Genetische Algorithmen

in der

Losgrößenbestimmung

Bachelorarbeit

Vorgelegt bei:

Professor Dr. Andreas Größler

Betriebswirtschaftliches Institut der Universität Stuttgart, Abteilung X, Abteilung für Allgemeine Betriebswirtschaftslehre und Produktionswirtschaft

Von:

Dominik Larche
Abschlussziel: Bachelor of Science
Studienfach: Wirtschaftsinformatik

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Steigende Anforderungen bei dem Kunden sowie die daraus resultierende erhöhte Komplexität in den Produkten erfordern den Einsatz geeigneter Planungswerkzeuge in der Produktion. Darüber hinaus drängt der intensive Wettbewerb auf dem Markt die produzierenden Unternehmen zur Anwendung von kosteneinsparenden Methoden, um langfristig die eigene Existenz sicherzustellen.[1] Um die prognostizierte Nachfrage für einen bestimmten Zeithorizont zu bedienen und die dabei anfallenden Kosten gering zu halten, muss der Einsatz der zur Verfügung stehenden Ressourcen und Kapazitäten genau geplant werden.[2] In der Herstellung komplexer Endprodukte, d.h. Produkte die aus mehreren Bauteilen auf verschiedenen Produktionsebenen gefertigt werden, spielt die Materialbedarfsplanung eine besondere Rolle.[3] Die Bestimmung optimaler Losgrößen bildet eine Kernaktivität in der Materialbedarfsplanung und folglich in der gesamten Produktionsplanung.[4]

Die Losgrößenprobleme lassen sich in die Kategorien „1-Level Probleme“ und „n-Level Probleme“ jeweils mit und ohne Kapazitätsrestriktionen unterteilen. In 1-Level-Produktionen werden Rohmaterialien direkt zu Endprodukten verarbeitet, während in n-Level-Produktionen die Verarbeitung erst über mehrere Zwischenprodukte geschieht. Liegt in den betrachteten Sachverhalten eine Beschränkung in den Ressourcen vor, so ordnet man diese der Kategorie „Mit Kapazitätsrestriktionen“ zu, andernfalls der Kategorie „Ohne Kapazitätsrestriktionen“.[5]

Die Komplexitätstheorie sowie computergestützte Analysen zeigen auf, dass die meisten Probleme bezüglich der optimalen Losgröße nur schwer zu bewerkstelligen sind.[6] Zwar liegen einerseits entsprechende Optimierungsverfahren vor, andererseits sind die dahinterstehenden Algorithmen meist sehr komplex, sodass die Bewältigung solcher Fragestellungen entweder eine inakzeptabel hohe Laufzeit oder eine unerwünscht große Diskrepanz zum optimalen Ergebnis nach sich zieht.[7] Um nun Resultate nahe am Optimum zu gewährleisten und zugleich die hierfür benötigte Rechenzeit im Rahmen zu halten, wurden in der Produktionswirtschaft diverse Heuristiken entwickelt.[8] Genetische

[1] vgl. Dellaert u.a. (2000), S. 241
[2] vgl. Gelders und van Wassenhove (1981), S. 101
[3] vgl. Dellaert u.a. (2000), S. 241
[4] vgl. Goren u.a. (2010), S. 575; Prasad und Krishnaiah Chetty (2001), S. 520
[5] vgl. Goren u.a. (2010), S. 578f.
[6] vgl. ebenda, S. 576
[7] vgl. Dellaert u.a. (2000), S.242
[8] vgl. Goren u.a. (2010), S. 577

Algorithmen sind in der Lage mit verhältnismäßig geringem Rechenaufwand, (nahe-)optimale Lösungen auch für komplexe Fragestellungen zu finden, indem sie parallel eine Vielzahl an Unterräumen im gesamten Lösungsraum untersuchen.[9] Diese Arbeit befasst sich mit den für das beschriebene Problem entwickelten genetischen Algorithmen.

Im Folgenden vermittelt Kapitel 2 die für den Hauptteil erforderlichen theoretischen Grundlagen zu genetischen Algorithmen und Aspekten der Produktionsplanung, insbesondere der Losgrößenbestimmung. Kapitel 3 ist nach drei disjunkten Problemkategorien in aufsteigender Komplexität unterteilt und in jeder dieser Unterkapitel werden die für die Problemstellung vorliegenden Modelle und Algorithmen erläutert sowie ihre Leistungsfähigkeit thematisiert. Das Unterkapitel „Vergleich und Analyse“ befasst sich mit Mustern innerhalb und Unterschieden zwischen den drei Kategorien in Bezug auf ihre Verfahren und Ergebnisse. Kapitel 4 fasst die Arbeit sowie die darin gesammelten Erkenntnisse noch einmal zusammen und gibt einen kurzen Ausblick für das Thema.

2. Theoretische Grundlagen

In diesem Abschnitt gehen wir auf die theoretischen Grundlagen von genetischen Algorithmen, der Produktionsplanung und der Losgrößenbestimmung ein, bevor im Hauptteil die in der Literatur vorliegenden Modelle und Algorithmen zu den jeweiligen Problemkategorien erläutert werden.

2.1 Genetische Algorithmen

Genetische Algorithmen basieren auf dem Prinzip von Evolution und genetischer Vererbung. Genau wie die natürliche Auslese in der Natur das Überleben sowie die Reproduktion derjenigen Individuen sicherstellt, welche am besten an die äußeren Rahmenbedingungen angepasst sind, können genetische Algorithmen ihre Suche nach einer Lösung für ein Optimierungsproblem in einem komplexen Suchraum durch die Weiterverarbeitung und Verfeinerung vielversprechender sowie die Eliminierung unbrauchbarer Lösungsansätze effizienter gestalten.[10] Auf diese Weise untersuchen genetische

[9] vgl. Holland (1992), S. 68; Goren u.a. (2010), S. 578
[10] vgl. Srinivas und Patnaik (1994), S. 17

Algorithmen ein breites Spektrum an Lösungsansätzen und bewältigen Probleme, deren Struktur nicht vollständig durchdringbar ist.[11]

Die Überlebensfähigkeit eines Individuums hängt von seinen Eigenschaften ab. Diese Eigenschaften werden wiederum durch seine Gene bestimmt, die in ihrer Gesamtheit das Chromosom bilden.[12] Die Selektion, in welcher sich die besser angepassten Individuen und folglich die besser angepassten Gene durchsetzen, sowie die Reproduktion, welche die Vielfalt bezüglich des genetischen Materials in der Bevölkerung erhöht, sind die beiden Treiber der Evolution.[13] Computerprogramme, die die Prinzipien des Evolutionsprozesses nutzen, werden als genetische Algorithmen bezeichnet.[14]

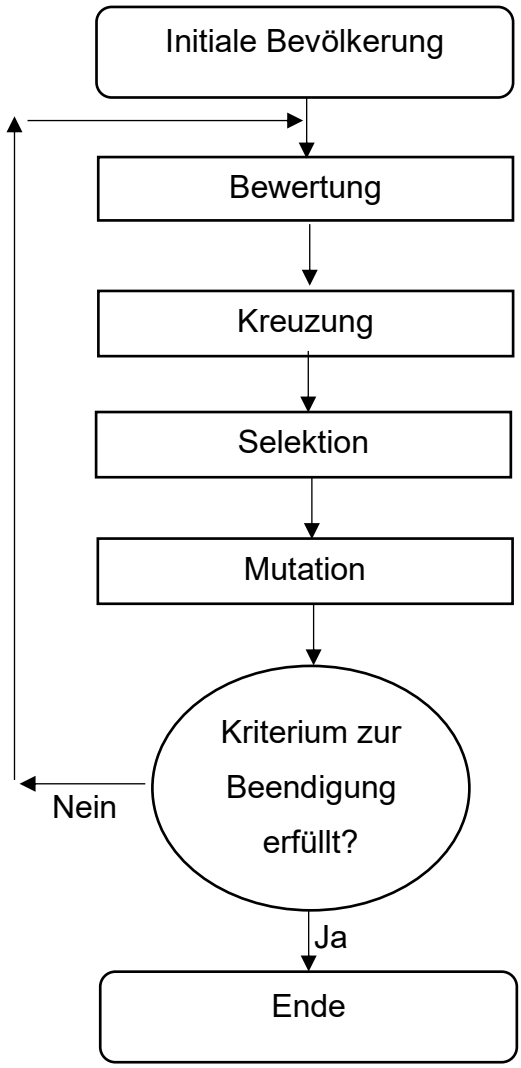
Um einen genetischen Algorithmus zu entwickeln, müssen das Kodierungsverfahren, die Fit-Funktion, das Selektionsverfahren, die eingesetzten genetischen Operatoren und die Kontrollparameter spezifiziert werden. Das Kodierungsverfahren übersetzt jede mögliche Lösung für das vorliegende Optimierungsproblem in eine Kette von Binärziffern (z.B. 0110101011).[15] Die entstehende Binärkette wird dabei als Chromosom bezeichnet.[16] Dies ist notwendig damit die einzelnen Lösungen auf die gleiche Weise verarbeitet werden können, wie es mit dem genetischen Code (DNS) in der Natur geschieht.[17]

Der genetische Algorithmus beginnt mit einer initialen Bevölkerung aus zufällig gewählten Binärketten (Chromosomen), welche die dahinterstehenden Lösungen (Parameterdeklarationen) repräsentieren.[18] Nun erfolgt eine Schleife, in der die Binärketten zuallererst Werte zugewiesen bekommen, die aus der Fit-Funktion resultieren. Die Fit-Funktion berechnet die Qualität einer Binärkette, meistens über den Wert den die dahinterstehende Lösung in der Zielfunktion erreicht, und übersetzt diesen ggf. in einen normierten Wertebereich von 0 bis 1.[19] Die Bewertung der Qualität einer Lösung kann beispielsweise anhand der zu erwartenden Profitabilitätssteigerung in der Betriebswirtschaft oder anhand beliebig anderer Kriterien erfolgen.[20] Abhängig von den zugeordneten Fit-Werten werden in der anschließenden Selektion die einzelnen Binärketten mit gewisser Wahrscheinlichkeit gar nicht, einmal oder beliebig oft reproduziert. Dabei führt ein höherer Fit-Wert zu einer tendenziell höheren Anzahl an Reproduktionen und damit zu einem

[11] vgl. Holland (1992), S. 66
[12] vgl. Srinivas und Patnaik (1994), S. 18
[13] vgl. Holland (1992), S. 66; Srinivas und Patnaik (1994), S. 18
[14] vgl. Srinivas und Patnaik (1994), S. 18
[15] vgl. ebenda, S. 19
[16] vgl. Whitley (1994), S. 65; Goren u.a. (2010), S. 578
[17] vgl. Holland (1992), S. 66
[18] vgl. Whitley (1994), S. 65
[19] vgl. Srinivas und Patnaik (1994), S. 19f.
[20] vgl. Holland (1992), S. 67

häufigeren Vorhandsein der jeweiligen Binärkette in der nachfolgenden Generation. Im Anschluss an die Selektion folgen die beiden genetischen Operatoren, nämlich die Kreuzung und die Mutation. In der Kreuzung werden jeweils zwei Binärketten zufällig gewählt und ihre Inhalte mit einer gewissen Wahrscheinlichkeit zu zwei neuen Binärketten kombiniert. Beispielsweise kann eine Kreuzung der Binärketten 101 und 010 zu den Kombinationen 110 und 001 o.ä. führen. Die neu entstandenen Binärketten werden in die Bevölkerung aufgenommen, die beiden Ursprünglichen entfernt.[21] Alternativ können die beiden ursprünglichen Binärketten auch behalten werden. In dieser Variante wird die Reproduktion über die genetischen Operatoren vollzogen und die Selektion findet zur Aussortierung erst nach Einsatz aller genetischer Operatoren statt.[22] Nach der Kreuzung werden diverse Stellen in einigen Binärketten zufällig gewählt und mit einer gewissen Wahrscheinlichkeit mutiert. Bei der Mutation ändert sich der Wert an der jeweiligen Stelle der betrachteten Binärkette von 0 in 1 oder von 1 in 0.[23] Beispielsweise kann eine Mutation in der Binärkette 101 die Binärkette 100 o.ä. ergeben.[24] Die Mutation als Operator stellt die Varietät innerhalb der Bevölkerung sicher und wirkt der Entstehung einer stark homogenen Bevölkerung mit nur noch geringen Weiterentwicklungspotenzialen entgegen.[25] Nun ist die nachfolgende Generation kreiert und die Schleife beginnt von Neuem, solange keine Bedingung zur Beendigung des Algorithmus erreicht wurde.[26] Abbildung 1 zeigt einen simplen genetischen Algorithmus und ein Beispiel für die Entstehung einer neuen Generation durch die Anwendung von Selektion, Kreuzung und Mutation auf die Binärketten in der Bevölkerung. Der Fit-Wert ergibt sich hierbei aus der relativen Häufigkeit der Einsen in der Binärkette. B1 ist die Bevölkerung in der Generation z und B4 ist die Bevölkerung in der Generation z+1.[27]

[21] vgl. Srinivas und Patnaik (1994), S. 19f.
[22] vgl. Holland (1992), S.68
[23] vgl. Srinivas und Patnaik (1994), S. 19
[24] vgl. ebenda, S. 20
[25] vgl. Holland (1992), S. 68
[26] vgl. Goren u.a. (2010), S. 578
[27] vgl. ebenda; Srinivas und Patnaik (1994), S. 20



B1:

Binärketten:	Fit-Wert:
00000111000	0.3
1000011111	0.6
0110101011	0.6
1111111011	0.9

B2: Nach Selektion

Binärketten:	Fit-Wert:
1000011111	0.6
0110101011	0.6
1111111011	0.9
1111111011	0.9

B3: Nach Kreuzung

Binärketten:	Fit-Wert:
1000011011	0.5
0110101011	0.6
1111111011	0.9
1111111111	1.0

B4: Nach Mutation

Binärketten:	Fit-Wert:
1000011011	0.5
0110111011	0.7
1111111011	0.9
0111111111	0.9

Abb. 1: Simple Darstellung eines genetischen Algorithmus und Beispiel eines Generationenübergangs[28]

Der Algorithmus endet je nach Definition, sobald eine Lösung mit akzeptabel hohem Fit-Wert erreicht wurde, nach einer bestimmten Zahl an kreierten Generationen oder sobald die Binärketten innerhalb der Bevölkerung sich kaum noch unterscheiden. Die Bevölkerungszahl, welche die zulässige und zugleich notwendige Anzahl an Binärketten für alle Generationen festlegt, die Kreuzungsrate, welche die Wahrscheinlichkeit zweier zufällig gewählter Binärketten zur Kreuzung sowie die Mutationsrate, welche die Mutationswahrscheinlichkeit der einzelnen Stellen in den Binärketten bestimmen, bilden die Kontrollparameter des genetischen Algorithmus.[29]

[28] Goren u.a. (2010), S. 578; Srinivas und Patnaik (1994), S. 20
[29] vgl. Srinivas und Patnaik (1994), S. 20

Insgesamt wird im genetischen Algorithmus die optimale Lösung dadurch erreicht, dass die Bestandteile der Binärketten, welche maßgeblich zu einem hohen Wert in der Zielfunktion beitragen, wiederholt miteinander kombiniert und letztlich so nebeneinander positioniert werden, dass die resultierende Binärkette die Zielfunktion maximiert.[30] In einem komplexen, mehrdimensionalen Lösungsraum existieren häufig zahlreiche lokale Maxima, in der sich gewöhnliche Suchalgorithmen, wie zum Beispiel „Hill-Climbing", leicht verharren und den Weg zum globalen Maximum nur schwer bestimmen können.[31] Der genetische Algorithmus hingegen untersucht zeitgleich mehrere Unterräume. Ein Unterraum kann als Muster verstanden werden, welches innerhalb einer Binärkette erscheinen kann. Beispielsweise bilden alle Binärketten, die mit der Zeichenfolge 10 beginnen (z.B. 10101, 10000, 10001, …), einen Unterraum. Durch ihre Vielfältigkeit können einzelne Binärketten einer Vielzahl an Unterräumen zugeordnet werden, sodass eine Bevölkerung von ein paar Tausend Binärketten bereits eine große Zahl an Unterräumen abdeckt.[32]

2.2 Produktionsplanung

In der Produktionsplanung wird die zu produzierende Menge eines Produkts für mehrere Zeitperioden (z.B. Monate) in einem festgelegten Zeitintervall bestimmt. Das betroffene Zeitintervall wird als Planungshorizont bezeichnet. Darüber hinaus legt die Produktionsplanung die zu erwartenden Lagerbestände und die zur Erfüllung der Produktionspläne benötigten Ressourcen (z.B. Material-/Personalbedarf) fest.[33] Die Art, nach der die Produktionsplanung vollzogen wird, hängt unmittelbar von dem Herstellungsverfahren ab. Besteht eine Fertigung nach Maß, also nach den individuellen Wünschen eines Kunden, so wird die hierfür herangezogene Produktionsplanung nur stoßweise (nach Auftrag des Kunden) durchgeführt. Handelt es sich hingegen um ein standardisiertes (Massen-)Produkt, das auf Lager gefertigt wird, dann muss die Produktionsplanung über mehrere Zeitperioden hinweg, wiederholt durchgeführt werden.[34] Die unterschiedlichen Aggregationsstufen in der Produktionsplanung bilden eine hierarchische Struktur.[35] Die hierarchische Planung dient der Entscheidungsunterstützung in komplexen Produktionslandschaften.[36] Die Planung auf einer Stufe beeinflusst Möglichkeiten sowie Einschränkungen auf der

[30] vgl. ebenda, S. 21
[31] vgl. Holland (1992), S. 67
[32] vgl. ebenda, S. 68
[33] vgl. Thomas und McClain (1993), S. 333
[34] vgl. Gelders und van Wassenhove (1981), S. 101
[35] vgl. ebenda, S. 102
[36] vgl. Meal u.a. (1987), S. 947

darunterliegenden Stufe.[37] Genauso hängen die Elemente auf der übergeordneten Stufe (z.B. Endprodukte) von den Bestimmungen bzgl. der untergeordneten Stufe (z.B. Bauteile) ab. Insbesondere schränken definierte Rohstoffmengen und Kapazitäten die mögliche Produktionsmenge eines bestimmten Endproduktes ein.[38]

Die unterschiedlichen Aggregationsstufen werden vom MRP-II erfasst. MRP-II ist ein Framework für die Produktionsplanung, das den Planungsprozess in vier Einheiten unterteilt.[39] Zu Beginn erfolgt eine aggregierte Planung, in der die mittelfristigen Kapazitäten berechnet werden.[40] Da die Mengen mancher Ressourcen (z.B. Personal) zu bestimmten Kosten änderbar sind (z.B. durch Einstellungen und Entlassungen), wird in der aggregierten Planung auch eine Anpassung der Ressourcenmengen vorgenommen.[41] Die Zusammenfassung der einzelnen Produkte in der aggregierten Planung reduziert die Streuung und folglich den Fehler in den prognostizierten Nachfragewerten, wodurch Überbestände sowie Engpässe in den Ressourcen bei der künftigen Produktion vermindert werden. Basierend auf der aggregierten Planung erfolgt die Bestimmung der zu produzierenden Mengen der einzelnen Endprodukte für den vorliegenden Planungshorizont in einem Produktionsprogramm.[42] Auf der Grundlage des Produktionsprogramms werden in der Materialbedarfsplanung die zeitversetzten Bedarfe für die einzelnen Komponenten und Rohmaterialien entwickelt.[43] Aus den Produktionsplänen der Endprodukte lässt sich eine zeitversetzte Bedarfsplanung für die jeweiligen Komponenten ableiten. Abhängigkeiten zwischen Komponenten in der Planung, vorliegende Kapazitätsbeschränkungen sowie Unsicherheiten in der Nachfrage erschweren die Materialbedarfsplanung, welche das Ziel einer optimalen Abfolge von Produktion und Lagerhaltung verfolgt.[44] Die Herausforderung liegt in der Bestimmung adäquater Losgrößen für die einzelnen Komponenten, sodass einerseits die Erfordernisse aus der Produktionsprogrammplanung bedient und andererseits die entstehenden Produktionskosten gering gehalten werden.[45] Nach der aggregierten Planung, der Produktionsprogrammplanung und der Materialbedarfsplanung folgt im MRP-II die Ablaufplanung, in der auf der niedrigsten Stufe der Planungshierarchie die Fertigungsaufträge auf die verschiedenen Arbeitsstationen verteilt und die zeitliche Reihenfolge ihrer Bearbeitung festgelegt wird. Eine gute Ablaufplanung sorgt dafür, dass

[37] vgl. Gelders und van Wassenhove (1981), S. 102
[38] vgl. Thomas und McClain (1993), S. 333
[39] vgl. ebenda, S. 353
[40] vgl. Thonemann und Albers (2011), S. 304
[41] vgl. Thomas und McClain (1993), S. 333
[42] vgl. Gelders und van Wassenhove (1981), S. 102f.
[43] vgl. Meal u.a. (1987), S. 947
[44] vgl. Gelders und van Wassenhove (1981), S. 103
[45] vgl. ebenda, S. 104

zur Verfügung stehende Produktionskapazitäten wirtschaftlich sinnvoll genutzt und die vorgegebenen Lieferfristen eingehalten werden.[46]

2.3 Losgrößenbestimmung

Die Produktionsmengen der einzelnen Produkte und Bauteile werden in der Losgrößenbestimmung zu Produktionslosen zusammengefasst.“[47] Die Losgrößenbestimmung beinhaltet die zeitliche und mengenmäßige Abstimmung der Produktionslose für alle Komponenten aller Produkte. Im einfachsten Fall wird die Summe aus Rüst- und Lagerhaltungskosten über alle betrachteten Perioden hinweg unter Beachtung weniger Nebenbedingungen minimiert.[48] Die zugehörige Zielfunktion sieht dabei aus wie folgt:

$$min \sum_{Periode\ t=1}^{Letzte\ Periode} (Rüstkosten_t * WirdProduziert_t + LagerhaltungskostenJeStück_t * Lagermenge_t + Fertigungsmenge_t * Stückkosten_t) \quad (1)$$

Die Lagermenge am Ende der Periode *t* berechnet sich wie folgt:

$$Lagermenge_t = Lagermenge_{t-1} + Fertigungsmenge_t - Bedarf_t \quad (2)$$

Die Nebenbedingungen zur Zielfunktion (1) bestehen zum einen in der Gleichung für die Lagermenge (2). Des Weiteren muss die Lager- sowie die Fertigungsmenge in (1) und (2) für jede Periode größer/gleich 0 sein. $WirdProduziert_t$ in (1) nimmt den Wert 1 an, wenn in der Periode *t* produziert wird und 0, wenn nicht.[49]

Für ein solches Problem existiert ein Algorithmus von Wagner und Whitin, das mit dynamischer Programmierung ein optimales Ergebnis liefert. Ein alternatives Verfahren bietet die Silver-Meal-Heuristik, welche die Losgröße für eine Periode, in welcher der Bedarf ohne Produktion nicht erfüllt werden kann, derart festlegt, dass der Bedarf für diese und die nachfolgenden k Perioden gedeckt wird. Die Variable k wird so gewählt, dass die durchschnittlichen Lagerhaltungskosten für diese und die nachfolgenden k Perioden minimiert werden. Dieser Ansatz wird in der Praxis häufig gewählt, da sie mit relativ geringem Aufwand der optimalen Lösung ziemlich nah kommt.[50]

[46] vgl. ebenda, S. 107
[47] vgl. Thonemann und Albers (2011), S. 306
[48] vgl. Gelders und van Wassenhove (1981), S. 105
[49] vgl. Goren u.a. (2010), S. 577
[50] vgl. Gaafar (2006), S. 434

Allerdings erhöht sich die Komplexität des Problems mit zunehmender Anzahl an Produktionsebenen (n-Level) sowie mit der Berücksichtigung von Einschränkungen in den Kapazitäten, wenn die für die Produktion benötigten Ressourcen nicht in beliebiger Menge zur Verfügung stehen.[51] Bei n-Level-Problemen erschweren Abhängigkeiten zwischen den Rohmaterialien und Komponenten auf über- und untergeordneter Produktionsebene die Berechnung optimaler Losgrößen.[52] Im Fall einer beschränkten Kapazität muss die Produktion bestimmter Produkte aus dem Produktionsprogramm zeitlich verschoben oder sogar ausgelassen werden, auch wenn dies strategisch nicht sinnvoll ist, weil die dahinterstehenden Produktionskapazitäten nicht für alle Lose ausreichend sind. Die Losgrößenbestimmung ist eng verbunden mit der nachfolgenden Ablaufplanung, da Nachlässigkeiten in diesem Bereich zu unvorhergesehen verlängerten Lieferzeiten der Komponenten führen, wodurch der gesamte Produktionsplan gefährdet wird.[53]

3. Genetische Algorithmen für die Losgrößenoptimierung

In diesem Abschnitt wird die Funktionsweise von verschiedenen genetischen Algorithmen für die entsprechenden Kategorien von Losgrößenproblemen erläutert und die Ergebnisse beschrieben, die diese Modelle in den jeweiligen Experimenten erzielen.

3.1 1-Level Probleme ohne Kapazitätsrestriktionen

Der erste Algorithmus ist für die Produktion eines einzigen Endprodukts ohne Kapazitätseinschränkungen. Bei dem betrachteten Endprodukt liegen keine Komponenten vor, für welche die Bedarfe abhängig von der Nachfrage am Endprodukt zu bestimmen sind. Entsprechend sieht die Zielfunktion wie folgt aus:

$$min \sum_{Periode\ t=1}^{Letzte\ Periode} Rüstkosten * WirdProduziert_t + LagerhaltungskostenJeStück * Lagermenge_t \quad (3)$$

Im Gegensatz zur Zielfunktion in (1) werden hierbei die Transaktionskosten und nicht die gesamten Produktionskosten berechnet. Des Weiteren hängen in (3) die Rüstkosten und

[51] vgl. Gelders und van Wassenhove (1981), S. 105
[52] vgl. Goren u.a. (2010), S. 577
[53] vgl. Gelders und van Wassenhove (1981), S. 107

die Lagerhaltungskosten je Stück nicht von der jeweiligen Periode ab und es gilt zusätzlich zu den Nebenbedingungen der Zielfunktion in (1), dass das initiale Lager leer ist.[54]

Der hierfür entwickelte genetische Algorithmus kodiert jede Lösung (Produktionsplan) für dieses Problem in eine einzelne Binärkette, dem Chromosom. Jede Stelle in der Binärkette repräsentiert eine Periode im Planungshorizont. Die erste Stelle steht folglich für die erste Periode, die n-te Stelle für die n-te Periode, wobei der Planungshorizont n Perioden umfasst. In der Binärkette wird an der Stelle *t* eine 1 verbucht, wenn in der Periode *t* produziert und eine 0, wenn nicht produziert wird.[55] Die genaue Losgröße ergibt sich aus der Summe der Bedarfe von der betrachteten Periode und allen folgenden Perioden, in denen nicht produziert wird.[56]

Um nun eine initiale Bevölkerung zu kreieren werden abhängig vom Kontrollparameter s, das die Anzahl an Chromosomen in der Bevölkerung bestimmt, zufällige Binärketten erstellt, bei denen an jeder Stelle mit einer 50-prozentigen Wahrscheinlichkeit eine 1 oder eine 0 gesetzt wird. Die Fit-Funktion berechnet den Fit-Wert einer Binärkette, indem sie den dahinterstehenden Produktionsplan in die Zielfunktion aus (3) einsetzt und mit den aus (3) resultierenden Gesamt-Transaktionskosten GTK die Division 1/GTK vollzieht. Auf diese Weise wird jeder Binärkette ein adäquater Fit-Wert aus dem normierten Bereich von 0 bis 1 zugewiesen. Der Fit-Wert ist höher, je geringer die Gesamt-Transaktionskosten, die aus dem Produktionsplan hinter der jeweiligen Binärkette resultieren. Im Anschluss daran folgt die Reproduktion, in der jede Binärketten mit einer Wahrscheinlichkeit r durch den Einsatz der Kreuzung als genetischer Operator vermehrt und rekombiniert wird. Die Wahrscheinlichkeit r ergibt sich durch den Quotienten aus dem Fit-Wert der betrachteten Binärkette und der Summe aller Fit-Werte in der Bevölkerung. Das bedeutet insbesondere, dass die Wahrscheinlichkeit zur Reproduktion bei niedrigeren (eigens verursachten) Gesamt-Transaktionskosten steigt und bei niedrigeren (gesamtheitlich in der Bevölkerung verursachten) Gesamt-Transaktionskosten sinkt. Zur Kreuzung werden jeweils zwei (zur Reproduktion gewählte) Binärketten zufällig herangezogen und ihre Informationen in zwei neue Binärketten (den Nachfahren) hineinkopiert. Danach werden die Informationen, ob in einer jeweiligen Periode produziert wird oder nicht, von einer zufällig gewählten Periode t beginnend (t liegt im Intervall von 2 bis n) bis zur letzten Periode n in den beiden Nachfahren getauscht.[57] Im Anschluss an die Kreuzung kommt der zweite

[54] vgl. Hernandez und Suer (1999), S. 2280f.
[55] vgl. ebenda
[56] vgl. van Hop und Tabucanon (2005), S. 132
[57] vgl. Hernandez und Suer (1999), S. 2281f.

genetische Operator, die Mutation, zum Einsatz. Dabei wird für jede Periode in allen Nachfahren mit einer sehr geringen Wahrscheinlichkeit entweder der Marker für das Produzieren entfernt oder falls nicht vorhanden, gesetzt. Letztlich folgt die Selektion, welche die Bevölkerungszahl s in der nachfolgenden Generation erhält, indem sie die ursprünglichen Binärketten sowie die Nachfahren aussortiert, die einen unbrauchbaren Produktionsplan repräsentieren und folglich einen geringen Fit-Wert aufweisen. Jede Binärkette *bk* wird mit einer Wahrscheinlichkeit *d* (zwischen 0 und 1) aussortiert, die sich folgendermaßen berechnen lässt:[58]

$$d = 1 / (\sum_{b \in Bevölkerung} (Fit(bk) / Fit(b))) \quad (4)$$

Ein zweiter genetischer Algorithmus für 1-Level Produktionen mit unbeschränkter Kapazität löst das gleiche mathematische Modell wie der Erste mit dem einzigen Unterschied, dass sich die Lagerhaltungskosten je Stück durch das Produkt aus Stückkosten und Lagerhaltungskostensatz (bzw. Opportunitätskostensatz) ergeben. Hierbei wird dieselbe Kodierung verwendet, wie im ersten Algorithmus. Die Fit-Funktion hingegen berechnet zwar ebenfalls die Gesamt-Transaktionskosten, allerdings ohne Normierung.[59] Die genetischen Operatoren bestehen in der Reproduktion, der Mutation und der Kreuzung. Die Mutation sowie die Kreuzung erfolgen auf die gleiche Weise wie im ersten Algorithmus, die Reproduktion meint hier die simple Duplizierung einer bestehenden Binärkette mit allen enthaltenen Informationen.[60]

Der Algorithmus beginnt mit einer initialen Bevölkerung von s = 1000 zufällig kreierten Binärketten. Alle Binärketten werden bewertet, um den höchsten Fit-Wert in der Bevölkerung zu bestimmen. Hinzu werden die Kreuzungs-, die Mutations- und die Reproduktionsrate initial festgelegt. Nun beginnt die Schleife, die erst nach Erreichen einer Bedingung zur Beendigung des Algorithmus hält und den besten Produktionsplan in der neusten Generation zurückgibt. Am Anfang der Schleife werden alle Binärketten abhängig von ihrem Fit-Wert gar nicht, einmal oder beliebig oft gewählt und gemäß den Kontrollparametern (Kreuzungs-, Mutations-, Reproduktionsrate) werden die genetischen Operatoren entsprechend oft auf sie angewendet, um die neue Bevölkerung zu erstellen und die ursprünglichen Binärketten werden entfernt. Das bedeutet insbesondere, dass die Kreuzungs-/Mutations-/Reproduktionsrate angibt, welcher Anteil an Binärketten in der neuen

[58] vgl. ebenda, S. 2282
[59] vgl. van Hop und Tabucanon (2005), S. 131f.
[60] vgl. ebenda, S. 133f.

Bevölkerung durch Kreuzung/Mutation/Reproduktion entstanden ist. Die drei Raten summieren sich dementsprechend auf 1. Im Anschluss daran wird für jede Binärkette in der neuen Bevölkerung der Fit-Wert berechnet. Nun gilt es festzustellen, welcher Anteil an der mit dem jeweiligen genetischen Operator (z.B. Kreuzung) erstellten Teilbevölkerung einen besseren Fit-Wert erzielt, als der beste Fit-Wert in der ursprünglichen Gesamtbevölkerung. Dieser Anteil bestimmt den neuen Wert des entsprechenden Kontrollparameters (z.B. den Wert der Kreuzungsrate) für die nachfolgenden Schleifendurchgänge. Damit sich die drei Raten wieder auf 1 summieren, wird jede dieser durch die Summe aller drei Raten geteilt. Am Ende wird der höchste Fit-Wert in der neuen Bevölkerung berechnet und die Schleife beginnt von Neuem.[61]

Der wesentliche Unterschied zum ersten Algorithmus besteht darin, dass die Kontrollparameter (die Kreuzungs-, die Mutations- und die Reproduktionsrate) in jedem Schleifendurchgang, also von Generation zu Generation, anhand der Qualität der durch die dahinterstehenden Operatoren erzeugten Nachfahren, angepasst werden.[62]

Für den dritten Algorithmus wird ein ähnliches mathematisches Modell wie zuvor betrachtet. Der Unterschied in der Problemstellung besteht nun darin, dass Lieferrückstände erlaubt sind und mit einer Strafzahlung je Stück je Periode in den Gesamt-Transaktionskosten verrechnet werden. Des Weiteren wird Batch-Kommissionierung angewendet, sodass die Losgrößen ein Vielfaches einer ganzzahligen minimalen Produktionsmenge betragen müssen.[63]

Aus diesem Grund kann an jeder Stelle in der Binärkette eine 0, eine 1 oder eine 2 stehen. Eine 0 repräsentiert keine Produktion, eine 1 oder 2 steht für eine Losgröße, die den Bedarf für die nächsten k Perioden ohne Produktion deckt. Bei einer 1 beträgt die Losgröße das gegenüber der benötigten Produktionsmenge nächstkleinere Vielfache (dabei fallen Strafzahlungen für Lieferrückstände an), bei einer 2 das nächstgrößere Vielfache der minimalen Produktionsmenge. Der Fit-Wert wird hierbei erneut auf Basis der verursachten Transaktionskosten berechnet. Die initiale Bevölkerung enthält s = 200 zufällige Binärketten. Die nachfolgende Generation wird durch den Einsatz verschiedener genetischer Operatoren erzeugt, wobei die Binärketten nach dem „Roulette Wheel“-Schema ausgewählt werden.[64] Das bedeutet, dass jeder Binärkette durch den Quotienten aus

[61] vgl. ebenda
[62] vgl. ebenda
[63] vgl. Gaafar (2006), S. 433
[64] vgl. ebenda, S. 435f.

dem eigenen Fit-Wert und der Summe aller Fit-Werte in der Bevölkerung und der Multiplikation dieses Quotienten mit 2π, ein entsprechender Sektor mit dem berechneten Winkel im sogenannten „Roulette Wheel“ zugeordnet wird. Das „Roulette Wheel“ wird „gedreht“, und die dem erwählten Feld zugeordnete Binärkette wird entsprechend gekreuzt oder mutiert.[65] Dieser Vorgang wird solange wiederholt, bis die vorher festgelegte Anzahl an Kreuzungen und Mutationen durgeführt sind. Um sicherzustellen, dass qualitativ hochwertige Lösungen nicht verloren gehen, werden die Binärketten mit den besten Fit-Werten (quasi als „Elite“) direkt in die nachfolgende Generation aufgenommen und ergeben zusammen mit den erzeugten Nachfahren wieder eine Bevölkerung von s = 200 Binärketten. Der Algorithmus endet nachdem 100 Generationen kreiert worden sind.[66]

Aus den vollzogenen Experimenten geht hervor, dass der erste Algorithmus nach im Schnitt etwa 300 erstellten Generationen das optimale Ergebnis erreicht. Die Wahl der Bevölkerungszahl und der Reproduktionsrate spielt dabei zwar keine große Rolle, allerdings kann nach 300 Generationen ein tendenziell besseres Ergebnis erwartet werden, je stärker die Fit-Funktion dazu beiträgt, dass diejenigen Binärketten, die bessere Produktionspläne repräsentieren, sowohl in der Reproduktion als auch in der Selektion „bevorzugt behandelt“ werden.[67] Bei einer Veränderung in den Rüst- bzw. in den Lagerhaltungskosten, passen sich die Ergebnisse aus dem genetischen Algorithmus an und liefern entsprechend eine tendenziell höhere bzw. niedrigere Anzahl an Perioden im Planungshorizont, an denen produziert werden soll. Auch bei variierenden Bedarfswerten über den Planungshorizont hinweg kommen Ergebnisse zustande, die entweder dem Optimum entsprechen oder nah am Optimum liegen.[68]

Mit dem zweiten Algorithmus kann in den hierzu beschriebenen Experimenten ein (nahe) optimaler Produktionsplan bereits nach weniger als 100 Generationen erreicht werden, wodurch eine deutliche Verbesserung in der Laufzeit gegenüber dem ersten Algorithmus vorliegt, bei dem ein solches Ergebnis erst nach (im Schnitt) ca. 300 Generationen erzielt werden kann. Die stetige Anpassung der Kontrollparameter im genetischen Algorithmus führt demnach zu einer Steigerung in der Performance.[69]

Die Experimente zu dem dritten Algorithmus sollen den Einfluss fünf unterschiedlicher Faktoren (Bedarfsmuster, minimale Produktionsmenge, Verhältnis Rüst-/Lagerkosten,

[65] vgl. Srinivas und Patnaik (1994), S. 19
[66] vgl. Gaafar (2006), S. 436f.
[67] vgl. Hernandez und Suer (1999), S. 2283ff.
[68] vgl. ebenda, S. 2285f.
[69] vgl. van Hop und Tabucanon (2005), S. 135

Verhältnis Lager-/Strafkosten, Länge des Planungshorizonts) auf die Qualität der Ergebnisse des beschriebenen genetischen Algorithmus sowie der Silver-Meal-Heuristik untersuchen und vergleichen. Diese zeigen auf, dass der genetische Algorithmus höherwertigere Ergebnisse liefert, als die Silver-Meal-Heuristik.[70] Lediglich eine signifikante Verlängerung des Planungshorizonts führt zu einer nennenswerten Abweichung vom optimalen Ergebnis im genetischen Algorithmus. Dieser Effekt tritt auch in der Silver-Meal-Heuristik auf, weiterhin entstehen hierbei erhebliche Verschlechterungen in den Ergebnissen durch eine signifikante Erhöhung des Verhältnisses zwischen Rüst- und Lagerkosten sowie den Übergang von konstanten in saisonalen Bedarf. Der starke Abfall in der Qualität durch den Einfluss von dem Verhältnis von Rüst-/Lagerkosten auf die Ergebnisse der Silver-Meal-Heuristik lässt sich auf das Verfahren zurückführen. Dabei wird k so gewählt, dass die durchschnittlichen Lagerkosten für die nächsten k Perioden minimiert werden und folglich in der Periode k+1 erneut Rüstkosten anfallen. Wenn die Rüstkosten nun erheblich höher als die Lagerkosten sind (z.B. Faktor 800), wird nach der Silver-Meal-Heuristik in der Periode k+1 produziert, auch wenn die Rüstkosten die Gesamt-Transaktionskosten ins Unendliche laufen lassen. Abbildung 2 verdeutlicht diesen Zusammenhang. Die gestrichelte Linie repräsentiert den genetischen Algorithmus, die durchgezogene Linie die Silver-Meal Heuristik. Auf der Abszisse ist das Verhältnis zwischen Rüstkosten und Lagerkosten je Einheit abzulesen, auf der Ordinate die prozentuale Abweichung der Ergebnisse vom Optimum.[71]

[70] vgl. Gaafar (2006), S. 437ff.
[71] vgl. ebenda, S.439f.

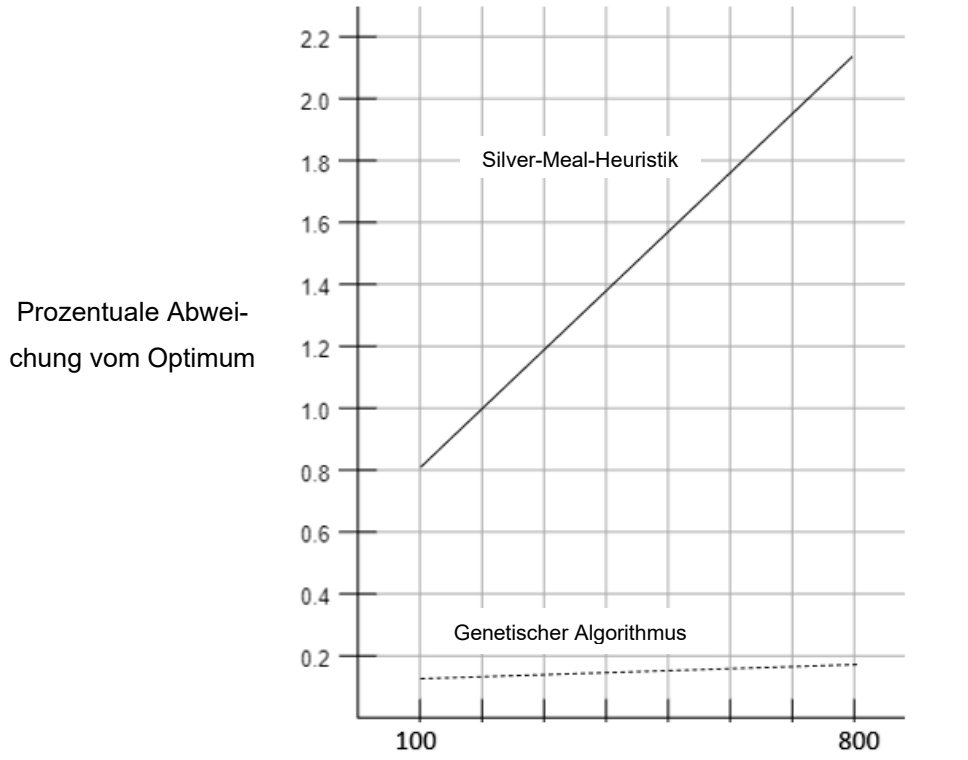


Abb. 2: Einfluss vom Verhältnis Rüst-/Lagerkosten auf die Qualität der Ergebnisse[72]

Bezüglich der minimalen Produktionsmenge sowie dem Verhältnis von Lager-/Strafkosten können keine nennenswerten Sensitivitäten erfasst werden.[73] Unabhängig von den Faktoren liegen die Ergebnisse des genetischen Algorithmus in allen Experimenten näher am Optimum, als die der Silver-Meal-Heuristik.[74]

3.2 1-Level Probleme mit Kapazitätsrestriktionen

Aufgrund der Einschränkungen in den Kapazitäten kann für eine bestimmte Periode höchstens eine definierte maximale Losgröße gewählt werden. Im ersten Modell ist des Weiteren auch die Lagermenge beschränkt. Eine Überschreitung der maximalen Lagermenge, zieht Zusatzkosten je überschrittene Einheit (Überbestandskosten) nach sich. Mögliche Lieferrückstände werden in diesem Modell berücksichtigt und mit hohen Strafkosten je fehlender Einheit (Unterbestandskosten) verrechnet. Die Zielfunktion sieht wie folgt aus:

[72] ebenda, S. 440
[73] vgl. ebenda, S.439f.
[74] vgl. ebenda, S. 438f.

$$min \sum_{Periode\ t=1}^{Letzte\ Periode} \sum_{Losgröße\ x=0}^{Maximale\ Losgröße} (WirdProduziert_{x,t} * Rüstkosten + Fertigungsskosten_{x,t}) + \sum_{Periode\ t=1}^{Letzte\ Periode} (Überbestandskosten_t + Lagerkosten_t + Unterbestandskosten_t) \quad (5)$$

Die Lagermenge am Ende der Periode *t* berechnet sich dabei wie folgt:

$$Lagermenge_t = Lagermenge_{t-1} + (\sum_{x \in X} (Losgröße_x * WirdProduziert_{x,t})) - Bedarf_t \quad (6)$$

Die Variable $WirdProduziert_{x,t}$ in der Zielfunktion (5) und in der Gleichung (6) nimmt den Wert 1 an, wenn in der Periode *t* die Losgröße *x* gewählt wird und 0 wenn nicht. X in der Gleichung (6) definiert die Menge aller realisierbaren Losgrößen und $Fertigungskosten_{x,t}$ in der Zielfunktion (5) bestimmt die Fertigungskosten für die Produktionsmenge *x* in der Periode *t*, unter der Berücksichtigung sinkender Kosten je zu produzierende Einheit bei größeren Stückzahlen. Im Fall von Unterbestand ergibt das Produkt aus den festgelegten Strafkosten je Einheit und dem entsprechenden Unterbestand, die Unterbestandskosten für die jeweilige Periode. Sonst werden die Lagerkosten je Stück mit der Lagermenge multipliziert, um die Lagerkosten zu berechnen. Im Fall von Überstand wird das Produkt aus zusätzlichen Lagerkosten je Stück und der überschrittenen Lagermenge zu den sonstigen Lagerkosten hinzuaddiert. Die Gleichung in (6) bildet eine Nebenbedingung. In einer weiteren Nebenbedingung fallen Strafkosten nur bei Unterbestand, Zusatzkosten nur bei Überbestand und Lagerhaltungskosten nur bei einem Bestand größer 0 an. Das initiale Lager ist leer und jeder Periode wird genau eine realisierbare Losgröße zugeordnet. Sämtliche Lagermengen und Losgrößen müssen größer/gleich 0 sein.[75]

Aufgrund der Kapazitätsrestriktionen genügt es nicht mehr, nur die Reihenfolge von Rüstung und Lagerhaltung zu optimieren, solange kein festdefinierter Produktionszyklus bei konstantem Bedarf gewählt werden kann, da eine Produktion in der jeweiligen Periode nicht notwendigerweise die Bedarfe für diese und die nächsten k Perioden deckt.[76] Vielmehr muss jede mögliche Losgröße für jede einzelne Periode untersucht werden. Sei x die Anzahl an möglichen Losgrößen und n die Anzahl an Perioden im Planungshorizont. Dann existieren x^n Losgrößenkombinationen, folglich entsprechend viele

[75] vgl. Megala und Jawahar (2006), S. 1179f.
[76] vgl. Khouja u.a. (1998), S. 509f.

Produktionspläne für das jeweilige Losgrößenproblem, sodass die Komplexität der Berechnung relativ schnell ansteigt.[77]

Der hierfür entwickelte genetische Algorithmus kodiert jede mögliche Kombination von Losgrößen in eine Kette von binärverschlüsselten Zahlen, die jeweils eine Losgröße in der jeweiligen Periode repräsentieren (z.B. steht <1010, 0100, 1000> für <10, 4, 8> als Losgrößen in den Perioden 1, 2 und 3). Die Bevölkerungsgröße s wird, abhängig von der Komplexität des Problems, durch das Produkt aus der Anzahl der Perioden und der maximalen Losgröße gebildet. Für die initiale Bevölkerung werden s Losgrößenkombinationen bzw. Chromosomen zufällig erstellt. Die Fit-Funktion berechnet die entstehenden Gesamt-Kosten für die jeweilige Losgrößenkombination anhand der Zielfunktion in (5). Dieser Fit-Wert wird im Anschluss mit $e^{-0.0001*Fit\text{-}Wert}$ auf ein Intervall zwischen 0 und 1 normiert. Zu Beginn der Schleife wird das Chromosom mit dem besten Fit-Wert gespeichert. Im Anschluss daran erfolgt die Selektion nach dem „Roulette Wheel"-Schema, wobei sich die Übernahmewahrscheinlichkeit für einen bestimmten Produktionsplan in die nächste Generation durch den Quotienten aus dem Fit-Wert des betrachteten Produktionsplans und der Summe der Fit-Werte aller Chromosomen ergibt. Danach wird auf jedes Chromosom mit einer Wahrscheinlichkeit von 60% „PMX-Crossover", eine für Zahlenketten geeignete Variante der Kreuzung, angewendet.[78] Nach der Kreuzung erfolgt die Mutation, in der jede einzelne Losgröße in allen Losgrößenkombinationen mit einer Wahrscheinlichkeit von 5% durch einen neu generierten Wert zwischen 0 und der maximalen Losgröße ersetzt wird. Nun ist die neue Generation kreiert und die Schleife beginnt von Neuem, bis eine bestimmte Zahl an Generationen erstellt worden ist. In dem vorliegenden Algorithmus wird diese Zahl, ähnlich wie die Bevölkerungsgröße s, abhängig von der Komplexität des Problems gewählt, nämlich das fünffache des Produkts aus der maximalen Losgröße und der Anzahl an Perioden im Planungshorizont. Nach Beendigung der Schleife wird die beste Losgrößenkombination unter allen in jedem Schleifendurchgang gespeicherten Chromosomen dekodiert und der resultierende Produktionsplan sowie seine Gesamtkosten ausgegeben.[79]

Im zweiten Modell werden minimale Kosten in der Produktion zweier ähnlicher Produkte angestrebt, wobei das eine Produkt etwas höherwertiger ist. Im Fall eines Engpasses

[77] vgl. Megala und Jawahar (2006), S. 1180
[78] Für nähere Erläuterungen zu „PMX-Crossover": https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/PMXCrossoverOperator.aspx - Zuletzt abgerufen am: 18.06.2021
[79] vgl. Megala und Jawahar (2006), S. 1181f.

kann das höherwertige Produkt (im folgenden P2) das minderwertige Produkt (im folgenden P1) substituieren. Des Weiteren werden Rückgaben berücksichtigt, die in der Fertigung wiederaufbereitet/recycelt und im Anschluss als Neuware verkauft werden können. Fremdvergabe ist ebenfalls erlaubt. Sowohl in der Neuproduktion als auch im Recycling wird Batch-Kommissionierung angewendet, sodass die Losgrößen ein Vielfaches einer für beide Produkte festgelegten minimalen Produktionsmenge betragen müssen. Die für die Neuproduktion und für die Wiederaufbereitung beider Produkte zur Verfügung stehenden Ressourcen sind in jeder einzelnen Periode beschränkt. Die Zielfunktion sieht aus wie folgt:

$$min \sum_{WirdRecycelt\ r=0}^{1} \sum_{Produkt\ i=1}^{2} \sum_{Periode\ t=1}^{Letzte\ Periode} \begin{matrix} (Rüstkosten_{r,i,t} * WirdProduziert_{r,i,t} + Stückkosten_{r,i,t} \\ * Fertigungsmenge_{r,i,t} + LagerkostenJeStück_{r,i,t} * Lagermenge_{r,i,t} \\ + (1-r) * FremdvergabekostenJeStück_{i,t} * Fremdvergabemenge_{i,t}) \end{matrix} \quad (7)$$

Die Gesamtkosten in (7) werden durch die Summe aus den Gesamtkosten für die Neuproduktion *(r = 0)* und den Gesamtkosten für die Wiederaufbereitung *(r = 1)* jeweils für das minderwertige (*i = 1)* und für das höherwertige *(i = 2)* Produkt berechnet. Die Gesamtkosten für jede Kombination aus Neuproduktion/Wiederaufbereitung bzgl. Produkt 1/Produkt 2 wird durch die Summe aus Rüst-, Fertigungs- und Lagerhaltungskosten gebildet, wobei in der Neuproduktion ggf. anfallende Fremdvergabekosten hinzuaddiert werden. Die Kosten für Rüstung, Fertigung und Lagerhaltung unterscheiden sich je nachdem, ob sie Rückgaben betreffen oder nicht.[80]

Die Lagermenge für das Produkt *i* am Ende der Periode *t* berechnet sich dabei wie folgt…

Für Rückgaben:

$$Lagermenge_{i,t} = Lagermenge_{i,t-1} - Fertigungsmenge_{0,i,t} + Rückgabemenge_{i,t} \quad (8)$$

Für verkaufsbereite Produkte:

$$Lagermenge_{1,t} = Lagermenge_{1,t-1} + Fertigungsmenge_{1,t} + Fremdvergabemenge_{1,t} + Substitutionsmenge_{2,t} - Bedarf_{1,t} \quad (9)$$

$$Lagermenge_{2,t} = Lagermenge_{2,t-1} + Fertigungsmenge_{2,t} + Fremdvergabemenge_{2,t} - Substitutionsmenge_{2,t} - Bedarf_{2,t} \quad (10)$$

[80] vgl. Li u.a. (2007), S. 303f.

In den Gleichungen (9) und (10) gibt $Substitutionsmenge_{2,t}$ an, wie viele Einheiten von Produkt 2 zur Erfüllung des Bedarfs von Produkt 1 in der Periode *t* verwendet werden und $Fertigungsmenge_{i,t}$ gibt die Gesamtzahl der über Neuproduktion und über Recycling produzierten Mengen des Produkts *i* in der Periode *t* an. Die Gleichungen in (8) bis (10) sind Nebenbedingungen der Zielfunktion in (7). Darüber hinaus dürfen die benötigten Ressourcen für die geplanten Fertigungsmengen sowohl in der Neuproduktion als auch in der Wiederaufbereitung die in den jeweiligen Perioden zur Verfügung stehenden Mengen an Ressourcen nicht überschreiten. Um eine sinnvolle Substitution von Produkt 1 durch Produkt 2 zu gewährleisten, entsprechen alle Kosten (Rüst-, Lager-, usw.) bzgl. Produkt 2 mindestens den Kosten bzgl. Produkt 1 und die Fremdvergabekosten für Produkt 1 sind stets höher als die Herstellungskosten für Produkt 2. Des Weiteren ist die Summe aus Rüst- und Herstellungskosten für ein Vielfaches der minimalen Produktionsmenge eines Produkts in allen Fällen maximal so hoch wie die Fremdvergabekosten für dieselbe Menge des Produkts.[81]

Da bei diesem Problem die Suche nach einem optimalen Produktionsplan mittels dynamischer Programmierung alleine zu viel Rechenzeit in Anspruch nimmt, wird ein hybrider Ansatz gewählt, in welcher der genetische Algorithmus von der dynamischen Programmierung begleitet wird. Der Ansatz versucht zuerst eine Lösung ohne Fremdvergaben zu finden. Hierzu wird jeder mögliche Produktionsplan in eine 4Xn-Matrix (das Chromosom) kodiert, wobei n für die Anzahl an Perioden im Planungshorizont steht. In jeder Zeile einer Matrix wird eine Binärkette gespeichert, die angibt, in welchen Perioden jeweils eine Rüstung erfolgen muss.[82] Das ist ein signifikanter Unterschied zum ersten Algorithmus, wo die genauen Losgrößen in der Binärkette hinterlegt und folglich vom genetischen Algorithmus direkt verarbeitet werden.[83] Die erste Zeile in der Matrix repräsentiert Neuproduktionen, die zweite Wiederaufbereitungen, beide bzgl. Produkt 1 und die dritte Zeile repräsentiert Neuproduktionen, die vierte Wiederaufbereitungen, beide bzgl. Produkt 2.[84] Anders als im ersten Algorithmus müssen zur Berechnung des Fit-Werts erst die optimalen Losgrößen für die mit 1 gekennzeichneten Perioden in den jeweiligen Zeilen der Matrix mittels dynamischer Programmierung ermittelt und anschließend die resultierenden

[81] vgl. ebenda, S. 304f.
[82] vgl. ebenda, S. 305f.
[83] vgl. Megala und Jawahar (2006), S. 1181
[84] vgl. Li u.a. (2007), S. 306

Gesamtkosten mit der Zielfunktion in (7) ausgerechnet werden.[85] Das Chromosom mit dem besten Fit-Wert wird direkt in die nächste Generation aufgenommen. Die übrigen Chromosomen werden nach ihrem Fit-Wert in absteigender Reihenfolge sortiert und abhängig von ihrer Position im Ranking mit der Wahrscheinlichkeit 2*(Bevölkerungsgröße+1–Position) / (Bevölkerungsgröße*(Bevölkerungsgröße+1)) zur Kreuzung gewählt. Der jeweilige Partner wird per Gleichverteilung zufällig aus der übrigen Bevölkerung ermittelt.[86] Da ein Produktionsplan wieder als eine simple Kette von binären Ziffern vorliegt, ist der Einsatz von speziellen, im ersten Algorithmus verwendeten Verfahren (PMX-Crossover) nicht erforderlich, sodass die Kreuzung größtenteils genauso abläuft, wie im unbeschränkten Fall.[87] Dabei werden die beiden Matrizen in derselben zufällig gewählten Spalte (Periode) geteilt und jeweils die linke Seite der einen Matrix mit der rechten Seite der anderen Matrix verknüpft. Auf diese Weise bilden sich zwei neue Matrizen (die Nachfahren), deren Fit-Werte im Anschluss ermittelt werden und der Bessere der beiden Nachfahren ersetzt die Schwächere der beiden ursprünglichen Matrizen in der neuen Generation. Die anschließende Mutation wandelt in jeder Matrix mit einer geringen Wahrscheinlichkeit alle Werte einer zufällig gewählten Spalte jeweils von 0 in 1 oder umgekehrt.[88] Der Algorithmus endet sobald entweder die festgelegte maximale Anzahl an Generationen kreiert worden ist, die Fit-Werte zwischen den Generationen sich kaum mehr verbessern oder die Matrizen innerhalb der Generationen sich kaum mehr unterscheiden. Anders als bei Losgrößenproblemen mit unbeschränkter Kapazität kann hierbei nicht jede Matrix in die Bevölkerung aufgenommen werden, daher ist sowohl bei der Erstellung der initialen Bevölkerung als auch nach jeder Kreuzung und Mutation eine Überprüfung der entstehenden Matrizen auf Zulässigkeit erforderlich.[89] Wenn der Algorithmus keine zulässige Matrix findet, werden Fremdvergaben in Betracht gezogen, um eine passende Lösung zu finden.[90]

Die Ergebnisse aus den Experimenten zum ersten genetischen Algorithmus werden mit den Ergebnissen aus einem alternativen Ansatz, basierend auf künstlichen neuronalen Netzen (Hopfield neural network), für unterschiedlich komplexe Losgrößenprobleme anhand ihrer Abweichung vom Optimum verglichen. In den meisten Fällen entsprechen die Ergebnisse des genetischen Algorithmus dem Optimum oder kommen diesem sehr nah.

[85] vgl. ebenda, S. 308; Megala und Jawahar (2006), S. 1181
[86] vgl. Li u.a. (2007), S. 306
[87] vgl. ebenda; Megala und Jawahar (2006), S. 1182
[88] vgl. Li u.a. (2007), S. 306
[89] vgl. ebenda, S. 307
[90] vgl. ebenda, S. 305

Folglich kann der genetische Algorithmus in denselben Experimenten Ergebnisse mit einer höheren Qualität erzielen, als der Ansatz mit den neuronalen Netzen.[91]

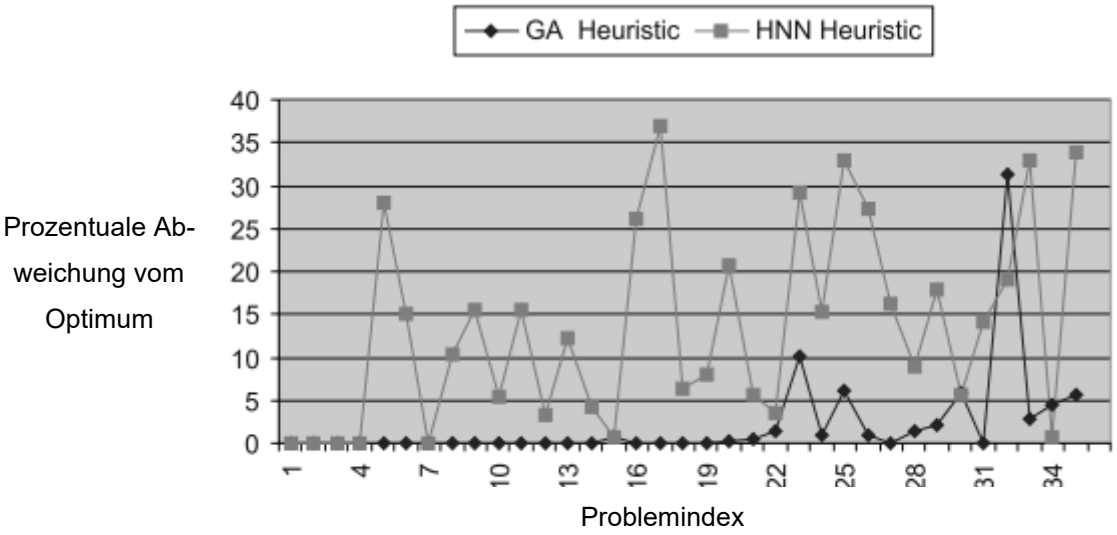


Abb. 3: Qualität der Ergebnisse des genetischen Algorithmus (GA Heuristic) gegenüber der der neuronalen Netze (HNN Heuristic)[92]

Aus den Untersuchungen bzgl. dem zweiten, hybriden Algorithmus geht hervor, dass der vorgestellte Ansatz Produktionspläne in akzeptabler Rechenzeit liefert, die ebenfalls sehr nah am Optimum liegen. Dieses Verfahren wird einem branch-and-bound Algorithmus gegenübergestellt. Bei dem hybriden Ansatz fallen die resultierenden Gesamtkosten in fast allen Durchgängen geringer aus, als bei dem Vergleichsalgorithmus. Darüber hinaus benötigt dieser Ansatz nur wenige Minuten Rechenzeit, während der Vergleichsalgorithmus in manchen Fällen auch nach mehreren Stunden keine Lösung findet, weswegen die Durchgangslänge in den Experimenten auf 10 Minuten beschränkt worden ist.[93]

3.3 n-Level Probleme ohne bzw. mit Kapazitätsrestriktionen

Gewöhnlich werden Endprodukte aus einer Vielzahl von Komponenten zusammengebaut, die wiederum aus einer Vielzahl an Unterkomponenten bestehen. Dies setzt sich bis zur untersten Ebene fort, in welcher die benötigten Teile nicht mehr vom eigenen Unternehmen produziert, sondern eingekauft werden. Abbildung 4 zeigt verschiedene

[91] vgl. Megala und Jawahar (2006), S. 1184f.
[92] ebenda, S. 1185
[93] vgl. Li u.a. (2007), S. 311f.

Wege, wie die Bauteile (2), 3, 4, 5 in die Endprodukte 1, (2) eingearbeitet werden können.[94]

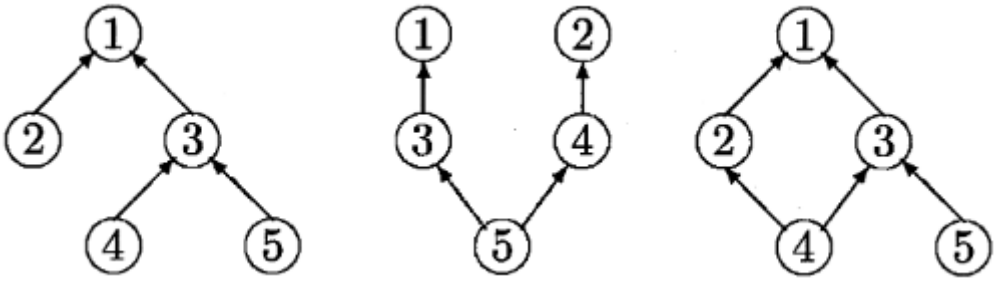


Abb. 4: Simple Beispiele für den Aufbau von Produktionsstrukturen[95]

Die Berechnung optimaler Losgrößen für alle Bauteile auf allen Ebenen zieht der Komplexität geschuldete enorme Rechenzeit nach sich, weshalb in praktischen Situationen Heuristiken zu bevorzugen sind, die eine Balance zwischen Qualität der Ergebnisse und benötigter Rechenzeit schaffen.[96] Ein Ansatz aus der Praxis besteht darin, Algorithmen für 1-Level Probleme (z.B. die Silver-Meal Heuristik) abwärts in der Produktionsstruktur auf jede Komponente einzeln anzuwenden. Allerdings werden hier bei der Wahl der Losgrößen die Einflüsse der Entscheidungen auf die Komponenten der darunterliegenden Ebenen nicht berücksichtigt und entsprechende Potenziale zur Kosteneinsparung übersehen, weshalb ein adäquater Ansatz für ein solches n-Level Problem auf der gesamten Produktionsstruktur basieren muss.[97]

In dem ersten betrachteten Modell werden alle Produkte (Endprodukte, Komponenten und Rohmaterialen) in topologischer Reihenfolge durchnummeriert. Das bedeutet, dass ein Bauteil stets einen höheren Index zugewiesen bekommt, als die Komponente/das Endprodukt, für dessen Herstellung es verwendet wird. Ist der Einsatz eines Bauteils in der Fertigung eines bestimmten Erzeugnisses erforderlich, so ergibt sich das jeweilige Produktionsverhältnis zwischen Bauteil und Erzeugnis aus der Anzahl an Einheiten des Bauteils, die für die direkte Herstellung einer Einheit des Erzeugnisses benötigt wird. Das Produktionsverhältnis ist gleich 0, wenn ein Bauteil erst zur Zwischenkomponente und dann zum betrachteten Erzeugnis (Komponente oder Endprodukt) verarbeitet wird. Die Zielfunktion lautet:

[94] vgl. Dellaert u.a. (2000), S. 243
[95] ebenda, S. 243
[96] vgl. ebenda, S. 242
[97] vgl. Prasad und Krishnaiah Chetty (2001), S. 520

$$min \sum_{Produkt\ i=1}^{Anzahl\ Produkte} \sum_{Periode\ t=1}^{Letzte\ Periode} (Stückskosten_{i,t} * Liefermenge_{i,t} + Rüstkosten_{i,t} * WirdProduziert_{i,t} + LagerkostenJeStück_{i,t} * Lagermenge_{i,t}) \quad (11)$$

Sowohl die Stück- und die Rüstkosten als auch die Lagerkosten je Stück variieren zwischen den einzelnen Perioden im Planungshorizont. Die Nebenbedingungen lauten:

$$Lagermenge_{i,t} = Liefermenge_{i,t} - Nettobedarf_{i,t} \quad (12)$$

$$Nettobedarf_{i,t} = Bruttobedarf_{i,t} - Lagermenge_{i,t-1} - GeplanterLagerzugang_{i,t} \quad (13)$$

$$Bruttobedarf_{i,t} = \sum_{alle\ Erzeugnisse\ j\ von\ Produkt\ i} (Produktionsverhätnis_{i,j} * Liefermenge_{j,t'}) \quad (14)$$

Die Periode *t'* in (14) ist die um die Lieferzeit von Produkt *j* nach hinten verschobene Periode *t*. Die neue Lagermenge des Produkts *i* ergibt sich durch die Differenz aus der Liefermenge und dem Nettobedarf jeweils in der Periode *t*. Um den vorlaufverschobenen Bruttobedarf eines Bauteils *i* in der Periode *t* zu ermitteln, wird das jeweilige Produktionsverhältnis zwischen dem Produkt *i* und seinem Erzeugnis mit der geplanten Liefermenge des Erzeugnisses für die Periode *t+l* (*l* = Lieferzeit des Erzeugnisses) multipliziert und die Resultate über alle Erzeugnisse von *i* hinweg aufsummiert.[98]

Im vorliegenden genetischen Algorithmus wird jeder mögliche Produktionsplan in eine binäre pXn-Matrix kodiert, wobei p die Anzahl der Produkte (Endprodukte, Komponenten und Rohmaterialien) und n die Anzahl der Perioden im Planungshorizont repräsentiert.[99] Da nun Losgrößenprobleme wieder ohne Restriktionen in den Kapazitäten betrachtet werden, genügt an jeder Position in der Matrix eine 1 oder eine 0, um erforderliche Rüstungen in den jeweiligen Perioden anzugeben, ohne die genauen Losgrößen zu speichern. Folglich stellt jede Zeile in der Matrix eine Binärkette für ein jeweiliges Produkt dar.[100] Alternativ können die Zeilen auch hintereinander geschrieben werden, sodass eine Binärkette mit der Länge p*n entsteht.[101] Um die Qualität einer Matrix zu bewerten, werden die Losgrößen jedes Produkts für die Perioden, in denen eine Rüstung verbucht ist, anhand der Bedarfe dieser und aller nachfolgenden Perioden ohne Rüstung ermittelt und die daraus resultierenden Gesamtkosten mit der Zielfunktion in (11) berechnet.[102] Für die initiale Bevölkerung werden Matrizen nach der „lot-for-lot" Heuristik erstellt. Das bedeutet, dass für jedes Produkt in jeder Periode, in der ein Bedarf vorliegt, eine Rüstung

[98] vgl. Dellaert u.a. (2000), S. 243f.
[99] vgl. ebenda, S. 244f.; Prasad und Krishnaiah Chetty (2001), S. 520
[100] vgl. Dellaert u.a. (2000), S. 244f.
[101] vgl. Prasad und Krishnaiah Chetty (2001), S. 521
[102] vgl. Dellaert u.a. (2000), S. 248; Prasad und Krishnaiah Chetty (2001), S. 521

erfolgt.[103] Typischerweise wird die Bevölkerungsgröße auf das doppelte der Anzahl an Einträgen in der Matrix, also auf s=2*n*p gesetzt.[104] Diese Matrizen werden wiederholt mutiert, um eine Vielfalt in der Bevölkerung herzustellen.[105] Auf sie werden im Anschluss genetische Operatoren der Reihe nach wiederholt mit gewisser Wahrscheinlichkeit angewendet, um Nachfahren für die neue Bevölkerung zu kreieren. Abhängig von ihrem Wert in der Zielfunktion werden die ursprünglichen Matrizen mit einer entsprechenden Wahrscheinlichkeit in die neue Bevölkerung mitaufgenommen und bilden zusammen mit ihren Nachfahren die neue Generation.[106] Um den Algorithmus effizienter zu gestalten wird der Lösungsraum verkleinert, indem jede Matrix vor der Aufnahme in die neue Bevölkerung auf Zulässigkeit geprüft und ggf. angepasst oder entfernt wird. Unzulässige Matrizen enthalten Binärketten, in der entweder bis zur ersten Bedarfsperiode nicht genau eine, zwischen zwei Bedarfsperioden mehr als eine oder nach der letzten Bedarfsperiode mindestens eine Rüstung erfolgt.[107] Darüber hinaus müssen alle Bedarfe unter Berücksichtigung der Lieferzeiten erfüllbar sein (keine Rohstoffe und Komponenten können vor der ersten Periode produziert oder eingekauft werden).[108] Um die Suche zu beschleunigen, können die Zeilen, welche (unnötige) Rüstungen in Perioden enthalten, in denen kein Bedarf vorliegt, korrigiert werden.[109] Für die Reproduktion werden zwei Arten der Mutation, zwei Arten der Kreuzung und die Inversion eingesetzt. In der 1-Bit Mutation wird der Wert bei einem beliebigen Produkt in einer zufälligen Periode in der Matrix von 0 in 1 oder umgekehrt gesetzt. In der kumulativen Mutation wird die entsprechende Mutation auch auf alle Komponenten, aus denen das betroffene Produkt direkt gefertigt wird (Produktionsverhältnis größer 0), um die Lieferzeit vorlaufverschoben, angewendet. Bei beiden Varianten entsteht jeweils ein Nachfahre. In der periodenbezogenen Kreuzung wird jede Binärkette der beiden Matrizen in einer beliebigen Periode geteilt und die linke Hälfte der Binärkette der einen Matrix mit der rechten Hälfte der Binärkette der anderen Matrix verknüpft.[110] Die Periode, in der die Teilung erfolgt, wird für jede Zeile neu gewählt.[111] In der produktbezogenen Kreuzung werden die beiden Matrizen zwischen zwei beliebig gewählten aufeinanderfolgenden Zeilen geteilt und jeweils die oberen Zeilen der einen Matrix mit den unteren Zeilen der anderen Matrix verknüpft.[112] Bei beiden Varianten

[103] vgl. Dellaert u.a. (2000), S. 248
[104] vgl. Prasad und Krishnaiah Chetty (2001), S. 521
[105] vgl. Dellaert u.a. (2000), S. 248
[106] vgl. ebenda, S. 249
[107] vgl. ebenda, S. 246f.
[108] vgl. ebenda, S. 245f.
[109] vgl. Prasad und Krishnaiah Chetty (2001), S. 522
[110] vgl. Dellaert u.a. (2000), S. 247f.
[111] vgl. ebenda, S. 248ff.
[112] vgl. ebenda

entstehen jeweils zwei Nachfahren. Für die Inversion wird der Wert bei einem beliebigen Produkt in einer zufälligen Periode mit einem (falls vorhanden) abweichenden Wert in der vorherigen oder nachfolgenden Periode getauscht. Dies wird ebenfalls für alle Komponenten durchgeführt, aus denen das betroffene Produkt gefertigt wird.[113] Der Algorithmus endet, sobald entweder eine bestimmte Anzahl an Generationen erstellt worden ist, oder wenn keine bedeutsamen Verbesserungen zwischen den Generationen mehr vorliegen.[114]

Der Einsatz von MRP in der Praxis sorgt für eine hohe Relevanz der Entscheidungsfindung in der kapazitätsbeschränkten Losgrößenoptimierung für verschiedene Endprodukte, die aus mehreren Bauteilen jeweils in aufeinanderfolgenden Produktionsstufen gefertigt werden. Dabei muss die Fertigung der einzelnen Bauteile genau geplant werden, um die Nachfrage an den Endprodukten zu bedienen, Kapazitätsrestriktionen nicht zu überschreiten und die Gesamtkosten gering zu halten.[115] Wird darüber hinaus auch die Zeit berücksichtigt, welche für die Rüstungen der Maschinen/Maschinengruppen in Anspruch genommen wird, entsteht ein nur schwer durchdringbares Optimierungsproblem. Ähnlich wie im Fall ohne Restriktionen in den Kapazitäten nehmen hierfür entwickelte Heuristiken für gewöhnlich eine Optimierung Stufe für Stufe in der jeweiligen Produktionsstruktur vor.[116]

Im vorliegenden mathematischen Modell wird die Produktion mehrerer verschiedener Endprodukte betrachtet, welche in Produktgruppen unterteilt sind. Jedes Produkt ist genau einer Produktgruppe zugeordnet. Lieferrückstände sind erlaubt und mit entsprechenden Strafkosten je Einheit zu verrechnen. Im Gegensatz zu den vorangegangenen Zielfunktionen werden die Rüstkosten im vorliegenden Fall nicht berücksichtigt:

$$min \sum_{Produkt\ i=1}^{Anzahl\ Produkte} \sum_{Periode\ t=1}^{Letzte\ Periode} (StrafkostenJeStück_i * Unterbestand_{i,t} + LagerkostenJeStück_i * Lagermenge_{i,t}) \quad (15)$$

Die Summe aus Straf- und Lagerkosten soll unter Berücksichtigung der Kapazitätsrestriktionen möglichst geringgehalten werden. Die Nebenbedingungen zur Lagermenge sowie zum Netto- und zum Bruttobedarf sind dieselben wie in den Gleichungen (12) – (14), mit dem einzigen Unterschied, dass in der Gleichung aus (13) zum Nettobedarf eines

[113] vgl. ebenda, S. 248
[114] vgl. ebenda, S. 249
[115] vgl. Xie und Dong (2002), S. 263
[116] vgl. ebenda, S. 264

Produkts *i* in der Periode *t* nicht der geplante Lagerzugang aus der Periode *t* abgezogen, sondern stattdessen die vorliegende Rückstandsmenge aus der vorangegangenen Periode (*t-1*) hinzuaddiert wird. Des Weiteren gilt folgende Restriktion:

$$\sum_{Produkt\ i=1}^{Anzahl\ Produkte} AuslastungJeStück_{m,i} * Losgröße_{i,t} \leq Kapazität_{m,t} - Rüstzeit_{m,t} \quad (16)$$

In der Ungleichung (16) definiert $Kapaztität_{m,t}$ die festgelegte Dauer, die eine Maschine/Maschinengruppe *m* in der Periode *t* zur Verfügung steht. $AuslastungJeStück_{m,i}$ gibt an, wie lange die Maschine/Maschinengruppe *m* für die Produktion einer Einheit des Produkts *i* benötigt und $Rüstzeit_{m,t}$ gibt die gesamte Dauer an, die eine Maschine/Maschinengruppe *m* in der Periode *t* mit der Vorbereitung auf die Produktion (Rüstung) aller für die Periode *t* geplanten Produktgruppen beschäftigt ist und daher nichts produziert. Die Rüstungen werden anhand der Produktgruppen vollzogen, d.h. die Losgröße eines Produkts ist gleich 0, wenn in derselben Periode die entsprechende Maschine/Maschinengruppe nicht auf die dahinterstehende Produktgruppe gerüstet wird. Weiterhin müssen die Losgrößen eines Produkts *i* so gewählt werden, dass sie die Ungleichung aus (16) erfüllen.[117] Die Betrachtung unterschiedlicher Endprodukte mit komplexen Produktionsstrukturen unter Berücksichtigung verschiedener Ressourcen (Maschinen) stellt für die Anwendbarkeit entwickelter Methoden in der praxisbezogenen Produktionsplanung ein wichtiges Kriterium dar.[118]

Ähnlich wie im zweiten Algorithmus im zweiten Unterkapitel des Hauptteils wird hierfür ein hybrider Ansatz gewählt, der die Stärken des genetischen Algorithmus und der linearen Programmierung vereint. Der genetische Algorithmus legt fest, in welchen Perioden bei den jeweiligen Produktgruppen eine Rüstung erfolgen soll und zieht die hierfür benötigte Gesamt-Rüstdauer der Maschine/Maschinengruppe *m* von ihrer insgesamt zur Verfügung stehenden Kapazität ab. Dazu werden die Produktionspläne in pXn-Matrizen kodiert, wobei p diesmal die Anzahl der Produktgruppen repräsentiert. Eine 1 in der Matrix steht für eine Rüstung der Maschine/Maschinengruppe auf die jeweilige Produktgruppe in der entsprechenden Periode. Um den Fit-Wert einer Matrix zu berechnen werden anhand der neuen Kapazitätswerte obere Schranken für die Losgrößen der einzelnen Produkte ermittelt. Durch den Einsatz von linearer Programmierung werden unter

[117] vgl. Toledo u.a. (2013), S. 911f.
[118] vgl. Xie und Dong (2002), S. 264

Berücksichtigung der jeweiligen Schranken optimale Losgrößen berechnet und mit der Zielfunktion aus (15) ausgewertet. Der genetische Algorithmus arbeitet parallel mit mehreren Bevölkerungen. Basierend auf den Fit-Werten wird innerhalb der einzelnen Bevölkerungen eine Baumstruktur kreiert, in der die besseren, als Knoten repräsentierten, Produktionspläne, stets auf einem höheren Level zu finden sind.[119]

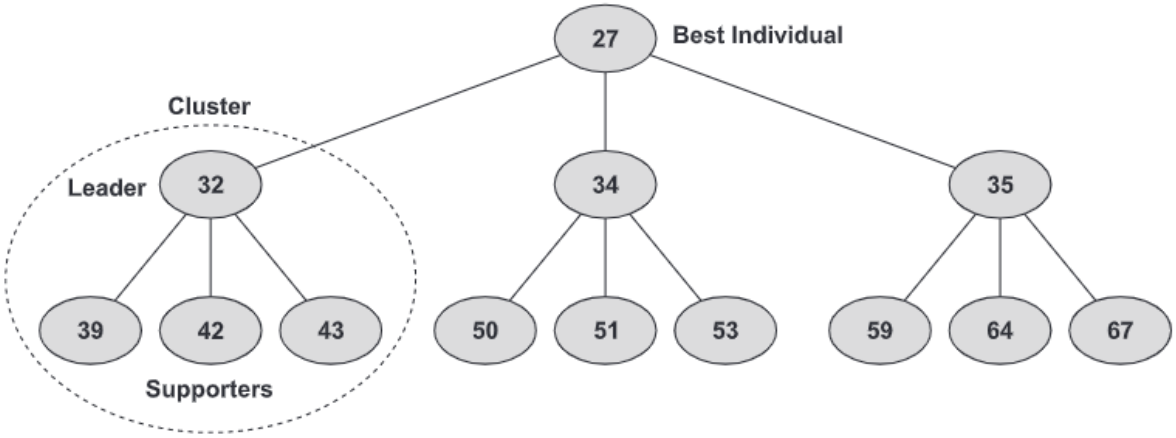


Abb. 5: Beispielhafte Baumstruktur innerhalb einer Bevölkerung[120]

Eltern-Kind-Knotenpaare werden zufällig zur Kreuzung gewählt und mit den genetischen Operatoren wird jeweils genau ein Nachfahre gebildet und mutiert. Wenn der Nachfahre einen besseren Fit-Wert als die Schlechtere der beiden ursprünglichen Matrizen aufweist, ersetzt der Nachfahre diese und der Vorgang wird in der Bevölkerung der festgelegten Kreuzungsrate entsprechend oft wiederholt. Am Ende wird die Baumstruktur in der Bevölkerung neu organisiert, sodass die besseren Produktionspläne wieder über den Schlechteren stehen. Dieser Prozess wird für jede Bevölkerung solange wiederholt, bis die durch die Kreuzung entstehenden Matrizen nicht mehr in die Bevölkerung aufgenommen werden (weil ihre Fit-Werte die der ursprünglichen Matrizen nicht mehr übertreffen).

Anschließend wird die beste Matrix über alle Bevölkerungen gewählt und mit einer weiteren Heuristik (fix-and-optimize) optimiert.[121] Diese Heuristik sieht nur eine bestimmte Anzahl an aufeinanderfolgenden Zeilen (Produktgruppen) oder Spalten (Perioden) in der Matrix als variabel an. Für die variablen Positionen berechnet die Heuristik optimale Werte (jeweils 0 oder 1), während die übrigen Positionen als konstant betrachtet werden.[122] Ist die Heuristik in der Lage den Fit-Wert der ausgewählten Matrix zu erhöhen, so

[119] vgl. Toledo u.a. (2013), S. 912f.
[120] ebenda, S. 913
[121] vgl. ebenda, S. 912f
[122] vgl. ebenda, S. 914f.

wird die Wurzel (oberster Knoten) der Baumstruktur jeder Bevölkerung auf die verbesserte Matrix gesetzt. Ansonsten wird die Anzahl an aufeinanderfolgenden Spalten/Zeilen, die in der Heuristik als variabel betrachtet werden sollen, (für den nächsten Durchgang) um eins erhöht. In beiden Fällen beginnt der Algorithmus nun von vorne und endet, sobald ein definiertes Zeitlimit erreicht wird.[123]

In den Experimenten zum ersten Algorithmus werden kleine Probleme mit 10 unterschiedlichen Produkten über 4 Produktionsstufen und einen Planungshorizont von 12 Perioden sowie mittelgroße Probleme mit 50 Produkten über 10 Produktionsstufen und einen Planungshorizont von einmal 24 und einmal 36 Perioden betrachtet. In beiden Fällen werden Produktionsstrukturen unterschiedlicher Komplexität getestet. Die Komplexität ergibt sich aus der Anzahl an direkten Abhängigkeiten zwischen den Produkten, d.h. wenn ein Produkt in der Herstellung eines anderen Produkts verarbeitet wird.[124] Dabei kann der genetische Algorithmus die besten Ergebnisse erzielen, wenn die Bedingungen so festgelegt werden, dass der Algorithmus endet, sobald er in 50 aufeinanderfolgenden Generationen keine signifikante Verbesserung mehr erzielt. Die resultierenden Gesamtkosten durch den genetischen Algorithmus sind geringer (bei mittelgroßen Problemen sogar deutlich), als die der Vergleichsalgorithmen „lot-for-lot“ und „best item ordering cycle“. Bei Letzterem wird für jedes Produkt ein eigener Produktionszyklus berechnet, welcher die Kosten minimiert. Darüber hinaus liegt die Rechenzeit des genetischen Algorithmus in kleinen Problemen bei etwa 1,11 Sekunden, in mittelgroßen Problemen bei etwa 30 – 180 Sekunden und damit in einem akzeptablen Rahmen.[125]

Die Experimente, die für den zweiten Algorithmus durchgeführt werden, untersuchen die Qualität der Ergebnisse des vorgestellten hybriden Ansatzes im Vergleich zu zwei anderen Heuristiken. Hieraus geht hervor, dass bei einem Losgrößenproblem für 11 Produktgruppen in 16 Perioden und einem Zeitlimit von 100 – 300 Sekunden der hybride Ansatz bessere Ergebnisse liefert. Je komplexer die Produktionsstrukturen, desto deutlicher zeichnet sich dies ab. Bei einem längeren Planungshorizont von 24 Perioden ist dies allerdings nicht mehr der Fall.[126] Setzt man das Zeitlimit auf das 10-fache hoch, so verbessern sich die Ergebnisse des hybriden Ansatzes laufend, während die Ergebnisse der anderen beiden Heuristiken nach einer gewissen Zeit konvergieren.[127] Der genetische

[123] vgl. ebenda, S. 912f.
[124] vgl. Dellaert u.a. (2000), S. 249f.
[125] vgl. ebenda, S. 251ff.
[126] vgl. Toledo u.a. (2013), S. 915f.
[127] vgl. ebenda, S. 916f.

Algorithmus durchsucht nämlich den gesamten Lösungsraum nach einem globalen Optimum, während die beiden anderen Heuristiken nach einer bestimmten Rechenzeit im lokalen Optimum verharren.[128] Bei einem längeren Planungshorizont erreicht der hybride Ansatz somit bessere Ergebnisse, wenn das Zeitlimit entsprechend angepasst wird.[129]

3.4 Vergleich und Analyse

In 1-Level Problemen ohne Kapazitätsrestriktionen genügt eine einzelne Binärkette zum Darstellen eines potenziellen Produktionsplans. Hierfür müssen keine exakten Losgrößen gespeichert werden, diese lassen sich nämlich sehr einfach aus der betrachteten Binärkette ableiten. Die Binärkette zeigt auf, in welchen Perioden eine Rüstung zu erfolgen hat und in welchen nicht. Die Losgröße einer Periode, in der eine Rüstung erfolgt, muss den Bedarf für diese und alle nachfolgenden Perioden bis zur nächsten Rüstung decken. Auf diese Weise lässt sich der Fit-Wert direkt aus der Binärkette berechnen, indem die Losgrößen abhängig von den Bedarfswerten bestimmt und die resultierenden Kosten ermittelt werden. Die genetischen Operatoren sind bei dieser Form der Kodierung leicht anzuwenden. Eine Kreuzung erfolgt durch eine Teilung der Binärketten, eine Mutation durch das Setzen eines Wertes von 0 auf 1 oder umgekehrt. In einer Variante mit Batch-Kommissionierung muss die Kodierung des Produktionsplans so angepasst werden, dass zu jeder Rüstung die Information enthalten ist, ob bei der Bestimmung der Losgröße das nächstgrößere oder das nächstkleinere Vielfache zu wählen ist. Die genetischen Algorithmen können einen optimalen Produktionsplan bei dieser Klasse von Problemen relativ zeiteffizient berechnen. Je variabler die Wahl der Bedingungen (z.B. saisonalen statt konstanten Bedarf), desto deutlicher zeichnet sich die Stabilität der Ergebnisse des genetischen Algorithmus gegenüber der seiner Vergleichsalgorithmen ab.

In 1-Level Problemen mit Kapazitätsrestriktionen dürfen die Losgrößen nur bis zu einer oberen Grenze hoch gewählt werden. Daher kann die Rüstung in einer Periode nicht den Bedarf von beliebig vielen nachfolgenden Perioden decken. Es gibt zwei Varianten, um dies in genetischen Algorithmen zu realisieren. Bei der einen Variante werden in der Kodierung sämtliche Losgrößen direkt gespeichert. Die resultierende Binärkette ist folglich eine Aneinanderreihung von binärverschlüsselten Losgrößen. Der Vorteil besteht darin, dass der Fit-Wert direkt aus dem Chromosom (der Losgrößenkette) berechnet werden

[128] vgl. Holland (1992), S. 67
[129] vgl. Toledo u.a. (2013), S. 917f.

kann, da die Losgrößen hierzu lediglich abgelesen und in die Zielfunktion eingesetzt werden müssen. Für die Reproduktion der Zahlenketten wird eine spezielle Variante der Kreuzung eingesetzt. Auch der resultierende Produktionsplan kann am Ende des Algorithmus einfach aus der Losgrößenkette entnommen werden. Allerdings sind die Binärketten, welche die potenziellen Produktionspläne repräsentieren, dadurch sehr lang und der Lösungsraum ist deutlich größer, als bei den Algorithmen, in deren Kodierung nur die Rüstungen verbucht werden. Die andere Variante beschreibt einen hybriden Ansatz, in welcher dynamische Programmierung innerhalb des genetischen Algorithmus zum Einsatz kommt. Dazu werden die Rüstungen in einer gewöhnlichen Binärkette der Länge n (Anzahl Perioden) festgehalten, wie es auch bei den Problemen ohne Kapazitätsrestriktionen der Fall ist. Der Fit-Wert kann jedoch nicht mehr direkt aus der Binärkette bestimmt werden, denn hierfür müssen zunächst die optimalen Losgrößen für diejenigen Perioden, in denen eine Rüstung verbucht ist, mittels dynamischer Programmierung berechnet werden. Hingegen können sowohl die Kreuzung als auch die Mutation auf die gleiche Weise vonstattengehen, wie beim unbeschränkten Fall, da die Binärketten hier auf demselben Muster basieren. Allerdings können die genetischen Operatoren aufgrund der Restriktionen Produktionspläne hervorbringen, die in der vorliegenden Produktionslandschaft nicht umsetzbar sind, weswegen jede Binärketten vor der Aufnahme in die Bevölkerung auf Zulässigkeit geprüft werden muss. Bei der Klasse von Problemen liegen die Ergebnisse aus den Experimenten mit akzeptabler Rechenzeit nah am Optimum und zeigen auch eine bessere Qualität, als die Ergebnisse hochwertiger Verfahren, wie der künstlichen neuronalen Netze.

Für n-Level Probleme wird die binäre Matrixdarstellung bei der Kodierung der Produktionspläne gewählt, wobei jede Zeile der Matrix den Produktionsplan eines Endprodukts oder eines Bauteils repräsentiert. Bei der Berechnung der Gesamtkosten kommt eine doppelte Summe zum Einsatz, in der die Kosten über sämtliche Perioden aller Produkte aufsummiert werden. Ohne Kapazitätsrestriktionen kann der Fit-Wert auf eine ähnliche Weise direkt berechnet werden wie bei dem entsprechenden 1-Level Problem. Im Fall von Kapazitätsrestriktionen bietet sich ein hybrider Ansatz an, in der optimale Losgrößen mittels linearer Programmierung für die entsprechenden Einträge in der binären Matrix berechnet werden. Für Letzteres kommt im hierzu vorgestellten Algorithmus eine weitere Heuristik zum Einsatz, welche die binäre Matrix mit dem besten Fit-Wert noch einmal separat optimiert, ohne konkrete Losgrößen zuzuweisen. In den Experimenten wird gezeigt, dass die Verbesserungen durch genetische Algorithmen gegenüber vergleichbaren

Heuristiken deutlicher werden, je komplexer die Produktionsstruktur ist. Der genetische Algorithmus arbeitet dabei am besten, wenn eine längere Laufzeit zugelassen und ihm auf diese Weise die Gelegenheit gegeben wird, den gesamten Lösungsraum nach einem globalen Optimum zu durchsuchen.

4. Fazit

In dieser Arbeit wurden verschiedene Ansätze genetischer Algorithmen für Losgrößenprobleme vorgestellt, die auf jeweils unterschiedlichen mathematischen Modellen basieren. Dabei können signifikante Unterschiede in der Kodierung der Produktionspläne zwischen den einzelnen Kategorien von Losgrößenproblemen festgestellt werden. Abhängig von der Wahl der Kodierung unterscheidet sich folglich die Berechnung der Fit-Werte sowie die Ausgestaltung der genetischen Operatoren. Durch einen hybriden Ansatz mit linearer/dynamischer Programmierung kann die Hürde von Einschränkungen in den Kapazitäten überwunden werden. Es ist festzuhalten, dass die genetischen Algorithmen für alle betrachteten Kategorien von Losgrößenproblemen im Allgemeinen qualitativ hochwertige Ergebnisse liefern, die sehr nah am Optimum liegen und daher gegenüber den vergleichbaren Heuristiken, die eine ähnliche Rechenzeit in Anspruch nehmen, zu bevorzugen sind. Besonders bei komplexen Problemen zeichnet sich eine deutliche Verbesserung in den Ergebnissen gegenüber den alternativen Verfahren ab.

Das Konzept der Genetischen Algorithmen, welches eine adäquate Entscheidungsunterstützung bei der Optimierung von Losgrößen darstellt, ist auch in der Lage in übergeordneten betriebswirtschaftlichen Fragestellungen einen ähnlichen Nutzen zu erweisen. Liegt ein kombinatorisches Problem vor, bei welchem durch die Ausprägung verschiedener Faktoren signifikante Unterschiede in den ökonomischen Auswirkungen der möglichen Kombinationen resultieren, eignen sich genetische Algorithmen zur effizienten Lösungsfindung. Bei der Wahl optimaler Produktionsstandorte werden aus vielen Regionen einige gewählt, welche unter Berücksichtigung der Faktoren (z.B. Mitarbeiterkosten, Steuern, Entfernung zu anderen Standorten, etc.) in Kombination zueinander sämtliche Märkte zu möglichst geringen (Transport-)Kosten bedienen. Bei der Gestaltung der Transportwege innerhalb einer Lieferkette liegt hin und wieder die Möglichkeit zur Kosteneinsparung durch die Wahl einer alternativen Route vor. Die Bestimmung einer optimalen Route sowie die Wahl von optimalen Produktionsstandorten kann mit genetischen

Algorithmen unterstützt werden. Auch strategische Fragen, z.B. welche Märkte zu welchen Zeiten betreten oder verlassen werden sollen, können hierdurch vereinfacht werden.

Aus diesen Gründen bietet der Einsatz von genetischen Algorithmen um das Management der Produktion bzw. der gesamten Lieferkette oder auch das strategische Management zu optimieren, ein umfangreiches Feld für weitere Forschungsaktivitäten.

Literaturverzeichnis

Dellaert, N.; Jeunet, J.; Jonard, N. (2000): A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs. In: *International Journal of Production Economics* 68 (3), S. 241–257.

Gaafar, Lotfi (2006): Applying genetic algorithms to dynamic lot sizing with batch ordering. In: *Computers & Industrial Engineering* 51 (3), S. 433–444.

Gelders, Ludo F.; van Wassenhove, Luk N. (1981): Production planning: a review. In: *European Journal of Operational Research* 7 (2), S. 101–110.

Guner Goren, Hacer; Tunali, Semra; Jans, Raf (2010): A review of applications of genetic algorithms in lot sizing. In: *Journal of Intelligent Manufacturing* 21 (4), S. 575–590.

Hernandez, W.; Suer, G. A. (1999): Genetic algorithms in lot sizing decisions. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC99. July 6-9, 1999, Mayflower Hotel, Washington, D.C., USA. Congress on Evolutionary Computation; Evolutionary Programming Society; IEEE Neural Networks Council; Institution of Electrical Engineers. New York: IEEE.

Holland, John H. (1992): Genetic Algorithms. In: *Scientific American* 267 (1), S. 66–72.

Khouja, Moutaz; Michalewicz, Zgibniew; Wilmot, Michael (1998): The use of genetic algorithms to solve the economic lot size scheduling problem. In: *European Journal of Operational Research* 110 (3), S. 509–524.

Li, Yongjian; Chen, Jian; Cai, Xiaoqiang (2007): Heuristic genetic algorithm for capacitated production planning problems with batch processing and remanufacturing. In: *International Journal of Production Economics* 105 (2), S. 301–317.

Meal, Harlan C.; Wachter, Manfred H.; Whybark, D. Clay (1987): Material requirements planning in hierarchical production planning systems. In: *International Journal of Production Research* 25 (7), S. 947–956.

Megala, N.; Jawahar, N. (2006): Genetic algorithm and Hopfield neural network for a dynamic lot sizing problem. In: *The International Journal of Advanced Manufacturing Technology* 27 (11-12), S. 1178–1191.

Prasad, P. S. S.; Krishnaiah Chetty, O. V. (2001): Multilevel Lot Sizing with a Genetic Algorithm Under Fixed and Rolling Horizons. In: *The International Journal of Advanced Manufacturing Technology* 18 (7), S. 520–527.

Srinivas, M.; Patnaik, L. M. (1994): Genetic algorithms: a survey. In: *Computer* 27 (6), S. 17–26.

Thomas, L. Joseph; McClain, John O. (1993): Chapter 7 An overview of production planning. In: *Handbooks in Operations Research and Management Science* 4, S. 333–370.

Thonemann, Ulrich; Albers, Marc (20]11): Operations Management. Konzepte, Methoden und Anwendungen. 2., aktualisierte Aufl., [Nachdr.]. München: Pearson Studium (Wi - Wirtschaft).

Toledo, Claudio Fabiano Motta; Oliveira, Renato Resende Ribeiro de; Morelato França, Paulo (2013): A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. In: *Computers & Operations Research* 40 (4), S. 910–919.

van Hop, Nguyen; Tabucanon, Mario T. (2005): Adaptive genetic algorithm for lot-sizing problem with self-adjustment operation rate. In: *International Journal of Production Economics* 98 (2), S. 129–135.

Whitley, Darrell (1994): A genetic algorithm tutorial. In: *Statistics and Computing* 4 (2), S. 65–85.

Xie, Jinxing; Dong, Jiefang (2002): Heuristic genetic algorithms for general capacitated lot-sizing problems. In: *Computers & Mathematics with Applications* 44 (1-2), S. 263–276.