Simulation in Produktion
und Logistik 2021
Jörg Franke & Peter Schuderer (Hrsg.)
Cuvillier Verlag, Göttingen 2021

Zuweisung bester Abstellplätze im LIFO-Lager unter Berücksichtigung der Kommissionierungsreihenfolge mittels genetischem Algorithmus

Allocation of best storage locations in the LIFO warehouse taking into account the picking sequence using genetic algorithm

Dominik Kuhn, Jan Adelsbach, Attique Bashir, Leenhard Hörauf, Rainer Müller,
ZeMA gGmbH, Saarbrücken (Germany), d.kuhn@zema.de, j.adelsbach@zema.de,
a.bashir@zema.de, l.hoerauf@zema.de, rainer.mueller@zema.de

Abstract: We present a practical approach for storage optimization of a LIFO (Last-In-First-Out) topology warehouse using complementary brute-force and genetic algorithms in a simulated environment using real-time data from the real-life counterpart in order to assist personal in allocation planning. Both the optimizations for placement and removal of storage units are being discussed.

1 Einleitung, Motivation und Ausgangssituation

Bodenlager dienen in der Intralogistik als Puffer zwischen beispielsweise einer Produktion und dem Warenausgang. Je nach Layout des Lagers und örtlichen Gegebenheiten unterliegen diese spezifischen Einlagerungs- und Auslagerungsprinzipien, wie bspw. FIFO (First-In-First-Out) oder LIFO (Last-In-First-Out). Je nach Produktionsmenge und verfügbarer Lagerkapazität kann eine manuell geplante Einlagerung mit Hinblick auf die Kommissionierung suboptimal sein. Bemerkbar macht sich dieses Problem, wenn eine Zugänglichkeit zu einem zu kommissionierenden Produkt (Palette) nicht ohne weiteres ermöglicht wird.

Getränkehersteller produzieren ihre Produkte in der Regel mit einer bestimmten Vorlaufzeit und organisieren ihre Lagereinheiten in Paletten. Dabei wird eine Lagereinheit an dem Zeitpunkt der Herstellung nicht zwangsläufig mit einem Kommissionsauftrag in Verbindung gebracht (Keine Auftragsspezifische Produktion). Bei einer zufälligen (ungünstigen) Positionierung von Paletten können bei der Kommissionierung Aufträge einander den Zugang blockieren, wenn bspw. ein später platzierter Auftrag später kommissioniert werden soll. Daher werden Abstellplätze durch einen Mitarbeiter der Lagerlogistik so ausgewählt, dass Paletten sich entsprechend der geplanten Kommissionssequenz nicht blockieren. Gerade bei hohen Produktionsmengen ist die Erstellung eines Belegungsplans unüberschaubar

und für den Lagerleiter nicht mehr zuverlässig zu erstellen. Dies führt zu vermehrten Blockaden im Lager. Folglich ist der Lagerist mit Umplatzierungsaufwänden konfrontiert, sodass eine weitere unplanmäßige Umplatzierung zu weiterer Eskalation führt.

Um die Umplatzierungsaufwände für den Lageristen zu verringern, wird im vorliegenden Paper eine Methode beschrieben, die bei bekannter Komissionierungsreihenfolge einer oder mehreren fertiggestellten Paletten einen Platz zuweist. Zur Anwendung kommt hierbei als Metaheuristik einen genetischen Algorithmus, der zwar nicht die optimale Lösung garantiert, aber meistens zu einer annehmbaren Lösung kommt, da dieser die Lösung gezielt hinsichtlich einer Bewertungsfunktion zu minimieren versucht.

2 Stand der Technik

Die Zuweisung bester Abstellplätze ist ein bekanntes Problem, dem unter dem Begriff „Storage Location Assignment Problem - SLAP“ nachgegangen wird. (Kofler et al., 2010; Battista et al., 2011) Gängige Methoden versuchen das Problem mathematisch zu beschreiben und mittels Linearer Programmierung (LP) zu lösen. Weitere Ansätze nutzen Metaheuristiken, wie die genetischen Algorithmen oder Simulated Annealing, um beste Lagerplätze zu ermitteln. Die Anwendung des Kuhn-Munkres Algorithmus ist ebenfalls eine Optimierungsmethode zur Zuweisung von Produkt oder Bauteil auf Lagerplätze. Viele Arbeiten berücksichtigen jedoch nicht explizit die Restriktionen beim Einlagern in Lager mit einem LIFO Betriebskonzept. Beim LIFO (Last-In-First-Out) Prinzip können Paletten nur in der umgekehrten Einlagerreihenfolge wieder ausgelagert werden.

3 Lagerbeschreibung

Der Typ des Fertiglagers im betrachteten Fall (siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**) wird als Bodenlager in Blockform bezeichnet. Lagereinheiten, sind in diesem Fall Paletten, die direkt und lückenlos nebeneinander, hintereinander und paarweise übereinandergestapelt werden. Der Zugriff auf die jeweiligen Lagereinheiten folgt dem LIFO-Prinzip, sodass einzelne Reihen (Spalten) eines Blocks nur in einer Richtung befüllt oder geleert werden können. Reihenübergreifend lässt das Lager beliebige Abstellplätz zu. Einzelne Reihen oder Blöcke sind in der Regel rechtwinklig oder auch schräg zur eigentlichen Fahrgasse angeordnet.

Bei dem Kommissionierungslager handelt es sich ebenfalls um ein Bodenlager in Blockform. Hier werden Lagereinheiten auftragsbezogen für die jeweiligen Kunden zusammengestellt. Der Lagertyp des Kommissionierungslagers orientiert sich in Grundzügen an dem FIFO-Prinzip. Aus dem Fertiglager entnommene Lagereinheiten werden ausschließlich in einer Richtung in den jeweiligen Reihen gestapelt, wobei die Beladung der LKWs aus genau entgegengesetzter Richtung erfolgt.

Softwaretechnisch lassen sich die beiden Lager tabellenförmig mittels Matrizen modellieren. Die jeweiligen Einträge spiegeln die reale Belegung der jeweiligen Lagerplätze wieder, ein Nulleintrag steht für einen unbelegten Platz. Die Kopplung der Einträge mit den dazugehörigen Varianten sowie Länderkennung stellt ein

notwendiges eindeutiges Unterscheidungsmerkmal dar, welches bei der Bestimmung der Lagerplätze von großer Bedeutung ist. Im konkreten Anwendungsfall wird jeder Eintrag durch einer sogenannten Paletten-ID beschrieben. Analog zu dem realen Fertiglager werden die Reihen der Matrix durch eingehende Aufträge von vorne befüllt. Lagereinheiten, die zur Kommission bereitgestellt werden, werden durch eine Entnahme zuerst mit einem Nulleintrag von vorne in den Spalten versehen. Ähnlich verhält es sich mit der softwaretechnischen Beschreibung des Kommissionierungslagers. Die Modellierung des Kommissionierungslagers spielt im konkreten Optimierungsproblem eine untergeordnete Rolle, aus diesem Grund wird in diesem Paper auf diesen Lagerteil im Folgenden nicht näher eingegangen.

4 Problemformulierung

Sei R ein Lager mit N heterogenen Reihen $R = \langle R_1, R_2, \dots, R_n \rangle$ bestehend aus Lagereinheiten. Letztere sind definiert als $\mathrm{p} \in \mathbb{N} \cup \{\emptyset\}$, welches entweder zu einer Paletten-ID oder $\emptyset$ als nicht-besetzt entspricht. Der Produkttyp einer Palette $p \neq \emptyset$ ist definiert als $P_t(p)$, das Herstellungsdatum als $P_d(p)$. Als Kurzhandform wird für eine Lagereinheit in Reihe n und Tiefenposition m auch $p_{n,m}$ verwendet.

Bei der Einlageroptimierung sollen zu allen Paletten in einer gegebenen Liste E die bestmöglichen Positionen nach einem Bewertungsschema gefunden werden. Zur Bewertung der Lösung wird bei der Einlageroptimierung der kumulativ höchste normierte Wert für das gesamte Lager $s = \frac{1}{|R|}\sum_i^N s_i$ aus der Summe von mehreren im späteren beschriebenen normierten Bewertungsverfahren B für jede Reihe $s_n = \sum_{b \in B} s_{b,n}$ gesucht.

Bei der Auslageroptimierung sollen für eine Kommissionierungssequenz O passende Palettenpositionen im Lager gefunden werden. Hierbei wird versucht die Anzahl von Umlagerungen für das gesamte Lager zu minimieren.

Die Optimierung soll anhand des tatsächlichen Lagerbestandes simulieren, wie die aktuell produzierten Paletten am besten eingelagert werden können bzw. welche Paletten des Lagers ausgelagert werden sollen. Das System soll dabei den Lagerplaner in seiner Arbeit unterstützen. Die Optimierung wird dabei über eine Schnittstelle aus dem proprietären ERP-System des Getränkeherstellers heraus aufgerufen. Als Laufzeitumgebung wurde die Python Programmiersprache gewählt und zur Umsetzung der Algorithmik genutzt.

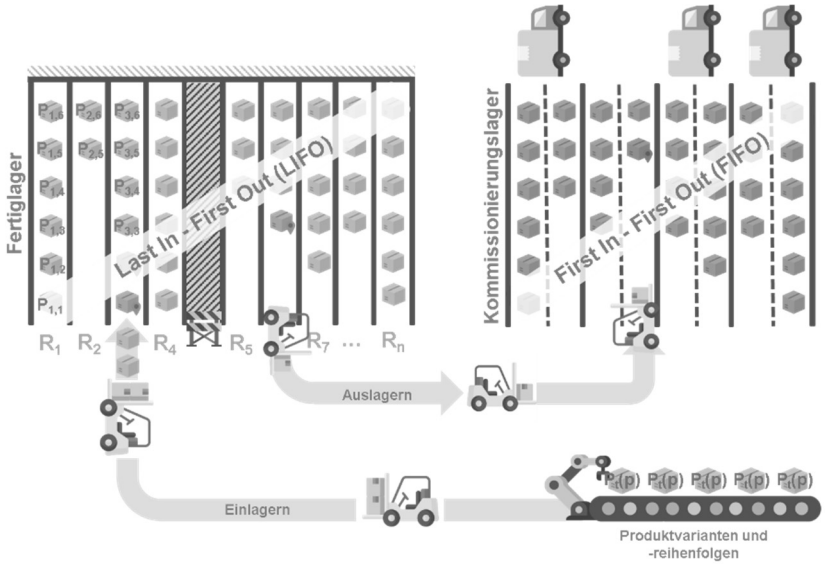

Abbildung 1: *Lagermodell und Problembeschreibung*

5 Genetischer Algorithmus

5.1 Biologisches Vorbild sowie Grundidee

Der genetische Algorithmus stellt eine Metaheuristik aus der Klasse der evolutionären Optimierungsverfahren dar. Dabei orientiert sich dieser Algorithmus an der Evolutionstheorie von Charles R. Darwin. Nach Darwin erfolgt im Zuge der biologischen Evolution eine natürliche Selektion von Individuen mit Eigenschaften, die diesem eine höhere Überlebenschance bieten. Diese Art von Optimierung wurde ursprünglich unter anderem in Turing (1950) beschrieben.

Analog dazu werden bei genetischen Algorithmen Populationen von Individuen betrachtet und somit eine Vielzahl von möglichen Lösungen in einem gegeben Suchraum. Diese Individuen werden in Anlehnung zur Genetik häufig auch als Chromosomen bezeichnet. Eine Stelle oder Sequenz eines Chromosoms, also eine Entscheidungsvariable des betrachteten Problems, wird wiederum als Genom bezeichnet. Für die Charakterisierung der Lösungsgüte bezüglich der betrachteten Problemstellung wird die sogenannte Fitnessfunktion betrachtet. Diese beschreibt die Güte eines Individuums mit einem reellen Wert.

Entsprechend der biologischen Mechanismen der Kreuzung, Mutation und der Selektion wird auch bei den genetischen Algorithmen verfahren, um eine Weiterentwicklung der Population der Individuen zu ermöglichen.

5.2 Aufbau des genetischen Algorithmus

Der moderne genetische Algorithmus entspricht den Konzepten, die von Bremermann beschrieben wurden, siehe Anderson und Conrad (1995) und besteht grundlegend aus den folgenden Phasen, die dieser teilweise iterativ durchläuft:

1. Initialisierung einer zufälligen Ausgangspopulation
2. Errechnen der Fitness für jedes Individuum
3. Selektion
4. Rekombination/ Crossover
5. Mutation
6. Wiederholung ab Schritt 2, bis ein Abbruchkriterium erfüllt ist.

Die initiale Kodierung der Individuen beim genetischen Algorithmus wird häufig in Form von Permutationen vorgenommen. Beispielsweise bei der Ermittlung eines optimalen Weges, indem verschiedenen Wegpermutationen, über einer Fitnessfunktion, Zielkriterien, wie zum Beispiel der Zeit, zugeordnet werden. Für eine zufällig initialisierte Startpopulation wird anschließend über eine geeignete Fitnessfunktion die Güte der einzelnen Lösungen bestimmt. Die Selektion dient dazu, aus der Population die vermeintlich fittesten Individuen zur Übernahme in die Folgepopulation auszuwählen. Dabei gibt es verschiedene Methoden. Der Großteil der Methoden haben gemein, dass sie bei der Wahl der fittesten Individuen zusätzlich eine stochastische Komponente berücksichtigen. Die bekanntesten Selektionsmethoden sind (Mehmeti und Amrein et al., 2014):

- Monte-Carlo Selektion (Roulette-Wheel-Selektion)
- Rangbasierte Selektion
- Tournament-Selektion

Eine weit verbreitete Methode ist die Tournament-Selektion. Dabei werden zufällige Individuen ausgewählt und jenes mit dem höchsten Fitness-Wert von diesen selektiert. Dabei können die Selektionen auch Elitär verfahren, wobei nur die Individuen mit dem höchsten Fitness-Wert einbezogen werden. Aus zwei selektierten Individuen werden bei der Rekombination durch das sogenannte Crossover neue Individuen generiert. Aus den neu entstandenen Individuen werden mit einer geringen Wahrscheinlichkeit Kandidaten für Mutationen ausgewählt, um die Exploration des Suchraums zu stärken. Bei der Mutation wird ein zufälliges Allel geändert. (Mehmeti und Amrein et al., 2014) Für die Zusammensetzung der neuen Generation werden wieder insgesamt N Individuen aus der alten Generation und den neuen Individuen ausgewählt.

6 Umsetzung der Algorithmik

6.1 Einlageroptimierung

6.1.1 Bewertungsverfahren

Für die Einlageroptimierung werden mehrere Bewertungsverfahren $s_{t,n}, s_{l,n}, s_{d,n}, s_{a,n}$ und $s_{u,n}$ für eine Reihe n (normierte Werte $[0,1]$) verwendet, welche folgend im Detail abgehandelt werden, diese werden anhand von definierbaren Koeffizienten c_k summiert

$$s_n = \frac{1}{5}\left(c_t s_{t,n} + c_l s_{l,n} + c_d s_{d,n} + c_a s_{a,n} + c_u s_{u,n}\right) \quad (1)$$

Die kumulative Bewertung für das gesamte Lager ist ein wie folgt abgeleiteter Normierter Wert $s = \frac{1}{N}\sum_i^N s_i$ dieser wird zu maximieren versucht.

Das Typen-Clustering leitet die Typenreinheit anhand der Anzahl von Paletten k ab, die den gleichen Typ haben und zu Beginn der Reihe stehen, sowie die Anzahl der restlichen Paletten in der Reihe l:

$$s_{t,n} = \frac{max(0, k-1) - l}{|R_n|} \tag{2}$$

Gegeben das arithmetische Mittel der Produktionsdaten aus allen besetzten Lagerplätzen in einer Reihe, $\overline{P_{d,n}} = min(V_n) + \frac{1}{2}[min(V_n) + max(V_n)]$ für $V_n = \{P_d(p) : p \in R_n \wedge p \neq \emptyset\}$, sowie das arithmetische Mittel über das gesamte Lager $\overline{P_d} = \frac{1}{N}\sum_i^N \overline{P_{d,i}}$ wird der Produktionsdaten-Clustering Bewertung als:

$$s_{d,n} = max\left(0, \frac{\overline{P_{d,n}} - \overline{P_d}}{\overline{P_{d,n}}}\right) \tag{3}$$

errechnet. Beim Leer-Clustering wird errechnet, wie viele Paletten fehlen, um eine gegebene Belagerung c_u zu erreichen. Hierbei wird gezählt, wie viele Paletten in der Reihe frei sind in Relation zu der Anzahl von Lagerplätzen in der Reihe:

$$s_{l,n} = |\{z \in R_n \mid z = \emptyset\}| \,/\, |R_n| - c_u \tag{4}$$

Der Belagerungskoeffizient kann entweder manuell definiert werden, oder für das gesamte Lager kann durch das arithmetische Mittel der einzelnen Reihenbelegungen errechnet werden:

$$c_u = \frac{1}{N}\sum_i^N s_{l,i}\Big|_{c_u=0} \tag{5}$$

Beim Auftrags-Clustering wird anhand einer Kommissionssequenz O gezählt, wie viele Paletten in der aktuellen Reihe auf diese passen, bzw. nicht passen. Die Reihenfolge der Paletten in der Kommission spielt hierbei keine Rolle, da es in den gegeben Einsatzszenario irrelevant ist, in welcher Reihenfolge diese auf Kommissionsplätze verschoben werden.

$$s_{a,n} = |\{z \in R_n \mid : z \in O\}| \,/\, |R_n| \tag{6}$$

Hierbei wird jedes Element in O nur maximal einmal gezählt. Bei der Umlagerbewertung wird, gegeben eine Liste an auszulagernden Paletten $a_{n,k} \in A$ errechnet, wie viele Umlagerungen benötigt werden, um jede von diesen Paletten zu erreichen.

$$s_{u,n} = 1 - max|\{z \mid z_m \in R_n \wedge m < k\}| \,/\, |R_n| \tag{7}$$

Hierbei werden, wenn Paletten des gleichen Typs an einer vorherigen Position in der Reihe auftreten jedoch nicht zum Auslagern markiert sind, diese anstatt der eigentlich markierten Paletten in A zum errechnen dieser Bewertung genommen.

6.1.2 Brute-Force Methode

Bei $|E| = 1$ wird eine Brute-Force Methode genutzt. Dabei werden die beschriebenen Bewertungen für alle Reihen errechnet, und die Reihe mit der höchsten Bewertung herausgesucht. Sollten mehrere diesen teilen, so wird eine zufällige dieser ausgewählt.

6.1.3 Genetischer Algorithmus

Für das Einlagern von mehreren Paletten $|E| > 1$ hingegen stellt sich das Einlagern als Kombinatorisches Problem mit einer Anzahl von Möglichkeiten nach dem Binomialkoeffizienten da. Dieses Problem wird mit einem genetischen Algorithmus gelöst.

Zuerst werden alle freien Lagerplätze im Lager in einen Vektor T herausgesucht, dieser enthält die Indices von Reihen mit freien Lagerplätzen. Dabei repräsentiert das mehrfache Auftreten eines Index eine entsprechende Anzahl von freien Lagerpositionen in dieser Reihe, diese sind jedoch als Optimierung auf die Anzahl von einzulagernden Paletten limitiert $min(|E|, |\langle z \in R_n \mid z = \emptyset \rangle|)$, siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**.

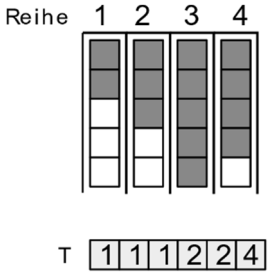

Abbildung 2: *Generation des Positionsvektors T*

Der Individuums-Vektor I welcher dieselbe Anzahl von Elementen wie T enthält und mit diesem korrespondiert, enthält die Paletten-IDs der einzulagernden Paletten oder $\emptyset$. Hierbei tritt jede Paletten-ID nur einmal auf. Zum Rückwärtsauflösen wird der Individuen-Vektors I sequentiell durchlaufen, leere Elemente übersprungen und die entsprechenden Paletten in die Reihe entsprechen des Indexes im Vektor T eingelagert, siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Für das Initialisieren, werden in einem leeren Vektor entsprechend der Anzahl von einzulagernden Paletten zufällige Positionen gewählt und die Paletten diesen zugeordnet. Dabei kann optional eine strikte Reihenfolge der Paletten eingehalten werden. Beim Mutieren wird zufällig eine von drei Methoden, Twors, RSM (Reverse-Sequence-Mutation) und CIM (Center-Inversion-Mutation) wie beschrieben unter anderem in Abdoun et al. (2012) mit einer gleichen Wahrscheinlichkeit gewählt. Bei der Twors Mutation werden zufällige Positionen vertauscht. Bei der RSM, entsprechend Abdoun et al. (2012) werden zwei Punkte $1 \leq a \leq b \leq |I|$ per Zufall gewählt und alle Elemente zwischen diesen in der Reihenfolge umgekehrt. Bei der CIM wird ein Punkt $1 \leq a \leq |I|$ zufällig gewählt und die Subsequenzen davor und danach in der Reihenfolge umgekehrt.

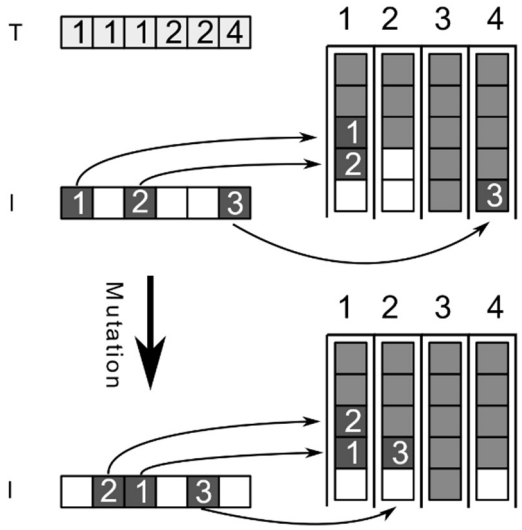

Abbildung 3: *Rückwärtsauflösung der Vektoren I und T*

Die Rekombination wird durch einen Ordered Crossover (OX) nach Abdoun und Abouchabaka (2012) durchgeführt. Um eine Standard-Implementierung der OX Rekombination zu ermöglichen, werden alle nicht gesetzten Werte der beiden Eltern-Individuen durch absteigende negative Integer ersetzt, sodass alle Elemente distinktive Werte beinhalten. Letzteres setzt voraus, dass es keine negativen Paletten-IDs gibt. Zur Selektion wird eine Tournament Selektion wie in Goldberg und Deb sowie Blickle und Thiele (1996) benutzt. Der Fitnesswert wird berechnet, in dem die Paletten des Individuums-Vektor auf das Lager temporär angewendet werden und die bereits beschriebene Bewertung für das gesamte Lager s errechnet wird. Hierbei werden die Bewertungen für einzelne Reihen nur neu errechnet.

6.2 Auslageroptimierung

Bei der Auslagerungsoptimierung wird jeder Eintrag o_k auf einer gegebenen Kommissionierungssequenz $O = \langle o_1, o_2, \dots, o_k \rangle$ mit der Position einer Palette des entsprechenden Typs im Lager zu befüllt. Hierbei gilt es die Positionen der Paletten so zu wählen, dass diese mit möglichst wenig Umlagerungen aufgegriffen werden können. Hierbei kann zwischen „direkt“ und „indirekt“ freien Paletten, sowie „blockierten“ Paletten unterschieden werden: Direkte Paletten stehen in erster Position einer Reihe. Indirekte Paletten werden durch andere Paletten, die ebenfalls in der Kommissionierungssequenz zur Auslagerung zugeordnet worden freigelegt, $p_{n,m}$ wenn $\{p_{n,1}, p_{n,2}, \dots, p_{n,m-1}\} \subset O$. Blockierte Paletten werden durch andere Paletten blockiert, die zwingend umgelagert werden müssen, $\exists p \notin O$ für $p \in \{p_{n,1}, p_{n,2}, \dots, p_{n,m-1}\}$.

6.2.1 Brute-Force Methode

Bei der Brute-Force Methode wird die gegebene Kommissionierungssequenz sequentiell durchlaufen und es wird versucht für jeden geforderten Palettentyp eine direkt- oder indirekt freistehende Palette des Lagers zu finden. Um indirekt

freistehende Paletten zu finden, werden bereits zugewiesene Paletten aus dem Lager entfernt. Paletten, die nicht auf diese Weise zugewiesen werden können $O_f \subseteq O$ und dadurch blockiert sind werden daraufhin mit dem im Folgenden beschriebenen genetischen Algorithmus ausgewählt bzw. optimiert.

6.2.2 Genetischer Algorithmus

Sei O_f das Subset an nicht direkt- oder indirekt freistehende Paletten und gegeben das Lager ohne die bereits durch den Brute-Force Algorithmus zugewiesenen Paletten. Hierbei wird versucht die Anzahl von Umlagerungen s_v zu minimieren. Diese ist hierbei die Summe der Maxima der Tiefenpositionen von den zu auslagernden Paletten im Individuums-Vektor I definiert.

$$s_{v,n} = \sum_{i}^{N} max\{m : p_m \in I \wedge p_m \in R_i\} - 1 \tag{8}$$

Bei der Mutation wird ein zufälliges Element des Individuums-Vektors $x \in I$ gewählt. Danach wird aus einer Liste von Paletten selbigen Typs, die sich im Lager befinden, jedoch nicht bereits an einer anderen Stelle im Individuen-Vektors zugewiesen sind $y \in \{p \in P \mid P_t(p) = P_t(x) \wedge \nexists z \in I\}$ eine zufällige gewählt und die Position von dieser mit x vertauscht. Sollten sich keine weiteren Paletten eines Typs im Lager befinden, so wird sofern vorhanden, ein weiteres Element des Individuum-Vektors vom gleichen Typ ausgewählt, $y \in \{p \in I : P_t(p) = P_t(x)\}$ und das Element x mit diesem vertauscht. Für die Rekombination werden die beiden Eltern-Vektoren I_1, I_2 in Sub-Vektoren nach Palettentyp t partitioniert $I_{k,t} = \{z \mid z \in I_k \wedge z \in P_t\} \forall t$. Dies ist nötig, da nur Paletten des gleichen Typs rekombinierbar sind. Die beiden Sub-Partitionen jedes Typs der Eltern $I_{1,t}, I_{2,t}$ werden daraufhin mit einem Cyclic Crossover (CX) wie beschrieben in Abdoun und Abouchabaka (2012) rekombiniert. Im Anschluss werden die rekombinierten Sub-Partitionen, entsprechend der originalen Indexe zurück zu zwei Kinder-Vektoren zusammengefügt. Für den Fitness-Wert, wird die oben beschriebene Kosten-Funktion $s_{v,n}$ für das Lager entsprechend der Paletten im Individuums-Vektor errechnet.

7 Diskussion und Ausblick

Die vorgestellte Einlageroptimierung sucht eine ideale Position von einzulagernden Paletten anhand des Ist-Zustandes des Lagers und der Kommissionsequenz zu dem Zeitpunkt der Optimierung. Dabei werden in diese Optimierungsstrategie lediglich die Anzahl der Paletten einbezogen, welche sich im Puffer zwischen Abfüllung und Lager befinden. Zukünftige Herstellungs- und Kommissionsequenzen können durch diese Methode nicht berücksichtigt werden, was eine gewisse Limitierung des Optimierungspotentials darstellt. Der Einsatz von probabilistischen Modellen könnten an dieser Stelle das gegebene Potential um eine gewisse Vorausschau erweitern. Die Verwendung von reihenbasierenden Bewertungskriterien ist limitiert in der Anzahl von Metriken, die abgeleitet werden können. Mit Hilfe von Machine Learning (ML) Algorithmen könnte zudem der Lagerbestand in einem definierten zukünftigen Zeitraum im Voraus optimiert werden.

Die Auslagermethode ist limitiert, da sie nur versucht definierte Aufträge in einer definierten Reihenfolge zu vollfüllen. Ein weiterer Ansatz hierbei könnte sein die Kommissionierungssequenz selber anhand von dem Ist-Zustandes des Lagers, der Verfügbarkeit von Kommissionierungsplätzen und anhand von den Auslieferzeiten zu optimieren. Dabei könnten Kommissionierungen, die zeitlich verschiebbar sind so umgeplant werden, dass es fast nie zu Umlagerungen kommt. Der Ansatz, zuerst die Paletten anhand von einer Brute-Force Methode zuzuweisen und nur die nicht durch diese Methode zuweisbaren Paletten durch einen GA zu optimieren ist aufgrund von Zykluszeiten getroffen worden. Dies hat jedoch den trade-off, dass der GA nur die bestmöglichste Lösung relativ zu dem Ergebnis der Brute-Force Methode findet. Ein anderer Ansatz hierbei wäre es die gesamte Auslageroptimierung durch den GA zu optimieren.

Aktuell befindet sich der beschriebene Ansatz in der Validierung im laufenden Produktivbetrieb. Hierbei wird observiert ob dieser Vorteile gegenüber der manuellen Planung hervorbringt. Offline Simulationen mit ca. 1/5 so großen Lagerdaten, wie im konkreten Anwendungsfall, zeigten deutlich die Funktionsweise dieser Methode in der Einlagerungsoptimierung.

Literatur

Abdoun, O.; Abouchabaka, J.: A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem (2012).

Abdoun, O.; Abouchabaka, J.; Tajani, C.: Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem. IJES (2012).

Anderson, R.W.; Conrad, M.: Hans J. Bremermann: A pioneer in mathematical biology. Biosystems 34 (1995) 1-3, S. 1–10.

Battista, C.; Fumi, A.; Giordano, F.; Schiraldi, M.: Storage Location Assignment Problem: implementation in a warehouse design optimization tool. In: Breaking down the barriers between research and industry, Padua (Italy), 14-16.09.2011, 2011,

Blickle, T.; Thiele, L.: A Comparison of Selection Schemes Used in Evolutionary Algorithms. Evolutionary Computation 4 (1996) 4, S. 361–394.

Goldberg, D.E.; Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In: FOGA 1990, S. 69–93.

Kofler, M.; Beham, A.; Wagner, S.; Affenzeller, M.; Reitinger, C.: Reassigning Storage Locations in a Warehouse to Optimize the Order Picking Process. In: 22nd European Modeling and Simulation Symposium EMSS 2010, Morocco, 13.10.2010, 2010,

Mehmeti, Q.; Amrein, V.; Kulms, M.; Macin, N.; Loenser, C.; Bogonos, D.; Waidhas, M.: Genetische Algorithmen: Wirtschaftsmathematisches Projekt zur Numerik. Hg. v. Technische Universität Dortmund, Fakultät für Mathematik; Angewandte Mathematik und Numerik Dortmund, 2014,

Turing, A.M.: I.—COMPUTING MACHINERY AND INTELLIGENCE. Mind LIX (1950) 236, S. 433–460.