Center for
Cognitive Interaction Technology
Kognitronik und Sensorik
Prof. Dr.-Ing. U. Rückert

Simulationsgestützte Optimierung in verteilten, heterogenen Umgebungen anhand virtueller Werkzeugmaschinen

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEUR (Dr.-Ing.)

der Technischen Fakultät der Universität Bielefeld

vorgelegte Dissertation

von

Raphael-Elias Reisch

Referent: Prof. Dr.-Ing. Ulrich Rückert

Bielefeld / November 2021

Gedruckt auf alterungsbeständigem Papier °° ISO 9706

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich auf dem Weg zur Erstellung fachlich und persönlich unterstützt haben. Insbesondere bedanke ich mich bei der Arbeitsgruppe „Computational Materials Science and Engineering“ unter der Leitung von Prof. Christian Schröder. Die vielen interessanten und aufschlussreichen fachlichen Diskussionen haben maßgeblich zur Entstehung dieser Arbeit beigetragen.

Ein ganz besonderer Dank gilt meiner Frau, die mir immer den Rücken freigehalten hat. Ihr ist diese Dissertation gewidmet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

1. Einleitung

Innerhalb der letzten Jahre haben sich Schlagwörter wie Industrie 4.0, Internet der Dinge oder Smart Monitoring als eine Art Sammelbegriff für die digitale Transformation der industriellen Produktion etabliert. In dem Zusammenhang sind etwa seit Beginn des aktuellen Jahrzehnts diverse Initiativen – beispielsweise in Form öffentlicher Förderprogramme – entstanden, die diese Begriffe durch die Entwicklung konkreter Lösungen greifbar machen sollen. Die Ausrichtung dieser Forschungsinitiativen erfolgt üblicherweise hochgradig interdisziplinär, da die zu bearbeitenden Fragestellungen aus ingenieurswissenschaftlicher Sicht getrieben werden, allerdings Lösungen aus diversen Forschungsfeldern der Informatik erfordern. Offensichtlich ist das Zusammenwachsen dieser beiden Disziplinen nicht vollkommen neu. Entwicklungen wie Computer Aided Design (CAD) oder Computer Aided Manufactoring (CAM) sind bereits seit den 50er Jahren des 20. Jahrhunderts aufgekommen [1]. Darüber hinaus werden zur numerischen Lösung von Differentialgleichungen seit mehreren Jahrzehnten Finite-Elemente-Methoden eingesetzt, die bis heute als Standard gelten [2]. Nichtsdestotrotz spricht man, insbesondere in Deutschland, vom Aufbruch der vierten industriellen Revolution, der sich aus der stärkeren Verzahnung der beiden genannten Disziplinen ergebe [3] [4]. Diese Einschätzung lässt sich im Wesentlichen auf zwei Entwicklungen zurückführen. Zunächst eröffnet die rasante Entwicklung des Internets über die letzten 20 Jahre die Konzeptionierung vollkommen neuer Geschäftsmodelle im Kontext des Cloud Computing. So ist davon auszugehen, dass die Abgrenzung innerhalb von Märkten, deren Geschäft auf Hardware beruht, künftig über den Verkauf von Software und Services sichergestellt wird. Im Laufe der vorliegenden Arbeit wird dieser Aspekt vertieft. Der zweite Grund für die Einschätzung, dass wir uns am Anfang einer neuen industriellen Revolution befinden, liegt in den Erfolgen der Grundlagenforschung in Bereichen wie künstlicher Intelligenz (z.B. durch künstliche neuronale Netze, Sprach- und Gestenerkennung etc.) und heuristischer Optimierung, die auf Grundlage nicht deterministischer Verfahren Lösungen für unterschiedliche Fragestellungen eröffnen. Prominente Vertreter dieser Fragestellungen sind zum Beispiel *Predictive Maintenance* zur Vorhersage von Maschinenausfällen und das sogenannte autonome Fahren, also die selbstständige Navigation eines Fahrzeugs, die ohne jeglichen manuellen Eingriff auskommt.

Ein entscheidender Aspekt im Zusammenhang mit den Verfahren, die nahezu sämtlichen Fragestellungen im beschriebenen Kontext zugrunde liegen, ist die Notwendigkeit paralleler bzw. verteilter Rechnerarchitekturen. Auch in dem Gebiet hat es über die letzten Jahrzehnte entscheidende Meilensteine gegeben, die effiziente Berechnungen, etwa für das Trainieren eines komplexen neuronalen Netzwerks oder die Simulationen eines Finite-Elemente Netzes mit einer hohen Anzahl an Knoten, realisierbar machen. In

dem Zusammenhang sind, neben dem Cloud Computing, die Nutzung hochskalierbarer Grafikkarten (GPGPU) für nebenläufige Berechnung oder Public Resource Computing zu nennen.

Die vorliegende Arbeit beschäftigt sich mit dem Thema der simulationsgestützten Optimierung im Anwendungsfeld von CNC-Maschinen. Spezifischer formuliert ist es das Ziel, herauszuarbeiten, welche Optimierungspotenziale sich bei der Konfiguration von Werkzeugmaschinen für ein bestimmtes NC-Programm unter Annahme einer bestimmten Maschine ergeben. Der Fokus liegt demnach nicht darauf, Verfahrwege des Werkzeugarms zu optimieren, da dies einen sehr invasiven Eingriff in die Produktion bedeuten würde, sondern vielmehr auf Optimierungsmöglichkeiten während der Arbeitsplanung.

Ein Blick in die wesentlichen Anwendungsfelder simulationsgestützter Optimierung zeigt, dass die vorliegende Thematik mit Problemen konfrontiert ist, die in diesem Kontext ansonsten eine eher untergeordnete Rolle spielen. Domänen, in denen Simulationen in eine Optimierungsschleife eingebettet werden, finden sich hauptsächlich in der Logistik, der Gebäudeplanung oder der intelligenten Verkehrsplanung [5] [6]. Eine Reihe weiterer Publikationen geht nicht über die Betrachtung künstlich erzeugter „Spielzeuganwendungen“ hinaus [7] [8]. Was diese Anwendungsfelder wesentlich von einer programmspezifischen Optimierung unterscheidet, ist die Tatsache, dass eine optimierte Konfiguration für ein System gesucht wird, das in der anschließenden Realisierung recht statisch ist, etwa wenn es um die Planung von Produktionsstraßen oder Verkehrstrassen geht. Aus diesem Grund kann eine lange Berechnungsdauer für eine simulationsgestützte Optimierung relativ problemlos in Kauf genommen werden. Die Situation kann bei der CNC-Bearbeitung deutlich anders geartet sein. Grundsätzlich befindet man sich hier zunächst in der Situation, dass für jedes NC-Programm und jede Maschine, auf die das NC-Programm grundsätzlich übertragen werden kann, jeweils eine optimierte Konfiguration berechnet werden muss. Da das Spektrum für subtraktive Fertigung sehr weit reicht, kann keine grundsätzliche Aussage über den Produktzyklus gemacht werden, der dem betrachteten NC-Programm zugrunde liegt. Aus diesem Grund ist neben der reinen Anwendbarkeit von simulationsgestützter Optimierung eine schnelle Konvergenz des Verfahrens von zentraler Bedeutung.

Ein Teil der Beschleunigung der Verfahren soll durch paralleles bzw. verteiltes Rechnen erreicht werden. Es wird sich im Verlauf der Arbeit zeigen, dass hierbei grundsätzlich verschiedene Strategien verfolgt werden können, da die verteilten Ressourcen entweder auf Simulations- oder auf Optimierungsebene einsetzbar sind. Weitere Herausforderungen im Kontext der Parallelisierung der Optimierung entstehen durch Aspekte wie heterogene Rechnerarchitekturen im Simulationscluster, etwa in Cloud- oder Gridumgebungen. Neben weiteren Aspekten ist dies ein Grund für die Notwendigkeit, ein Verfahren zu entwickeln, das ein möglichst geringes Maß an

Synchronisierung der Rechnerknoten untereinander benötigt. Insofern bündelt die vorliegende Arbeit eine spezifische und eine generelle Fragestellung. Die spezifische Fragestellung ist, welche Potenziale durch simulationsgestützte Optimierung bei der Einrichtung von Werkzeugmaschinen gehoben werden können. Die generelle Fragestellung beschäftigt sich mit der Thematik, ein Verfahren zu entwickeln, das eine verbesserte Aufspannsituation in akzeptabler Zeit liefert. Trotz der spezifischen Formulierung ist diese insofern generell, weil eine schnelle Konvergenz auch für andere Domänen entscheidend ist.

Die Arbeit wird wie folgt strukturiert sein: Zunächst wird ein genauer Blick auf den konkreten Simulationsgegenstand, also Werkzeugmaschinen, geworfen, um ein grundlegendes Verständnis zu garantieren. In dem Zusammenhang wird der Aspekt der Materialabtragsimulation und deren Geschichte beleuchtet.

Auf der Basis erfolgt eine konkretere Beschreibung der Verfahren, die in die Lösung der Fragestellung eingegangen sind, wobei hierbei jeweils der Stand der Technik erläutert wird. Im Wesentlichen sind hierbei das Forschungsgebiet der simulationsgestützten Optimierung und die mathematische Repräsentation des sogenannten G-Codes erwähnenswert. Es wird gezeigt, dass simulationsgestützte Optimierung durch den Einsatz paralleler bzw. verteilter Rechnerarchitekturen zwar bis zu einem gewissen Zeitpunkt problemlos linear skaliert, allerdings dennoch die Notwendigkeit der Entwicklung intelligenter Methoden gegeben ist, die die Anzahl der Simulationen drastisch reduzieren. Bezüglich der mathematischen Repräsentation des G-Codes wird gezeigt, dass sich hieraus ein Verfahren ableiten lässt, mit dem sehr effizient Kennzahlen ermittelt werden können, ohne auf die Durchführung einer komplexen Simulation angewiesen zu sein.

Danach wird auf die Frage eingegangen, welche Freiheitsgrade für die Einrichtung einer Werkzeugmaschine zur Verfügung stehen und welche Optimierungspotenziale sich daraus schließlich ergeben. Es wird sich zeigen, dass hierbei im Wesentlichen die initiale Positionierung eines Werkstücks im Mittelpunkt steht, die durch Translation oder Rotation adjustierbar ist. Neben der Frage, welche Freiheitsgrade zur Verfügung stehen, muss auf der Basis auch die Frage beantwortet werden, wie die Veränderung der entsprechenden Parameter an die Maschine gebracht werden kann. Insbesondere die Rotation eines Werkstücks erfordert erhebliche Eingriffe, da für deren Umsetzung das NC-Programm verändert werden muss. Es wird gezeigt, auf welche Art und Weise diese Veränderung automatisiert ermöglicht wird.

Auf der Grundlage der vorangegangen Kapitel ist dann die Grundlage gelegt, um die Evolution des schließlich entwickelten Verfahrens zu verdeutlichen. Hierzu werden die zuvor formulierten Anforderungen nach und nach zu einem Algorithmus geführt, mit dessen Hilfe im besten Fall lediglich eine Simulation notwendig ist, um eine optimale Lösung für die Maschinenkonfiguration zu finden. Die experimentellen Untersuchungen

innerhalb des Kapitels sind auf Basis realer Fertigungsszenarien entstanden. Neben der Betrachtung dieser Szenarien werden außerdem Untersuchungen anhand theoretischer Benchmarkfunktionen durchgeführt, die die Betrachtung größerer Statistiken erlauben.

Zum Abschluss der Betrachtung zur simulationsgestützten Optimierung wird diskutiert, wie sich das Verfahren in ein komplexeres System, das verschiedene Optimierungsszenarien bündelt, einbetten lässt. Für eine tiefere Betrachtung zur prototypischen Umsetzung einer Plattform, in der ein solches System implementiert und umfangreich evaluiert wurde, sei auf die entsprechende Buchveröffentlichung verwiesen [9].

Da die Arbeit sich insgesamt nah am spezifischen Thema *virtuelle Werkzeugmaschinen* bewegt, ist es zum Abschluss wichtig, zu verdeutlichen, dass die entwickelten Verfahren grundsätzlich generalisierbar sind. Während des Projekts sind in dem Kontext mehrere Überlegungen entstanden, inwieweit simulationsgestützte Optimierung für das Curvefitting im Kontext von Monte Carlo Simulation zur Feststellungen von magnetischen Wechselwirkungen durch Abgleich von Experiment und Simulation eingesetzt werden kann. Hierbei lassen sich leicht Szenarien konstruieren, die den Anforderungen des Forschungsprojekts sehr ähnlich sind. So ist es durchaus realistisch, durch Einsatz von Public Resource Computing heterogene Hardwareumgebungen herzustellen.

1.1. Werkzeugmaschinen

In diesem Kapitel wird ein grober Überblick über die Funktionsweise und den Aufbau von Werkzeugmaschinen präsentiert. Für eine detailliertere Betrachtung dieser Thematik sei an dieser Stelle auf [10] verwiesen. Die Schwerpunkte, die im vorliegenden Kontext für das Verständnis entscheidend sind, beschränken sich im Wesentlichen auf die Unterscheidung der Bewegungsachsen von Dreh- und Fräsmaschinen, die Programmierung mittels NC-Steuerungen und die Rolle von speicherprogrammierbaren Steuerungen. Darüber hinaus wird der Begriff der Kollision eingeführt und kurz vorgestellt, wie der Prozess zur Einrichtung einer Werkzeugmaschine aussieht.

Zunächst stellt sich die Frage, wie eine Werkzeugmaschine generalisiert definiert werden kann. Laut [10] kann eine Werkzeugmaschine ausgedrückt werden als „Maschine […], die der Fertigung mechanischer Komponenten, definierter, reproduzierbarer Form mit Hilfe von Werkzeugen“ diene, wobei die Formgebung „durch eine mechanisiert angetriebene und geführte Relativbewegung zwischen Werkzeug und Werkstück, die sich in Prozess- und Vorschubbewegung unterteilen lässt“, geschehe. Entscheidend ist zudem, dass

Werkzeugmaschinen in der sogenannten subtraktiven Fertigung zum Einsatz kommen. Die Fertigung erfolgt also durch Materialabtrag eines Rohteils.

Die Topologie von Werkzeugmaschinen weist eine starke Diversität auf. Aus dem Grund wird an dieser Stelle die Maschinengeometrie nach und nach durch die Beschreibung der üblicherweise verbauten Komponenten entwickelt. Es ist sinnvoll, hierbei mit der Werkzeugspindel und dem Maschinentisch zu beginnen. Die Werkzeugspindel dient als Greifer für das verwendete Werkzeug. Einfach ausgedrückt handelt es sich um einen Drehantrieb mit integrierter Werkzeugaufnahme. Der Maschinentisch dient offensichtlich als Unterlage für die Positionierung des zu bearbeitenden Werkstücks. Üblicherweise wird eine Aufspannvorrichtung wie beispielsweise ein Schraubstock auf dem Tisch befestigt, mit deren Hilfe das Werkstück positioniert und fixiert wird. Die initiale Beschreibung dieser beiden Komponenten ist an dieser Stelle opportun, weil es sich um jene Komponenten handelt, deren Bewegungen über Bahnachsen kontrolliert werden. Im einfachsten Fall weist eine Werkzeugmaschine drei orthogonale Linearachsen (X, Y, Z) auf. Die Maschinenkoordinaten erfolgen nach der 3-Finger-Regel der rechten Hand, wobei zwischen horizontaler und vertikaler Z-Achse unterschieden werden kann [11]. Die Zuordnung zwischen Komponenten und Bahnachsen ist abhängig von der Maschinenkonfiguration. So existieren Maschinen, bei der die Werkzeugspindel entlang der Z- und X-Achse verfährt, während die Bewegung des Tisches über die Y-Achse geregelt ist. Ebenso sind Topologien denkbar, bei denen die Werkzeugspindel ausschließlich durch die Z-Achse gesteuert wird, wobei der Maschinentisch auf der X- und Y-Achse verfährt. Unabhängig von der Topologie liegen auf Basis der aktuellen Beschreibung lediglich drei Achsen vor. Im Umfeld des Maschinenbaus wird allerdings zwischen 3-Achs- und 5-Achsbearbeitung unterschieden, womit sich die Frage stellt, auf welche Weise zwei weitere Achsen zur Verfügung gestellt werden können. Für jede Linearachse (X, Y, Z) wird für diesen Zweck eine Drehachse (A, B, C) eingeführt. Per Konvention bezeichnet A die Drehachse um X, B die Drehachse um Y und C die Drehachse um Z. In der Praxis werden über die Drehachsen Schwenkvorgänge des Arbeitstisches bzw. der Werkzeugspindel oder Drehvorgänge des Arbeitstisches realisiert.

Neben Tisch und Spindel ist im vorliegenden Kontext das Werkzeugmagazin als Komponente einer Werkzeugmaschine zu erwähnen. Üblicherweise besteht die Bearbeitung eines Werkstücks durch eine Fräsmaschine aus einer Reihe heterogener Fräsvorgängen, deren Durchführung unterschiedliche Anforderungen an das verwendete Werkzeug aufweist. So existieren spezialisierte Werkzeuge, etwa zum Schruppen, Lochfräsen, zur Kantenbearbeitung oder zum Fräsen bestimmter Nuten. Soll ein Werkzeug gewechselt werden, verfährt die Maschine zum sogenannten **Werkzeugwechselpunkt.** An diesem Punkt wird das ausgewählte Werkzeug in die Spindel eingespannt und kann von dem Zeitpunkt an für die Fortsetzung der Werkstückbearbeitung verwendet werden.

Nach Erläuterung der grundlegenden Geometrien stellt sich nun die Frage nach der Benutzerschnittstelle einer Werkzeugmaschine. Die Verfahrwege der Bahnstrecken werden über eine sogenannte CNC (Computerized Numerical Control) gesteuert. Hierbei handelt es sich um maschinennahe Software, die üblicherweise auf leistungsfähigen Industrie-PCs im Feld läuft. Diese hat die Aufgabe, Achsverfahrwege hinsichtlich der „Zykluszeit der Lageregler“ dahingehend zu interpolieren, dass die an der Bewegung beteiligten Achsen entlang vorgegebener Sollpositionen verfahren [12]. Für die Programmierung dieser Verfahrwege steht grundsätzlich eine Reihe von Schnittstellen zur Verfügung. Im Rahmen dieser Arbeit gehen wir lediglich von dem Fall aus, dass diese maschinenfern während der Arbeitsvorbereitungsphase vorgenommen wird. In der realen Fertigung ist der Fall, dass die Programmierung direkt an der Maschine durch einen Facharbeiter erfolgt, allerdings ebenfalls üblich. Zu einer Art Industriestandard hat sich seit einigen Jahrzehnten CAD-CAM entwickelt. Der Begriff CAD bezeichnet „Computer-aided Design“ und dient als eine Art Synonym für sämtliche Prozesse, die mit der rechnergestützen Erzeugung eines Modells einer bestimmten Zielgeometrie für ein Werkstück zusammenhängen. Im vorliegenden Zusammenhang ist es hinreichend, CAD als Weg von der geplanten, aber nicht digitalisierten, Zielgeometrie zu einer mathematischen Repräsentation dieser Geometrie in Form eines Standardformats (z.B. STL) zu betrachten. Auf Basis dieser Repräsentation kann dann in einem ebenfalls rechnergestützten Prozess das CNC-Programm erzeugt werden. Dieser Prozess wird als CAM („Computer-aided Manufacturing) bezeichnet. Im Wesentlichen ist hiermit die Verwendung einer Software durch einen Mitarbeiter in der Arbeitsvorbereitung zu verstehen, die eine CAD-Datei entgegennimmt und die Erstellung des CNC-Programms maschinenspezifisch unterstützt. Die grobe Vorgehensweise besteht aus zwei Schritten. Der erste Schritt umfasst die Festlegung der zu verfahrenden Bahnwege. Hierzu erhält das System zunächst einige Informationen über die Randbedingungen in der Fertigung (verwendete Maschine, Material des Werkstücks etc.). Im Anschluss schlägt das Programm geeignete Verfahrwege für die Produktion vor. In Interaktion mit dem Nutzer wird nach und nach die Folge der Bahnwege konstruiert, die die Fertigung des Rohteils hin zur Zielgeometrie ermöglichen. Ein wichtiger Aspekt ist hierbei, dass die Folge der Bahnwege nicht dem eigentlich CNC-Programm entspricht, das schließlich an die Steuerung übertragen wird, da das Ergebnis dieses ersten Schritts noch nicht in einem maschinenlesbaren Format vorliegt. Diese Aufgabe wird vom sogenannten Postprozessor übernommen. Der maschinenspezifische Postprozessor nimmt die Folge der Bahnwege in einem internen CAM-Format entgegen und übersetzt das Programm in eine Sprache, auf deren Basis die Maschine verfahren kann. Je nach Maschinentyp kann es sich hierbei um DIN 66025/ISO 6983, auch G-Code genannt, oder beispielsweise um Heidenhain Klartext handeln [11]. G-Code ist für sämtliche CNC-Maschinen interpretierbar. Da die Syntax und Semantik von G-Code im späteren Verlauf in aller Tiefe behandelt werden, sei an dieser Stelle lediglich erwähnt, dass im G-Code jegliche Information für die Fertigung abgebildet ist, also Bahnwege, Werkzeugwechsel, Vorschübe etc.

Ein großer Vorteil der Arbeitsvorbereitung mittels CAD-CAM ist die Möglichkeit, mittels grafischer Nutzerschnittstellen eine Art Simulation einzelner Fräsvorgänge durchzuführen. Das ermöglicht den präventiven Ausschluss von Kollisionen in der Fertigung schon während der Arbeitsvorbereitung. Aufgrund der zentralen Bedeutung des Kollisionsbegriffs für die vorliegende Arbeit wird dieser im Folgenden definiert:

*Definition: Als **Kollision** wird eine unerlaubte Berührung zweier beliebiger Entitäten während der Bearbeitung eines Werkstücks betrachtet. Erlaubte Berührungen können hierbei lediglich zwischen Werkzeugkopf und Werkstück stattfinden.*

Demnach kann eine Kollision beispielsweise durch Berührung eines Werkzeugs und dem Arbeitstisch beziehungsweise dem Spannmittel zustande kommen. Sowohl für CAD als auch für CAM steht eine Vielzahl kommerzieller oder auch freier Lösungen zur Verfügung. Eine Übersicht findet sich in folgenden Referenzen [13] [14].

1.2. Materialabtragsimulation

Aufgrund von signifikanten finanziellen Schäden, die durch eine fehlerhafte Produktion mit Werkzeugmaschinen entstehen können, sind zuverlässige Simulationsumgebungen unerlässlich. Diese ermöglichen einerseits die Vorhersage, dass ein gefertigtes Werkstück die korrekte Zielgeometrie aufweist, wodurch Ausschuss in der realen Produktion ausgeschlossen werden kann. Andererseits werden Kollisionen vermieden, die je nach Schwere erhebliche Folgen für die Produktion verursachen. Zunächst stellt sich die Frage, ob die Notwendigkeit von Materialabtragssimulationen im Widerspruch zu der Tatsache steht, dass sowohl das Fertigen von Zielgeometrien als auch die Überprüfung von Kollisionen bereits im CAM-Prozess abgebildet werden. Dies ist aus dem einfachen Grund nicht der Fall, weil jegliche Überprüfungen in der CAM Software vor dem Einsatz des Postprozessors durchgeführt werden [11]. Insofern kann eine starke Gewährleistung der Richtigkeit eines CNC-Programms nur durch eine detailgetreue Simulation erfolgen. Neben einer Reihe von industriellen Lösungen in dem Bereich ist die Thematik seit den 1980er Jahren ein Bestandteil verschiedener Forschungsfelder. Die Fragestellungen innerhalb dieser Thematik bewegen sich in einem relativ breiten Spektrum. So wird in vielen Publikationen die visuelle Darstellung von Simulationen in Form von Animationen in den Mittelpunkt gestellt, was im Zusammenhang der vorliegenden Arbeit ein relativ unbedeutender Aspekt ist. Andere Autoren stellen die geometrische Modellierung von Werkstücken und Werkzeugen in den Mittelpunkt, wobei hauptsächlich über die optimale Balance zwischen Simulationsgeschwindigkeit und Genauigkeit des berechneten Materialabtrags diskutiert wird. Grundsätzlich können zwei Ansätze für die Durchführung einer Materialabtragssimulation durchgeführt werden – mithilfe einer exakten, analytischen Berechnung des Abtrags oder durch approximative Verfahren [15]. Es ist seit früher Forschung in dem Feld bekannt, dass die exakte Lösung in der Komplexitätsklasse $O(N^4)$ liegt, wobei N der Anzahl der Wegbefehle für das Werkzeug

entspricht [16], was für deren Anwendung in der Praxis eine extreme Hürde darstellt. Da die Anforderungen an Materialabtragssimulationen allerdings grundsätzlich eher industriegetrieben sind, sind nicht erkannte Kollisionen schwer hinnehmbar. Insofern sind Annäherungen an die exakte Lösung nur in einem recht geringen Ausmaß möglich, was sich üblicherweise auch durch subjektiv hohe Simulationsdauern innerhalb der industriellen Praxis äußert. Im Folgenden werden grob ausgewählte Verfahren vorgestellt, was vor allem der formalen Definition von Kollisionen und einer groben Vorstellung des Forschungsfelds dienen soll. Für tiefere Betrachtungen sei auf die jeweiligen Veröffentlichungen verwiesen. Um das Wesentliche aus den einschlägigen Veröffentlichungen zu bündeln, ist es hinreichend, festzuhalten, dass diverse entscheidende Geometrien (Werkstücke, Spannmittel, Werkzeuge etc.) als Kollektion von Voxeln repräsentiert werden. Mengentheoretisch kann man sich dann den Bauraum als ein dreidimensionales Grid von Voxeln vorstellen. Jeder Voxel kann dann markiert werden mit der Information, ob dieser von einer kollisionsrelevanten (Spannmittel, Werkzeugspindel) oder kollisionsirrelevanten (Werkstück, Werkzeug) Geometrie „geschnitten“ wird. Sämtliche übrige Voxel werden als 0 markiert. Dann kann die Durchführung des NC-Programms als Veränderung der Voxelmarkierungen über die Zeit bezüglich der NC-Sätze aufgefasst werden. Tritt während der Bearbeitung beispielsweise ein logischer Schnitt zwischen Werkstück und Werkzeug auf, findet eine boolesche Subtraktion an der Stelle auf. Auf die Weise können offensichtlich auch Kollisionen formalisiert werden. Eine Kollision lässt sich dann auffassen als ein boolescher Subtraktionsschritt, bei dem mindestens eine kollisionsrelevante Markierung beteiligt ist. Nähere Betrachtungen und Formalisierungen für derartige Betrachtungen liefern [17] und [18].

2. Methoden

2.1. Simulationsgestützte Optimierung

Simulationsgestützte Optimierung hat in den letzten Jahren extrem an Bedeutung gewonnen, insbesondere im Bereich *Decision Support* [19] [20] [21]. Die Methode ist in diversen unterschiedlichen Disziplinen, in denen Simulationen zum Einsatz kommen, erfolgreich verwendet worden. Nach kurzer Vorstellung des grundsätzlichen Verfahrens werden einige konkrete Anwendungsfälle grob vorgestellt, die mit simulationsgestützter Optimierung erfolgreiche Ergebnisse erzeugen konnten. Im Anschluss werden einige Optimierungsansätze vorgestellt und erläutert, aus welchem Grund eine direkte Anwendung des Verfahrens im vorliegenden Kontext nicht erfolgsversprechend ist.

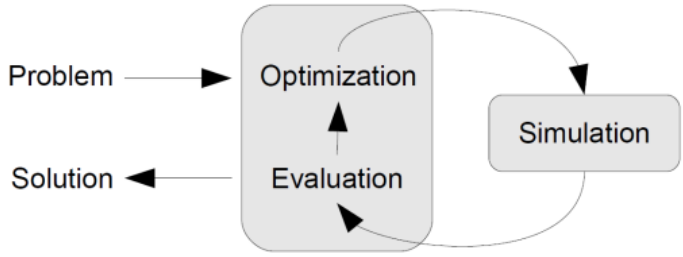

Abb. 1 Zyklus zur simulationsgestützten Optimierung

Abb. 1 zeigt das grundsätzliche Vorgehen für die simulationsgestützte Optimierung. Man beginnt mit einem Problem, das daraus besteht, für ein bestimmtes Modell, das hier nicht näher definiert ist, die optimale Parameterkonfiguration zu finden. Üblicherweise wird, innerhalb festgelegter Randbedingungen, zunächst eine zufällige Konfiguration ermittelt, die dann als Eingabe für eine Simulation dienen kann. Das Ergebnis, dessen Struktur an dieser Stelle ebenfalls noch nicht definiert ist, wird dann als Information genutzt, um auf intelligente Art und Weise eine neue Konfiguration zu ermitteln. Dieses Vorgehen wird bis zum Erreichen eines Abbruchkriteriums wiederholt. Die beste Lösung im Sinne des Optimierungsproblems wird schließlich ausgegeben. Es stellt sich zunächst die Frage, wie diese beste Lösung allgemein definiert werden kann. Hierzu sei zunächst ein Simulationsmodell in seiner allgemeinsten Form definiert als Abbildung $S: \mathrm{X} \rightarrow \mathrm{Y}$, wobei $X := \{x_1, x_2, \dots, x_n\}$,und $Y := \{y_1, y_2, \dots, y_m\}$ bzw. $Y := S(X)$, wobei $x_i, y_j \in \mathbb{R}$. Die Eingabevariablen X und Ausgabevariablen Y sind hier nicht genauer definiert und können einen beliebigen Definitionsbereich aufweisen. Auf der Basis kann eine sogenannte Fitnessfunktion definiert werden als Abbildung $f: Y \rightarrow \mathbb{R}$. Für ein bestimmtes Simulationsmodell lässt sich also eine beliebige Fitnessfunktion $f\big(S(X)\big)$ definieren. Sei nun $\check{X} \subseteq X$ die Menge der Eingangsvariablen, die beliebig einstellbar ist.

Dann ist das Ziel der simulationsgestützten Optimierung, die Werte für $\check{X}$ so zu wählen, dass $f\big(S(X)\big) \to min$, falls ein Minimum gesucht ist bzw. $f\big(S(X)\big) \to max$, falls ein Maximum gesucht ist. Die Aufgabenstellung entspricht formal gesehen also der Suche nach einem Extremwert einer nichtdifferenzierbaren Funktion. Für deren Lösung steht eine Reihe von Heuristiken zur Verfügung, deren Vorgehensweisen in verschiedene Klassen fallen. Eine grobe Einteilung lässt sich durch die Unterscheidung zwischen populations-, bzw. schwarmbasierten Verfahren und Einzelpunktsuchen vornehmen. Im Folgenden werden exemplarisch drei Algorithmen vorgestellt, die innerhalb des Projekts zum Einsatz gekommen sind.

2.1.1. Partikelschwarmoptimierung

Bei der Partikelschwarmoptimierung handelt es sich um ein sogenanntes naturanaloges Optimierungsverfahren, das 1995 von Kennedy und Eberhart veröffentlicht worden ist [22] und auch heute noch weiterentwickelt wird [23] [24]. Die Idee besteht hierbei darin, das Verhalten von natürlichen Schwärmen, etwa Fischen oder Vögeln, abzubilden, um daraus eine Regelmäßigkeit abzuleiten, die für eine numerische Optimierung genutzt werden kann. Motiviert wird dieser Ansatz durch die Tatsache, dass Schwarmintelligenzen danach streben, als Gesamtsystem für jedes Individuum ein gewisses Optimum zu erreichen, etwa zwecks Ausweichens von Fressfeinden oder Einsparens benötigter Energie. Hierbei wirken im Wesentlichen zwei implizite Kräfte. Einerseits neigt der Schwarm dazu, nicht auseinander zu driften, was dazu führt, dass der Schwarm sich selbst erhält. Andererseits werden einzelne Individuen innerhalb des Schwarms stets einen gewissen Abstand zu jedem anderen Individuum wahren.

Es stellt sich also die Frage, wie sich diese Beobachtungen formal fassen lassen, um ein Optimierungsverfahren mit mehrdimensionalen Eingangsgrößen daraus abzuleiten. Dazu seien zunächst einige Größen definiert. Sei $\omega \in \mathbb{R}$ eine Trägheitskonstante und $\varsigma_g, \varsigma_l \in \mathbb{R}$ zwei Gewichtskonstanten, wobei ς_g eine globale Gewichtung und ς_l eine lokale Gewichtung repräsentiert. Darüber hinaus wird $n \in \mathbb{N}$ definiert als die Größe des Schwarms, also die Anzahl der Partikel. Dann kann $i \in [1, n]$ als Indizierung für jedes Partikel verwendet werden. Dann sei $\vec{\rho}_{global} \in \mathbb{R}^d$ definiert als der optimale Eingangsvektor, der bis zum aktuellen Zeitpunkt ermittelt worden ist, wobei $d \in \mathbb{N}$ die Anzahl der Dimensionen des Suchraums repräsentiert. Zusätzlich wird für jedes einzelne Partikel der Vektor $\vec{\rho}^{\,i}_{local} \in \mathbb{R}^d$ vorgehalten, also der Eingangsvektor, der bezogen auf Partikel i den bis dahin besten Fitnesswert erzeugt hat. Es wird also unterschieden zwischen einem globalen und einem lokalen Gedächtnis. Für die Iteration werden zusätzlich jeweils eine Positions- und ein Geschwindigkeitsvektor für sämtliche Partikel benötigt. Diese seien notiert als $\vec{x}_i$ für die aktuellen Positionen und $\vec{v}_i$ für die Geschwindigkeiten.

Algorithmus: Partikelschwarmoptimierung (PSO)
Input: Fitnessfunktion f, Konstanten $d, n, \omega, \varsigma_g, \varsigma_l$
Initialisierung: $\vec{x}_i, \vec{v}_i \sim \mathcal{N}(b_{min}, b_{max}), \vec{\rho}_{global} \rightarrow \vec{x}_{\mathrm{i}} : \min\left(f(\vec{x}_i)\right), \vec{\rho}^{i}_{local} \rightarrow \vec{x}_i \forall i, \rho^{best}_{global} \rightarrow \min\left(f(\vec{x}_i)\right), \rho^{best_i}_{local} \rightarrow f(\vec{x}_i) \forall i, t \rightarrow 0$
While (Kein Konvergenzkriterium ist erreicht)
$t \rightarrow t + 1$
For $(i \rightarrow [1, n])$
$v_{i,t} \rightarrow \omega * v_{i,t-1} + \varsigma_g * \left(\vec{\rho}_{global} - \vec{x}_i\right) + \varsigma_{local} * (\vec{\rho}^{i}_{local} - \vec{x}_i)$
$\vec{x}_i \rightarrow \vec{x}_i + \vec{v}_i$
If $(f(\vec{x}_i) < \rho^{best_i}_{local}$
$\vec{\rho}^{i}_{local} \rightarrow \vec{x}_i, \rho^{best_i}_{local} \rightarrow f(\vec{x}_i)$
If ($f(\vec{x}_i) < \rho^{best}_{global}$**)**
$\vec{\rho}_{global} \rightarrow \vec{x}_i, \rho^{best}_{global} \rightarrow f(\vec{x}_i)$
Endif
Endif
Endfor
Endwhile

Algorithmus 1 Pseudocode der Partikelschwarmoptimierung zur Minimierung einer Fitnessfunktion

Algorithmus 1 illustriert das grundsätzliche Vorgehen der PSO. Insgesamt handelt es sich um ein sehr simples Verfahren. Bis zu dem Zeitpunkt, an dem ein bestimmtes Konvergenzkriterium erreicht ist, werden für jedes Partikel neue Geschwindigkeitsvektoren berechnet, wobei es jeweils eine Kraft zum persönlichen und eine Kraft zum globalen Optimum gibt. Die Werte für ς_{local} und ς_{global} entscheiden demnach, ob das Verfahren eher eine explorative Suche im Raum vornimmt oder der komplette Schwarm versucht, schnell zu einem globalen Optimum zu konvergieren. Falls der Gewichtungsfaktor für das globale Optimum zu groß gewählt wird, besteht eine höhere Gefahr, das tatsächliche globale Optimum der zugrundeliegenden Fitnessfunktion nicht zu erreichen, sondern in einem lokalen Optimum stecken zu bleiben.

Die Partikelschwarmoptimierung ist seit ihrer Veröffentlichung für diverse Anwendungsfälle erfolgreich evaluiert worden. Besonders beliebt ist das Verfahren zur Optimierung von Antennendesigns, insbesondere für die Steuerung von Phased-Array Antennen [25] [26] [27] [28], im biomedizinischen Kontext, beispielsweise für die Ableitung von regulatorischen Netzwerken in der Genetik [29] [30] [31] [32] oder im Kontext von Kommunikationsnetzwerken, etwa zur Optimierung des Routings [33] [34] [35]. Eine Gesamtübersicht über Applikationen, in denen Partikelschwarmoptimierungen erfolgreich zum Einsatz gekommen sind, liefert [36].

Neben den Publikationen, die Anwendungen der Partikelschwarmoptimierung für eine bestimmte Domäne präsentieren, findet man in der Literatur darüber hinaus einige Vorschläge, wie das Verfahren grundsätzlich verbessert werden kann, wenn bestimmte Bedingungen vorliegen. So wurden Vorschläge gemacht, wie mit kombinatorischen Problemen umzugehen ist. Außerdem wurde der Umgang mit Mehrzieloptimierungen behandelt.

2.1.2. Genetische Algorithmen

Eine weitere häufig genutzte Methode zur heuristischen Optimierung sind die sogenannten genetischen Algorithmen, die in die Klasse der evolutionären Algorithmen fallen. Analog zur Partikelschwarmoptimierung handelt es sich um ein Verfahren, das biologisch motiviert ist. Ebenso verläuft der Algorithmus iterativ bis zum Eintreten eines beliebigen Konvergenzkriteriums. Der Ablauf sei hier nur grob erklärt, da genetische Algorithmen im Rahmen des Projekts weniger im Fokus standen als die Partikelschwarmoptimierung. Abb. 2 skizziert das grobe Vorgehen. Zunächst werden ebenfalls zufällig verschiedene Initialkonfigurationen erzeugt, die bezüglich ihrer Fitness evaluiert werden. Dann erfolgt zunächst ein Selektionsschritt, bei dem einige sogenannte Individuen aus der aktuellen Generation ausgewählt werden, um gemeinsam Nachkommen zu erzeugen. Für die Selektion gibt es unterschiedliche Ansätze. Die einfachste Variante ist die sogenannte „Best Selection“, bei der die $k \in \mathbb{N}$ Individuen mit der besten Fitness gewählt werden. Für eine verbesserte Exploration kann das Vorgehen zugunsten einer sogenannten „Tournament Selection“ ersetzt werden, bei der auch weniger optimale Individuen einer Generation an der Erzeugung von Nachkommen beteiligt werden können. Der zweite Schritt besteht in der Rekombination. Dabei handelt es sich im Wesentlichen um die Erzeugung von Kreuzungen der Elternindividuen. Daraufhin werden verschiedene Mutationen möglich gemacht, damit der Suchraum weiträumig erfasst werden kann. Nach der kompletten Erzeugung einer neuen Generation kann eine weitere Selektion stattfinden und die Schleife beliebig oft wiederholt werden.

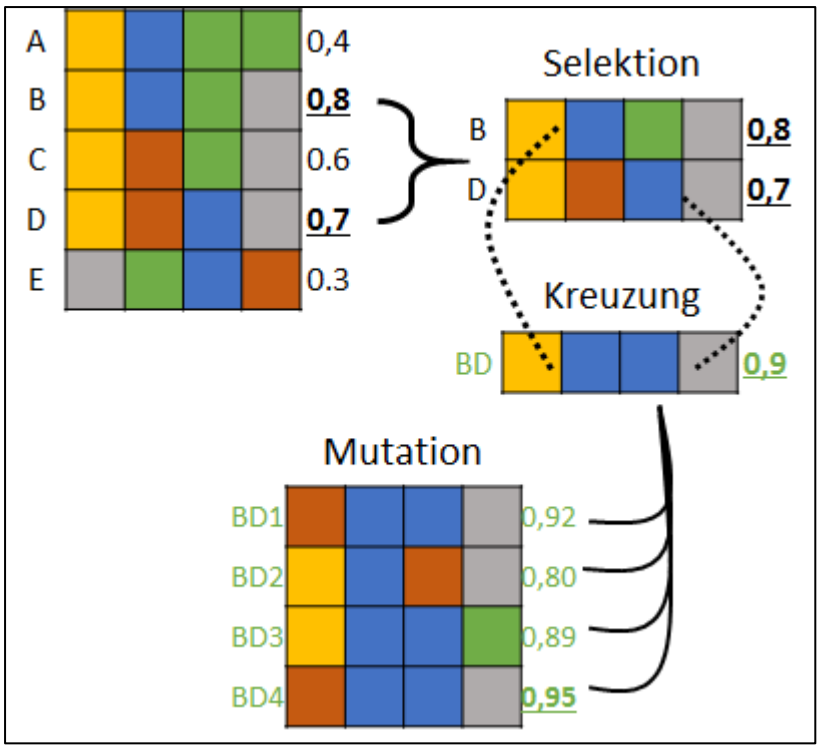

Abb. 2 Grobe Darstellung evolutionäre Algorithmen

2.1.3. Simulated Annealing

Ein weiteres verbreitetes Verfahren für die numerische Optimierung ist Simulated Annealing, wobei es sich um ein Verfahren aus der Klasse der Monte-Carlo Algorithmen handelt. Von den bisher beschriebenen Verfahren unterscheidet sich Simulated Annealing vor allem darin, dass die Optimierung nicht partikel- oder populationsbasiert, sondern lediglich durch die Suche nach dem Optimum mittels einem einzigen Punkt erfolgt. Das Verfahren wurde zuerst 1983 von Kirkpatrick et al. vorgestellt [37]. Bei diesem Algorithmus wird, wie der Name verrät, ein Abkühlungsprozess simuliert. Stellt man sich ein Medium mit sehr hoher Temperatur vor, so ist die Wahrscheinlichkeit für eine starke Unordnung innerhalb dieses Mediums sehr groß, da Moleküle zu einer hohen Beweglichkeit neigen. Die Energie innerhalb dieses sehr warmen Zustands hängt hauptsächlich von der hohen Temperatur ab, womit die Position von Atomen nahezu beliebig ist. Während eines langsamen Abkühlungsprozesses werden die Atome iterativ zu einer energetisch immer günstigeren Ordnung gelangen, womit das gesamte Medium zu einem energetisch günstigen Zustand konvergiert. Dieses Verhalten wird durch Simulated Annealing zwecks Optimierung nachgebildet.

Sei $f: \mathbb{R}^d \to \mathbb{R}$ eine Fitnessfunktion, $\vec{x} \in \mathbb{R}^d$ die initiale Konfiguration, $T_i : i \in [0, n]$ eine monoton fallende Folge von Temperaturen mit $T_0 := T_{Start} \gg 0$ und $T_n \approx 0 \,\&\, T_n > 0$. Außerdem sei $U: \mathbb{R}^d \to \mathcal{P}(\mathbb{R}^d)$ eine Abbildung, die eine Menge potentieller Nachbarn für die Dimensionalität des Problems erzeugt. $\mathcal{P}(\mathbb{R}^d)$ beschreibt hierbei die Potenzmenge über den d-dimensionalen Suchraum. Dann verläuft das Vorgehen wie folgt: Zuerst wird die Fitness für den zufälligen Initialvektor $f(\vec{x})$ berechnet. Im

Anschluss wird ein neuer Punkt in unmittelbare Nachbarschaft von $\vec{x}$ gesucht durch $\vec{y} = U(\vec{x})$. Nun findet die Prüfung statt, ob $f(\vec{y}) < f(\vec{x})$. Falls der Wert aus der Nachbarschaft zu einer verbesserten Fitness führt, wird dieser Wert als aktuelles Optimum akzeptiert, so dass $\vec{x} = \vec{y}$ gesetzt wird. Ist die Fitness des alten Werts besser, kann der neue Zustand trotzdem akzeptiert werden mit der Wahrscheinlichkeit:

$$p_{akzeptiere_y} = \exp\left(-\frac{f(\vec{y}) - f(\vec{x})}{T_i}\right)$$

Auf diese Art und Weise können während der Iteration auch Transitionen in energetisch ungünstigere Zustände erfolgen. Das hat den Grund, dass ein einseitiges Akzeptieren von verbesserten Energietermen bei ungünstiger Initialisierung fast sicher dazu führt, dass das Verfahren gegen ein lokales Optimum konvergiert.

Auch Simulated Annealing wird in verschiedenen Szenarien bis heute erfolgreich verwendet.

2.2. Effiziente G-Code Interpretation

Im Folgenden soll es darum gehen, aufzuzeigen, wie aus NC-Programmen relevante Informationen extrahiert werden können, ohne komplexe Simulationen durchführen zu müssen. Die Idee ist zunächst hauptsächlich entstanden, um im Vorfeld einer Simulation Kennzahlen zu ermitteln, die für die Maschinenauswahl wichtig sind. Es hat sich allerdings gezeigt, dass eine solche Vorinterpretation ebenso interessant für die Optimierung einer einzelnen Simulation ist. Das grundlegende Programm für diese Vorinterpretation wurde von B. Jurke entwickelt [38]. Die Herangehensweise unterscheidet sich in der Auswahl der betrachteten Befehle und den Randbedingungen von der Formalisierung, die im Anschluss vorgestellt wird. Sämtliche Betrachtungen, die zur mathematischen Formulierung einzelner NC-Sätze ausgeführt werden, beruhen auf der Annahme, dass die drei Linearachsen orthogonal zueinander stehen. Zur Illustration von Beispielen wird im Folgenden das Programmierhandbuch der Firma Siemens für die Sinumerik 840D sl/828D zugrunde gelegt [39].

Zunächst stellt sich die Frage, welche Kennzahlen auf Basis des NC-Programms interessant im Sinne einer Optimierung sein könnten.

- **Fertigungsdauer**: Abschätzung der Dauer für die Bearbeitung eines Werkstücks mit dem vorliegenden NC-Programm auf Basis von Verfahrwegen und Vorschubregelungen. Dieser Aspekt steht im Zentrum der Betrachtungen innerhalb des Projekts inklusive dieser Arbeit.
- **Maximale Auslenkungen der einzelnen Achsen**: Es stellt sich die Frage, wie groß der Abstand der beiden äußersten angefahrenen Positionen jeder einzelnen Achse ist. Das Optimierungspotential kann dadurch entstehen, dass durch Verkleinerung des Aktionsraums andere Maschinen für die Durchführung des NC-Programms infrage kommen könnten.

- **Gesamtstrecke jeder einzelnen Achse:** Der Aspekt kann für eine Optimierung interessant werden, falls das Optimierungsziel der Energieverbrauch für die Durchführung des NC-Programms sein soll. So erfordert eine Veränderung des Werts der X-Achse eine Bewegung des Tisches, was bei schweren Werkstücken ein hohes Maß an kinetischer Energie bedeutet. Da die Massen für Werkzeuge, Spindel und Werkstück üblicherweise nicht bekannt sind, kann eine solche Optimierung lediglich auf Annahmen beruhen und wird in den späteren Experimenten nicht genauer betrachtet. Nichtsdestotrotz wird durch die Formalisierung des G-Codes der Weg in die Richtung gebahnt, wodurch nachfolgende Arbeiten diese Zielstellung genauer verfolgen könnten.

Es stellt sich nun die Frage, wie der Algorithmus für die Vorinterpretation des G-Codes umgesetzt werden kann. Dazu kann man auf abstrakter Ebene veranschaulichen, wie das Vorgehen aussieht. Dazu muss zunächst eine Reihe an Begriffen formal definiert werden. Ein NC-Programm führt sequentiell Befehle aus, die als NC-Sätze bezeichnet werden.

Formal wird an dieser Stelle ein *parametrierter NC-Satz* definiert als 4-Tupel $\Pi := \{\beta, \kappa, \sigma, \delta\}$, wobei β genau der Befehlseingabe des NC-Programms entspricht. Die aktuelle Position des Werkzeugs in Bezug auf das Werkstückkoordinatensytem wird mit $\kappa \in \mathbb{R}^3$ angegeben. $\sigma \in \mathbb{R} \times (\mathbb{R}^3, \mathbb{R}^3) \times \mathbb{R}^3$ gibt den jeweils aktuellen Wert der oben beschriebenen zu ermittelnden Kennzahlen an. Das Tupel mit den jeweils dreidimensionalen Vektoren gibt hierbei die maximale und die minimale Auslenkung für die jeweiligen Achsen an. Die Orientierung des Werkstückkoordinatensystems wird als Matrix mit $\delta \in \mathbb{R}^{3\times3}$ angegeben.

Die Anwendung eines parametrierten NC-Satzes erhält die Bezeichnung *NC-Transition* und wird definiert als Abbildung $\nu: \Pi \rightarrow \beta^+: \Pi'$, wobei Π' das Tripel $\{\kappa, \sigma, \delta\}$ ausdrückt mit $\kappa \in \mathbb{R}^3$, $\sigma \in \mathbb{R} \times (\mathbb{R}^3, \mathbb{R}^3) \times \mathbb{R}^3$ und $\delta \in \mathbb{R}^{3\times3}$ analog zu den Parametern in Π. Darüber hinaus ist β^+ der nachfolgende NC-Befehl. Es handelt sich also um eine Abbildung, die einen NC-Befehl und den aktuellen Status für den gegebenen Zeitpunkt entgegennimmt und einen neuen Status zurückgibt.

Sei nun $\mathrm{B}_n := \{\beta_1, \beta_2, \dots, \beta_n\}$ eine Folge von Befehlen eines NC-Programms der Länge n und Π' eine beliebige Startkonfiguration. Zusätzlich wird $\mathrm{B}_{i:n} \subseteq \mathrm{B}_n, 1 \leq i < n$ als die Teilfolge von B_n notiert, die die ersten $i-1$ Befehle ausspart, also $\{\beta_i, \beta_{i+1}, \dots, \beta_n\}$. Darüber hinaus sei der Befehl zum Beenden eines NC-Programms notiert mit $\tilde{\beta}$. Dann ist $\widetilde{\mathrm{B}} := \{\tilde{\beta}\}$ eine Folge, die lediglich aus dem Schlussbefehl besteht. Dann kann die Auswertung eines NC-Programms durch folgende Rekursion beschrieben werden:

$$\mathrm{A}: (B_n, \Pi') \rightarrow \kappa$$

$$A\left(\widetilde{\mathrm{B}}, \Pi'\right) = \kappa_{\Pi'}$$

$$A(\mathrm{B_{i:n}}, \Pi') = A(\mathrm{B}_{i+1:n}, \nu(\beta_\mathrm{i}: \Pi'))$$

Der Doppelpunktoperator kann hierbei als Abbildung verstanden werden, die für einen NC-Befehl β und ein Tripel $\Pi' := \{\kappa, \sigma, \delta\}$ das 4-Tupel $\Pi := \{\beta, \kappa, \sigma, \delta\}$ zurückgibt.

Die beschriebene Operation bildet die Grundlage für die Berechnung der beschriebenen Kennzahlen auf Basis eines NC-Programms. Offensichtlich besteht die Komplexität in der Fragestellung darin, zu ermitteln, wie die Transition ν in der Praxis umgesetzt wird. Offensichtlich muss hierzu jeder mögliche NC-Befehl separat betrachtet werden. Hierzu seien zunächst die grundlegenden Wegbefehle betrachtet.

Tabelle 1 Übersicht über die grundlegenden G-Befehle

Befehl	**Bedeutung**
G0	Bewegung im Eilgang
G1	Lineare Bewegung mit definiertem Vorschub
G2	Kreisbewegung im Uhrzeigersinn mit definiertem Vorschub
G3	Kreisbewegung gegen den Uhrzeigersinn mit definiertem Vorschub

Zunächst muss der parametrisierte NC-Satz initialisiert werden. Diese Initialisierung verläuft wie folgt: Sei $\mathrm{P}_w \in \mathbb{R}^3$ der Werkzeugwechselpunkt. Dann ist $\Pi_0 := \{\beta_0, \kappa_0, \sigma_0, \delta_0\}$ mit:

$$\beta_0 = \epsilon$$

$$\kappa_0 = \mathrm{P}_w$$

$$\sigma_0 = \{0, \begin{pmatrix} \infty & \infty \\ \infty & \infty \\ \infty & \infty \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\}$$

$$\delta_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Der NC-Befehl β_0 wird mit dem Platzhalter ϵ belegt, der einen Befehl repräsentiert, der keine Auswirkung hat, also formal gesprochen:

$$\nu(\epsilon, \kappa^*, \sigma^*, \delta^*) = (\kappa^*, \sigma^*, \delta^*)$$

Als initiale Position im Bauraum wird in diesem Kontext der Werkzeugwechselpunkt angenommen, da das Programm üblicherweise mit einer Werkzeugaufnahme startet. Der Zustand σ startet jeweils mit neutralen Werten, also 0 für die Bearbeitungsdauer und die jeweiligen Gesamtstrecken der Achsen, negative Unendlichkeit für die maximale Auslenkung in Negativrichtung beziehungsweise positive Unendlichkeit für die

Auslenkung in Positivrichtung, so dass die erste Auslenkung jeder Achse als Grenzwert akzeptiert wird. Für δ_0 wird zunächst die Standardbasis im $\mathbb{R}^3$ festgesetzt.

Nun seien nach und nach die Wegbefehle betrachtet, wie sie in Tabelle 1 zu sehen sind. Der Einfachheit halber sei zunächst die Linearverarbeitung G1 betrachtet, also die NC-Transition:

$$\nu_{G1} \coloneqq \nu(G1, \kappa, \sigma, \delta) \rightarrow (\kappa', \sigma', \delta')$$

In der Praxis ist der G1-Befehl wie folgt aufgebaut:

$$\text{G1 X... Y... Z... F...}$$

Die Werte für X, Y und Z beschreiben hierbei die Zielkoordinaten und F den Vorschub in $^{mm}/_{min}$. Für die Geradeninterpolation lässt sich relativ leicht erfassen, was das für die NC-Transition ν_{G1} bedeutet:

$$\kappa' = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\sigma' = \left(\frac{\|\kappa - \kappa'\|_2}{F}, \left(\begin{pmatrix} \min(X, \sigma_X) \\ \min(Y, \sigma_Y) \\ \min(Z, \sigma_Z') \end{pmatrix}, \begin{pmatrix} \max(X, \sigma_X) \\ \max(Y, \sigma_Y) \\ \max(Z, \sigma_Z') \end{pmatrix} \right), \quad \Sigma + \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \right)$$

$$\delta' = \delta$$

Demnach ändert sich κ dahingehend, dass der Zielwert übernommen wird. Die Orientierung δ bleibt konstant. Der Status σ wird auf sämtlichen Ebenen aktualisiert. Die Bearbeitungsdauer für den NC-Satz wird durch einen Quotienten aus einer Strecke und einer Geschwindigkeit berechnet. Im Zähler des Quotienten steht der euklidische Abstand zwischen Start- und Zielposition des Befehls. Der Nenner enthält den definierten Vorschub. Die maximalen Auslenkungen der jeweiligen Achsen werden aktualisiert, indem lediglich die Zielwerte betrachtet werden. Aufgrund der linearen Interpolation können keine neuen Grenzwerte außer den Zielwerten selbst entstehen. Zu beachten ist, dass in der Formalisierung eine neue Schreibweise eingeführt worden ist. Die Variablen σ_X, σ_Y und σ_Z beschreiben offensichtlich die aktuellen Grenzwerte. Die kumulierten Bewegungen für jede Achse lassen sich im linearen Fall ebenfalls ziemlich einfach berechnen. Hierbei handelt es sich um einfache Vektoraddition der aktuellen kumulierten Bewegungen und dem Vektor, der die Zielposition repräsentiert. Von diesem Moment an wird diese kumulierte Summe als Σ notiert.

Programmcode	Kommentar
N10 G17 S400 M3	; Wahl der Arbeitsebene, Spindel rechts
N20 G0 X20 Y20 Z2	; Anfahren der Startposition
N30 G1 Z-2 F40	; Zustellen des Werkzeugs
N40 X80 Y80 Z-15	; Fahren auf einer schräg liegenden Geraden
N50 G0 Z100 M30	; Freifahren zum Werkzeugwechsel

Abb. 3 Programmierbeispiel für Geradeninterpolationen und Eilgänge

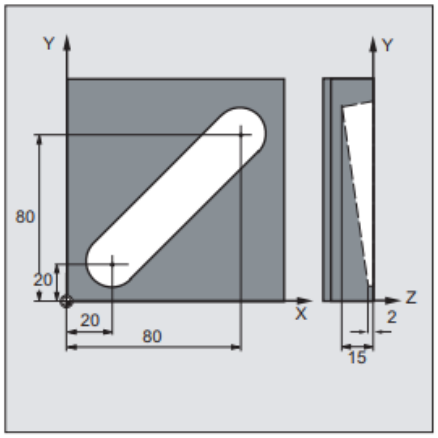

Abb. 4 Grafische Repräsentation eines Nutfräsvorgangs laut vorherigem Programmbeispiel

Abb. 3 und Abb. 4 zeigen exemplarisch, wie die Programmierung von Geradeninterpolationen sich auf die Zielgeometrie auswirkt. Beim NC-Satz *N20* wird zunächst die Startposition im Eilgang angefahren. *N30* zeigt dann, wie lediglich die Z-Position sich zwecks Anstellens des Werkzeugs mit einem definierten Vorschub verändert. Danach findet eine Geradeninterpolation auf sämtlichen Linearachsen statt. Hierbei ist eine Anmerkung wichtig. Der definierte Vorschub aus *N30* ist modal wirksam. Das heißt, dass dieser Vorschub so lange gilt, bis dieser Wert durch einen anderen NC-Satz überschrieben wird.

Nach der formalen Beschreibung des Vorgehens bei G1 ist es sehr einfach, die Transition ν_{G0} zu definieren. Sei F_{max} der maximale Vorschub, der von einer gegebenen Werkzeugmaschine ausgeführt werden kann. Dann gilt:

$$\nu_{G0} \coloneqq \nu_{G1}, mit\ F = F_{max}.$$

Insofern entspricht die Interpolation für den Eilgang einer einfachen Geradeninterpolation mit maximalem Vorschub. Üblicherweise ist F_{max} in der Maschinensteuerung hinterlegt.

Die Kreisinterpolation gestaltet sich deutlich komplexer als die Geradeninterpolation, was einerseits damit zusammenhängt, dass sich Kreise grundsätzlich analytisch schwerer

fassen lassen, und andererseits aufgrund der Tatsache, dass es für die Implementierung von Kreisinterpolationen diverse Möglichkeiten gibt. Dazu sei im Folgenden ein Beispiel angebracht.

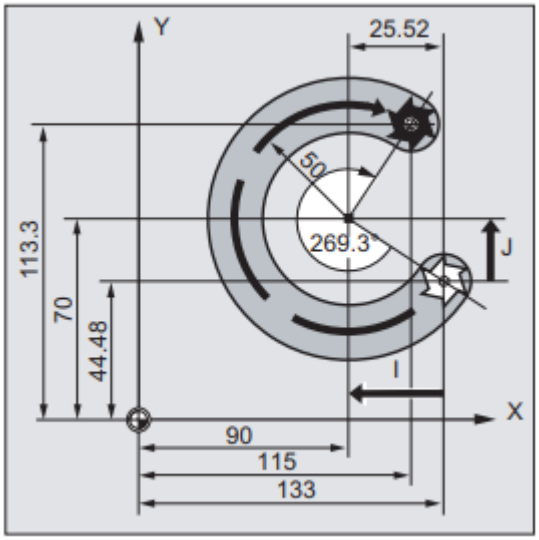

Abb. 5 Beispiel für eine Kreisinterpolation im Uhrzeigersinn

Programmcode	Kommentar
N10 G0 G90 X133 Y44.48 S800 M3	; Startpunkt anfahren
N20 G17 G1 Z-5 F1000	; Zustellen des Werkzeugs
N30 G2 X115 Y113.3 I-43 J25.52	; Kreisendpunkt, Mittelpunkt im Kettenmaß
N30 G2 X115 Y113.3 I=AC(90) J=AC(70)	; Kreisendpunkt, Mittelpunkt im Absolutmaß
N30 G2 X115 Y113.3 CR=-50	; Kreisendpunkt, Kreisradius
N30 G2 AR=269.31 I-43 J25.52	; Öffnungswinkel, Mittelpunkt im Kettenmaß
N30 G2 AR=269.31 X115 Y113.3	; Öffnungswinkel, Kreisendpunkt
N30 N30 CIP X80 Y120 Z-10	; Kreisendpunkt und Zwischenpunkt:
I1=IC(-85.35) J1=IC(-35.35) K1=-6	; Koordinaten für alle 3 Geometrieachsen
N40 M30	; Programmende

Abb. 6 Verschiedene Möglichkeiten zur programmatischen Umsetzung von Kreisinterpolationen im NC-Code

Zunächst zeigt Abb. 5 die Darstellung einer Kreisinterpolation im Uhrzeigersinn in einer zweidimensionalen Betrachtung. Hierbei sind verschiedene Kenngrößen eingezeichnet, die der formalen Beschreibung der durchzuführenden Kreisinterpolation dienlich sind. Betrachtet man im Anschluss Abb. 6, stellt man fest, dass die Bezeichnung *N30* redundant auftritt. Das hängt damit zusammen, dass hier sechs verschiedene Möglichkeiten erfasst sind, eine Kreisinterpolation zu implementieren. Um zu beantworten, wie man damit umgehen kann, wird zunächst das grundsätzliche Vorgehen für Kreisinterpolationen erläutert. Im Anschluss wird gezeigt, wie die Berechnung der gewünschten Kennzahlen für die einzelnen Implementierungen konkret umgesetzt wird. Für die grundsätzliche Betrachtung kann man zunächst auf eine bewährte Möglichkeit

zurückgreifen, einen Kreis formal zu definieren. Sei $(x_M, y_M) \in \mathbb{R}^2$ der Mittelpunkt eines Kreises auf einer zweidimensionalen Fläche. Sei zudem $r \in \mathbb{R}$ der Radius des Kreises und $\gamma \in [0,2\pi]$. Dann lässt sich ein Kreis in parametrierter Form darstellen als:

$$x = x_M + r\cos(\gamma)$$

$$y = y_M + r\sin(\gamma)$$

Auf diese Art und Weise lässt sich jeder Punkt (x, y) auf einem Kreis bei gegebenem Mittelpunkt und Radius eindeutig über die Positionierung auf dem Kreisbogen darstellen. Aus diesem Grund liegt φ im Intervall zwischen 0 und 2π. Das entspricht genau dem Umfang eines Kreises im Bogenmaß. Das Vorgehen sieht grundsätzlich so aus, die Kreisinterpolation ausgehend von der Startposition zu einer großen Anzahl an Geradeninterpolation zu approximieren, indem in kleinen Schritten zwischen φ_{Start} und φ_{Ziel} iteriert wird. Das hat neben der Einfachheit der Berechnung der Transitionen den Vorteil, dass G2 und G3 grundsätzlich auf die gleiche Weise behandelt werden können. Es ändert sich lediglich die Iterationsrichtung.

Nun sei zunächst die erste Implementierung von *N30* laut Abb. 6 betrachtet. Der NC-Satz lautet:

N30 G2 X115 Y133.3 I-43 J25.52

Semantisch lässt sich das auf folgenden Art und Weise formulieren. Ausgehend von der aktuellen Position soll die Zielposition $(115,133.3)$ im Uhrzeigersinn angefahren werden, wobei die Differenz vom aktuellen Punkt zum Mittelpunkt in X-Richtung -43 Größeneinheiten (I) und in Y-Richtung 25.52 Größeneinheiten (J) beträgt. Der Mittelpunkt wird hier also nicht durch Absolutwerte, sondern implizit durch einen Abstand ausgedrückt. Daraus folgt für diesen Fall:

$$x_M = \kappa_X + I$$

$$y_M = \kappa_Y + J$$

Der Radius des Kreises lässt sich dann problemlos als Abstand zwischen Mittelpunkt und Startpunkt berechnen, also:

$$r = \left\| \begin{pmatrix} \kappa_X \\ \kappa_Y \end{pmatrix} - \begin{pmatrix} x_M \\ y_M \end{pmatrix} \right\|_2$$

Von Interesse sind wie bereits angedeutet die Positionen des Start- und Zielpunkts auf dem Kreisbogen. Bei Betrachtung der parametrierten Form des Kreises könnte man also folgende Zuordnung berechnen:

$$x = x_M + r\,cos(\gamma)$$

$$x - x_M = r\cos(\gamma)$$

$$\frac{x - x_M}{r} = \cos(\gamma)$$

$$\gamma = \arccos\left(\frac{x - x_M}{r}\right)$$

Analog käme man bei atomarer Betrachtung der Y-Richtung zu dem Ergebnis:

$$\gamma = \arcsin\left(\frac{y - y_M}{r}\right)$$

Es braucht allerdings nur ein einfaches Beispiel, um zu sehen, dass eine einfache Betrachtung nicht hinreichend ist. Dazu sei der Parameter für die inversen trigonometrischen Funktionen betrachtet. Der Wertebereich etwa für den Bruch $x - x_M/r$ liegt im Intervall $[-1,1]$. Selbiges gilt für die Betrachtung von y. Die Werte für x_M & r sind fix. Betrachten sei der Bruch als Funktion:

$$f(x) = \frac{x - x_M}{r}$$

Dann ist x offensichtlich der einzige Freiheitsgrad. Stellt man sich nun vor, dass man durch die Kreisinterpolation potentiell einen vollständigen Kreis betrachtet, erfüllt $f(x)$ nicht mehr die notwendigen Bedingungen, um als Funktion zu gelten, da bei atomarer Betrachtung der X-Achse zwei verschiedene Y-Werte als Ausgabe stehen.

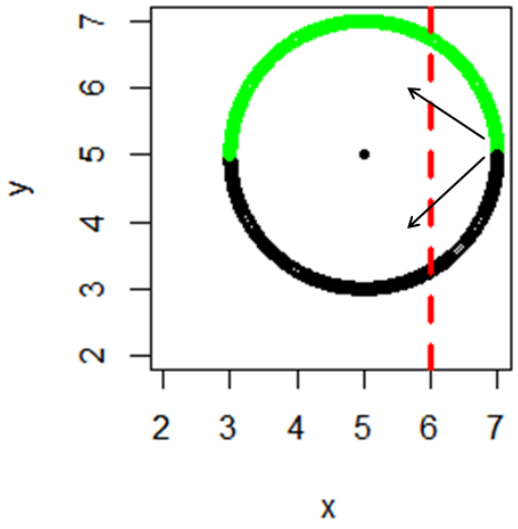

Abb. 7 Exemplarische Kreisdarstellung. Atomar betrachtete Halbkreise bilden jeweils Funktionen. Pfeilrichtungen geben die Orientierung von arccos für die jeweiligen Kreise an

Diese Tatsache wird auf Abb. 7 veranschaulicht. Von dieser sehr einfachen Grafik lassen sich zusätzlich Aspekte ableiten, die als Hilfestellung für das Vorgehen dienen. Es zeigt sich nämlich, dass sich durch einen Schnitt mit einer Geraden, die durch den Mittelpunkt und parallel zur X-Achse verläuft, zwei Halbkreise bilden lassen, für deren Definitionsbereich die Funktion $f(x)$ jeweils die Bedingungen für eine Formel erfüllt. Der

Wertebereich liegt wie bereits erwähnt im Intervall $[-1,1]$. Die schwarzen Pfeile sollen verdeutlichen, dass für die Orientierung ausgehend von $\gamma = 0 := 2\pi$ für die beiden Funktionen unterschiedlich betrachtet werden müssen. Schließlich kommt man auf die Gleichung:

$$\gamma = \begin{cases} \arccos\left(\frac{x - x_M}{r}\right), falls\ y \leq y_M \\ 2\pi - \arccos\left(\frac{x - x_M}{r}\right), andernfalls \end{cases}$$

Die Betrachtung von y dient also lediglich der Unterscheidung der beiden Halbkreise. Da für den oben beschriebenen Fall die Start- und Zielkoordinaten bekannt sind und die Koordinaten für den Mittelpunkt sowie der Radius aus den Abständen zum Mittelpunkt abgeleitet werden konnten, können auf die Art und Weise also jeweils die Werte für φ durch oben genannte Formel berechnet werden.

Sei nun der zweite Fall laut Abb. 6 betrachtet. Dort wird die Kreisinterpolation angegeben durch:

N30 G2 X115 Y113.3 I=AC(90) J=AC(70)

Dieser Fall unterscheidet sich vom vorherigen lediglich dadurch, dass der Mittelpunkt des Kreises in absoluten Koordinaten angegeben ist. Der Radius lässt sich nun einfach berechnen durch den euklidischen Abstand zwischen Start- oder Zielkoordinaten und dem genannten Mittelpunkt:

$$r = \left\| \begin{pmatrix} \kappa_X \\ \kappa_Y \end{pmatrix} - \begin{pmatrix} I \\ J \end{pmatrix} \right\|_2$$

Wie oben gesehen, reichen Mittelpunkt und Radius aus, um φ für die beiden Koordinaten zu berechnen. Insofern entspricht das weitere Vorgehen dem bisher Beschriebenen.

Der dritte Fall lautet wie folgt:

N30 G2 X115 Y113.3 CR=-50

Offensichtlich werden hier lediglich der Radius CR und die Zielkoordinate (X, Y) angegeben. Gesucht ist hier demnach der Mittelpunkt, der sich daraus ergibt. Die Koordinate des Mittelwerts ergibt sich als Schnittpunkt zweier Kreise, die mit dem angegebenen Radius um jeweils die Start- und die Zielkoordinate gezogen werden. Dazu sei die klassische Kreisgleichung betrachtet:

$$(x - x_M)^2 + (y - y_M)^2 = r^2$$

Sei nun (κ'_X, κ'_Y) die Zielkoordinate. Dann müssen hier folgende Kreise betrachtet werden:

$$k_{Ziel} = (x - \kappa'_X)^2 + (y - \kappa'_Y)^2 = CR^2$$

$$k_{Start} = (x - \kappa_X)^2 + (y - \kappa_Y)^2 = CR^2$$

Dann ergibt sich die sogenannte Kreischordale $\mathcal{C}$ durch:

$$\mathcal{C} = k_{Ziel} - k_{Start}$$

Diese Chordale entspricht einer Geraden, die $x = my + n$ annehmen kann. Einsetzen dieser Form in eine der ursprünglichen Kreisgleichungen führt nach Auflösung nach y und folgendem Auflösen nach x zu potentiell zwei Schnittpunkten. Daraufhin muss entschieden werden, welcher dieser beiden Punkte als Mittelpunkt für die gewünschte Kreisinterpolation infrage kommt. Für diesen Zweck braucht man das Vorzeichen des angegeben Radius. Berechnet man den einfachen Abstand $x_M - \kappa_X$ wird für den ersten potentiellen Mittelpunkt ein positiver und für den zweiten potentiellen Mittelpunkt ein negativer Wert das Ergebnis sein. Entspricht der Differenz dem Vorzeichen von CR, wird dieser Mittelpunkt ausgewählt. Auf der Basis kann nach aufgrund der Bekanntheit des Radius und des Mittelpunkts wie bekannt fortgefahren werden.

Die Betrachtung der nächsten Implementierung für die Kreisinterpolation laut Abb. 6 zeigt einen alternativen Ansatz:

N30 G2 AR=269.31 I=43 J25.52

In diesem Fall werden die Zielkoordinaten nicht explizit angegeben, sondern nur die Größe des Öffnungswinkels (wie auf Abb. 5 zu sehen ist). Für die Kenngrößenbestimmung muss neben den Parametern für die parametrierte Kurve hier also auch die Zielkoordinate berechnet werden. Mittelwert und Radius zu berechnen folgt analog zum vorherigen Vorgehen bei Repräsentation im Kettenmaß. Da die Startkoordinate bekannt ist, kann also γ für die Startposition problemlos ermittelt werden. Es existiert ein direkter, linearer Zusammenhang zwischen Winkelmaß und Gradmaß, weswegen die Position für die Zielkoordinate ebenfalls sehr einfach berechnet werden kann. Es gilt:

$$n^\circ := n^\circ * \frac{\pi}{180^\circ}\ rad$$

Als Abbildung würde man demnach formulieren:

$$\rho(n) = n * \frac{\pi}{180}$$

Insofern ergibt sich die Zielkoordinate direkt aus der Parameterdarstellung des Kreises:

$$x = x_M + r * \cos(\gamma + \rho(AR))$$

$$y = y_M + r * \sin(\gamma + \rho(AR))$$

Die Entsprechung dieser Implementierung der Kreisinterpolation durch den folgenden NC-Satz beinhaltet die Zielkoordinate, was wiederum die Berechnung des Radius und des Mittelpunkts erforderlich macht.

$$\text{N30 G2 AR=269.31 X115 Y113.3}$$

Dazu wird eine weitere Größe eines Kreises benötigt. Als *Kreissehne* wird für zwei gegebene Punkte, die auf dem Kreis liegen, die Sekante bezeichnet, die diese beiden Punkte verbindet.

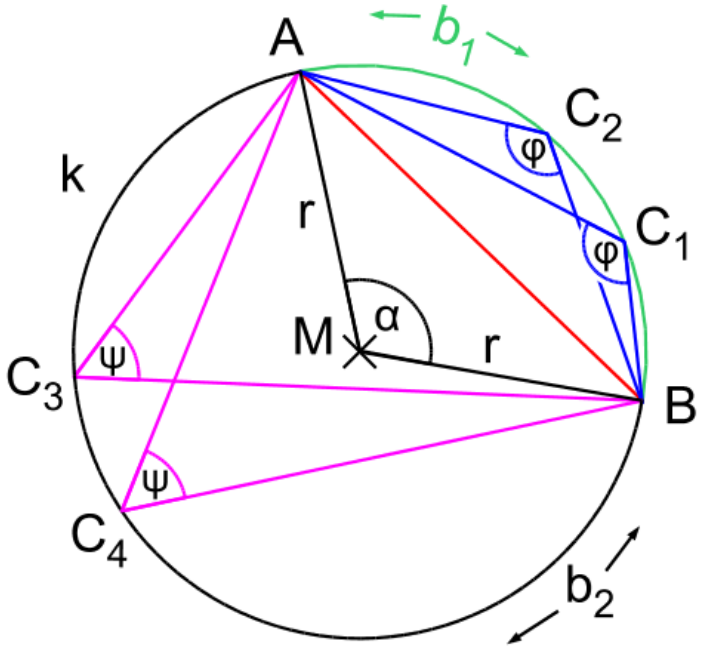

Abb. 8 Exemplarische Darstellung von verschiedener Kreissehnen [40]

Abb. 8 zeigt exemplarisch einige Kreissehnen, wobei für den vorliegenden Kontext insbesondere die Verbindung der beiden Punkte A & B relevant ist. Für die Länge dieser Kreissehne gilt:

$$s = 2r * \sin\left(\frac{\alpha}{2}\right)$$

Liegt also ein NC-Satz wie oben beschrieben vor, sind Start- und Zielkoordinate bekannt. Somit lässt sich deren Abstand s' als $\|\kappa' - \kappa\|_2$. Der Winkel α' ergibt sich durch:

$$\alpha' = \begin{cases} AR, falls\ AR < 180° \\ 360° - AR, andernfalls \end{cases}$$

Somit ließe sich der Radius berechnet durch:

$$r = \frac{s'}{2 \sin\left(\frac{\alpha'}{2}\right)}$$

Die Mittelpunktberechnung kann dann wieder nach dem bekannten Schema durchgeführt werden.

Durch diese Formalismen ist demnach die Grundlage gelegt, die einfachen G-Befehle G0, G1, G2 und G3 effizient interpretieren zu können. Erwähnt sei an der Stelle noch der Befehl G4, der einfach eine Verweilzeit angibt. Das kann beispielsweise folgendermaßen aussehen:

N50 G4 F25

Dieser Befehl bedeutet einfach, dass die Bearbeitung für 25 Sekunden am aktuellen Ort verweilt. Ohne es zu formalisieren, wird hier zur Vollständigkeit angegeben, dass diese 25 Sekunden auf die Bearbeitungsdauer addiert werden.

Stand jetzt wurden nur einfache G-Befehle betrachtet. Für die Analyse wird allerdings zusätzlich ein M-Befehl interpretiert, der M6-Befehl. Dieser Befehl teilt der Steuerung mit, dass ein Werkzeugwechsel ausgelöst werden soll. Demnach entspricht die Interpretation dieses Befehls einer Gradeninterpolation im Eilgang (G0) vom aktuellen Punkt zum **Werkzeugwechselpunkt.**

Bei der bisherigen Betrachtung fällt auf, dass die Datenstruktur für den aktuellen Zustand stets die Matrix δ mitführt, die Stand jetzt noch nicht näher beleuchtet wurde. Es stellt sich die Frage, unter welchen Umständen sich diese verändert und welche Auswirkungen das auf die Interpretation des G-Codes hat. Dazu muss man sich ein Konzept ansehen, mit dessen Hilfe gleiche Geometrien auf verschiedenen Flächen im Bauraum gefräst werden können, ohne dabei den G-Code zu verändern. Hierbei handelt es sich um sogenannte Frames. Diese erlauben es, die Orientierung der Referenzbasis für das Werkstückkoordinatensystem zu verändern. In der Praxis sieht das beispielsweise folgendermaßen aus:

N60 ROT Z60

Durch diesen NC-Satz wird das Referenzkoordinatensystems des Werkstücks um 60° um die Z-Achse gedreht. Für die NC-Transition $\nu_{ROT_Z} \coloneqq \nu(ROT(Z_\theta), \kappa, \sigma, \delta) \rightarrow (\kappa', \sigma', \delta')$

bedeutet das demnach:

$$\kappa' = \kappa$$

$$\sigma' = \sigma$$

$$\delta' = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Das entspricht offensichtlich genau der Rotationsmatrix um die Z-Achse im kartesischen Koordinatensystem. Neben dem Schlüsselwort „ROT“ existiert darüber hinaus die Möglichkeit, eine Drehung des Koordinatensystems durch das Schlüsselwort „AROT“ zu erzeugen. Der Aufruf erfolgt analog:

N60 AROT Z60

Die Transition $\nu_{AROT_Z} := \nu(ROT(Z_\theta), \kappa, \sigma, \delta) \rightarrow (\kappa', \sigma', \delta')$ unterscheidet sich lediglich in einem Detail:

$$\kappa' = \kappa$$

$$\sigma' = \sigma$$

$$\delta' = \delta * \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Bei „AROT“ handelt es sich um eine additive Rotation des Referenzkoordinatensystems, während „ROT“ eine absolute Rotation um die Standardbasis vornimmt und jegliche vorher definierte Rotation überschreibt. Die Wirksamkeit von δ wird hergestellt, indem jeder Positionsvektor vor der Interpretation mit dem aktuellen Wert von δ multipliziert wird.

Es ist zu erwähnen, dass es sich bei obiger Betrachtung um eine Vereinfachung handelt, die einige Aspekte der NC-Bearbeitung nicht berücksichtigt. So wird das komplette Spektrum von Standardzyklen vollständig ignoriert. Das hängt damit zusammen, dass die Anzahl vordefinierter Zyklen deutlich zu unübersichtlich ist, um eine Abbildung zu rechtfertigen. Darüber hinaus zeigt eine grobe Durchsicht verfügbarer NC-Programme sehr schnell, dass diese eingeschränkte Betrachtung reale Szenarien bereits sehr gut abzubilden vermag.

2.3. Paralleles und verteiltes Rechnen

Paralleles und verteiltes Rechnen ist seit Mitte des 20. Jahrhunderts Forschungsgegenstand und hat insbesondere innerhalb der letzten 20 Jahre durch die hohe Verfügbarkeit verteilter Rechnerressourcen und paralleler Rechnerarchitekturen massiv an Bedeutung gewonnen. Hier soll es darum gehen, einige Begrifflichkeiten zu verdeutlichen und eine Notation für parallele Berechnungen vorzustellen, die im späteren Verlauf für die Vorstellung paralleler Algorithmen benötigt wird.

Da von parallelem und verteiltem Rechnen die Rede ist, existiert offensichtlich ein Unterschied zwischen diesen Begrifflichkeiten. Paralleles Rechnen kann im Grunde genommen als Überbegriff für die Parallelisierung gewisser Aspekte eines Programms aufgefasst werden. Demnach wäre das verteilte Rechnen nur eine Art und Weise des parallelen Rechnens. Hier wird zur Vereinfachung eine scharfe Grenze zwischen den Begriffen gezogen. Ist vom parallelen Rechnen die Rede, sei hier davon ausgegangen, dass jeder Prozess auf denselben Speicher zugreifen kann. Dieses sogenannte „Shared-Memory Computing“ ist beispielsweise die Grundlage für Multithreadingsysteme, etwa bei Mehrkernprozessoren. Es gibt eine Reihe an Möglichkeiten, parallele Programme mit gemeinsamem Speicher zu implementieren. Hierzu zählen POSIX-Threads oder das Framework OpenMP. Verteiltes Rechnen basiert auf Topologien, bei denen kein zentraler, gemeinsamer Speicher zur Verfügung steht. Das ist bei jeder Art von Rechnernetzen der Fall. Die Synchronisierung der einzelnen Prozesse findet hierbei üblicherweise über Nachrichtenaustausch zwischen den einzelnen Rechnerknoten statt. Als Standard für diese Art des Programmierens hat sich das Message Passing Interface (MPI) etabliert.

Es gilt nun, eine Schreibweise vorzustellen, die parallele Programmabläufe präsentiert, die nicht von der Fragestellung möglicher Rechnerarchitekturen abhängt und offenlässt, ob deren Implementierung mit Zugriff auf geteilten Speicher oder etwa über einen Nachrichtenaustausch realisiert werden kann. Dazu wird im Pseudocode ein Block „**Prozess** π $[\phi \in \{0, \dots, \varphi - 1\}]$“ definiert. Hierbei beschreibt die Variable ϱ die Anzahl der parallelen Blöcke. Insofern handelt es sich um einen konstanten Wert, der für sämtliche Prozesse identisch ist. Der Wert ρ ist die Kennzeichnung jedes einzelnen Prozesses und zunächst die einzige Unterscheidung verschiedener Prozesse. Ein Prozessblock wird mit der Kennung „**Endprozess**“ abgeschlossen.

Algorithmus: Hallo Welt – parallel	
Prozess π $[\phi \in \{0, \dots \varphi - 1\}]$	
	print („Hallo Welt! Ich bin Prozess ϕ von φ.“)
Endprozess	

Algorithmus 2 Verdeutlichung der Schreibweise für parallelen Pseudocode durch einfaches "Hallo Welt" Programm

Ausgabe: Algorithmus 2 mit $\varphi = 4$
> Hallo Welt. Ich bin Prozess 2 von 4.
> Hallo Welt. Ich bin Prozess 1 von 4.
> Hallo Welt. Ich bin Prozess 3 von 4.
> Hallo Welt. Ich bin Prozess 0 von 4.

Abb. 9 Ausgabe des parallelen Programms

Algorithmus 2 verdeutlicht, wie das Schlüsselwort ***Prozess*** zu interpretieren ist. Eine mögliche Ausgabe der Ausführung des Programms wird in Abb. 9 präsentiert. Da die Prozesse nicht synchronisiert sind, ist die Reihenfolge der Ausgaben beliebig. Um eine

Synchronisierung in der gewählten Schreibweise ausdrückbar zu machen, wird zusätzlich das Schlüsselwort „***Barriere***“ eingeführt. Hierzu existieren Entsprechungen sowohl in OpenMP (als #pragma omp barrier) als auch in MPI (als MPI_Barrier). Eine Barriere markiert eine Art Pause für jeden Prozess, die erst zu dem Zeitpunkt beendet wird, wenn jeder Prozess diese erreicht hat.

3. Beschleunigung atomarer Simulationen durch Instanzparallelisierung

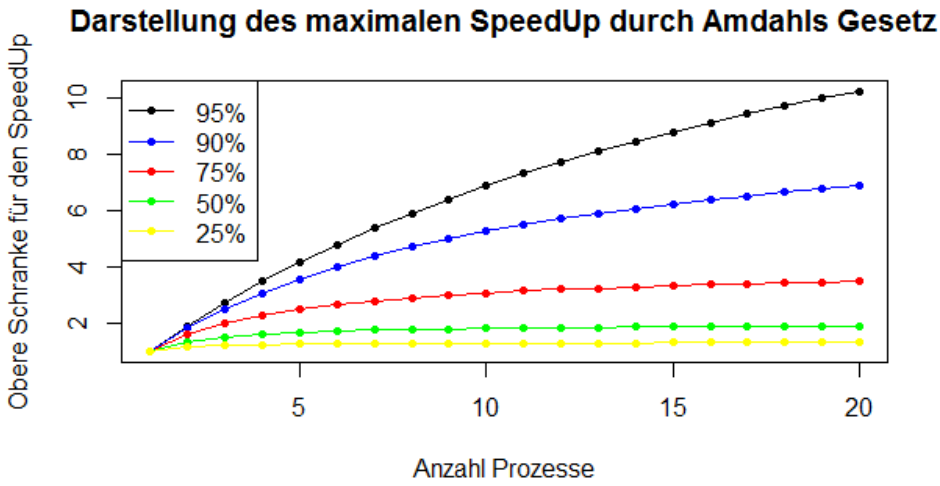

Abb. 10 Skizzierung des Gesetzes von Amdahl

Bei der Parallelisierung schwarm- bzw. populationsbasierter Verfahren liegt offensichtlich eine lineare Skalierung bezüglich der Einzelsimulationen pro Generation vor. Bei der Nutzung von Einpunktverfahren wie Simulated Annealing ist eine solche Parallelisierung offensichtlich nicht möglich. Darüber hinaus können auch bei Mehrpunktoptimierungen mehr Ressourcen zur Verfügung stehen als Individuen gewünscht sind. Daraus lässt sich direkt die Fragestellung formulieren, ob einzelne Instanzen von Simulationen simultan mehrere Ressourcen besetzen und damit durch den Einsatz von Parallelisierung beschleunigt werden können. Um das Thema Parallelisierung in diesem Zusammenhang zu motivieren, seien zunächst zwei verbreitete Gesetze aus dem Forschungsgebiet erwähnt. Zuerst spielt das Gesetz von Amdahl eine Rolle, das bereits 1967 formuliert wurde [41]. Mit diesem soll eine Abschätzung für die maximal erreichbare Beschleunigung eines Algorithmus durch den Einsatz paralleler Rechenressourcen berechnet werden. Sei T die Gesamtlaufzeit einer beliebigen Berechnung. Zusätzlich sei t_s die Laufzeit der Berechnung, die sich nicht durch Parallelisierung beschleunigen lässt und t_p die Laufzeit des parallelisierbaren Abschnitts der Berechnung, wobei offensichtlich ohne den Einsatz von Parallelisierung gilt:

$$T = t_s + t_p.$$

Nehmen wir nun an, dass n Prozesse für die Parallelisierung der Berechnung zur Verfügung stehen. Durch die Tatsache, dass der parallelisierbare Teil des Programms t_p

maximal durch den Faktor n zu beschleunigen ist, ergibt sich für die Beschleunigung S eine obere Schranke:

$$S \leq T/(t_s - t_p/n) \leq T/t_s.$$

An einem konkreten Zahlenbeispiel lässt sich einfach deutlich machen, welche Bedeutung hinter diesem Gesetz steckt. Angenommen, T entspreche 20 Sekunden und t_s sei 10 Sekunden, d.h. die Hälfte der Programmlaufzeit ließe sich durch Parallelisierung beschleunigen. Dann ist, unabhängig von der Anzahl der eingesetzten Rechenressourcen maximal die Beschleunigung $S = 2$ zu erreichen, was einer Halbierung der Programmlaufzeit entspricht. Eine Veranschaulichung dazu bietet Abb. 10. Unter Annahme verschiedener parallelisierbarer Anteile im Programm erkennt man deutlich, wie die obere Schranke für die Beschleunigung konvergiert. Die Kernaussage dieses Gesetzes lautet also, dass der limitierende Faktor für die Beschleunigung eines Programms durch Parallelisierung der Anteil der parallelisierbaren Programmabschnitte ist. Es ist offensichtlich, dass diese Einschätzung zunächst pessimistisch ist, da eine beliebig große Hinzunahme von Rechenressourcen nach einer gewissen Zeit nicht mehr zu einer effizienteren Laufzeit, sondern zu einer Sättigung der Beschleunigung führt. Aus diesem Grund ist es in dem Zusammenhang notwendig, dem Gesetz von Amdahl das Gesetz von Gustafson gegenüber zu stellen, das 1988 formuliert wurde [42] [43]. Hierfür betrachten wir zunächst relative Laufzeiten, wobei P den Anteil der parallelisierbaren Programmteile an der Gesamtlaufzeit beschreibt, womit sich der sequentielle, und damit nicht parallelisierbare, Anteil an der Laufzeit, durch $1 - P$ ausdrücken lässt. Ganz offensichtlich gilt für eine Berechnung, in der keine Parallelisierung zum Einsatz kommt:

$$1 = P + (1 - P)\ .$$

Betrachtet man auf der Basis zusätzlich analog zu Amdahls Gesetz, dass sich durch den Einsatz von n Rechenressourcen der parallelisierbare Anteil maximal um den Faktor n beschleunigen lässt, erhält man für S_n:

$$S_n = n * P + (1 - P).$$

Entscheidend ist hierbei zunächst, dass die betrachtete Beschleunigung im Gegensatz zur Beschleunigung, die durch Amdahl formuliert wurde, im direkten Zusammenhang mit der Anzahl der eingesetzten Prozesse steht. Hierbei ist zu erwähnen, dass die beiden Gesetze nicht im Widerspruch zueinander stehen, sondern eher einer unterschiedlichen Betrachtungsweise unterliegen. Während Amdahl in seiner Betrachtung davon ausgeht, dass es eine feste Problemgröße gibt, die den parallelisierbaren Anteil eindeutig bestimmt, zieht Gustafson zusätzlich die Tatsache in Betracht, dass eine höhere Skalierung der Problemgröße im Normalfall mit einer besseren Skalierbarkeit durch den Einsatz einer höheren Anzahl von Rechenressourcen einhergeht.

Inwiefern die beiden Gesetze in diesem Zusammenhang eine Rolle spielen, wird im Anschluss in die Beschreibung einer Möglichkeit zur parallelen Berechnung einzelner Simulationen genauer betrachtet. Zunächst soll es also um die Fragestellung gehen, wie

die Berechnung einer Instanz der Materialabtragssimulation umgesetzt werden kann, so dass eine befriedigende Beschleunigung erreicht wird, die zur besseren Skalierung des Gesamtsystems beiträgt. Auf den ersten Blick scheint die Möglichkeit hier nicht zu bestehen. Gewöhnliche Berechnungen, die durch Parallelisierung beschleunigt werden, zeichnen sich häufig dadurch aus, dass unabhängige Programmteile intrinsisch vorliegen. So lassen sich beispielsweise bei der Berechnung einfacher Schleifen, die in großer Anzahl in mathematischen Vektor- und Matrixoperationen Anwendung finden, problemlos in einzelne uniforme Teilschleifen separieren, die dann unabhängig voneinander berechnet und anschließend problemlos zu einem Gesamtergebnis zusammengefasst werden können. Im speziellen Fall der im vorliegenden Kontext verwendeten Materialabtragssimulation hat man es mit einem Problem zu tun, das von Natur aus durch und durch sequentiell ist. Die Information, ob zu einem bestimmten Zeitpunkt eine Kollision im Programm auftritt, kann nur plausibel ermittelt werden, wenn bekannt ist, welche Teile des ursprünglichen Werkstücks zu diesem Zeitpunkt abgetragen wurden. Sieht man sich allerdings genauer an, was der Materialabtragssimulation zugrunde liegt, erkennt man, dass sich das Problem in zwei Teilaufgaben separieren lässt, die in ähnlicher Weise auch durch die Umsetzung des Setup Optimizers (siehe Kapitel 6.3.2) adressiert werden. Formal nimmt die Berechnung des Materialabtrags in vereinfachter Form geometrische Informationen zum Werkstück vor der Verarbeitung, einer Liste verwendeter Werkzeuge und dem Spannmittel und das zu fertigende NC-Programm entgegen und gibt bei erfolgter Berechnung die Geometrie des Werkstücks nach dem Abtrag, die Bearbeitungszeit und die Information über aufgetretene Kollisionen zurück. Es sei hierbei angemerkt, dass im Sinne der Vereinfachung Informationen zur Aufspannposition und die verwendete Maschine vernachlässigt werden. Wir stellen also zunächst fest, dass wir folgende Abbildung betrachten können:

$$f: C_{w^p} \times C_s \times C_t^n \times nc \rightarrow C_{w^a} \times \mathbb{R} \times \{0,1\} \ .$$

Hierbei ist C_x als geometrische Repräsentation eines beliebigen Festkörpers zu interpretieren, wobei $x := w^p$ die geometrischen Informationen des Werkstücks vor und $x := w^a$ analog dazu die Informationen nach der Bearbeitung enthält, während $x := s$ und $x := t$ Platzhalter für Spannmittel und Werkzeuge darstellen, wobei zu beachten ist, dass in der Formulierung davon ausgegangen wird, dass n Werkzeuge für die Bearbeitung eingesetzt werden. Hierbei muss wiederum erwähnt werden, dass an der Stelle starke Vereinfachungen stattgefunden haben, da grundsätzlich auch Werkstücke, Spannmittel und NC-Programme als Mengen auftreten könnten.

Nun stellt sich die Frage, wie mit den Vorbetrachtungen ein Verfahren zur Parallelisierung entwickelt werden kann. Hierzu müssen wir annehmen, dass auf Basis des NC-Programms und der Informationen zu Werkstück- und Werkzeuggeometrien beispielsweise eine CAD-Datei erstellt werden kann, die die Bahnen, inklusive der Breiten, die der Werkzeugradius vorgibt, und der Werkzeugtiefen, darstellt. Diese

Bahninformationen müssten im Anschluss durch eine logische Mengensubtraktion mit dem Werkstück verrechnet werden, so dass die Geometrie von w^a ermittelt werden kann, ohne den Overhead, der durch die virtuelle Werkzeugmaschine entsteht, hinnehmen zu müssen. Diese Berechnung könnte dann als folgende Abbildung aufgefasst werden:

$$g: C_{w^p} \times C_s \times C_t^n \times nc \rightarrow C_{w^a} \times \mathbb{R}.$$

Logischerweise kann diese Form der Steuerungsnachbildung keine verlässliche Information darüber liefern, ob das zugrunde liegende Setting tatsächlich eine valide Maschineneinrichtung repräsentiert. Dazu ist der Einsatz der 1:1 Simulation, die neben der geometrischen Informationen beispielsweise fehlerhafte Situationen wie Softwareendschalter erkennt, nicht zu verhindern. Allerdings helfen die Darstellungen der oben beschriebenen mathematischen Abbildungen, ein Konzept zu formulieren, das den Einsatz von parallelem Rechnen sinnvoll einsetzbar macht. Sei $R := \{r_1 \dots r_m\}$ eine Menge verfügbarer Rechenressourcen und seien C_{w^p}, C_s, C_t^n, nc gegeben. Dann wird τ definiert als eine Abbildung, die ein NC-Programm in eine Menge von Teilprogrammen splittet, also:

$$\tau_m: nc \rightarrow \{nc^1 \dots nc^m\}.$$

Hierbei ist zu bedenken, dass jedes NC-Programm nc^i als einzelnes und unabhängiges NC-Programm ausführbar sein muss und dass sich die Semantik des Teilprogramms durch seine unabhängige Betrachtung nicht ändert. So sind beispielsweise Informationen über eventuelle Nullpunktverschiebungen oder Aufhebung programmierbarer Frames, die im Teilprogramm nc^{i-1} hinterlegt sind, unbedingt mitzuführen. Darüber hinaus muss beachtet werden, dass falls die Separierung zwischen einem T-Befehl, also einer Werkzeuganwahl, und dem eigentlich Werkzeugwechsel, der durch den Befehl M6 gekennzeichnet ist, durchgeführt wird, die entsprechende T-Nummer im Teilprogramm nc^i hinterlegt wird. Neben der Abbildung, die die Aufteilung eines NC-Programms repräsentiert, definieren wir $C_t^{\tau_i} \subseteq C_t^n$, als Teilmengen von Werkzeugen, die innerhalb der Bearbeitung von nc_i gebraucht werden. Auf der Basis kann eine Vorverarbeitung formuliert werden, mit deren Hilfe die Simulation sinnvoll parallelisiert werden kann. Hierzu wird zunächst für ein gegebenes NC-Programm nc und eine gegebene Anzahl an Ressourcen m τ_m berechnet. Anschließend werden aus jedem Teilprogramm nc^i die relevanten Werkzeuge extrahiert. Im einfachsten Fall kann jetzt sequentiell folgende Rechenanweisung für jede verfügbare Ressource r_i berechnet werden:

$$g_i: C_{w_i^p} \times C_s \times C_t^{\tau_i} \times nc_i \rightarrow C_{w_{i+1}^a} \times \mathbb{R}$$

Hier ist selbstverständlich nur der eigentliche Materialabtrag beschrieben. Bis jetzt ist durch das Aufteilen des NC-Programms noch kein Gewinn erzielt worden. Da nun allerdings die Geometrie des Materialabtrags zu Beginn jedes NC-Teilprogramms bekannt ist, können auf der Grundlage die entsprechenden Simulationen ebenfalls

parallel durchgeführt werden. Es kann also jeder einzelne Prozess folgende Berechnung selbstständig durchführen. Formal betrachten wir die eigentliche Simulation der Teilprogramme nun als folgende Vereinfachung:

$$f_i' : C_{w_i^p} \times C_s \times C_t^{\tau_i} \times nc_i \rightarrow \{0,1\}$$

Da die geometrischen Aspekte des Materialabtrags als Information bereits durch die beschriebene Steuerungsnachbildung vorliegen, ist nur noch die Fragestellung relevant, ob eine Kollision im Gesamtprogramm vorliegt oder nicht. Im Endeffekt gilt hierbei folgende Aussage:

$$f = 1, iff\ f_1' = 1 \wedge f_2' = 1 \ldots \wedge f_m' = 1.$$

Die Bezeichnung *iff* ist hierbei zu lesen als „wenn, und ausschließlich wenn […]“. Es muss also jedes einzelne Teilprogramm kollisionsfrei gefertigt worden sein, um schließlich von einer kollisionsfreien Fertigung des Gesamtteils auszugehen.

Es handelt sich aufgrund der Komplexität in der Umsetzung einer robusten Steuerungsnachbildung, die eindeutig die richtigen Teilgeometrien liefert, lediglich um eine theoretische Betrachtung. Aus diesem Grund ist es nun sinnvoll, zu überprüfen, welches Potential durch den Ansatz zu erwarten ist und inwiefern die Gesetze von Amdahl und Gustafson dabei eine Rolle spielen. Hierzu sollte man sich zunächst überlegen, wie groß der sequentielle Anteil an der Durchführung des gesamten Verfahrens ist. Bei der Betrachtung der Art und Weise, wie das Verfahren hier beschrieben ist, verläuft die komplette Vorverarbeitung, das heißt die Aufteilung der NC-Programme und die Berechnung der entsprechenden Materialabträge sequentiell. Grundsätzlich lässt sich die Berechnung der einzelnen Materialabträge selbst parallelisieren, da davon ausgegangen wird, dass zunächst Verfahrwege als 3D-Modell berechnet werden, die schließlich durch logische Subtraktion vom Werkstück zum eigentlichen Materialabtrag verrechnet werden. Die Berechnung dieser Verfahrwege verläuft offensichtlich ohne jegliche Abhängigkeiten. Hingegen müssen die einzelnen bearbeiteten Teilwerkstücke, die als Grundlage für die einzelnen Simulationen dienen, kaskadisch berechnet werden und sind somit dem sequentiellen Anteil anzurechnen. Es folgt für jede einzelne Simulation der übliche Workflow, der in der virtuellen Fertigung beschrieben ist. Als Grundlage hierfür muss für jeden Teilsimulationsauftrag zunächst die entsprechende Geometriedatei erstellt werden. Daraufhin werden die Server angesteuert, die Steuerung hochgefahren, die entsprechenden Sitzungen hochgeladen und übertragen und anschließend das NC-Programm ausgeführt. Diese Vorgänge verlaufen zwar unabhängig voneinander, es ist allerdings trotzdem sinnvoll, diesen Prozess als sequentiellen Anteil des Programms zu betrachten, da dieser Overhead logischerweise beim Einsatz von m Instanzen der virtuellen Werkzeugmaschine genau durch den Faktor m ansteigt. Zum Zwecke einer simplen Betrachtung wird zunächst folgendes angenommen: Es soll ein NC-Programm simuliert werden, das bei sequentieller Berechnung eine Rechendauer von 45 Minuten aufweist. Der Workflow

vom Erstellen der Geometriedatei bis zum Start der Simulation dauere 10 Minuten. Den zusätzlichen Overhead durch die Vorbereitung der einzelnen Teilprogramme, der sich nicht zuverlässig vorhersagen lässt, beziffern wir mit 5 Minuten, so dass insgesamt eine Berechnungsdauer von einer Stunde entsteht. Die eigentliche Materialabtragssimulation kann dann als parallelisierbarer Teil des gesamten Prozesses angesehen werden, der einen Anteil von 75% an der Rechendauer ausmacht. Für diesen Fall ist die maximale Beschleunigung demnach bestimmt durch:

$$\lim_{m \to \infty} (0.25 + ({}^{0.75}/_{m}))^{-1} = 4$$

Für diesen konstruierten Fall würde man also, zumindest auf den ersten Blick, maximal eine 4-fache Beschleunigung erreichen. Wenn man den konstruierten Fall etwas weiterentwickelt, kann man in vereinfachter Art und Weise davon ausgehen, dass der Overhead durch Annahme eines komplexeren NC-Programms nicht wesentlichen steigt oder möglicherweise konstant bleibt. Hier ist der Hinweis notwendig, dass diese Aussage unter keinen Umständen der Realität entsprechen muss, weil nicht ohne weiteres vorhergesagt werden kann, wie stark die Berechnung der Steuerungsnachbildung in ihrer Komplexität bei steigender Problemgröße ansteigt. Nichtsdestotrotz gehen wir hier aus Gründen der Vereinfachung davon aus, dass die Rechendauer für eine andere Simulation bei konstantem Overhead nun bei 105 Minuten liegt. Dann erhält man folgende obere Schranke:

$$\lim_{m \to \infty} (0.125 + ({}^{0.875}/_{m}))^{-1} = 8$$

Wie bereits erwähnt, sind diese einfachen Rechenbeispiele nicht als konkrete Betrachtungen realistischer Szenarien zu interpretieren. Vielmehr sollen sie zwei Dinge verdeutlichen. Der erste Punkt ist, dass die Anwendung paralleler Einzelsimulationen für die Skalierung des Gesamtsystems nur dann sinnvoll ist, wenn das System nicht ausgelastet ist. Andernfalls skaliert das Gesamtsystem in seiner vollständigen Betrachtung linear mit der Anzahl der zur Verfügung stehenden Instanzen der virtuellen Werkzeugmaschine, was einer optimalen Beschleunigung entspricht. Der zweite Punkt ist, dass bei der Auswahl der Simulationen, die innerhalb des Systems durch Parallelisierung beschleunigt werden sollen, die Betrachtung von Gustafsons Gesetz sinnvoll ist. Es sollte demnach nach Möglichkeit eine Priorisierung stattfinden, die diejenigen Simulationen auf mehrere Knoten verteilt, die eine höhere Rechenzeit erfordern. Eine grobe Information darüber liefert der vorgestellte NC-Interpreter, da davon auszugehen ist, dass die physische Bearbeitungszeit in etwa mit der Simulationszeit korreliert.

Bei genauerer Betrachtung der konkreten Problemstellung innerhalb dieses Vorhabens zeigt sich, dass gewisse Gesetzmäßigkeiten, die innerhalb des Forschungsgebiets des parallelen und verteilten Rechnens üblicherweise als unumstößlich gelten und die auch bei den Darstellungen von Amdahl und Gustafson eine zentrale Rolle spielen, hier keine

allgemeingültige Anwendung finden können. Diese basieren nämlich alle auf der Grundlage, dass durch den Einsatz von m Prozessoren maximal eine Beschleunigung um den Faktor m erreicht werden kann. Dass dieser Leitsatz hier nicht gilt, kann man sich ziemlich einfach klar machen, indem man sich die Regelmäßigkeit vor Augen führt, dass ein Bauteil nur dann kollisionsfrei gefertigt wurde, wenn alle Teilprogramme ebenfalls in der Simulation keine Kollisionen aufgewiesen haben. Das bedeutet konkret, dass hier die Umsetzung einer sogenannten *lazy evaluation* möglich ist. Es besteht also die Möglichkeit, bei Auftritt einer Kollision innerhalb eines beliebigen Teilprogramms sämtliche Teilsimulationen unmittelbar abzubrechen, da das Gesamtergebnis zu diesem Zeitpunkt nicht mehr geändert werden kann. Aus Gründen der Effizienz des Programms sollte dies auch unbedingt umgesetzt werden. Für die Betrachtung der Beschleunigung bedeutet das, dass von einer Verlangsamung im Gegensatz zum sequentiellen Programm, bis zu einer Beschleunigung, die einen Faktor, der größer ist als m, aufweist, alle Möglichkeiten bestehen. Eine Verlangsamung entsteht offensichtlich genau dann, wenn eine Kollision zu einem sehr frühen Zeitpunkt der Fertigung auftritt. In diesem Fall hätte die sequentielle Durchführung des Programms ebenfalls sehr früh abgebrochen und der Overhead durch die Vorbereitung der Parallelisierung hätte eingespart werden können. Im Gegensatz dazu kann eine massive Beschleunigung erreicht werden, falls eine Kollision im letzten Teilprogramm nc_m auftritt. Diese hypothetische Kollision träte bei sequentieller Durchführung auf einer einzelnen Instanz zu einem relativ späten Zeitpunkt auf, während der entsprechende Prozess den Zeitpunkt der Kollision im Verhältnis deutlich früher erreichen wird.

Insgesamt lässt sich festhalten, dass durch die mögliche Betrachtung der Parallelisierung einzelner Instanzen der virtuellen Werkzeugmaschine eine spannende Forschungsfrage ergibt, die jedoch in der Umsetzung innerhalb eines Produktivsystems nur bei geringer Auslastung sinnvoll eingesetzt werden kann.

4. Verfahren zur Optimierung atomarer Werkstücksimulationen

4.1. Optimierungspotenziale durch Simulationen

Im Kapitel zur effizienten Interpretation von G-Code ist ausführlich aufgezeigt worden, welche Kennzahlen für ein vorgegebenes NC-Programm berechnet werden können. Im Folgenden soll es darum gehen, welche Eingangsparameter bei der Optimierung berücksichtigt werden und welche Überlegungen dieser Auswahl zugrunde liegen. Darüber hinaus wird verdeutlicht, wie die technische Umsetzung realisiert wurde. Zuletzt wird kurz vorgestellt, welche Freiheitsgrade zur Einrichtung der CNC-Maschine zusätzlich diskutiert wurden und aus welchen Gründen diese nicht weiter betrachtet wurden.

4.1.1. Werkstücktranslation

Die einfachste Form der Veränderung einer Positionierung eines Werkstücks im Bauraum ist die Translation eines Werkstücks, also die lineare Verschiebung der drei Raumrichtungen, wobei in der realen Fertigung üblicherweise hauptsächlich die beiden Linearachsen entlang des Arbeitstisches der Maschine relevant sind, da sich hier das Werkstück zumindest theoretisch frei verschieben lässt. Nichtsdestotrotz sollte eine Verschiebung in die Richtung, die orthogonal zu der Tischfläche verläuft, nicht ignoriert werden.

Zunächst sei allerdings der Fall betrachtet, dass das Werkstück lediglich auf dem Tisch frei beweglich ist. Es stellt sich nun einerseits die Frage, wie ein solches Szenario innerhalb einer Optimierungsumgebung abgebildet werden kann und andererseits diejenige, welches Optimierungspotential man sich von einer solchen Translation erhoffen kann. Um das Optimierungspotential durch Veränderung dieser Freiheitsgrade abschätzen zu können, genügt es, sich zu überlegen, was die Bearbeitung eines verschobenen Werkstücks von der Bearbeitung des Werkstücks in der Ursprungsposition unterscheidet und vor allem, dass dieser Unterschied sich auf ein Minimum beschränkt. Während des eigentlichen Abtragvorgangs liegen offensichtlich keinerlei Unterschiede vor. Die Auslenkungen der einzelnen Bahnachsen bleiben konstant. Das lässt sich an einem Beispiel verdeutlichen. Seien $\overrightarrow{x_1^t}, \overrightarrow{x_2^t} \in \mathbb{R}^2, t \in \mathbb{N}, \overrightarrow{x_1^t} - \overrightarrow{x_2^t} \neq \binom{0}{0}$ zwei Positionen auf dem Tisch, der die aktuelle Position der Werkzeugspindel zu einem beliebigen Zeitpunkt t beschreibt. Nun sei $\vec{m} = \binom{k}{0}, k \neq 0$ eine Bewegung der Werkzeugspindel auf nur einer Achse. Offensichtlich gilt, dass $\overrightarrow{x_1^t} + \vec{m}$ sowie $\overrightarrow{x_2^t} + \vec{m}$ parallel auf dieser Achse verlaufen. Das bedeutet, dass die Positionierung des Werkstücks auf dem Tisch die Richtungen der Verfahrwege nicht beeinflusst. Basierend auf der Erkenntnis stellt sich die Frage, inwiefern die Position eines Werkstücks trotzdem in ein Optimierungsszenario integriert werden kann. Dazu sei $\mathcal{W} \in \mathbb{R}^3$ definiert als Werkzeugwechselpunkt. Wird ein neues Werkzeug benötigt, verfährt die Werkzeugspindel von der aktuellen Position zu diesem Punkt. Diese Aktion wird ausgelöst durch den NC-Befehl „M6“ ausgelöst. Sei

$p^c \in \mathbb{R}^3$die aktuelle Position der Spindel im Bauraum. Dann lässt sich der Befehl formal definieren als:

$$M6(p^c) = G0(p^c, \mathcal{W})$$

Bei einem programmierten Werkzeugwechsel verfährt demnach die Spindel im Eilgang von der aktuellen Position zum Werkzeugwechselpunkt. Daraus lässt sich unmittelbar das Optimierungspotential durch Verschiebung eines Werkstücks im Bauraum ableiten. Falls die Summe der Fahrten zum Werkzeugwechselpunkt und vom Werkzeugwechselpunkt zum Werkstück minimal ist, reduziert sich die gesamte Bearbeitungszeit für das NC-Programm.

4.1.2. Werkstückrotation

Nach Betrachtung der Werkstücktranslation stellt sich natürlich die Frage, welche Optimierungspotentiale durch eine Rotation des Werkstücks zu erwarten sind. Wie zuvor festgestellt, lassen sich durch Translationen des Werkstücks durch Veränderung der Strecke zwischen Werkzeugwechselpunkt Nebenzeiten reduzieren, was offensichtlich zu einer schnelleren Bearbeitungszeit führt. Da die Bearbeitungszeit sich in Diskussionen rund um das vorliegende Projekt als kritischster Punkt erwiesen hat, wäre es natürlich wünschenswert, durch Rotationen ebenfalls Reduzierungen in der Bearbeitungszeit erreichen zu können. In Ausnahmefällen könnte dieser Effekt theoretisch erreichbar sein. In der Praxis spielt die Werkstückrotation praktisch keine Rolle für die Bearbeitungszeit. Die ursprüngliche Hoffnung bei der Definition der Aufgabenstellung einer optimierten Maschineneinrichtung war es, unterschiedliche Achsgeschwindigkeiten auszunutzen, um die Bearbeitung auf Achsen zu verlagern, die mit einer höheren Geschwindigkeit verfahren können. Grundsätzlich wäre so etwas durch eine Optimierung vorstellbar, allerdings wird in den betrachteten Szenarien niemals der Punkt erreicht, an dem die Bearbeitung unnötig gebremst wird, weil eine bestimmte Achse den programmierten Vorschub nicht erreichen kann. Aus diesem Grund ist es sinnvoll, sich zu überlegen, welche weiteren Optimierungspotentiale entstehen können. Ein Stück weit kann man diese Potentiale den Kennzahlen ablesen, die im Zuge der Beschreibung der effizienten G-Code Interpretation vorgestellt worden sind. So ist es vorstellbar, dass die starke Verwendung bestimmter Achsen sich als energetisch günstiger erweist. Einige Achsen sind für die Bewegung des Maschinentisches verantwortlich, während andere Achsen die Spindel bewegen. Natürlich ist davon auszugehen, dass hierbei unterschiedliche Massen bewegt werden müssen, was Auswirkungen auf die Bewegungsenergie hat. Gleiches gilt für den Verschleiß. Es besteht grundsätzlich die Möglichkeit, dass bestimmte Achsen anfälliger sind, zu verschleißen, womit durch eine Verlagerung der Bearbeitung auf andere Achsen die Wahrscheinlichkeit von Stillständen reduziert werden könnte.

Darüber hinaus könnte es interessant sein, zu beobachten, inwiefern der Bauraum eingeschränkt werden kann, der für die reine Fräsbearbeitung beansprucht wird. Durch

Rotation könnte dieser Raum verkleinert werden, wodurch das zu bearbeitende Werkstück möglicherweise auf alternativen Anlagen gefertigt werden könnte.

Trotz der Tatsache, dass eine Reduzierung der Nebenzeiten durch Werkstückrotation nicht möglich ist, sollte man noch die Möglichkeit erwähnen, die Rotation als Eingangsgröße für die Optimierung von Nebenzeiten zu verwenden. Es ist offensichtlich, dass hierdurch der Suchraum auf den ersten Blick künstlich um eine Dimension erweitert wird. Folglich wird dieser Wert formal gesehen bei der Auswertung der Fitnessfunktion lediglich Rauschen erzeugen. Es ist allerdings denkbar, dass eine Translation des Werkstücks unter Annahme einer festgelegten Ausrichtung des Werkstücks zu einer Kollision führt. In Randfällen wäre es dann möglich, durch Rotation um den verschobenen Punkt zu einer Konfiguration zu kommen, die gleichzeitig die Nebenzeiten minimiert und eine kollisionsfreie Fertigung ermöglicht.

Auf dem Weg zur Entwicklung eines Verfahrens, das sich für den gegebenen Anwendungsfall eignet, hat es Evolutionsstufen in der Entwicklung gegeben, um die Anforderungen, die durch heterogene Rechnerumgebungen vorgegeben sind, zu erfüllen. Zunächst stand im Mittelpunkt, die Anzahl notwendiger Simulationen zu minimieren, wobei zu Beginn der Überlegungen die effiziente NC-Interpretation noch kein Thema gewesen ist. In diesem Zusammenhang ist ein Verfahren entstanden, das mittels Dimensionsreduktion den Suchraum einschränkt und somit die Konvergenz des Optimierungsverfahrens beschleunigt. Wie bereits beschrieben, zeigte sich, dass der Problemstellung kein hochdimensionaler Suchraum zugrunde liegt, weswegen für die Beschreibung des Ansatzes auf den Ausblick (Kapitel 7) verwiesen sei. Es folgte die Fragestellung, welche Anforderungen durch die Verwendung verteilter Systeme im Kontext simulationsgestützter Optimierung im Allgemeinen und in Bezug auf die Dienstleistungsplattform im Speziellen entstehen. In dem Zusammenhang wurde herausgearbeitet, dass für viele Fälle ein gewisses Maß an Asynchronität, insbesondere bei schwarmbasierten Methoden, sinnvoll wäre. Deshalb wurde auf theoretischer Grundlage betrachtet, ob eine asynchrone Berechnung schwarmbasierter Optimierungsverfahren sinnvoll ist. Im Anschluss wird das letztlich entwickelte Verfahren präsentiert, das auf der Kombination aus effizienter NC-Interpretation und komplexer Materialabtragssimulation der virtuellen Werkzeugmaschine beruht. Hierbei werden Ergebnisse für drei verschiedene NC-Programme präsentiert. Da die Komplexität der einzelnen Simulationen die Erzeugung einer größeren Stichprobe an Ergebnissen unmöglich gemacht hat, folgen im Anschluss Untersuchungen, die eine vereinfachte Kollisionsprüfung zur Grundlage haben.

4.3. Asynchrone Berechnung

Betrachtet man den Standardalgorithmus zur Partikelschwarmoptimierung, kommt man schnell zu der Erkenntnis, dass sich das Verfahren hervorragend dazu eignet, parallel berechnet zu werden.

Algorithmus: Parallele Berechnung der PSO							
Input: $I :=$ Anzahl Iterationen, $\varpi :=$ Populationsgröße, $f(\vec{x}) :=$ Fitnessfunktion							
Globale Variablen: $\rho_{global}^{best}, \vec{\rho}_{global}$							
Prozess π $[\phi \in \{0, \dots, \varphi - 1\}]$							
	Initialisiere $\vec{\rho}_{local}^{\phi}, \vec{x}_{\phi}, \vec{v}_{\phi}$						
	While (Kein Konvergenzkriterium ist erreicht)						
		$t \to t + 1$					
		$v_{\phi,t} \to \omega * v_{\phi,t-1} + \varsigma_g * (\vec{\rho}_{global} - \vec{x}_{\phi}) + \varsigma_{local} * (\vec{\rho}_{local}^{\phi} - \vec{x}_{\phi})$					
		$\vec{x}_{\phi} \to \vec{x}_{\phi} + \vec{v}_{\phi}$					
		If $(f(\vec{x}_{\phi}) < \rho_{local}^{best_{\phi}}$					
			$\vec{\rho}_{local}^{\phi} \to \vec{x}_{\phi}, \rho_{local}^{best_{\phi}} \to f(\vec{x}_{\phi})$				
			If ($f(\vec{x}_{\phi}) < \rho_{global}^{best}$**)**				
				$[[\vec{\rho}_{global} \to \vec{x}_{\phi}, \rho_{global}^{best} \to f(\vec{x}_{\phi})]]$			
			Endif				
		Endif					
	Endwhile						
Endprozess							

Algorithmus 3 Durchführung der asynchronen Berechnung einer Partikelschwarmoptimierung

Algorithmus 3 zeigt, wie dieses Vorgehen konkret aussieht. Hierbei wird also sämtlichen Partikeln genau ein Prozess zugeordnet. Von einer Shared Memory Architektur ausgehend, wird das globale Optimum als geteilte Variable definiert, während jeder Prozess die Information über das persönliche Optimum des korrespondierenden Partikels für sich vorhält. Die asynchrone Berechnung geht dann davon aus, dass eine Aktualisierung von Position und Geschwindigkeit nicht vom Abschluss einer Iteration abhängig ist, sondern auf die Information zurückgegriffen wird, die zum Zeitpunkt der abgeschlossenen Evaluation des Partikels vorliegt. Bei der Formalisierung des Algorithmus wurde hier die Schreibweise „[[…]]“ eingeführt. Diese soll ausdrücken, dass das Schreiben in den gemeinsamen Speicher an der Stelle unter wechselseitigem Ausschluss erfolgt. Die Idee, schwarm- oder populationsbasierte Verfahren asynchron bezüglich der Iterationen zu berechnen, ist nicht erst im Rahmen dieses Projekts entstanden, sondern wurde schon im Vorfeld in einigen Publikationen beschrieben. So beschreiben Scriven et al. in einer Publikation, wie ein großer Schwarm in Teilschwärme zerlegt wird, so dass jeder Schwarm unabhängig durch einen Prozessor bearbeitet werden kann [44]. Die Autoren verfolgten das Ziel, Optimierungen mittels Partikelschwarmoptimierung für heterogene, verteilte Rechnerarchitekturen, z.B. Grids, anwendbar zu machen. Gleichzeitig wurden fehleranfällige Umgebungen untersucht. Die Autoren kommen zu dem Ergebnis, dass in beiden Fällen eine teilsynchrone Durchführung der Optimierung durchgeführt werden sollte. Eine Untersuchung von Koh et al. zeigt, dass die synchrone, parallele Durchführung der Updateschritte zu einer schlechten Ausnutzung der parallelen Ressourcen führt, wenn die Ressourcen einem Ungleichgewicht unterliegen [45]. Aus diesem Grund wird im Rahmen der genannten

Veröffentlichung ebenfalls eine asynchrone Variante der PSO evaluiert. Grundlage für die Evaluationen sind hierbei künstliche Testfälle und eine Problemstellung aus der Biomechanik, ein kinematisches Knöchelgelenkmodell. Venter et al. nutzten die asynchrone PSO im Rahmen eines Optimierungsproblems im Bereich von Aerodynamiksimulationen. Ziel hierbei war das Auffinden der maximalen Spannweite eines Flügels unter Berücksichtigung des aerodynamischen Widerstands und des Flügelgewichts [46]. Rada-Vileda et al. führten im Rahmen zweier Veröffentlichungen Performanzstudien auf Basis von zehn verschiedenen Testfunktionen durch, wobei gleichzeitig verschiedene Topologien betrachtet wurden. Hierbei kommen die Autoren zu dem Ergebnis, dass die asynchrone PSO zu schnelleren und besseren Ergebnissen führt als die synchrone Variante [47] [48]. Eine weitere Anwendung einer asynchronen Durchführung der PSO liefern Hsieh et al. im Rahmen einer Untersuchung, in der in einem diskreten Szenario die Automatisierung einer Feature Selektion implementiert und evaluiert wird [49].

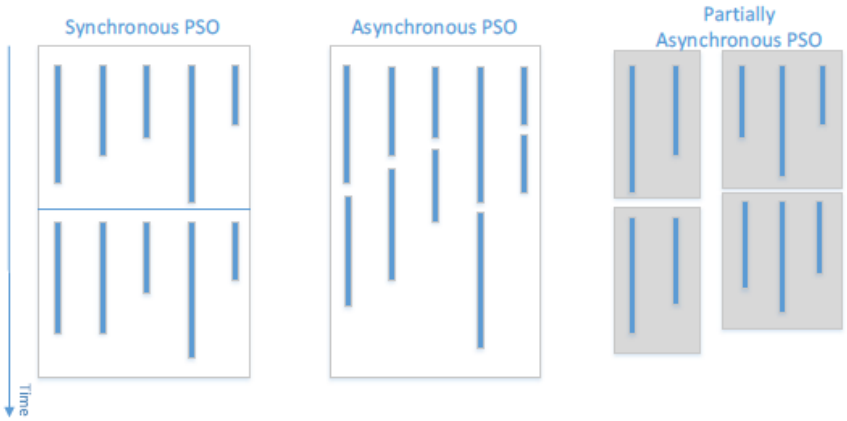

Abb. 11 Schematische Darstellung der synchronen, asynchronen und teilsynchronen Durchführung der PSO

Ziel einer Veröffentlichung im Rahmen des vorliegenden Kontexts war es insbesondere, Fälle zu identifizieren, für die ein gewisses Maß an Asynchronität sinnvoll erscheint [50]. So müsste man zunächst den Fall betrachten, dass innerhalb eines Systems, das Rechnerressourcen dynamisch zur Verfügung stellt, eine statische, generationsbasierte Aktualisierung der Positionen und Geschwindigkeiten zu erheblichen Wartezeiten für den Algorithmus führen kann. Ein zweiter Aspekt ist die extreme Heterogenität der Simulationsdauern, die sowohl auf verschieden performante Rechnerknoten als auch die Möglichkeit auftretender Kollisionen zurückgeführt werden kann. Insbesondere der Kollisionsaspekt ist im vorliegenden Fokus interessant, da eine Kollision während der Simulation zu der Möglichkeit führt, die entsprechende Simulation abzubrechen. Der dritte, extrem entscheidende Aspekt, ist die Möglichkeit von Knotenausfällen. Hier muss es sich nicht notwendigerweise um hardwarebedingte Ausfälle handeln, sondern beispielsweise um auftretende Fehlerfälle während der Simulation, die zu einer Art Deadlock führen. In dem Fall ist der Prozess nicht in der Lage, zu identifizieren, dass die

entsprechende Simulation nicht mehr determinieren kann, was bei einer generationsbasierten Aktualisierung zu einem Stillstand der kompletten Optimierung führt. Natürlich lässt sich der Aspekt der Knotenausfälle generalisieren, etwa, wenn man im HPC-Bereich an ‚Exascale Computing' denkt, also das Rechnen auf verteilten Systemen, deren Taktung auf der Exa-FLOPs Skala liegt. Aufgrund der enormen Anzahl an Knoten ist hierbei selbst bei einer niedrigen Ausfallwahrscheinlichkeit einzelner Knoten die Chance eines störungsfreien Betriebs nahezu ausgeschlossen [51] [52].

Um die genannten Punkte zu berücksichtigen, wurden verschiedene Experimente auf Basis der Griewankfunktion mit der Partikelschwarmoptimierung durchgeführt. Es ist offensichtlich, dass die Betrachtung etwas einseitig ausfällt, sowohl bezüglich der Fitnessfunktion als auch des Optimierungsalgorithmus. Es kann allerdings an dieser Stelle vorweggenommen werden, dass die entsprechende Auswertung hinreichend war, um zu erkennen, dass eine Optimierung nur auf dieser Basis noch nicht hinreichend für eine effiziente Durchführung ist. Zunächst sei die Griewankfunktion vorgestellt, die an späterer Stelle für weitere theoretische Betrachtungen zugrunde gelegt wird. Sei $n \in \mathbb{N}$ die Anzahl der Dimensionen im Suchraum. Dann ist die Griewankfunktion folgendermaßen definiert [53]:

$$f_{Griewank} = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

4.3.1. Experimente

Auf Basis der Griewankfunktion wurden Experimente durchgeführt, deren Ergebnisse im Folgenden präsentiert werden. Zunächst geht es um die Fragestellung, wie sich die Konvergenz der synchronen und der asynchronen Variante bezüglich der Anzahl der Evaluationen verhält. Hierbei wurde zusätzlich eine teilsynchrone Topologie implementiert. Die drei Varianten, die hier verglichen werden, sind in Abb. 11 dargestellt. Formal lässt sich die Erstellung der Topologien folgendermaßen darstellen: Sei $P := [1 \dots i]$ die Menge der Partikel und $G := [1 \dots j], j \leq i$ die Menge der Gruppen, in die die jeweiligen Partikel eingeteilt werden. Sei zusätzlich $p \in P$ und $g \in G$. Dann lässt sich die Einteilung der einzelnen Partikel in die jeweiligen Gruppen als Abbildung $f: P \to G$ darstellen, wobei $g = \left\lceil p * {}^{j}/_{i} \right\rceil$. Offensichtlich können die vollständig asynchrone sowie die synchrone Variante der PSO dann als Spezialfall der teilsynchronen Darstellung interpretiert werden. Dazu müssen die Randfälle $i = j$ (asynchron) und $i = 1$ (synchron) betrachtet werden.

Für den Vergleich der Konvergenz wurde die Griewankfunktion mit jeweils 10 und 50 Dimensionen als Fitnessfunktion zugrunde gelegt. Auch die Schwarmgröße für die PSO wurde variiert. So wurden Untersuchungen mit 30, 50 und 100 Partikeln durchgeführt.

Für die teilsynchrone Variante wurde $j = 4$ festgelegt, wodurch Grad der Asynchronität (d_g) lediglich von der Schwarmgröße abhängig ist. Dieser lässt sich einfach aus der Formel $j = i * d_g$ ableiten.

Um aus der Auswertung einer theoretischen Benchmarkfunktion eine Aussage für ein simulationsgestütztes Szenario unter Annahme verteilter Ressourcen zu treffen, muss die Dauer einer Einzelsimulation modelliert werden. Hierbei wird eine Besonderheit des eigentlichen Simulationsgegenstands berücksichtigt. Ausgehend davon, dass die Dauer der NC-Bearbeitung minimiert werden soll und zusätzlich eine exakte Simulationsumgebung vorliegt, korreliert die Fitnessfunktion mit der Dauer der Simulation. Jeder einzelne Prozess berechnet demnach zunächst die Fitnessfunktion und blockiert im Anschluss für die Dauer d_b, wobei $d_b = f_{griewank} * r_{[0,9-1,1]}$. Der Ausdruck $r_{[a,b]}$ repräsentiert einen Rauschfaktor, wobei $a \leq r_{[a,b]} \leq b$. Dieser Faktor wird als Zufallszahl einer Gleichverteilung ermittelt.

Tabelle 2 Grad der Asynchronität unter Annahme verschiedener Schwarmgrößen

Schwarmgröße i	**Grad der Asynchronität d_g**
30	13,33%
50	8%
100	4%

Abb. 12 illustriert ausgewählte Szenarien für den Konvergenzvergleich. Jedes Experiment wurde mit 100 Wiederholungen durchgeführt und die Ergebnisse im Anschluss gemittelt. Die rote Linie beschreibt jeweils die Konvergenz der asynchronen Implementierung, die blaue Linie die der vollsynchronen Implementierung und die grüne Linie die der teilsynchronen Implementierung. Die blaue und die grüne Linie zeigen sich hier jeweils als Stufenfunktion, weil die Auswertung der Fitnessfunktionen jeweils erst mit Abschluss der entsprechenden Generation stattfindet. In der Teilabbildung oben links ist die Betrachtung von 10 Dimensionen bei 30 Partikeln zu sehen. Es fällt auf, dass die Anzahl der benötigten Evaluationen im Vergleich zu den synchronen Verläufen deutlich niedriger ist. Dieser Effekt ist bei höherer Dimensionsgröße in der Form nicht mehr zu sehen. Der simplere Suchraum scheint dazu zu führen, dass ein kleiner Anteil an Partikeln diesen gut explorieren kann, ohne Informationen über sämtliche Partikel zu haben. Die allgemeine Aussage, die sich hier ableiten lässt, ist, dass eine asynchrone Ausführung der PSO im gegebenen Szenario nicht zu einer Mehrzahl an Evaluationen führt, weswegen diese zum Zweck der besseren Ressourcenauslastung vorzuziehen ist.

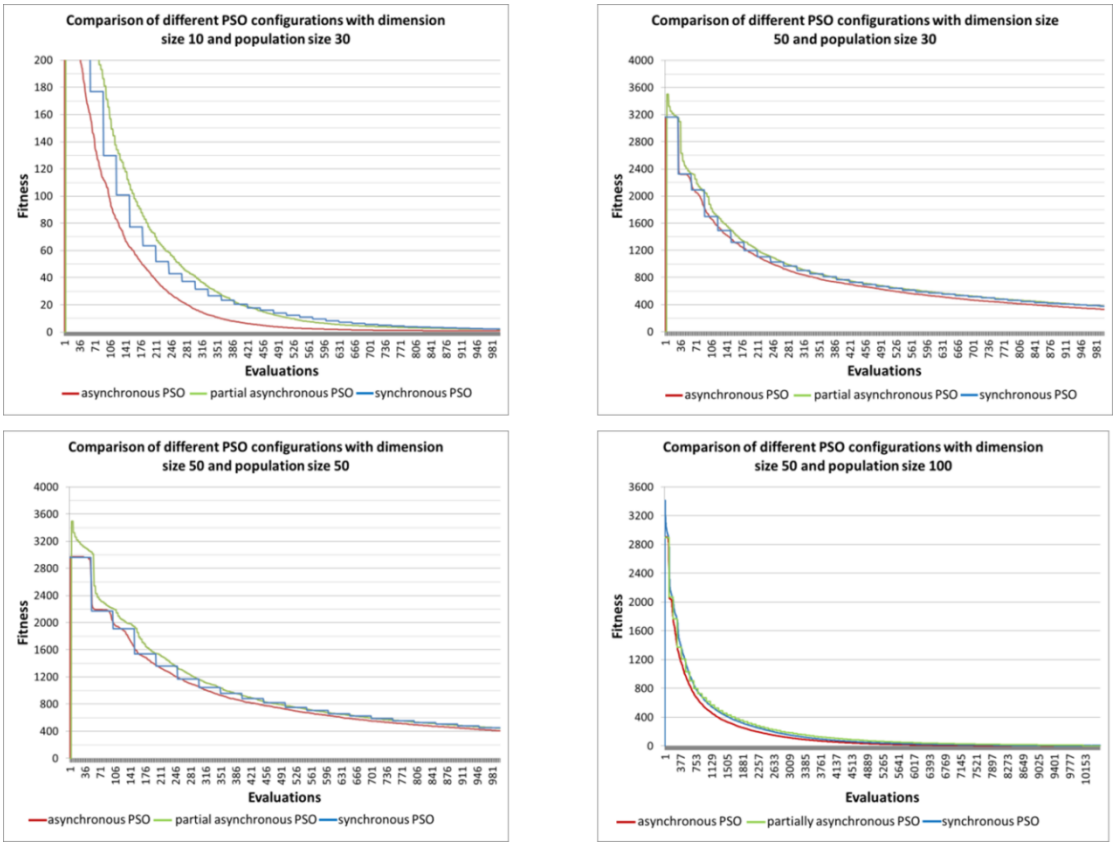

Abb. 12 Vergleich der Konvergenz von synchroner, asynchroner und teilsynchroner PSO

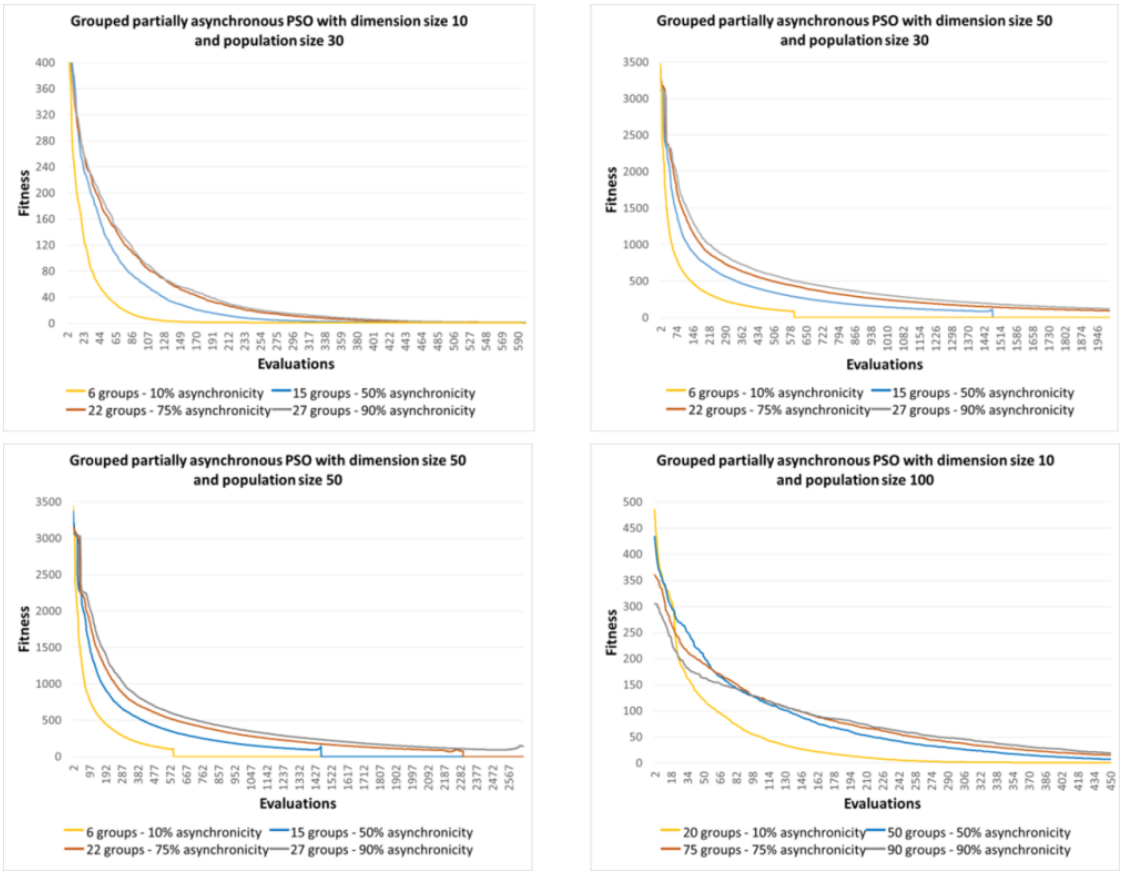

Abb. 13 Vergleich verschiedener Grade der Asynchronität nach Evaluationen

Die Ergebnisse, die auf Abb. 13 zu sehen sind, beziehen sich auf Versuche, in denen das Maß an Asynchronität variiert worden ist. Die vollständig asynchrone und vollständig synchrone Variante sind hierbei nicht vertreten. Es zeigt sich in sämtlichen Grafiken, dass ein niedriges Maß an Asynchronität vorzuziehen ist, was scheinbar der Aussage des Experiments konterkariert. Die Vermutung hinter diesen Ergebnissen ist, dass innerhalb der einzelnen Gruppen tendenziell eine starke Diversität an Simulationszeiten vorliegt. Gleichzeitig führt die asynchrone Durchführung zu einer Gemengelage, in der die Wechselwirkung der einzelnen Partikel untereinander mit sinkender Synchronität abnimmt. Vereinfacht gesagt, bekommt man durch die Teilasynchronität die Nachteile beider Welten, also unbefriedigende Auslastung der Ressourcen und eine schwächere Exploration im Raum.

Zuletzt wurden noch zwei Versuche gemacht, in die absoluten Zeiten der Optimierungsläufe in Millisekunden miteinander verglichen wurden. Die Ergebnisse dieser Versuche sind in Abb. 14 zu sehen. Der Versuch auf der linken Seite behandelt die Problematik, dass die verfügbaren Rechnerknoten nicht ganzzahlig durch die Anzahl der Partikel teilbar sind. In diesem Versuch werden 8 Rechnerknoten bei 17 Partikeln angenommen. Das bedeutet, dass bei synchroner Durchführung zunächst zweimal 8 Partikel parallel evaluiert werden, was zu im Anschluss zu einem einzelnen Partikel führt, der für den Aktualisierungsschritt der PSO ausgewertet werden muss, während die restlichen Ressourcen auf neue Positionsvektoren warten. Offensichtlich ist das ein Extremfall, weil als Schwarmgröße eine Primzahl gewählt wurde. Demnach zeigt der Graph auf der linken Seite, dass der Effekt der vollständig asynchronen Variante hierbei signifikant ist.

Der Graph auf der rechten Seite der Abbildung zieht mögliche Knotenausfälle während der Optimierung in Betracht. Hierbei wird angenommen, dass für 30 Partikel 30 Knoten zur Verfügung stehen. Die Griewankfunktion wurde mit 10 Dimensionen gewählt. Mögliche Knotenausfälle wurden folgendermaßen modelliert: Sei $p_{Ausfall} = 0{,}001$ die Wahrscheinlichkeit, dass ein Knoten ausfällt und $p_{Reperatur} = 0.009$. Zusätzlich wird eine Statusvariable $\mathbb{s} \in [0,1]$ für jeden Knoten definiert, die initial auf 1 gesetzt ist. Falls $\mathbb{s}$ auf 1 gesetzt ist, wird sekündlich eine gleichverteilte Zufallszahl gezogen. Liegt der Wert der Variable unter $p_{Ausfall}$, wird $\mathbb{s}$ für diesen Knoten auf 0 gesetzt. In diesem Fall kann dieser Knoten nicht mehr für die Evaluation der Funktion eingesetzt werden. Es werden allerdings weiterhin Zufallszahlen gezogen, die den Status wieder auf 1 setzen, falls diese über dem Wert von $p_{Reperatur}$ liegen. Auch dieses Ergebnis entspricht den Erwartungen. Die asynchrone Variante der PSO verhält sich in einem fehleranfälligen Umfeld deutlich robuster.

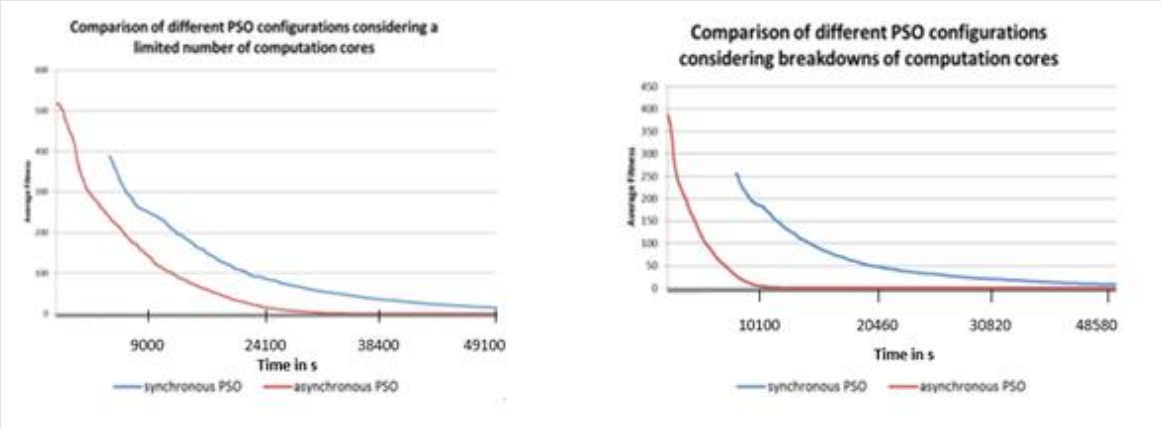

Abb. 14 Vergleich der Konvergenz nach Rechenzeit bei Betrachtung limitierter Rechenknoten (links) und simulierten Knotenausfällen (rechts)

Alles in allem haben die Untersuchungen gezeigt, dass heterogene, verteilte Umgebungen den Einsatz asynchroner Optimierungsverfahren erfordern, falls populations- oder schwarmbasierte Verfahren wie die PSO zum Einsatz kommen. Die anfänglichen Betrachtungen führen allerdings zu der Annahme, dass sich die Anzahl der notwendigen Simulationen nicht so maßgeblich reduziert, dass der Einsatz asynchroner Optimierungstechniken alleine ausreicht, um optimale Ergebnisse in akzeptabler Zeit zu erreichen. Offensichtlich könnten die Untersuchungen an einigen Stellen weiter vertieft werden, etwa durch die Betrachtung alternativer Metaheuristiken oder durch Veränderungen in der Modellierung der Simulationsdauern. Da dadurch allerdings keine hohen Mehrgewinne für die konkrete Fragestellung zu erwarten sind, wurde die Entscheidung getroffen, auf intelligentere Verfahren zu setzen, die im weiteren Verlauf der Arbeit präsentiert werden.

4.4. Hybrides Verfahren auf Basis effizienter G-Code Interpretation

Obwohl grundsätzlich zu sehen war, dass die asynchrone Durchführung schwarm- oder populationsbasierter Optimierungsalgorithmen in verteilten Umgebungen zielführend sein kann, liegt damit noch kein Verfahren vor, das im vorliegenden Kontext vollumfänglich die Anforderungen erfüllt. Insbesondere während der ersten Iterationen verhalten sich sämtliche Optimierungsverfahren ein Stück weit explorativ, was potentiell zu einer unverhältnismäßig großen Anzahl an Simulationen führt, deren Fitness weit unter dem Optimum liegt. Aus diesem Grund ist ein Verfahren entstanden, das auf der effizienten Auswertung des G-Codes beruht. Obwohl es sich dabei nur um eine Annäherung handelt, ergeben sich einige Vorteile, diese zwecks Einrichtungsoptimierung zu verwenden:

- **Deutlich beschleunigte Auswertung**
  Die Auswertung eines NC-Programms mittels effizienter NC-Interpretation erfordert keinerlei Overhead durch Hochfahren und Vorbereiten der virtuellen Werkzeugmaschine, sondern iteriert lediglich über die jeweiligen NC-Sätze, was

im schlimmsten Fall wenige Sekunden bei komplexen NC-Programmen in Anspruch nehmen kann. Abhängig von der Komplexität des zu optimierenden NC-Programms kann die NC-Simulation selbst (also abzüglich Vorbereitung) Zeiten in Anspruch nehmen, die im Stundenbereich liegen.

- **Verhältnismäßig höhere Genauigkeit der Berechnung der Fertigungsdauer**
  Die ursprüngliche Idee, die der Implementierung der effizienten NC-Interpretation zugrunde lag, war die Approximation von Kennzahlen für ein vorliegendes NC-Programm. Die Praxis hat gezeigt, dass die Berechnung der Fertigungsdauer innerhalb der NC-Simulation durch die virtuelle Werkzeugmaschine tendenziell ungenauere Ergebnisse liefert als der Interpreter. Relative Verbesserungen der Aufspannsituation führen zwar in qualitativer Hinsicht zu verbesserten Durchführungszeiten im Simulationsbericht. Eine zuverlässige quantitative Ermittlung hat es allerdings nicht gegeben.

- **Möglichkeit zur Erhebung mehrerer Kennzahlen**
  Da der NC-Interpreter eine Implementierung ist, die im Rahmen der Arbeit entstanden ist, liegt eine vollständige Kontrolle über die Ermittlung der relevanten Kennzahlen vor. So lassen sich Werte wie maximale Auslenkungen über die jeweiligen Achsen und die Gesamtstrecke der einzelnen Achsen nicht aus dem Simulationsbericht ablesen.

Diese Tatsachen führten zu der Erkenntnis, dass die virtuelle Werkzeugmaschine lediglich eine Kenngröße liefert, die sich nicht aus der reinen NC Interpretation ableiten lässt. Dabei handelt es sich offensichtlich um die Kollisionsprüfung. Insofern wird im Folgenden ein Verfahren vorgestellt, das im ersten Schritt durch Optimierung eine Kollektion an Kandidaten mit guter Fitnessfunktion erzeugt und im Anschluss durch Kollisionsprüfung die zulässigen Kandidaten herausfiltert.

4.4.1. Formale Beschreibung

Sei $F: \mathbb{R}^d \to \mathbb{R} \times \{0,1\}$ eine Fitnessfunktion, die für eine gegebene Konfiguration sowohl die numerische Fitness als auch die Validität berechnet. Sei nun $f: \mathbb{R}^d \to \mathbb{R}$ eine Abbildung, die für einen gegebenen Input die numerische Fitness ausgibt und $g: \mathbb{R}^d \to \{0,1\}$ eine Abbildung, die für einen Eingangsvektor prüft, ob dieser valide ist. Auf der Basis lässt sich ein flexibles Verfahren definieren, das aus zwei Phasen besteht. Die erste Phase besteht darin, Kandidaten zu sammeln. Die Sammlung dieser Kandidaten erfolgt durch die Durchführung eines beliebigen Optimierungsverfahrens unter Annahme der Fitnessfunktion f, in dessen Verlauf sämtliche Funktionsevaluationen persistiert werden. Sei dann $X := \{(a,b): a \in \mathbb{R}^d, b \in \mathbb{R}, b = f(a); b \leq \theta\}$ die Menge der Tupel aus Vektoren und den korrespondierenden Fitnesswerten, die während dieser Iteration erfasst worden sind. Hierbei gibt $\theta \in \mathbb{R}$ einen Grenzwert an. Liegen Werte über diesem Grenzwert, sind diese nicht hinreichend relevant im Sinne der Optimierung. Hier wird

ohne Beschränkung der Allgemeinheit davon ausgegangen, dass es sich um ein Minimierungsproblem handelt.

Sei nun $k \in \mathbb{N}$ die Anzahl der verfügbaren, verteilten Instanzen für die Durchführung von Simulationen. Dann wird im zweiten Schritt eine sinnvolle Verteilung der gesammelten Kandidaten auf die einzelnen Rechnerknoten gesucht. Sei dafür $S := (s_1, s_2, \dots, s_k)$ mit $s_i \subseteq X, \forall i \in [1, k]$. Weiterhin sei a der Eingangsvektor und $b = f(a)$. Dann sei angenommen, dass alle s_i absteigend bezüglich b sortiert sind. Es stellt sich nun zunächst die Frage, wie eine sinnvolle Zerlegung S hergestellt werden kann. Im Nachgang werden Methoden vorgestellt, die sich dazu eignen. Angenommen, es liegt eine Zerlegung S vor. Dann stehen also zunächst k Listen mit absteigend sortierten Fitnesswerten für die Evaluation der Funktion g zur Verfügung. Es wird davon ausgegangen, dass die Berechnungsdauer für g signifikant größer ist als die Berechnung von f, so dass es lohnenswert ist, die jeweiligen Auswertungen von g auf die Rechnerknoten zu verteilen. Zunächst wird also jeweils der beste Kandidat aus jeder Teilmenge evaluiert. Sobald die Evaluation eines Kandidaten abgeschlossen ist, wird unterschieden, ob der entsprechende Eingangsvektor valide ist oder nicht. Sei $i \in \mathbb{N}$ nun der Index einer Teilmenge s_i, dann kann s_i aus X entfernt werden, da sämtliche Kandidaten in dieser Teilmenge einen schlechteren Fitnesswert aufweisen. Darüber hinaus können in sämtlichen Teilmengen $S \setminus s_i$ diejenigen Werte aus den Listen entfernt werden, für die gilt: $s_{\setminus i}^b > s_i^b$, wobei $s_{\setminus i}^b$ den Fitnesswert in einer beliebigen Teilmenge $s_j, j \neq i$ beschreibt und s_i^b den als valide identifizierten Wert aus der Teilmenge s_i. Auf diese Art und Weise werden die Kandidaten nach und nach aus der Datenstruktur entfernt. Das Verfahren terminiert, sobald sämtliche Kandidaten entweder evaluiert oder gelöscht worden sind.

Unüberwachtes Lernen zum Clustern von Datensätzen

Wie man sieht, ist die Art und Weise, wie S berechnet wird, grundsätzlich beliebig. Wenn man sich Gedanken darüber macht, wie man Kandidaten am sinnvollsten auf verschiedene, parallele Instanzen verteilt, muss man sich zunächst Gedanken darüber machen, was das eigentliche Optimierungsziel ist. Da gibt es grundsätzlich zwei Möglichkeiten.

1. **Bestes Ergebnis**

   Das offensichtlichste Ziel, das man verfolgen kann, wäre es, aus dem gegebenen Kandidatenpool möglichst schnell den besten Kandidaten herauszufinden, der eine valide Konfiguration aufweist. Ist dies der Fall, ist es nicht sinnvoll, sich über eine Zerlegung, wie sie hier diskutiert worden ist, Gedanken zu machen. Die schnellste Lösung, den besten validen Kandidaten zu ermitteln, ergibt sich, wenn alle Kandidaten sortiert werden. Aus dieser Sortierung entspringt dann direkt die Priorisierung der Durchführungsreihenfolge der Auswertungen. In diesem Fall werden also zunächst die k besten Kandidaten an die k Rechnerknoten verteilt.

Sobald eine Evaluierung beendet ist, wird die Validität geprüft. Falls der Eingangsvektor valide ist, kann die komplette Prozedur abgebrochen werden. Andernfalls wird der nächstbeste Kandidat aus der Auftragsliste geholt und evaluiert.

2. **Schnelles Ergebnis**
   Die einseitige Suche nach dem besten Ergebnis birgt die Gefahr, dass die zugrundeliegenden Kandidaten tendenziell in relativer Nachbarschaft innerhalb des betrachteten Vektorraums liegen. Für diesen Fall wird ziemlich viel Rechenzeit beansprucht, ausschließlich in einem Bereich zu suchen, in dem eigentlich keine validen Eingangsvektoren zu erwarten sind. Wenn man sich an die Optimierungsebenen in einem realen Szenario vor Augen führt (vgl. Forschungsprojekt InVorMa [9]), kommt man schnell zu dem Schluss, dass das beste Ergebnis gar nicht unbedingt das einzige Kriterium im Gesamtkontext sein muss. Hierbei reagiert die Optimierung der Arbeitsplanung unmittelbar auf eine optimierte Aufspannsituation, falls diese zu einer reduzierten Bearbeitungsdauer führt. Da die Optimierung also nicht für sich steht, sondern sich in einen größeren, dynamischen Kontext einbettet, ist es also grundsätzlich interessant, den Kandidatenraum explorativ zu betrachten. Aus diesem Grund wird im Folgenden ein Verfahren erläutert, das dieses Vorgehen auf Basis der Menge X möglich macht. Daher wird in der Folge auf das Clustering von Daten eingegangen.

Einfach gesprochen ist die Fragestellungen nach dem Clustering sehr intuitiv: Gegeben eine Menge nicht-markierter Beobachtungen, soll ein Clusteringverfahren möglichst sinnvolle Teilmengen (Cluster) erzeugen, wobei die Vektoren innerhalb eines Clusters ähnlich zueinander sein sollen, während die Distanzen zwischen den Clustern sich signifikant unterscheiden sollen. Sei also $\mathcal{X} \in \mathbb{R}^{d \times n}$ eine Menge mit n d-dimensionalen Beobachtung und k die Anzahl der gewünschten Cluster. Außerdem sei $\mathcal{S} := \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k\}: \bigcap_{i=1}^{k} \mathcal{S}_i := \mathcal{X}$ die Menge der Zerlegungen über $\mathcal{X}$. Dann ist das formale Ziel für eine optimale Zuordnung gegeben durch:

$$\arg\min_{\mathcal{S}} \sum_{i=1}^{k} \sum_{x \in \mathcal{S}_i} \|x - \mu_i\|^2$$

Eine analytische Berechnung dieses Minimums ist in hinnehmbarer Zeit nicht möglich, da es sich um ein NP-schweres Problem handelt [54] [55]. Ein heuristisches Verfahren zum Finden dieser Zerlegungen ist der k-Means Algorithmus, der ursprünglich von S. Lloyd vorgestellt wurde [56]. Sei $\mathrm{M}^0 := \{\mu_1^0, \dots, \mu_k^0\}: \mu_i^0 \in \mathbb{R}^d$ eine zufällig initialisierte Menge im d-dimensionalen Raum.

Dann ergibt sich zu einem beliebigen Iterationsschritt t die Zuordnung $\mathcal{S}^t$ durch:

$$\mathcal{S}_i^t = \{x: \|x - \mu_i^t\|^2 < \left\|x - \mu_j^t\right\|^2 \; \forall \, j \in \{1, \dots, k\} \wedge j \neq i\}$$

Demnach wird jede Beobachtung dem Punkt μ_i zugeordnet, der dieser Beobachtung bezogen auf die euklidische Distanz am nächsten liegt. Im Anschluss folgt der Aktualisierungsschritt, um die sogenannten Repräsentanten zu verschieben. Dieser lässt sich ausdrücken als:

$$\vec{\mu}_i^{t+1} = \frac{\sum_{x \in \mathcal{S}_i^t} \vec{x}}{\left|\mathcal{S}_i^t\right|}$$

Folglich werden die Repräsentanten so verschoben, dass sie im Schwerpunkt ihrer zugeordneten Beobachtungen liegen. Dieser Schritt wird wiederholt, bis ein bestimmtes Konvergenzkriterium erfüllt ist. Im vorliegenden Kontext handelt es sich bei dem Konvergenzkriterium um den Zeitpunkt, in dem die Positionen der Repräsentanten sich nicht mehr verändern.

4.4.2. Ergebnisse

Die Betrachtung der Ergebnisse für den konkreten Anwendungsfall wird sich in zwei Teile gliedern. Zunächst wird eine weitere Abstraktionsstufe eingeführt, mit deren Hilfe die Validitätsprüfung der virtuellen Werkzeugmaschine approximiert wird. Im Rahmen des Projekts wurde in der Arbeitsgruppe „Wirtschaftsinformatik, insbesondere CIM“ ein Framework entwickelt, das NC-Bearbeitungen auf eine zweidimensionale Ebene projiziert und durch Einfügen von Hindernissen die Möglichkeit auftretender Kollisionen induziert.

Abb. 15 illustriert, auf welche Art und Weise Anwendungsfälle mit Hilfe des Werkzeugs erzeugt werden können. Die Anwendungsfälle sowie die an späterer Stelle präsentierten Optimierungsergebnisse sind einer gemeinsamen Veröffentlichung mit der Arbeitsgruppe entnommen [57].

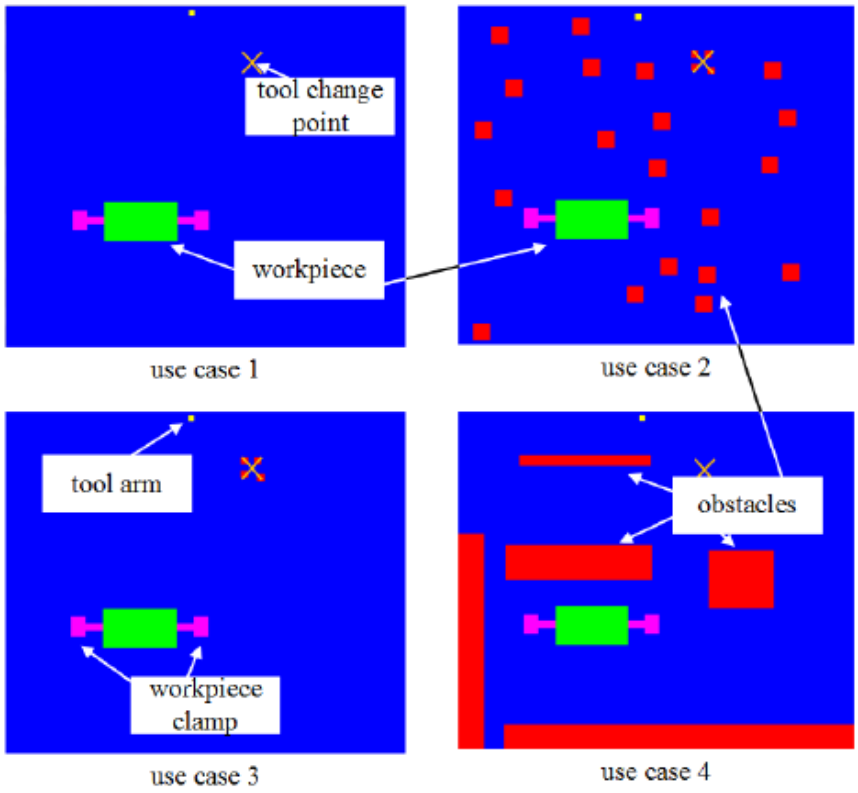

Abb. 15 Anwendungsfälle für die approximierte 2D-Projektion der NC-Bearbeitung [57]

Konkret geht dieser Kollisionsinterpreter folgendermaßen vor: Der Nutzer muss auf der Oberfläche die Position und Größe des Rohteils inklusive Spannmittel, die Position des Werkzeugwechselpunkts und den Nullpunkt der Werkzeugspindel definieren. Optional können darüber hinaus Hindernisse festgelegt werden. Daraufhin wird die folgende simple Prozedur gestartet: Das Verfahren iteriert satzweise über das NC-Programm. Bei Änderung einer X- oder Y-Koordinate wird eine Gradeninterpolation zum Zielpunkt durchgeführt. Veränderungen in der Z-Achse sowie Zyklen oder andere unbekannte Befehle werden ignoriert. Tritt im Laufe der Interpolation eine Kollision mit einem definierten Hindernis oder dem Spannmittel auf, wird die gegebene Konfiguration als nicht valide markiert. Man erkennt, dass so auch in sehr einfachen Aufspannsituationen Kollisionen auftreten können, etwa wenn das Spannmittel exakt auf dem Nullpunkt oder dem Werkzeugwechselpunkt liegt. Enthält ein NC-Satz den Befehl M6, wird vom aktuellen Punkt linear zum Werkzeugwechselpunkt interpoliert. Natürlich ist das eine sehr vereinfachte Repräsentation der Kollisionsprüfung, die es aber erlaubt, Experimente in größerem Ausmaß durchzuführen, als es mit der extrem rechenintensiven virtuellen Werkzeugmaschine möglich wäre. Die Experimente, die auf der Basis durchgeführt worden sind, sahen wie folgt aus: Als Fitnessfunktion f diente der NC-Interpreter, wobei lediglich die Bearbeitungsdauer als Fitnesswert betrachtet wurde, da hier ausschließlich translatorische Veränderungen durchgeführt wurden. Die Validitätsfunktion g bildete

dann die approximierte 2D-Kollisionsprüfung. Für die Wahl der Zerlegung S wurden die adaptierte binäre Suche und k-Means verglichen, um zu überprüfen, ob sich durch die Wahl der Zerlegung Unterschiede ergeben. Gemessen wurde die Anzahl der Iterationen, die man bis zum Erhalt des ersten Ergebnisses brauchte.

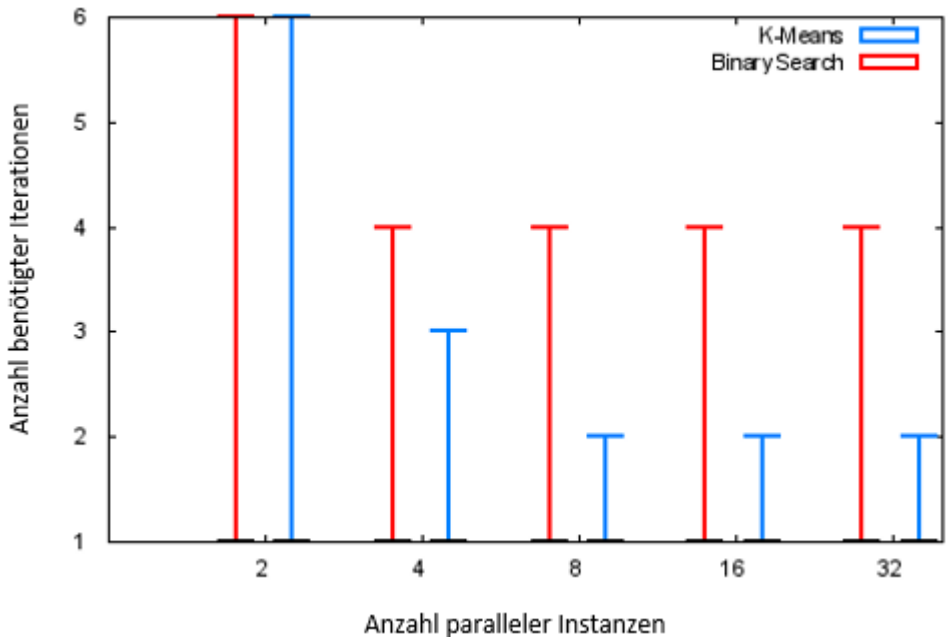

Abb. 16 Vergleich zwischen binäre Suche und k-Means für Use Case 1

Abb. 16 zeigt in Whiskerplots, wie sich k-Means und die binäre Suche für den ersten Use Case laut Abb. 15 unterscheiden. Zur Versuchsanordnung ist hierbei zu sagen, dass eine Partikelschwarmoptimierung mit 20 Individuen und Standardparameter für Exploration und Trägheit verwendet wurde. Unter Annahme verschiedener Zufallszahlen für die Initialisierung wurde diese Partikelschwarmoptimierung für jeden Use Case 50 Mal wiederholt. Es wurden jeweils diejenigen Ergebnisse gespeichert, die bessere Fitnesswerte aufgewiesen haben als die ursprüngliche Aufspannsituation. Die persistierten Vektoren wurden dann mit beiden Verfahren mit je 2,4,8,16 und 32 Zentren geclustert. Im Whiskerplot sieht man dann in schwarz den Median für die benötigte Anzahl der Iterationen. Die Grenzen der Boxen (soweit vorhanden) markieren das 25-Quantil (unten) und das 75-Quantil (oben). Die jeweils obere und untere Linie illustrieren die Ausreißer in beiden Richtungen. Use Case 1 zeigt im Wesentlichen ein erwartbares Ergebnis auf. Aufgrund der Einfachheit des Fallbeispiels bekommt man unabhängig von der Anzahl der Cluster nahezu immer in der ersten Iteration ein valides Ergebnis. Während der Optimierungsläufe scheint ein Datensatz entstanden zu sein, der unabhängig von der Anzahl der Cluster dafür sorgt, dass mehrere Iterationen benötigt werden. Dass dieser Ausreißer bei k-Means tendenziell kleiner ausfällt, hat allerdings keine Signifikanz und kann ignoriert werden.

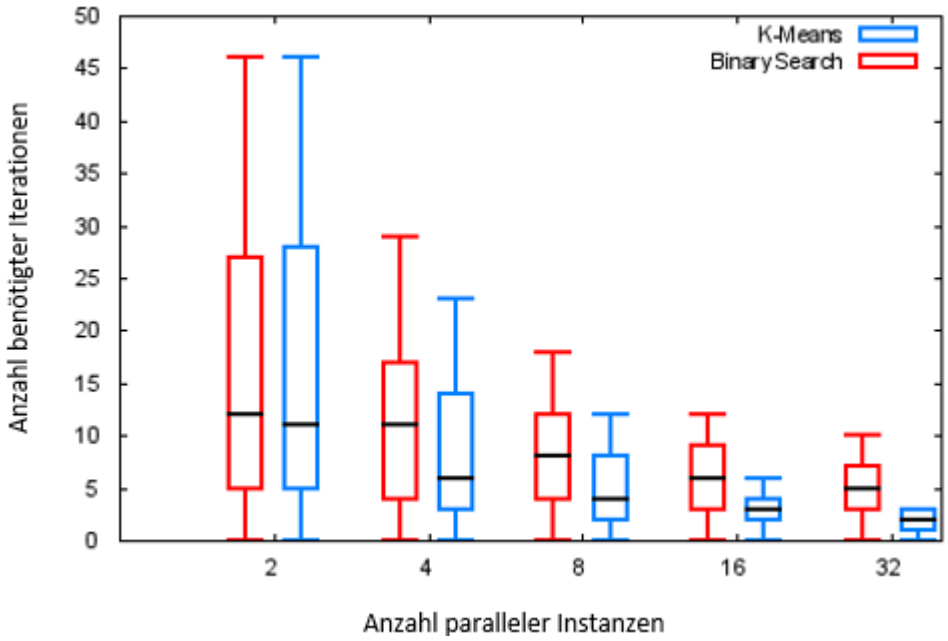

Abb. 17 Vergleich zwischen binäre Suche und k-Means für Use Case 2

Im deutlich komplexeren Use Case 2 ist eine klare und signifikante Differenzierung zwischen k-Means und binärer Suche zu erkennen. Insbesondere die Betrachtung des Medians führt hier zu der Erkenntnis, dass k-Means zu deutlich schnelleren Ergebnissen führt. Bei sehr wenigen Clusterzentren fällt auf, dass sämtliche Cluster offensichtlich eine sehr große Schnittmenge mit einem Bereich auf der Fläche haben, in dem nahezu ausschließlich nicht valide Konfigurationen liegen. Das führt zu einer Anzahl an Iterationen, die so groß ist, dass ein Unterschied zwischen dem zweistufigen Verfahren und einer Partikelschwarmoptimierung, die komplett auf rechenintensiven Fitnessfunktionen $F: \mathbb{R}^d \rightarrow \mathbb{R} \times \{0,1\}$ beruhen, kaum noch erkennbar ist. Daraus leitet sich die klare Empfehlung ab, in komplexen Szenarien die Anzahl der Cluster k nicht unbedingt auf die Anzahl der verfügbaren parallelen Ressourcen festzulegen, sondern möglicherweise eine größere Anzahl an Clustern zu erzeugen, so dass auch während der Evaluationsphase eine stärkere Exploration erfolgen kann.

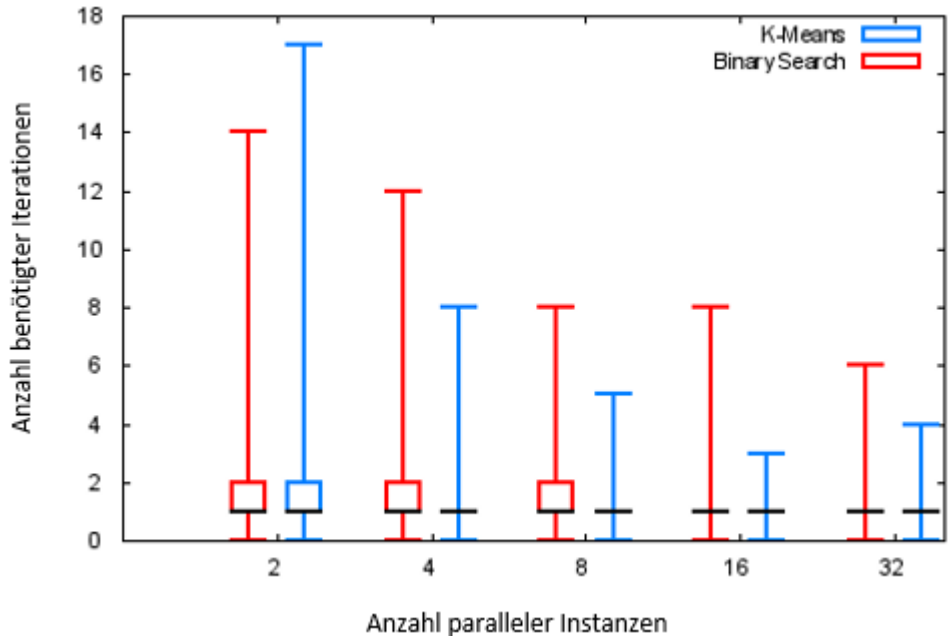

Abb. 18 Vergleich zwischen binäre Suche und k-Means für Use Case 3

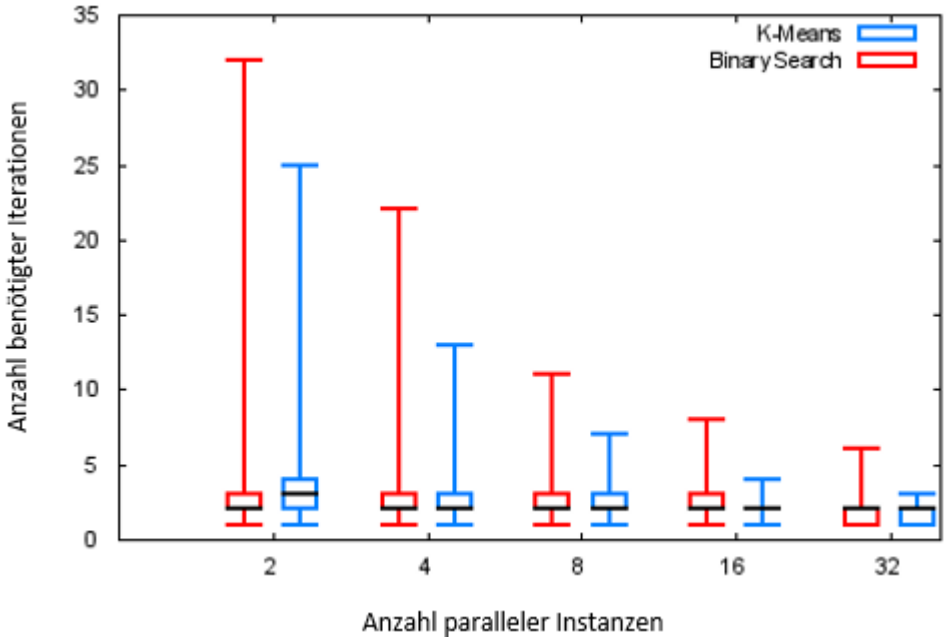

Abb. 19 Vergleich zwischen binäre Suche und k-Means für Use Case 4

Abb. 18 und Abb. 19 können gemeinsam betrachtet werden. Hierbei sieht man das Verhalten der beiden Zerlegungsverfahren bei Use Cases mit mittlerer Komplexität. Interessanterweise zeigt sich, dass einerseits der Median für beide Verfahren nahezu konstant bei zwei Iterationen liegt und die Anzahl der Iterationen gleichzeitig vollkommen unabhängig von der Anzahl der Cluster ist. Gleichzeitig sieht man keinen Unterschied zwischen den beiden Verfahren außerhalb der Betrachtung der Ausreißer. Es scheint sich hier zu bestätigen, dass k-Means weniger anfällig für starke Ausreißer ist.

Allerdings gilt weiterhin, dass die Datenlage nicht ausreichend ist, um an der Stelle eine abschließende Bewertung dieser Beobachtung vorzunehmen.

Auf Basis der vorangegangen Ergebnisse wurden drei Szenarien mit der virtuellen Werkzeugmaschine betrachtet. Eine größere Statistik zu erzeugen hätte eine höhere Anzahl an parallelen Instanzen der virtuellen Werkzeugmaschine erforderlich gemacht. Zunächst seien die Szenarien näher betrachtet:

Tabelle 3 Grobe Kennzahlen für die realen NC-Szenarien

Szenario	#Werkzeugwechsel	Bearbeitungsdauer
1	11	3588 Sekunden
2	6	922 Sekunden
3	0	354 Sekunden

Es werden also Szenarien mit unterschiedlicher Komplexität behandelt. Diese Szenarien unterscheiden sich nicht ausschließlich in ihrer Bearbeitungsdauer und der Anzahl der benötigten Werkzeuge, sondern auch durch die vorliegenden Freiheitsgrade bezüglich der Translationsoptimierung. So ist Szenario 1 insbesondere dadurch gekennzeichnet, dass durch seine Größe ein erheblicher Teil des Maschinentisches bedeckt ist, was die Möglichkeiten von Translationen in X- und Y-Richtung erheblich einschränkt. Szenario 2 nimmt lediglich einen geringen Teil des Maschinentisches ein und lässt sich somit recht frei auf der Fläche bewegen.

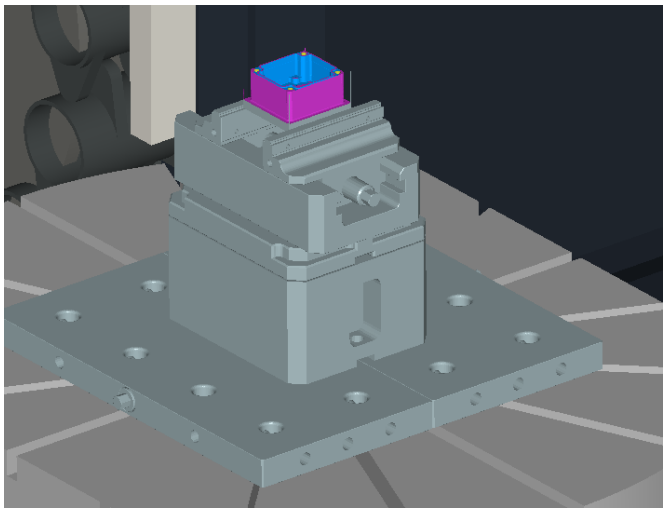

Abb. 20 Szenario 2. Aufnahme nach erfolgreicher NC-Bearbeitung

Abb. 20 zeigt einen Screenshot des zweiten Szenarios nach vollendeter NC-Bearbeitung. Szenario 3 ist unter gleichen Bedingungen in Abb. 21 dargestellt. Für Szenario 1 kann an dieser Stelle aus rechtlichen Gründen kein Screenshot gezeigt werden.

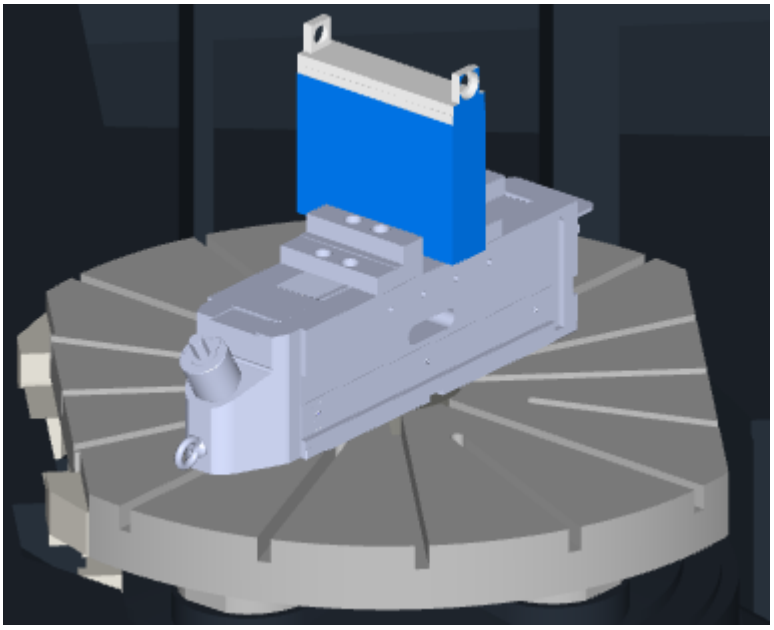

Abb. 21 Szenario 3. Aufnahme nach erfolgreicher NC-Bearbeitung

Für alle drei Szenarien kann zunächst ohne Kollisionsprüfung auf Basis der effizienten G-Code Interpretation das Optimierungspotential ermittelt werden. Für diesen Zweck werden die drei Szenarien, wie sie in Tabelle 3 dargestellt sind, mit Hilfe des G-Code Interpreters mit einer Partikelschwarmoptimierung optimiert. Zunächst sei dazu die Optimierung der Bearbeitungsdauer im 2-dimensionalen Fall betrachtet, also eine Translation in X- und Y-Richtung.

Tabelle 4 Optimierungspotential durch zweidimensionale Translation

Szenario	Optimierungspotential in Sekunden	Optimierungspotential in %
1	11,3 Sekunden	0.31%
2	5,1 Sekunden	0.54%
3	0 Sekunden	0%

Tabelle 4 zeigt nun das Potential, das entsteht, wenn man die Szenarien, so wie sie vorliegen, einer Optimierung bezüglich der Translation unterzieht. Die absoluten Zahlen betrachtend fällt zunächst auf, dass das Optimierungspotential subjektiv auf den ersten Blick recht enttäuschend aussieht. Dazu muss man allerdings bedenken, dass das Optimierungspotential mit der zu fertigenden Stückzahl skaliert. So können in einer Serienproduktion durchaus auch kleine Optimierungspotentiale pro Auftrag zu einem beträchtlichen Mehrgewinn in der Jahresbilanz führen oder die Flexibilität während der Arbeitsplanung signifikant verbessert werden. Szenario 3 liefert das erwartete Ergebnis und dient für diesen Versuch praktisch als Negativprobe. Da kein einziger Werkzeugwechsel vorliegt, wird auch kein Optimierungspotential erwartet. Das Experiment zeigt, dass genau dieser Fall hier eintritt.

Im Anschluss ergibt sich die Möglichkeit, zu testen, wie die Anzahl der Werkzeugwechsel für die genannten Szenarien bezüglich des Optimierungspotentials skalieren. Dazu kann man ein einfaches Experiment durchführen. An zufälligen Stellen im NC-Code werden weitere *M6*-Befehle eingefügt, um einen Werkzeugwechsel anzudeuten. Das heißt, an beliebigen Stellen im NC-Code werden künstlich Fahrten zum Werkzeugwechselpunkt induziert.

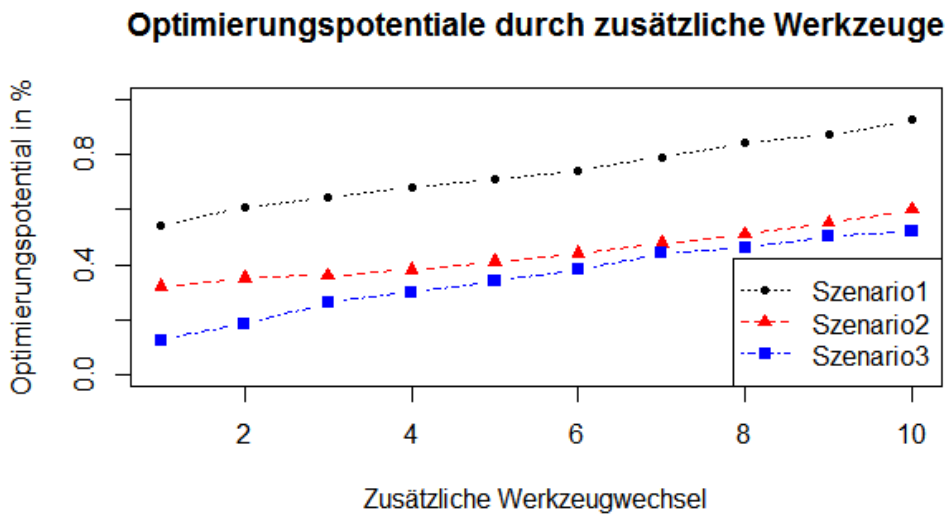

Abb. 22 Darstellung der theoretischen Optimierungspotentiale durch Einfügen virtueller Werkzeugwechsel

Abb. 22 zeigt, wie sich das relative Verhalten durch Einfügen zusätzlicher Werkzeuge entwickelt. Die Beobachtung entspricht dem, was man erwarten würde. Es liegt ein etwa linearer Anstieg des relativen Optimierungspotentials vor. Insofern unterstreicht diese Beobachtung, dass eine Optimierung der Werkstücktranslation besonders bei Bearbeitungen mit einer Vielzahl verschiedener Werkzeuge sinnvoll ist.

Darüber hinaus sei für die drei gegebenen Szenarien die dreidimensionale translatorische Optimierung betrachtet, die zusätzlich eine Veränderung der Z-Achse, also eine Verschiebung in die Höhe erlaubt. Es ist offensichtlich, dass ein solcher Fall in der Realität weniger interessant ist, da eine Veränderung der Aufspannlage in Z-Richtung einen Austausch des geplanten Spannmittels erfordert. Nichtsdestotrotz lässt sich dieser Fall virtuell modellieren und sollte in Betracht gezogen werden.

Tabelle 5 Theoretisches Optimierungspotential durch Translation im dreidimensionalen Raum

Szenario	Optimierungspotential in Sekunden	Optimierungspotential in %
1	25,7 Sekunden	0.72%
2	9,2 Sekunden	1%
3	0,09 Sekunden	0,02%

Tabelle 5 zeigt das Ergebnis dieser ziemlich theoretischen Betrachtung. Dabei ist allerdings interessant zu sehen, dass eine Verschiebung in Z-Richtung einen signifikanten Mehrgewinn für die Optimierung bieten könnte. Das hängt offensichtlich damit zusammen, dass der Weg vom Werkzeugwechselpunkt zum Werkstück logischerweise in Z-Richtung am größten ist.

Im Anschluss ist es interessant zu beobachten, wie die Szenarien sich nach der Optimierung bei der Auswertung der Validität verhalten. Dazu sei gesagt, dass aufgrund der begrenzten Ressourcen sämtliche Experimente nur einmalig durchgeführt werden konnten.

Tabelle 6 Optimierte Werte nach Auswertung der Validität unter Annahme verschiedener Clustergrößen

Szenario	Optimaler Wert nach Evaluation	#Cluster	#Iterationen
1	6,7 Sekunden := 0,19%	k = 2	4 (2)
		k = 4	3 (2)
		k = 8	1
2	5,1 Sekunden := 0,54%	k = 2	1
		k = 4	1
		k = 8	1
3	0 Sekunden = 0%	k = 2	NN
		k = 4	NN
		k = 8	NN

Hierbei sind natürlich grundsätzlich zwei Faktoren relevant. Die erste Fragestellung ist, ob das theoretische Optimierungspotential einen validen Wert liefert und falls nicht, wie viel von dem theoretischen Potential übrig bleibt. Auf der anderen Seite ist natürlich die Effizienz des Algorithmus interessant. Bei Betrachtung von Szenario 1 ist zu sehen, dass man in dem konkreten Fall acht Simulationsläufe bis zum Erreichen des Optimums braucht. Der Wert in Klammern deutet an, wie viele Iterationen notwendig waren, um ein Ergebnis auszugeben, das besser als die ursprüngliche Aufspannsituation gewesen ist. Hierbei hat es sich in den Fällen für k=2 und k=4 um den gleichen Vektor gehandelt, der eine Optimierung um etwa 4 Sekunden ergeben hat. Bei der Betrachtung von k=8 war der beste Wert gleichzeitig der erste, der als valide identifiziert wurde. Szenario 2 war wie erwartet für die Optimierung insofern recht einfach, weil die „erlaubten“

Amplituden für beide Achsen recht groß waren. Das theoretische Optimierungspotential entspricht hierbei genau dem praktischen Optimierungspotential, so dass der Algorithmus für jede Auswahl von k lediglich eine Iteration benötigt hat. Wenn das der Fall ist, ist es offensichtlich auch so, dass die Wahl der Zerlegung keine Rolle spielt, da die bestmögliche Konfiguration innerhalb seines sortierten Clusters stets an erster Stelle steht und somit unmittelbar ausgewertet wird. Szenario 3 muss hier nicht näher betrachtet werden. Da kein Optimierungspotential vorliegt, müssen auch keinerlei Simulationen durchgeführt werden.

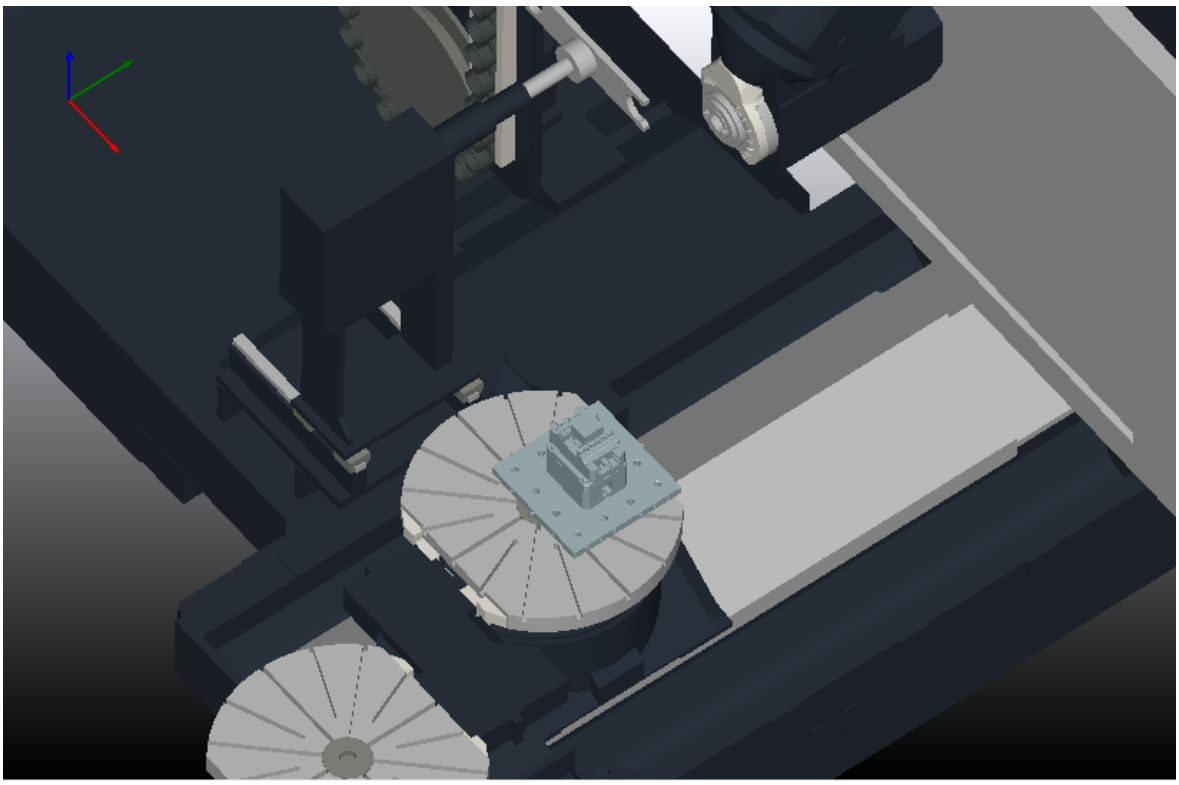

Abb. 23 Optimierte Aufspannposition für die Translation in X- und Y- Richtung

Abb. 23 zeigt die optimale Aufspannposition bei zweidimensionaler Betrachtung. Es zeigt sich eine deutliche Tendenz einer Verschiebung in Y-Richtung.

Tabelle 7 Optimierte Werte nach Auswertung der Validität unter Annahme verschiedener Clustergrößen (3D)

Szenario	Optimaler Wert nach Evaluation	#Cluster	#Iterationen
1	14,0 Sekunden := 0,3%	k = 2	3 (2)
		k = 4	1
		k = 8	1
2	9,2 Sekunden := 1%	k = 2	1
		k = 4	1
		k = 8	1
3	0,09 Sekunden = 0,02%	k = 2	1
		k = 4	1
		k = 8	1

Tabelle 7 zeigt die Auswertung der Evaluationen unter Berücksichtigung der Möglichkeit, eine Translation in Z-Richtung vornehmen zu können. Der Hauptaspekt, der sich aus der Tabelle ableitet, ist, dass eine Translation in Z-Richtung ganz offensichtlich deutlich weniger anfällig für die Verursachung von Kollisionen ist, was sich insbesondere in der Anzahl der benötigten Iterationen zeigt. Auch der kaum messbare Mehrgewinn für Szenario 3 ist in dem Fall evaluiert worden und hat mit der ersten Evaluation zu einem validen Ergebnis geführt. Abb. 24 zeigt die optimierte Aufspannsituation bei erlaubter Verschiebung in Z-Richtung. Man sieht deutlich, dass das Werkstück buchstäblich „in der Luft hängt“. Die Z-Richtung hat also eindeutig den größten Anteil an der Optimierung.

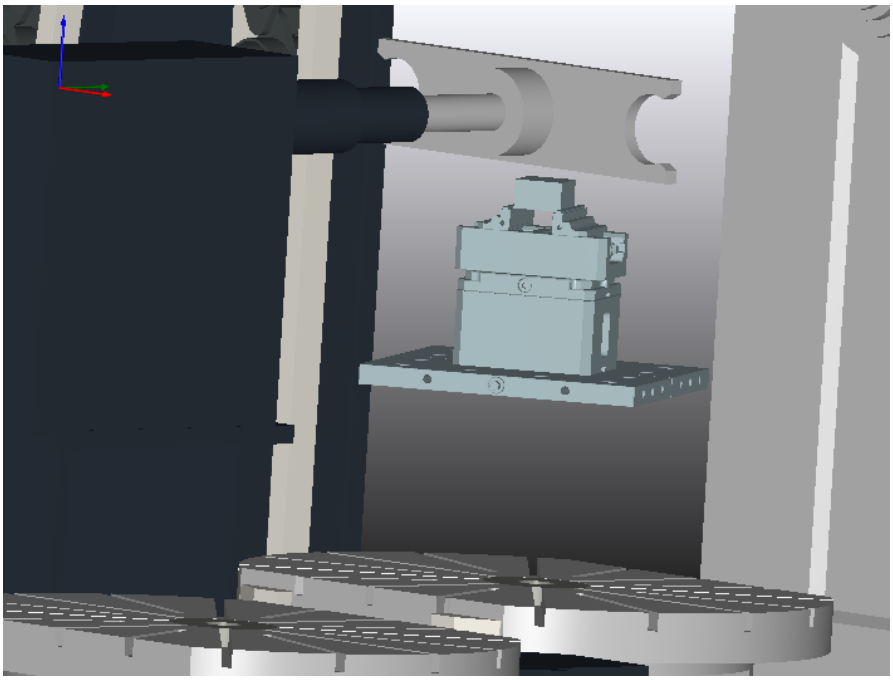

Abb. 24 Optimierte Aufspannsituation bei erlaubter Verschiebung in Z-Richtung

Im Folgenden seien noch die Auswertungen illustriert, die sich auf die Rotation eines Werkstücks beziehen. Zunächst wurde erwähnt, dass die Verlagerung der Achsbewegungen auf bestimmte Achsen theoretisch zu einem energetischen Einsparpotential führen kann. Es ist wichtig, zu erwähnen, dass diese Optimierung fast vollständig auf Annahmen beruht. Nichtsdestotrotz sei an der Stelle die allgemeine Formel für die kinetische Energie betrachtet:

$$E_{Kin} = \frac{1}{2} m v^2$$

Hierbei beschreibt m die Masse des sich bewegenden Körpers und v die Geschwindigkeit. Die Geschwindigkeit für die jeweiligen Achsen lässt sich mithilfe der Vorschübe und der kumulierten Achsauslenkungen problemlos berechnen. Für die Masse müssen, wie bereits erwähnt, Annahmen getroffen werden. Dazu wird hier von folgender Maschinentopologie ausgegangen: Die X- und Z-Achse seien verantwortlich für die Bewegung der Werkzeugspindel. Entlang der Y-Achse bewegt sich der Maschinentisch. Da keine konkreten Zahlen über die Massen, die bewegt werden müssen, bekannt sind, sei die Masse der Spindel konstant $m_{Spindel} = 1$. Dann lassen sich unterschiedliche Verhältnisse der Massen von Spindel und Maschinentisch ausschließlich über die Masse m_{Tisch} abbilden.

Tabelle 8 Optimierungspotential der kinetischen Energie durch Rotation

Szenario	**Relatives Optimierungspotential**	
1	$m_{Tisch} = 0{,}5$	5,2%
	$m_{Tisch} = 1$	0%
	$m_{Tisch} = 2$	9,4%
	$m_{Tisch} = 4$	10,3%
	$m_{Tisch} = 8$	11,0%
2	$m_{Tisch} = 0{,}5$	3,6%
	$m_{Tisch} = 1$	0%
	$m_{Tisch} = 2$	3,9%
	$m_{Tisch} = 4$	3,1%
	$m_{Tisch} = 8$	2,9%
3	$m_{Tisch} = 0{,}5$	4,8%
	$m_{Tisch} = 1$	0%
	$m_{Tisch} = 2$	4,0%
	$m_{Tisch} = 4$	2,2%
	$m_{Tisch} = 8$	1,2%

Tabelle 8 zeigt die theoretischen relativen Potentiale, die durch eine Veränderung der Rotation bezüglich der kinetischen Energie unter bestimmten Annahmen entstehen. Man sieht nicht wirklich eine Regelmäßigkeit in den drei Szenarien. Es existiert allerdings wiederum eine Negativprobe, die dann greift, wenn sämtliche Massen identisch sind. Für

den Fall hat die Rotation keine Auswirkung auf die kinetische Energie, da auch die kumulierten Geschwindigkeiten konstant bleiben. Eine detaillierte Auswertung zur Kollisionsprüfung erübrigt sich an dieser Stelle, da sämtliche Rotationen um die Z-Achse zu einer kollisionsfreien Bearbeitung geführt haben.

Schließlich wird noch kurz die letzte genannte Optimierungsebene ausgewertet, die im Rahmen der möglichen Optimierungsgrößen vorgestellt wurde. Es stellt sich die Frage, ob sich das Volumen der Gesamtbearbeitung einschränken lässt.

Tabelle 9 Ermittlung des Optimierungspotential für die Bearbeitungsfläche

Szenario	Maximale relative Einschränkung der Bearbeitungsfläche
1	11,4%
2	9,1%
3	1,5%

Tabelle 9 zeigt die Ergebnisse dieses Experiments. Grundsätzlich lässt sich die Bearbeitungsfläche reduzieren. Ob dadurch tatsächlich Potentiale für die Durchführung auf alternativen Maschinen entstehen, ist im Rahmen des Projekts nicht untersucht worden und könnte Gegenstand zukünftiger Forschungsarbeiten sein.

4.5. Theoretische Untersuchungen

Das beschriebene hybride Verfahren, das auf der effizienten Berechnung von Kennzahlen eines G-Code Programms basiert, ist in erster Linie stark durch den spezifischen Anwendungsfall motiviert. Darüber hinaus sind die wesentlichen Anforderungen, die zu dem Vorgehen geführt haben, aus dem Zusammenspiel der Komponenten des Gesamtsystems innerhalb des Verbundprojekts (siehe Kapitel 6) gewachsen, etwa die Notwendigkeit asynchroner Simulationsläufe. Nichtsdestotrotz ist es sinnvoll, Untersuchungen anzustellen, inwiefern das entwickelte Verfahren für unterschiedliche Domänen generalisierbar ist. Diese Fragestellung ist Inhalt des folgenden Kapitels.

Hierfür ist es zunächst notwendig, unterschiedliche Benchmarkszenarien zu entwerfen, die als eine Art Platzhalter für Simulationen dienen. In der Betrachtung der asynchronen Durchführung der Partikelschwarmoptimierung wurde bereits die Griewankfunktion vorgestellt. Es seien an dieser Stelle weitere übliche Testfunktionen für die numerische Optimierung vorgestellt:

$$f_{Rastrigin}(x) = An + \sum_{i=1}^{n} [x_i^2 - A\cos 2\pi x_i\ ], x \in \mathbb{R}^n$$

Die Rastriginfunktion ist eine n-dimensionale Funktion. $A = 10$ beschreibt hierbei eine Konstante. Abb. 25 zeigt sowohl den Konturplot als auch den 3D-Plot der Funktion. Es ist klar zu erkennen, dass die Schwierigkeit für die Optimierung in der großen Anzahl lokaler Minima liegt.

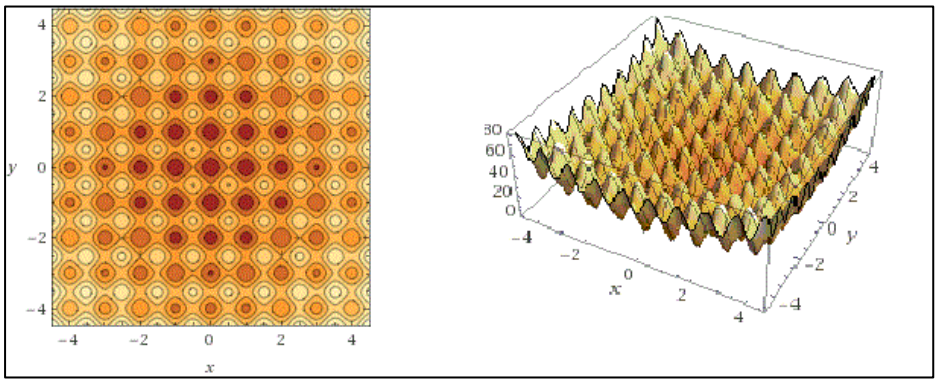

Abb. 25 Konturplot und 3D-Plot der Rastriginfunktion

Eine ebenfalls übliche Funktion in dem Zusammenhang ist die Rosenbrockfunktion:

$$f_{Rosenbrock}(x) = \sum_{i=1}^{N-1} [100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2], x \in \mathbb{R}^n$$

In der Literatur wird die Rosenbrockfunktion häufig als Rosenbrocks Bananenfunktion bezeichnet. Das lässt sich darauf zurückführen, dass das globale Optimum innerhalb eines gekrümmten Tals zu finden ist, wobei dieses Tal sich durch eine Vielzahl lokaler Minima auszeichnet.

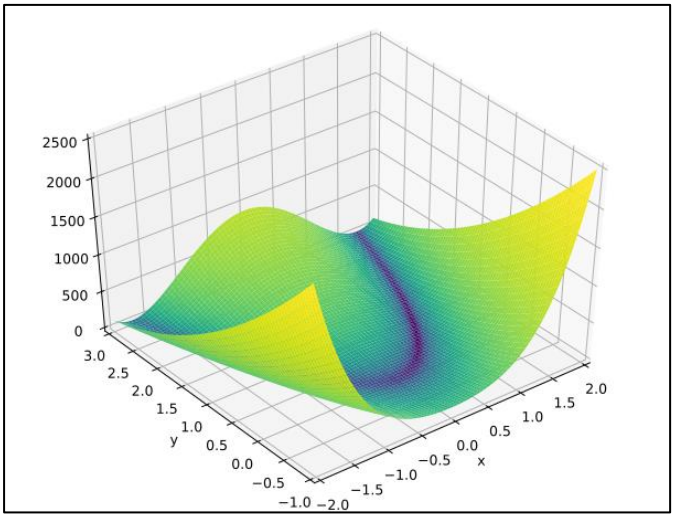

Abb. 26 3D-Darstellung der Rosenbrockfunktion [58]

Als eine Art Negativprobe wird den eher komplexen Funktionen mit einer hohen Anzahl lokaler Minima häufig die generalisierte Quadratfunktion entgegengestellt:

$$f_{quadrat}(x) = \sum_{i=1}^{n} x_i^2$$

Der hybride Ansatz beruht auf der Annahme, dass die Fitnessfunktion einer Abbildung der Form $f: \mathbb{R}^n \to \mathbb{R} \times \{0,1\}$ entspricht, die sich in zwei separate Funktionen $\bar{f}: \mathbb{R}^n \to \mathbb{R}$ und $g: \mathbb{R}^n \to \{0,1\}$ zerlegen lässt, so dass $f(x) := (\bar{f}(x), g(x))$. Die Benchmarkfunktionen, die bislang vorgestellt wurden, sind offensichtlich lediglich in der Lage, $\bar{f}(x)$ zu repräsentieren. Um Aussagen darüber treffen zu können, wie der vorgestellte Ansatz generalisiert eingesetzt werden kann, ist es notwendig, Abbildungen der Form $g: \mathbb{R}^n \to \{0,1\}$ für beliebige $n \in \mathbb{N}$ zu erzeugen, die ein möglichst realistisches Abbild verschiedener Simulationsszenarien darstellen. Dazu muss zunächst diskutiert werden, welche Anforderungen an die Gestaltung dieser Abbildungen gestellt werden müssen:

1. **Zusammenhängende „Inseln“**
   Um eine Generalisierbarkeit des Ansatzes feststellen zu können, ist dies ein wesentlicher Aspekt. Die zentrale Aussage dahinter ist, dass benachbarte Teilmengen der Eingangsgrößen eine hohe Wahrscheinlichkeit aufweisen sollten, der gleichen Ausgangsgröße (also 0 oder 1) zugeordnet zu werden. Formal könnte man sich die gewünschte Struktur als Voronoi-Diagramm vorstellen, wobei die Zahl der Kanten überschaubar sein sollte.

2. **Ermöglichung nichtlinearer Grenzen zwischen den Zuordnungen**
   Voraussichtlich ist von einem realistischen Szenario nicht zu erwarten, dass Grenzen zwischen zwei Zentren mit unterschiedlicher Ausgangsgröße linear verlaufen. Aus diesem Grund sollte die Möglichkeit nichtlinearer Zusammenhänge unbedingt sichergestellt sein.

3. **Ermöglichung verschieden verteilter Ausgangsgrößen**
   Dieser Punkt soll dem Aspekt Rechnung tragen, dass die Verteilung „erlaubter“ Eingangsgrößen in unterschiedlichen Szenarien einer extremen Diversität unterliegt. Diese Beobachtung ist bei der Analyse der Einrichtungsoptimierung von Werkzeugmaschinen zum Tragen gekommen. Einige Werkstücke (vor allem große) erlauben beispielsweise fast keinen Spielraum für die Positionierung, während andere sich nahezu frei auf dem Tisch bewegen lassen. Dieser Effekt ist für eine Vielzahl von Domänen und vor allem domänenübergreifend ebenfalls zu erwarten. Aus dem Grund sollte sichergestellt sein, dass künstliche Szenarien

erzeugt werden können, die in der Mehrheit unerlaubte bzw. erlaubte Areale enthalten.

4. **Anpassbarkeit des Definitionsraums**
   Es ist notwendig, dass die Evaluationen von $\bar{f}(x)$ und $g(x)$ in vergleichbaren Definitionsräumen mit gleicher Dimension stattfinden. Da die vorgestellten Benchmarkfunktionen für $\bar{f}(x)$ auf verschiedenen, endlichen Intervallen für die Eingangsgrößen operieren und nicht der komplette reelle Zahlenraum betrachtet wird, müssen diese Intervalle als Basis für $g(x)$ zugrunde gelegt werden. Andernfalls kann keine Vergleichbarkeit sichergestellt werden.

5. **Semantische Unabhängigkeit von der numerischen Fitnessfunktion**
   Es muss die Möglichkeit gewährleistet werden können, dass die Evaluation von $g(x)$ in keiner Abhängigkeit zu $\bar{f}(x)$ steht. Die Formulierung ist bewusst etwas vage gehalten, da dieser Fall nicht gänzlich ausgeschlossen werden sollte. So ist wäre folgendes Szenario denkbar. Sei $\bar{f}: f_{Rastrigin}$ die numerische Evaluierungsfunktion zur Kandidatenbestimmung und:

$$g(x) = \begin{cases} 1, falls\ k_1 \leq \bar{f}(x) < k_2 : k_1, k_2 \in \mathbb{R} \\ 0, andernfalls \end{cases}$$

   In dem Fall wären also sämtliche Eingangssignale erlaubt, deren numerische Fitnessevaluation in ein bestimmtes Intervall fällt. Da durchaus Szenarien denkbar sind, in denen eine derartige Bedingung realistisch ist, stellt diese Bedingung keine harte Einschränkung für die Konstruktion von $g(x)$ dar. Daneben ist es jedoch wichtig, Szenarien zu entwerfen, die einer vollständigen Unabhängigkeit der beiden Funktionen unterliegen, da in vielen Fällen kein Zusammenhang zwischen $\bar{f}(x)$ und $g(x)$ besteht.

Auf der Basis kann ein Versuchsaufbau erstellt werden, der die oben genannten Kriterien – teilweise mit Einschränkungen – erfüllt. Zu diesem Zweck sei im Folgenden eine Methode aus dem Gebiet des überwachten maschinellen Lernens vorgestellt, die sogenannte Support Vector Machine. Darauf aufbauend wird im Anschluss erläutert, wie das Verfahren in diesem Zusammenhang genutzt werden kann.

4.5.1. Support Vector Machines

Wie bereits erwähnt gehören Support Vector Machines (im Folgenden *SVM*) zur Klasse der überwachten Lernverfahren und werden üblicherweise als Klassifikator eingesetzt, wobei auch eine Anwendung für nichtlineare Regressionsprobleme möglich ist. Die Methode wurde zuerst von Cortes und Vapnik vorgestellt [59]. Sei $\mathrm{A} := \{(\vec{a}_i, b_i), \vec{a}_i \in \mathbb{R}^d, b_i \in \{-1,1\}\}$ eine Menge an Datenpunkten mit einer binären Zuweisung. Hierbei

beschreibt jeder Vektor $\vec{a}_{j,j\geq i}$ eine Beobachtung und der Skalar b die Zuweisung zu einer Klasse. Gesucht ist eine Hyperebene, die die Datenpunkte der jeweiligen Klassen separiert. Formal lässt sich eine Hyperebene beschreiben als $\{\vec{x} \mid \vec{x} \cdot \vec{w} + c = 0\}$, wobei $\vec{w}$ einen Normalvektor, und c den sogenannten Bias beschreibt. Die Zugehörigkeit einer Klasse bei gegebener Hyperebene lässt sich ausdrücken als:

$$\vec{b}_i = sgn(\vec{x} \cdot \vec{w} + c)$$

Für linear separierbare Daten lässt sich zeigen, dass die optimale Hyperebene durch Minimierung des Terms $\frac{1}{2}\|\vec{w}\|_2^2$ gefunden werden kann, wobei $\vec{w}$ und c als Freiheitsgrade adjustiert werden können.

Für den Fall, dass Daten nicht linear separierbar sind, können die Daten mittels eines Kernels in einen höherdimensionalen Datenraum abgebildet werden. Sei $\phi: \mathbb{R}^d \to \mathbb{R}^{d\prime}$ mit $d \leq d'$. Dann ergibt sich der Klassifikator aus der Optimierung der Hyperebene $\{\phi(\vec{x}) \mid \phi(\vec{x}) \cdot \vec{w} + c = 0\}$. In der Praxis existiert eine Vielzahl an bewährten Kernelfunktionen. Da die Support Vector Machine im vorliegenden Zusammenhang nur ein Werkzeug für die Erzeugung passender Benchmarks darstellt, sei für einen tieferen Einblick auf folgende Referenzen verwiesen: [60] [61].

Da es sich bei dem Verfahren um einen binären Klassifikator handelt, ist eine minimale Erweiterung notwendig, um beliebig viele Klassen separieren zu können. Diese besteht lediglich daraus, je nach Anzahl unterschiedlicher Klassen weitere SVMs hinzuzufügen und auf diese Art und Weise entsprechend viele Hyperebenen zu berechnen.

4.5.2. Konstruktion binärer Landschaften mit Benchmarkdatensätzen

Wie die vorangegangene Erläuterung zeigt, ermöglicht es die SVM, aus einer gegebenen Trainingsmenge ein Datenmodell zu entwickeln, das einem Datenpunkt, der dem Modell präsentiert wird, eine Klasse zuweist. Es stellt sich nun die Frage, wie das Verfahren im vorliegenden Zusammenhang genutzt werden kann. Die Idee ist, verschiedene Datensätze zu nutzen, um aus diesen einen binären Klassifikator zu lernen, der als die Funktion $g(x)$ repräsentiert. Die Datensätze, die hierfür verwendet werden, stammen aus dem Paket *mlbench* [62] für die Programmiersprache R.

Tabelle 10 Datensätze als Grundlage für die Erstellung von g(x). Datensätze, die künstlich erzeugt werden, sind durch einen kursiven Namen gekennzeichnet

Name des Datensatz	Anzahl Beobachtungen	Anzahl Dimensionen	Analyseziel
Boston Housing Data	506	14	Regression
Wisconsin Breast Cancer	699	10	Binäre Klassifikation
Glass Identification Database	214	10	Mehrklassenklassifikation ➔ 7 Klassen
John Hopkins University Ionosphere Database	351	35 – 2	Binäre Klassifikation
Letter Image Recognition Data	20000	17	Mehrklassenklassifikation ➔ 26 Klassen
Friedman1	*Beliebig*	*10*	*Regression*
Peak Benchmark Problem	*Beliebig*	*Beliebig*	*Regression*
XOR Problem	*Beliebig*	*Beliebig*	*Mehrklassenklassifikation* ➔ 2^{d-1} Klassen ➔ $d := \#Dim$
Pima Indians Diabetes Database	768	9	Binäre Klassifikation

Tabelle 10 zeigt eine Auswahl von Datensätzen, mit deren Hilfe Abbildungen erzeugt werden, die der Form $g: \mathbb{R}^n \to \{0,1\}$ entsprechen, wobei $n \in \mathbb{N}$ beliebig gewählt werden kann. Beim Blick auf die Spalte „Analyseziele“ kann man erkennen, dass die Datensätze unterschiedlichen Strukturen folgen, was mit den spezifischen Fragestellungen zusammenhängt, für die diese Benchmarks ursprünglich erzeugt wurden. Zur Erläuterung der Tabelle sei gesagt, dass es eine Unterscheidung zwischen künstlich erzeugten und „echten“ Datensätzen gibt. Die künstlich erzeugten Datensätze sind in der Tabelle kursiv gehalten und zeichnen sich durch eine gewisse Flexibilität aus, was die Anzahl der Beobachtungen bzw. die Anzahl der Dimensionen betrifft. Diese Datensätze seien zunächst näher betrachtet:

1. **Friedman1 Regression**
   Dieses Regressionsproblem wurde 1991 von Jerome H. Friedman vorgestellt [63]. Sei $n \in \mathbb{N}$ die Anzahl der gewünschten Beobachtungen. Dann wird die Trainingsmenge $T \in \mathbb{R}^{10 \times n} \times \mathbb{R}$ folgendermaßen erstellt:
   Die Eingangsvektoren $(x_1, x_2, \dots, x_{10})$ werden aus einer gleichverteilten Zufallsvariable im Interval $[0,1]$ erzeugt. Auf der Basis wird die abhängige Variable y berechnet als:

$$y = 10 \sin(\pi x_1 x_2) + 20(\pi x_3 - 0{,}5)^2 + 10x_4 + 5x_5 + e, e \sim \mathcal{N}(0, sd)$$

Die Abkürzung sd bezeichnet hierbei die Standardabweichung des Rauschterms. Die Standardabweichung ist ein Freiheitsgrad bei der Erstellung des Datensatzes, die standardmäßig auf 1 gesetzt ist. Wie man sieht, liefern die Eingangsgrößen $(x_6, x_7, \dots x_{10})$ keine relevanten Informationen für die Ausgangsgröße.

2. **Peak Benchmark Problem**
   Auch hierbei handelt es sich um ein Regressionsproblem. Sei $r = 3u, u \sim U(0,1)$, $d \in \mathbb{N}$ die Anzahl der gewünschten Dimensionen und $n \in \mathbb{N}$ die Anzahl der gewünschten Beobachtungen. Dann wird die Trainingsmenge $T \in \mathbb{R}^{10\times n} \times \mathbb{R}$ gebildet mit $(x_1, x_2, \dots, x_d)$ als gleichverteilte Zufallsvariablen innerhalb der d-dimensionalen Kugel mit Radius r und der Zielvariable $y = 25\exp(-0.5r^2)$.

3. **XOR Problem**
   Das XOR Problem gilt als eines der bekanntesten Beispiele für nicht linear separierbare Trainingsdaten. Sei erneut $n \in \mathbb{N}$ die Anzahl der Beobachtungen und $d \in \mathbb{N}$ die Anzahl der Dimensionen.

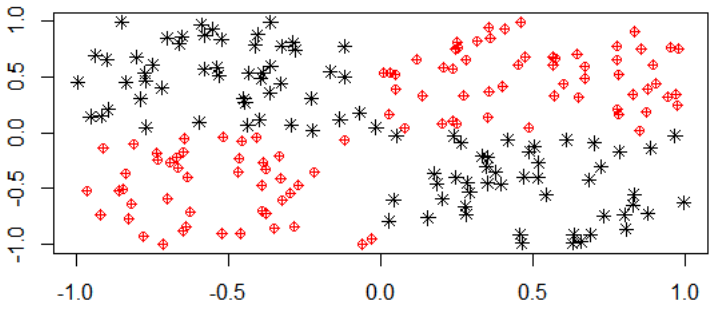

Abb. 27 Beispielplot für einen kontinuierlichen XOR-Datensatz mit 2 Dimensionen

Die Trainingsmenge $T \in \mathbb{R}^{10\times n} \times \mathbb{N}$ wird dann folgendermaßen erzeugt. Die Eingangsvektoren$(x_1, x_2, \dots, x_d)$ werden gleichverteilt im Intervall $[-1, +1]$ gezogen.

Die Zuordnung kann folgendermaßen definiert werden: Zwei Vektoren $(x_1, \dots, x_d), (\tilde{x}_1, \dots, \tilde{x}_d)$ werden dann derselben Klasse zugeordnet, falls folgender prädikatenlogischer Ausdruck gilt:

$$\sum_{i=1}^{d} 2^{sgn'(x_i)} = \sum_{i=1}^{d} 2^{sgn'(\tilde{x}_i)}, \text{ wobei}$$

$$sgn'(x_i) = \begin{cases} 1, falls\ sgn(x_i) \geq 0 \\ 0, falls\ sgn(x_i) < 0 \end{cases}$$

Einfach ausgedrückt bedeutet das, dass zwei Punkte, die im gleichen Quadranten oder im gegenüberliegenden Quadranten liegen, zu einer Klasse gehören. Auf die Art und Weise entstehen genau 2^{d-1} verschiedene Klassen. Abb. 27 verdeutlicht, dass eine lineare Separation eines solchen Datensatzes nicht möglich ist.

Die übrigen Datensätze sind weder in ihrer Dimensionalität noch bezüglich ihrer Beobachtungsmenge flexibel, da es sich um Ergebnisse aus realen Studien handelt. Eine kleine Erläuterung ist hierbei lediglich für den Datensatz *„John Hopkins University Ionosphere Database“* notwendig, da in der Tabelle „35 – 2“ als Anzahl der Dimensionen erfasst ist. Das hängt damit zusammen, dass es sich bei zwei der Eigenschaften um kategorielle Größen handelt. Da in diesem Zusammenhang nur numerische Werte relevant sind, werden diese beiden Größen ignoriert und der Datensatz wird als 33-dimensioneller, realwertiger Vektorraum aufgefasst.

Auf den ersten Blick scheint die feste Anzahl der Dimensionen ein Problem im vorliegenden Kontext darzustellen. Als die Bedingungen an die Erstellung von geeigneten Funktionen $g(x)$ definiert wurden, wurde klar definiert, dass die Anzahl der Dimensionen für die Eingangsgrößen von $\bar{f}(x)$ und $g(x)$ logischerweise gleich sein müssen. Eine einfache Lösung, diese Problematik zu umgehen, wäre es, unter Annahme eines bestimmten Datensatz für die Erstellung der Funktion g die Dimensionalität für $\bar{f}$ festzusetzen. Unter den Umständen könnte die numerische Optimierung mit einer der beschriebenen Benchmarkfunktionen (z.B. $f_{Rastrigin}$) lediglich auf 10 Dimensionen durchgeführt werden, falls für g ein Schätzer auf Basis des *„Wisconsin Breast Cancer“* Datensatzes gewählt wurde. Da allerdings eine höhere Flexibilität wünschenswert ist, wird ein Instrument eingeführt, das es ermöglicht, die Datensätze bezüglich ihres zugrunde liegenden Vektorraums beliebig zu skalieren.

Sei $x = (x_1, x_2, \dots, x_{d_1})$ ein Vektor, dessen Ausgangsgröße (also 0 oder 1) mithilfe einer Funktion $g(\check{x})$ berechnet wird, wobei $\check{x} = (\check{x}_1, \check{x}_2, \dots, \check{x}_{d_2})$ mit $d_1, d_2 \in \mathbb{N}, d_1 \neq d_2$. Offensichtlich gibt es nun zwei Möglichkeiten:

1. $\boldsymbol{d_1 < d_2}$
   Für den Fall, dass die Funktion $g(x)$ auf einer größeren Menge an Eingangsgrößen als die Funktion $\bar{f}(x)$ operiert, kann auf zwei grundliegende Instrumente zurückgegriffen werden. Das naive Vorgehen würde darin liegen, eine beliebige Teilmenge von x zugrunde zu legen, also einen Teil der Messgrößen zu ignorieren.
   Grundsätzlich wäre in diesem Fall auch die Durchführung einer Dimensionsreduktion, etwa durch Hauptkomponentenanalyse oder Autoencoder, möglich.

2. $d_1 > d_2$

   Der Fall, dass die Datenbasis des gewählten Datensatz im Vergleich zum Vektorraum für die numerische Evaluation eine zu niedrige Anzahl Dimensionen aufweist, sieht auf den ersten Blick komplexer aus. Es wird nach einer Lösung gesucht, eine Transformation in einen höherdimensionalen Vektorraum zu ermöglichen, ohne dabei unnötiges Rauschen zu erzeugen. Für diesen Zweck wird an dieser Stelle ein Aspekt ausgenutzt, der bei künstlichen neuronalen Netzen eine Rolle spielt. Hierzu sei der grobe Aufbau eines solchen Netzes skizziert.

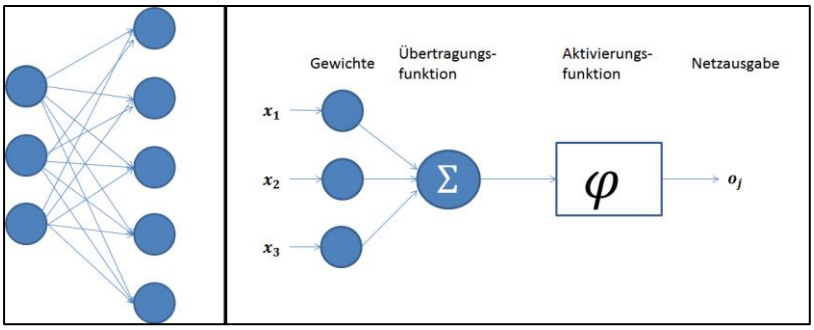

Abb. 28 Grobe Skizzierung der Funktionsweise von neuronalen Netzen. Links: Topologie eines Netzes mit Ein- und Ausgabeschicht; Rechts: Funktionsweise eines künstlichen Neurons

Abb. 28 zeigt exemplarisch den grundsätzlichen Aufbau eines künstlichen neuronalen Netzes in extrem vereinfachter Darstellung, die für die vorliegenden Zwecke allerdings absolut ausreichend ist. Auf der linken Seite der Abbildung ist ein Netz mit dreidimensionaler Ein- und fünfdimensionaler Ausgabeschicht skizziert. Die Kanten des Graphen sind hierbei gewichtet. In dem skizzierten Fall ist jedes Neuron der Ausgabeschicht mit jedem Neuron der Eingabeschicht verbunden. Das heißt, dass sämtliche Eingaben einen Effekt auf sämtliche Ausgaben haben. Auf der rechten Seite der Abbildung wird dargestellt, wie ein Neuron auf Basis seiner Eingaben aktiviert wird. Zunächst werden die Eingaben, die zu dem entsprechenden Neuron führen, durch eine Übertragungsfunktion aggregiert. Im vorliegenden Fall sei das immer die Summe der gewichteten Eingaben. Der aggregierte Wert dient als Eingabe für die Aktivierungsfunktion φ. Dabei handelt es sich in den meisten Fällen um eine monoton steigende Funktion. Im einfachsten Fall ist die Aktivierungsfunktion linear, also beispielsweise einfach $\varphi(x) = x$. Eine tiefere Betrachtung der Funktionsweise von künstlichen neuronalen Netzen ist an dieser Stelle nicht von Interesse, da die bisherige Darstellung die Bedingungen für die vorliegenden Zwecke bereits erfüllt. Sei $\vec{x} \in \mathbb{R}^{d_2}$ und $\tau: \mathbb{R}^{d_2} \to \mathbb{R}^{d_1}$ die gewünschte Abbildung in den höherdimensionalen Vektorraum. Für die vorliegenden Zwecke wird τ modelliert als zweischichtiges neuronales Netz mit zufälligen Gewichten w_{d_2,d_1},einer linearen Aktivierungsfunktion und die Übertragungsfunktion $\sum_{i=1}^{d_2} x_i w_{i,j}$, wobei $j \in [1, d_1]$ ein Neuron der Ausgabeschicht darstellt. Hierbei sei angemerkt, dass das Netz in keiner Weise trainiert, sondern einfach zufällig initialisiert wird, um die Abbildung in den d_1-dimensionalen Raum zu realisieren.

4.5.3. Experimente und Auswertung

Auf Basis der beschrieben Methoden und Referenzfunktionen bzw. Referenzdatensätze lässt sich nun eine analytische Versuchsanordnung erstellen, mit deren Hilfe überprüft werden kann, inwieweit das vorgestellte Verfahren für beliebige Problemstellungen generalisiert werden kann. Hierbei muss beachtet werden, dass die hohe Anzahl der Freiheitsgrade auf vielen Ebenen der Datenverarbeitungskette kombinatorisch zu einer recht unübersichtlichen Anzahl relevanter Fragestellungen führen kann. Aus diesem Grund werden die theoretischen Untersuchungen iterativ durch die vorgestellte Schrittkette vorgestellt. Zu Beginn werden Beobachtungen dokumentiert, die ausschließlich auf Basis der Optimierungen gemacht wurden, aus denen erste Ableitungen getroffen werden können, wie sich unterschiedliche Verfahren unter ähnlichen Bedingungen verhalten. Darüber hinaus können die Beobachtungen bei der Bewertung helfen, ob die gewählten Benchmarkfunktionen ausreichend sind, um damit realistische Simulationsszenarien abzubilden.

Untersuchung des Explorationsverhaltens verschiedener Optimierungsalgorithmen bezüglich der definierten Benchmarkfunktionen

Für diese Versuchsreihe seien die Optimierungsverfahren PSO, Simulated Annealing und Genetische Algorithmen betrachtet. Als Benchmarkfunktionen dienen $f_{Rastrigin}$, $f_{Griewank}$, $f_{Rosenbrock}$ und $f_{quadrat}$. Zunächst werden die jeweiligen Optimierungsverfahren unter Annahme ihrer Standardparameter verwendet, da diese sich in diversen realistischen Optimierungsszenarien bewährt haben.

Die Parameter sind demnach folgendermaßen gewählt (Tabelle 11):

Tabelle 11 Standardparametrierung der Optimierungsalgorithmen

Partikelschwarmoptimierung	
Konvergenz	Fixe Anzahl Iterationen $n = 1000$
Schwarmgröße	$\lfloor 10 + 2 * \sqrt{d} \rfloor$
Trägheit	$\frac{1}{2 * \log(2)}$
Faktor für globales Optimum	$\frac{1}{2} + \log(2)$
Faktor für lokales Optimum	$\frac{1}{2} + \log(2)$
Simulated Annealing	
Konvergenz	Keine Verbesserung der Fitness
Initiale Temperatur	3000
Genetischer Algorithmus	
Konvergenz	Fixe Anzahl Iterationen $n = 100$
Populationsgröße	50
Crossoverwahrscheinlichkeit	0,8
Mutationswahrscheinlichkeit	0,1

Für die ersten Betrachtungen sei die Anzahl Dimensionen $d = 5$ festgelegt. Die Wahl dieser Größe hat insbesondere Visualisierungsgründe. Die Fokussierung auf eine einzelne Dimension lässt nichtsdestotrotz interessante Rückschlüsse zu, wie die entsprechenden Grafiken verdeutlichen werden.

Sei $X \in \mathbb{R}^{d \times n}$eine Matrix mit den Zeileneinträgen $\vec{\chi}_i, i \in [1, n], \vec{\chi}_j \in \mathbb{R}^d \; \forall j$, die sämtliche Größen enthält, die während der Optimierungsphase evaluiert worden sind. Sei zusätzlich $\Upsilon \in \mathbb{R}^{\mathrm{n}}$ der Vektor, der die entsprechenden Fitnesswerte enthält, also $\Upsilon_i = f(\chi_{\mathrm{i}})$. Der Wert $\theta \in \mathbb{R}$ beschreibt einen beliebigen Schwellwert, der anzeigt, welcher Grenzwert unterschritten werden muss, um für den vorliegenden Zweck als Kandidat zu gelten.

Dann kann folgende Menge definiert werden:

$$\overline{\Upsilon} := \{\vec{\chi}_i, \vec{\chi}_i \in X | \Upsilon_i \leq \theta\}$$

Das ist offensichtlich genau die Menge der evaluierten Ausgangsgrößen, die einen gesetzten Schwellwert nicht unterschreiten. Für die Rastriginfunktion wird $\theta = 0{,}75$ betrachtet.

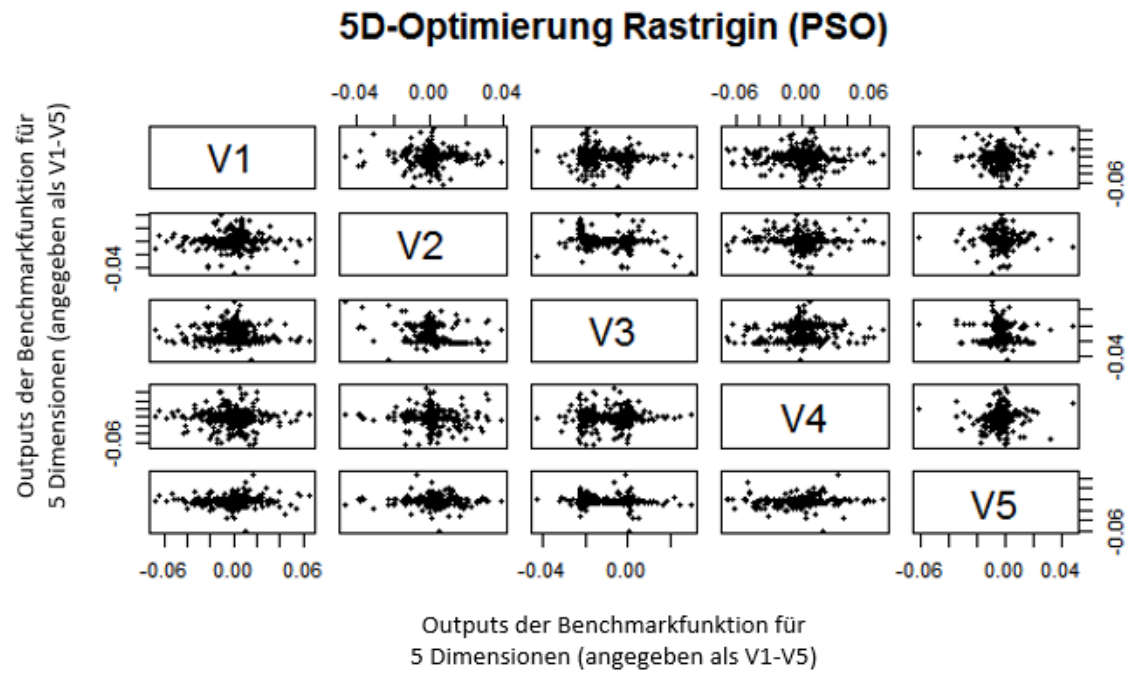

Abb. 29 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rastriginfunktion mittels PSO mit Standardparametern als paarweise Scatterplot

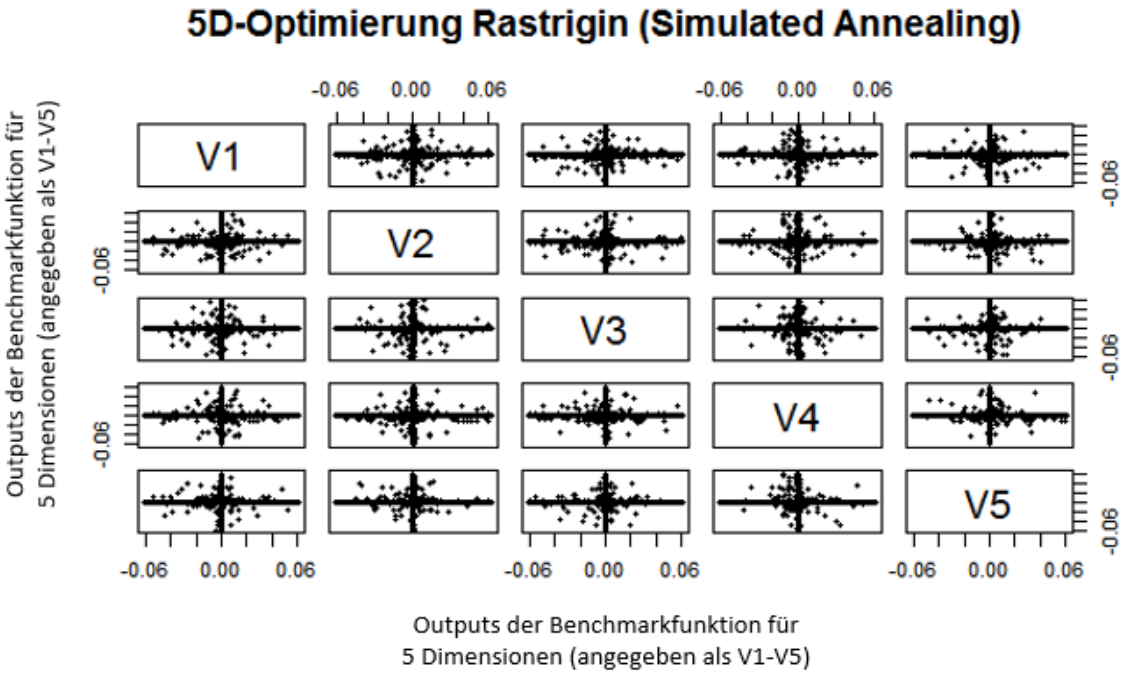

Abb. 30 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rastriginfunktion mittels Simulated Annealing mit Standardparametern als paarweise Scatterplot

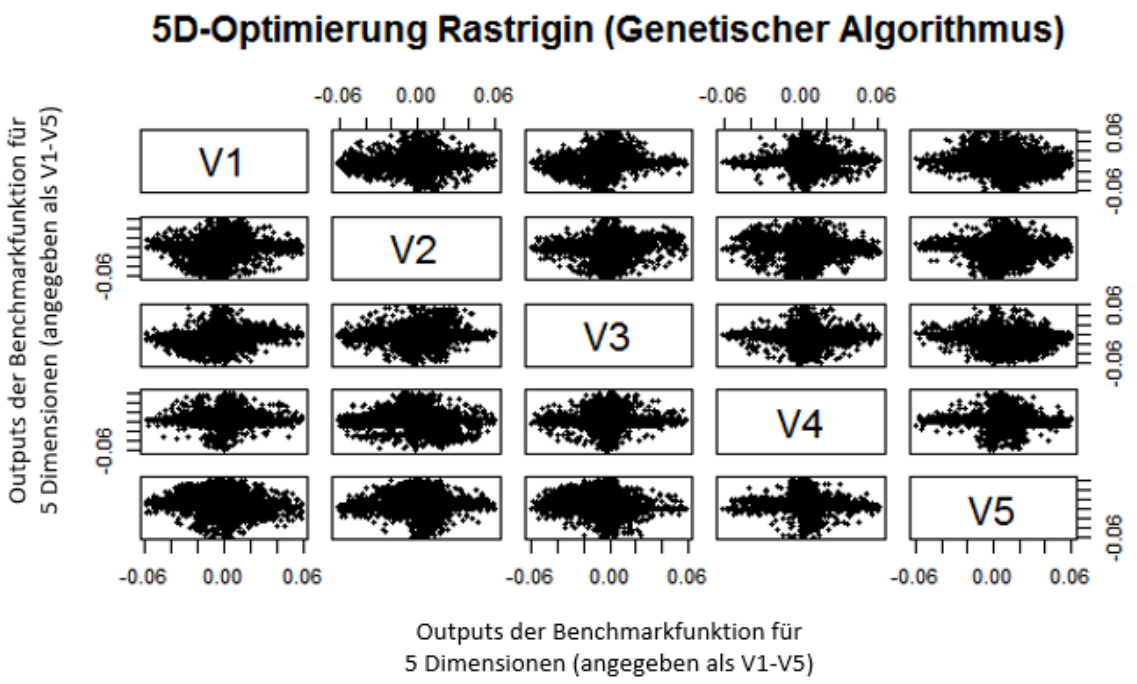

Abb. 31 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rastriginfunktion mittels genetischem Algorithmus mit Standardparametern als paarweise Scatterplot

Die Abbildungen Abb. 29, Abb. 30 und Abb. 31 zeigen in paarweisen Scatterplots die Struktur der besten Ergebnisse für die Rastriginfunktion an. Hierbei kann man einige interessante Beobachtungen machen. Logischerweise haben sämtliche Methoden ihren Schwerpunkt etwa im Ursprung, da hier auch das globale Optimum zu finden ist. Daran lässt sich erkennen, dass hier sämtliche Verfahren in der Lage sind, das globale Optimum zu erfassen. Was man eindeutig sieht, ist das unterschiedliche explorative Verhalten der beiden populations- beziehungsweise schwarmbasierten Verfahren im Gegensatz zur Einzelpunktsuche mittels Simulated Annealing. Simulated Annealing zeigt hierbei nur wenige Abweichungen zu einem Kreuzmuster, das sich in sämtlichen Parameterpaaren identifizieren lässt. Das ist für den vorliegenden Kontext insofern eine hochinteressante Beobachtung, da ein zu starker Fokus auf bestimmte Muster innerhalb des Suchraums dazu führen kann, dass valide Kandidaten laut $g(x)$, von dem Gesamtverfahren übersehen werden. Bezogen auf die Standardeinstellungen der Algorithmen sieht man zusätzlich, dass die Partikelschwarmoptimierung deutlich weniger Kandidaten erzeugt als der genetische Algorithmus. Gleichzeitig werden offensichtlich einige lokale Optima innerhalb des Suchraums ausgespart. Dadurch verpasst die Partikelschwarmoptimierung möglicherweise ebenfalls Kandidaten, die sich durch die Evaluation von $g(x)$ als valide erweisen könnten.

Zunächst wird nun überprüft, ob sich das Verhalten basierend auf den anderen Benchmarkfunktionen reproduzieren lässt. Dazu wird nun die Rosenbrockfunktion betrachtet. Weitern gilt, dass $d = 5$ und $\theta = 0.75$.

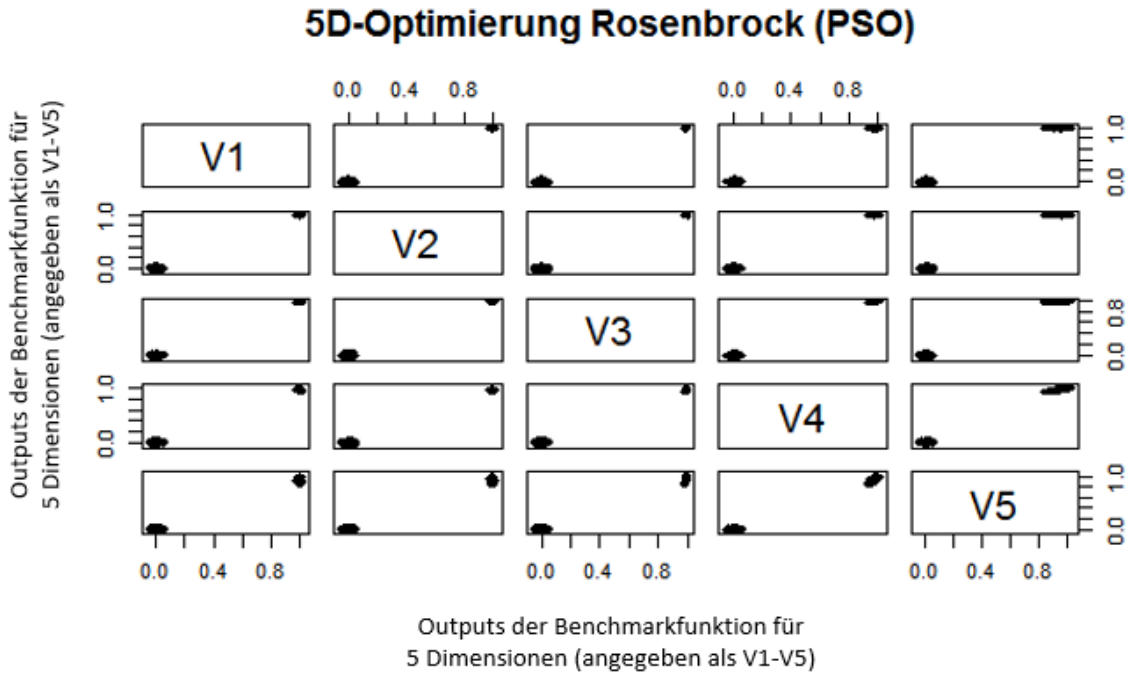

Abb. 32 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rastriginfunktion mittels PSO mit Standardparametern als paarweise Scatterplots

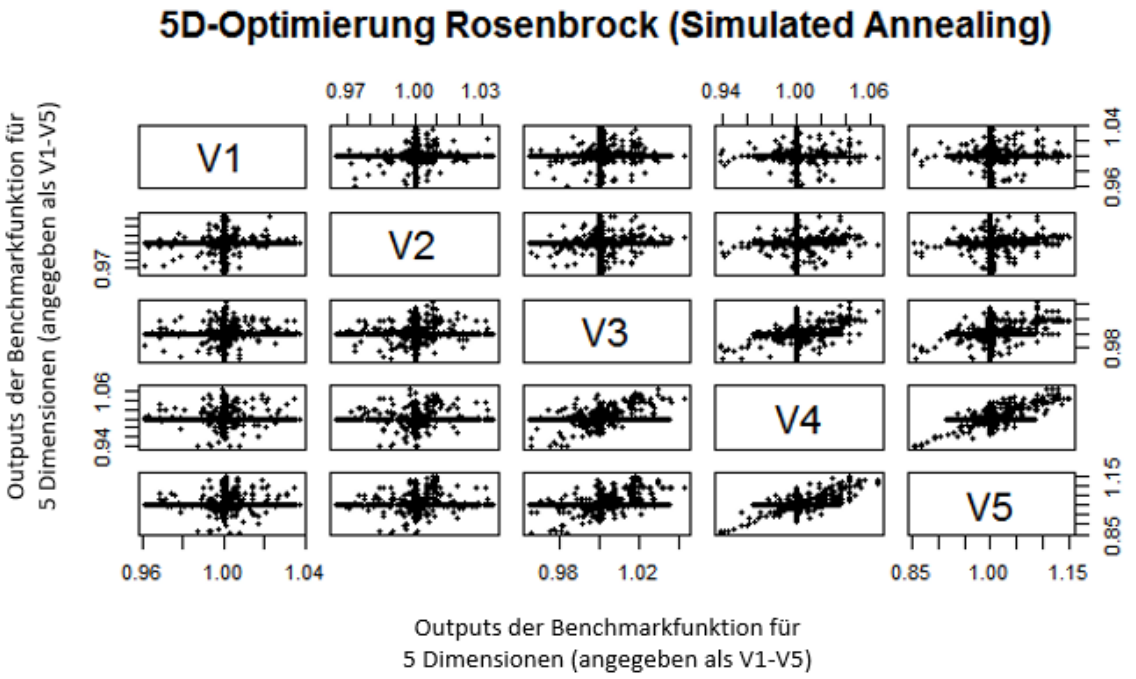

Abb. 33 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rastriginfunktion mittels Simulated Annealing mit Standardparametern als paarweise Scatterplots

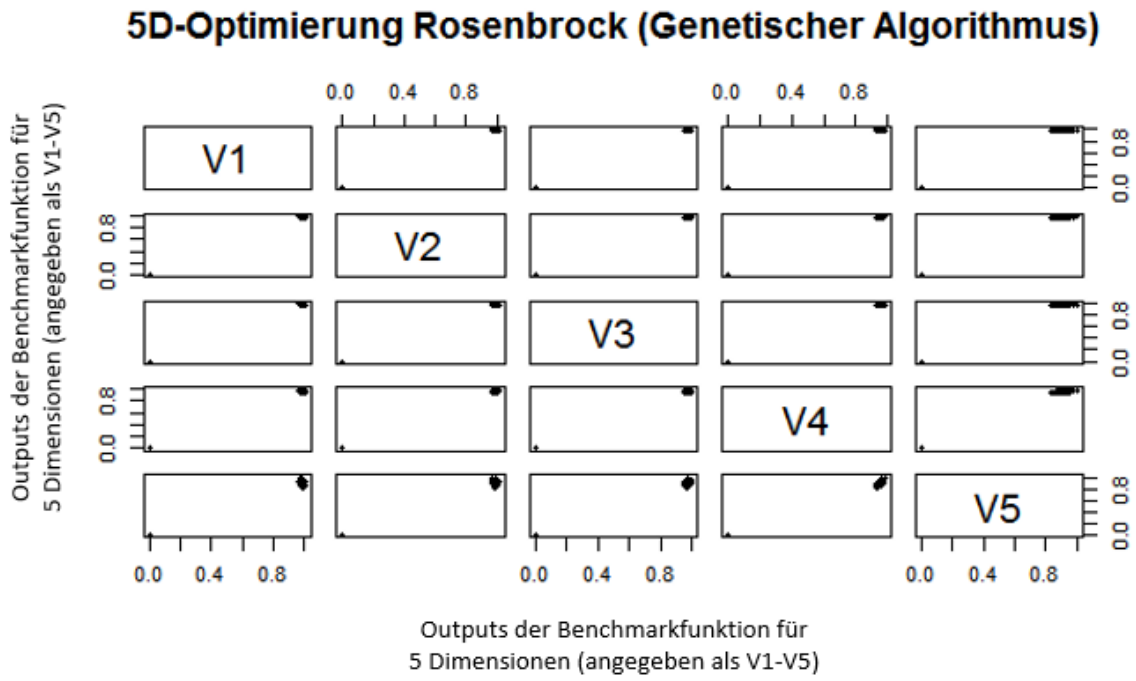

Abb. 34 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rastriginfunktion mittels genetischem Algorithmus mit Standardparametern als paarweise Scatterplots (Iterationen auf n = 2000 erhöht)

Die Betrachtung des gleichen Experiments folgt zwangsläufig zu zwei interessanten Beobachtungen. Zunächst lässt sich festhalten, dass die Anwendung von Simulated Annealing auf die Rosenbrockfunktion nahezu das gleiche Muster erzeugt, das bei der Rastriginfunktion beobachtet werden konnte. Bezogen auf die PSO und den genetischen Algorithmen werden bei jedem paarweisen Scatterplot zwei Cluster sichtbar. Diese sind bei der PSO deutlich ausgeglichener verteilt als beim genetischen Algorithmus. Zusätzlich muss erwähnt werden, dass bei die Anzahl der notwendigen Iterationen beim genetischen Algorithmus deutlich auf 2000 erhöht werden musste, da bei 500 Iterationen noch keine Konvergenz zu einem Wert erfolgt ist, der unter θ liegt. Aus den Mustern folgt trotz der bislang extrem niedrigen Stichprobe bereits, dass sich voraussichtlich keine allgemeingültige Empfehlung für ein bestimmtes Verfahren für die Optimierung aussprechen lässt. Um das Experiment zu vervollständigen, sei zusätzlich die Griewankfunktion als Benchmark betrachtet. Unverändert gilt, dass $d = 5$ und $\theta = 0{,}75$.

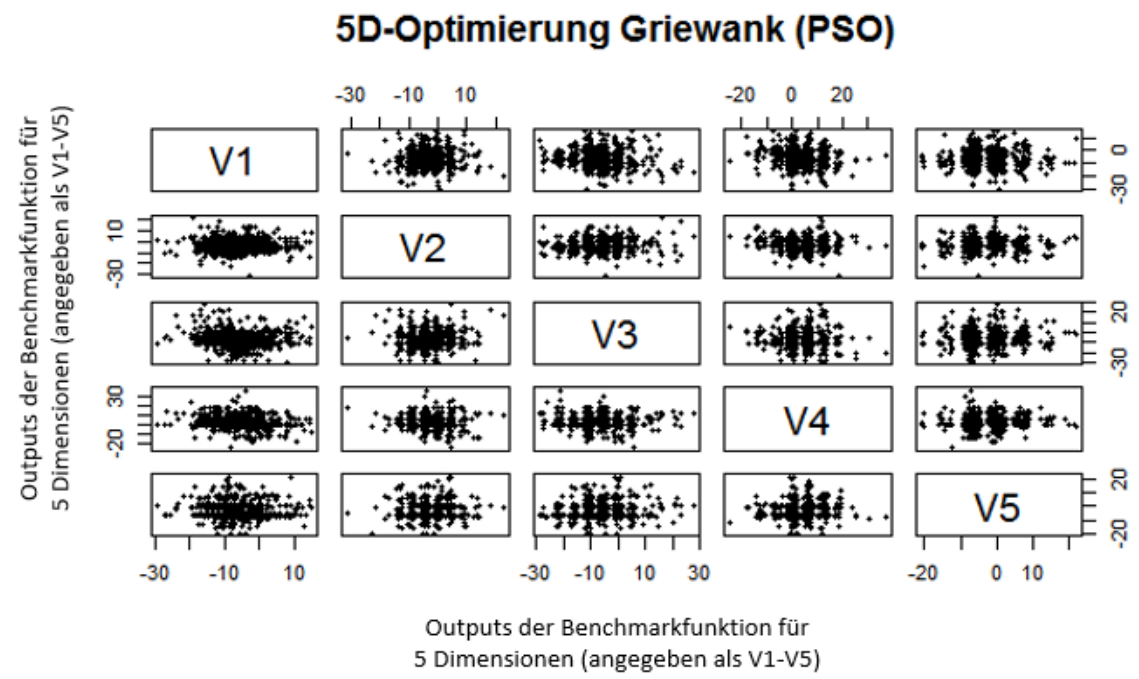

Abb. 35 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Griewankfunktion mittels PSO mit Standardparametern als paarweise Scatterplots

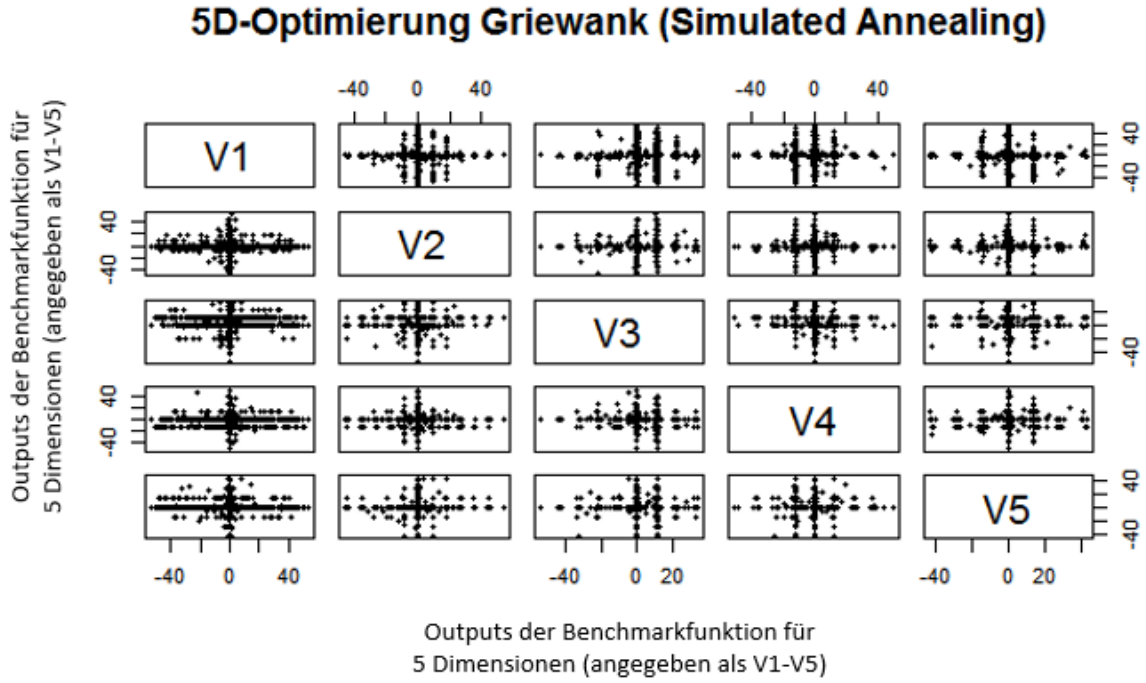

Abb. 36 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Griewankfunktion mittels Simulated Annealing mit Standardparametern als paarweise Scatterplots

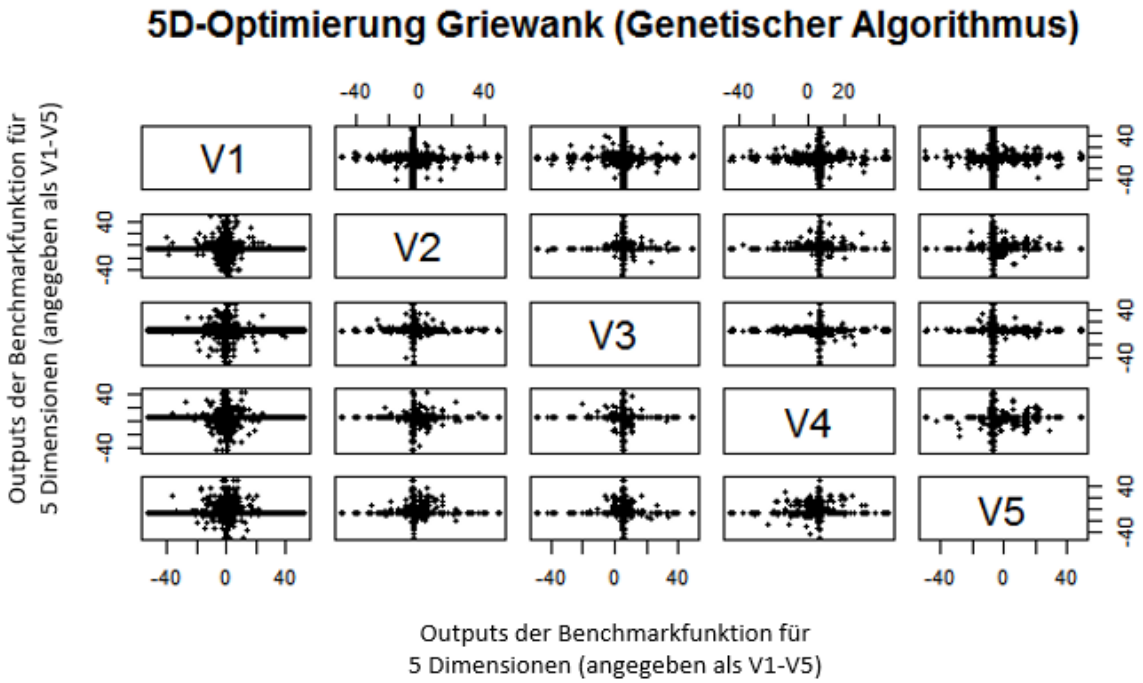

Abb. 37 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Griewankfunktion mittels genetischem Algorithmus mit Standardparametern als paarweise Scatterplots

In dieser Anordnung ist es sehr interessant, dass sich hier der genetische Algorithmus am wenigsten explorativ verhält. Simulated Annealing tendiert weiterhin deutlich zu der Erzeugung eines erkennbaren Gittermusters, in dem die Zwischenräume kaum gefüllt sind. Allerdings könnte es sich in dem konkreten Fall positiv auswirken, dass die Amplitude der Werte verhältnismäßig hoch ist. Die PSO erzeugt auf einem etwas engeren Raum ein dichteres Muster mit relativ hohen Amplituden. Rund um das globale Optimum ist hier die Raumabdeckung klar am besten.

Verbesserung der Vergleichbarkeit zwischen Einzel- und Mehrpunktoptimierung

Basierend auf den bisherigen Ergebnissen, kann man offensichtlich zu dem Schluss kommen, dass ein direkter Vergleich zwischen den beiden Ansätzen nicht unbedingt „fair“ ist, weil die unterschiedlichen Positionen einzelner Individuen im Suchraum möglicherweise exploratives Verhalten begünstigen. Insofern richtet die folgende Versuchsreihe ihr Augenmerk konkret auf die Einzelpunktoptimierung durch Simulated Annealing.

Aus diesem Grund schließt sich ein simples Experiment an. Der Optimierungslauf mit Simulated Annealing wird im Folgenden mit zehn Wiederholungen durchgeführt, wobei sämtliche Ergebnisse gesammelt werden. Insofern liegt der Fokus auf der Fragestellung, ob simultane Durchführungen von Einzelpunktoptimierungen zu einem anderen Muster führen als ein atomarer Optimierunglauf. Die Ergebnisse sind auf Abb. 38, Abb. 39 und Abb. 40.

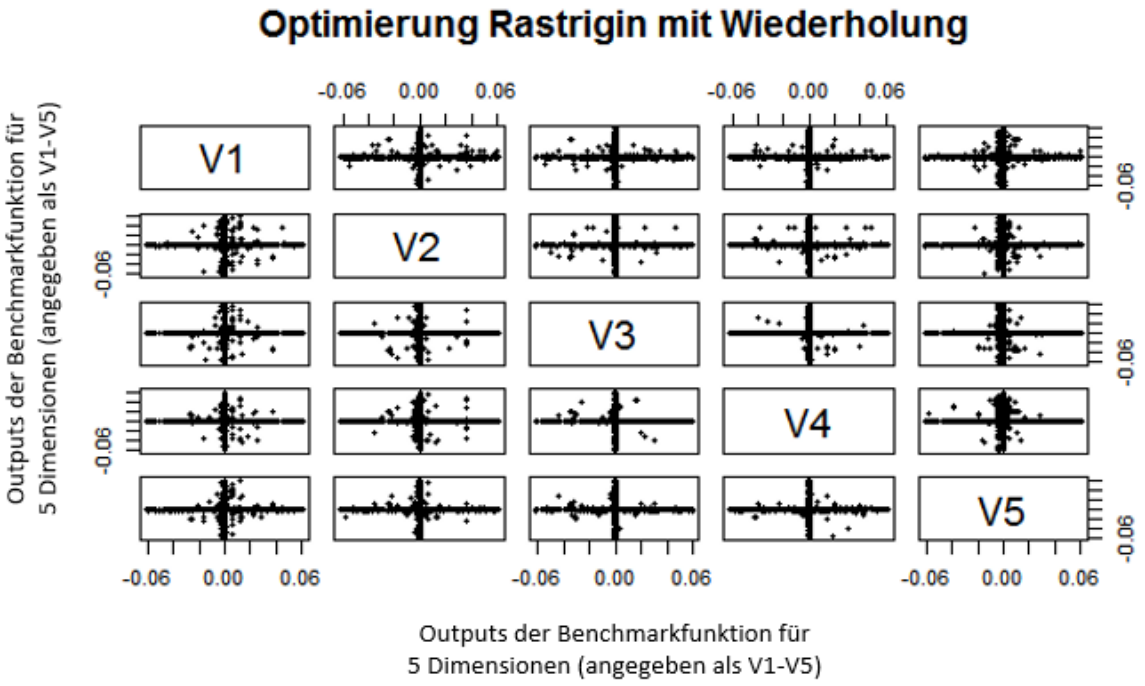

Abb. 38 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rastriginfunktion mittels Simulated Annealing mit zehnfacher Wiederholung als paarweise Scatterplots

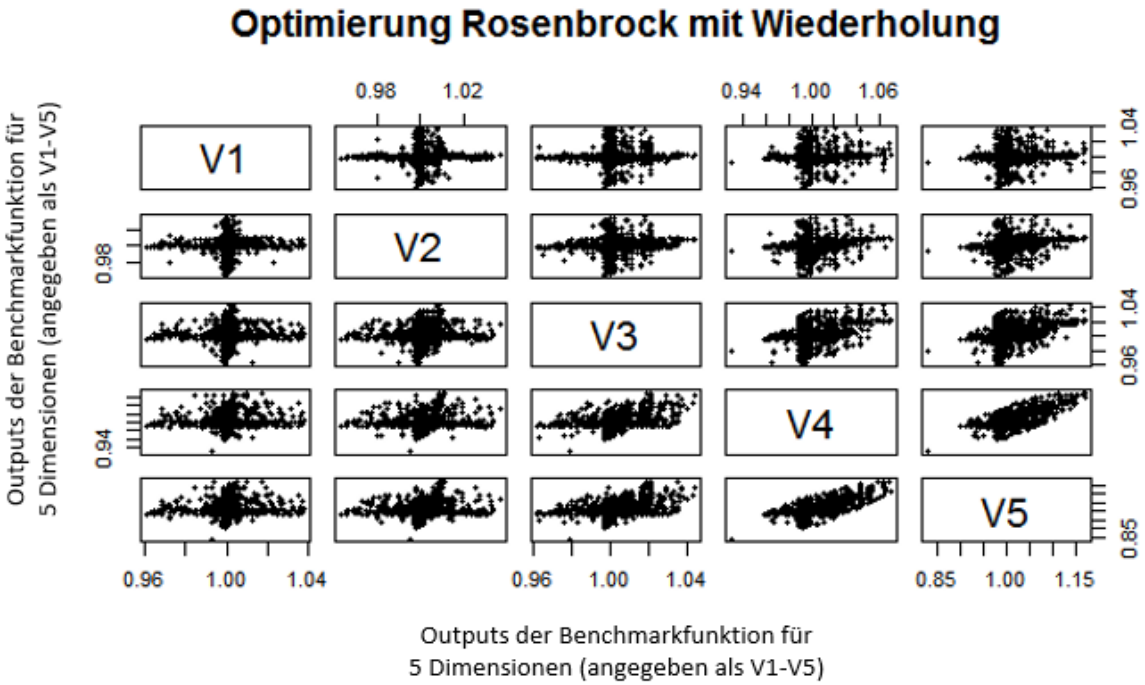

Abb. 39 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Rosenbrockfunktion mittels Simulated Annealing mit zehnfacher Wiederholung als paarweise Scatterplots

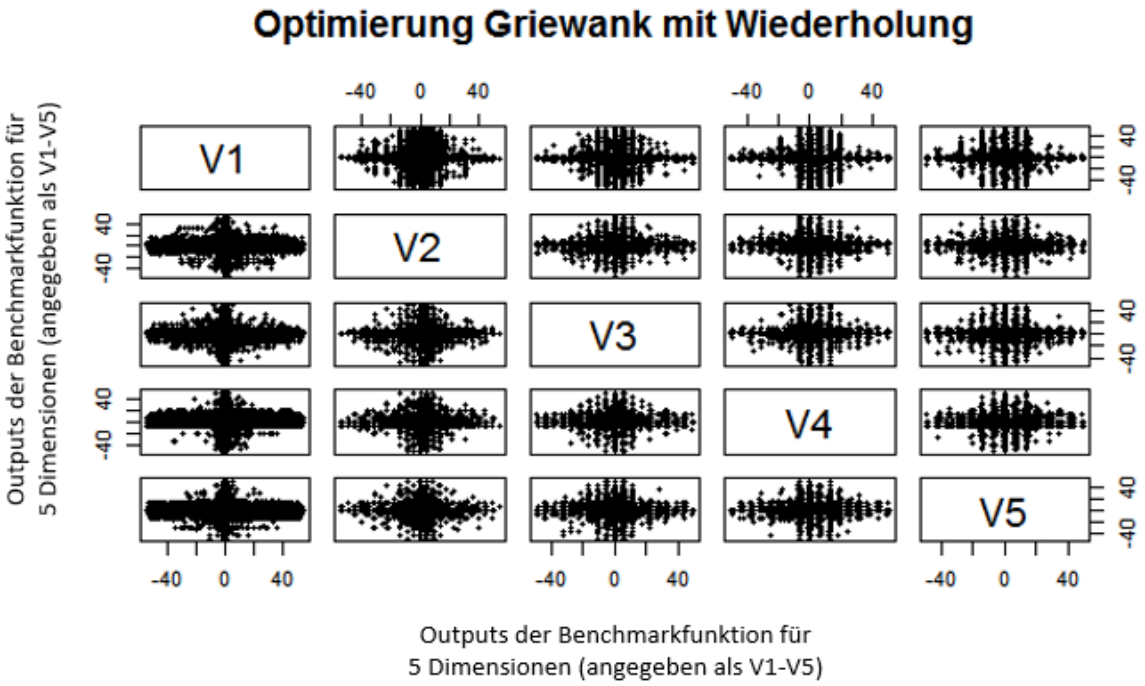

Abb. 40 Darstellung der besten Datenpunkte nach Optimierung der 5-dimensionalen Griewankfunktion mittels Simulated Annealing mit zehnfacher Wiederholung als paarweise Scatterplots

Man kann beobachten, dass die Wiederholungen nicht konsistent zu einer stärkeren Exploration führen. Insbesondere die Betrachtung der Rastriginfunktion zeigt, dass im Extremfall keinerlei Veränderung im Muster erkennbar ist.

Auswertungen der Evaluationsfunktionen

Im Folgenden richtet sich der Blick auf die Auswertungen der Evaluationsfunktionen ausgehend von den gesammelten Daten der jeweiligen Optimierungsläufe. Dazu wird jeder der in Tabelle 10 vorgestellten Datensätze separat betrachtet. Um die Anzahl der auftretenden Freiheitsgrade hierbei minimal zu halten, sei zunächst weiterhin der 5-dimensionale Fall betrachtet.

Wie Tabelle 10 zeigt, handelt es sich beim Boston Housing Datensatz um ein Regressionsproblem. Die Zielgröße trägt hierbei den Namen „*medv*“. Es werden an dieser Stelle drei unterschiedlich komplexe Szenarien betrachtet. Sei $\Theta \in \mathbb{R}$ ein Grenzwert und $r : \mathbb{R}^{13} \to \mathbb{R}$ der Regressor, der den Wert für „*medv*“ vorhersagt. Dann ist die Verifikationsfunktion $g_{Regression}$ definiert als:

$$g(x) = \begin{cases} 0, falls\ r(x) < \Theta \\ 1, andernfalls \end{cases}$$

Hierbei werden $\Theta_{0.25} = 17{,}025, \theta_{median} = 21{,}2$ und $\Theta_{0.75} = 25$ betrachtet, also jeweils die Quantile und der Median, um unterschiedlich „schwierige“ Szenarien zu erzeugen. In der folgenden Betrachtung wird eine Betrachtung über sämtliche Fitnessfunktionen f präsentiert, die stets davon ausgeht, dass $k = 4$, also vier Cluster berechnet worden sind. Von besonderem Interesse ist hierbei in erster Linie, die Annahme zu bestätigen, dass populations- oder schwarmbasierte Algorithmen schnellere Ergebnisse erzeugen können. Die Anzahl der Eingangsdimensionen für die jeweiligen Verfahren wird für diesen Zweck nicht fixiert. Für $d = 2{,}5{,}10{,}20\ \&\ 50$ wurden jeweils 50 Messungen durchgeführt und im Anschluss jeweils eine Messung mit kMeans und zufälligem Subsampling durchgeführt.

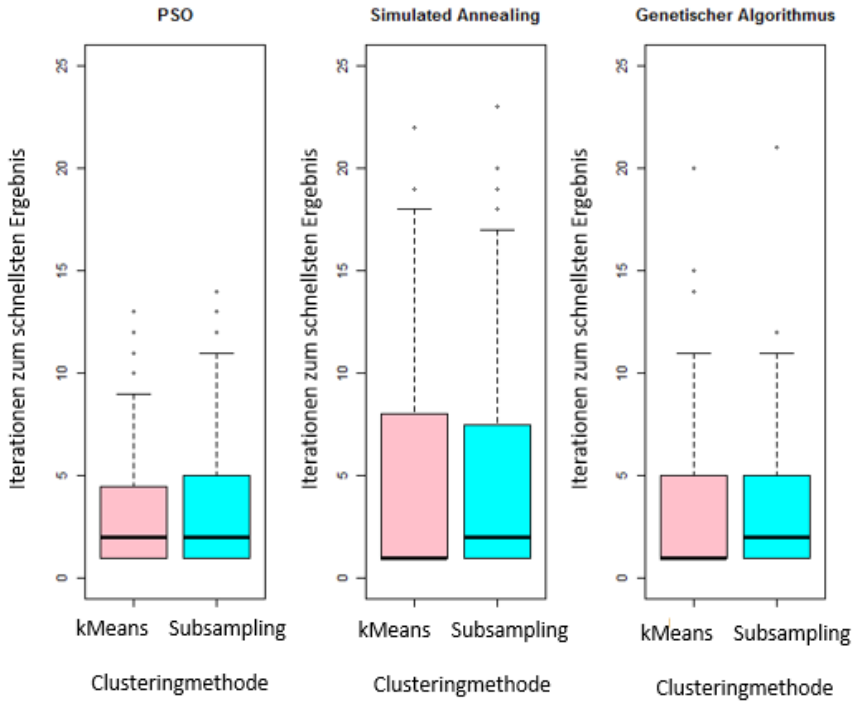

Abb. 41 Vergleich der Optimierungsalgorithmen für den Boston Housing Datensatz unter Annahme von $\Theta_{0.25}$

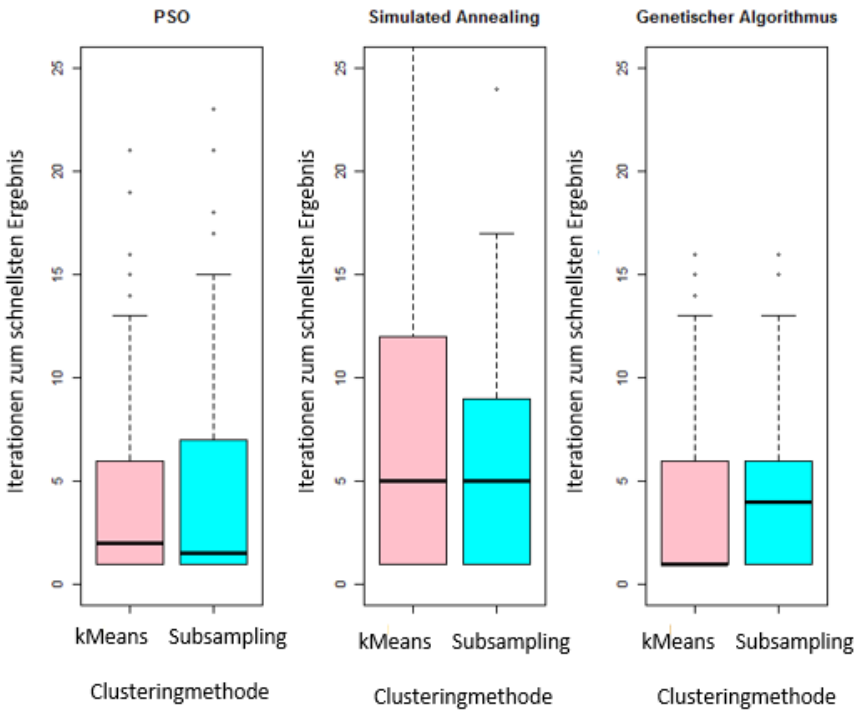

Abb. 42 Vergleich der Optimierungsalgorithmen für den Boston Housing Datensatz unter Annahme von Θ_{median}

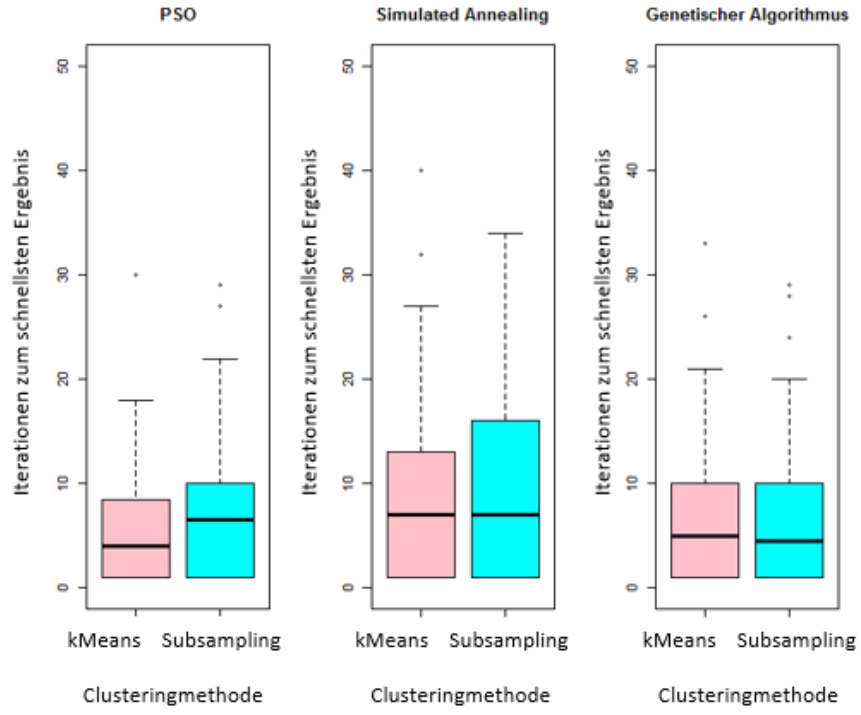

Abb. 43 Vergleich der Optimierungsalgorithmen für den Boston Housing Datensatz unter Annahme von $\Theta_{0.75}$

Die getroffene Annahme wird hier eindeutig bestätigt. Schwarm- und populationsbasierte Verfahren führen in allen drei Szenarien zu deutlich schnelleren

Ergebnissen. Gleichzeitig bestätigt sich die Annahme, dass komplexere binäre Landschaften statistisch gesehen zu mehr Iterationen bis zum Erreichen eines schnellen Ergebnisses führen. Weiterhin zeigt sich, dass insgesamt die explorative Abdeckung durch kMeans zu einem besseren Ergebnis führt als ein zufälliges Subsampling.

Man könnte die Erkenntnisse des obigen Experiments durch die Betrachtung der zusätzlichen genannten Benchmarkdatensätze bestätigen. Auf diesen Fokus wird an der Stelle verzichtet, da im Folgenden andere Fragestellungen im Vordergrund stehen sollen. Aus diesem Grund wird der Fokus zunächst daraufgelegt, festzustellen, welche Rolle die Eingangsdimensionalität spielt. Aus diesem Grund wird die Anzahl der Dimensionen hier als einziger Freiheitsgrad betrachtet. Das Experiment besteht darin, sämtliche Fitnessfunktionen mithilfe der Partikelschwarmoptimierung unter Annahme der Eingangsdimensionen $d = 2{,}5{,}10{,}20\ \&\ 50$ zu optimieren und die besten Ergebnisse zu sammeln. Sämtliche Optimierungen werden hierbei unter Annahme der Standardparameter durchgeführt und 50-fach wiederholt. Die Kollektionen werden dann mit kMeans geclustert, wobei weiterhin gilt: $k = 4$. Das Clustering wurde dabei für jeden Fall ohne Wiederholung durchgeführt. Daraus leitet sich eine Stichprobe von 250 Beobachtungen ab, die jeweils durch sämtliche Benchmarkdatensätze untersucht wurde.

Tabelle 12 Verifikationsfunktionen für die einzelnen Datensätze

Name des Datensatz	**Erstellen der binären Landschaft**
Boston Housing Data	$g(x) = \begin{cases} 0, falls\ r(x) < \Theta_{\text{median}} \\ 1, andernfalls \end{cases}$
Wisconsin Breast Cancer	$g(x) = \begin{cases} 0, falls\ c_{wisconsin}(x) = benign \\ 1, andernfalls \end{cases}$
Glass Identification Database	$g(x) = \begin{cases} 0, falls\ c_{glass}(x) \leq 4 \\ 1, andernfalls \end{cases}$
John Hopkins University Ionosphere Database	$g(x) = \begin{cases} 0, falls\ c_{ionosphere}(x) = good \\ 1, andernfalls \end{cases}$
Letter Image Recognition Data	$g(x) = \begin{cases} 0, falls\ c_{letter}(x) \leq 'm' \\ 1, andernfalls \end{cases}$
Friedman1	$g(x) = \begin{cases} 0, falls\ r(x) < \Theta_{\text{median}} \\ 1, andernfalls \end{cases}$
Peak Benchmark Problem	$g(x) = \begin{cases} 0, falls\ r(x) < \Theta_{\text{median}} \\ 1, andernfalls \end{cases}$
XOR Problem	$g(x) = \begin{cases} 0, falls\ c_{xor}(x) < \frac{2^{d-1}}{2} \\ 1, andernfalls \end{cases}$
Pima Indians Diabetes Database	$g(x) = \begin{cases} 0, falls\ c_{diabetes}(x) = neg \\ 1, andernfalls \end{cases}$

Tabelle 12 zeigt, wie die einzelnen Verifikationsfunktionen für verschiedene Datensätze gewählt wurden. Bei binären Klassifikatoren wurde eine der beiden Klassen als 0 und die andere als 1 angenommen. Bei Mehrklassenklassifikationen wurde je eine Hälfte der Klasse 0 und die andere Hälfte der Klasse 1 zugeordnet. Regressionsprobleme wurden, wie bereits bei der separaten Betrachtung des Boston Housing Datensatzes, bezüglich des Medians des Zielparameters zugeordnet. Nun stellt sich also die Frage, ob die Dimensionalität des Problems eine entscheidende Rolle für das Verfahren spielt.

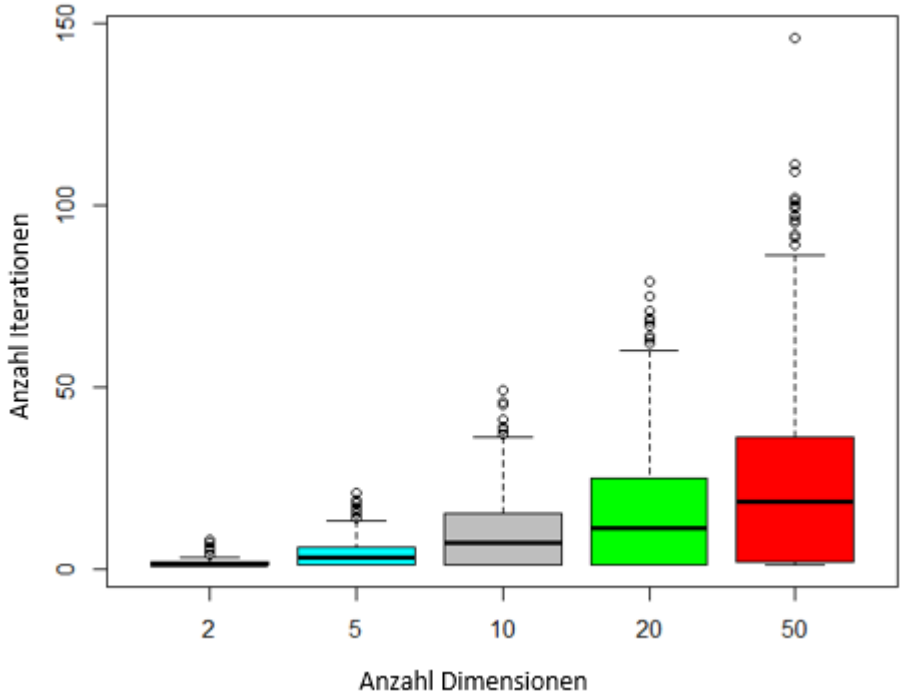

Abb. 44 Betrachtung verschiedener Dimensionen

Abb. 44 zeigt das Ergebnis dieser Betrachtung. Im Mittel steigt die Anzahl der benötigten Iterationen leicht an. Nichtsdestotrotz bleibt die Anzahl der benötigten Iterationen absolut überschaubar. Was zu erkennen ist, ist, dass eine höhere Dimensionalität mit einem Anstieg der Abweichung einhergeht. Niedrigdimensionale Räume weisen relativ wenige Ausreißer auf. Es ist offensichtlich, dass in hochdimensionalen Räumen die Abweichung der betrachteten Datenpunkte innerhalb der Cluster sehr groß ist. Das kann dazu führen, dass in diesen Clustern zunächst Teilbereiche untersucht werden, deren ähnliche Fitnesswerte mit ihrer räumlichen Nähe zusammenhängen. Das führt dann dazu, dass „erlaubte“ Konfigurationen erst zu einem sehr späten Zeitpunkt untersucht werden. Insofern verhärtet sich an der Stelle der Verdacht, dass es unter bestimmten Umständen sinnvoll ist, die Anzahl der Cluster nicht an die verfügbaren Rechnerknoten zu koppeln.

Kurzfazit der Auswertungen

Die beschriebenen theoretischen Untersuchungen bilden erste Erkenntnisse ab, die sich bezüglich der grundsätzlichen Betrachtung ableiten lassen. Im Wesentlichen wurde hier der Grundstein gelegt, um eine tiefere Auswertung vornehmen zu können. Bislang wurden signifikante Aspekte, wie etwa die Veränderung von Optimierungskonstanten zur Verbesserung der Exploration, komplett außer Acht gelassen. So wäre die Erwartung, dass eine stärkere Exploration zu ausdifferenzierteren Clustern führt. Auch die Ausdifferenzierung über verschiedene Fitnessfunktionen kann offensichtlich näher beleuchtet werden. Nichtsdestotrotz konnte über die theoretischen Auswertungen gezeigt werden, dass für beliebige Anwendungsfälle, die auf die genannte Art und Weise modelliert werden können, Zeitersparnisse möglich sind. Es deutet sich allerdings auch an, dass eine sortierte Betrachtung der einzelnen Cluster nicht immer zielführend ist, insbesondere dann, wenn eine hochdimensionaler Raum mit einer relativ kleinen Anzahl an Clustern betrachtet wird.

5. Intelligente Arbeitsvorbereitung durch virtuelle Werkzeugmaschinen

In diesem Kapitel soll es darum gehen, zu motivieren, wie sich der beschriebene Ansatz in eine übergeordnete Plattform einbettet, die im Rahmen des Forschungsprojekts „Intelligente Arbeitsvorbereitung auf Basis virtueller Werkzeugmaschinen“ entstanden ist.

5.1. Virtuelle Werkzeugmaschine

Grundlage für die Realisierung des Forschungsprojekts ist die Virtualisierungssoftware der Firma DMG Mori Seiki, die DMG Virtual Machine. Hierbei handelt es sich um eine Software, die sämtliche physische Dreh- und Fräsmaschinen des Produktportfolios ganzheitlich abbildet. Um eine realistische Abbildung zu gewährleisten, die insbesondere Kollisionen zuverlässig erkennt, werden Komponenten zum Teil originalgetreu übernommen. Das gilt insbesondere für die Steuerungskomponenten, also die SPS (speicherprogrammierbare Steuerung) und die CNC. Da die CNC wie bereits beschrieben auf einem angepassten Industrie-PC läuft und die SPS im Grunde genommen als leistungsschwacher Rechner betrachtet werden kann, lassen sich beide Komponenten problemlos virtualisieren. Einhergehend mit der CNC wird die HMI (Human-Machine-Interface) so dargestellt, wie man es von einer physischen Dreh- oder Fräsmaschine kennt. Analog zur realen Maschine in der Fabrik werden Dateien, die NC-Programme enthalten, an die CNC übertragen und können über ein „Control Panel“ gestartet werden. Die Konsequenz aus dieser Architektur ist, dass es für die Steuerungen keinen Unterschied macht, ob sie mit einer physischen oder einer virtuellen Werkzeugmaschine kommuniziert. Konkret kann das beispielsweise bedeuten, dass eine offene Maschinentür innerhalb der Simulationsumgebung dazu führt, dass die SPS die Ausführung von NC-Befehlen blockiert. Die Übersetzung der Kinematik, also die eigentlich NC-Simulation, wird durch ein proprietäres Modul der Firma Machineworks [64] realisiert. Diese bietet ein recht breites Spektrum an Lösungen für die NC-Simulation an. In diesem konkreten Fall kommt das Framework *Vericut* zum Einsatz.

Neben den technischen Komponenten existiert selbstverständlich eine grafische Oberfläche, die eine Simulation visualisieren kann. Hierfür ist eine geometrische Repräsentation für Maschinen, Spannmittel, Rohteile und Werkzeuge notwendig. Für deren Umsetzung wird ein proprietäres Datenformat eingesetzt, das eine Baumstruktur enthält. Dieses wird als VMDE (Virtual Machine Data Exchange) bezeichnet. An dieser Stelle werden nicht sämtliche Aspekte dieses Formats beleuchtet. Es ist allerdings für die spätere technische Umsetzung wichtig, einige Begriffe und die grundsätzliche Struktur zu erläutern.

```
<Workpiece>
      <Shape>
         // Geometrieinformationen (z.B. Quader, Zylinder,
         FacedSet etc.)
      </Shape>
      <ZeroPoint location = "30 30 30">
      <PositionFrame rotationMatrix = "1 0 0; 0 1 0; 0 0 1"
      translation = "0 0 0">
</Workpiece>
```

Abb. 45 Grobe Darstellung eines Werkstücks als VMDE

Abb. 45 zeigt die grobe Darstellung einer Werkstückgeometrie im VMDE-Format. Die eigentliche Geometrieinformation sei in diesem Kontext nicht weiter von Interesse. Vereinfacht kann man festhalten, dass der Geometrieknoten analog zu einer *Scene* im X3D-Format modellieren kann. Für nähere Informationen sei auf [65] verwiesen. Von Interesse sind allerdings die Knoten *ZeroPoint* und *PositionFrame*. Mit der Angabe *„location“* im Knoten *ZeroPoint* wird der Referenzpunkt für die NC-Bearbeitung angegeben. Es handelt sich also um den Werkstücknullpunkt, auf den sich die Geometrieangaben innerhalb eines NC-Programms beziehen, falls dieses nicht mit Absolutmaßen arbeitet. Der sogenannte Position Frame beschreibt die Positionierung eines Werkstücks im Bauraum. Definiert ist dieser als Tupel $\Psi \coloneqq (o \in \mathbb{R}^{3\times3}, \tau \in \mathbb{R}^3)$ aus einer Rotationsmatrix und einem Translationsvektor. Um die genaue Funktionsweise zu verstehen, ist es hilfreich, zunächst die Beschreibung eines Spannmittels in VMDE zu betrachten.

```
<WorkpieceClamp>
     <Shape>
        // Geometriebeschreibung analog zu Werkstück
     </Shape>
     <PositionFrame rotationMatrix = "1 0 0; 0 1 0; 0 0 1"
     translation = "0 0 0">
     <DockingFrame rotationMatrix = "1 0 0; 0 1 0; 0 0 1"
     translation = "0 0 0">
</WorkpieceClamp>
```

Abb. 46 Grobe Darstellung eines Spannmittels in VMDE

Wie in Abb. 46 zu sehen ist, wird die Beschreibung des Spannmittels im Gegensatz zum Werkstück um den sogenannten Docking Frame ergänzt. Bezüglich seiner formalen Beschreibung entspricht dieser exakt dem Position Frame, allerdings kommt ihm eine andere Aufgabe zu. Das Werkstück wird auf dem Spannmittel positioniert, so dass Position Frame des Werkstücks und DockingFrame des Spannmittels exakt aufeinanderliegen. Der Position Frame des Spannmittels wiederum entscheidet über die Positionierung des Spannmittels im Bauraum, da der Maschinentisch in seiner inhärenten Geometrie ebenfalls einen Docking Frame aufweist.

5.2. Zielsetzung Simulation as a Service

Die Kernfragestellung des Forschungsprojekts basiert auf der Idee, die Software zur Materialabtragssimulation um die Möglichkeit zu erweitern, als Service innerhalb einer Cloudumgebung zu fungieren. Hieraus ergeben sich offensichtlich verschiedene Vorteile sowohl für den Betreiber der Plattform als auch für den Anwender. Zunächst liegt bei der virtuellen Werkzeugmaschine eine Situation vor, die in ähnlicher Form häufig auftritt, wenn viele externe Komponenten unter dem Dach einer Software zusammengefasst sind: Sie ist relativ unflexibel in Bezug auf die Plattform, auf der sie betrieben werden kann. So ist die im Projekt verwendete Version der virtuellen Werkzeugmaschine lediglich lauffähig, wenn der Installationsrechner mit dem Betriebssystem *Windows 7* arbeitet. Wenn also die IT-Infrastruktur eines Unternehmens auf einem älteren Betriebssystem von Microsoft oder gar Linuxderivaten beruht, ist die erfolgreiche Installation nicht möglich. In diesem speziellen Fall besteht zusätzlich das Problem, dass die integrierte virtuelle Siemenssteuerung im Hintergrund eine virtuelle Maschine auf VMware-Basis betreibt. Sollte sich ein Anwender also entscheiden, eine *Windows 7* Installation für die erfolgreiche Einrichtung der virtuellen Werkzeugmaschine auf einem virtualisierten Rechner zu betreiben, ist das lediglich durch Nutzung einer VMware-Instanz zu erreichen, allerdings nicht über eine VirtualBox-Instanz. Das zeigt exemplarisch, dass es bei hochspezialisierter Software sinnvoll sein kann, Instanzen zu zentralisieren, so dass jeder Nutzer als Service auf verschiedene Instanzen zugreifen kann, ohne dafür Änderungen an der eigenen Infrastruktur vornehmen zu müssen. Was für die Installation gilt, gilt gleichermaßen für etwaige Updates der Software. Versionssprünge oder das Einpflegen neuer Maschinengeometrien müssen innerhalb einer serviceorientierten Architektur nicht bei jedem einzelnen Mandanten vorgenommen werden, sondern können ebenfalls zentral erfolgen, wodurch beim Anwender keinerlei Aufwand entsteht.

Vorteile durch die Zentralisierung der Simulationssoftware innerhalb einer Dienstleistungsplattform entstehen natürlich nicht nur kundenseitig, sondern auch auf Seiten des Anbieters. Vom technischen Standpunkt betrachtet bietet die Transformation von lokalen Installationen einer Software hin zu Software als cloudbasierte Dienstleistung die Möglichkeit, den Support an einem Ort zu bündeln, da durch den globalen Zugriff jegliche Eingriffe per Fernwartung vorgenommen werden können. Viel interessanter als dieser Aspekt ist jedoch die sich bietende Möglichkeit, das eigene Portfolio um serviceorientierte Geschäftsmodelle zu erweitern und damit einhergehend neue Märkte zu erschließen. Insbesondere Diskussionen mit Domänenexperten zu Beginn des Projekts haben gezeigt, dass ingenieursgetriebene Weiterentwicklungen der Werkzeugmaschinen selbst kaum noch zu Alleinstellungsmerkmalen führen, die im globalen Markt zu Kaufentscheidungen führen. Im Vordergrund stand in den Ausführungen der Domänenexperten, sowohl des Projektverantwortlichen DMG Mori Seiki, als auch den Pilotkunden, vor allem der indische Markt, der in den letzten Jahren

bezüglich seiner Werkzeugmaschinen ein extrem hohes Wachstum verzeichnen konnte. Das unterstreicht auch eine Recherche in den einschlägigen Onlinemedien [66] [67] [68]. Da die indischen und auch andere asiatische Unternehmen deutlich günstiger produzieren können und laut Domänenexperten der Qualitätsunterschied nach und nach abnimmt, ist die genannte Erschließung digitaler Geschäftsmodelle als Ergänzung zum reinen Verkauf von Anlagen unerlässlich. Simulationen als Service in der Cloud anzubieten und zeitgleich als Komponente in verschiedene Optimierungsschleifen einzubinden ist also als Teil der Strategie zu verstehen, Erfahrungswerte mit digitalen Geschäftsmodellen zu sammeln.

Da die oben genannten Vorteile eigentlich stark wirtschafts- und wachstumsgetrieben sind, stellt sich die Frage, warum das Thema nicht als Teil der strategischen Produktentwicklung, sondern zunächst als Innovationsprojekt innerhalb eines geförderten Forschungsclusters platziert worden ist. Das hängt insbesondere damit zusammen, dass die Implementierung einer solchen Dienstleistungsplattform nicht nur Potentiale, sondern auch Risiken birgt, die zum Teil bis heute und insbesondere zum Zeitpunkt der Antragsstellung eine unmittelbare Inbetriebnahme verhindern. Neben der technischen Herausforderung, eine Dienstleistungsplattform effektiv und gewinnbringend zu entwickeln, besteht die größte Hürde darin, die Akzeptanz von cloudbasierten Systemen in produzierenden Unternehmen zu stärken. Abhängig vom Anwendungsfall fällt diese Hürde unterschiedlich hoch aus. Offensichtlich besteht die Problematik in diesem konkreten Fall darin, dass die Nutzung von zentralen Ressourcen zur Materialabtragsimulation erfordert, dass sensible Informationen wie NC-Programme, genutzte Maschinen oder genutzte Werkzeuge ebenfalls zentral hinterlegt werden. Bei Sicherheitsmängeln in der Dienstleistungsplattform wäre demnach die große Gefahr von Produktpiraterie beziehungsweise Industriespionage gegeben. Da die umzusetzende Plattform nicht nur die reine Simulation, sondern darüber hinaus die Optimierung der Arbeitsplanung als Service anbieten soll, betrifft dieser Aspekte sogar weitere sensible Informationen wie Maschinenausfälle oder Verfügbarkeit von bestimmten Materialien.

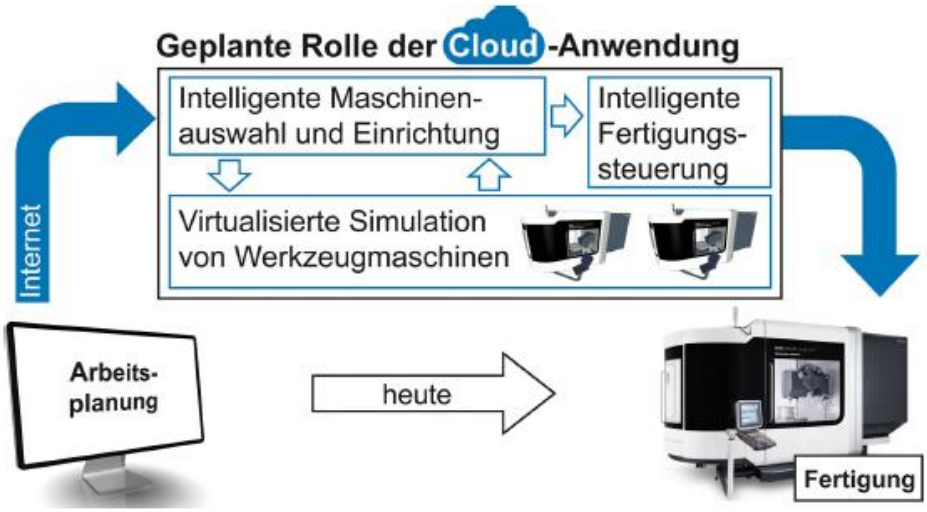

Abb. 47 Schematische Darstellung der Dienstleistungsplattform [69]

5.3. Globale Fragestellung – Optimierungsebenen

Wie bereits im vorangegangen Kapitel angedeutet, ergibt sich aus der Transformation einer statischen Software, die eine lokale Installation erfordert, hin zu einer serviceorientierten Architektur die Möglichkeit, Informationen, die während der Durchführung von Simulationen anfallen, für die Entwicklung verschiedener Optimierungen im Produktionsprozess zu verwerten. Demnach ist das Ziel des Gesamtprojekts, unterschiedliche Optimierungsebenen unter dem Dach einer zentralen Dienstleistungsplattform zu vereinigen, wobei die jeweiligen Ebenen direkt oder indirekt mit Simulationsergebnissen in Verbindung stehen. Die Grundidee wird in Abb. 47 skizziert. Oberflächlich formuliert soll eine Plattform entstehen, in der relevante Informationen für die Arbeitsplanung in eine Plattform einfließen, die zunächst auf Basis von Materialabtragssimulationen ermittelt, welche Teile, die in der Fertigung anfallen, auf welcher Maschine kollisionsfrei hergestellt werden können und wie die die Einrichtung einer Maschine, also beispielsweise die Positionierung eines Rohteils im Bauraum inklusive Programmnullpunkt, nach bestimmten Kriterien optimiert werden kann. Demnach soll die Dienstleistungsplattform zwei Module enthalten, die unmittelbar mit der Virtualisierung der physischen Werkzeugmaschinen kommunizieren. Beide Module sollen ihre Ergebnisse einem weiteren Modul, der intelligenten Fertigungssteuerung übergeben.

Im Folgenden werden die einzelnen Optimierungsebenen kurz skizziert:

5.3.1. Intelligente Maschinenauswahl

Eine zentrale Aufgabe im Zuge des Arbeitsplanungsprozesses stellt die Auswahl einer geeigneten Werkzeugmaschine für die Bearbeitung eines bestimmten Werkstücks dar. Dabei greift der Arbeitsplaner im Wesentlichen auf seine persönliche Erfahrung zurück, um zu entscheiden, welche Maschine sich für die Durchführung eines Auftrags eignet. Falls der Maschinenpark in der Fertigung eine Vielzahl an Anlagen aufweist, die sich gleichzeitig durch eine starke Heterogenität auszeichnen, ist allerdings kaum davon auszugehen, dass ein Arbeitsplaner sämtliche Möglichkeiten durchdenkt, die potentiell für die Fertigung in Betracht gezogen werden können. Man kann sich leicht vorstellen, dass diese Tatsache dazu führt, dass versteckte Potentiale für eine effiziente Arbeitsplanung nicht ausgeschöpft werden. So ist es beispielsweise denkbar, dass eine Veränderung der Aufspannsituation eines Rohteils – etwa durch eine Rotation um 90° um eine beliebige Achse – eine Fertigung auf einer Anlage ermöglicht, die einen kleineren Bauraum aufweist als diejenige, die ursprünglich für die Fertigung vorgesehen gewesen ist. Dadurch können Ressourcen für Anlagen mit einem relativ großen Bauraum zur Verfügung gestellt werden, was innerhalb der Arbeitsplanung zu einer höheren Flexibilität führen kann.

5.3.2. Intelligente Maschineneinrichtung

Die Einrichtung einer Werkzeugmaschine ist ebenfalls ein zentraler Prozessschritt in der Arbeitsvorbereitung. Hierbei geht es im Wesentlichen darum, die Maschine hinsichtlich ihres Arbeitsauftrags korrekt zu bestücken. Das umfasst die Platzierung der benötigten Werkzeuge im Werkzeugmagazin, das Einspannen des Rohteils auf dem Spannmittel, das Übertragen des NC-Programms auf die Steuerung und schließlich das Starten der Fertigung. Darüber hinaus spielen etwaige Umrüstungen eine Rolle, falls ein Rohteil nicht durch eine atomare Maschineneinrichtung gefertigt werden kann. Es stellt sich nun die Frage, welche Optimierungspotentiale hierbei ausgeschöpft werden können und welche Freiheitsgrade dafür zur Verfügung stehen. Grundsätzlich ergeben sich Potentiale an zwei Stellschrauben.

Zunächst lässt sich ein Werkstück innerhalb des Bauraums unterschiedlich positionieren. Durch eine Translation des Werkstücks können Potentiale zur Minderung von Verfahrwegen bei Werkzeugwechseln entstehen. Eine Rotation des Werkstücks kann einerseits energetische Vorteile haben und andererseits die Volumina von benötigten Bauräumen minimieren. Diese Aspekte werden zu einem späteren Zeitpunkt vertieft.

Neben einer veränderten Positionierung des Werkstücks lässt sich grundsätzlich auch die Veränderung der Werkzeugreihenfolge innerhalb des Magazins verändern. Auf diese Weise kann die kumulierte Wegstrecke des Magazins über die Zeit minimieren, was in

geringem Maße energetisch interessant sein kann und gleichzeitig eine Verzögerung des Verschleißes des Magazins mit sich führen kann.

Bezüglich der Umsetzung dieser beiden Optimierungspotentiale innerhalb der Maschineneinrichtung zeigt sich ein signifikanter Unterschied. Die Veränderung der Position eines Rohteils im Raum impliziert das Risiko, Kollisionen während der Verarbeitung zu erzeugen. Insofern ist für diese Optimierung eine Prüfung mittels einer NC-Simulation zwingend erforderlich. Eine Veränderung der Werkzeugreihenfolge kann ihrerseits keine Kollisionen induzieren, wodurch diese Form der Optimierung komplett ohne die Prüfung durch NC-Simulation berechnet werden kann. Bezogen auf den Gesamtkontext des Projekts weist die Optimierung der Werkstückeinrichtung stärkere Implikationen für die gesamte Arbeitsplanung auf. Durch die potentielle Minimierung von Fertigungszeiten können auch hier Potentiale freigesetzt werden, die für die Arbeitsplanung von Interesse sein können. Die Bestückung des Werkzeugmagazins ist wiederum nicht nur autark von der Werkstückeinrichtung zu betrachten, sondern birgt gleichzeitig kaum Potentiale für die Arbeitsplanung. Trotz einer möglichen Minimierung der Wegstrecke des Werkzeugmagazins wird aus einem einfachen Grund in nahezu allen Fällen keine Zeit während der Durchführung der Fertigung gespart. Das hängt damit zusammen, dass der eigentliche Dreh- oder Fräsvorgang und die Bewegung des Magazins parallel stattfinden. Liegt also ein Werkzeugwechsel an, steht das ausgewählte Werkzeug schon am Werkzeugwechselpunkt zur Verfügung. Eine Optimierung der Gesamtlaufzeit des NC-Programms könnte nur unter Bedingung entstehen, wenn es am Werkzeugwechselpunkt zu einer Wartezeit kommt. Da dieser Fall in den seltensten Fällen eintritt, ist die Veränderung der Werkzeugreihenfolge aus oben genannten Gründen zwar ein interessanter Optimierungsansatz, der allerdings komplett unabhängig vom Gesamtkontext betrachtet werden muss.

5.3.3. Intelligente Arbeitsplanung

Bei der Betrachtung der vorangegangen Optimierungsebenen war häufig die Rede davon, dass sich durch deren Umsetzung Potentiale für eine intelligente Arbeitsplanung ergeben können. Auch dieser Prozess soll natürlich innerhalb der Dienstleistungsplattform automatisiert stattfinden. Bei dieser Optimierung handelt es sich grundsätzlich um ein Schedulingproblem, also der Ermittlung eines Arbeitsplans, der auf Grundlage eines mathematischen Modells die höchsten Einnahmen generiert. Hierbei liegt offensichtlich eine Vielzahl an Freiheitsgraden vor. Betrachtet man den Aspekt der optimalen Arbeitsplanung zunächst für sich, ist die Vielseitigkeit der Freiheitsgrade bereits schnell erkennbar. Der Optimierungsalgorithmus muss die Anzahl der verfügbaren Anlagen betrachten, wobei gleichzeitig die komplette Schrittkette einer Produktion vom Rohteil bis zur Auslieferung betrachtet werden muss. Offensichtlich ist es nicht sinnvoll, die Produktion der Werkstücke bezogen auf die Werkzeugmaschinen

insgesamt zu beschleunigen, falls dadurch ein Stau bei einem späteren Produktionsschritt, etwa der Lackierung oder Verpackung, entsteht. Für die Ermittlung der Kosten muss eine gewisse Priorisierung ermittelt werden, die beispielsweise von Strafkosten bei Nichtauslieferung abhängt. Außerdem muss die Möglichkeit in Betracht gezogen werden, dass ein Teil extern gefertigt werden kann. Die Kosten für eine derartige Auslagerung der Produktion können offensichtlich sehr volatil sein. Neben den Anlagen müssen außerdem personelle Ressourcen berücksichtigt werden. Hierbei macht es beispielsweise einen Unterschied, ob die Produktion im Zwei- oder Dreischichtbetrieb stattfindet. Diverse Seitenaspekte wie plötzliche Maschinenausfälle, Eilaufträge, Urlaubszeiten etc. machen die Optimierung zu einer extrem komplexen Fragestellung.

Neben diesen intrinsischen Aspekten der intelligenten Arbeitsplanung ist in diesem Kontext die Notwendigkeit der Flexibilität des Optimierungsansatzes von hoher Bedeutung. Wie in Abb. 47 zu sehen ist und auch bereits im Vorfeld angedeutet wurde, findet eine ständige Kommunikation zwischen der intelligenten Arbeitsplanung und den anderen Optimierungsebenen statt. So wird dem System jedes ermittelte Potential unmittelbar mitgeteilt. Falls beispielsweise das Modul zur intelligenten Maschineneinrichtung mitteilt, dass ein Auftrag mit einer hohen Stückzahl - bei anderer Einrichtung - um zehn Sekunden beschleunigt werden kann, muss die intelligente Arbeitsplanung sofort darauf reagieren können. Selbiges gilt, falls ermittelt wurde, dass ein bestimmtes Rohteil unter veränderten Bedingungen auf einer alternativen Anlage gefertigt werden kann.

5.3.4. Virtualisierte Simulationen von Werkzeugmaschinen

Der letzte Aspekt, der auf Abb. 47 zu sehen ist, ist die Durchführung virtualisierter Simulationen von Werkzeugmaschinen. Im Gegensatz zu den vorangegangen Modulen scheint es sich hierbei zunächst um eine Ebene im System zu handeln, die keinen Effekt auf den Endanwender hat beziehungsweise wenig Optimierungspotential birgt. Das hat den Hintergrund, dass die simulationsbasierte Kollisionsprüfung komplett im Hintergrund durchgeführt wird, ohne dass ein Anwender der Plattform Transparenz diesbezüglich erhält. Außerdem wird der Nutzen, der sich aus der Optimierung dieses Moduls für den Anwender ergibt, nicht direkt sichtbar. Nichtsdestotrotz ergeben sich hier Potentiale, deren Vorteile sich je nach Installationsszenario unterscheiden können. In erster Linie geht es hierbei um die Fragestellung, wie die Simulationsressourcen am sinnvollsten über verschiedene Mandanten, die die Plattform bedienen, verteilt werden können. Auch hierbei spielen verschiedene Faktoren eine Rolle. Eine zentrale Fragestellung ist, wie einzelne Simulationsaufträge in der Plattform priorisiert werden müssen, wobei die verschiedenen Optimierungsebenen und die Gleichbehandlung verschiedener Nutzer in Betracht gezogen werden müssen.

Aufgrund der Komplexität einer einzelnen Simulation ist darüber hinaus die Fragestellung interessant, ob die NC-Simulation für ein Rohteil parallelisiert werden kann und welche Skalierung für diesen Fall zu erwarten ist. Diese Fragestellung muss insbesondere im Gesamtkontext diskutiert werden, da eine Parallelisierung von Teilsimulationen offensichtlich eine Vielzahl an Simulationsressourcen bindet.

5.4. Grundlegendes zur technischen Gesamtumsetzung

Im Rahmen der Umsetzung ist ein modulares System entstanden, das über eine webbasierte Schnittstelle vom Nutzer bedient werden kann. Hierbei hat der Nutzer die Möglichkeit, jegliche Informationen über seinen Maschinenpark inklusive Ausfällen, verfügbare Spannmittel und Werkstücke, Urlaubs- und Schichtpläne etc. zu hinterlegen. Diese Informationen werden für jeden Nutzer in einer zentralisierten Datenbank abgelegt. Darüber hinaus ist der Nutzer in der Lage, einen Fertigungsauftrag anzulegen, womit eine gewisse Schrittkette losgetreten wird. Dieser Fertigungsauftrag enthält Informationen über die verwendete Maschine, das NC-Programme, Werkzeuge, Spannmittel und Rohteil.

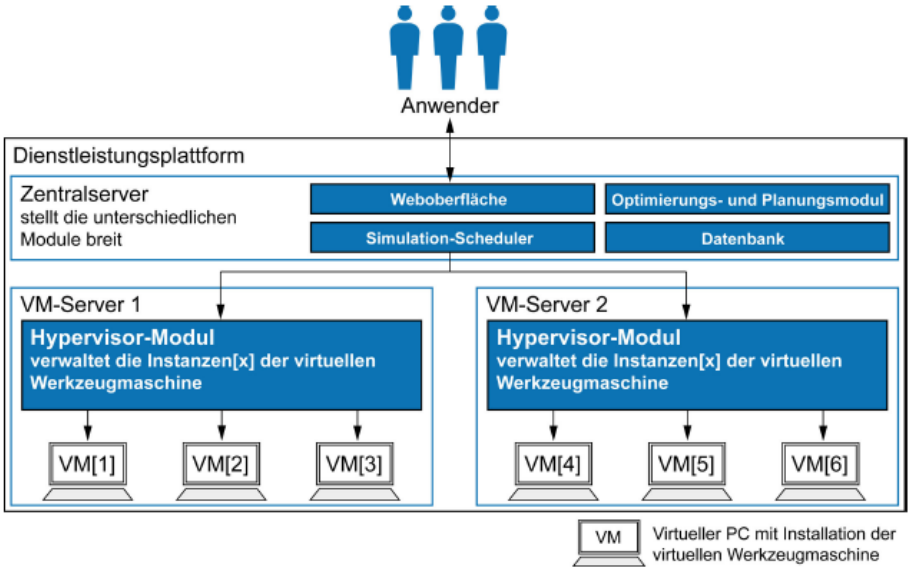

Abb. 48 Schnittstelle zwischen Dienstleistungsplattform und virtualisierten Rechnern [69]

Nachdem ein solcher Auftrag angelegt ist, wird dieser unmittelbar an den „Simulation Scheduler“ übergeben, der verteilte Rechnerknoten verwaltet, auf denen die Software zur Materialabtragsimulation installiert ist. Jeder Rechnerknoten basiert auf einem Hypervisor, der wiederum eine gewisse Anzahl virtualisierter Rechner kapselt. Jeder dieser virtualisierten Rechner entspricht einer Instanz der virtuellen Werkzeugmaschine (Abb. 48). Eine VMDE-Datei, die nach Anlegen des Auftrags erstellt worden ist, wird,

sobald ein Rechnerknoten zur Verfügung steht, per HTTP-Post Request an die entsprechende Instanz übermittelt. Daraufhin kann diese Instanz die Simulation für den Auftrag durchführen und übermittelt dem System nach Vollendigung den Report inklusive der Information, ob der Auftrag kollisionsfrei durchgeführt worden ist. Parallel zur Simulation des konkreten Auftrags werden im Hintergrund die verschiedenen Optimierungsebenen angestoßen. Das Ontologiesystem erzeugt potentielle Aufträge auf alternativen Maschinen, die dann ebenfalls zum Simulation Scheduler geschickt werden. Gleichzeitig verfolgt das Modul zur Optimierung der Maschineneinrichtung das Ziel, verbesserte Aufspannsituationen zu identifizieren und generiert ebenfalls Simulationsaufträge, die dem Scheduler übermittelt werden. Dieses Modul wird als „Setup Optimizer“ bezeichnet. Jegliche Kommunikation zwischen den Komponenten des Gesamtsystems findet über eine zentralisierte SQL-Datenbank statt. So fragen alle Module aktiv nach, ob neue Aufträge existieren oder ob einzelne Optimierungsschleifen zum Erfolg geführt haben. Falls etwa der Setup Optimizer für einen bestimmten Auftrag meldet, dass es eine verbesserte Aufspannlage gibt, kann das Modul für die intelligente Arbeitsplanung diese Information aus der entsprechenden Datenbanktabelle entnehmen. Dieses Modul nennt sich „Production Optimizer“.

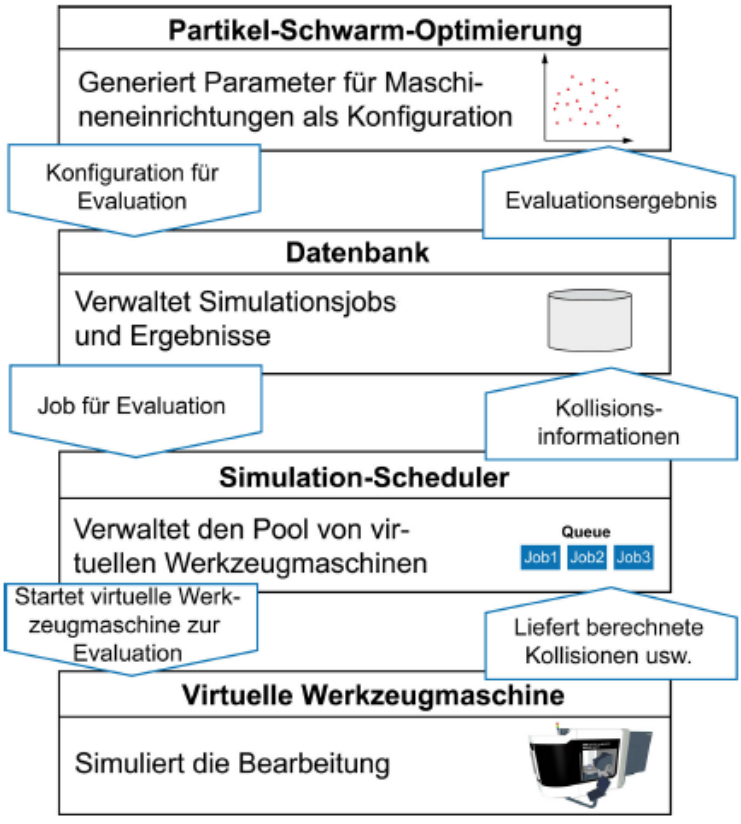

Abb. 49 Schnittstelle zwischen Setup Optimizer und Simulation Scheduler [69]

Abb. 49 zeigt diesen Informationsaustausch exemplarisch für Setup Optimizer und Simulation Scheduler. Es findet kein direkter Austausch zwischen den Modulen statt, sondern lediglich Aktualisierungen in der Datenbank. Dieses Vorgehen ist im Wesentlichen der Natur des Projekts geschuldet, da sämtliche Optimierungsebenen durch verschiedene Arbeitsgruppen realisiert worden sind.

6. Fazit und Ausblick

Es hat sich gezeigt, dass auf Basis der bekannten Verfahren simulationsgestützter Optimierung ein Workflow entwickelt werden konnte, der die beschriebenen Problemstellungen löst, die einerseits die heterogene Rechnerstruktur und der damit einhergehenden Notwendigkeit, Simulationen asynchron zu berechnen, mit sich bringt, und andererseits die Anzahl der benötigten Simulationsläufe reduzieren kann. Darüber hinaus konnte festgestellt werden, dass das Optimierungspotenzial bei kleiner Losgröße zu vernachlässigen ist, wohingegen bei Serienfertigungen, die über einen langen Zeitraum laufen, Ersparnisse möglich sind. Sämtliche Betrachtungen, die bis zu diesem Zeitpunkt angestellt wurden, beziehen sich direkt auf die gegebene Problematik der Simulationen von Dreh- und Fräsmaschinen bzw. auf theoretischen Betrachtungen. Aus diesem Grund soll der Ausblick nicht auf allgemeiner Ebene erfolgen, sondern für eine Problemstellung aus der statistischen Physik eine Lösung skizzieren, die auf den Erkenntnissen, die im Laufe des Projekts entstanden sind, basiert. Auf diese Art und Weise kann bewiesen werden, dass ein Verfahren entstanden ist, das die Optimierungen durch Simulationen stark beschleunigt, falls die Möglichkeit einer effizienten Approximation besteht. In dem Zuge wird ein weiterer Ansatz vorgestellt, der im Zusammenhang mit dem Projekt zur Einschränkung notwendiger Simulationen im Zyklus der simulationsgestützten Optimierung entwickelt, allerdings aufgrund der Natur des Optimierungsproblems nicht weiter verfolgt wurde. Hierbei geht es darum, das Wissen vergangener Simulationen auszunutzen, um den Suchraum durch invertierbare Dimensionsreduktionsverfahren einzuschränken.

6.1. Optimization@home – Ein Framework zum Curvefitting auf Basis von Monte Carlo Simulationen in Gridumgebungen

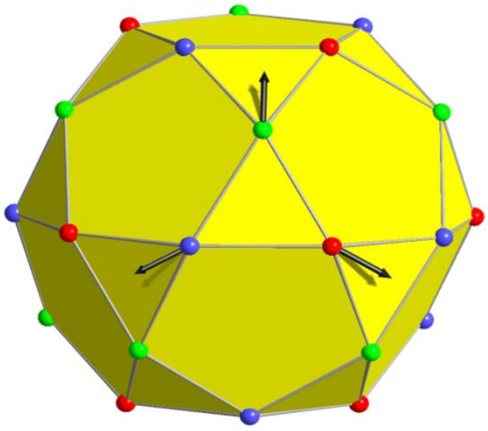

Abb. 50 Nächster-Nachbar Austauschmodell für frustrierte magnetische Moleküle

In der Forschung zum molekularen Magnetismus wird eine Fragestellung sehr intensiv verfolgt. Gegeben seien Beobachtungen aus experimentellen Untersuchungen. Wie lässt sich dieses Ergebnis durch Simulationen reproduzieren [70]?

Dazu sei auf Abb. 50 verwiesen. Hierbei wird die Topologie eines magnetischen Moleküls repräsentiert als eine Menge von Spins im Raum, wobei die Kanten zwischen den Spins als gewichtete Kanten betrachtet werden müssen, deren Wert Austauschwechselwirkungen zwischen zwei Spins beschreibt.

Für dieses spezielle Modell ist der sogenannte Hamiltonoperator definiert als:

$$\widetilde{H} = \sum_{<i,j>} J_{i,j}\ \overrightarrow{S_i}\,\overrightarrow{S_j} + g\mu_b\vec{H}\ \cdot \sum_i \overrightarrow{S_i}$$

Dabei beschreibt $J \in \mathbb{R}^{i\times j}$ die Matrix der Wechselwirkungsparameter, $\overrightarrow{S_i} \in \mathbb{R}^3$ die Orientierung eines Spins im Raum und, während g, μ_b und $\vec{H}$ Konstanten sind.

Im Labor können unterschiedliche Eigenschaften solcher Strukturen ermittelt werden, wobei bestimmte Randbedingungen, wie Temperaturen oder magnetische Felder verändert werden. Ein Wert, der in dem Zusammenhang von Interesse sein kann, ist die sogenannte magnetische Suszeptibiltät, die in der Literatur üblicherweise mit χ angegeben wird. Es handelt sich um eine Größe, die als Maß der Magnetisierbarkeit eines Festkörpers verstanden werden kann [71].

Die Spins neigen dazu, sich so auszurichten, dass das energetische Minimum erreicht wird. Da der Hamiltonoperator die Energie des Systems repräsentiert, muss also herausgefunden werden, unter welcher Ausrichtung der Wert für $\widetilde{H}$ minimal ist. Sei also J festgelegt, finde für alle i diejenigen Vektoren S_i, für die $\widetilde{H}$ den geringsten Wert annimmt. Es konnte bereits früh gezeigt werden, dass sich diese Fragestellung durch Monte Carlo Simulationen lösen lässt, im speziellen Fall durch den „Metropolis Algorithmus“ [72] [73] .

Metropolis Algorithmus		
Input: $J_{i,j} \in \mathbb{R}^3$, Konstanten $g, \mu_b, \vec{H}, k, T$, Anzahl Iterationen ξ, $\rho = 0$		
Initialisierung: Wähle $\overrightarrow{S_i}$ zufällig, $\forall i$, berechne $\widetilde{H} = \sum_{<i,j>} J_{i,j}\ \overrightarrow{S_i}\,\overrightarrow{S_j} + g\mu_b\vec{H}\ \cdot \sum_i \overrightarrow{S_i}$		
While$(\rho < \xi)$		
	Wähle beliebigen Spin und führe eine Flipoperation aus.	
	$\widetilde{H_,} = \Sigma_{<i,j>}\overrightarrow{S_i}\,\overrightarrow{S_j} + g\mu_b\vec{H}\ \cdot \Sigma_i\overrightarrow{S_i}$	
	$\rho = \rho + 1$	
	If $(\widetilde{H_,} < \widetilde{H})$	
		Akzeptiere neue Konfiguration
	Else	
		Akzeptiere neue Konfiguration mit Wahrscheinlichkeit $p = \exp\left(-\left(\widetilde{H_,} - \widetilde{H}/_{kT}\right)\right)$
	Endif	
Endwhile		

Algorithmus 4 Pseudocode Metropolis Algorithmus

Je nach Komplexität der Struktur kann eine sehr hohe Anzahl benötigter Iterationen dazu führen, dass eine einzelne Simulation extrem zeitintensiv ist. Es kann allerdings bewiesen werden, dass eine parallele Berechnung des Algorithmus existiert, die extrem gut skaliert. Im Folgenden wird diese Art und Weise der Parallelisierung beschrieben:

Metropolis Algorithmus – parallele Berechnung				
Input: $J_{i,j} \in \mathbb{R}^3$, Konstanten $g, \mu_b, \vec{H}, k, T$, Anzahl Iterationen ξ, $\rho[\varphi] = 0[\varphi]$				
Initialisierung: Wähle $\overrightarrow{S_\iota}[\varphi]$ zufällig, $\forall i, \phi \in \{0, \dots, \varphi - 1\}$, berechne $\widetilde{H} = \sum_{<i,j>} J_{i,j}\ \overrightarrow{S_\iota}\overrightarrow{S_J} + g\mu_b\vec{H} \cdot \sum_i \overrightarrow{S_\iota}$				
Prozess π $[\phi \in \{0, \dots, \varphi - 1\}]$				
	While ($\rho_\phi < {}^{\xi}/_{\varphi}$)			
	Wähle beliebigen Spin und führe eine Flipoperation aus			
		$\widetilde{H'} = \Sigma_{<i,j>}\overrightarrow{S_\iota}\overrightarrow{S_J} + g\mu_b\vec{H} \cdot \Sigma_i\overrightarrow{S_\iota}$		
		$\rho_\phi = \rho_\phi + 1$		
		If $(\widetilde{H'} < \widetilde{H})$		
		Akzeptiere neue Konfiguration		
			Else	
		Akzeptiere neue Konfiguration mit Wahrscheinlichkeit $p = \exp\left(-\left({}^{\widetilde{H'} - \widetilde{H}}/_{kT}\right)\right)$		
			Endif	
		Endwhile		
	Barriere $\phi \in \{0, \dots, \varphi - 1\}$			
	If $(\phi = 0)$			
		$\overrightarrow{S_{\mathbb{r}}} = \mathbb{r}(\vec{S}, \mu)$		
	Endif			
Endprozess				

Algorithmus 5 Parallele Berechnung des Metropolisalgorithmus

Es ist zu erkennen, dass außerhalb der Initialisierung und einer einzigen Aggregationsoperation über alle Prozesse kein sequentieller Anteil vorliegt. Demnach liegt offensichtlich ein Problem vor, bei dem sich problemlos eine lineare Skalierung über eine nahezu beliebige Anzahl an genutzten Prozessen erzielen lässt. Allerdings wird diese etwas beschränkt, da je nach Struktur eine Mindestanzahl an Iteration zum „Aufwärmen“ der Spinorientierung vorausgesetzt wird. So ist es beispielsweise nicht denkbar, eine Struktur, für deren Berechnung sequentiell 10000 Iterationen notwendig sind, auf 10000 Prozesse mit jeweils einer Iteration zu verteilen. Die lineare Skalierung wird auf diese Art und Weise also nach einer Zeit eine Sättigung erfahren.

Nun stellt sich die Frage, inwiefern der Metropolis Algorithmus innerhalb der beschriebenen Anwendung interessant für die Thematik der simulationsgestützten Optimierung sein kann. Dazu sei angenommen, dass im Labor unter verschiedenen Bedingungen, also unter Annahme verschiedener Temperaturen und verschiedener magnetischer Feldstärken, Werte für die magnetische Suszeptibilität ermittelt wurden. Sei also T_n eine Menge betrachteter Temperaturen und $\vec{H}_m$ eine Menge betrachteter magnetischer Feldstärken. Dann ist $\mathbb{S}_{i,j}, i \leq n, j \leq m$ die Suszeptibilität, die unter $t_i \in T_n$ und $\vec{h}_j \in \overrightarrow{H_n}$ experimentell ermittelt wurde. In Algorithmus 4 ist erkennbar, dass Temperatur und magnetisches Feld als Konstanten angegeben sind. Nichtsdestotrotz

sind diese offensichtlich so adjustierbar, dass sich die experimentellen Bedingungen in der Simulation nachbilden lassen. Sei also $\mathcal{S}_{i,j}$ die Suszeptibilität, die als Ergebnis einer Monte Carlo Simulation unter t_i und $\overrightarrow{h_j}$ hergeleitet wurden. Dann ist das Ziel die Matrix der Austauschwechselwirkungen J so zu wählen, dass:

$$\sum_{<i,j>} \left(\mathbb{S}_{i,j} - \mathcal{S}_{i,j}\right)^2 \rightarrow min$$

Die Austauschwechselwirkungen sollen demnach so gewählt werden, dass das experimentelle Ergebnis mittels Simulation bestmöglich angenähert werden kann. Bei genauerer Betrachtung lässt sich erahnen, dass eine Annäherung dieser Problematik mit „klassischer“ simulationsgestützter Optimierung mit großen Herausforderungen verbunden ist. So müssen für den kompletten Abgleich zwischen Simulation und Experiment insgesamt $m * n$ Simulationen für einen einzelnen Satz an ausgewählten Wechselwirkungsparametern berechnet werden, wobei jede einzelne Simulation grundsätzlich für sich selbst genommen schon eine spürbare Komplexität aufweisen kann. Hinzu kommt, dass der Suchraum für Strukturen mit vielen disjunkten Wechselwirkungen nahezu beliebig groß werden kann, wodurch die Anzahl der notwendigen Simulationen zusätzlich massiv anwächst. Das Verfahren, das für die Optimierung der Aufspannsituation in Werkzeugmaschinen entwickelt wurde, besteht aus den beiden Phasen Optimierung auf Basis approximativer Berechnungen und Verifikation auf Basis vollständiger, komplexer Simulationen. Es stellt sich die Frage, wie dieser Ansatz in dem beschriebenen Zusammenhang eine sinnvolle Anwendung finden kann.

Dazu wird zunächst ein Fokus auf die Möglichkeiten der effizienten Approximierung in diesem Zusammenhang gelegt. Tatsächlich ergibt sich eine Reihe von Möglichkeiten, verhältnismäßig effizient Approximationen für ausgewählte Parametersets zu erzeugen. Dazu sollte zunächst ein genauerer Blick auf die hier gewählte Fitnessfunktion geworfen werden. Es wird für jede Kombination aus Temperatur und magnetischer Feldstärke ein Wert für die Suszeptibilität berechnet. Eine relativ wenig invasive Möglichkeit, eine Art notwendiges Kriterium zu erzeugen, besteht darin, sich zunächst auf eine fixe Temperatur oder ein fixe magnetische Feldstärke zu konzentrieren und nur über den jeweils anderen Freiheitsgrad zu iterieren. Dann erhält man folgende Fitnessfunktion:

$$f(J) = \begin{cases} \sum_i \left(\mathbb{S}_{i,k} - \mathcal{S}_{i,k}^J\right)^2 mit\ k\ konstant, falls\ m \leq n \\ \sum_j \left(\mathbb{S}_{k,j} - \mathcal{S}_{k,j}^J\right)^2 mit\ k\ konstant, falls\ n > m \end{cases}$$

Durch Einsatz dieser Fitnessfunktion werden demnach um den Faktor $\max(m,n)$ weniger Simulationen benötigt. Offensichtlich wird dadurch Genauigkeit eingebüßt, allerdings ist nicht davon auszugehen, dass ein Parameterset, das unter den

Bedingungen schlechte Ergebnisse erzeugt, über den kompletten Betrachtungsraum eine passende Konfiguration darstellt.

Offensichtlich lässt sich diese Idee noch deutlich weitertreiben, indem die „Stichprobe“ für die betrachteten Temperaturen noch deutlich weiter reduziert wird. Eine Beobachtung, die man nach einigen Versuchen manueller Anpassung zwischen Simulation und Experiment häufig macht, ist, dass bei Steigung der Temperaturen der Einfluss der Auswahl der Wechselwirkungsparameter extrem nachlässt bzw. komplett verfällt. Das legt nahe, zunächst nur jene Simulationen zu betrachten, die unter Annahme falscher Wechselwirkungen für hohe Abweichungen beim Abgleich zum Experiment verantwortlich sind. Insofern entsteht folgende Hypothese: Für die Optimierungsphase des vorgestellten Algorithmus ist es hinreichend, für jede Auswahl an Wechselwirkungsparametern lediglich eine (niedrige) Temperatur und eine magnetische Feldstärke zu berücksichtigen, womit folgende Fitnessfunktion entsteht:

$$f(J) = (\mathbb{S}_{t,h} - \mathcal{S}_{t,h}^{J})^2, t \in T_n, h \in \overrightarrow{H_m}$$

Auf diese Art und Weise können weniger aussichtsreiche Kandidaten frühzeitig als solche gekennzeichnet werden und diejenigen Kandidaten, die sich potentiell als Lösungen anbieten, innerhalb der Verifikationsphase des Algorithmus genauer betrachtet werden. Die Anzahl der notwendigen Simulationen für die erste Phase wäre somit lediglich von der verwendeten Metaheuristik und der Größe des Suchraums abhängig.

Die Anzahl der Iterationen für eine atomare Simulation wird üblicherweise vom Anwender anhand seiner eigenen Erfahrung mit der Thematik ausgewählt. Da in der Literatur wenig über die Theorie des Erwartungswerts für die Konvergenz von Monte Carlo Simulationen zu finden ist, wird dieser Wert meistens großzügig eingeschätzt, um das Ergebnis der Simulation nicht zu gefährden. Offensichtlich ergibt sich hieraus ein sehr interessanter Ansatzpunkt für die Approximation von Monte Carlo Simulationen. So könnte es sich als durchaus sinnvoll erweisen, den Wert für ξ niedrig zu wählen und somit lokale Optima in einzelnen Simulationen in Kauf zu nehmen. Da insbesondere schwarmbasierte Metaheuristiken mehrere Kandidaten betrachten, die räumlich recht nah zueinander liegen, ist hierbei die Hypothese, dass dieser Effekt für die schnelle Sammlung von Simulationsergebnissen nicht ausschlaggebend ist. Hierbei ergibt sich ein interessantes Forschungsfeld, da beliebig mit diesem Wert experimentiert werden kann.

Für sämtliche Ausprägungen statistischer Simulationen, zu denen auch die Monte Carlo Methoden zählen, ergibt sich eine weitere spannende Forschungsfrage durch die Inkaufnahme numerischer Fehler innerhalb der Berechnungen. So stellt sich die Frage, ob eine Beschränkung auf einer 32- oder sogar 16-Bit Repräsentation der Fließkommazahlen das Verfahren signifikant beschleunigen kann, ohne hierbei eklatant an Genauigkeit zu verlieren. Dies ist insbesondere interessant, wenn die Beschleunigung

mit besonderer Berücksichtigung auf die verwendete Hardware erreicht werden soll (etwa mit GPGPU oder FPGAs).

Bislang wurde also gezeigt, wie im Kontext von Monte Carlo Simulationen zur Ermittlung magnetischer Suszeptibilität eine Fitnessfunktion $f: \mathbb{R}^n \to \mathbb{R}$ aussehen kann, die Grundlage für die erste Phase des zweistufigen Algorithmus ist. Möchte man den Algorithmus eins zu eins auf die vorgestellte Problemstellung übertragen, muss eine Abbildung $g: \mathbb{R}^n \to \{0,1\}$ gefunden werden, die einen ermittelten Kandidaten aus der ersten Phase verifiziert. Das kann relativ einfach durch Einführung einer Akzeptanzgrenze erreicht werden. Sei also $\gamma \in \mathbb{R}$ beliebig. Weiterhin seien $T_n, \overrightarrow{H_m}$ Mengen der abzugleichenden Temperaturen und äußeren Feldstärken. Dann lässt sich g definieren als:

$$g(J) = \begin{cases} 0, falls \sum_{<i,j>} \left(\mathbb{S}_{i,j} - \mathcal{S}_{i,j}\right)^2 > \gamma \\ 1, falls \sum_{<i,j>} \left(\mathbb{S}_{i,j} - \mathcal{S}_{i,j}\right)^2 \leq \gamma \end{cases}$$

So wird also als Parameter angegeben, welche Abweichung vom Experiment maximal in Kauf genommen wird. In der einfachsten denkbaren Variante entsteht also ein Verfahren, das einem aus dem ermittelten Kandidatenset die erste Wechselwirkungskonfiguration zurückgibt, die unter den angegebenen Grenzwert fällt.

Durch diese Definition der beiden notwendigen Funktionen für den vorgestellten Optimierungsalgorithmus kann dieser also auch im Rahmen der automatisierten Suche von Wechselwirkungsparametern genutzt werden. Der konkrete Vorschlag wurde in der Überschrift dieses Kapitels als „Optimization@home“ bezeichnet. Die Idee hierbei ist, für die verteilte Durchführung einzelner Simulationen auf sogenanntes „Public Ressource Computing“ zu setzen. Dieser Begriff bezeichnet den Ansatz, verfügbare Rechnerressourcen im Heimbereich, für hochkomplexe Berechnungen zur Verfügung zu stellen und die dabei entstehenden Teilergebnisse zusammenzutragen. Hierbei gibt es einige prominente Beispiele wie „SETI@Home“, bei dem es um die gezielte Suche nach außerirdischem Leben geht [74] oder „Folding@Home“, das das Ziel verfolgt, aus einer gegebenen Aminosäuresequenz die Faltung des entstehenden Proteins vorherzusagen [75]. Für die Idee, ein solches Projekt mit dem Ziel der automatisierten Suche nach realistischen Wechselwirkungskonstanten umzusetzen, kann auf vorhandenen Vorarbeiten aufgesetzt werden, die im Rahmen des Forschungsprojekts „Spinhenge@Home“ entstanden sind [76].

6.2. Einschränkung des Suchraums durch Dimensionsreduktion

Vergleicht man die beiden betrachteten Optimierungsszenarien, die in dieser Arbeit beleuchtet werden, fällt auf, dass es sich um zwei vollkommen verschiedene Szenarien handelt. Während bei der Optimierung der Aufspannsituation in virtuellen Werkzeugmaschinen der Suchraum recht niedrigdimensional ist und ein sehr starkes Performanzgefälle zwischen Approximierung und Simulation besteht, liegen bei der Annäherung von Experiment und Simulation durch Monte Carlo Simulationen sehr heterogene Suchräume vor, die beliebig komplex werden können. Zudem handelt es sich bei Monte Carlo Simulationen um eine klassische HPC-Anwendung, die linear skalierbar ist, während die Parallelisierung von Materialabtragssimulationen ein deutlich komplexeres Feld ist, deren Profit kaum im Vorfeld vorhersehbar ist. Insbesondere während der Betrachtung der Anwendungsfälle mit hochdimensionalen Suchräumen ist eine weitere Idee entstanden, die zum Abschluss dieser Arbeit als Konzept erläutert werden soll. Da der Fokus im Gesamtkontext eindeutig auf der Optimierung der Materialabtragssimulation liegt, ist der Ansatz nicht weiter verfolgt worden. Wie bereits im Vorfeld geschildert, spielt die Größe des Suchraums eine entscheidende Rolle für die Konvergenz verschiedener Metaheuristiken. Aus diesem Grund ist der Ansatz entstanden, auf Basis bekannter Simulationsergebnisse, die entweder im Vorfeld aus einer Wissensdatenbank entnommen werden, falls diese für den Anwendungsfall vorhanden ist, oder im Laufe der Optimierung gesammelt werden, ein Modell für eine Dimensionsreduktion zu lernen. Auf diese Weise kann in einem eingeschränkten Suchraum nach einer optimalen Lösung gesucht werden. Für das Prozedere sind nicht sämtliche Dimensionsreduktionsverfahren geeignet, da zum Teil keine inversen Abbildungen in den ursprünglichen Raum möglich sind. Insofern eignen sich insbesondere Hauptkomponentenanalysen, die Kern der folgenden Betrachtungen sein werden. Eine Umsetzung des Ansatzes ist grundsätzlich auch über neuronale Netzwerke mittels Autoencodern zu realisieren. Nichtlineare Verfahren wie t-SNE oder multidimensionale Skalierung eignen sich hingegen nicht. Zur Beschreibung des Verfahrens sei zunächst die grundsätzliche Funktionsweise der Hauptkomponentenanalyse (PCA) beschrieben. Hierbei handelt es sich um ein Verfahren aus der multivariaten Statistik, das bereits 1903 von Karl Pearson vorgestellt wurde [77]. Trotz zahlreicher Forschungsarbeiten, die sich im Themenspektrum der Dimensionsreduktion bewegen, ist die PCA aufgrund ihrer analytischen Interpretierbarkeit nach wie vor weit verbreitet, was sich in zahlreichen Publikationen zeigt, die sich damit beschäftigen, wie die Berechnung der PCA effizienter gemacht werden kann [78] [79]. Die PCA basiert auf der Annahme, dass diejenigen Dimensionen, die eine relativ hohe Varianz aufweisen, den Großteil der Information eines vorliegenden Datensatzes enthalten.

Sei $X := \{\vec{x}^{\alpha}\}_{\alpha=1\dots N}, x \in \mathbb{R}^d, N \in \mathbb{N}$ ein d -dimensionaler Datensatz mit N Beobachtungen. Der Datensatz sei hierbei zentriert, also:

$$\frac{1}{N}\sum_{\alpha=1}^{N}\vec{x}^{\alpha} = 0$$

Zunächst wird die geschätzte Varianz im Unterraum der Richtung $\hat{\varrho}$ betrachtet.

$$F(\hat{\varrho}) = \frac{1}{N}\sum_{\alpha=1}^{N}(\vec{x}^{\alpha\tau}\,\hat{\varrho})^2$$

$$= \frac{1}{N}\sum_{\alpha=1}^{N}\hat{\varrho}^{\tau}\vec{x}^{\alpha}\vec{x}^{\alpha\tau}\hat{\varrho}$$

$$= \hat{\varrho}^{\tau}\left[\frac{1}{N}\sum_{\alpha=1}^{N}\vec{x}^{\alpha}\vec{x}^{\alpha\tau}\right]\hat{\varrho}$$

Da der Ausdruck $\frac{1}{N}\sum_{\alpha=1}^{N}\vec{x}^{\alpha}\vec{x}^{\alpha\tau}$ – hier hervorgehoben durch eckige Klammern – genau der Definition der Kovarianzmatrix C entspricht, erhält man schlussendlich die Gleichung:

$$F(\hat{\varrho}) = \hat{\varrho}^{\tau}C\hat{\varrho}$$

Die Kovarianzmatrix ist positiv semidefinit und symmetrisch. Demnach existiert eine Zerlegung $C = UDU^{\tau}$, mit $U = [\vec{u}_1, \dots, \vec{u}_d]$, die genau den Eigenvektoren von C entsprechen und $D = diag(\lambda_1, \dots, \lambda_d)$, wobei λ_i der Eigenwert zum Eigenvektor $\vec{u}_i$ ist. Gesucht ist die Projektion, die die Varianz maximiert, also:

$$\sigma_{max}^2 = max_{\vec{\varrho}}\,F(\vec{\varrho}) = \max\frac{\vec{\varrho}^{\tau}C\vec{\varrho}}{\vec{\varrho}^{\tau}\vec{\varrho}}$$

Gemäß der Standardmethodik aus der Analysis kann das Maximum durch die Berechnung der Nullstellen der partiellen Ableitungen ermittelt werden, also durch Lösung der Gleichung $\nabla_{\vec{\varrho}}F = 0$. Hierbei erhält man den Ausdruck $C\vec{\varrho} = \frac{\vec{\varrho}^{\tau}C\vec{\varrho}}{\vec{\varrho}^{\tau}\vec{\varrho}}\,\vec{\varrho}$. Da der Faktor $\frac{\vec{\varrho}^{\tau}C\vec{\varrho}}{\vec{\varrho}^{\tau}\vec{\varrho}}$ in dem Fall skalar ist, kann die Gleichung geschrieben werden als $C\vec{\varrho} = \lambda\vec{\varrho}$, was genau der Eigenwertbedingung entspricht. Folglich wird die Varianz von F maximiert durch den Eigenvektor von C zum größten Eigenwert.

Im Rahmen dieser Überlegungen sind erste Auswertungen durchgeführt und publiziert worden. Dabei sind zur Evaluation die Fitnessfunktionen $f_{Rosenbrock}$ und $f_{Rastrigin}$ zum Einsatz gekommen. Nun wurden jeweils drei Fälle miteinander verglichen. Während der Durchführung der Partikelschwarmoptimierung wurden sämtliche Eingangsvektoren

persistiert, die im Zuge der Suche evaluiert worden sind. Daraufhin wurden nach jeweils 50, 150 und 250 Iterationen diejenigen Vektoren gefiltert, die zu den 500 besten Fitnesswerten geführt haben.

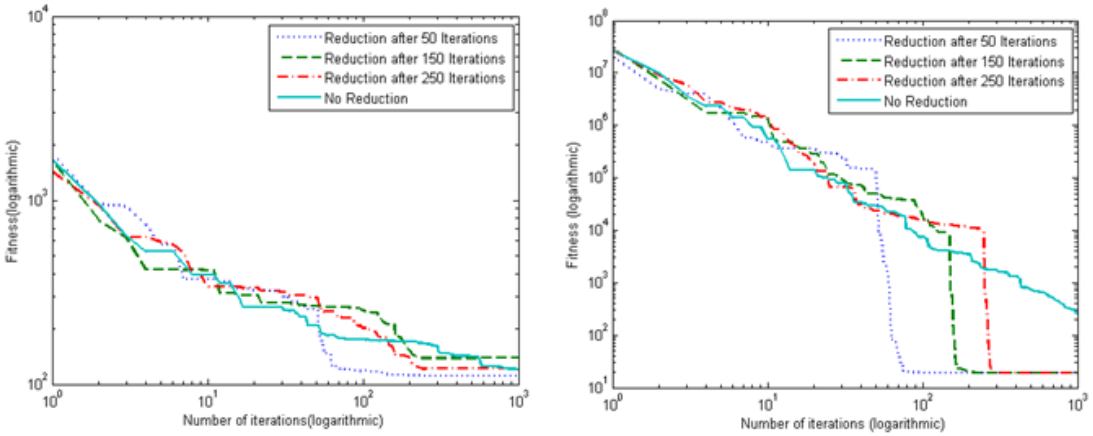

Abb. 51 Konvergenzverhalten der Benchmarkfunktionen mit und ohne Dimensionsreduktion

Die Darstellung des jeweiligen Konvergenzverhaltens für diese Prozedur ist auf Abb. 51 zu sehen. Dabei sind deutlich zwei Effekte zu beobachten. Direkt im Anschluss an die Durchführung der Dimensionsreduktion stellt sich jeweils eine deutliche Beschleunigung der Konvergenz ein. Ab diesem Zeitpunkt ist nur noch ein Minimum an Iterationen bis zum Erreichen eines Plateaus notwendig. Besonders auffällig ist allerdings ein Effekt, der im Plot auf der linken Seite der Abbildung zu sehen ist. Es scheint die große Gefahr zu bestehen, in ein lokales Optimum zu konvergieren. Nichtsdestotrotz braucht die Durchführung der Partikelschwarmoptimierung im Anschluss noch eine Reihe große Reihe an Iterationen, um eine Konfiguration zu finden, die zu einem ähnlich guten Fitnesswert führt. Um also in schnellerer Zeit zu einem passablen Ergebnis zu kommen, lohnt es sich demnach, eine Dimensionsreduktion in die Optimierungsschleife zu integrieren.

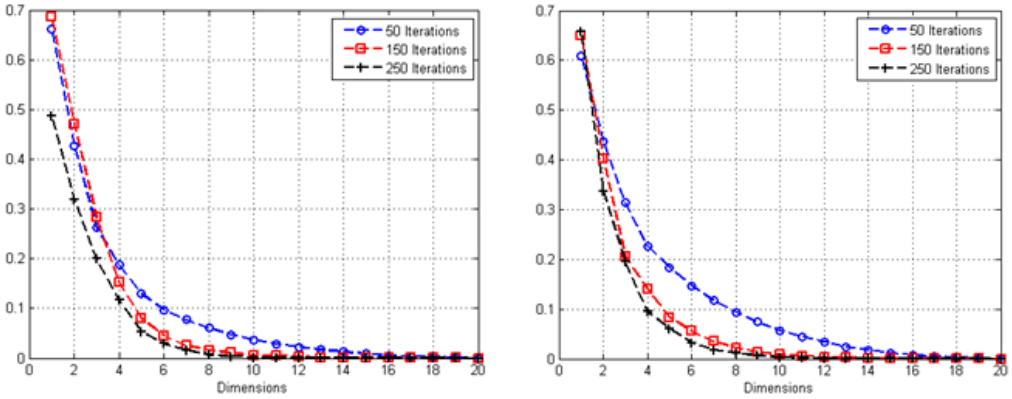

Abb. 52 Informationsverlust durch PCA nach Betrachtung verschiedener Iterationen

Abb. 52 zeigt eine Abbildung des berechneten Informationsverlusts bezüglich der jeweiligen Zieldimensionen. Das Verhalten der jeweiligen Kurve entspricht in etwa den Erwartungen. Je mehr Daten gesammelt wurden, desto klarer zeigt sich ein niedrigdimensionales Muster in den hochdimensionalen Daten, das durch eine PCA mit vergleichsweise niedrigem Informationsverlust abgebildet werden kann. Auch die asymptotische Konvergenz gegen null ist zu erwarten. Man sieht, dass unabhängig von der Anzahl der Dimensionen und der Funktion, bei der Reduktion auf 14 Dimensionen nahezu kein Informationsverlust mehr hinzunehmen ist. Hierbei ist wichtig zu erwähnen, dass sich dieser Informationsverlust lediglich auf die bis dahin gesammelten Daten bezieht. Da die Exploration im Suchraum zu den gemessenen Zeitpunkten noch im Gange war, kann die Dimensionsreduktion dazu führen, dass bestimmte Punkte im Suchraum nicht mehr erreicht werden.

Was hier präsentiert wurde, ist lediglich ein Konzept mit ersten Auswertungen, das noch tiefergehende Überlegungen erfordert. Grundsätzlich deutet sich an, dass eine Dimensionsreduktion zu einer sinnvollen Einschränkung des Suchraums führen kann, die für einige Anwendungen, etwa bei der Betrachtung einer vierstelligen Anzahl an Dimensionen, absolut notwendig für eine performante Durchführung einer simulationsgestützten Optimierung ist. Weitere Untersuchungen könnten beispielsweise untersuchen, ob solche Dimensionsreduktionen sich als Zwischenschritte eignen, um Startpunkte für weitere Iterationen zu erzeugen. Dieses Vorgehen würde den Effekt ausnutzen, dass eine schnelle Konvergenz nach der Dimensionsreduktion stattfindet, aber trotzdem das Steckenbleiben in lokalen Optima vermeiden.

6.3. Resümee

Nachdem das Fazit bislang über konkrete Fragestellungen motiviert worden ist, kann nun in aller Kürze ein abschließendes Resümee gezogen werden. Die vorliegende Arbeit zeigt, dass im Rahmen des Forschungsprojekts „Intelligente Arbeitsvorbereitung auf Basis virtueller Werkzeugmaschinen“ ein Verfahren zur zweistufigen simulationsgestützten Optimierung entstanden ist, das einerseits die projektspezifischen Anforderungen erfüllt, und andererseits das Potential hat, bezüglich verschiedenartiger Domänen in heterogenen Rechnerumgebungen zum Erfolg zu führen, falls die Möglichkeit einer weniger zeitintensiven Approximation von Simulationen möglich ist. Der konkrete Ausblick auf die automatisierte Suche nach Wechselwirkungskonstanten mittels Monte-Carlo Simulationen zeigt, dass zu erwarten ist, dass der Erfolg der vorgestellten Methodik sich auch bei klassischen High Performance Computing Applikationen einstellen wird. Bezogen auf das Optimierungsziel der Maschineneinrichtung zwecks Effizienzsteigerung kann abschließend konstatiert werden, dass vor allem die Reduzierung von Nebenzeiten bei Serienfertigungen möglich gemacht wird. Optimierungen bis zur Losgröße 1, wie sie im Rahmen diverser Industrie-4.0 Illustrationen häufig beworben werden, sind durch

eine optimierte Aufspannlage nicht sinnvoll zu erreichen. Eine weitere zentrale Erkenntnis des Gesamtprojekts ist, dass eine Parallelisierung atomarer Simulationen im Kontext von simulationsgestützter Optimierung virtueller Werkzeugmaschinen wenig sinnvoll ist, da die Skalierung auf Basis der Optimierung selbst, also die Skalierung über mehrere Partikel einer Generation, zu einer deutlich höheren und robusteren Beschleunigung führt. Nichtsdestotrotz ergeben sich durch die konkrete Umsetzung der theoretischen Überlegungen Forschungsfragen, die in näherer Zukunft näher beleuchtet werden sollten.

7. Literaturverzeichnis

[1] D. Ross, Computer-Aided Design: A Statement of Objectives., M.I.T. Electronic Systems Laboratory, 1960.

[2] O. Zienkiewicz, R. Taylor und J. Zhu, The Finite Element Method: Its Basis and Fundamentals, Butterworth-Heinemann, 2013.

[3] J. Jasperneite, „Computer & Automation, Was hinter Begriffen wie Industrie 4.0 steckt," Fraunhofer IOSB, 19 Dezember 2012. [Online]. Available: https://www.computer-automation.de/steuerungsebene/steuern-regeln/artikel/93559/0/. [Zugriff am 22 April 2019].

[4] „"BIBB : Industrie 4.0 und die Folgen für Arbeitsmarkt und Wirtschaft"," August 2015. [Online]. Available: http://doku.iab.de/forschungsbericht/2015/fb0815.pdf. [Zugriff am 22 April 2019].

[5] L. Chong und C. Osorio, „A Simulation-Based Optimization Algorithm for Dynamic Large-Urban Transportation Problems," *Transportation Science, vol. 52,* pp. 497-737, Juli 2017.

[6] A. Nguyen, S. Reiter und P. Rigo, „A review on simulation-based optimization methods applied to building performance analysis," *Applied Energy, vol. 113,* pp. 1043-1058, 2014.

[7] H. Jalali und I. Van Nieuwenhuyse, „Simulation optimization in inventory replenishment: A Classification," *IIE Transactions, vol. 47:11,* pp. 1217-1235, 2015.

[8] D. He, L. H. Lee, C.-H. Chen, M. Fu und S. Wasserkrug, „Simulation Optimization Using the Cross-Entropy Method with Optimal Computing Budget Allocation," *ACM Transactions on Modeling and Computer Simulation, vol. 20,* Januar 2010.

[9] W. Dangelmaier und J. Gausemeier, Intelligente Arbeitsvorbereitung auf Basis virtueller Werkzeugmaschinen, Springer Vieweg, 2019.

[10] R. Neugebauer, Werkzeugmaschinen: Aufbau, Funktion und Anwendung von spanenden und abtragenden Werkzeugmaschinen, Springer, 2012.

[11] H. B. Kief und H. A. Roschiwal, CNC-Handbuch 2013/14, Carl Hanser Verlag GmbH & Co. KG, 2013.

[12] „Wikipedia - Computerized Numerical Control - Steuern und Regeln," [Online]. Available: https://de.wikipedia.org/wiki/Computerized_Numerical_Control. [Zugriff am 20 Dezember 2018].

[13] J. Anderson und E. E., „Converting Geometry Between BRL-CAD and Other Formats," in

BRL-CAD Tutorial Series, Aberdeen Proving Ground: United States Army Research Laboratory., 2004.

[14] L. Butler, E. Edwards und D. Kregel, BRL-CAD Tutorial Series: Volume III – Principles of Effective Modeling, Aberdeen Proving Ground: United States Army Research Laboratory, 2003.

[15] A. Köhler, „Real Time Simulation and Visualization of NC Milling Processes for Inhomogenous Materials on Low-End Graphics Hardware," in *Proceedings. Computer Graphics International*, Hannover, 1998.

[16] I. T. Chappel, „The use of vectors to simulate material removed by numerically controlled milling," *Computer Aided Design,* Bd. 15, Nr. 3, pp. 156-158, 1983.

[17] C. Chen, J. Hong und L. Yan, „Research on Spherical Voxel-Based Machining Simulation," in *Computer Modelling and New Technologies, Vol. 17, No.4*, Riga, Lettland, Transport and Telecommunication Institute, 2013, pp. 112-120.

[18] S. Ratchev, S. Nikov und I. Moualek, „Material removal simulation of peripheral milling of thin wall low-rigidity structures using FEA," in *Advances in Engineering Software*, Elsevier, 2004, pp. 481-491.

[19] L. März, W. Krug, O. Rose und G. Weigert, Simulation und Optimierung in Produktion und Logistik: Praxisorientierter Leitfaden mit Fallbeispielen., Heidelberg, Dordrecht, London, New York: Springer, 2011.

[20] C. Laroque und P. J.P., „An Automatic Approach for Parameter Optimization of Material Flow Simulation Models based on Particle Swarm Optimization.," in *4. International Conference on Advances in System Simulation (Tagungsband)*, Lissabon, 2012.

[21] C. Laroque, B. Urban und E. M., „Parameteroptimierung von Materialflusssimulationen durch Partikelschwarmalgorithmen.," in *Tagungsband der Multikonferenz Wirtschaftsinformatik.*, Göttingen, 2010.

[22] R. Eberhart und J. Kennedy, „A New Optimizer Using Particle Swarm Theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995.

[23] R. Eberhart und Y. Shi, „ A modified particle swarm optimizer," in *Proceedings of IEEE International Conference on Evolutionary Computation*, Anchorage, AK, USA, 1998.

[24] M. R. Bonyadi und Z. Michalewicz, „Particle swarm optimization for single objective continuous space problems: a review," in *Evolutionary Computation, 25(1)*, MIT Press, 2017, pp. 1-54.

[25] M. Donelli, R. Azaro, F. De Natale und A. Massa, „An innovative computational approach

based on a particle swarm strategy for adaptive phased-arrays control.," in *IEEE Transactions on Antennas and Propagation, 54*, IEEE, 2006, pp. 888-898.

[26] R. Azaro, M. Donelli, F. De Natale, G. Franceschini, D. Franceschini, A. Massa und S. Piffer, „Frontiers in multiple-agents evolutionary techniques applied to adaptive arrays design," in *Proceedings of Antennas and Propagation Society International Symposium*, 2005.

[27] D. Gies und Y. Rahmat-Samii, „Reconfigurable array design using parallel particle swarm optimization," in *Antennas and Propagation Society International Symposium*, 2003.

[28] B. Yeo und L. Yilong, „Adaptive array digital beamforming using complex-coded particle swarm optimization-genetic algorithm," in *Microwave Conference Proceedings*, 2003.

[29] R. X. und D. Wunsch, „Gene regulatory networks inference with recurrent neural network models," in *IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, 2005.

[30] H. Ressom, Y. Zhang, X. J., Y. Wang und C. R., „Inference of gene regulatory from time course gene expression data using neural networks and swarm intelligence," in *CIBCB '06. 2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, 2006.

[31] Y. Liu und H. Yokota, „Modeling transcriptional regulation in chondrogenesis using particle swarm optimization," in *CIBCB '05. Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2005.

[32] S. Doong, „Protein homology modeling with heuristic search for sequence alignment," in *HICSS 2007. 40th Annual Hawaii International Conference on System Sciences*, 2007.

[33] G. Di Caro, F. Ducatelle und L. Gambardella, „Swarm intelligence for routing in mobile ad hoc networks," in *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, 2005.

[34] J. Wang, X. Wang und H. Min, „A hybrid intelligent qos multicast routing algorithm in ngi," in *PDCAT 2005. Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2005.

[35] B. Tatomir und L. Rothkrantz, „Hierarchical routing in traffic using swarm-intelligence," in *Intelligent Transportation Systems, 2006. Proceedings. 2006 IEEE*, 2006.

[36] R. Poli, „An Analysis of Publications on Particle Swarm Optimisation Applications," in *Journal of Artificial Evolution and Applications*, 2008.

[37] S. Kirkpatrick, C. Gelatt und M. Vecchi, „Optimization by simulated annealing," *Science, vol. 220,* pp. 671-680, 1983.

[38] B. Jurke, Interviewee, *Effiziente G-Code Interpretation.* [Interview]. 2013-2016.

[39] S. Aktiengesellschaft, „Programmierhandbuch 840D sl/828D Grundlagen," Siemens AG, [Online]. Available: https://cache.industry.siemens.com/dl/files/635/28705635/att_75718/v1/26SP1_840D sl_828D_Programmierhandbuch_Grundlagen_de-DE.pdf. [Zugriff am 22. Februar 2019].

[40] „Wikipedia Kreissehne," [Online]. Available: https://de.wikipedia.org/wiki/Sehne_(Geometrie)#/media/File:Sehne.png. [Zugriff am 13. Juli 2018].

[41] G. Amdahl, „Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *AFIPS Conference Proceedings (30),* pp. 483-485, 1967.

[42] J. Gustafson, „Reevaluating Amdahl's Law," *Communications of the ACM,* pp. 532-533, 1988.

[43] J. Gustafson und R. Benner, „Development and analysis of scientific application programs on a 1024-processor hypercube," *SIAM Journal on Scientific and Statistical Computing,* pp. 609-638, 1988.

[44] I. Scriven, D. Ireland, A. Lewis, S. Mostaghim und J. Branke, „Asynchronous Multiple Objective Particle Swarm Optimization in Unreliable Distributed Environments," *Proceedings of the CEC 2008,* pp. 2481-2486, 2008.

[45] B. Koh, A. George, R. Haftka und B. Fregly, „Parallel asynchronous particle swarm optimization," *International Journal for numerical methods in engineering (67),* pp. 578-595, 2006.

[46] G. Venter und J. Sobieszczanski-Sobieski, „Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations," *Journal of Aerospace Computing, Information, and Communication, vol. 3,* pp. 123-137, 2006.

[47] J. Rada-Vilela, M. Zhang und W. Seah, „Performance Study on Synchronous and Asynchronous Updates in Particle Swarm Optimization," *Proceedings of GECCO 2011, ACM Press,* pp. 21-28, 2011.

[48] J. Rada-Vilela, M. Zhang und W. Seah, „Random Asynchronous PSO," *Proceedings of ICARA 2011, IEEE Press,* pp. 220-225, 2011.

[49] W. Hsieh und S. Horng, „Feature Selection Based on Asynchronous Discrete Particle Swarm Optimal Search Algorithm," *Proceedings of PAAP 2012, IEEE Press,* pp. 262-268, 2012.

[50] R. Reisch, J. Weber, C. Schröder und C. Laroque, „Asynchronous optimization techniques for distributed computing applications," *Simulation Series,* Nr. 47, pp. 49-56, 2015.

[51] D. Dauwe, P. S., A. Maciejewski und H. Siegel, „An Analysis of Resilience Techniques for Exascale Computing Platforms," *IEEE International Parallel and Distributed Processing Symposium Workshops,* pp. 914-923, 2017.

[52] F. Capello, „Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities," *International Journal of HPC Applications, vol. 23, no.3,* pp. 212-226, 2009.

[53] A. Griewank, „Generalized Decent for Global Optimization," *Journal of Optimization Theory and Applications, vol.34,* pp. 11-39, Mai 1981.

[54] D. Aloise, A. Deshpande, P. Hansen und P. Popat, „NP-hardness of Euclidean sum-of-squares clustering," *Machine Learning,* Bd. 2, Nr. 75, pp. 245-249, 2009.

[55] M. Mahajan, P. Nimbhorkar und K. Varadarajan, „The Planar k-Means Problem is NP-Hard," *Lecture Notes in Computer Science,* Nr. 5431, pp. 274-285, 2009.

[56] S. Lloyd, „Least squares quantization in PCM," *IEEE Transactions on Information Theory,* pp. 129-137, 1982.

[57] A. Mueß, J. Weber, R. Reisch und B. Jurke, „Implementation and Comparison of Cluster-Based PSO Extensions in Hybrid Settings with Efficient Approximation," *Niggemann O., Beyerer J. (eds) Machine Learning for Cyber Physical Systems. Technologien für die intelligente Automation (Technologies for Intelligent Automation),* 20 Februar 2016.

[58] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Rosenbrock_function#/media/File:Rosenbrock_function.svg. [Zugriff am 7 März 2019].

[59] C. Cortes und V. Vapnik, „Supprt-vector networks," *Machine Learning,* Bd. 20, Nr. 3, pp. 273-297, 1995.

[60] A. Ben-Hur, D. Horn, H. Siegelmann und V. N. and Vapnik, „Support vector clustering," *ournal of Machine Learning Research,* pp. 125-137, 2001.

[61] K. Crammer und Y. Singer, „On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines," *Journal of Machine Learning Research,* pp. 265-292, 2015.

[62] F. Leisch und E. Dimitriadou, „mlbench: Machine Learning Benchmark Problems," 10 Juli 2012. [Online]. Available: https://CRAN.R-project.org/package=mlbench. [Zugriff am 22 November 2018].

[63] J. Friedman, „Multivariate Adaptive Regression Splines," *The Annals of Statistics,* pp. 1-67, 1991.

[64] „Machineworks Unternehmenswebseite," MachineWorks Limited, [Online]. Available:

https://www.machineworks.com/. [Zugriff am 20. Dezember 2018].

[65] Web3D Consortium, „About Web3d Consortium," [Online]. Available: http://www.web3d.org/about. [Zugriff am 20. Dezember 2018].

[66] B. Schulz und U. Schnell, „Indien setzt auf eigene Werzeugmaschinen," MaschinenMarkt, 22. September 2011. [Online]. Available: https://www.maschinenmarkt.vogel.de/indien-setzt-auf-eigene-werkzeugmaschinen-a-332221/. [Zugriff am 14. Oktober 2018].

[67] R. Reith, „Indien braucht Werkzeug und Maschinen," WELT, 9. September 2013. [Online]. Available: https://www.welt.de/print/die_welt/wirtschaft/article119825373/Indien-braucht-Werkzeug-und-Maschinen.html. [Zugriff am 14. Oktober 2018].

[68] S. Itasse, „Deutscher Werkzeugmaschinenbau erobert Indien," MaschinenMarkt, 6. Main 2016. [Online]. Available: https://www.maschinenmarkt.vogel.de/deutscher-werkzeugmaschinenbau-erobert-indien-a-527029/. [Zugriff am 14. Oktober 2018].

[69] G. Rehage, F. Isenberg, R. Reisch, J. Weber, B. Jurke und P. Pruschek, „Intelligent Arbeitsvorbereitung in der Cloud," in *wt Werkstattstechnik online*, 2016, pp. 77-82.

[70] D. Landau und K. Binder, A Guide to Monte Carlo Simulations in Statistical Physics, Cambridge University Press, 2005.

[71] H. Schmidt, C. Schröder, E. Hägele und M. Luban, „Dynamics and thermodynamics of a pair of interacting magnetic dipoles," *Journal of Physics A: Mathematical and Theoretical ,* Bd. 48, Nr. 18, 2015.

[72] N. Metropolis, A. Rosenbluth, M. Rosenbluth, E. Teller und A. Teller, „Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics,* Bd. 21, pp. 1087-1092, 1953.

[73] W. Hasting, „Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika,* Bd. 57, pp. 97-109, 1970.

[74] „SETI@Home," [Online]. Available: https://setiathome.berkeley.edu/. [Zugriff am 5. Oktober 2018].

[75] „Folding@Home," [Online]. Available: https://foldingathome.org/. [Zugriff am 5. Oktober 2018].

[76] C. Schröder und T. Hilbig, „Spinhenge@Home," [Online]. Available: https://web.archive.org/web/20060716151930/http://spin.fh-bielefeld.de/. [Zugriff am 10. Dezember 2018].

[77] K. Pearson, „ On lines and planes of closest fit to a system of points in space,“ *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. Series 6, 2,* pp. 559-572, 1901.

[78] S. Roweis, „EM Algorithms for PCA and SPCA,“ in *Advances in Neural Information Processing Systems. Ed. Michael I. Jordan, Michael J. Kearns, and Sara A. Solla*, MIT Press, 1998.

[79] E. Candes, X. Li, Y. Ma und J. Wright, „Robust Principal Component Analysis?,“ *Journal of the ACM,* Bd. 3, Nr. 58, 2011.

[80] M. Kaever, Steuerungsintegrierte Fertigungsprozessüberwachung bei spanender Bearbeitung, Aachen: RWTH Aachen, 2004.

[81] J. Yi, J. You und H. Xiaojian, „A particle swarm based network hosts clustering algorithm for peer-to-peer networks,“ in *International Conference on Computational Intelligence and Security*, 2006.

[82] A. Caneshi, G. D., C. Sangregorio, R. Sessoli, L. Sorace, A. Cornia, M. Novak, C. Paulsen und W. Wernsdorfer, „The molecular approach to nanoscale magnetism,“ *Journal of Magnetism and Magnetic Materials,* Nr. 200, pp. 182-201, 1999.

Anhang

Übersicht der verwendeten NC-Befehle

G-Befehle

Bei G-Befehlen handelt es sich üblicherweise um Vorgaben, die die zu verfahrende Bahn betreffen.

Befehl	**Bedeutung**	**Nomenklatur**
G0	Bewegung zu einem definierten Zielwert im Eilgang	G0 X… Y… Z… X := Zielwert in X-Richtung Y := Zielwert in Y-Richtung Z := Zielwert in Z-Richtung
G1	Bewegung zu einem definierten Zielwert über eine lineare Bahn	G1 X… Y… Z… F… X := Zielwert in X-Richtung Y := Zielwert in Y-Richtung Z := Zielwert in Z-Richtung F := Vorschubgeschwindigkeit
G2	Bewegung zu einem definierten Zielwert über eine Kreisbahn im Uhrzeigersinn Verschiedene Nomenklaturen möglich	G2 X… Y… Z… I… J… X := Zielwert in X-Richtung Y := Zielwert in Y-Richtung Z := Zielwert in Z-Richtung I := Differenz der aktuellen Position zum Kreismittelpunkt in X-Richtung J := Differenz der aktuellen Position zum Kreismittelpunkt in Y-Richtung
		G2 X… Y… Z… I=AC(…) J=AC(…) X := Zielwert in X-Richtung Y := Zielwert in Y-Richtung Z := Zielwert in Z-Richtung I := Absolutwert X-Koordinate des Kreismittelpunkts J := Absolutwert Y-Koordinate des Kreismittelpunkts
		G2 X… Y… Z… CR… X := Zielwert in X-Richtung Y := Zielwert in Y-Richtung Z := Zielwert in Z-Richtung CR:= Radius der Kreisbahn

		G2 AR… I… J… AR := Größe des Öffnungswinkels I := Absolutwert X-Koordinate des Kreismittelpunkts J := Absolutwert Y-Koordinate des Kreismittelpunkts
		G2 AR… X… Y… Z… AR := Größe des Öffnungswinkels X := Zielwert in X-Richtung Y := Zielwert in Y-Richtung Z := Zielwert in Z-Richtung
G3	Bewegung zu einem definierten Zielwert über eine Kreisbahn gegen den Uhrzeigersinn	Analog zu G2
G4	Programmpause	G4 F… F := Dauer der Programmpause in Sekunden

Sonstige Befehle

Befehl	**Bedeutung**	**Nomenklatur**
T	Werkzeugauswahl	T… T := Ausgewähltes Werkzeug (über Werkzeugnummer/T-Nummer)
M6	Werkzeugwechsel	M6 Das aktuell ausgewählte Werkzeug wird eingesetzt
M30	Programmende	M30