Simulation in Production
and Logistics 2015
Markus Rabe & Uwe Clausen (eds.)
Fraunhofer IRB Verlag, Stuttgart 2015

Bessere Prioritätsregeln für komplexe Produktionssysteme mittels multi-kriterieller simulationsbasierter Optimierung

Improved Dispatching Rules for Complex Manufacturing Systems Using Multi-objective Simulation-based Optimisation

Torsten Hildebrandt, Michael Freitag, BIBA – Bremer Institut für Produktion und Logistik GmbH an der Universität Bremen, Bremen (Germany), hil@biba.uni-bremen.de, fre@biba.uni-bremen.de

Abstract: This document describes the use of simulation-based, multi-objective optimisation (multi-objective Genetic Programming) to automatically develop improved dispatching rules. As the result of the optimisation is a new scheduling heuristic this can be categorized as a hyper-heuristic approach. For a complex manufacturing scenario from semiconductor manufacturing we show that resulting rules clearly outperform state-of-the-art rules from the literature. As a further novelty we demonstrate that the method allows to explicitly and effectively search for rules that are both good and short. Using rule length as an additional objective dimension we are able to find rules that are easy to understand and can also quantify the trade-offs possible between rule quality and rule complexity.

1 Einleitung

Im Rahmen dieses Beitrags wird die multi-kriterielle simulationsbasierte Optimierung eingesetzt, um verbesserte Prioritätsregeln für die operative Reihenfolgeplanung für ein komplexes Szenario aus dem Bereich der Halbleiterfertigung zu erstellen. Dieses ist gekennzeichnet durch: über 200 Maschinen; bis zu 355 Operationen, um ein Produkt zu fertigen; reihenfolgeabhängige Rüstzeiten; Batchprozesse; zyklische Materialflüsse. Im Unterschied zu den meisten Vorarbeiten aus der Literatur, können mittels multi-kriterieller Optimierung nun Regeln gefunden werden, die sowohl Kriterien wie die Durchlaufzeit als auch Termintreue verbessern und es außerdem erlauben, die Regellänge als ein weiteres Zielkriterium zu betrachten. Somit wird es möglich, explizit nach guten und kurzen Regeln sowie den dazwischen möglichen Abstufungen zu suchen. Diese Erweiterung bietet einen deutlichen Mehrwert für den Nutzer, da als Optimierungsergebnis eine Lösungs*menge* resultiert, in der jede darin enthaltene Regel einen in sich optimalen Kompromiss bspw. zwischen den größtenteils konfliktären Zielgrößen Durchlaufzeit und Termintreue darstellt. An-

hand dieser Ergebnismenge kann dann durch den Nutzer auf Basis der Menge möglicher Kompromisslösungen eine fundierte Entscheidung getroffen werden, welche Regel am besten seinen Präferenzen entspricht und den für ihn besten Kompromiss in der Zielerreichung darstellt.

Zunächst wird kurz das Problem der Reihenfolgeplanung für komplexe Produktionssysteme vorgestellt, wie es beispielsweise in der Halbleiterfertigung auftritt. Darauf folgt eine Vorstellung des Lösungsansatzes zur Kopplung der multi-kriteriellen Genetischen Programmierung (GP) als verwendetem Optimierungsverfahren mit der Simulation. Nach einer knappen Beschreibung des betrachteten Szenarios und des genauen Experimentaufbaus erfolgt eine Darstellung und Diskussion der Ergebnisse.

2 Stand der Technik

2.1 Automatischer Entwurf von Prioritätsregeln

Prioritätsregeln zur operativen Reihenfolgeplanung (vgl. Haupt 1989) sind ein weit verbreitetes Verfahren in Softwarelösungen zur Fertigungsfeinplanung. Immer, wenn eine Ressource frei wird und noch Aufträge auf Bearbeitung warten, wird jeder Auftrag mittels einer Prioritätsregel bewertet. Der Auftrag mit dem höchsten Prioritätswert wird als nächstes gestartet. Zur Beliebtheit von Prioritätsregeln trägt bei, dass sie einfach zu verstehen und zu implementieren sind. Darüber hinaus stellen sie nur minimale Rechenzeitanforderungen, so dass sie auch als Echtzeit/Online-Verfahren eingesetzt werden können und somit ihre Entscheidungen auf Basis der jeweils aktuellsten verfügbaren Daten treffen können. Dies ist wichtig zur geeigneten Reaktion auf unvorhergesehene Störungen (z.B. Maschinenausfälle). Beispiele für einfache Standardregeln sind FIFO (First In (buffer), First Out) oder die Kürzeste-Operationszeit-Regel (engl. SPT, Shortest Processing Time first). Simulationssoftware als auch die meisten MES (Manufacturing Execution System) mit Feinplanungsfunktion enthalten meist eine Reihe an Standardregeln, aus denen man die für die jeweilige Problemstellung beste auswählen kann.

Um darüber hinaus gehende Verbesserungen zu erreichen, wird oft auf einen manuellen Entwurfsprozess zurückgegriffen, in dem ein Systemexperte sich auf Basis seiner Erfahrungen und den Simulationsergebnissen mit Standardregeln neue Regeln ausdenkt. Diese werden implementiert und mittels Simulation bewertet. Dies führt zu einem zyklischen Verbesserungsprozess, an dessen Ende eine verbesserte Regel steht. Um diesen sehr zeit- und kostenaufwendigen, größtenteils manuellen Entwurfsprozess zu verbessern, werden zunehmend Verfahren vorgeschlagen, die automatisiert Prioritätsregeln erstellen. Eine aktuelle Übersicht über diese Arbeiten findet sich in Branke et al. (2015). Die meisten dieser Verfahren verwenden die Genetische Programmierung (Poli et al. 2008) als sogenannte Hyper-Heuristik. Hyper-Heuristiken (Burke et al. 2013) sind ein relativ neues Forschungsfeld, insbesondere auch im Zusammenhang mit Simulation. Hierin werden Verfahren untersucht, bei denen eine Meta-Heuristik (wie die Genetische Programmierung) nicht eingesetzt wird, um direkt ein Problem zu lösen (z. B. ein Reihenfolgeplanungsproblem mit konkreten bekannten Aufträgen). Stattdessen stellt das Ergebnis der Hyper-Heuristik eine neue Heuristik dar, mit der die zugrunde liegende Problemstellung gelöst werden kann.

2.2 Multi-kriterielle Optimierung von Prioritätsregeln

Klassische Ansätze zur simulationsbasierten Optimierung gehen davon aus, dass sich die zu minimierende bzw. zu maximierende Zielgröße als ein einzelner skalarer Wert ausdrücken lässt. Für viele Fragestellungen stellt dies jedoch eine starke Vereinfachung dar. Die multi-kriterielle Optimierung (Marler und Arora 2004) ermöglicht auch in solchen Fällen eine Lösung. Ohne Beschränkung auf eine einzelne Zielgröße bzw. ohne die a-priori Festlegung von Gewichtungen für die einzelnen (Teil-)Zielgrößen, lassen sich hierzu insbesondere die auf dem Konzept der Pareto-Optimalität beruhenden Ansätze verwenden, die in den letzten Jahren zunehmend an Popularität gewinnen (Coello Coello 2006). Hierbei ist die Lösung eines Optimierungslaufs nicht eine einzelne „optimale“ Lösung, sondern stattdessen eine Pareto-Front, also eine Menge an Lösungen, die alle einen bestimmten erreichbaren Kompromiss zwischen den Teilzielgrößen darstellen. Form und Lage dieser Lösungsmenge aus jeweils nicht-dominierten Lösungen stellen für den Nutzer wichtige Informationen dar, um aus dieser Menge an möglichen Kompromiss-Lösungen die für seine Präferenzen beste auszuwählen.

In der Literatur konnten wir nur zwei Arbeiten identifizieren, die ebenfalls multi-kriterielle Optimierung im Kontext der Erstellung von Prioritätsregeln einsetzen. Tay und Ho (2008) betrachtet drei Zielkriterien für eine flexible Werkstattfertigung mit bis zu 15 Maschinen. Während der Optimierung werden die drei Zielkriterien jedoch mit festen Gewichtungen zu einer gemeinsamen Zielfunktion kombiniert. Somit wird letztlich eine uni-kriterielle Optimierung durchgeführt und das Ergebnis der Optimierungsläufe stellt somit keine Regelmenge dar. Deutlich näher an den in diesem Beitrag durchgeführten Untersuchungen ist Nguyen et al. (2013). Darin werden Prioritätsregeln für ein dynamisches Werkstattfertigungsszenario mit 10 Maschinen entwickelt. Es wird ein Pareto-Ansatz verwendet, d. h., das Ergebnis eines Optimierungslaufs ist tatsächlich eine Regelmenge wie in den hier vorgestellten Experimenten. Das von Nguyen et al. (2013) verwendete Szenario ist jedoch deutlich einfacher als das diesem Beitrag zugrunde liegende. Darüber hinaus verwendet der von uns entwickelte Ansatz zur multi-kriteriellen Genetischen Programmierung verschiedene Techniken, um die Konvergenzgeschwindigkeit der Optimierungsläufe zu verbessern (Normierung der Attributwerte, Erkennung und Entfernung von Duplikaten). Eine Betrachtung der Regellänge als weiterem Zielkriterium findet ebenfalls nicht statt.

3 Lösungsansatz

Grundsätzlich arbeitet das für dieses Paper verwendete Optimierungsverfahren der Genetischen Programmierung (GP) mit einer Baumrepräsentation für die Lösungen. Bei der hier vorliegenden Anwendung der Erstellung von Prioritätsregeln ist dieser Baum die Baumstruktur des arithmetischen Ausdrucks der Regel. Ein Bespiel hierzu ist in Abbildung 1 (links) dargestellt. Der dort dargestellte Ausdrucksbaum stellt die (hypothetische) Prioritätsregel „0-(PT+PT+S+B)“ dar, wobei PT, S und B jeweils Attribute des zu bewertenden Auftrags darstellen oder Größen zur Charakterisierung des aktuellen Systemzustands repräsentieren. Die möglichen Bestandteile eines solchen Baums, also die möglichen Rechenoperationen und die zur Verfügung stehenden Attribute, sind vor dem Optimierungslauf festzulegen. Im vorliegenden Beitrag wurden hierfür im Wesentlichen die häufig in Standardregeln verwendeten

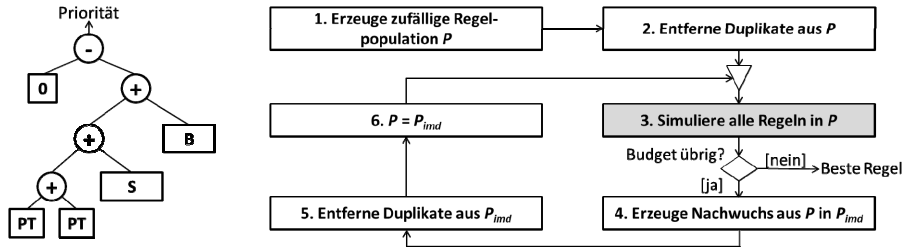

Abbildung 1: *Beispiel einer als Baum repräsentierten Prioritätsregel (links); Grafische Darstellung des verwendeten Optimierungsalgorithmus (rechts)*

Rechenoperationen und Attribute verwendet. Die Größe des Baumes muss bei GP nicht im Voraus festgelegt werden, sondern ergibt sich aus den Eigenschaften des zu lösenden Problems. Als Maß für die Länge einer Prioritätsregel wird im vorliegenden Beitrag die Anzahl an Knoten des entsprechenden Baumes betrachtet, im Beispiel in Abbildung 1 (links) ist sie also 9.

Der grundlegende Ablauf der Optimierung ist in Abbildung 1 (rechts) dargestellt. In Schritt 1 wird zunächst eine Menge an zufälligen Regeln erzeugt. Aus diesen lassen sich in Schritt 2 mithilfe des in Hildebrandt und Branke (2015) näher beschriebenen Vorgehens ohne rechenaufwändige Simulationsläufe syntaktisch unterschiedliche, in ihrem Verhalten jedoch gleiche Regeln erkennen und entfernen. Die Hauptiteration besteht aus den Schritten 3 bis 6. Im mit Abstand rechenaufwändigsten Schritt 3 werden nun alle Regeln der Regelpopulation P mittels Simulation bewertet. Auf Basis dieser Bewertung werden anschließend unter Nutzung der Grundprinzipien der Evolution (Selektion, Mutation, geschlechtliche Fortpflanzung/Cross-Over) eine neue Zwischenpopulation P_{imd} erzeugt. Aus dieser werden dann wie für die Anfangspopulation in Schritt 2 eventuelle Duplikate erkannt und durch zusätzlich aus P erzeugte Regeln ersetzt. In Schritt 6 schließlich ersetzt die Zwischen-population P_{imd} die aktuelle Population und eine neue Iteration beginnt. Als Abbruchbedingung wird nach Schritt 3 geprüft, ob eine bestimmte Anzahl an Simulationsläufen bereits erreicht wurde die Hauptschleife ggfs. verlassen.

Die Erweiterung zu dem in Hildebrandt et al. (2014) verwendeten Vorgehen besteht nun zum einen in der Verwendung eines anderen Selektionsoperators in Schritt 4. Hierzu wird das NSGA-II-Verfahren adaptiert (Non-dominated Sorting Genetic Algorithm-II, Deb et al. 2002). Dies ermöglicht es, mehreren Zielkriterien gerecht werden zu können und während eines einzelnen Optimierungslaufs eine Menge nichtdominierter Lösungen zu ermitteln.

Darüber hinaus erwies sich eine zusätzliche Feinauswahl nach Ende der Haupt-Iteration als sinnvoll zur Erzielung besserer Ergebnisse. Dies ist ebenfalls eine Erweiterung zum bisherigen Vorgehen, bei dem die beste Regel aus der letzten Iteration der Optimierung als Gesamtergebnis des Optimierungslaufs zurückgegeben wurde. Die für dieses Paper verwendete Feinauswahl berücksichtigt stattdessen die jeweils beste Regel der letzten 5 Iterationen und bewertet diese in jeweils 30 von der Optimierung unabhängigen Replikationen, um dann mit höherer Wahrscheinlichkeit die tatsächlich beste Regel zurückliefern zu können. Anstelle einer einzelnen Regel ergibt sich bei mehreren Zielgrößen am Ende jeder Iteration jeweils eine Menge nichtdominierter Lösungen, aus denen dann analog die endgültige Regelmenge ermittelt wird.

4 Szenariobeschreibung und Experimentaufbau

4.1 Szenario und Benchmark-Regeln

Das für diesen Beitrag betrachtete Produktionsszenario aus dem Bereich der Halbleiterfertigung ist das FAB6-Modell des „MIMAC (Measurement and Improvement of Manufacturing Capacities) testbed“ (Feigin et al. 1996). Dieses Modell ist durch die folgenden Charakteristika gekennzeichnet: 104 Maschinengruppen mit insgesamt 223 Maschinen, Batch-Maschinen, reihenfolgeabhängige Rüstzeiten, 9 verschiedene Produkte, die zwischen 234 und 355 Operationen erfordern, zyklischer Materialfluss, Maschinenausfälle und –Wartung.

Das Modell wird für 18 Monate simuliert (etwa 3600 fertiggestellte Aufträge/Lose), wobei die hier wiedergegebenen Ergebnisse nur die letzten 12 Monate dieses Zeitraums erfassen und die ersten 6 Monate zur Vermeidung von Einschwingeffekten für die Ergebnisauswertung ignoriert werden. Als stochastische Einflussgrößen sind Maschinenausfälle und -wartung modelliert, alle weiteren Größen (insbesondere auch Auftragsankünfte) sind deterministisch. Bzgl. Produktmix werden hier die in Hildebrandt et al. (2014, Table 1) mit „Setting 1“ bezeichneten Einstellungen verwendet, auf deren Wiedergabe hier aus Platzgründen verzichtet wird. Im Unterschied hierzu wird jedoch ein Flussfaktor von 2.12 statt 2.30 verwendet, da sich letzterer als zu einfach erwies.

Wenn nicht anders angegeben, sind die im Folgenden wiedergegebenen Ergebnisse zur Leistungsbewertung einer Regel die Mittelwerte aus 30 unabhängigen Replikationen. Diese Replikationen sind ebenfalls unterschiedlich zu den während der Optimierung und (Fein-)Auswahl der besten Regeln verwendeten. Als Simulationssoftware kommt die am BIBA entwickelte Simulationsbibliothek jasima (JAva SImulator for MAnufacturing and logistics, http://jasima.googlecode.com/) zum Einsatz.

Vorgelagert zu den Optimierungs-Experimenten wurde in einer konventionellen Simulationsstudie zunächst die beste Standard-Regel für dieses Szenario ermittelt. Hierzu wurde eine Menge von 7 Standard-Regeln betrachtet, vgl. Haupt (1989): FIFO (first in first out), ERD (earliest release time first), EDD (earliest due date first), ODD (operational due date), MOD (modified operational due date), SPT (shortest processing time first), sowie CR (critical ratio). Darüber hinaus wird die ODD+ Regel untersucht (siehe Zhou und Rose (2012), dort als IFD-Regel bezeichnet). Diese stellt eine Erweiterung von ODD dar, die Aufträgen, die bereits länger als 2 Tage vor einer Maschine warten die höchste Priorität zuweist, unabhängig von ihrem Due Date. Bei allen 8 Regeln wird ERD als „tie breaker“ verwendet, sollten zwei Aufträge anhand der Basisregel die gleiche Priorität zugewiesen bekommen. Alle Regeln werden darüber hinaus mit der „Kürzesten Rüstzeit“-Regel kombiniert. Auf Maschinen mit reihenfolgeabhängigen *Rüstzeiten* werden also zunächst Aufträge mit möglichst kurzer Rüstzeit bevorzugt und nur bei gleicher Rüstzeit wird die Prioritätsregel herangezogen. Für die *Batchmaschinen* wird ein „greedy batching“ durchgeführt. Hierzu wird zunächst der Auftrag mit der höchsten Priorität gemäß der Basis-Regel ausgewählt. Mit diesem Auftrag werden dann möglichst viele batchkompatible Aufträge kombiniert und der Batch gestartet. Als Parameter in dieser Heuristik kann eine bestimmte minimale Batch-Größe (MBS-Minimum Batch Size) eingestellt werden. Im Folgenden wird dies mit MBS(u) gekennzeichnet, wobei u eine Zahl zwischen 0 und 1 ist. Ein u von 0 startet einen Batch sofort, sobald

mindestens ein Auftrag vorhanden ist. U=1 wartet immer, bis ein Batch der maximalen Größe gebildet werden kann. In der Voruntersuchung wurden alle 8 verschiedenen Regeln mit jeweils 6 Einstellungen der Steuerung der Batch-Maschinen kombiniert (MBS(0), MBS(0,2), …, MBS(0,8), MBS(1,0)).

4.2 Multi-kriterielle Genetische Programmierung

Die Simulation mittels *jasima* wurde mit dem Framework ECJ (Evolutionary Computation in Java, http://cs.gmu.edu/~eclab/projects/ecj/, Version 20) zur Durchführung der Optimierungsläufe gekoppelt. Die verwendeten GP-Parameter entsprechen den in Hildebrandt et al. (2014) genutzten Einstellungen, eine detailliertere Beschreibung lässt sich dort nachlesen. Insgesamt werden in einem Optimierungslauf jeweils 30000 Simulationsläufe durchgeführt (60 Generationen mit jeweils 500 Regeln). Die möglichen Bestandteile von Prioritätsregeln sind hierbei vorgegeben. Neben den vier Grundrechenarten, max und if-then-else als mathematischen Operationen und vier verschiedenen Konstanten (0, 1, 2, 5) stehen insgesamt 10 verschiedene Attribute zur Verfügung. Diese sind allesamt leicht zu ermittelnde Größen, wie sie auch in Standardregeln verwendet werden: (A1) Bearbeitungszeit der aktuellen Operation, (A2) Durchschnittliche Bearbeitungszeit aller vor der Maschine wartenden Aufträge, (A3) Rüstzeit, (A4) Durchschnittliche Rüstzeit, (A5) Anzahl der verbleibenden Operationen eines Auftrags, (A6) Verweilzeit des Auftrags im System, (A7) Verweilzeit des Auftrags in aktueller Warteschlange, (A8) Größe der Batch-Familie (wie viele rüstkompatible Aufträge sind in der Warteschlange), (A9) Due Date der aktuellen Operation, (A10) Durchschnittliches Due Date aller wartenden Operationen. Darüber hinaus wurde ECJ zur Verwendung der NSGA-II-spezifischen Klassen entsprechend konfiguriert und erweitert. Als Zielfunktionen wurden die mittlere Durchlaufzeit (DLZ), mittlere Verspätung (Versp.) und die Länge der Regeln als Zielkriterien verwendet. Im Einzelnen wurden folgende Kombinationen untersucht:

1. mittlere Durchlaufzeit (uni-kriteriell)
2. mittlere Durchlaufzeit, Regellänge (2 Zielkriterien)
3. mittlere Durchlaufzeit, mittlere Verspätung (2 Zielkriterien)
4. mittlere Durchlaufzeit, mittlere Verspätung, Regellänge (3 Zielkriterien).

Um Aussagen über die mittlere Güte der zu erwartenden Optimierungsergebnisse machen zu können, wurden für jede der 4 Einstellungen jeweils 10 unabhängige Optimierungsläufe durchgeführt.

5 Ergebnisse und Diskussion

5.1 Primäre Zielfunktion: Durchlaufzeit

In den durchgeführten Experimenten zur Auswahl der besten Benchmark-Regel erweist sich die ODD+-Regel als am besten. Unabhängig von der verwendeten Regel ist im betrachteten Szenario die Variante MBS(0) die beste. Es lohnt sich (wahrscheinlich aufgrund der hohen Auslastung von 95%) hier nicht, auf Aufträge zu warten, um einen Batch besser füllen zu können. Die Kombination ODD+/MBS(0) erzielt als beste Benchmark-Regel ein Ergebnis von 27,37 (±0,19) Tagen mittlerer Durchlaufzeit bei einer mittleren Verspätung von 76,9 (±31,7) Minuten. Etwa 11% aller Aufträge sind verspätet.

Tabelle 1: *Mittlere Qualität der besten DLZ-Regeln für verschiedene Zielfunktionen*

Zielkriterien	DLZ [d]	Versp. [min]	Regellänge
DLZ	24,87 (±0,05)	2.9 (±2,7)	233,5 (±62,9)
DLZ, Länge	24,89 (±0,06)	3.7 (±3,1)	48,0 (±11,9)
DLZ, Versp.	24,83 (±0,03)	0.1 (±0,1)	197,5 (±50,5)
DLZ, Versp., Länge	24,87 (±0,08)	1.1 (±1,6)	84,9 (±23,9)

Um die Qualität der erreichbaren Regeln in den Optimierungs-Varianten 1-4 bewerten und einfacher miteinander vergleichen zu können, erfolgt eine Fokussierung auf das Zielkriterium „mittlere Durchlaufzeit“. Das heißt, bei den Varianten mit mehr als einem Zielkriterium, die also eine Regelmenge zum Ergebnis haben, erfolgt jeweils eine Auswahl der nach dem Zielkriterium „mittlere Durchlaufzeit“ besten Regel. Somit ergibt sich je Variante aus den jeweils 10 durchgeführten Optimierungsläufen eine Menge von 10 Regeln. Die sich hieraus ergebenden Resultate sind in Tabelle 1 aufgeführt. Jede Zeile beschreibt dabei eine der untersuchten Varianten und gibt die hierfür erzielten mittleren Ergebnisse (gemittelt über die jeweils 10 Optimierungsläufe) hinsichtlich Durchlaufzeit, mittlerer Verspätung der Aufträge und Regellänge an. Die in Klammern angegebenen Werte stellen jeweils den doppelten Standardfehler der Ergebnisse aus den 10 Optimierungsläufen dar. Wie zu erkennen ist, führen alle untersuchten Varianten bzgl. Durchlaufzeit zu sehr ähnlichen Ergebnissen. Zumindest nach 10 Optimierungsläufen ist keine signifikant besser oder schlechter. Dies ist ein wichtiges Ergebnis, zeigt sich doch, dass gegenüber einem uni-kriteriellen Vorgehen (Variante 1) keine Abstriche hinsichtlich Regelqualität zu machen sind. Im Vergleich mit der besten Benchmarkregel lässt sich die Durchlaufzeit sehr deutlich um ca. 2,5 Tage bzw. 9.0% reduzieren.

Was das Kriterium der mittleren Verspätung angeht, erzielen alle Varianten ebenfalls sehr gute Werte. Obwohl die Tabelle 1 zugrunde liegenden Regeln allesamt „Spezialisten“ für das Kriterium der mittleren DLZ sind, kommen allenfalls einzelne Aufträge zu spät, so dass sich insgesamt nur sehr kleine mittlere Verspätungen ergeben. Die Verbesserungen in Bezug auf die mittlere DLZ sind so groß, dass der verwendete Flussfaktor, der für die Benchmark-Regeln eine große Herausforderung darstellt, für die optimierten Regeln kein Problem ist. Die untersuchten Varianten, die während der Optimierung explizit die Verspätung mit berücksichtigen, erzielen ggfs. etwas geringere Verspätungen, die Differenzen sind aber statistisch ebenfalls nicht sehr aussagekräftig und sollten in weiteren Arbeiten in Versuchen mit engeren Due Dates näher untersucht werden.

Sehr deutlich ist der Effekt der Reduktion der Regellänge, wenn die Länge als zusätzliches Kriterium während der Optimierung mit berücksichtigt wird. Wird nur nach Durchlaufzeit minimierenden Regeln gesucht, schrumpft die Länge auf ein Viertel ihres ursprünglichen Werts. Wird nach Regeln gesucht, die sowohl nach DLZ als auch Verspätung gut sind, halbiert sich die mittlere Regellänge immerhin noch. Diese Betrachtung bezieht sich allerdings nur auf die Regeln maximaler Länge. Auf Basis der Lösungs*menge* als Ergebnis der multi-kriteriellen Optimierung lassen sich jedoch auch Aussagen treffen, welche Ergebnisse sich mit kürzeren Regeln erzielen lassen. Hierauf wird in Abschnitt 5.2 noch näher eingegangen.

Tabelle 2: *Ergebnisse der jeweils besten gefundenen Regeln*

Zielfunktionen	DLZ [d]	Versp. [min]	Regellänge
DLZ	24,79 (±0,10)	0,08 (±0,06)	388
DLZ, Länge	24,73 (±0,17)	8,76 (±17,06)	37
DLZ, Versp.	24,77 (±0,10)	0,05 (±0,07)	303
DLZ, Versp., Länge	24,55 (±0,37)	0,80 (±1,12)	69

In Tabelle 2 sind die Ergebnisse der jeweils besten Regel aus den jeweils 10 Optimierungsläufen einzeln aufgeführt. Die in Klammern aufgeführten Zahlen stellen den doppelten Standardfehler über die 30 genutzten Replikationen dar. Die Ergebnisse unterstreichen im Grunde die eben gemachten Beobachtungen auf Basis der gemittelten Werte. Die insgesamt beste Regel wird interessanterweise von der eigentlich komplexesten Optimierungs-Variante mit drei gleichzeitigen Zielkriterien gefunden. Diese erzielt eine mittlere DLZ von 24,55 Tagen und somit ein um 2,8 Tage bzw. 10,3% besseres Ergebnis als die beste Benchmark-Regel.

5.2 Abwägung Durchlaufzeit vs. Regellänge

Die explizite Berücksichtigung der Regellänge während der Optimierung von Prioritätsregeln stellt eine Neuheit dieses Beitrags dar. In den Experimenten unter Einbeziehung der Regellänge kann nun nicht nur nach guten, sondern nach guten und möglichst kurzen Regeln gesucht werden. Grundsätzlich ist eine Abwägung erforderlich: kurze Regeln sind gut interpretierbar und lassen sich besonders schnell berechnen. Andererseits ist zur Erreichung besonders guter Ergebnisse eine gewisse Regellänge notwendig. Dass die Berücksichtigung der Regellänge als Zielfunktion deutlich die Regellänge senkt, ohne dass hierdurch die Qualität bzgl. des Zielkriteriums mittlere Durchlaufzeit sinkt, konnte bereits in Abschnitt 5.1 gezeigt werden. Im Folgenden erfolgt eine detaillierte Betrachtung der Abwägung Regellänge vs. erreichbare Durchlaufzeit.

Hierzu wurden die Ergebnisse aller durchgeführten Optimierungsläufe ausgewertet und aus allen Regeln diejenigen ausgewählt, die jeweils einen in sich optimalen Kompromiss zwischen DLZ und Regellänge darstellen. Dies ergibt eine Menge an 11 Regeln mit Mindestgröße 5, welche in Abbildung 2 grafisch dargestellt sind. Wie man sieht, beträgt der Unterschied zwischen der kürzesten (Länge 5), aber schlechtesten Regel und der besten, dafür aber komplexesten Regel (Länge 69) etwa einen Tag an mittlerer Durchlaufzeit. 9 Abstufungen liegen jeweils dazwischen. Je nach Präferenz hin zu einfacheren oder aber möglichst guten Regeln kann nun eine fundierte Auswahl getroffen werden, um die letztlich am besten geeignete Regel auszuwählen und diese dann bspw. in operativen Planungs- und Steuerungssystemen einzusetzen.

6 Zusammenfassung und Ausblick

Dieser Beitrag beschreibt den Einsatz der multi-kriteriellen, simulationsbasierten Optimierung, um deutlich leistungsfähigere und kürzere Prioritätsregeln zur Reihen-

folgeplanung für ein komplexes Produktionsszenario aus dem Bereich der Halbleiterfertigung mit einem automatisierten Verfahren zu erstellen.

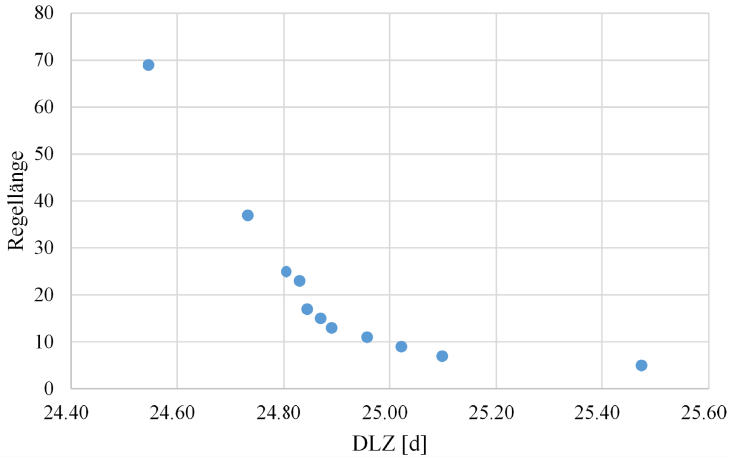

Abbildung 2: *Mögliche Kompromisslösungen zwischen Regellänge und erzielbarer Durchlaufzeit*

Der Beitrag beschreibt nach bestem Wissen der Autoren zum ersten Mal in der Literatur den Einsatz multi-kriterieller Optimierung für die Prioritätsregelentwicklung in einem Szenario der Komplexität wie das hier betrachtete. Mit den entwickelten Regeln konnte die mittlere Durchlaufzeit von Aufträgen um bis zu 9,4% gesenkt werden. Darüber hinaus ergaben die durchgeführten Experimente keine Hinweise auf Leistungseinbußen gegenüber einer uni-kriteriellen Vorgehensweise. Als weitere Neuerung gegenüber der bestehenden Literatur wird die multi-kriterielle Optimierung eingesetzt, um explizit nach *kurzen*, guten Prioritätsregeln zu suchen. Während diese Vorgehensweise in der GP-Literatur für verschiedene Problemstellungen bereits erfolgreich angewendet wurde (vgl. de Jong und Pollak 2003), stellt ihr Einsatz im Rahmen der Entwicklung von Prioritätsregeln eine Neuerung dar. In den durchgeführten Experimenten konnte hierdurch die Regellänge sehr deutlich reduziert werden. Hierdurch ergibt sich auf Basis eines einzelnen Optimierungslaufs eine fundierte Entscheidungsgrundlage, um zwischen kurzen, leicht interpretierbaren Regeln, die aber nur verhältnismäßig schlechte Ergebnisse erreichen, und sehr guten, dafür aber komplexen und schwer interpretierbaren, Regeln die geeignetste auszuwählen.

Im Ergebnis wird also eine Erweiterung der automatisierten Entwicklung von Prioritätsregeln vorgestellt, die die Anwendbarkeit und den Nutzen des Verfahrens durch den Einsatz der simulationsbasierten multi-kriteriellen Optimierung in mehrfacher Hinsicht verbessern. Das vorgestellte Verfahren fließt in ein entsprechendes Optimierungsmodul ein, das im Laufe der letzten Jahre bereits sehr erfolgreich zur uni-kriteriellen Entwicklung von Prioritätsregeln eingesetzt wurde. Im Rahmen einer aktuellen Ausgründung aus dem BIBA werden hierdurch interessierten Unternehmen diese neuen Möglichkeiten zur Effizienzsteigerung ihrer Prozesse eröffnet.

Danksagung

Wir danken dem Bundesministerium für Wirtschaft und Energie für Unterstützung im Rahmen des EXIST-Gründerstipendiums „jasima solutions“.

Literatur

Branke, J.; Nguyen, S.; Pickardt, C.; Zhang, M.: Automated design of production scheduling heuristics: A Review. IEEE Transactions on Evolutionary Computation 2015, doi:10.1109/TEVC.2015.2429314, to appear.

Burke, E.; M. Gendreau; M. Hyde; G. Kendall; G. Ochoa; E. Ozcan; Qu, R.: Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society 64 (2013) 12, S. 1695–1724.

Coello Coello, C.: Evolutionary multi-objective optimization: A historical view of the field. IEEE Computational Intelligence Magazine 1 (2006) 1, S. 28–36.

Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2002) 2, S. 182-197.

Feigin, G.; Fowler, J.; Leachman, R.: MASM test data sets. 1996. Available from http://masmlab.engineering.asu.edu/ftp.htm. Accessed June 06, 2014.

Haupt, R.: A survey of priority rule-based scheduling. OR Spektrum 11 (1989) 1, S. 3–16.

Hildebrandt, T.; Branke, J.: On using surrogates with Genetic Programming. Evolutionary Computation 2015, doi:10.1162/EVCO_a_00133, to appear.

Hildebrandt, T.; Goswami, D.; Freitag, M.: Large-scale simulation-based optimization of semiconductor dispatching rules. In: Proceedings of the 2014 Winter Simulation Conference, Savannah 2014, S. 2580–2590.

de Jong, E.; Pollak, J.: Multi-objective methods for tree size control. Genetic Programming and Evolvable Machines 4 (2003) 3, S. 211-233.

Marler, R.; Arora, J.: Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optim. 26 (2004) 6, S. 369–395.

Nguyen, S.; Zhang, M.; Johnston, M.; Tan, K.: Dynamic multi-objective job shop scheduling: A genetic programming approach. In: Uyar, A.; Ozcan, E.; Urquhart, N. (Hrsg.): Automated Scheduling and Planning. Heidelberg, New York, Dordrecht, London: Springer 2013, S. 251–282.

Poli, R.; Langdon, W., McPhee, N.: A field guide to genetic programming. 2008. http://www.gp-field-guide.org.uk. Accessed June 06, 2014.

Tay, J.; Ho, N.: Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers & Industrial Engineering 54 (2008) 3, S. 453–473.

Zhou, Z.; Rose, O.: WIP balance and due date control in a wafer fab with low and high volume products. In: Proceedings of the 2012 Winter Simulation Conference, Berlin 2012, S. 2019-2026.