese explorierenden Eigenschaften entsprechen nicht annähernd dem eines Einzellösungsverfahrens. (siehe Abschnitt 3.3) Durch die Verwendung des multiagentenbasierten Ansatzes lassen sich diese Beobachtungen jedoch erklären. In jeder Iteration werden die besten drei Partikel zur Fortsetzung der Suche verwendet. Sobald sie in einer Iteration effektive Veränderungen erreicht haben, nehmen sie den Platz des besten Partikels ein. Infolgedessen konnte innerhalb weniger Iterationen ein akzeptabler Fitnesswert und damit eine gute Lösung für das zu variierende System gefunden werden.

6.7 Experimentelle Untersuchungen alternativer Eingabeparameter

Die Entwicklung lässt dem Anwender einige Freiheiten in Bezug auf die Parametrisierung, insbesondere der Distanz. Eine Alternative für den Initialisierungsprozess ist eine kleinere Distanz von d=2. Dieser Fall bildet einen Teil der experimentellen Untersuchung. Zudem wird überprüft, ob andere Startvektoren Einfluss auf das Ergebnis nehmen. Dazu werden mehrfache Experimente mit einer Distanz von d=3 für eine Gabelstapleranzahl von G=2 durchgeführt. Mit sehr unterschiedlichen Startvektoren soll gezeigt werden, dass das Prinzip allgemein gültig ist. Jeder Startvektor wird mit einer Permutation von fünf gebildet. (siehe Abschnitt 6.5) Da die „3“ im Rahmen der Initialisierung, aufgrund der Distanz von d=3, keiner Variation unterliegt, wird überprüft, ob die Position der „3“ im Startvektor Einfluss auf das Ergebnis der Suche nimmt.

6.7.1 Variation des Startvektors

Tabelle 6-2 zeigt bereits ein erstes Experiment. Dabei befindet sich die „3“ zufällig an zweiter Position. Für die folgenden Startvektoren wird die Position der „3“ alle weiteren Positionen einnehmen. Die Ergebnisse der Experimente werden wie in Abschnitt 6.6.2. in tabellarischer Form zusammengetragen. (siehe Tabelle 6-4, 6-5, 6-6 und 6-7) Die Variation des Startvektors zeigt zunächst, dass jede Permutation von fünf mit beliebiger Position der „3“ eine gute Lösung bezüglich des Fitnesswerts liefert. Abweichungen der Ergebnisse sind mit den stochastischen Einflüssen der zugrundeliegenden Simulation zu begründen. (siehe Abschnitt 2.4) Es entstehen minimale Fitnesswerte von 0,1 (siehe Tabelle 6-2) bis zu 0,9 (siehe Tabelle 6-6). Die Veränderungen der maximalen und durchschnittlichen Fitnesswerte sind in allen Fällen ähnlich zu dem Verlauf in Abbildung 6-9.

Tabelle 6-4: Ergebnisse erste Variation des Startvektors

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 3**				
	3	**2**	**4**	**5**	**1**	**2**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	16	28	23	18	17	19	19	17	15	17	13
Fitnesswert MAX	938,8	463,2	359,8	357,3	269,8	310,7	332,5	337,3	289,1	329,1	311,9
Fitnesswert ø	418,8	168,3	131,9	134,8	72,4	83,0	108,3	108,4	93,1	110,4	112,5
Fitnesswert MIN	95,4	26,9	11,3	13,7	0,8	5,9	2,4	2,2	12,5	17,0	16,9

Tabelle 6-5: Ergebnisse zweite Variation des Startvektors

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 3**				
	2	**1**	**3**	**5**	**4**	**2**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	16	24	17	14	16	15	12	11	15	15	21
Fitnesswert MAX	852,4	266,2	177,1	130,2	298,7	262,8	261,9	269,3	290,7	274,3	293,7
Fitnesswert ø	475,2	63,3	57,1	57,1	106,1	78,0	54,5	69,8	123,8	132,8	99,1
Fitnesswert MIN	3,7	5,8	1,4	8,5	1,5	0,6	5,5	19,3	10,0	2,2	4,9

Tabelle 6-6: Ergebnisse dritte Variation des Startvektors

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 3**				
	4	**5**	**1**	**3**	**2**	**2**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	16	26	22	19	16	18	19	17	11	14	15
Fitnesswert MAX	932,9	328,7	290,7	306,7	274,8	291,1	281,2	341,4	343,8	334,9	283,8
Fitnesswert ø	591,4	131,9	107,4	109,0	89,9	194,6	79,8	107,4	89,2	123,0	94,3
Fitnesswert MIN	20,2	19,5	6,3	4,2	7,9	4,3	6,0	0,9	2,7	9,2	5,7

Tabelle 6-7: Ergebnisse vierte Variation des Startvektors

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 3**				
	4	**1**	**5**	**2**	**3**	**2**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	16	24	17	14	16	15	12	11	15	15	21
Fitnesswert MAX	852,4	266,2	177,1	130,2	289,7	262,8	261,9	269,3	290,7	274,4	293,7
Fitnesswert ø	528,9	63,3	57,1	57,2	106,1	78,0	54,5	69,8	123,8	132,8	99,1
Fitnesswert MIN	37,0	5,8	1,4	8,5	1,5	0,6	5,5	19,3	10,0	2,2	4,9

Auffällig an den Versuchsreihen ist, dass die erste und dritte Variation (siehe Tabelle 6-4 und Tabelle 6-6) lediglich ein Minimum aufweisen. (wie im ersten Experiment Tabelle 6-2) Die beiden anderen Variationen weisen mehrfache Minima auf. Für das Experiment in Tabelle 6-7 wird der Verlauf des Minimums für den Fitnesswert zur Veranschaulichung graphisch dargestellt (siehe Abbildung 6-12).

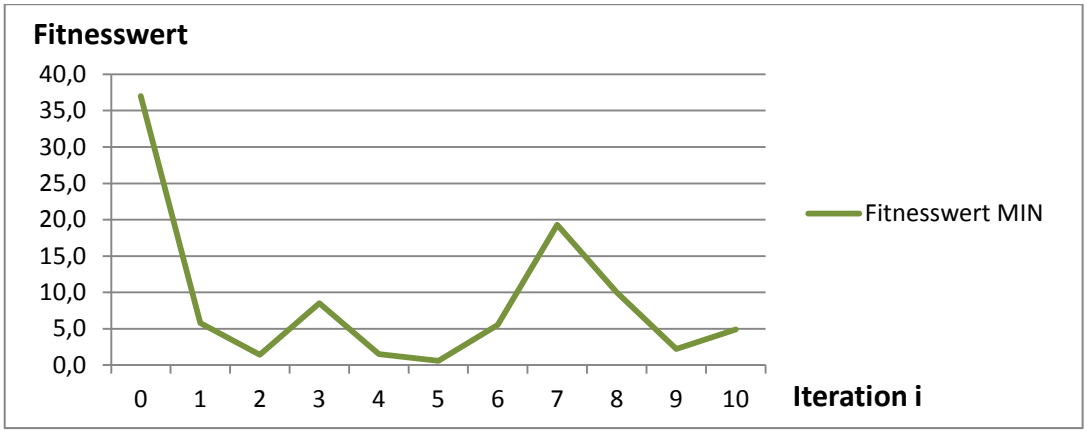

Abbildung 6-12: Auftreten mehrfacher Minima

Das Auftreten mehrfacher Minima lässt sich mit der Verwendung der Tabuliste begründen. Die Partikel befinden sich offenbar in vielversprechenden Regionen des Lösungsraums. Dabei verhindert die Tabuliste, dass eine Lösung mehrfach erreicht wird. Die Minima in Abbildung 6-12 sind daher verschiedenen Vektorkodierungen zugeordnet, die potentielle Alternativen zur besten Lösung des Suchprozesses bilden.

6.7.2 Alternative Distanz im Rahmen der Initialisierung

Eine weitere Untersuchung soll zeigen, welchen Einfluss die Parametrisierung der Distanz auf die Ergebnisse der Suche nimmt. Um einen besseren Vergleich vorzunehmen, wird eine Vektorkodierung verwendet, die bereits mit einer Distanz von d=3 ausgewertet wurde. Dazu wird die Vektorkodierung aus dem Experiment in Tabelle 6-2 erneut mit veränderter Distanz durchgeführt. Die Ergebnisse sind in tabellarischer Form zusammengetragen. (siehe Tabelle 6-8)

Tabelle 6-8: Experiment mit alternativer Distanz d=2

Startvektor:	AS1	AS2	AS3	AS4	AS5	G	Distanz **d = 2**				
	5	**3**	**2**	**1**	**4**	**2**					
Iteration i	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8	i = 9	i = 10
Populationsgröße	108	22	16	12	10	11	12	16	18	18	17
Fitnesswert MAX	871,5	304,9	267,3	259,3	281,0	283,9	306,9	319,5	295,4	268,2	291,7
Fitnesswert ø	446,4	142,8	54,5	46,8	71,5	129,1	98,7	101,8	92,0	106,7	97,1
Fitnesswert MIN	23,6	14,0	10,5	1,5	12,0	15,7	11,9	2,9	4,2	13,0	1,4

Die Fitnesswerte weisen Ähnlichkeiten zu den bereits ausgewerteten Experimenten auf. Es wurden insgesamt zwei Minima gefunden, deren Werte sich nicht signifikant von dem Minimum der Suche mit d=3 unterscheidet. Die Verwendung einer geringeren Distanz erweist sich somit nicht als nötig. Zudem muss die Populationsgröße als kritisches Bewertungskriterium betrachtet werden. Eine Distanz von d=2 führt zu einer Startpopulation, die 108 Partikel umfasst. Im Vergleich zu den 16 initialisierten Startpartikeln bei einer definierten Distanz von d=3 erweist sich diese Alternative als eine schlechtere Parametrisierung.

6.7.3 Interpretation der Experimente

Mit unterschiedlichen Startvektoren wurden Experimente durchgeführt, deren Ziel es war, eine beste Lösung für das zu variierende Produktionssystem zu finden. Im Folgenden werden die Startvektoren mit ihren zugehörigen minimalen Fitnesswerten und der besten Lösung im Simulationsdurchlauf zusammengetragen (siehe Tabelle 6-9).

Tabelle 6-9: Beste Lösung für das exemplarische Produktionssystem

Startvektor						Fitnesswert	beste Lösung					
AS1	AS2	AS3	AS4	AS5	G	MIN	AS1	AS2	AS3	AS4	AS5	G
5	3	2	1	4	2	0,1	4	2	5	2	2	2
3	2	4	5	1	2	0,8	4	3	5	2	5	2
2	1	3	5	4	2	0,6	4	1	5	2	5	2
4	5	1	3	2	2	0,9	4	5	5	2	5	2
4	1	5	2	3	2	0,6	4	1	5	2	5	2

Identische Werte der Vektorkodierung, die als beste Lösung identifiziert wurden, sind als grüne Ziffern erkenntlich. Das erste Experiment scheint eine Ausnahme gewesen zu sein, da sich insbesondere der Wert für die fünfte Arbeitsstation unterscheidet. Die Anzahl eingesetzter Maschinen in Arbeitsstation AS2 variieren in jedem Durchgang. Es sollte daher sicherheitshalber nicht der minimalste Wert verwendet werden. Für die Variation des exemplarischen Produktionssystems würde die Vektorkodierung (4 2 5 2 5 2) eine sichere Kombination für die Auftragslage ergeben, wie sie im Rahmen der Simulation verwendet wurde. Mithilfe der Kenntnisse über Engpässe und Termintreue (siehe Abschnitt 2.1) kann interpretiert werden, dass Arbeitsstation AS3 den Engpass des Produktionssystems bildet, da die volle Kapazität an Maschinen verwendet wird. Die erhöhte Anzahl an Maschinen in Arbeitsstation AS5 deutet auf eine Sicherstellung der Termintreue hin.

6.8 Abschließende Bewertung der Multiagenten-Tabu-Suche

Das entwickelte Konzept einer Multiagenten-Tabu-Suche beinhaltet alle wesentlichen Eigenschaften, die nötig sind, um als Schwarmoptimierung ein Gleichgewicht zwischen Exploration und Fokussierung einzunehmen. Die Wirkung der Tabu-Suche spiegelt sich auch in den Experimenten wieder. Denn im Rahmen der Experimente kam es zum Auftreten mehrfacher Minima innerhalb eines Simulationsdurchlaufs, das mit der Tabuliste begründet werden können. Die lokale Nachbarschaftssuche führt in den durchgeführten Experimenten in jedem Fall zu verbesserten Fitnesswerten. Die Tabu-Suche ist allerdings nicht der einzige Grund für das positive Resultat. Ein bedeutender Stellenwert kommt der Initialisierung zu. Diese nutzt mit der sequentiellen Diversifikation eine Strategie, die es für diskrete mehrdimensionale Lösungsräume möglich macht, den Raum weiträumig abzudecken. Die Evaluierung der Startpopulation entscheidet recht schnell, in welcher Region die Nachbarschaftssuche fokussieren muss. Für die exemplarische Umsetzung kann die Suche als erfolgreich bewertet werden. Auch das Konzept erweist sich somit als funktionsfähig.

Das Konzept ist in seiner Entwicklung (Kapitel 5) sehr allgemein gehalten, und stellt damit eine typische Metaheuristik dar. (siehe Abschnitt 2.4) Für die Spezifizierung ist je nach Lösungsraum eine entsprechende Parametrisierung vorzunehmen. Die Populationsgröße wird bei komplexeren Lösungsräumen aufgrund einer erhöhten Anzahl an Lösungskandidaten zunehmen. Für die Schwarmoptimierung ist es daher erforderlich, ausreichend Startlösungen zu initialisieren. Für die exemplarische Umsetzung hat sich gezeigt, dass dabei eine höhere Distanz effizienter sein kann als eine kleinere. (siehe Abschnitt 6.7.2)

Abschließend kann festgestellt werden, dass das Konzept sinnvolle Anwendungsschritte bietet. Zunächst werden die passenden Kodierungen festgelegt. Um die lokale Suche auf kleinere Teilräume eines komplexen Lösungsraums zu begrenzen, findet eine Lösungsraumzerlegung statt, die sich auch in der experimentellen Durchführung als funktionsfähig erweist.

7 Zusammenfassung und Ausblick

Das Ergebnis dieser Arbeit ist die Entwicklung einer Multiagenten-Tabu-Suche, die ein Gleichgewicht zwischen explorativer und fokussierender Suchstrategie mittels eines agentenbasierten Ansatzes findet. Das Konzept orientiert sich insbesondere an zu variierenden Produktionssystemen, die Maschinen oder Transportmittel unterschiedlichen Typs oder unterschiedlicher Menge verwenden. Da diese Produktionssysteme ereignisdiskrete Systeme bilden, wurde auch der Aspekt der Modellierung aufgegriffen und begründet, weshalb die Modellierung des betrachteten Produktionssystems auf THORNs basiert.

Als Vorbereitung für die Entwicklung in dieser Arbeit wurden zunächst klassifizierende Eigenschaften von Heuristiken betrachtet. Zudem wurden beispielhafte Verknüpfungsmöglichkeiten von Einzellösungsverfahren und populationsbasierten Lösungsverfahren aufgezeigt. Mithilfe der gewonnen Kenntnisse wurde die Multiagenten-Tabu-Suche in mehreren Schritten entwickelt. Insbesondere die Kodierung und die Initialisierung haben einen bedeutenden Stellenwert eingenommen, da sie die Voraussetzungen für eine erfolgreiche lokale Nachbarschaftssuche unter Verwendung der Tabuliste bilden.

Die Übertragung auf ein exemplarisches zu variierendes Produktionssystem zeigt, dass das Konzept funktioniert. Innerhalb weniger Iterationen wird eine Lösung gefunden, die eine gute Lösungsqualität aufweist. Die entwickelte hybride Metaheuristik kann somit erfolgreich zur Verkürzung der Simulationslaufzeit im Rahmen einer simulationsgestützten Optimierung zu variierender Produktionssysteme eingesetzt werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Bekrar, Abdelghani; Chaabane, Sondes; Trentesaux, Damien; Bornschlegell, Augusto; Pelle, Julien; Harmand, Souad (2011): Hybrid PSO-Tabu Search for Constrained Non-linear Optimization Problems. In: EISTI (Hrsg.): ICSI 2011 International conference on swarm intelligence. Cergy and France, 2011, abrufbar unter: http://icsi11.eisti.fr/papers/paper_22.pdf (28.01.2017).

Biethahn, Jörg; Lackner, Andreas; Range, Michael (2004): Optimierung und Simulation. München: Oldenbourg Wissenschaftsverlag GmbH, 2004.

Blum, Christian; Roli, Andrea (2003): Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys 35 (2003) 3, S. 268-308.

Bogon, Tjorben (2013): Agentenbasierte Schwarmintelligenz. Wiesbaden: Springer Fachmedien Wiesbaden, 2013.

Bracht, Uwe; Geckler, Dieter; Wenzel, Sigrid (2011): Digitale Fabrik – Methoden und Praxisbeispiele. Berlin, Heidelberg, Springer-Verlag, 2011.

Clerc, Maurice (2006): Particle Swarm Optimization. London: ISTE, 2006.

Cooren, Yann; Clerc, Maurice; Siarry, Patrick (2009): Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm. Swarm Intell (Swarm Intelligence) 2 (2009) 3, S. 149-178.

De la Cruz, Jair J.; Paternina-Arboleda, Carlos D.; Cantillo, Victor; Montoya-Torres, Jairo R. (2013): A two-pheromone trail ant colony system – tabu search approach for the heterogeneous vehicle routing problem. J Heuristics (Journal of Heuristics) 19 (2013) 2, S. 233-252.

Dietmaier, Christopher (2014): Mathematik für angewandte Wissenschaften. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

Du, Ke-Lin; Swamy, M. N. S. (2016): Search and Optimization by Metaheuristics – Techniques and Algorithms Inspired by Nature. Cham: Springer International Publishing, 2016.

Dyckhoff, Harald; Spengler, Thomas Stefan (2007): Produktionswirtschaft – Eine Einführung für Wirtschaftsingenieure, 2. Auflage. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2007.

Eiben, A. E.; Schippers C. A. (1998): On evolutionary exploration and exploitation. Fundamenta Informaticae 35 (1998) 1-16, S. 35-50, abrufbar unter: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.4885&rep=rep1&type=pdf (15.02.2017).

ElMaraghy, Hoda A. (Hrsg.) (2009): Changeable and Reconfigurable Manufacturing Systems. London: Springer, 2009.

Fritzsche, Albrecht (2009): Heuristische Suche in komplexen Strukturen – Zur Verwendung Genetischer Algorithmen bei der Auftragseinplanung in der Automobilindustrie. Wiesbaden: Gabler Verlag/ GWV Fachverlage GmbH, 2009.

Glover, Fred; Laguna, Manuel (1995): Tabu Search. In: Reeves, Colin R. (Hrsg.): Modern Heuristic Techniques for Combinatorial Problems. London: McGraw-Hill, 1995, S. 70-150.

Glover, Fred; Laguna, Manuel (1997): Tabu Search. Boston, Dordrecht, London: Kluwer Academic Publishers, 1997.

Klemmt, Andreas; Horn, Sven; Weigert, Gerald (2011): Simulationsgestützte Optimierung von Fertigungsprozessen in der Halbleiterindustrie. In: März, Lothar; Krug, Wilfried; Rose, Oliver; Weigert, Gerald (Hrsg.): Simulation und Optimierung in Produktion und Logistik – Praxisorientierter Leitfaden mit Fallbeispielen. Berlin, Heidelberg: Springer-Verlag, 2011, S. 49-63.

Larabi Marie-Sainte, Souad; Berro, Alain; Ruiz-Gazen, Anne (2010): An Efficient Optimization Method for Revealing Local Optima of Projection Pursuit Indices. In: Dorigo, Marco et al. (Hrsg.): Swarm Intelligence – 7th International Conference, ANTS 2010, Brussels, Belgium, September 8-10, 2010. Proceedings. Berlin, Heidelberg: Springer-Verlag, 2010, S. 60-71.

Law, Averill M. (2007): Simulation Modeling and Analysis, 4. Auflage. Boston: McGraw-Hill, 2007.

Lödding, Hermann (2010): Wandlungsfähige Produktionsplanung und -steuerung – Anforderungen aus schwankenden Auftragseingängen. In: Nyhuis, Peter (Hrsg.): Wandlungsfähige Produktionssysteme. Berlin: GITO-Verlag, 2010, S. 45-62.

Marett, Richard; Wright, Mike (1996): A comparison of neighborhood search techniques for multi-objective combinatorial problems. Computers & Operations Research 23 (1996) 5, 465-483.

März, Lothar; Krug, Wilfried; Rose, Oliver; Weigert, Gerald (Hrsg.) (2011): Simulation und Optimierung in Produktion und Logistik – Praxisorientierter Leitfaden mit Fallbeispielen. Berlin, Heidelberg: Springer-Verlag, 2011.

Nissen, Volker (1997): Einführung in Evolutionäre Algorithmen – Optimierung nach dem Vorbild der Evolution. Braunschweig: Vieweg, 1997.

Oster, Norbert (2007): Automatische Generierung optimaler struktureller Testdaten für objekt-orientierte Software mittels multi-objektiver Metaheuristiken. Erlangen: Friedrich-Alexander-Universität Erlangen-Nürnberg, Diss., 2007 (Arbeitsberichte des Instituts für Informatik Bd. 40, Nr. 2).

Puente León, Fernando; Kiencke, Uwe (2013): Ereignisdiskrete Systeme – Modellierung und Steuerung verteilter Systeme, 3. Auflage. München: Oldenbourg Verlag, 2013.

Reeves, Colin R. (1999): Landscapes, operators and heuristic search. Anneals of Operations Research 86 (1999), S. 473-490.

Reisig, Wolfgang (2010): Petrinetze – Modellierungstechnik, Analysemethoden, Fallstudien. Wiesbaden: Vieweg+Teubner Verlag/ Springer Fachmedien Wiesbaden GmbH, 2010.

Righini, G. (1993): Modular Petri nets for simulation of flexible production systems. International Journal of Production Research 31 (1993) 10, S. 2463-2477.

Schöf, Stefan (1997): Verteilte Simulation höherer Petrinetze. Oldenburg: Universität Oldenburg, Diss., 1997 (Berichte aus dem Fachbereich Informatik Nummer 2/97).

Talbi, El-Ghazali (2009): Metaheuristics – From Design to Implementation. Hoboken, New Jersey: Wiley, 2009.

Westkämper, Engelbert; Zahn, Erich (Hrsg.) (2009): Wandlungsfähige Produktionsunternehmen – Das Stuttgarter Unternehmensmodell. Berlin, Heidelberg: Springer-Verlag, 2009.

Wiendahl, H.-P.; ElMaraghy, H. A.; Nyhuis, P.; Zäh, M. F.; Wiendahl, H.-H.; Duffie, N.; Brieke, M. (2007): Changeable Manufacturing - Classification, Design and Operation. CIRP Annals - Manufacturing Technology 56 (2007) 2, S. 783-809.

Winz, Gerald (2012): Die dynamische Gestaltung der Produktions- und Logistikketten mittels prozessorientiertem Operating Curve Management – Logische Gesetzmäßigkeiten im schlanken Materialfluss. Berlin: Logos Verlag, 2012.

Zäpfel, Günther; Braune, Roland (2005): Moderne Heuristiken der Produktionsplanung – Am Beispiel der Maschinenbelegung. München: Verlag Granz Vahlen GmbH, 2005.