Simulation in Produktion und Logistik
Entscheidungsunterstützung von der Planung bis zur Steuerung
Wilhelm Dangelmaier, Christoph Laroque & Alexander Klaas (Hrsg.)
Paderborn, HNI-Verlagsschriftenreihe 2013

Simulationsbasierte Reihenfolgeoptimierung von Fertigungsaufträgen für Montagesysteme mittels eines genetischen Algorithmus unter Beachtung von Reihenfolgebeschränkungen

Simulation-based sequence optimization for assembly systems using a genetic algorithm and considering line constraints

Marco Lemessi, John Deere GmbH & Co. KG, Mannheim (Germany),
LemessiMarco@JohnDeere.com
Thomas Schulze, Christian Tänzer, Otto-von-Guericke-Universität Magdeburg, Magdeburg (Germany), tom@iti.cs.uni-magdeburg.de, christian.taenzer@st.ovgu.de

Abstract: Sequence optimization is a well-known problem in planning the production of mixed model assembly systems. Simulation is often used to describe the structure and the behaviour of these systems. This paper presents a simulation-based algorithm for the sequence optimization of mixed model assembly systems. In multiple optimization runs a genetic algorithm is executed to improve a start sequence of jobs regarding to the cycle time of this sequence. In addition, the optimization algorithm considers line constraints limiting the order of jobs. With these constraints layout and technical aspects of assembly systems are taken into account using Boolean algebra for the definition.

1 Einleitung

Kürzer werdende Produktlebenszyklen sowie eine steigende Kundennachfrage nach mehr Produktvarianten erfordern eine stetige Erweiterung des Produktportfolios eines Unternehmens. So werden auf einer Montagelinie mehrere Varianten eines Produkts produziert (Bukchin et al. 2002). In diesem Zusammenhang ist das Problem der Reihenfolgeplanung von Fertigungsaufträgen, mit dem Ziel eine zeitliche Abfolge der Aufträge für eine gegebene Zielstellung zu bestimmen eine bekannte Aufgabe. Mögliche Zielstellungen sind beispielsweise die Minimierung der Zykluszeit oder die Minimierung der Abweichungen vom Fälligkeitszeitpunkt (Kiener 2006; Pinedo 2012; Zahn und Schmid 1996). Analytische Lösungsverfahren für das Reihenfolgeplanungsproblem liefern optimale Lösungen, jedoch besitzen diese Lösungen, aufgrund von Annahmen zur Vereinfachung, keine Relevanz in realen Produktionssystemen. Aus diesem Grund werden vielfach Heuristiken, zum

Beispiel Tabu-Suche, Simulated Annealing oder genetische Algorithmen verwendet, die in annehmbarer Zeit gute Ergebnisse liefern, ohne aber das Optimum der Lösungen zu gewährleisten (Völker und Schmidt 2010; Zahn und Schmid 1996). Diese Heuristiken werden oftmals mit der Simulation kombiniert, um verschiedene Parameterkonfigurationen des Systems zu evaluieren (Völker und Schmidt 2010).

Für das Problem der Reihenfolgeplanung wird vielfach der genetische Algorithmus aus der Klasse der Heuristiken zur Optimierung herangezogen. In Chen et al. (1995), Iltzsche et al. (2011), Rotondo et al. (2012) sowie in Yasin et al. (2010) wurde gezeigt, dass der Einsatz dieses Algorithmus gute Ergebnisse liefert. Jedoch wird der genetische Algorithmus zur Verbesserung des Ergebnisses nicht mehrmals angewendet.

Oftmals müssen im Zusammenhang mit der Planung von Reihenfolgen gegebene Beschränkungen aus den zu optimierenden Systemen berücksichtigt werden. Hierzu gehören Terminfälligkeiten von Fertigungsaufträgen, die sich aus Kundenanforderungen ergeben. Aus diesen Fälligkeiten lassen sich Bearbeitungsprioritäten ableiten. So haben Fertigungsaufträge mit einer kurzen Terminfrist eine höhere Priorität als Aufträge mit einer größeren Terminfrist. Für diese Art von Restriktionen existieren bereits zahlreiche Lösungsansätze in Form von Reihenfolgeplanungsproblemen mit terminbezogenen Zielgrößen, wie beispielsweise die Minimierung der Verspätungen (Jeffcoat und Bulfin 1993; Mattfeld und Bierwirth 2004). Andere Beschränkungen sind im Layout und in technischen Aspekten eines Montagesystems begründet (Jones und Wilson 1996). Weitere Ursachen für Reihenfolgebeschränkungen sind zum Beispiel spezielle Operationen oder Tätigkeiten sowie Lieferanten-Beziehungen (Jones et al. 1997). Beispiele für derartige Restriktionen sind (Lackes und Awiszus 2012):

- Direkt nach Auftrag A darf kein Auftrag B folgen,
- Zwei Aufträge vom Typ A dürfen nicht direkt aufeinander folgen oder
- Zwischen zwei Aufträgen vom Typ A müssen Aufträge vom Typ B und C vorhanden sein.

Die letztgenannte Art von Reihenfolgebeschränkungen wurde bisher nur unzureichend bei der Reihenfolgeplanung in Verbindung mit Heuristiken berücksichtigt.

Zur Verbesserung der Ergebnisse bei der simulationsbasierten Reihenfolgeoptimierung für Montagesysteme wurde ein neuer Lösungsansatz entwickelt, der charakterisiert ist durch die mehrmalige Anwendung des genetischen Algorithmus, der Verwendung einer booleschen Algebra zur Definition von Restriktionen für Reihenfolgen und die Beachtung von Reihenfolgebeschränkungen bei der Generierung von Reihenfolgen.

In den folgenden Kapiteln wird beginnend die Zielfunktion zur Reihenfolgeoptimierung in Montagesystemen definiert. Anschließend wird der entwickelte Optimierungsalgorithmus erläutert und die Beschreibung von Restriktionen mittels boolescher Algebra aufgezeigt. Die Implementierung des Ansatzes in ein existierendes generisches Simulationsmodell für Montagelinien sowie die Anforderungen der Nutzer werden in einem weiteren Kapitel fokussiert. An einem Anwendungsfall werden anschließend die erreichten Verbesserungen aufgezeigt. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick.

2 Optimierungsproblem

Das vorliegende Problem der Reihenfolgeplanung ist ein Flow-Shop-Problem mit N Fertigungsaufträgen für eine Montagelinie über einen bestimmten Planungszeitraum. Als Optimierungsziel wird die Minimierung der Zykluszeit der zu montierenden Produktreihenfolge definiert (s. Gl. 1). Zur Bestimmung der Zykluszeit einer Reihenfolge wird diese Auftragsreihenfolge mit dem Simulationsmodell simuliert, die Ergebnisdaten gesammelt und ausgewertet. Restriktionen bezüglich der Reihenfolge werden als Nebenbedingung berücksichtigt (s. Gl. 2). Hierbei müssen alle gegebenen Reihenfolgebeschränkungen erfüllt werden. Zusammengefasst lässt sich das Optimierungsproblem formal wie folgt beschreiben:

Variablendeklaration:

L	Anzahl von Reihenfolgen (k = 1,…,L)
N	Anzahl von Fertigungsaufträgen (i = 1,…,N)
R	Anzahl von Reihenfolgerestriktionen (r = 1,…,R)
Seq_k	Reihenfolge k mit k = 1,…,L
C_r	Reihenfolgebeschränkung r mit r = 1,…,R
CT	Zykluszeit
$Z(Seq_k)$	Zykluszeit von Reihenfolge k mit k = 1,…,L
$\Phi(C_r,Seq_k)$	Funktion zur Bestimmung, ob Reihenfolge k die Restriktion r erfüllt mit k = 1,…,L und r = 1,…,R

Zielfunktion:

$$\text{Min} \rightarrow CT = \min_{k=1}^{L} Z(Seq_k) \tag{1}$$

Nebenbedingungen:

$$\sum_{r=1}^{R} \phi(C_r, Seq_k) = R, \forall k = 1,..., L \tag{2}$$

mit

$$\Phi(C_r, Seq_k) = \begin{cases} 1, \text{wenn Reihenfolge k die Restriktion r erfüllt} \\ 0, \text{sonst} \end{cases}$$

$L \in \mathbb{N}^+$; $N \in \mathbb{N}^+$; $R \in \mathbb{N}$; $CT \in \mathbb{R}^+$; $Z(Seq_k) \in \mathbb{R}^+$; $\Phi(C_r,Seq_k) \in \{0,1\}$

3 Optimierungsalgorithmus

Der Optimierungsalgorithmus absolviert mehrere Optimierungsläufe, bis keine signifikante Verbesserung der Zielgröße erzielt oder eine vorgegebene Laufzeitbeschränkung erreicht wurde. Eine Verbesserung bedeutet in diesem Kontext eine geringere Zykluszeit als die Auftragsstartreihenfolge des aktuellen Optimierungslaufs. In jedem Optimierungsdurchlauf wird der genetische Algorithmus, basierend auf einer neuen Startreihenfolge von Aufträgen ausgeführt. Die optimierte Reihenfolge des vorherigen Laufs geht als Startreihenfolge in den genetischen Algorithmus des nächsten Optimierungsdurchlaufs ein. Hierbei dient

die Startreihenfolge als Ausgangspunkt für die Bildung neuer Auftragsreihenfolgen innerhalb des genetischen Algorithmus. Als Eingabegrößen für den Algorithmus werden das Simulationsmodell der Montagelinie, eine existierende Startreihenfolge der Fertigungsaufträge sowie mögliche Reihenfolgebeschränkungen verwendet (s. Abb. 1). Für alle zu simulierenden Reihenfolgen wird das Simulationsmodell mit einem identischen Startzustand initialisiert. Diese Initialisierung bildet eine realitätsnahe Belegung der Montagelinie als Ausgangssituation ab.

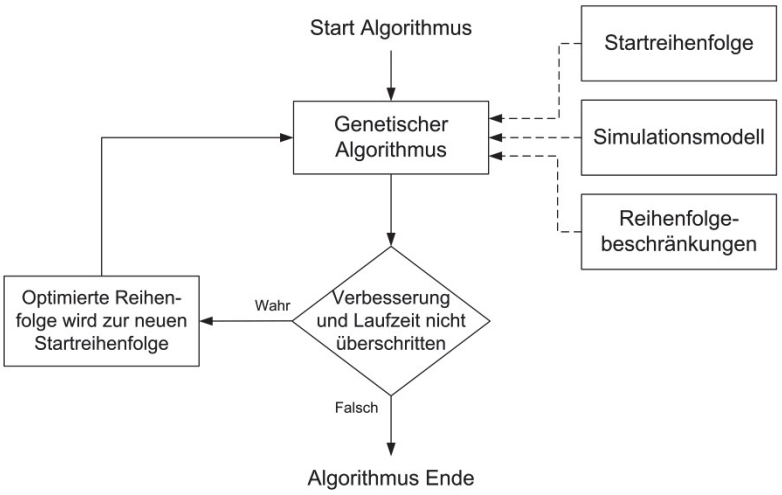

Abbildung 1: *Schematischer Ablauf der Optimierung*

Der genetische Algorithmus ist ein populationsbasierter, evolutionärer Algorithmus, der sich an der biologischen Evolution der Natur orientiert. Damit wird die Fähigkeit der Adaption aufgegriffen (Gerdes et al. 2004). Für das vorgestellte Reihenfolgeplanungsproblem generiert der genetische Algorithmus neue Reihenfolgen von Fertigungsaufträgen durch Anwendung der Selektion und durch Einsatz der genetischen Operatoren (Rekombination und Mutation). Dem Nutzer stehen hierfür innerhalb des genetischen Algorithmus mehrere Verfahren zur Auswahl:

- Selektion: Remainder Stochastic Sampling with Replacement, Roulette Wheel Selection oder Stochastic Universal Sampling
- Rekombination: Cycle Crossover oder Order Crossover
- Mutation: Inversion Mutation oder Swap Mutation

Die Reihenfolgerestriktionen werden direkt im genetischen Algorithmus beachtet. Hierbei existieren zwei Möglichkeiten. Die erste Möglichkeit ist die Nutzung eines Bestrafungsfaktors $p>0$. Bei Nichterfüllung von Reihenfolgebeschränkungen verringert dieser Faktor den Fitnesswert der Reihenfolge, sodass die Selektion dieser Reihenfolge verschlechtert wird. Die zweite Möglichkeit, welche im vorliegenden Algorithmus angewendet wird, besteht in der Vermeidung von unzulässigen Auftragsreihenfolgen bei der Erzeugung der Anfangspopulation, bei der

Rekombination und bei der Mutation (Gerdes et al. 2004). Eine generierte Reihenfolge wird nur als valide angesehen, wenn alle Restriktionen erfüllt sind. Für den Fall, dass nicht-valide Reihenfolgen erzeugt werden, werden diese durch valide Reihenfolgen ersetzt. Dies hat den Vorteil, dass der genetische Algorithmus ausschließlich mit validen Auftragsreihenfolgen arbeitet und somit das Ergebnis der Optimierung ebenfalls alle gegebenen Reihenfolgerestriktionen erfüllt.

Die Bewertung einer Reihenfolge erfolgt durch Anwendung der Simulation. Für jede neu bestimmte Reihenfolge wird ein Simulationsexperiment mit dem Simulationsmodell des Montagesystems gestartet, in dem bei Vorliegen von stochastischen Eingangsgrößen mehrere unabhängige Simulationsläufe durchgeführt werden. Eine stochastische Eingangsgröße für ein Montagesystem ist beispielsweise die Montagezeit an einer Station (Miller und Osdel 2005). Beim Vorhandensein stochastischer Ergebnisgrößen wird eine dynamische Konfidenzintervallschätzung, bei gegebenen Konfidenzniveau durchgeführt. Dafür werden mindestens 10 und maximal 50 Simulationsläufe je Simulationsexperiment ausgeführt.

4 Reihenfolgerestriktionen

Eine Beschränkung der Reihenfolge von Fertigungsaufträgen C_r ist definiert durch eine Folge von logischen Ausdrücken BE_r, durch einen Restriktionstyp T_r und durch die Anzahl von Übereinstimmungen m_r, sodass gilt:

$$C_r = (BE_r, T_r, m_r), \forall r = 1,...,R . \tag{3}$$

Für die Beschreibung einer Folge von logischen Ausdrücken müssen zunächst Restriktionsvariablen definiert werden. Hierfür wird die Menge der Variantenmerkmale bzw. Features $\overline{F}$; verwendet, die im Simulationsmodell definiert sind und zur Spezifizierung einer Variante genutzt werden. Dementsprechend gilt für alle Restriktionsvariablen I_q mit $Q \in \mathbb{N}^+$ als Anzahl dieser Variablen:

$$I_q \subseteq \overline{F}, \forall q = 1,...,Q . \tag{4}$$

Diese Menge aller Variantenmerkmale charakterisiert ebenso alle Fertigungsaufträge J_i, um verschiedene Varianten eines Produkts abzubilden (siehe Gl. 5).

$$J_i \subseteq \overline{F}, \forall i = 1,..., N \tag{5}$$

In Kombination mit den logischen Operatoren ($\wedge,\vee,\neg$), den Klammersymbolen sowie dem Semikolon bilden die Restriktionsvariablen die Grundlage zur Definition eines logischen Ausdrucks E_{rj} ($\forall j = 1,\ldots,M_r$, $\forall r = 1,\ldots,R$), wobei das Semikolon das Ende eines Ausdrucks darstellt. Mehrere solcher Ausdrücke spezifizieren eine Folge BE_r mit $M_r \leq N$, wobei $M_r \in \mathbb{N}^+$ die Anzahl von logischen Ausdrücken in dieser Folge für die Reihenfolgerestriktion C_r ist.

In der vorliegenden Arbeit werden drei Typen von Reihenfolgestriktionen unterschieden:

- *required*: Eine Übereinstimmung der Folge von logischen Ausdrücken muss mindestens einmal in der Auftragsreihenfolge vorhanden sein.
- *prohibited*: Die Folge von logischen Ausdrücken darf nicht in der Auftragsreihenfolge vorkommen.
- *optional*: Nach Übereinstimmung einer bestimmten Anzahl von logischen Ausdrücken $m_r \in \mathbb{N}^+$ mit $m_r < M_r$ für eine Restriktion C_r müssen die nachfolgenden logischen Ausdrücke ebenfalls erfüllt sein.

Demgemäß ist $T_r \in \{\text{required, prohibited, optional}\}$ $(\forall r = 1,\ldots,R)$. Weiterhin wird die Anzahl der Übereinstimmungen m_r wie folgt beschrieben:

$$m_r = \begin{cases} 1, \ldots, M_r - 1, & \text{wenn Typ } T_r \text{ optional ist} \\ 0, & \text{sonst} \end{cases}, \forall r = 1, \ldots, R. \tag{6}$$

Für die Bestimmung, ob eine Reihenfolge Seq_k von Aufträgen eine Restriktion C_r erfüllt, muss die zugehörige Folge von logischen Ausdrücken BE_r gegen jede Teilreihenfolge der Länge M_r überprüft werden. Zur Auswertung eines logischen Ausdrucks für einen Fertigungsauftrag werden zunächst die Restriktionsvariablen des Ausdrucks überprüft. Basierend auf den Variantenmerkmalen ist eine Übereinstimmung einer Restriktionsvariable I_q mit einem Auftrag J_i vorhanden, wenn $I_q \subseteq J_i$. Die Restriktionsvariablen können durch logische Operatoren miteinander verknüpft werden. Hierbei ergibt sich das Ergebnis aus den Ergebnissen der Variablen und deren Verknüpfung durch den Operator. Im Folgenden wird dieses Vorgehen an einem Beispiel erläutert.

Beispiel:

In einem Simulationsmodell sind die Menge der Variantenmerkmale $F;^{-}=\{a,b,c\}$ und fünf Fertigungsaufträge ($J_1=\{a\}$, $J_2=\{a,b\}$, $J_3=\{a\}$, $J_4=\{b\}$, $J_5=\{c\}$) gegeben. Für die Reihenfolge $Seq_1= J_1, J_2, J_3, J_4, J_5$ sind drei Reihenfolgerestriktionen (C_1, C_2, C_3) zu beachten (siehe Gl. 7). Hierfür wurden drei Restriktionsvariablen ($I_1=\{a\}$, $I_2=\{b\}$, $I_3=\{c\}$) definiert.

$$\begin{aligned} C_1 &= (I_1; I_2; , \text{required}, 0) \text{ mit } M_1=2, \\ C_2 &= (I_3; I_1; , \text{prohibited}, 0) \text{ mit } M_2=2, \\ C_3 &= (I_1 \wedge I_2; I_1; I_2; , \text{optional}, 2) \text{ mit } M_3=3. \end{aligned} \tag{7}$$

Die Folge von logischen Ausdrücken ist für die erste Restriktion $BE_1=I_1;I_2;$ und besteht demnach aus zwei logischen Ausdrücken, sodass die Länge $M_1=2$ ist. Der Restriktionstyp von C_1 ist T_1=required mit $m_1=0$. Infolgedessen ist die erste Restriktion erfüllt, wenn mindestens ein Vorkommen von BE_1 in der Sequenz zu finden ist. Der Vergleich beginnt mit dem ersten logischen Ausdruck $E_{11}=I_1$; und dem ersten Auftrag J_1. Die Auswertung der Restriktionsvariable I_1 ergibt, dass $I_1 \subseteq J_1$ ist. Folglich ist der erste logische Ausdruck für den ersten Fertigungsauftrag erfüllt, sodass der nächste logische Ausdruck $E_{12}=I_2$; mit dem Auftrag J_2 verglichen wird. Dieser Ausdruck ist ebenso erfüllt, da $I_2 \subseteq J_2$. Demzufolge ist ein Vorkommen von BE_1 in Seq_1 zu finden und die Restriktion C_1 ist nicht verletzt.

Für die Restriktion C_2 ist die Folge $BE_2=I_3;I_1;$ definiert mit der Länge $M_2=2$. Der Typ dieser Restriktion ist T_2=prohibited. Demzufolge ist die Restriktion nicht erfüllt,

wenn eine Übereinstimmung der Folge von logischen Ausdrücken in der Reihenfolge von Fertigungsaufträgen vorhanden ist. Bei der Überprüfung des ersten logischen Ausdrucks $E_{21}=I_3$; mit allen Fertigungsaufträgen ist keine Übereinstimmung vorhanden, da $I_3 \not\subset J_1$, $I_3 \not\subset J_2$, $I_3 \not\subset J_3$, $I_3 \not\subset J_4$ und $I_3 \not\subset J_5$. Demnach ist die Folge von logischen Ausdrücken für die zweite Restriktion nicht in der Sequenz zu finden, sodass diese Restriktion erfüllt ist.

Die Restriktion C_3 besteht aus der Folge von logischen Ausdrücken $BE_3=I_1 \wedge I_2;I_1;I_2;$ der Länge $M_3=3$ mit den logischen Ausdrücken $E_{31}=I_1 \wedge I_2;$, $E_{32}=I_1;$ und $E_{33}=I_2;$. Der Restriktionstyp von C_3 ist T_3=optional mit $m_3=2$. Dieser Typ von Restriktionen überprüft hierbei die ersten beiden logischen Ausdrücke E_{31} und E_{32}, da $m_3=2$ ist. Sind diese erfüllt, dann muss auch der nachfolgende Ausdruck E_{33} übereinstimmen. In E_{31} sind die Restriktionsvariablen I_1 und I_2 durch den logischen Und-Operator verknüpft, sodass dieser Ausdruck nur wahr ist, wenn I_1 und I_2 erfüllt sind. Dies ist für den Auftrag J_2 gegeben, da $I_1 \subseteq J_2$ und $I_2 \subseteq J_2$. E_{32} ist ebenfalls für den Fertigungsauftrag J_3 erfüllt. Demnach muss E_{33} ebenfalls für den Auftrag J_4 übereinstimmen, damit die Restriktion C_3 nicht verletzt wird. E_{33} ist für den Auftrag J_4 erfüllt. Infolgedessen ist BE_3 in der Sequenz zu finden und die Restriktion ist nicht verletzt.

5 Implementierung in einem generischen Simulationssystem

5.1 Generisches Simulationsmodell eines Montagesystems

Der vorgestellte Lösungsansatz wurde in ein bestehendes IT-System zur Simulation von Montagesystemen integriert. Der zugrundeliegende Modellierungsansatz für solche Montagesysteme ist in Miller und Osdel (2005) beschrieben. In dem zu modellierenden Montagesystem werden verschiedene Varianten eines Produktes mit spezifischen Variantenmerkmalen bzw. Features, abhängig von den Anforderungen des Kunden montiert. Die durchzuführenden Montagearbeiten an einer Station sind dabei abhängig von der Produktvariante und werden als Operationen zusammengefasst. Ausgehend von diesem Ansatz sowie einer Taktzeit, kann die tatsächlich notwendige Bearbeitungszeit für eine Operation von der Taktzeit abweichen und zu zeitlichen Verzögerungen in der Montage führen.

Produkte und Werker im Montagesystem werden als aktive Elemente modelliert, wohingegen Stationen, an denen physisch die Arbeitsschritte vollzogen werden, als passive Entitäten dargestellt werden. Das Simulationssystem selbst ist in SLX implementiert. SLX (Simulation Language with eXtensibility) ist eine, von der Wolverine Software Corporation entwickelte, Programmiersprache zur Entwicklung von diskreten, ereignisgesteuerten Simulationen (Henrikson 2013).

Gleichzeitig wird die Idee der generischen Modellierung eines Simulationsmodells genutzt (Lemessi et al. 2011). Hierbei wird das konzeptuelle Modell in einer Datenbank gespeichert. Mit den dort hinterlegten Informationen werden, basierend auf einer Modellklassen-Bibliothek, beim Start des Simulationsmodells alle notwendigen Simulationsobjekte erstellt und instanziiert. Infolgedessen ist nur ein konzeptuelles Modell zu pflegen.

5.2 Anforderungen der Nutzer

Die potentiellen Nutzer der Optimierung sind Fertigungsplaner, die das Optimierungstool zur Festlegung der täglichen Montagereihenfolge verwenden. Daraus ergibt sich eine wesentliche Anforderung hinsichtlich der Ausführungszeit der Optimierung. Bei stochastischen Eingabegrößen im Simulationsmodell müssen für jede neu ermittelte Reihenfolge von Fertigungsaufträgen mehrere Simulationsläufe durchgeführt werden. Des Weiteren wird der genetische Algorithmus mehrmals ausgeführt. Für die maximale Laufzeit der simulationsbasierten Optimierung wird ein Richtwert von acht Stunden angenommen. Um diesem zeitlichen Limit gerecht zu werden, werden alle Schrittfolgen, die für die Optimierung nicht relevant sind, wie zum Beispiel die Berechnung von unwichtigen Ergebnis- und Animationsdaten, ausgeblendet. Nach einem Simulationslauf werden im Simulationsmodell temporäre Elemente gelöscht und statische Elemente zurückgesetzt sowie die Zufallszahlengeneratoren neu initialisiert. In diesem Zusammenhang ist ebenso das einmalige Einlesen von statischen Eingabedaten für die Optimierung als auch für das Simulationsmodell zu Beginn des Optimierungsprozesses zu erwähnen (Lemessi et al. 2011).

Eine weitere signifikante Anforderung ist eine einfache und intuitive Benutzerführung der Optimierung durch den Anwender auf der operativen Steuerungsebene. Hierfür wurde eine graphische Benutzeroberfläche erstellt, die den Nutzer sowohl bei der Konfiguration und bei der Ausführung der Optimierung, als auch bei der Interpretation und Evaluation der Optimierungsergebnisse unterstützt. Für die Anwender, die keine Kenntnisse über den genetischen Algorithmus besitzen, werden Standardeinstellungen vorgegeben:

- Populationsgröße: 40,
- Anzahl von Generationen: 50,
- Selektionsverfahren: Stochastic Universal Sampling,
- Rekombinationswahrscheinlichkeit: 70%,
- Rekombinationsverfahren: Cycle Crossover,
- Mutationswahrscheinlichkeit: 0,75%,
- Mutationsverfahren: Inversion Mutation.

6 Anwendungsbeispiel

Die Vorteile der Nutzung der simulationsgestützten Reihenfolgeoptimierung werden an dem folgenden Anwendungsfall aufgezeigt. Die Montagelinie besteht aus 17 sequentiellen Stationen mit jeweils identischer Auftragsfolge. Demzufolge kann das Reihenfolgeplanungsproblem als Permutations Flow-Shop-Problem klassifiziert werden. Auf der Montagelinie werden drei verschiedene Produkttypen in unterschiedlichen Funktions- und Designvariationen montiert. Die Montagezeiten für die jeweiligen Produkttypen an den Stationen sind deterministisch. Zusätzlich gegeben ist eine Reihenfolgerestriktion, die untersagt, dass ein bestimmtes Produkt nacheinander gefertigt werden darf. Die vorhandene Ausgangslösung für die Reihenfolge mit 50 Fertigungsaufträgen erfüllt die gegebene Reihenfolgerestriktion. Ein Umfang von 50 Fertigungsaufträgen entspricht dem Montagevolumen von einem Arbeitstag. Das angestrebte Ziel der Optimierung besteht in dem Finden einer

besseren Reihenfolge, hinsichtlich der Reduzierung der Zykluszeit für die 50 Aufträge.

Für das Permutations Flow-Shop-Problem mit N Fertigungsaufträgen existieren N! mögliche Auftragsfolgen (Kiener 2006; Zahn und Schmid 1996). Dementsprechend besteht der Suchraum für eine Reihenfolge mit 50 Aufträgen aus annähernd $3{,}04 \cdot 10^{64}$ potenziellen Lösungskandidaten, bei Vernachlässigung der Reihenfolgerestriktion.

Bei Anwendung der Standardeinstellung für das Optimierungstool wurde bei einer Startreihenfolge als Ausganglösung mit einer Zykluszeit von 1349 Minuten eine bessere Reihenfolge mit einer Zykluszeit von 1314 Minuten gefunden. Infolgedessen wurde eine Verbesserung der Zykluszeit um 35 Minuten erzielt. Der Verlauf der Optimierung ist in Abbildung 2 veranschaulicht. Drei Optimierungsläufe wurden durchgeführt. Mit den Standardeinstellungen wurden 798 Reihenfolgen untersucht bei einer Gesamtrechenzeit von 66 Minuten.

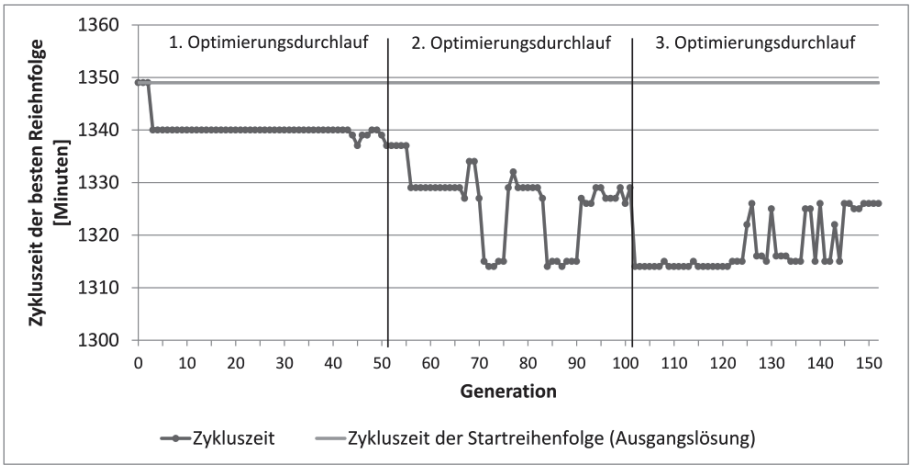

Abbildung 2: *Optimierungsverlauf*

7 Zusammenfassung und Ausblick

Dieser Beitrag erläutert einen simulationsbasierten Optimierungsansatz für das Flow-Shop-Problem der Reihenfolgeplanung von Fertigungsaufträgen in Montagesystemen. Dabei wird der genetische Algorithmus mehrmals ausgeführt. Neue Auftragsreihenfolgen werden durch Anwendung dieses Algorithmus generiert. Des Weiteren werden Reihenfolgebeschränkungen berücksichtigt, die mittels boolescher Algebra definiert werden. Der Optimierungsalgorithmus wurde in ein bestehendes Simulationsmodell für Montagesysteme integriert und mit einer graphischen Benutzeroberfläche zur Anwenderunterstützung verknüpft. Weiterhin wird an einem Anwendungsbeispiel die erreichte Verbesserung der Reihenfolge aufgezeigt. Der vorgestellte Ansatz ist konzeptuell gesehen nicht auf die simulationsbasierte Reihenfolgeplanung von Fertigungsaufträgen für Montagesysteme beschränkt.

Weiterführende Arbeiten fokussieren die parallele Ausführung von Simulationsläufen, um die Laufzeit der Optimierung zu reduzieren. Denkbar ist ebenfalls die Anwendung anderer Optimierungsziele, auch im Sinne einer multikriteriellen Optimierung.

Literatur

Bukchin, J.; Dar-El, E. M.; Rubinovitz, J.: Mixed model assembly line design in a make-to-order environment. In: Computers and Industrial Engineering (2002) 4, S. 405-421.

Chen, C.-L.; Vempati, V. S.; Aljaber Nasser: An application of genetic algorithms for the flow shop problems. In: European Journal of Operational Research (1995) 2, S. 389-396.

Gerdes, I.; Klawonn, F.; Kruse, R.: Evolutionäre Algorithmen. Genetische Algorithmen - Strategien und Optimierungsverfahren - Beispielanwendungen. Wiesbaden: Vieweg 2004.

Henrikson, J. O.: Wolverine Web. http://www.wolverinesoftware.com/, 30.04.2013.

Iltzsche, L.; Schmidt, P.-M.; Völker, S.: Simulationsbasierte Optimierung der Einsteuerungsreihenfolge für die Automobil-Endmontage. In: März, L.; Krug, W.; Rose, O.; Weigert, G. (Hrsg.): Simulation und Optimierung in Produktion und Logistik. Praxisorientierter Leitfaden mit Fallbeispielen. Heidelberg: Springer 2011, S. 117-132.

Jeffcoat, D. E.; Bulfin, R. L.: Simulated annealing for resource-constrained scheduling. In: European Journal of Operational Research (1993) 1, S. 43-51.

Jones, R. E.; Wilson, R. H.: A survey of constraints in automated assembly planning. In: IEEE International Conference on Robotics and Automation, Proceedings, 1996, S. 1525-1532.

Jones, R. E.; Wilson, R. H.; Calton, T. L.: Constraint-Based Interactive Assembly Planning. In: IEEE International Conference on Robotics and Automation, Proceedings, 1997, S. 913-920.

Kiener, S.: Produktions-Management. Grundlagen der Produktionsplanung und -steuerung. München, Oldenbourg 2006.

Lackes, R.; Awiszus, E.: Considering Complex Sequence Constraints in Production Scheduling – Results of a Practical Implementation in a German Trailer Company. In: International Proceedings of Economics Developement & Research, 2012, S. 32-37.

Lemessi, M.; Schulze, T.; Rehbein, S.: Simulation-based Optimization of Paint Shops. In: Jain, S.; Creasey, R. R.; Himmelspach, J.; White, K. P.; Fu, M. (Hrsg.): Proceedings of the 2011 Winter Simulation Conference, 2011, S. 2351-2362.

Mattfeld, D. C.; Bierwirth, C.: An efficient genetic algorithm for job shop scheduling with tardiness objectives. In: European Journal of Operational Research (2004) 3, S. 616-630.

Miller, T. G.; van Osdel, N.: Modeling Methods for Lean Assembly Systems Using SLX. In: Schulze, T.; Horton, G.; Preim, B.; Schlechtweg, S. (Hrsg.): Simulation und Visualisierung 2005. Proceedings der Tagung "Simulation und Visualisierung 2005" am Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg am 3. und 4. März 2005, 2005, S. 87-99.

Pinedo, M.: Scheduling. Theory, algorithms, and systems. 4. Auflage, New York: Springer 2012.

Rotondo, A.; Geraghty, J.; Young, P.: Using simulation and hybrid sequencing optimization for makespan reduction at a wet tool. In: Laroque, C.; Himmelspach, J.; Pasupathy, R.; Rose, O.; Uhrmacher, A. M. (Hrsg.): Proceedings of the 2012 Winter Simulation Conference, 2012.

Völker, S.; Schmidt, P.-M.: Simulationsbasierte Optimierung von Produktions- und Logistiksystemen mit Tecnomatix Plant Simulation. In: Zülch, G.; Stock, P. (Hrsg.): Integrationsaspekte der Simulation: Technik, Organisation und Personal. Karlsruhe: KIT Scientific Publishing, 2010, S. 93-100.

Yasin, A.; Puteh, N.; Daud, R.; Omar, M.; Syed-Abdullah S. L.: Product assembly sequence optimization based on genetic algorithm. In: International Journal on Computer Science and Engineering (2010) 2, S. 3065-3070.

Zahn, E.; Schmid, U.: Produktionswirtschaft I. Grundlagen und operatives Produktionsmanagement. Stuttgart, Lucius und Lucius 1996.